Firmware Manual

EX-Technik par Excellence

TERMEX 750 Terminal

Firmware Version VS1.09
Document Version: 1.09a
(Oct2000)

EX TEC Oesterle GmbH
Entwicklung und Produktion von
EX-Anzeige- und Bediensystemen
EX-Hard- und Software
EX-Komponenten

Schorndorfer Straße 55
D-73730 Esslingen
Telefon 07 11 / 31 54 55-0
Telefax 07 11 / 31 54 55-29
e-mail info@extec.de

http://www.extec.de

1 Table of Contents

2 Important Notes

This documentation describes the behavior of the firmware for the TERMEX 7xx terminal. It is intended for the designer of an application project as well as for the operator of the machine system.
Please refer to the following documentation to install and connect the terminal:

"Technical Manual TERMEX 7xx" [TERMEX]

References to this documentation are labeled [TERMEX].

Please refer to the following documentation to generate project designs with the TERMEX PRO design program:

"Technical Manual TERMEX PRO" [TERMEXPRO]

References to this documentation are labeled [TERMEXPRO].

Please refer to the following documentation to program the terminals using EPCA:

"The EPCA Programming System" [EPCA]

This manual describes the following topics:
- Setup options in the setup menu and dialog
- Protocols and commands of the serial interface(s)
- Display options
- Programming options

The chapter called **"For the User"** is intended for the user of the terminal and for the person responsible for setting it up. It provides basic information about operation and the functions of the terminal.
The chapter called **"For the Project Designer"** is intended for those persons who are responsible for integrating the terminal into its system environment. It includes, for example, information about setting up the user interface and addressing various protocols.

If you come across any terms in the documentation which are not absolutely clear, please consult the **Glossary** on page 99 or the **Index** on page 101.

For questions beyond the scope of this documentation please contact the following e-mail address:

support@extec.de

You can call up the latest information about the terminals on the support pages of our web site:

http://www.extec.de

as well as in our regular electronic **EX TEC Newsletter**. Take a look at our web site to find out how to receive it.

You can update the terminal firmware to the latest version at any time. This version is permanently available for downloading from the support page of our web site. Please refer to page 54 for further information about how to do so.

3 Introduction

Characteristics

- Graphics terminal, display with 640 x 480 pixels
- 256 colors
- Display of text, graphics, messages, variables, bar graphs and trend plotters
- 6 character sets in various sizes already installed, additional sets can be loaded
- Projects consisting of screens, messages and programs (EPCA programming system) can be permanently stored in the flash memory of the terminal
- 50 keys
- Integrated mouse with two buttons
- Integrated setup menu for configuration, accessible directly on the terminal via the keyboard
- Setup dialog via a serial interface using the terminal program on the PC
- Control interface for connecting a control computer (PLC, PC)
- Combined control/printer interface
- Up to three serial peripheral interfaces for connecting scales for hazardous areas as well as barcode scanners
- Optional digital I/O connections
- Firmware updates via serial interface
- Integrated load BIOS for controlling firmware updates

Typical applications:

Interface to a programmable logic controller (PLC)

- A PLC is connected to the SER1 control interface of the terminal via a separator with a data interface (ENT-DC). The two units communicate using a special protocol, depending on the type of the PLC, and exchange data via a shared data block.
- Screens, messages and programs are generated with the TERMEX PRO design software. They are then downloaded as a project from the design PC to the terminal (refer to **[TERMEXPRO]**). The project is stored residently in a flash memory in the terminal.
- Screens, for example, can be retrieved from the PLC via predefined data words within a data block. Messages can then be displayed via individual bits. The variables within the screens are used to indicate changing values (actual variables). They are supplied with these actual values via the data words in the data block that have been previously defined in TERMEX PRO.
- Values input on the terminal are set variables whose values are transferred to the PLC via predefined data words in the data block.
- The states of the function keys (actuated, not actuated) are inserted by the terminal into the data block as status bits.

Interface to a PC

- A PC or a similar type of controller is connected to the SER1 control interface of the terminal via a separator with a data interface (ENT-DC). The two units communicate using the EX TEC protocol. The PC sends the codes for the characters that must be output to the terminal directly, together with various control commands (ESC sequences). The terminal then sends the codes for the actuated keys back to the PC, along with responses to control commands.

- Screens can be generated with the TERMEX PRO design software and then downloaded to the terminal. In this case the transmission effort for building the display is reduced and more complex displays are possible.

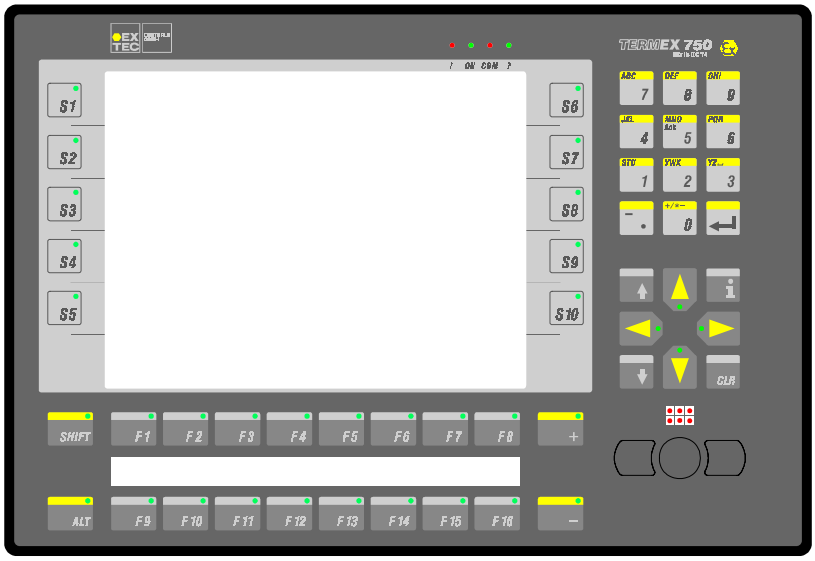
Extended interface to a PC

- The EX TEC protocol also allows access to the data block. In contrast with PLC protocols, the PC plays the role of communication master.
- Hence, a mixture of the two operating modes described above is possible, allowing a far greater range of options.
- The set variables can be used for inputs, for example, while at the same time the full command spectrum of the EX TEC protocol is available.

4 For the User

4.1 Key Assignment

The Key Codes in the EX TEC Protocol can be found on page 74.



F1...F16	*Function keys*: Their function is determined solely by the application; if the ⇑ key is pressed at the same time, F11...F20 are accessible.	F1...F16
S1...S10	*Special keys:* Additional keys that can be assigned as needed.	SO1... SO10
SHIFT	*Shift key:* Used to switch over keys with a dual function. Throughout the documentation "<⇑> key" is used as an abbreviation for shortcuts which require the Shift key to be pressed.	SHT
ALT	*Alt key:* Used to switch over keys with additional functions (with a pushbutton function/toggle, indicated by the LED on the key). The Alt key is used to switch to **alphanumeric inputs** for the various set variables and input windows. The characters ABC, DEF, etc. on the numeric keys are then valid instead. If you press one of these keys, the display toggles between the three characters. Release the key as soon as the desired character appears. You can exit the alphanumeric mode by pressing Alt again. The mode is exited automatically if you confirm your input with ↵ or if you switch to a different set variable.	ALT
0...9	*Number keys*: Used to enter values for set variables, for input windows or to output key codes.	0...9
●	*Period / comma:* Used to enter numeric values for set variables, for input windows or to output key codes.	.
CLR	*Clear key:* Used to delete the last character entered for set variables, for input windows or to output key codes.	CLR
↵	*ENTER key*: Used to terminate an entry for set variables, for input windows or to output key codes.	CR
ACK	*Message acknowledgment key* (<⇑> 5): Used to acknowledge a queued message.	
+ -	*Increment / decrement keys:* Used to increment / decrement the value of the selected set variable by one (note the possible scale factor for each variable!) The LEDs on the keys go out if the upper / lower limit of the variable's value range is reached.	PLS, MNS
▲ ▼	*Message shift keys:* Used to scroll through several messages that are waiting to be displayed.	TOP, BOT

	*Set variable shift keys:* Used to select the desired set variable. The system jumps between the set variables according to their geometric arrangement. The four cursor LEDs show the possible directions.	CUP, CDN, CLE, CRI
	*Info key:* Can be used to output additional information in applications (via EPCA programs).	INF

4.2 Status LEDs

The TERMEX 750 terminal has four status LEDs to the left of the display:

!	At least one message is active
ON	The terminal power is switched on and the firmware is running
COM	Communication error
?	(Not used)

4.3 Startup Messages

A startup message, extending over several lines, is displayed for a few seconds whenever the TERMEX 750 is switched on or after a reset. This time has been set to 4 seconds in the factory, but you can alter it if you wish with the *gnrl_StartupMessageTime* setup option.

```
S/N: 123456789
```

This is the serial number of the unit that has been fixed in the factory. It enables the unit to be identified uniquely. The serial number is also specified on the rating plate.

```
Firmware Version: VS1.07
Firmware Date: 29.02.2000  09:59
```

This is the version of the firmware that is currently running on the terminal. You can update the firmware yourself.

```
BIOS Version: 0091
```

The BIOS is the part of the integrated software which controls how the firmware is loaded and starts the unit. You cannot update the BIOS yourself. The BIOS version remains the same throughout the entire life of the unit.

```
Testing System RAM...Passed.
```

The RAM is being tested. If the word *Passed* appears, then the test was successful. If the message *Error at...* is displayed, an error has been found. In this case, please contact your nearest sales office. The terminal can no longer be guaranteed to function correctly, even if it does not at first sight appear to be behaving any differently.

```
Testing Project Memory...Passed.
```

The project memory is being tested. This test checks that the project structure is correct and verifies the checksums. If the word *Passed* appears, then the test was successful. If *No Project* appears, there are no projects in the memory. *Checksum Error* means that at least one checksum error has been detected, while *Frame Error* refers to an error in the project structure. The latter two error types may have occurred because the system crashed while the project was being loaded. If you attempt to load the project again, but the error message persists, even though there is nothing to indicate that the system has crashed, the cause may be an error in the project memory. In this case, please contact your nearest sales office.
The message *Block Length Error* may refer either to a memory error or to an error in the project.

```
Testing Setup Memory...Passed.
```

The stored setup data is being tested. If the word *Passed* appears, then the test was successful. *Checksum Error* means that a checksum error has been detected. In this case, the terminal attempts to interpret the error-free parts of the setup data. Errored entries are reset to the factory defaults. *Initialized* indicates that no setup data is stored and that the factory defaults will be used instead. If either of these two messages are displayed, you should attempt to save the setup data again (see below). If this does not help, please contact your nearest sales office.

```
Testing Event Memory...Passed (10 Events).
```

The resident event memory is being tested. The event memory contains the events of messages that have occurred and that can be output again in message lists. The message *Passed (xx Events)* indicates that events have been stored and states how many. *Initialized* means that no correct events have been found. A firmware update might be the reason for this. Events are normally deleted after an update. If this error message is displayed repeatedly, the cause could be a drop in the battery voltage. In this case, please contact your nearest sales office.

```
Testing Variable Archive...Passed.
```

The resident variable archives memory is being tested. The variable archives contain the values of variables in the course of time. The message *Passed* indicates that values have been stored. *Initialized* means that no correct values have been found. A firmware update might be the reason for this. Values are normally deleted after an update. As long as there no values in the archives, the message *Initialized* comes again and again.

```
< Press F1 to enter Setup ! >
< Press F2 to enter Download-Mode ! >
```

When these two lines are displayed, you can press F1 to open the integrated setup menu (see below) or F2 to change to download mode. The terminal is switched automatically to the EXTEC16 protocol in this mode and set to a special, configurable baud rate (refer to page 76). This saves you having to change the protocols in the setup and reset them again afterwards. In addition, you can set a higher baud rate for downloading than for the application.

4.4 How to Enter Settings in the Integrated Setup Menu

The terminal has a setup menu which makes it very easy to select the appropriate settings.

The following steps are necessary to open the integrated setup menu:

Press the shortcut

<SHIFT⇑> <ENTER ↵> 9

to reset the terminal. A startup message then appears on the display. It remains there for four seconds as default.

With the startup message displayed, press:

<SHIFT⇑> F1
or F1

This opens the main setup menu.

<SHIFT⇑> F1	Opens the setup menu which is permanently integrated in the firmware.
F1	Opens a special setup menu, if one is loaded. If a special setup menu is not loaded, the integrated setup menu is loaded instead.

The integrated setup menu is largely self-explanatory. A list of the possible settings and their effects can be found in the Appendix.

Important points about the menu:

- There are two types of page - menu selection pages (blue fields) and settings pages (yellow fields).
- You can select a submenu on the menu selection pages and then jump to it.
- On the settings pages you can navigate with the variable cursor and enter values (using the number keys as well as the increment and decrement keys). The variable cursor can be controlled using either the cursor keys or the left mouse button.
- There is a help key on every page, which you can press to display additional information.
- You can also display help for a selected setting by pressing the right mouse button.
- Some of the settings pages are multiple pages. You can scroll between these with the "Back" and "Forward“ function keys.
- When you quit the setup menu, you must decide whether or not you want to save the active settings. If you do not save your settings here, they will be lost the next time you start up the terminal. They will remain valid for the duration of your current session (e.g. for test purposes), but the previous settings will be restored as soon as you restart.

4.5 How to Enter Settings in the Setup Dialog

In addition to entering settings directly on the unit in the integrated setup menu, you can also configure the TERMEX 7xx remotely via the SER1 port. The terminal is normally connected for this purpose by means of a serial cable on the SK-LWL to a PC on which a terminal program (such as HyperTerminal in Windows 95/98) has been started. You can then display and change the settings in the dialog of this terminal program.

Procedure

1. The TERMEX 7xx must be set to the EXTEC16 protocol beforehand. In addition, please check that the settings for the transmission parameters (baud rate, parity) are the same at both ends.
2. In order to open the dialog mode, you must send the following command to the terminal on the PC: ESC '1' (the two characters 27 and 49). Most terminal programs allow you to type the ESC character using the Alt key. Hold this key pressed down while you enter the number 27 on the PC's number block. When you release the Alt key, the character corresponding to the entered code is sent off. You can generate the '1' directly by pressing the '1' key.

The TERMEX 7xx responds by outputting the message below:

```
TERMEX 750 Setup Dialog - type 'help' for command list
> □
```

You can then enter commands with parameters at the prompt > and confirm them by pressing the ↵ key. The terminal responds to each command either with a confirmation or with an error message. You can enter the "help" command to display a list of the available commands:

```
// Available commands:
//   set <item=setting>
//   showsettings
//   showsetting <item>
//   showpossibles
//   showpossible <item>
//   showlevel
//   enterlevel <level> <password>
//   reset
//   help
//   exit  (w/o save)
//   quit  (with save)
> □
```

Meanings of the commands:

set — Activate a setting

The identifier for the setting and the setting itself are transferred. The setting may take the form either of a numerical value that is within a valid range or of a predefined text.

```
e.g. set ser1BaudRate=9600              (fixed setting text)
e.g. set gnrl_KeyAutorepeatDelay=450    (numerical value)
```

showsettings — Show all settings

A list of all the settings is output in the following format:
set <item>=<setting>
You can use this command, for instance, to save all the settings you have entered on the PC. Terminal programs normally allow you to save these outputs in a file (in HyperTerminal: Transfer/Capture Text). You can use the stored file later on to save precisely these settings back to a terminal (HyperTerminal: Transfer/Send Text File).

showsetting — Show one (or more) specified settings

You can display any setting by specifying its identifier. If you only enter part of the identifier, such as "ser1" or "gnrl", all settings that begin with this character string will be displayed.

```
e.g. showsetting ser1
```

showpossibles — Show all possible settings

You can display a list of all possible settings, for example in order to see the valid number ranges or the default setting texts.

```
e.g.
ser1BaudRate[Baud]:300,1200,2400,4800,9600,19200,38400,57600,115200
```

showpossible — Show one (or more) specified possible settings

You can display any possible setting by specifying its identifier. If you only enter part of the identifier, such as "ser1" or "gnrl", all possible settings that begin with this character string will be displayed.

```
e.g. showpossible gnrl
```

showlevel — Show the current access level and the next access code

The currently active level is displayed.
The next access code is displayed in addition for levels 9 and 10 (see below)

enterlevel — Enter the password or the access code for a specific access level

You must specify both the desired level as a number and the password for it.

```
e.g. enterlevel 3 pass3
```

A special code derived from the access code is required for levels 9 and 10 (refer to showlevel). Not all users are allowed to access levels 9 and 10.

reset — Reset to factory settings

You can use this command to restore the factory settings. You should always use it with caution, because your custom settings may be lost. The factory settings are not saved until you enter the "quit" command. If you close the setup dialog with "exit", in other words, your old settings will still be valid the next time you start up the unit.
The reset command only applies to those settings which you are actually allowed to access on the current level.

help — Show command list

A list of all the available commands is displayed (see above).

exit — Close dialog without saving

The dialog box is closed, but your new settings are not saved in the flash memory. If you are unsure about the changes you have made, you can quit the dialog in this way without losing anything.
The new settings remain valid, however, until you restart the terminal and the system loads your old settings from the flash memory again.

quit — Close dialog and save changes
Your new settings are saved in the flash memory. The old settings are now lost and cannot be restored!

You can find a list and short description of the possible settings in the reference section on page 76.

Access levels

Both read and write access to the settings in the setup dialog are controlled by means of access levels:

- If you have sufficient access authorization to write, any changes you make to the settings (e.g. with "set") will be accepted. If not, you will be denied access and an error message will be output instead.
- If you have sufficient access authorization to read, the settings themselves, their values and the possible settings will be displayed. If not, you will be unable to see this information.
- When you first open the dialog, you have the authorization defined with the *gnrl_SetupEntryLevel* setting. The factory setting for *gnrl_SetupEntryLevel* is 2. You can change to a higher level by entering the level number together with its password.
- The levels and the corresponding settings are listed in the Appendix.

- You can control access to levels 2 to 8 by allocating suitable passwords. The machine operator can be authorized for level 2, for instance, while the person responsible for setting up the terminal could be given level 4 and the project designer level 8.
- The passwords themselves are also settings, in other words you can change them with "set pass_level2=meinpasswort", for instance. Each level includes access both to its own password and to the passwords of all lower levels.

The following passwords are active when the terminal is first delivered:

Level	Password
2	pass2
3	pass3
4	pass4
5	pass5
6	pass6
7	pass7
8	pass8

- We advise you to change these passwords before you install the unit. You should make a note of the new passwords immediately and keep it in a safe place, to ensure that you are never in a situation where you cannot access the setup.

- Not all users are allowed to access levels 9 and 10. Special access codes which are not normally publicized are required for these two levels.

4.6 Runtime Error Messages

Runtime errors occur mainly in connection with EPCA programs. You will be forced to stop working because all these errors are fatal. You can press any key to reset the terminal.
These error messages cannot be suppressed. Please also refer to Possible Settings on page 76.

Error message	Description
>>> RUNTIME ERROR <<< **Illegal Opcode**	An illegal (internal) EPCA operation code has occurred. Possible reasons: • Error in the loaded project. Try either reloading the project or initializing it from the flash memory. • Incompatibility between the EPCA development system and the firmware: a project program uses code that is not yet supported by the firmware. Check the version management settings under Project Properties in TERMEX PRO.
>>> RUNTIME ERROR <<< **Too many tasks.**	You have attempted to start more than the maximum allowed number of EPCA tasks. If the number of tasks entered in the task list for a particular screen is higher than the maximum number (refer to "EPCA/OS Menu"), you will have to delete some of them. This error should normally be detected by the latest version of TERMEX PRO as soon as you load a project.
>>> RUNTIME ERROR <<< **Too much memory allocated.**	You have attempted to request more than the available EPCA memory. You can try reducing the amount of required memory either by getting rid of variables or by defining smaller arrays in the EPCA tasks. This error should normally be detected by the latest version of TERMEX PRO as soon as you load a project.
>>> RUNTIME ERROR <<< **Unknown program.**	You have attempted to call an EPCA program which is not available in the currently loaded project. This problem cannot be solved by users; please contact EX TEC.
>>> RUNTIME ERROR <<< **Function stack depth exceeded.**	You have attempted to exceed the maximum EPCA function call depth. You should reduce the call depth by combining calling and called functions. This error should normally be detected by the latest version of TERMEX PRO when you load a project.
>>> RUNTIME ERROR <<< **Multiple memory allocation.**	You have attempted to request memory for the same EPCA function more than once. This problem cannot be solved by users; please contact EX TEC.
>>> RUNTIME ERROR <<< **Uninterruptible loop.**	A loop which cannot be interrupted has occurred while an EPCA program was being processed. This message can occasionally also occur when no EPCA programs are running. This problem cannot be solved by users; please contact EX TEC.

5 For the Project Designer

5.1 Converting TERMEX 2xx/3xx Projects

This section is intended for project designers who want to convert an existing TERMEX 2xx/3xx project to TERMEX 7xx, or who are at least already familiar with and have used a TERMEX 2xx/3xx project.

The operating procedures and the functional principles are basically identical. Certain adaptations were however unavoidable in a few instances. Elsewhere, it seemed prudent to make various changes in order to safeguard TERMEX 7xx's flexibility for future requirements.
We shall only mention the most important changes here and describe the modified commands in detail later on.

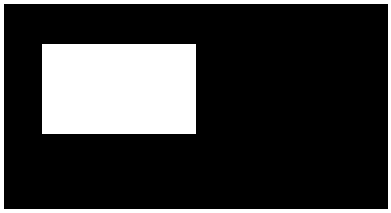
General

16-bit coordinates	The coordinates of the TERMEX 7xx display (640 x 480 pixels) can no longer be represented with 8 bits. 16 bits are therefore now used for all commands with coordinates. This means that a coordinate is represented at these points by one high byte plus one low byte. The high byte always precedes the low byte. The parameters are abbreviated in the command descriptions, e.g. x1high x1low.
16-bit handle numbers	Windows are given a so-called handle number when they are opened. This number allows them to be accessed again later on. Handle numbers for the TERMEX 7xx are now 16 bits wide. This means that two bytes, namely a high byte and a low byte, now have to be transmitted to the terminal for all commands that require a handle number to be specified.
8-bit color numbers	A color must now be specified for all outputs on the display. An 8-bit color number is used to do so. This color number represents an index within the currently active color palette. The palette is predefined when the unit is started up and can be modified as desired within the framework of TERMEX PRO projects.
Graphic display	All graphic elements (point, line, rectangle, box) are now output in windows. This entails the following steps: open special graphic windows, point the graphic handle to the required window and output the graphic elements. You must however remember to refer relative coordinates to the top left-hand corner of the window for all outputs.
Softkey bar	Softkey bars no longer exist in the firmware as independent objects with an automatic shift function. Texts or bitmaps can be positioned for the purpose of outputs using the Screen Editor in TERMEX PRO.

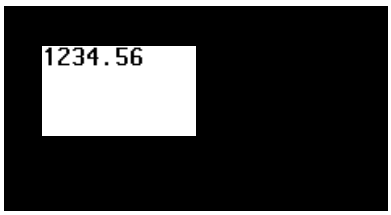
Commands which have changed since TERMEX 2xx/3xx

5.2 Text Output

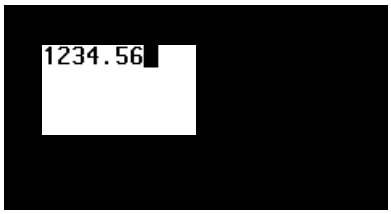
A window must always be opened before a text can be output on the TERMEX 7xx. The size and position of the window cannot be changed after it has been opened. There are several window properties that can be altered again at any time, however. The following example explains how to do so:



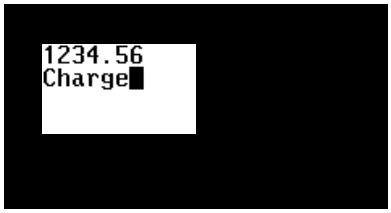
A window is opened in the top left. The light area is now reserved for the text output. The background has been deliberately left dark to make the window area easier to recognize in this example. If the background is empty (light), it is normally impossible to tell whether or not a window has been opened!
(Refer to **Open text window** on page 63)



The output handle is placed on this window.
Since it is possible for several different windows to be open at the same time and characters output in all of them, a kind of handle (or pointer) must be used to determine the window for the text output.
(Refer to **Set handle1 (text output)** on page 69)
Text can now be output in this window.
(Refer to **Possible Settings** on page 76)



The cursor, in other words the character position at which the next character will be output, has been invisible up to now. This property can be set with the "set window style" command.
(Refer to **Set window style** on page 65)
Examples of other window styles include the window border and inverse representation.



The **LF** control character (Line Feed, 0Ah) positions the cursor on the next line. You can then output more characters in this new line.
(Refer to **Line feed, LF** on page 71).

If you no longer need the window open, you can close it again. The background which was concealed when you opened the window is then restored.
(Refer to **Close window** on page 65)

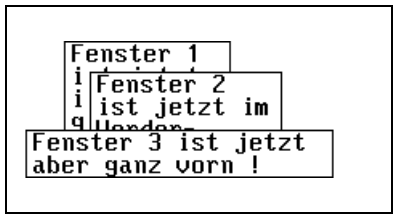
In addition to the above-mentioned LF, there are various other control characters which you can use to navigate the cursor:

- **BS** (backspace) moves the cursor one position to the left (refer to page 70).
- **CR** (carriage return) moves the cursor to the beginning of the line (refer to page 71).
- The **cursor commands** described on page 71 ff. allow you to move the cursor in all four directions or to set it to an absolute position.

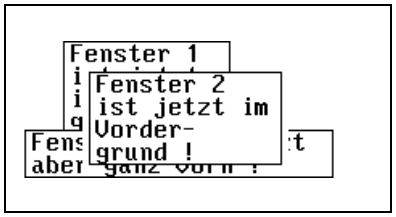
Display in the foreground or background with TERMEX 7xx

In addition to the options described above, the terminal window system incorporates a number of other special features which are described in the following example.

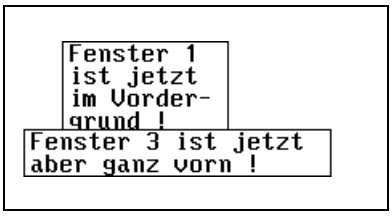
Window 1 is opened in the normal way. Window 2 is opened not next to it, but so that it overlaps window 1. Window 1 is **deactivated** as a result. This means that you can no longer output characters in this window, nor can you modify the window style. Non-active windows can be closed at any time, however.



An additional window is opened on top of window 1 and window 2. Since it overlaps both window 1 and window 2, these two windows are deactivated and you can only output characters in window 3. This example shows how a window structure can have several layers.



A special activation command (refer to page 70) allows you to move windows to the foreground explicitly. Window 2 is now active while windows 1 and 3 are deactivated.



A special command allows you to make window 2 invisible (refer to page 70). The window has not been deleted, even though it may appear that way. The activation command described above will bring it back into the foreground again instantly.
Window 3 is incidentally now active again.

Cursor blinking in windows

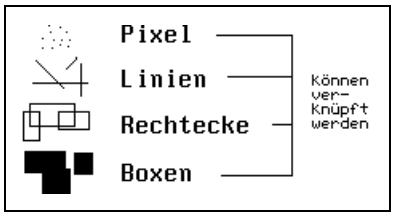
Only one cursor can blink on the entire terminal display at a time. The blinking frequency can be selected in the setup. You can also switch off cursor blinking completely there. Which particular cursor blinks in a given situation depends on the status of the terminal:

1. If a **set variable** is active in PLC mode (ready for input), the cursor for this variable blinks, regardless of whether or not it has been switched on in the corresponding window style.

2. If a set variable is not active, the cursor for the window which is waiting for user inputs via the **keyboard input handle** blinks (handle 2 points to a window, refer to page 69). This only applies, however, if the cursor for the window in question has been switched on under 'Style' (Handle2 ≠ 0).

3. If free keyboard inputs are allowed (Handle2 = 0), the cursor for the window in which the control computer outputs characters via the output handle (handle 1) blinks. This only applies, however, if the cursor for the window in question has been switched on under 'Style'. This setting is particularly useful for applications in which key codes must first be filtered by the control computer. This computer receives the key codes and converts them into character codes, which are then returned to the terminal.

5.3 Graphic Display



Pixels, lines, rectangles and boxes can be used to build graphics from individual elements. They must be output in special graphic windows. You can determine the output window by setting handle G. The output coordinates are relative (they refer to the top left-hand corner of the window).
These graphic elements can be used in TERMEX PRO projects (refer to **[TERMEXPRO]**) and output directly on the terminal using the EX TEC protocol (refer to page 73 ff.).

Bitmaps are ideal for complex drawings of every possible size. These rectangular graphic objects are output directly in windows in the same way as characters. Bitmaps can only be created in TERMEX PRO projects (refer to **[TERMEXPRO]**).
In the example on the left the bitmap fills almost the entire display. The explanatory text has been inserted in a separate window.

5.4 Variables

Variables are used to input and output actual values on a screen. Output values are written by the control computer at predefined data word positions in the data block (refer to page 28), while input values are transferred by the terminal to positions where they can be read by the control computer. Variables can only be used within projects and in conjunction with TERMEX PRO.

Actual variables

- In TERMEX PRO variables are defined by stating the variable type and the respective parameters. They can then be inserted on various screens, though only once per screen. Variables can only be used in conjunction with TERMEX PRO.

- How variables are updated depends to a large extent on the selected protocol. In the case of protocols where the terminal plays the role of communication master (e.g. 3964R/RK512), variables are automatically updated after the corresponding data word area has been read. With other protocols (EX TEC and MODBUS), they are updated after every write access to the variable area of the data block. There is also a function which allows variables to be updated at definable time intervals.

- Each variable requires a specific area within the data block. This may consist of several data words or of just one bit. Variable areas are normally organized without overlapping, so that values can be assigned to each variable independently. Occasionally, however, overlapping can be helpful, for example if you want to output a particular value simultaneously in different formats. In this case two variables will be stored in the same data words. It is always the responsibility of the project designer to make sure that variables are correctly assigned within the data block.

- If the window in which a variable is output is deactivated (for example, because it is overlapped by message windows), the variable can no longer be updated. It is important to remember this because the variable itself may well still be completely visible. It is best to avoid this kind of screen overlap altogether in order to exclude any risk of confusion.
  If the window containing the variable is activated again (because the message disappears), the new value will take effect immediately.

- Variables can be scrolled in windows just like normal text. As long as variables remain visible in the window they will continue to be updated correctly. If you scroll out of the window, the variables will disappear too.

Set variables

- Set variables generally behave like actual variables, but in addition permit values to be input on the terminal and transferred to the control computer.

- You can switch between the different set variables on a screen (refer to Key Assignment on page 6) .

- A cursor blinks at the beginning of the field for the currently active set variable.

- As long as the cursor is positioned to the beginning of the variable field, this field is updated in the same way as an actual variable field. As soon as you start to enter a value using the number block, on the other hand, the update function will be suspended until you have completed your input.

- An input is not interpreted and written in the data block until you confirm it by pressing the ENTER key. This only happens once; the data direction is then reversed and the set variable behaves like an actual variable.
  The PLC can thus reject the set values entered by you by immediately overwriting the data words in the data block. In this case, you will be able to watch your input value as it is overwritten.

- Instead of entering numbers, you can also increment or decrement the values of variables using the increment and decrement keys (refer to Key Assignment on page 6).

Variable type	Effect
BCD1 BCD01 BCD2 BCD02	The current value is incremented or decremented by one.
BINA VBINA	The cursor jumps to the next higher or lower value that can be displayed (depending on the scale factor).
BINB VBINB	The cursor jumps to the next higher or lower value that can be displayed.
TEXT	Use *increment* to change the bit from 0 to 1 and *decrement* to switch it from 1 to 0.
TEXT16	The cursor jumps to the next higher or lower variant value.
MSGLIST	The cursor is moved up or down and the screen is scrolled if necessary.

- If an active set variable is deactivated by an overlapping window, the terminal attempts to find another set variable that can be activated and will allow the cursor to be displayed.

Overview of available variable types (actual and set variables)

Variable type	Description	Details on page
BCD1	BCD number (max. 4 digits without leading zeros)	83
BCD01	BCD number (max. 4 digits with leading zeros)	83
BCD2	BCD number (max. 8 digits without leading zeros)	83
BCD02	BCD number (max. 8 digits with leading zeros)	83
BINA	Binary number (unsigned integer, scalable)	84
VBINA	Binary number (signed integer (positive or negative), scalable)	84
BINB	Binary number (long unsigned integer, not scalable)	85
VBINB	Binary number (long signed integer (positive or negative), not scalable)	85
TEXT	2 text variants, depending on bit state	86
TEXT16	Up to 65535 text variants, depending on DW value	87
ASCII	Character string (variable length)	88

Internal variables

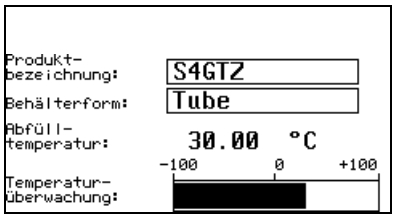
- There are also internal variables in addition to set and actual variables. They are designed to allow values which originate either in the terminal or in the connected peripherals to be output to the data block.

Overview of available internal variables:

Variable type	Description	Details on page
DATE	Output of date	88
TIME	Output of time	90
MSGFILTER	Output of current message filter	91
MSGLIST	Output of message lists	91
PLUGID	Output of plug ID (for mobile units)	93

5.5 Bar Graphs

Bar graphs are used to display the current values of variables graphically. The bar is automatically updated by the terminal. The control computer does not have to constantly redraw it with the aid of special commands.



In this example, the bar graph shows the temperature as a bar on a scale. At the same time the exact value is output as a VBINA variable. Since the display limits for the bar graph are freely selectable, either a rough overview or a very detailed view can be generated.

- Exactly one bar graph can be generated for each defined variable. The permitted variable types are BINA, VBINA, BINB and VBINB.
- If one variable value must be output on several bar graphs, a separate variable must be defined for each additional bar graph. The additional variables will normally have identical parameters.
- The bar graph is opened in the foreground **like a window**.
- The specified **variable name** must refer to a variable that has already been defined.
- The **orientation** (up, down, left or right) determines the direction in which the bar "grows" if the value increases.
- The specified **display limits** always refer to the previously scaled display values of the corresponding binary variables.

Example:

Variable:

Variable type:	BINA
Integer positions:	4
Decimal positions:	2
MIN (terminal):	0
MAX (terminal):	100000
MIN (PLC):	0)
MAX (PLC):	65000)

Bar graph Properties window:

MIN:	25000
MAX:	75000
Orientation:	UP

In this case the variable is output within the limit range from 0.00 to 1000.00. The binary reference range in the data block is 0 to 65000 (the decimal point is not specified).
The bar graph which is generated can display values between 250.00 and 750.00. If the actual value of the variable is less than 250.00, the bar is drawn with a length of zero. If the variable value is greater than 750.00, the bar appears with its maximum length.

- Bar graphs behave exactly like windows when they are closed. The previously concealed background is restored. A new bar graph can then be opened for the same variable if desired.
- Any scale lines can be drawn around the bar graph using line commands. Please remember, however, that this scale is not an integral part of the bar graph, in other words the background (if there is one) is not restored when the window is closed.
- Bar graphs are only updated if the corresponding variable is output on the same screen. It is not sufficient simply to define this variable.

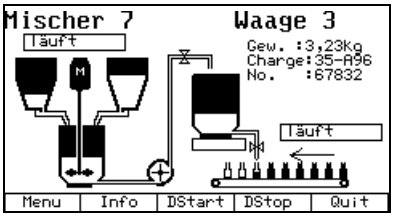
5.6 Trend plotters

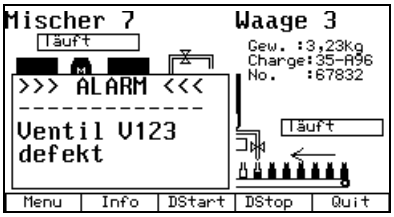
Trend plotters enable the time characteristics of variable values to be output. They can only be used within the framework of projects in conjunction with TERMEX PRO. Please refer to [TERMEXPRO] for a description of how to create trend plotters.

- Each trend plotter can show up to six graphs, each of which represents the curve for a different variable.
- The variable archive must be activated in order to be able to trace a variable (in the Attributes dialog box for the variable concerned). The time characteristic for this variable is then saved automatically by the terminal, regardless of whether or not a trend plotter with a graph for it is open. It is not possible to specify a sampling time for the variable archive, because all value changes which are detected by the terminal are saved automatically.
- The variable archive is resident, in other words if the terminal is switched off and then on again, the values remain stored. The memory is organized as a ring buffer, so that older entries are overwritten by new ones.
  Important: If the firmware of the TERMEX 750 is updated, the entries in the variable archive will normally be lost.
- When a trend plotter is opened in the Screen Editor of TERMEX PRO, its size can be freely defined in the same way as for a window.
- In addition, it is possible to set the number of values along the time axis as well as the time resolution between two displayed values. The display is updated together with the variable, in other words it may be slower than the time resolution (example: resolution 500 ms, update interval 1000 ms -> two values per update).
- The time resolution can also be described as the sampling time for the plot function. Please remember that the time characteristic is saved in the variable archive without any sampling losses. Which details are visible thus depends entirely on the time resolution of the trend plotter.
- In the current view, the right margin of the graph (current plot position) corresponds to the present time, while the left origin is calculated from the number of values and the time interval (e.g. 150 values, time interval 3 seconds -> origin -450 seconds = -7.5 minutes).
- The minimum and maximum values must be specified for the Y-axis. If the auto-scaling function is active, this value range is automatically stretched by the trend plotter at runtime.
- When the trend plotter is opened, a border with scale lines appears automatically. The size of the border can be specified separately in each direction. The time axis is divided up in such a way that suitable, integer time steps are used (e.g. 1 s, 2 s, 5 s, 10 s, etc.). The Y-axis is divided into powers of two (2, 4, 8, 16).
- If a graph reaches the right margin of the trend plotter, it is scrolled over to the left. The scroll range can be specified in character steps (e.g. in the above example, 15 means that after 15 values have been output, the graph is scrolled 15 steps = 10% to the left).
- You can specify for all graphs which are output by the trend plotter whether they should have a ramp or a staircase style. If you select a ramp style, the points on each graph are joined together by oblique lines, while vertical and horizontal lines are used for the staircase style.
- The required variable and the plot color must be selected separately for each graph. All numeric variable types are allowed. It is also possible to activate double-line representation in order to obtain thicker lines.
- Auxiliary variables can be defined for each trend plotter. They have the type PPARAMS and can output the following information: MIN value, MAX value, origin time and scale resolution of the time axis. These variables can be arranged around the defined trend plotter in any way using the Screen Editor. Labeling a trend plotter with auxiliary variables has the advantage that values which vary at runtime are always output correctly.

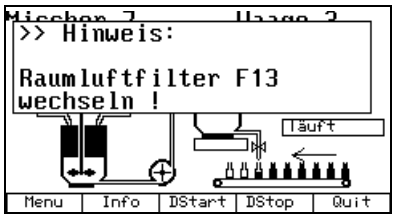
5.7 Messages

Messages are used to display relevant information, malfunctions and errors on an open screen:





This example shows a so-called MULTI message. This message type is very versatile. It allows you a considerable amount of design freedom. A message can consist of up to ten windows of any size and with any position and content.



If you do not want the project design to be too complex, it is best to use MONO messages. MONO messages only require the text to be specified. The position, size and character set of a MONO message window are already preset. Four lines of 24 characters each are available.

Messages are activated by setting a bit which is assigned to them in the data block. They are deactivated by resetting this bit (refer to Data Area under **DW46** on page 34 ff.).

Message acknowledgment

Messages on the terminal can be acknowledged by pressing the message acknowledgment key (refer to **Key Assignment** on page 6). They can also be acknowledged remotely by the control computer (refer to Data Area under **DW46** on page 34).
The numbers of the acknowledged messages can be read from the data block (refer to **DW2** on page 30).

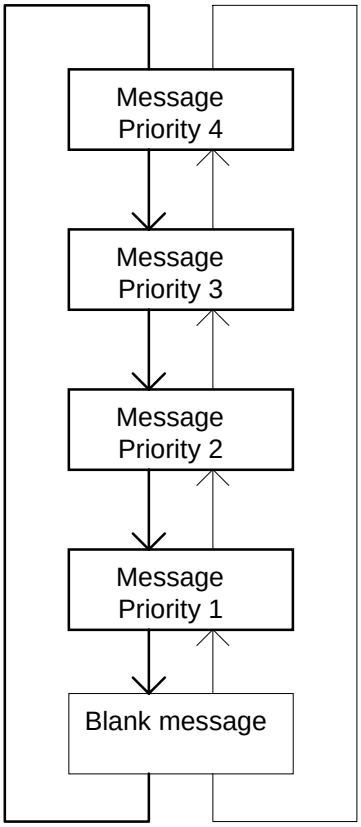
Message filters

The terminal message system supports several different filters that determine which messages are displayed.

You can select the desired message filter either in the setup or in **DW23** (refer to page 32) of the data block.
The following options are available:

Filter number	Short filter name	Description of message filter
0	AnQ	Messages are active and have not yet been acknowledged. Messages whose message bit is set but that have not yet been acknowledged are displayed. Non-active messages, messages that are no longer active and acknowledged messages are not displayed.
1	nQ	Messages are not acknowledged ("major faults"). Messages that were or still are active but that have not yet been acknowledged are displayed. Once a message has been activated, it will not disappear again until it has been acknowledged.
2	QA	Messages have already been acknowledged, but they are still active "queued messages"). This filter is mainly used to check which messages have already been acknowledged.
3	NONE	No messages are displayed. This filter can also be selected if you only want messages to be displayed in message lists.
4	A	Active messages. Active messages are displayed regardless of whether or not they have been acknowledged. It is possible to distinguish acknowledged messages from unacknowledged messages, however: If an active message has already been acknowledged, the message windows are inverted.
5	AnQ+QA	*Standard display (AnQ)* The oldest active message that has not yet been acknowledged is displayed. You cannot scroll between queued messages. *Alternative display (QA)* You can switch to this display by holding down the Shift key. The most recently acknowledged active message is displayed. The message shift keys allow you to scroll through all active and acknowledged messages in chronological order. Please refer to page 25 for additional information regarding this message filter.

The terminal forms a set of messages that match the selected filter. The system initially displays the message in this set that has the highest priority (exception: AnQ+QA, see above). The priority is determined when the message is defined in TERMEX PRO and remains unchanged throughout the terminal's runtime. If two messages have the same priority, the one with the higher message number is also ranked higher.



Comments

- You can use the message shift keys (refer to **Key Assignment** on page 6) to scroll through the complete list of queued messages in order of priority.
- Pressing the arrow down key ▼ displays the message with the next lower priority; pressing the arrow up key ▲ displays the message with the next higher priority.
- A blank message, in other words one which does not contain any of the queued messages, is added to the message list. You can thus continue working on the current screen, even if you cannot rectify the cause of a message immediately.
- New messages which have a higher priority than the displayed message continue to appear immediately. If you have already scrolled down to the message with priority 1 as shown in the example opposite and a new priority 3 message occurs, the new entry is added to the list and displayed immediately.
- If a displayed message is deactivated, the message with the highest priority that has not yet been displayed appears next. If, for example, you have scrolled your way through all the queued messages and the current message disappears, no more messages will be displayed. If a message with a lower priority than the previously displayed one is generated in the meantime, it will be displayed because you have not yet acknowledged it.
- Owing to the many different options that can be set for messages and the extremely wide range of chronologies, the system can be very complex. We recommend testing the equipment to verify whether or not your chosen settings will actually trigger the desired actions.
- The alarm status LED on the TERMEX 7xx lights up if at least one message is active on the terminal. This LED is independent of the type both of the displayed messages and of the lowest message priority. You can select the same function for the LED array in the setup.
- If you want to avoid being disturbed by "unimportant" messages during critical procedures, you can specify that messages should only be displayed if they have a minimum priority level. Messages with a lower priority than the set value will not be displayed (refer to **DW23** on page 32).

Comments on the AnQ+QA filter

- All other message filters have one thing in common: the order of the displayed messages depends on their priority.
- With the **AnQ+QA** filter, the priority is irrelevant. The deciding factor is the order in which the messages occur (the time at which they are activated).
- You can still disable messages with a lower priority by setting a minimum display priority in **DW23** (refer to page 32)!
- If this filter is used, no more than 254 messages can be active at any given time. This limit must be adhered to, as otherwise malfunctions may result.

- In practice, this filter means that messages can (or must) be acknowledged consecutively in the order in which they occur. All acknowledged messages that are still active will then appear in the alternative list. You can check there regularly whether or not these messages are still active. If the messages are deactivated in the meantime, they will disappear from the alternative display.
- You can specify the time for which the Shift key must remain pressed under "Shift Switch Time" in the *General Settings Menu*. This time can be anything from 0.5 seconds to 4 seconds. The default setting is 2.0 seconds.
- If the Shift key is also used for other functions, for example to acknowledge messages with <Shift> 5, you should be careful not to press it for too long on its own. The switch time is reset as soon as you press the second key (in this case "5"). You can thus acknowledge several messages consecutively by keeping the <Shift> key permanently pressed and repeatedly hitting "5".

Message lists

In addition to displaying messages in a freely definable format on an open screen, you can also output them in lists.
Windows containing a messages list are normally an integral part of the screen.

Message lists have the following characteristics:

- Several queued messages can be displayed simultaneously
- Each message entry usually takes up one line in the window
- The format of the message line can be defined freely using placeholders
- The filter for the message list is also freely definable
- The messages can be sorted in ascending or descending order using various filters
- If there are more messages than will fit into one window, they can be scrolled
- They can also be printed out on a line printer which is connected to SER2
- Several message lists can be displayed simultaneously if required (up to 8)

A message list is defined like a variable in the Variable Editor of TERMEX PRO and inserted on a screen as a variable using the Screen Editor. In order to be able to scroll through a message list, the message list variable must be a set variable.
If you position the variable cursor in the message list window using the cursor keys, you can then navigate in this list with the increment and decrement keys.
If you switch off the message display in the setup (None instead of AnQ, for instance), no more messages will be displayed, but the message lists will continue to be updated nonetheless.

Example of a message list containing all active messages, sorted according to priority:

```
Mess3    Supply temp. too high      Pri:103  Start:28.02.2000 17:21:33
Mess2    Boiler K10 pressure high   Pri:102  Start:28.02.2000 16:59:01
Mess12   Valve V9 damaged           Pri:101  Start:28.02.2000 17:10:21
```

The name of the message, the message text, the priority and the start time (date and time) are displayed.

Example of a message list containing all message events, sorted according to the message start time:

```
Mess3    A:28.02. 17:21  E:--.--. --:--  Q:--.--. --:--
Mess12   A:28.02. 17:10  E:--.--. --:--  Q:--.--. --:--
Mess2    A:28.02. 16:59  E:--.--. --:--  Q:--.--. --:--
Mess3    A:27.02. 11:43  E:27.02. 12:10  Q:27.02. 11:45
Mess2    A:26.02.  9:43  E:27.02. 12:10  Q:27.02. 11:45
```

The name of the message, the message start time, the message end time and the acknowledgment time are displayed. The hyphens mean that the message is either still queued or that it has not yet been acknowledged.

You can find an exact description of the formats, filters and sort options in the reference section under Variables (MSGLIST on page 91).

Message events

The messages which are output in message lists are read from a message event memory. This memory is used to keep track of all changes relating to messages:

- New messages
- Messages which disappear
- Messages which are acknowledged
- Messages which appear several times within a short period (chaining)

The message event memory only has a limited storage capacity. Older events must therefore be deleted from it if they are no longer needed. Since the memory is not organized according to the FIFO (first in, first out) principle, it is not necessarily the very oldest events that are deleted, but rather those which are considered to have been completed (because the messages have disappeared again) and which occurred some time ago.
You can specify this time in the setup:

mess_StopDelTimeDay	Number of days until deletion from Message_event.list
mess_StopDelTimeHour	Number of hours until deletion from Message_event.list

Note:

- Events which are not yet considered to be completed are not affected by this setting.
- All events are deleted if a new firmware version is installed! They only remain stored in a few exceptional cases. You can tell from the startup message that appears after the new firmware is loaded what has actually happened (refer to page 8).

5.8 Data Area

The data area has a length of 512 data words (DW). It is used to exchange data between the terminal and the connected controller. It exists both in the terminal and in the control computer.
The data area is split into two data blocks of 256 words each for certain protocols. The second data block has the same number as the first block plus one:

Data area	Data block	Address in DB
DW 0	x	DW 0
...		…
DW 255		DW 255
DW 256	x+1	DW 0
…		…
DW 511		DW 255

The next page shows an overview of the data word functions and the data bits. The individual data words and data word areas are described in detail afterwards.

T7xx Bit no.	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Direction TERM PLC	Description
DW0	MED	- Reserved -							Displayed screen								→ •	Return
DW1	MES	- Reserved -					Displayed message										→ •	Return
DW2	MQI	NQM	- Reserved --				Acknowledged message										→ •	Return
DW3																	→ •	
DW4	F16	F15	F14	F13	F12	F11	F10	F9	F8	F7	F6	F5	F4	F3	F2	F1	→ •	Key states
DW5							S10	S9	S8	S7	S6	S5	S4	S3	S2	S1	→ •	Ix: Digital
DW6							9	8	7	6	5	4	4	2	1	0	→ •	inputs
DW7			.	PLS	MNS	ENT	ALT	SHT	CLR	INF	CRI	CLE	CDN	CUP	BOT	TOP	→ •	at X9, X11
DW8	I15	I14	I13	I12	I11	I10	I9	I8	I7	I6	I5	I4	I3	I2	I1	I0	→ •	
DW9																	→ •	
DW10																	→ •	Reserved
DW11																	→ •	Reserved
DW12																LIVE	→ •	State
DW13															SAK	STA	→ •	
DW14																	→ •	Reserved
DW15	F16	F15	F14	F13	F12	F11	F10	F9	F8	F7	F6	F5	F4	F3	F2	F1	• ←	LEDs
DW16	!	ON	COM	?	ALB		S10	S9	S8	S7	S6	S5	S4	S3	S2	S1	• ←	Ox: Digital
DW17	PLS	MNS	ALT	SHT	CRI	CLE	CDN	CUP									• ←	outputs
DW18	O15	O14	O13	O12	O11	O10	O9	O8	O7	O6	O5	O4	O3	O2	O1	O0	• ←	at X10, X12
DW19	- Reserved -								Screen number								• ←	Screen call
DW20																	• ←	
DW21																	• ←	
DW22																	• ←	
DW23	MF	SDM	DSE	Preselected DS					Minimum display priority								• ←	
DW24	QSB										BPS	BPE	KSN	KSE	QSN	QSE	• ←	Control bits
DW25																	• ←	
DW26									ANZS (number of characters to send)								• ↔ •	
DW27	Z 1								Z 2								• ←	
DW28	Z 3								Z 4								• ←	
DW29	Z 5								Z 6								• ←	General
DW30	Z 7								Z 8								• ←	data
DW31	Z 9								Z 10								• ←	exchange
DW32	Z 11								Z 12								• ←	TERM ← PLC
DW33	Z 13								Z 14								• ←	
DW34	Z 15								Z 16								• ←	
DW35	Z 17								Z 18								• ←	
DW36																		
DW37																		
DW38																		
DW39																		
DW40																		
DW41																		
DW42																		
DW43																		
DW44																		
DW45																		
DW46	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	• ←	Message frame
-DW53	127	126	125	124	123	122	121	120	119	118	117	116	115	114	113	112		
DW54	143	142	141	140	139	138	137	136	135	134	133	132	131	130	129	128		
-DW61	255	254	253	252	251	250	249	248	247	246	245	244	243	242	241	240		
DW62	271	270	269	268	267	266	265	264	263	262	261	260	259	258	257	256		
-DW69	383	382	381	380	379	378	377	376	375	374	373	372	371	370	369	368		
DW70	399	398	397	396	395	394	393	392	391	390	389	388	387	386	385	384		
-DW77	511	510	509	508	507	506	505	504	503	502	501	500	499	498	497	496		
DW78																	• ↔ •	
...																	• ↔ •	Variable
-DW511																	• ↔ •	area

Bit no.	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	TERM PLC	Description
DW0	MED	- Reserved -							Displayed screen								→ •	Return

The terminal returns the number of the displayed screen in the low byte of data word 0. Data words DW0 and DW1 and the key states are synchronized. DW0 and DW1 indicate the screen on which each key was pressed and show whether a message was displayed at the time. Even though delays may occur in communication, it is thus always possible to tell exactly which key state refers to which displayed information.

As soon as the system starts to build a new screen, it switches to data word 0. If the selected screen is entered in DW19, the same number will appear in data word 0 after a short delay. Even if the function keys on the terminal are used to switch over locally, the displayed screen is entered there.
MED: If this bit is set, the terminal is in the message event display mode.

Bit no.	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	TERM PLC	Description
DW1	MES	- Reserved -					Displayed message										→ •	Return

If the MES bit is set, bits 9-0 indicate the number of the message which is currently visible on the terminal. If the MES bit is not set, the displayed number is invalid and there is no message on the display.

Bit no.	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	TERM PLC	Description
DW2	MQI	NQM	- Reserved -				Acknowledged message										→ •	Return

The number of the last message to have been acknowledged manually is indicated by bits 0-9 of DW2. This number is only valid if the MQI flag is set. Automatic acknowledgments (e.g. after a major fault or by means of acknowledgement bits from the control computer) are not entered. Both the number and the MQI flag remain unchanged until a new manual acknowledgment is received. If the last acknowledged message is deactivated and then activated again, however, both the number and the MQI flag are reset. The NQM flag indicates that some messages have not yet been acknowledged on the terminal. This flag is set by the terminal to 1 if at least one message is still unacknowledged. If all messages have been acknowledged, the NQM flag is set to 0.

The numbers of the acknowledged messages are returned automatically and consecutively (one number per cycle) with interfaces where the terminal plays the role of communication master. If messages are acknowledged in quick succession, the message numbers are buffered internally. With all other interfaces (EX TEC protocol, MODBUS), a new number is supplied (if one is available) for each read access to DW2. This procedure may cause problems if the protocol drivers of the controller read the data word asynchronously (usually also more frequently) with the subsequent data word processing steps. It can happen that some numbers are not able to be processed in the controller as a result.
For this reason, the numbers can now also be read securely by means of a kind of handshake:

- The QSE bit must be set for this transfer mode. It is not set for the conventional transfer mode (default: QSE bit not set).
- Each time the QSN bit is set, the next number is entered in DW2. It is not necessary to reset the QSN bit because the terminal does so automatically (refer also to **DW24** on page 32).

There is also a fully buffered variant of the acknowledgment number return. If the QSB bit is set, each new state that affects the acknowledgment number and the MQI flag is returned buffered in DW2 (refer also to **DW24** on page 32).
The main difference from the normal state (QSB=0) is that the reset of the last acknowledged message number (message deactivated and then activated again) is now also buffered.

Each rising edge of the QSN bit thus causes the next state of the MQI flag and the acknowledgment number to be written in DW2. The QSN flag is not reset automatically. It must be reset explicitly by the control computer.
If the state has not changed by the next rising edge of the QSN bit, this edge is rejected. Up to this point, the edge has been stored and the new state output immediately after a state change.
The NQM flag is also updated externally in this mode without being controlled, in other words it follows the message bits and the locally or remotely controlled acknowledgment exactly.

Bit no.	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	TERM PLC	Description
DW4	F16	F15	F14	F13	F12	F11	F10	F9	F8	F7	F6	F5	F4	F3	F2	F1	→ •	Key states
DW5							S10	S9	S8	S7	S6	S5	S4	S3	S2	S1	→ •	Ix: Digital
DW6							9	8	7	6	5	4	4	2	1	0	→ •	inputs
DW7			.	PLS	MNS	ENT	ALT	SHT	CLR	INF	CRI	CLE	CDN	CUP	BOT	TOP	→ •	at X9, X11
DW8	I15	I14	I13	I12	I11	I10	I9	I8	I7	I6	I5	I4	I3	I2	I1	I0	→ •	
DW9																	→ •	

The key bits are set to 1 as long as the corresponding key remains pressed (multiple keystrokes are possible). You can find an explanation of the abbreviations used for the integrated keys (e.g. PLC) under **Key Assignment** on page 6.

I15...I0: Digital inputs at interfaces X9 and X11 (refer to [TERMEX])

Bit no.	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	TERM PLC	Description
DW12																LIVE	→ •	State
DW13															SAK	STA	→ •	

Meanings of the bits:

STA: Start terminal: following a startup or reset, the terminal sets this bit to 1 exactly once. The PLC can evaluate the bit and then trigger a restart of the PLC software. The bit must then be reset by the PLC!

SAK: Update set value: this bit is set to 1 by the terminal each time you enter a set value. The PLC can evaluate the bit, check the plausibility of the set values and process the new set values further. The bit must be reset to 0 by the PLC program (in the PLC) after a 1 has been detected for this purpose.

LIVE: The terminal is switched on and communication is possible. This bit is set to 0 by the terminal during operation. The PLC can evaluate the bit and check whether communication is possible and the terminal is switched on. The bit must be reset to 1 by the PLC program (in the PLC) after a 0 has been detected for this purpose.

Bit no.	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	TERM PLC	Description
DW15	F16	F15	F14	F13	F12	F11	F10	F9	F8	F7	F6	F5	F4	F3	F2	F1	• ←	LEDs
DW16	!	ON	COM	?	ALB		S10	S9	S8	S7	S6	S5	S4	S3	S2	S1	• ←	Ox: Digital
DW17	PLS	MNS	ALT	SHT	CRI	CLE	CDN	CUP									• ←	outputs
DW18	O15	O14	O13	O12	O11	O10	O9	O8	O7	O6	O5	O4	O3	O2	O1	O0	• ←	at X10, X12

These bits are used to directly control the states of the integrated LEDs.

Status bit	Effect
0	LED off
1	LED on

Since the LEDs are driven by various instances (data block, EPCA, firmware), there is a series of masks for restricting access. In the default setting only the LED bits with light shading are enabled (F1...F16, S1...S10, ALB), while the bits with dark shading are reserved for the firmware. You can modify these masks using EPCA programs (refer to [EPCA]).

Bit no.	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	TERM PLC	Description
DW19	- Reserved -								Screen number								• ←	Screen call

You can call up screens by entering the required screen number in the low byte of DW19 (example: entering xx17h in DW19 opens screen 23).

Bit no.	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	TERM PLC	Description
DW23	MF	SDM	DSE	Preselected DS				Minimum display priority									• ←	Control bits
DW24	QSB												KSN	KSE	QSN	QSE	• ←	
DW25																	• ←	

The minimum priority for the displayed messages can be set with bits 0-8 of DW23. If you enter 0, all messages are displayed. If you enter a number > 255, no messages will be displayed because the highest possible message priority is 255.

MF: Major faults are enabled with MF=1
SDM: (Select Display Mode) The preselected DS value is accepted for the message filter with a 0→1 edge
DSE: (Display Select Enable) The DSE flag should always be set to 0
DS: (Display Select) Preselected value for the message filter, accepted with the SDM flag
QSB: QSB=1 switches on the fully buffered acknowledgment number return function (see below)
KSN: A rising edge (0→1) causes the new state to be entered in the key state bits, providing KSE=1. KSN must be reset to 0 again by the control computer
KSE: KSE=1 switches on the update function with KSN for the key state bits (EX TEC and MODBUS protocols only).
If KSE=0, the key state bits are updated with each read access
QSN: A rising edge (0→1) causes the new acknowledgment number to be entered in DW2, providing QSE=1. QSN must be reset to 0 again by the control computer
QSE: QSE=1 switches on the controlled return function with QSN for the acknowledgment numbers (EX TEC and MODBUS protocols only). If QSE=0, the new acknowledgement number is entered in DW2 with each read access

Example 1:

DW23	High byte								Low byte							
	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
	MF	SDM	DSE	Preselected DS				Minimum display priority								
	x	x	x	x	x	x	x	0	0	0	0	0	0	1	0	1

Only messages with a priority higher than or equal to 5 are displayed. All messages with a priority lower than 5 are not enabled for display.

Example 2:

DW23	High byte								Low byte							
	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
	MF	SDM	DSE	Preselected DS				Minimum display priority								
	x	x	x	x	x	x	x	1	0	0	0	0	0	0	0	0

Only messages with a priority higher than or equal to 256 are displayed. All messages with a priority lower than 256 are not enabled for display. Since the highest possible message priority is 255, this may mean that no messages are displayed at all. Bit 8 of DW23 disables the message display function.

Bit 15 enables major faults. The consequence of this is that all new messages with priority 250 are identified in the message history as major faults. Depending on how the message frame is used (refer to the section on using the message frame in the data block), all subsequent messages will be acknowledged automatically. If the MF bit is reset, this mode is exited again.

If the system detects a rising edge of the SDM bit (select display mode, bit 14), a new default value for the message filter is read from bits 9-12 (refer to page 22 ff. for the possible message types).

Example:

DW23	High byte								Low byte							
	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
	MF	SDM	DSE	Preselected DS				Minimum display priority								
	x	0	1	0	0	1	0	x	x	x	x	x	x	x	x	x

'2' has been entered in bits 9-12 as the default value for the message filter, in other words when the selection menu is opened, filter no. 2 (message acknowledged but still active) will be output as the default value. The DSE bit is set, i.e. the default value can be changed in the menu in order to select a different filter. The selection menu is opened by a rising edge (0→1) of the SDM bit.

Bit no.	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	TERM PLC	Description
DW26									ANZS (number of characters to send)								• ↔ •	
DW27	Z 1								Z 2								• ←	
DW28	Z 3								Z 4								• ←	
DW29	Z 5								Z 6								• ←	General
DW30	Z 7								Z 8								• ←	data
DW31	Z 9								Z 10								• ←	exchange
DW32	Z 11								Z 12								• ←	TERM ← PLC
DW33	Z 13								Z 14								• ←	
DW34	Z 15								Z 16								• ←	
DW35	Z 17								Z 18								• ←	

This frame must be used if commands or character strings are transferred from the PLC to the terminal via the data block in terminal mode.
Procedure in the PLC for transferring a character string:

1. Check whether ANZS (DW26 right) is set to 0. If not, a previous transfer has not yet been processed by the terminal. The check must be repeated.
2. Enter the characters that must be transferred, starting at DW27.
3. Enter the number of characters to send in ANZS (DW26 right).

You can also use this method to call up screens, for instance:

DW26									4h
DW27	ESC								'G'
DW28	Screen number (high byte)								Screen number (low byte)

Bit no.	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	TERM PLC	Description
DW46	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	• ←	Message frame
-DW53	127	126	125	124	123	122	121	120	119	118	117	116	115	114	113	112		
DW54	143	142	141	140	139	138	137	136	135	134	133	132	131	130	129	128		
-DW61	255	254	253	252	251	250	249	248	247	246	245	244	243	242	241	240		
DW62	271	270	269	268	267	266	265	264	263	262	261	260	259	258	257	256		
-DW69	383	382	381	380	379	378	377	376	375	374	373	372	371	370	369	368		
DW70	399	398	397	396	395	394	393	392	391	390	389	388	387	386	385	384		
-DW77	511	510	509	508	507	506	505	504	503	502	501	500	499	498	497	496		

Three different assignments can be defined for the message frame in the data block. You can specify how the message frame should be used in the setup under "General Settings - Message Use". There are three possible settings: 512 Messages or 256 Messages/256 Quit or 512 Messages/512 Quit.

If the "**512 Messages**" mode is selected, one message is assigned to each of the 512 bits in the message frame (default setting). A message is active if the corresponding bit is set.

DW46	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	• ←	Message bits
-DW53	127	126	125	124	123	122	121	120	119	118	117	116	115	114	113	112		
DW54	143	142	141	140	139	138	137	136	135	134	133	132	131	130	129	128		
-DW61	255	254	253	252	251	250	249	248	247	246	245	244	243	242	241	240		
DW62	271	270	269	268	267	266	265	264	263	262	261	260	259	258	257	256		
-DW69	383	382	381	380	379	378	377	376	375	374	373	372	371	370	369	368		
DW70	399	398	397	396	395	394	393	392	391	390	389	388	387	386	385	384		
-DW77	511	510	509	508	507	506	505	504	503	502	501	500	499	498	497	496		

If the "**256 Messages/256 Quit**" mode is selected, only 256 messages are supported. The PLC can both activate these messages and acknowledge them. Messages are assigned to the first 256 bits in the message frame. The remaining 256 bits are used to acknowledged messages from the PLC in a controlled manner. If major faults are active in this mode, subsequent messages are not acknowledged automatically. When designing projects with TERMEX PRO, you must be careful never to assign message numbers higher than 255 in this mode, because these messages will not be supported by the terminal (an attempt to activate one of these messages will be interpreted as an acknowledgment of another message).

DW46	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	• ←	Message bits
-DW53	127	126	125	124	123	122	121	120	119	118	117	116	115	114	113	112		
DW54	143	142	141	140	139	138	137	136	135	134	133	132	131	130	129	128		
-DW61	255	254	253	252	251	250	249	248	247	246	245	244	243	242	241	240		
DW62	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	• ←	Acknowledg-ment bits
-DW69	127	126	125	124	123	122	121	120	119	118	117	116	115	114	113	112		
DW70	143	142	141	140	139	138	137	136	135	134	133	132	131	130	129	128		
-DW77	255	254	253	252	251	250	249	248	247	246	245	244	243	242	241	240		

The "**512 Messages/512 Quit**" mode represents a combination of the two operating modes described above. All 512 messages can be acknowledged by the PLC remotely.
Important! The area starting at DW78 which was previously reserved for variables is now used for acknowledgment bits up to and including DW109. You must take account of this when placing variables in TERMEX PRO. Variables are only allowed to be placed in the data block in DW110 or higher data words. The protocol drivers in the terminal detect this setting automatically.

DW46	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	• ←	Message bits
-DW53	127	126	125	124	123	122	121	120	119	118	117	116	115	114	113	112		
DW54	143	142	141	140	139	138	137	136	135	134	133	132	131	130	129	128		
-DW61	255	254	253	252	251	250	249	248	247	246	245	244	243	242	241	240		
DW62	271	270	269	268	267	266	265	264	263	262	261	260	259	258	257	256		
-DW69	383	382	381	380	379	378	377	376	375	374	373	372	371	370	369	368		

DW70	399	398	397	396	395	394	393	392	391	390	389	388	387	386	385	384		
-DW77	511	510	509	508	507	506	505	504	503	502	501	500	499	498	497	496		
DW78	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	• ←	Acknowledg-ment bits
-DW87	127	126	125	124	123	122	121	120	119	118	117	116	115	114	113	112		
DW86	143	142	141	140	139	138	137	136	135	134	133	132	131	130	129	128		
-DW93	255	254	253	252	251	250	249	248	247	246	245	244	243	242	241	240		
DW94	271	270	269	268	267	266	265	264	263	262	261	260	259	258	257	256		
-DW101	383	382	381	380	379	378	377	376	375	374	373	372	371	370	369	368		
DW102	399	398	397	396	395	394	393	392	391	390	389	388	387	386	385	384		
-DW109	511	510	509	508	507	506	505	504	503	502	501	500	499	498	497	496		

Bit no.	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**	**TERM PLC**	**Description**
DW78																	• ↔ •	
...																	• ↔ •	Variable
-DW511																	• ↔ •	area

The entire **variable area** can be used to define variables in TERMEX PRO. In order to keep the time needed to transfer data as short as possible, it makes sense to arrange the variables without any gaps.
Ideally, all set variables should be grouped together. Internal variables which output data to the PLC (e.g. TIME, DATE, MSGFILTER) are treated as set variables.

5.9 Comparison of Supported Protocols

All the protocols which are supported by the terminal are described in the next few sections. The table below will help you to find the most suitable protocol for your application.

Protocol	Description	Page
EX TEC protocol	Standard protocol • Also used to load projects • Actively operated by the control computer by means of command sequences • Data exchange also possible via the data block • Easy to implement • High speed, low protocol overhead • No bus capability • Maximum transmission rate: 115200 baud	37
Siemens S5 AS511	Communication with Siemens S5 programming interface • Data exchange only via the data block • Very little effort necessary to use the protocol • Actively operated by the terminal • Secure • No bus capability Transmission rate: 9600 baud	40
Siemens S5 3964R	Communication with Siemens S5 communication module • Data exchange only via the data block • Little effort necessary to implement the protocol; only configuration required • Actively operated by the terminal • Secure • No bus capability • Maximum transmission rate: 19200 baud	40
Siemens S7 via Profibus DP	Interface to Siemens S7 controllers via Profibus DP • 3964R protocol used for the S7 interface • Data exchange only via the data block • Little effort necessary to implement the protocol; only configuration required • Actively operated by the terminal • Secure • Bus capability • Maximum transmission rate with 3964R protocol: 19200 baud	41
Siemens S7 via MPI	Interface to Siemens S7 controllers via MPI • 3964R protocol used for the S7 interface • Data exchange only via the data block • Very little effort necessary to set up the communication • Actively operated by the terminal • Secure • Bus capability • Maximum transmission rate with 3964R protocol: 19200 baud	46
MODBUS protocol	Universal protocol for communication with controllers • Data exchange only via the data block • Actively operated by the control computer (master) • Relatively high speed, low protocol overhead, still very secure due to CRC checksum • If a protocol driver is not already available, implementation is relatively complex (critical timing) • Bus capability via RS485 using slave addresses • Maximum transmission rate: 38400 baud	47

5.10 EXTEC16 Protocol

The EX TEC protocol is the standard protocol for all terminals. It is also used to load projects, even if a different protocol establishes the link to the control computer. This protocol is set in the setup. A detailed description of the protocol commands can be found on page 56.

Data direction: controller -> terminal

The controller sends either the direct ASCII codes of the characters that must be displayed or command sequences, all of which are triggered by ESC (hexadecimal: 1Bh). The length of the sequence varies with the command.

ASCII:	"1"	"2"	"3"	ESC	"["	";"	"H"	"4"	"5"	"6"	LF
HEX:	31h	32h	33h	1Bh	5Bh	3Bh	48h	34h	35h	36h	0Ah
	Character codes			Command sequence (cursor home)				Character codes			Control character (line feed)

In the example above, three characters are displayed initially ("123"). The cursor is then positioned in the top left-hand corner of a window and three more characters are displayed ("456"). Finally, the cursor is positioned to the beginning of the next line with the LF control character.
Some commands return a response. In addition, the buffer overflow monitor must be visible when characters are sent to the terminal (see below).

Data direction: terminal -> controller

The terminal sends either the codes for the pressed keys or the responses to the command sequences that were sent to it previously. These responses always start with the STX control character (hexadecimal: 02h) and end with the ETX control character (hexadecimal: 03h). The only exception to this rule is the response ACK (hexadecimal: 06h), which is sent alone in answer to the "auto-acknowledge" command. In addition, the XON (hexadecimal: 11h) and XOFF (hexadecimal: 13h) control characters can be sent if the buffer comes close to overflowing.

ASCII:	F1	"9"	"5"	"3"	"."	"1"	LF
HEX:	80h	39h	35h	33h	2Eh	31h	0Ah

In the above example, the user pressed function key F1 on the terminal, then entered the number "953.1" and confirmed it with ENTER, which would normally trigger an LF control character. You can find an overview of the key codes under **Key Codes in the EX TEC Protocol** on page 74.

The response frame for command sequences has the following format:

STX	Identifier	No. of bytes
02h		

Data bytes			

ETX
03h

The identifier byte determines the type of response. The byte with the number determines how many response bytes are sent before the end character ETX.

Identifier	**Description**	**See page**
D	Response to data block access	60

General features of the EX TEC protocol

- Some of the commands are directly compatible with the ANSI / VT100 standard.

- The terminal has an input buffer of 16 KB. The control computer can thus also send long command sequences to the terminal in quick succession, even though the terminal may not be capable of processing these commands at the same speed.
- The terminal sends a warning to the control computer prior to a buffer overflow in the form of the XOFF control character. If the control computer receives this character, it should not send any more characters until the terminal outputs XON to enable data transfer again. Please note that the control characters XON/XOFF appear only outside of command responses. Inside these frames there may be any binary data. These charcters must not be treated as control characters.
- Since the commands are processed in the order in which they are received, the command response times may be longer if the buffer is full.
- Commands to the terminal are not restricted by any character timeout; the terminal waits until the number of bytes specified in the command has been received.

5.11 EPCA Programming System

The EPCA programming system allows you to program the terminals "freely". This statement is actually a contradiction in itself, since a terminal is by definition only supposed to act as an interface between the user, the controller and possibly a peripheral, and never as a stand-alone unit with functions that can be defined by the project designer.

Practice has shown, however, that it would often be very useful to be able to program the units freely would often be very useful:

- Keys or shortcuts could be assigned any functions, either to relieve the load on the control computer or to reduce the response times.
- Variable values could be used for calculations and the results output immediately without having to make a detour via the controller.
- A stand-alone, "intelligent" unit would be created, something that is frequently needed for mobile systems, for instance, or for simple dosing computers.
- The functionality could be expanded and customized, since even the broadest spectrum of terminal functions can never satisfy the needs of every single user.
- Safer systems could be designed by integrating emergency shutdown devices or outputting additional warnings locally on the terminal, even if communication is interrupted.

All this is possible with the EPCA programming system.
EPCA can:

- React to keyboard inputs
- Calculate with different data types
- React to preset limits
- Control switching outputs
- Use various timer functions (measure times, react to elapsed times)
- Use the SER1 control interface (you could implement your own protocol)
- Generate any output on the display (open window, text output, graphic output)

Considerable importance was attached to two basic goals when EPCA was developed: **simplicity** and **security**. The system needed to be easy to handle and quick to master, so that its objective advantages were not completely outweighed during the project design phase by higher costs. In addition, secure and reliable processing of programs was vital, particularly in the industrial sector.

Measures to improve simplicity:

- The most common programming language of all is used, namely C
- The multitasking system permits several functions to be processed more or less simultaneously. This is a significant advantage when it comes to quick creation and easy maintenance of programs.

The system has been deliberately kept straightforward and there is no need to learn a large number of system commands in order to be able to switch the individual tasks.

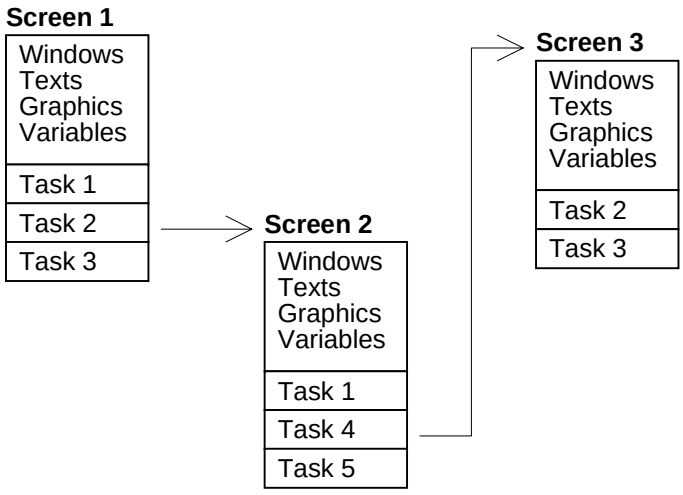
- All the data required for the EPCA programs is supplied preprocessed and displayed in a special data area. In addition, you can access the data block directly.
- The programming environment is fully integrated into the TERMEX PRO project design software.
- An increasing number of functions are already integrated in TERMEX PRO. Only a small amount of information still needs to be entered to enable TERMEX PRO to create tasks automatically.
- The time required for programming has been significantly reduced thanks to a number of library functions included with the software.
- Separate functions can be assigned to each screen. Each screen has its own list of tasks that are currently executing. Individual tasks can be added or removed easily without having to modify the remaining ones.

Measures to improve security:

- The EPCA programming system is subordinate to the firmware of the terminal. It is thus impossible for the firmware data areas to be interfered with by EPCA tasks.
- The tasks are absolutely separate from one another. Even a task that has been programmed incorrectly cannot sabotage others that are working normally. Local data areas are strictly local; a function cannot access the data areas of other functions.
- There are no interrupts (possible security risk) on the EPCA level.
- There is an integrated memory management functionality which automatically releases memory that is no longer needed.

How to create programs:

1. The first step is to determine the functionality you want to assign to a screen. All tasks which must be executed on a screen are called by the screen - not the other way round. When a new screen is called (either by a task or by an external controller), all active tasks are terminated and the new tasks are started automatically as soon as the new screen has been built. This results in a very clear structure, which also makes it easier to describe the behavior of a screen.



2. Next, you must determine how the screen functions can be distributed between the individual tasks. Since it is possible to execute a large number of tasks simultaneously, it is a good idea to split the functionality into several small tasks. You should take advantage of this option, because splitting the functions up in this way can often make it much easier to create the program as a whole. Key queries, for example, can be implemented very easily with one task per key. If the same key query is also needed on a different screen in conjunction with other keys, the task can be used again and combined with other key queries.

Detailed information about programming with EPCA can be found in the separate documentation.

5.12 Siemens S5 programmer interface / AS511

If you select the **AS511** operating mode under **Protocols** / SERIAL PORTS in the **TERMINAL SETUP** menu, the terminal is operated in PLC mode. You can use the **DB Number:** parameter in the **PROTOCOLS MENU** to select the required communication data block and **PLC Type:** to specify your Siemens programmable controller (PLC).
In this mode the terminal accesses the PLC data block which is specified in the **DB Number:** parameter directly via the programmer interface of the PLC, in other words it acts as the communication master. The terminal saves the appropriate changes (e.g. keystrokes) in this data block at the specified locations (refer to) and automatically retrieves the data (e.g. a character string starting at DW26 of the data block or a variable value starting at DW78) which is made available in the PLC data block by the user program. This process normally takes place at least once per second.
The selected communication data block must be created in the PLC with a size of 256 data words (L KF +256, E DB XY) in (OB20), OB21 and OB22. In addition, the EX TEC function block **FB 215 INITPGSS** must be integrated into these restart OBs and parameterized with the required communication data block XY. The selected communication data block XY is filled with 00h in this initialization block and a reset command is sent to the terminal.
Example with OB21:

```
                :                :
                :L      KF       +256
                :E      DB       87          (Create DB87 with a length of 256 data words)
                :
                :SPA    FB       215         (Initialize DB87 with 00h and trigger a terminal reset)
        NAME    :INITPGSS
        DB#A    :       KF       +87
                        :
```

The terminal now communicates with the PLC via the serial control port SER1 and the EX TEC mains buffer stage with the SK-LWL data interface. The 20 mA current loop is used for the bidirectional connection to the PLC.
For a description of how to install the required connecting cable, see [TERMEX].

5.13 Siemens S5 via the 3964R / RK512 Protocol

If you select the **Siemens S5 3964R** operating mode under **Protocols** in the **TERMINAL SETUP** menu, the terminal is operated in PLC mode. You can use the **DB Number:** parameter to select the required communication data block.
In this mode the terminal accesses the PLC data block which is specified in the **DB Number:** parameter directly, either via a communication module (e.g. CP524, CP525, CP544) or via the second serial port of a CPU (e.g. CPU 928), in other words it acts as the communication master. The terminal saves the appropriate changes (e.g. keystrokes) in this data block at the specified locations (refer to *Data Area* on page 28 ff.) and automatically retrieves the data (e.g. a character string starting at DW26 of the data block or a variable value starting at DW78) which is made available in the PLC data block by the user program. This process is repeated by the terminal automatically several times per second.
The selected communication data block must be created in the PLC with a size of 256 data words (L KF +256, E DB XY) in (OB20), OB21 and OB22. In addition, the EX TEC function block **FB 215 INIT3964** must be integrated into these restart OBs and parameterized with the required transfer block (DB#A), a flag byte for outputting error states of the communication module and the module address (BADR). The selected communication data block XY is filled with 00h in this initialization block and a reset command is sent to the terminal.
Example with OB21:

```
        :
        :L      KF      +256
        :E      DB      65          (Create DB65 with a length of 256 data words)
        :
        :SPA    FB      215         (Initialize DB87 with 00h and trigger a terminal reset)
NAME    :INIT3964
BADR    :       KF      +0
ERR     :       MB      200
DB#A    :       KF      +65
                :
```

The EX TEC function block **FB 214 CTRL3964** which is also required should be integrated into OB1, for example, and must be processed at least once per PLC cycle. If the PLC user programs are relatively large and the cycle times are in the region of 100 ms or more, FB 214 CTRL3964 should be called several times per PLC cycle. FB 214 CTRL3964 contains both a SEND ALL command and a RECEIVE ALL command, and safeguards communication between the terminal and the PLC via the communication module. Possible error states of the communication module are output in ANZW.
Example with OB1:

```
                :
        :SPA    FB      214
NAME    :CTRL3964
ANZW    :       MW      80
        :
```

A memory module (EPROM) is necessary to operate the interface on the CP524/525.
The following parameters must be selected for the module which is used with the Siemens parameterization software:

Procedure:	3964R
Interpreter:	RK512
Procedure parameters:	9600 baud, character length 8, 1 stop bit, even parity, low priority
Job list:	Job number 1
Job type:	SEND
Source / destination address:	Any
DB number:	Any
Coordination flag:	Any

Note: If data words higher than DW255 are used in the data area, this area is mapped in a second data block (refer to Data Area on page 28).

5.14 Siemens S7 via Profibus DP

Hardware requirements:

- CPU S7 (e.g. CPU S7 - 315 -2 -DP) — Siemens 6ES7 315-2AF01-0AB0
- 2x DP Profibus connector — Siemens 6ES7 972-0BA11-0XA0
- RS232 cable — Siemens 6ES7 901-1BF00-0XA0
- PC-MPI adapter — Siemens 6ES7 972-0CA20-0XA0
- Profibus cable
- EX TEC Profibus interface SK-PROFIBUS-DP-SPI3-HS (also referred to as SPI-3)
- SPI 3-ENT-DC interface cable (refer to technical manual for cable specification)
- EX TEC mains buffer stage ENT-DC -2.0-X-X HS
- Ex-i supply cable
- TERMEX 7xx terminal

- PC or programmer for configuring, parameterizing and programming

Software requirements:

- Step7 mini for 300 Siemens 6ES7 810-3BC02-0YX0
- Firmware module for TERMEX 7xx
- SPI 3 GSD (device data) (supplied on diskette)

The hardware configuration:

The TERMEX 7xx must be set to the 3964R / RK512 protocol in the setup
(refer to Readme750_d file **(*)** on the enclosed diskette or CD).
The SK-LWL is connected to the SPI-3 by means of a special cable (SKLWL - SPI-3).
The SPI-3 is connected to the Profibus by means of a standard bus connector. The SPI-3 needs a 24 V power supply.

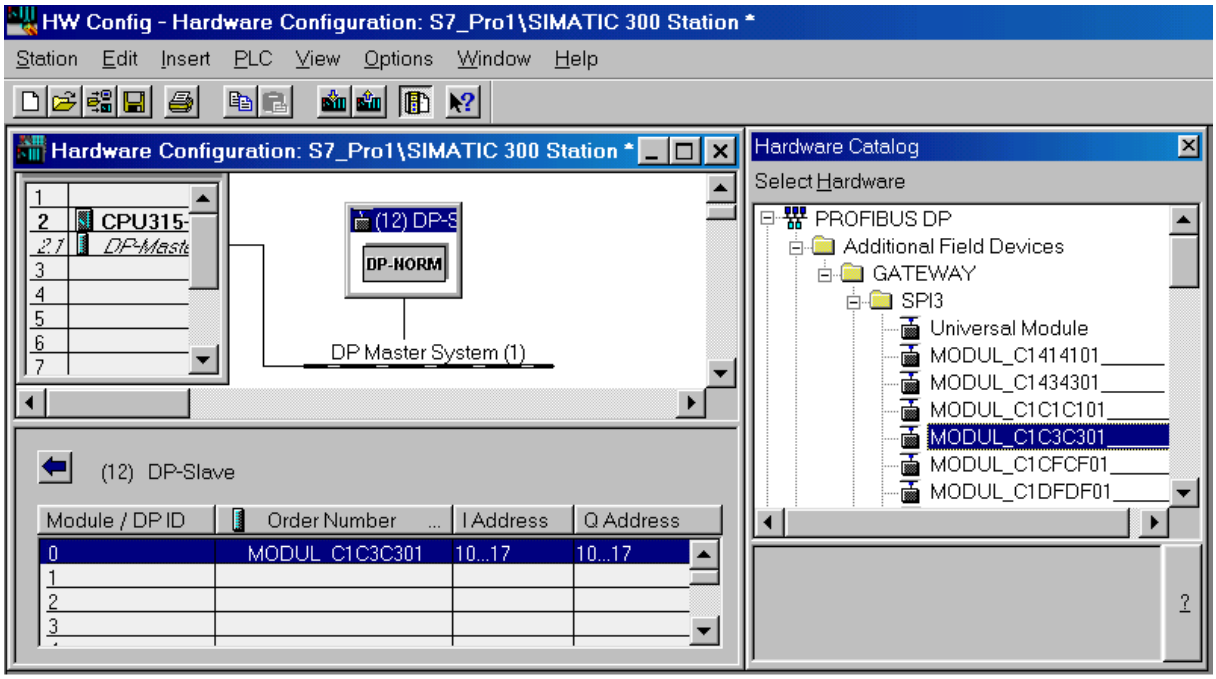
The communication principle:

The TERMEX 7xx is the communication master. It uses the 3964R protocol to communicate with the SPI-3, which forwards requests to the S7 via the Profibus.
These requests are processed by the S7 in a cyclically called function block (included with the firmware) and answered over the Profibus.
The SPI-3 converts the data packets back to the 3964R protocol for the TERMEX 7xx.

How to configure the system:

- Install Step7.
- Open the Explorer and copy the device master file Thdp0091.gsd from the enclosed diskette to the folder called Siemens\Step7\S7data\Gsd.
- Restart Step7 and create a new project with the Step7 Assistant. Use the mouse to select the station and the CPU.
- Set the **hardware configuration** in the Hardware Editor ***of STEP7*** (click on the Simatic3xx/4xx station, then double click on the "Hardware" icon): select *Options > Update GSD file* to import the GSD file via the serial port. Go to the folder called SPI3 in the hardware catalog in the Profibus DP\Weitere FELDGERÄTE \GATEWAY folder, then drag it to the bus bar and drop it there as the DP slave. Now open the SPI3 folder, select the C1C3C301 module (to send and receive eight bytes), drag it to the rack (in the bottom window) of the DP slave and drop it there.

Please note: To accelerate the communication choose the slave module Modul_C1CFCF01 (to send and receive 32 bytes).



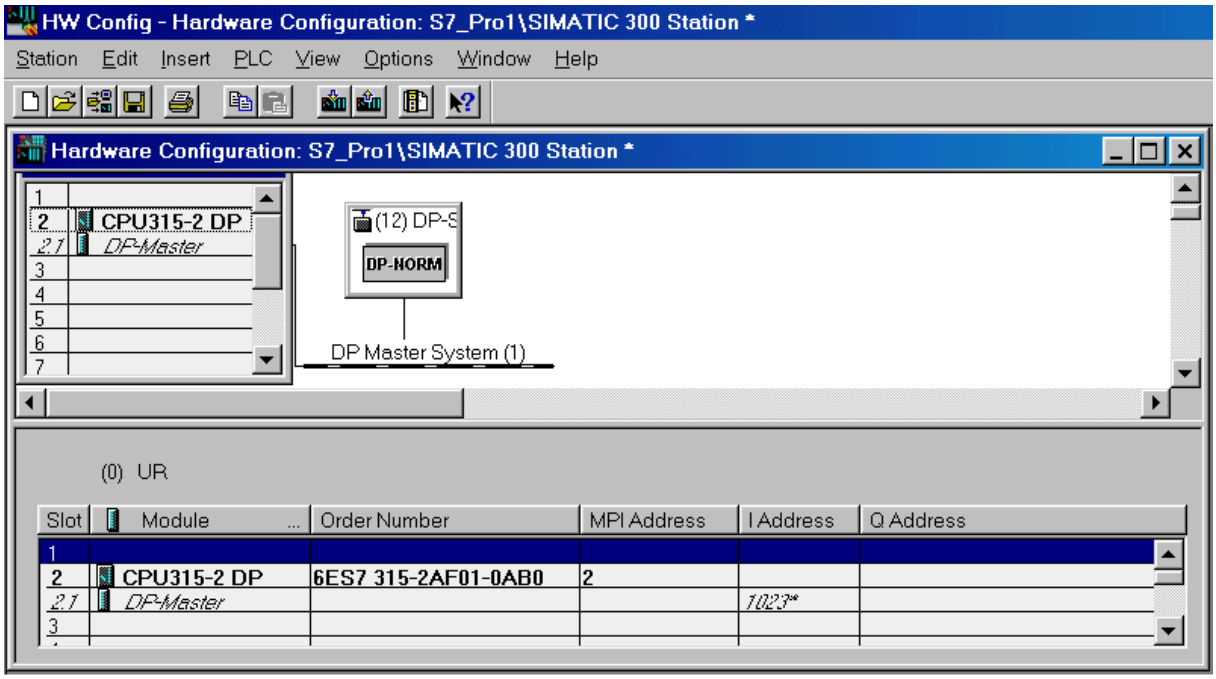
Select the 3964R protocol in the terminal setup menu ***on the TERMEX 7xx*** (refer to Readme750_d file [(*)]). You can set the data block required for communication here. If you leave the default value for the DB set to 2, you will be able to visualize the data exchange with the example supplied on the diskette.

- To **parameterize the hardware** you must first set the MPI address of the CPU (refer to Readme750_d file [(*)]). If you only have one CPU, the default value of the MPI address is 2 (otherwise the addresses are allocated in ascending order). The next step is to *parameterize* the Profibus of the *DP master*. Double click on the DP master and enter the address 2 (recommended). You can also set or display the following Profibus properties here:

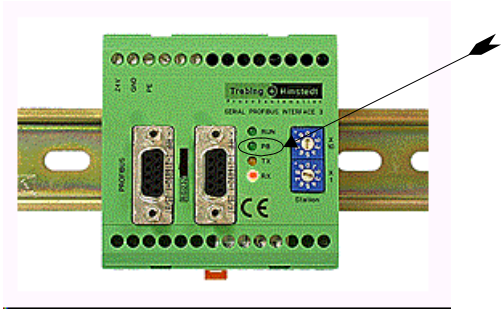
  Transmission rate (e.g. 12 Mbit/s),
  Network configuration (number of bus devices),
  Lines (number of repeaters, for example).

  By double clicking on the *DP slave* you can *allocate* a Profibus address for this slave in the General / Profibus menu. ***This address must be the same as the one which is set on the serial interface (SPI module).*** We recommend that you set the address to 3 at both ends. The following settings are required for the C1C3C301 module:

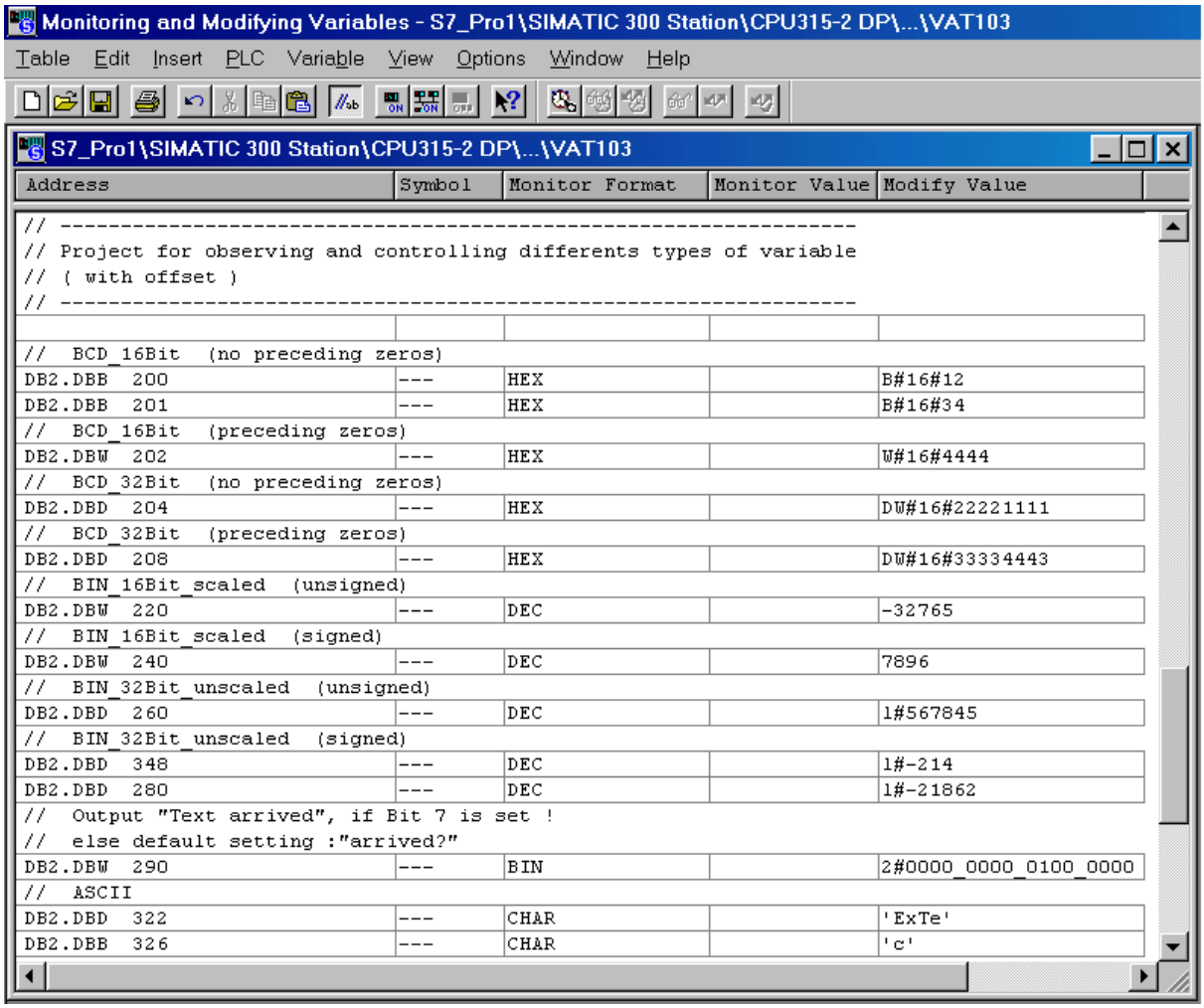
Transmission rate:	9600 baud
Handshake:	No
Xon character:	0
Xoff character:	0
Parity:	Even
Character frame:	8 bits
Delay time (*10 ms):	200
Transmission mode:	RK512R (with checksum)
Priority:	Low



- After the hardware has been configured, the configuration must be saved and compiled (select Station>Save and Compile in the menu bar). If the hardware configuration is downloaded to the CPU at this point (with *Target System>Download to Module* in the menu bar), the green PB LED on the SPI3 module must remain permanently lit.



- If you set a *communication data block* other than DB2 on the TERMEX 7xx (as in our example), you must *configure* this block. Take data block DB2 as a model and configure your communication data block in the same way.
- If you use communication data block DB2, you can visualize the data exchange via the initialized VAT103 block. Double click on VAT103, set up a connection to the directly connected CPU (either using the button provided for this purpose or with *Target System > Connect To > Direct* in the menu on the function bar) and click on the Observe Variable button. Set the trigger condition for observing variables to Permanently in the *Variable > Trigger* menu. You can view the settings in the variable area on the TERMEX 7xx immediately and permanently, starting at DB2.DBB200, in the first of the projects we have prepared for you (Variable test).



- If you are using an S7-300, you can ignore the message "Cannot copy organization block OB101" which appears when you attempt to copy the blocks from the PC to the PLC. This block is the organization block for restarting an S7-400 module.
- The enclosed diskette or CD contains two sample TERMEX PRO projects:
  Tpro750 and Tlist750.
  The Tpro750 project demonstrates the interaction of the different types of data variable in the TERMEX 750 with variables in the S7 world. The Tlist750 project shows how messages are displayed on the TERMEX 750 if a specific bit is set by the PLC.
  If you want to observe data communication when different data words (different variables) are addressed, load the Tpro750 project onto your TERMEX 750 (refer to Readme750_d file **(**)**).

- To observe data exchange or communication:
  TERMEX 7xx -> PLC.
  Enter numbers in the variable fields. You can copy these values to the status value fields, starting at DB2.DBB200 (variable area of the TERMEX 7xx), by pressing the ENTER key.

  PLC -> TERMEX 7xx
  Enter numbers (preferably in hex format) in the control value fields, starting at DB2.DBb200, then copy them to the appropriate variable fields by clicking on *Control Value-Activate* in the menu bar. Please note the position of the switch: it must be set to Run-P or Stop, so that values can be forced.

- **Please note:** If you want to *allocate* your ***own*** *DB-Q or DB-Z* or if you want to *allocate ANZW,* you must take account of how the RK512R message frame is evaluated (refer to page 17 of the

enclosed SPI3 manual). You can use the blocks which are supplied by us as a reference for creating new DBs and OBs.

Note on addressing:

The **S7** world is ***byte-oriented***, not *word-oriented* like the S5 world, in other words if you address data word DW78, you actually select byte 156 and byte 157 rather than the high and low bytes of data word DW78 (in contrast with the S5).

Note:

- The speed of communication between the TERMEX 750 and the PLC depends to a large extent on the PLC cycle time. If the PLC cycle time is greater than 100 ms, it is advisable to accommodate the blocks that are used for communication (SFC14, SFC15 and FB103) in a timer OB (e.g. OB35) and to set the trigger instant to roughly 10 ms.
- If data words higher than DW255 are used in the data area, this area is mapped in a second data block (refer to Data Area on page 28).

5.15 Siemens S7 via MPI

- The **MPI adapter** ist connected to the **MPI bus** of a Siemens S7 PLC by the attached cable. The power is normally supplied through this cable too. A separate power supply is only necessary under special conditions (see the short manual “Helmholz”). The power LED of the MPI adapter must remain permanently lit.
- The **MPI adapter** must be connected to the EXTEC supply unit SK-LWL by means of the standard SK-LWL PC cable (refer to [TERMEX]).
- Set the **protocol** to “3964R/RK51” in SERIAL PORTS MENU of the terminal.
- Set the number of the **communication data block** in the PROTOCOLS MENU.
- Set the communication parameters to **9600 Baud**, even Parity, 8 data bits in the SERIAL PORTS MENU. The higher communication speed of 19200 Baud may only be used under some special conditions:
  Use of the original EXTEC SKLWL-PC cable or a comparable cable with low inductivity and low capacitance.
  Short cable length between TERMEX and SK-LWL.
  Should the automatic baud rate recognition of the MPI adapter fail the speed should be switched to 9600 Baud.
- The **communication** works without any special drivers in the S7 PLC. The Connect LED must blink.
- The COM LED of the teminal will be switched on in case of **communication errors**. The MPI adapter shows its status via 3 LEDs (see the short manual “Helmholz”).
- To connect more than one terminal or additional devices to the MPI bus, every terminal needs its own MPI adapter (see the short manual “Helmholz”).

5.16 MODBUS Protocol

In contrast with the other PLC protocols, the terminal plays a passive role here and the control computer accesses the data area of the terminal independently using the MODBUS protocol. The terminal operates as a slave; the slave address can be selected in the setup.

Framing: 8 bits (RTU)
Slave address: 1...32 (selected in the setup)

The control computer sends a command (*message*) with a defined function code to the terminal, which then sends a *response* back to the control computer.
Only the slave whose address is specified in the message (as the first byte) responds.
The end-of-message detection function is based on a time condition: if no more characters are sent within 3.5 times the character transmission time corresponding to the selected baud rate, the addressee considers the message to be terminated, attempts to interpret it and regards the next character that is received as the slave address of the next message.

This principle results in the following timeout times:

Baud rate	**Timeout time**
300 baud	128 ms
1200 baud	32 ms
2400 baud	16 ms
4800 baud	8.0 ms
9600 baud	4.0 ms
19200 baud	2.0 ms
38400 baud	1.0 ms

When a MODBUS driver is developed, it is therefore vital for the interval between two characters in a transmitted message frame to be less than the timeout time. You should be careful not to venture too close to this limit, as otherwise even minimal timing shifts can lead to communication problems.

Available MODBUS functions:

Function code	Name	Description
1	Read coils	Individual bits or variable-length bit strings are read, starting at a specified bit address
3	Read output registers	Individual or multiple data words are read, starting at a specified data word address
4	Read input registers	Individual or multiple data words are read, starting at a specified data word address
5	Force coil	Exactly one bit is written
6	Load register	Exactly one data word is written
8	Loopback test	Test function for the communication system
15	Force multiple coils	Individual bits or variable-length bit strings are read, starting at a specified bit address
16	Load multiple registers	Individual or multiple data words are read, starting at a specified data word address

Supported MODBUS error messages:

Exception code	Name	Description
1	Illegal function	The received command (message) includes a function code which is not supported by the terminal
2	Illegal data address	An attempt has been made to access a bit or data word address which is outside the data block

The response which is returned by the terminal in case of an error has the following format:

Slave address	80h + function code	Exception code	Check-sum HI	Check-sum LO

The function code of the request is sent in the second byte with the highest bit set. This is equivalent to adding 80h. The third byte contains the exception code for the error message.

The MODBUS functions in detail

Message:

Slave address	1	Coil address HI	Coil address LO	Coil number HI	Coil number LO	Checksum HI	Checksum LO

Response:

Slave address	1	Byte number	Coils 7...0	Coils 15...8	...	Checksum HI	Checksum LO

Coil address (HI * 256 + LO)
Bit address at which the system must start reading. This address is derived as follows:

DW0	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DW1	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DW2	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
...																

Coil number (HI * 256 + LO)
Number of bits that must be read.

Byte number
Number of subsequent bytes containing coils.

Coils
Bits which are read from the data block. As many bytes are transmitted as are necessary for the required number of bits. If, for example, 14 bits need to be transmitted starting at bit address 13, the format of the transmitted bytes is as follows:

Bit no.	7	6	5	4	3	2	1	0
1st byte	20	19	18	17	16	15	14	13
2nd byte	-	-	26	25	24	23	22	21

Permitted value range: Coil address + coil number ≤ 8192

Message:

Slave address	3/4	Reg address HI	Reg address LO	Reg number HI	Reg number LO	Checksum HI	Checksum LO

Response:

Slave address	3/4	Byte number	Reg 0 HI	Reg 0 LO	...	Checksum HI	Checksum LO

Reg address (HI * 256 + LO)
Data word address at which the system must start reading.

Reg number (HI * 256 + LO)
Number of data words that must be read.

Byte number
Number of subsequent bytes containing data words.

Reg
Data words (high byte and low byte) read from the data block.

Permitted value range: Reg address + reg number ≤ 512

Message:

Slave address	5	Coil address HI	Coil address LO	Coil ON OFF indicator 0xFF (1) 0x00 (0)	Coil data 0x00	Checksum HI	Checksum LO

Response:

Slave address	5	Coil address HI	Coil address LO	0xFF 0x00	Coil data 0x00	Checksum HI	Checksum LO

Coil address (HI * 256 + LO)
Bit address at which the system must start writing. This address is derived as follows:

DW0	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DW1	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DW2	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
...																

Reg data
Value of the data word (high byte and low byte) that must be written.

Permitted value range: Coil address < 8192

Message:

Slave address	6	Reg address HI	Reg address LO	Reg data HI	Reg data LO	Checksum HI	Checksum LO

Response:

Slave address	6	Reg address HI	Reg address LO	Reg data HI	Reg data LO	Checksum HI	Checksum LO

Reg address (HI * 256 + LO)
Data word address that must be written.

Reg data
Value of the data word (high byte and low byte) that must be written.

Permitted value range: Reg address < 512

Message:

Slave address	8	Data diag code HI 0x00	Data diag code LO 0x00	Test data	Test data	Checksum HI	Checksum LO

Response:

Slave address	8	Data diag code Hi	Data diag code LO	Test data	Test data	Checksum HI	Checksum LO

Data diag code high, data diag code low
Diagnostic code (subfunction of function 8) that is to be used to test the communication system. The diagnostic code "Return Query Data" (0x00 0x00) is supported.

Test data
If the diagnostic code 0x00 0x00 is used, the data which is sent will be returned to the master unchanged. The data length is arbitrary within the MODBUS limits.

Message:

Slave address	15	Coil addr HI	Coil addr LO	Coil num HI	Coil num LO	Byte number	Coils 7...0	...	Check-sum HI	Check-sum LO

Response:

Slave address	15	Coil addr HI	Coil addr LO	Coil num HI	Coil num LO	Check-sum HI	Check-sum LO

Coil address (HI * 256 + LO)
Bit address at which the system must start writing. This address is derived as follows:

DW0	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DW1	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DW2	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
...																

Coil number (HI * 256 + LO)
Number of bits that must be written.

Byte number
Number of subsequent bytes containing coils.

Coils
Bits which are written in the data block. As many bytes are transmitted as are necessary for the required number of bits. If, for example, 14 bits need to be written starting at bit address 13, the format of the transmitted bytes is as follows:

Bit no.	7	6	5	4	3	2	1	0
1st byte	20	19	18	17	16	15	14	13
2nd byte	-	-	26	25	24	23	22	21

Permitted value range: Coil address + coil number ≤ 8192

Message:

Slave address	16	Reg addr HI	Reg addr LO	Reg num HI	Reg num LO	Byte number	Reg 0 HI	Reg 0 LO	...	Check-sum HI	Check-sum LO

Response:

Slave address	16	Reg addr HI	Reg addr LO	Reg num HI	Reg num LO	Check-sum HI	Check-sum LO

Reg address (HI * 256 + LO)
Data word address at which the system must start writing.

Reg number (HI * 256 + LO)
Number of data words that must be written.

Byte number
Number of subsequent bytes containing data words.

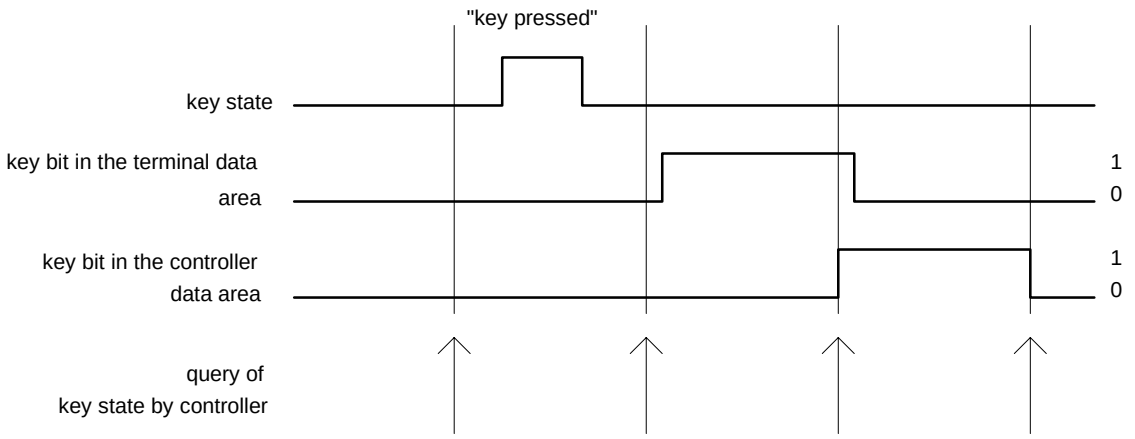
Reg
Data words (high byte and low byte) written in the data block.

Permitted value range: Reg address + reg number ≤ 512

Key states

The data words with the key bits must be read at regular intervals (approximately once or twice per second) in order to determine the key states in the control computer (functions 1, 3 or 4). The key states are updated after each read access to these data words. This means that any changes to the

key states are mapped in the key bits. A whole series of key state changes can be buffered in the terminal in this way.
The following example explains how this works:



It is clear from this that even a short keystroke which occurs between two queries is still detected and extended to one cycle time of the query (state spreading).
It is essential to ensure that all data words containing relevant key bits are read at the same time, because all key states are updated even if only some of the data words are read. If you ignore this warning, you may lose information!

LED status

LEDs are updated with each write access to one of the corresponding data words.

Message frames

The message display is updated with each write access to one of the corresponding data words. It is advisable to write all the message frames which are used with a single access.

Variables

Actual variables and non-active set variables are updated with each write access to the variable area. They are also updated with a constant time pulse according to the setting in the setup. Set variables are updated in the data block as soon as they are entered on the terminal and can be read at any time as and when required. Set variables can also be overwritten by the control computer at any time.
It is advisable to group variables together in the variable area without any gaps, so that they can be updated with a single access (or only a few accesses).

5.17 Load Error Messages

Load error messages sometimes appear on the display of the terminal immediately after you load a project. In this case, the project will not be accepted. Press any key on the terminal in order to reset it.

------ Download Error(s) ------:

Error message	Description
Too much EPCA global data !	The loaded project needs more global memory for EPCA than the terminal can provide. You can try to reduce the amount of memory that is required by getting rid of some of your global variables or by defining smaller global arrays in the EPCA tasks.
Too much EPCA noinit data !	The loaded project needs more NoInit (resident) memory for EPCA than the terminal can provide. You can try to reduce the amount of memory that is required by getting rid of some of your global NoInit variables or by defining smaller NoInit arrays in the EPCA tasks.
Too much EPCA local data !	The loaded project needs more local and parameter memory for EPCA than the terminal can provide. You can try to reduce the amount of memory that is required by getting rid of some of your local variables, by defining smaller local arrays or by specifying fewer function parameters in the EPCA tasks.
Too many EPCA progs (>x) !	The loaded project contains more EPCA programs (functions) than the terminal is able to store.
Too many bitmaps (>x) !	The loaded project contains more bitmaps and character sets than the terminal is able to store.
Wrong checksum !	The checksum for the loaded project is incorrect. The data link between the PC and the terminal may be faulty. After checking the cables and possibly reducing the transmission rate, you should load the project again.
Too many blocks (>x) !	The loaded project contains more data blocks (screens, messages, bitmaps, programs, text lists) than the terminal is able to handle.
EPCA task number (>x) !	The maximum number of simultaneously active EPCA tasks is too high on at least one of the screens of the loaded project. If the number of tasks entered in the screen task list is higher than the maximum number, you will have to delete a few tasks.
EPCA stack depth (>x) !	The maximum EPCA function call depth is exceeded on at least one of the screens of the loaded project. You can reduce the call depth by combining calling and called functions.
Too many palettes (>x) !	The loaded project contains more colour palettes than the terminal is able to handle.
Too many variables (>x) !	The loaded project contains more variables than the terminal is able to handle. You can reduce the number by deleting some variables in your project.

5.18 Printer

A serial line printer can be hooked up to the SER2 port of the TERMEX 750 using a serial cable that is connected to the SK-LWL in the safe area. A four-wire optical fiber connection is necessary for this purpose between the TERMEX 750 and the SK-LWL, so that there are two free wires for connecting the printer.
The SER2 port must be configured for a *Printer* in the setup.

Configuring the printer port

The printer port can be specially configured for your particular printer. You can enter the control sequences in the setup dialog. These sequences consist of printable characters or hex codes that start with a $, in other words $0A for line feed, for example.
The following settings are possible:

	Setup option	**Description**
Printer initialization	prnt_InitSeq	This could be the command for resetting the printer settings (ESC @ = $1B$40), for instance, or a font specification. You could also enter a short preliminary text here.
Line feed	prnt_LinefeedSeq	The sequence entered here is used by the terminal to tell the printer to start printing a new line. Typical sequence: CR LF = $0D$0A.
Form feed	prnt_FormfeedSeq	The sequence entered here is used by the terminal to tell the printer to start printing a new page. Typical sequence: $0C.
End of printing	prnt_EndSeq	This sequence is printed at the end of a printout.

Print options

You can only print out lists with the TERMEX 750, not graphics or whole screens.

	Resolution	Description
Message lists	<Shift> <7> (at the selected set variable)	The message list for a selected MSGLIST set variable can be printed out. The complete list is printed - not just the part of it that is visible in the window.

5.19 Firmware Updates

The terminal firmware can be updated at any time. You can obtain the latest version in either of the following ways:

- By downloading it from the EX TEC website (www.extec.de). The file is called *t750_xxx.exe* (where xxx is the version number). This archive file unpacks automatically to reveal two other files called *t750_xxx.bin* and *t750_xxx.key*.
- By requesting the current software CD from EX TEC. The FIRMWARE folder contains the same two files: *t750_xxx.bin* and *t750_xxx.key*.

Note:

- All the information provided here is correct at the time of going to print. The distribution method and the update procedures are however subject to change at any time.
- The new firmware is always distributed together with a *readme* file, which may contain even more recent information about the update. Please study this special information carefully.
- All new firmware is developed with the intention of keeping it compatible with earlier projects and applications. We will point out any exceptions of which we are aware explicitly. Nevertheless, we cannot completely rule out the possibility that the terminal will behave illogically after an update. You should always retest both the terminal and the application after installing a firmware update. If you encounter any problems, please do not hesitate to contact EX TEC.
- We are constantly endeavoring to improve the firmware. New versions incorporate solutions to known problems, functional enhancements and features for improving performance. You will

normally need the latest firmware version in order to be able to make use of the new functionalities in the TERMEX PRO project design software.

- All new firmware which is distributed has been verified several times with the aid of checksums. It can thus be safely assumed that if the new firmware starts up successfully on your unit it has not been manipulated.

How to install an update:

1. Connect the TERMEX 750 to a serial port on your PC via the SK-LWL.
2. Set the TERMEX 750 to the EXTEC16 protocol.
3. Start the EX TEC loader of TERMEX PRO (*exloader.exe*).
4. Set the COM interface and the transmission rate of the TERMEX 750 ("Interface" and "Settings").
5. Open the file called *t750_xxx.bin* ("Project"). A file called *t750_xxx.key* should be contained in the same folder.
6. Download the firmware ("Download").
7. The TERMEX 750 triggers a reset and the display goes black. The download will take a few minutes. The LED on the SK-LWL blinks at regular intervals to indicate that the download is still in progress.
8. When the download has finished, the new firmware is started automatically. Any projects that are already loaded are still open and all valid settings remain stored.

If the download procedure is interrupted, you must switch off the TERMEX 750 briefly and then switch it back on again. In this case the display will remain black, because there is no valid version of the firmware in the unit.
You can then attempt to load it again straight away (starting with step 5).
If you are still unable to load the new firmware after several attempts, please contact EX TEC.

6 Reference

6.1 EXTEC16 Protocol Commands

The following syntax is used for all commands to the terminal:

ESC: Escape control character (code: 1Bh)
'X': Code for the ASCII character X (here: 58h)
x: Variable (1 byte, 0...255)
aah: Hexadecimal aa
... : More characters must follow

ASCII: ASCII codes
HEX: HEX codes

Only the commands listed in this section are allowed to be sent to the terminal. All other command sequences may lead to errors!

Summary (in logical order)

Description	Terminal command	ANSI/VT100-compatible	Page
Clear display	ESC ' * '		**59**
Clear display (pixels with specified color)	ESC ESC ' * ' col		
Enable internal keyboard	ESC ' " '		**59**
Disable internal keyboard	ESC ' # '		**59**
Hardware reset	ESC 'A' '2'		**59**
Ready query	ESC 'A' '5'		**59**
Display screen n	ESC 'G' nhigh nlow		**60**
Change project	ESC ESC 'G' projhigh projlow nhigh nlow		**60**
Display firmware version no.	ESC '?'		**60**
Write data words in DB	ESC 'y' ...		**60**
Read data words from DB	ESC 'z' ...		**61**
Set date and time	ESC 'T' yh y mo d dw h mi s		**62**
Open setup dialog	ESC '1'		**62**
Increase contrast	ESC '2' steps		**62**
Reduce contrast	ESC '3' steps		**62**
Disable key bits	ESC ESC '#' num km_0...km_{11}		**63**
Open text window	ESC ESC 'W' xhigh xlow yhigh ylow xmz ymz nhigh nlow colfront colback hhigh hlow		**63**
Open graphic window	ESC ESC 'X' x1high x1low y1high y1low x2high x2low y2high y2low colfront colback hhigh hlow		**64**
Close text window	ESC ESC 'w' hhigh hlow		**65**
Set window style	ESC ESC 'Q' q1 q2 hhigh hlow		**65**
Set text window style attributes	ESC ESC 'C' q1 q2		**67**
Reset text window style attributes	ESC ESC 'c' q1 q2		**68**
Set window border	ESC ESC 'N' bord nhigh nlow		**68**
Set window border size	ESC ESC 'O' lefthigh leftlow righthigh rightlow tophigh toplow bothigh botlow bindex		**69**
Set handle1 (text output)	ESC ESC 'H' '1' hhigh hlow		**69**
Set handle2 (input from keyboard)	ESC ESC 'H' '2' hhigh hlow		**69**
Set handle G (graphic output)	ESC ESC 'H' 'G' hhigh hlow		**70**
Activate window	ESC ESC 'A' hhigh hlow		**70**
Window invisible	ESC ESC 'a' hhigh hlow		**70**
Backspace, BS	8h	•	**70**
Line feed, LF	Ah	•	**71**
Carriage return, CR	Dh	•	**71**
Cursor up	ESC '[' (nn) 'A'	•	**71**
Cursor down	ESC '[' (nn) 'B'	•	**71**

Cursor forward	ESC '[' (nn) 'C'	•	**71**
Cursor backwards	ESC '[' (nn) 'D'	•	**72**
Cursor absolute	ESC '[' (yy) ';' (xx) 'H'	•	**72**
Delete current line	ESC 'Y'		**72**
Clear window	ESC ESC 'd' hhigh hlow		**72**
Set color of graphic output	ESC ESC '+' col		**72**
Set pixel	ESC ESC 'P' xhigh xlow yhigh ylow		**73**
Draw line	ESC ESC 'L' x1high x1low y1high y1low x2high x2low y2high y2low		**73**
Draw rectangle	ESC ESC 'S' x1high x1low y1high y1low x2high x2low y2high y2low		**73**
Draw bar	ESC ESC 'B' x1high x1low y1high y1low x2high x2low y2high y2low		**73**

Description of the commands

Clear display	ASCII: **ESC ' * '**
	HEX: 1Bh 2Ah

- All pixels on the display are deleted (white background).
- All currently open text and graphic windows are closed automatically.
- All displayed variables are logically closed.
- If a message is queued at the time the display is cleared, it is generated again afterwards.

Clear display (pixels with specified color)	ASCII: **ESC ESC ' * ' col**
	HEX: 1Bh 1Bh 2Ah ...

- All pixels on the display with the specified color are deleted.
- All currently open text and graphic windows are closed automatically.
- All displayed variables are logically closed.
- If a message is queued at the time the display is cleared, it is generated again afterwards.

Enable internal keyboard	ASCII: **ESC ' " '**
	HEX: 1Bh 22h

- Enables the keyboard again after it has been disabled with ESC ' # '.

Disable internal keyboard	ASCII: **ESC ' # '**
	HEX: 1Bh 23h

- Any keys pressed after this command will be completely ignored, in other words they are neither buffered nor output later on.
- This command applies to all processing actions in the data block, in the firmware and in EPCA.

Hardware reset	ASCII: **ESC 'A' '2'**
	HEX: 1Bh 41h 32h

- Triggers a hardware reset, which is equivalent to re-booting the terminal. This command can be used to access the terminal setup without interrupting the power supply.
- The reset is triggered immediately. A few seconds will elapse before the startup message appears, however. The startup message is then displayed for four seconds as default.
- If you expressly want to make sure that the unit is ready for operation again after a reset, you can use the **Ready query** command on page 59.

Ready query	ASCII: **ESC 'A' '5'**
	HEX: 1Bh 41h 35h

- Answered with ACK (code: 06h) via SER1 if the unit is ready.
- If no response is received to this command after several attempts, the reason is likely to be a hardware problem in connection with the terminal interface.

Display screen n	ASCII: **ESC 'G' nhigh nlow**
	HEX: 1Bh 47h ...

Parameter: n: Screen number (1...255)

- Screen *n* is displayed.
- If a screen with the number *n* does not exist on the terminal, the command is ignored.

Change project	ASCII: **ESC ESC 'G' projhigh projlow nhigh nlow**
	HEX: 1Bh 1Bh 47h ...

Parameters: proj: Project number (1...255)
n: Screen number (1...255)

- The system changes to the project with the project number proj and then opens screen n.
- If a project with the number proj does not exist, the command is ignored.
- If a screen with the number *n* does not exist on the terminal for the project with the number proj, a new screen is not opened. The new project is active, but the old screen is still displayed.

Display firmware version no.	ASCII: **ESC '?'**
	HEX: 1Bh 3Fh

- The terminal sends the formatted terminal name and version number to the serial interface SER1, followed by a carriage return (e.g.: "VS1..00" LF for Version VS1.00).

Write data words in DB	ASCII: **ESC 'y' .....**
	HEX: 1Bh 79h ....

- You can access the data block which is otherwise used by the PLC protocols via the EXTEC16 protocol. This command allows you to use variables, bar graphs and messages for this interface too.
- The only difference as compared with other PLC protocols is that the control computer must actively read and write data.

Writing data words

In the following example four data words are written starting at address 80h (decimal 128).

Command header		DW start address		Number of DWs		Data words to be written							
		High byte	Low byte	High byte	Low byte	H	L	H	L	H	L	H	L
ESC	**'y'**	**0h**	**80h**	**0h**	**4h**	**0h**	**1h**	**0h**	**2h**	**0h**	**3h**	**0h**	**4h**
1Bh	79h	0...255		1...256		DW 128		DW 129		DW 130		DW 131	

Response from the terminal when data words are written

Start of frame		Length	Com-mand descrip-tion	DW start address		Number of DWs		End of frame
				High byte	Low byte	High byte	Low byte	
STX	**'D'**	**num**	**'y'**	**0h**	**80h**	**0h**	**4h**	**ETX**
2h	44h	5h	79h					3h

Result of the above example in the data block

DW	Con-tents
...	
80h	0001h
81h	0002h
82h	0003h
83h	0004h
...	

Read data words from DB	ASCII: **ESC 'z' .....**
	HEX: 1Bh 7Ah ....

Reading data words

In the following example four data words are read starting at address 80h (decimal 128).

Command header		DW start address		Number of DWs	
		High byte	Low byte	High byte	Low byte
ESC	**'z'**	**0h**	**80h**	**0h**	**4h**
1Bh	7Ah	0...255		1...256	

Response from the terminal to the read request

Start of frame		Length	Com-mand descrip-tion	DW start address		Number of DWs		Data words to be read								End of frame
				High	Low	High	Low	H	L	H	L	H	L	H	L	
STX	**'D'**	**num**	**'z'**	**0h**	**80h**	**0h**	**4h**	**0h**	**1h**	**0h**	**2h**	**0h**	**3h**	**0h**	**4h**	**ETX**
2h	44h		7Ah					DW128		DW129		DW130		DW131		3h

Set date and time	ASCII: **ESC 'T' yh y mo d dw h mi s**
	HEX: 1Bh 54h ....

Parameters:

- yh: Millenium in the four high data bits, century in the four low data bits (BCD format).
- y: Decade in the four high data bits, year in the four low data bits (BCD format).
- mo: First month digit in the four high data bits, second month digit in the four low data bits (BCD format).
- d: First day digit in the four high data bits, second day digit in the four low data bits (BCD format).
- dw: Day of the week (Sun - Sat) as a number from 0 to 6.
- h: First hours digit in the four high data bits, second hours digit in the four low data bits (BCD format). The hour is always specified using the 24 h format.
- mi: First minutes digit in the four high data bits, second minutes digit in the four low data bits (BCD format).
- s: First seconds digit in the four high data bits, second seconds digit in the four low data bits (BCD format).

Example:
To set the date and time to Tuesday, May 21st 1996, 22.00 hours, 15 minutes, 37 seconds:

Start of command	Com-mand code	Date century	Date year	Date month	Date day of month	Date weekday	Time hours	Time minutes	Time seconds
ESC	T	0x19	0x96	0x05	0x21	0x02	0x22	0x15	0x37

Open setup dialog	ASCII: **ESC '1'**
	HEX: 1Bh 31h

- The terminal exits the EXTEC16 protocol and opens the setup dialog via the serial interface.
- You can find further information about entering settings in the setup dialog on page 11.
- When you close the dialog again, the system returns to the EXTEC16 protocol.

Increase contrast	ASCII: **ESC '2' steps**
	HEX: 1Bh 32h ....

- You can adjust the contrast in 100 steps.
- This command increases the contrast by *steps*.
- You can increase the contrast slightly, for example, with steps=5 or considerably with steps=10.
- The new contrast setting is permanently stored.

Warning: The number of memory cycles is limited (to 100000 according to the manufacturer's information). The memory may be damaged if you use this command too often. You may reach this figure very quickly, for instance, if you are using program control (incorrectly).

Reduce contrast	ASCII: **ESC '3' steps**
	HEX: 1Bh 33h ....

- You can adjust the contrast in 100 steps.
- This command reduces the contrast by *steps*.
- You can reduce the contrast slightly, for example, with steps=5 or considerably with steps=10.

- The new contrast setting is permanently stored.

Warning: The number of memory cycles is limited (to 100000 according to the manufacturer's information). The memory may be damaged if you use this command too often. You may reach this figure very quickly, for instance, if you are using program control (incorrectly).

Disable key bits	ASCII: **ESC ESC '#' num km_0...km_{11}** HEX: 1Bh 1Bh 23h ....

Parameters:
num: Number of subsequent mask bytes (must be 12)
km_x: Mask bytes (bit=1 key enabled; bit=0 key disabled)

Assignment:

km0	F16	F15	F14	F13	F12	F11	F10	F9	F8	F7	F6	F5	F4	F3	F2	F1	**km1**	(DW 4)
km2							S10	S9	S8	S7	S6	S5	S4	S3	S2	S1	**km3**	(DW 5)
km4							9	8	7	6	5	4	4	2	1	0	**km5**	(DW 6)
km6			.	PLS	MNS	ENT	ALT	SHT	CLR	INF	CRI	CLE	CDN	CUP	BOT	TOP	**km7**	(DW 7)
km8	I15	I14	I13	I12	I11	I10	I9	I8	I7	I6	I5	I4	I3	I2	I1	I0	**km9**	(DW 8)
km10																	**km11**	(DW 9)

This command can be used to disable the status bits of the keys which are mapped in the data block. If a key is disabled by a "0" in the mask, then a "0" is mapped in the data block and not the true key state.
The functionality which is available to the user can be restricted explicitly in this way on certain screens. This considerably simplifies key processing by the controlling computer.
Another possible application might be to explicitly mask out number keys, which are normally only used to enter variables. Since masking a key does not result in any change in its state, a large number of message frames no longer needs to be sent to the PLC. This helps to speed up communication and shorten key response times, especially with relatively slow protocols.
This mask also applies to key processing in EPCA.
It does not apply to internal key processing in the firmware, however.

Open text window	ASCII: **ESC ESC 'W' xhigh xlow** **yhigh ylow** **xmz ymz** **nhigh nlow** **colfront colback** **hhigh hlow** HEX: 1Bh 1Bh 57h ....

Parameters:
x, y: Screen coordinates of the top left-hand corner of the window
xmz: Line length (in characters) in the text window
ymz: Number of lines in the text window
n: Character set in the text window (1...4 for integrated character sets)
colfront: Character color
colback: Background color
h: Handle number (individual window identification, 1...255)
1...239: Freely assignable window numbers
240...249: Recommended windows for MULTI messages
250...255: Reserved windows (e.g. for weight outputs)

- A window with *xmz* characters/line and *ymz* lines is opened at the transferred coordinates (*x, y*). The size of the window is automatically derived from the selected character set *n*, which is used exclusively in this window.
- If the coordinates (*x*, *y*) are outside the display, the open window command will be ignored.
- If *xmz* or *ymz* is so large that the right or bottom edge of the window extends beyond the display (coordinates out of range), the line length *xmz* and/or the number of lines *ymz* are reduced automatically so that the whole window fits inside the display.
- The window is opened in the foreground and is therefore active. If the new window overlaps other windows that were open previously, they are deactivated, i.e. no more characters can be output in them.
- The cursor is positioned to the origin of the window (leftmost column, top line).
- When the window is opened, the space which is taken up by it is not overwritten; the **second window** is simply **superimposed** on the first window. Opening this window causes any other windows which are overlapped by it to be deactivated. When the window is closed again, the background on which it was superimposed reappears.
- **Extended settings** can be defined for the window after it has been opened with the **"Text Window Style" command** (refer to page 65). A default setting (see "Text window style" command) applies until the window style is set; the window has no border and no cursor. Under normal circumstances, therefore, it is only possible to tell that a window has been opened from the fact that the previous window background disappears.
- Window settings which are defined at the time the window is opened (such as the character set) cannot be changed later on when it is already open. If you want to get round this problem, you must close the old window first and then open a new window with new settings.
- You must assign the transferred **handle number** specifically for this window. It is needed in order to access the window again later on. If you attempt to open a window with a handle number that has already been assigned, the open window command will be ignored.

Open graphic window	ASCII: **ESC ESC 'X' x1high x1low** **y1high y1low** **x2high x2low** **y2high y2low** **colfront colback** **hhigh hlow**
	HEX: 1Bh 1Bh 58h ....

Parameters:

x1, y1:	Screen coordinates of the top left-hand corner of the window
x2, y2:	Screen coordinates of the bottom right-hand corner of the window
n:	Character set in the text window (1...4 for integrated character sets)
colfront:	Character color
colback:	Background color
h:	Handle number (individual window identification, 1...255)
1...239:	Freely assignable window numbers
240...249:	Recommended windows for MULTI messages
250...255:	Reserved windows (e.g. for weight outputs)

- A window is opened is opened at the transferred coordinates (*x1*, *y1, x2, y2*).
- If the coordinates (*x1*, *y1, x2, y2*) are outside the display, the open window command will be ignored.
- The window is opened in the foreground and is therefore active. If the new window overlaps other windows that were open previously, they are deactivated, i.e. no more characters can be output in them.
- When the window is opened, the space which is taken up by it is not overwritten; the **second window** is simply **superimposed** on the first window. Opening this window causes any other

windows which are overlapped by it to be deactivated. When the window is closed again, the background on which it was superimposed reappears.

- **Extended settings** can be defined for the window after it has been opened with the **"Text Window Style" command** (refer to page 65). A default setting (see "Text window style" command) applies until the window style is set; the window has no border. Under normal circumstances, therefore, it is only possible to tell that a window has been opened from the fact that the previous window background disappears.
- You must assign the transferred **handle number** specifically for this window. It is needed in order to access the window again later on. If you attempt to open a window with a handle number that has already been assigned, the open window command will be ignored.
- In order to be able to output graphic elements in this window, you must first point handle G to it (refer to page 70). You can only output graphic elements in active windows, in other words windows that are displayed in the foreground. If the "update background" function is active, the elements will still be output, though somewhat slower.

Close window	ASCII: **ESC ESC 'w' hhigh hlow**
	HEX: 1Bh 1Bh 77h ...

Parameter: h: Handle no. (individual window identification, 1...255)

- This command closes the window which was previously opened with handle number *h*.
- If an open window with handle number *h* does not exist, the command will be ignored.
- The background on which the window was previously superimposed is rebuilt. Further access to this handle number is barred, in other words, the number is released again for opening other windows. The window style which was valid previously is also reset.
- Other windows which were previously overlapped by the window that is now closed are activated again, providing they are not overlapped by a third window.
- In order to close a window, it does not necessarily have to be active. Windows which are overlapped by other windows or windows which are completely invisible can also be closed. Any windows which are no longer overlapped after the active window is closed are activated again automatically.
- If Handle2 (input from keyboard) was pointing to handle number *h*, it is reset to 0 (free input). Handle1 and Handle3 are not affected by this.

Set window style	ASCII: **ESC ESC 'Q' q1 q2 hhigh hlow**
	HEX: 1Bh 1Bh 51h ....

Parameters: q1: Bit pattern 1
q2: Bit pattern 2
h: Handle no. (individual window identification, 1...255)

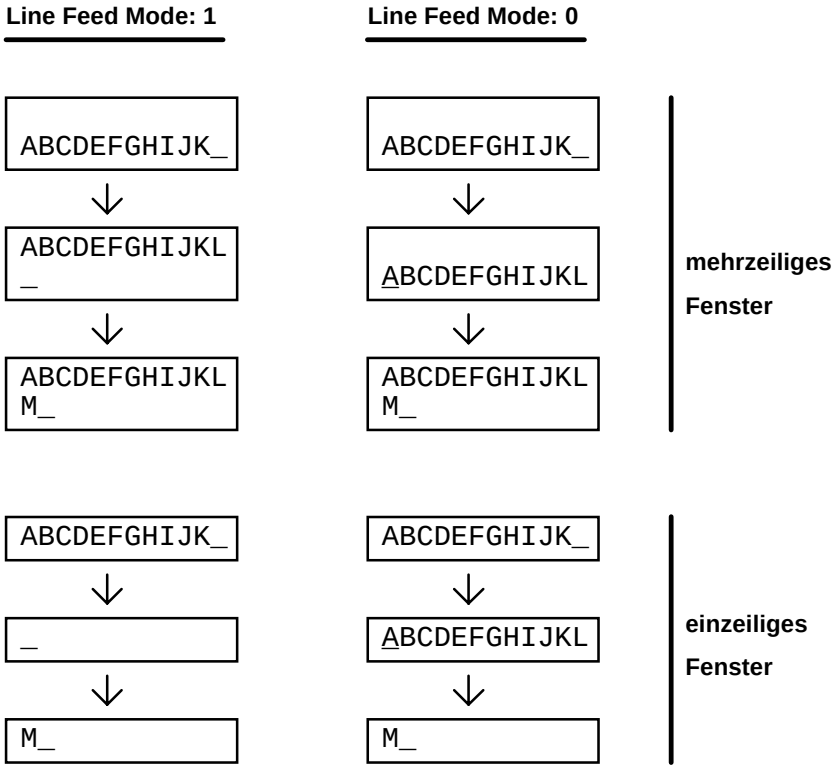
- A style is set for a window with handle number h which is already open.
- This command can be used any number of times on an open window to change this window's properties as and when required.
- The window must however be active, in other words it must be in the foreground. Any changes you make to the style of non-active windows will be ignored. They will not even take effect if these windows are activated again later on.
- All changes made with this command to an active window take effect immediately.
- Each bit in bit patterns q1 and q2 corresponds to one option, which influences the style independently of all the other options (see table below).
- The **default** values q1=0 and q2=0 apply to all windows until the style is set explicitly with this command.

The table below lists the meanings of the style bits. Some style bits are not available for graphic windows.

	Bit no.	Meaning of set bit	Value of set bit	Text	Graphics
q 2	0	Scrolling enabled	01h	•	
	1	Line feed mode: 1	02h	•	
	2	Window border	04h	•	•
	3	Cursor visible	08h	•	
	4	Input limit	10h	•	
	5	Inverse representation	20h	•	
	6	Character inversion	40h	•	
	7	Don't clear at page break	80h	•	
q 1	0	Inverse blinking	01h	•	
	1	Blinking on/off	02h	•	•
	2	(Reserved)	04h		
	3	(Reserved)	08h		
	4	(Reserved)	10h		
	5	(Reserved)	20h		
	6	(Reserved)	40h		
	7	(Reserved)	80h		

Scrolling enabled:
If this bit is set, a carriage return at the bottom of a text window causes the window to be scrolled up one line. If not, the cursor is positioned to the start of the window again (leftmost column, top line). Another flag determines whether the window should be cleared in this case or whether its contents should remain stored ("Don't clear at page break").

Line feed mode:
If this bit is set, a carriage return takes place after a character has been set in the last column of a line (normal terminal response). If not, the carriage return is effected before a character is set in the first column.
The diagram below shows how characters are printed in each of these two operating modes. The arrows each represent one character that is sent to the window. The underscore represents the cursor.



Mode 0 is mainly intended for single-line input windows, so that after a line feed (which causes a carriage return equivalent to typing "L" in the example above) the characters which have been entered do not disappear immediately, but remain visible until the next character is entered.

Window border:
If this bit is set, a border is drawn around the window in the same color as the window itself. The appearance of the border is determined by the **Set window border** commanddescribed on page 68 and by the **Set window border size** command described on page 69.

Cursor visible:
If this bit is set, the cursor appears as a block at the next character output position. This serves above all to indicate that the control computer is waiting for an input via the internal keyboard.

Input limit:
If this bit is set, character inputs via the internal keyboard are limited to (line length -1). All characters that are entered subsequently will be ignored. The function keys remain active, however, and continue to send the corresponding codes to the control computer via SER1.

Inverse representation:
If this bit is set, the whole window appears in inverse representation (background, border, cursor, characters).

Character inversion:
This flag can be used to invert individual characters in a window. If the bit is set, the characters appear in inverse representation. If it is reset again, all characters entered subsequently appear in normal representation.

Don't clear at page break:
If scrolling is switched off and a form feed occurs at the bottom of the page, the cursor is positioned in the top left-hand corner. If this flag is set, the window's contents remain stored. If not, the window is cleared.

Inverse blinking
The window is displayed alternately with normal colors and inverted colors. When it is inverted, the foreground color and the background color are swapped. You can define the blinking rate in the setup.

Blinking on/off
The window is alternately displayed and hidden. You can define the blinking rate in the setup. If you are working with large windows or with large background windows, blinking may cause the other terminal functions to be slowed down.

Set text window style attributes	ASCII: **ESC ESC 'C' q1 q2** HEX: 1Bh 1Bh 43h ....

Parameters:	q1:	Bit pattern 1
	q2:	Bit pattern 2

Reset text window style attributes	ASCII: **ESC ESC 'c' q1 q2**
	HEX: 1Bh 1Bh 63h ....

Parameters: q1: Bit pattern 1
q2: Bit pattern 2

These two commands allow you to activate and deactivate individual window style attributes explicitly. You do not necessarily need to know the previous window style settings for this purpose.
The commands always refer to the current output window, in other words to the window to which handle 1 is pointing.

Examples:
ESC ESC "C" 0 40h : Switches character inversion on
ESC ESC "c" 0 40h : Switches character inversion off

ESC ESC "C" 0 8h : Switches the cursor to visible
ESC ESC "c" 0 8h : Switches the cursor to invisible

Note that with the ESC ESC "c" ... command you must set precisely the bits whose style attributes you want to switch off.
If style attributes which are already set or reset are set or reset again, nothing happens.

Set window border	ASCII: **ESC ESC 'N' bord hhigh hlow**
	HEX: 1Bh 1Bh 4Eh ....

Parameters: bord: Window border type
h: Handle no. (1...255)

- This command defines the border type for the window with the specified number.
- It must be used before the style bit for the window border is set. Once the window border has been drawn, you can no longer change the border type.

The following border types are available:

bord	Description	Border thickness top/bottom	Border thickness left/right
0	Freely definable	Variable	Variable
1	Single-thickness border	8 pixels	16 pixels
2	Double-thickness border	8 pixels	16 pixels
3	3D border, dark	8 pixels	16 pixels
4	3D border, faint	8 pixels	16 pixels

- The thickness of a freely definable border (border size) is freely selectable, and you can output any graphic elements in any colors within the limits of this border.

How to set a freely definable border:
1. Specify the thickness (size) of the freely definable border
2. Set the type to "freely definable"
3. Set the style bit for a window which is already open
4. Point handle G to this window
5. Output graphic elements within the border

Set window border size	ASCII: **ESC ESC 'O' lefthigh leftlow righthigh rightlow tophigh toplow bothigh botlow bindex**
	HEX: 1Bh 1Bh 4Fh ....

Parameters:
- left: Border size (left) in pixels
- right: Border size (right) in pixels
- top: Border size (top) in pixels
- bot: Border size (bottom) in pixels
- bindex: Border index (0)

- You can specify the size of a freely definable border on all four sides of the window (left, right, top, bottom).
- The border size you specify with this command remains valid until you change it again.
- If you then activate a window border (by setting the style bit), and the type was previously set to "freely definable", this border is drawn with the size that was last specified using this command (see also **Set window border** command on page 68).

Set handle1 (text output)	ASCII: **ESC ESC 'H' '1' hhigh hlow**
	HEX: 1Bh 1Bh 48h 31h ...

Parameter:
- h: Handle no. (0...255)

This command determines the destination of the text channel for characters that are output.

h=0: Free output to the predetermined cursor position on the display.

h=1...255: Output to the window determined by handle number *h*.

- Even if a window with the specified handle number does not exist, the handle is still set. If you attempt to output characters using this handle, however, they will be ignored.

Set handle2 (input from keyboard)	ASCII: **ESC ESC 'H' '2' hhigh hlow**
	HEX: 1Bh 1Bh 48h 32h ...

Parameter:
- h: Handle no. (0...255)

This command determines the destination of the text channel for characters that are received from the internal keyboard.

h=0: Direct output to the control computer via SER1 (no output on the display).

h=1...255: Output to the window determined by handle number *h* with output of characters to the control computer via SER1 after inputs have been confirmed with ENTER. The end character of the transferred character string is normally the control character CR, though this can be changed to LF in the setup.

- Even if a window with the specified handle number does not exist, the handle is still set. If you attempt to output characters using this handle, however, they will be ignored.
- If you clear a window to which handle2 is pointing, this handle is automatically reset to 0.
- The cursor in the window to which the keyboard handle is pointing blinks at the rate specified in the setup, providing it is switched on in the text window style.

Set handle G (graphic output)	ASCII: **ESC ESC 'H' 'G' hhigh hlow**
	HEX: 1Bh 1Bh 48h 47h ...

Parameter: h: Handle no. (1...255)

This command determines the window to which graphic elements are output.

- Even if a window with the specified handle number does not exist, the handle is still set. If you attempt to output graphic elements using this handle, however, they will be ignored.
- The graphic elements (pixels, lines, rectangles and boxes) are output to the window with the specified handle number. Please remember that the relative coordinates which are specified there must refer to the top left-hand corner of the window.

Activate window	ASCII: **ESC ESC 'A' hhigh hlow**
	HEX: 1Bh 1Bh 41h ...

Parameter: h: Handle no. (individual window identification, 1...255)

- The (non-active) window h is activated, in other words it is placed in the foreground. Any other windows which are now overlapped are automatically deactivated.

Window invisible	ASCII: **ESC ESC 'a' hhigh hlow**
	HEX: 1Bh 1Bh 61h ...

Parameter: h: Handle no. (individual window identification, 1...255)

- Window h (either active or non-active) is placed in the background, in other words it becomes completely invisible, regardless of any other windows which are also displayed.
- Any other windows which were previously only overlapped by this window are now activated again automatically.
- This command is extremely useful, for example, if you are working with interrupt windows. These windows may already be pre-built when you start the system. You can then make them invisible, so that they can be displayed again very quickly in case of an interrupt.

Backspace, BS	ASCII: **BS**
	HEX: 08h

- The cursor moves one position to the left in the handle1 window and deletes the character at that position.
- If the cursor is already at the left margin, the command will be ignored. The cursor does not move up one line.

Line feed, LF	ASCII: **LF** HEX: 0Ah

- The cursor moves to the beginning of the next line in the handle1 window.
- If the cursor is already in the bottom line, either the window is scrolled up one line or the cursor is positioned to the top of the window, depending on the selected window style (see **Set window style** command on page 65 for further details).

Carriage return, CR	ASCII: **CR** HEX: 0Dh

- The cursor moves to the beginning of the current line in the handle1 window.

Cursor up	ASCII: **ESC '[' (nn) 'A'** HEX: 1Bh 5Bh 41h

Parameter: (nn): Number of lines

- The cursor in the handle1 window moves up nn lines.
- The system expects the parameter (nn) to be formatted, in other words a character string made up of the digits (e.g. 12 lines: ASCII '1' '2' ; HEX 31h 32h). The character string can consist of either one or two digits.
- This command is compatible with ANSI/VT100.
- The cursor only moves up as far as the top edge of the window; all other navigation commands in the same direction will be ignored.

Cursor down	ASCII: **ESC '[' (nn) 'B'** HEX: 1Bh 5Bh 42h

Parameter: (nn): Number of lines

- The cursor in the handle1 window moves down nn lines.
- The system expects the parameter (nn) to be formatted, in other words a character string made up of the digits (e.g. 8 lines: ASCII '8' ; HEX 38h). The character string can consist of either one or two digits.
- This command is compatible with ANSI/VT100.
- The cursor only moves down as far as the bottom edge of the window; all other navigation commands in the same direction will be ignored.

Cursor right	ASCII: **ESC '[' (nn) 'C'** HEX: 1Bh 5Bh 44h

Parameter: (nn): Number of characters

- The cursor in the handle1 window moves nn lines to the right.
- The system expects the parameter (nn) to be formatted, in other words a character string made up of the digits (e.g. 17 characters: ASCII '1' '7' ; HEX 31h 37h). The character string can consist of either one or two digits.
- This command is compatible with ANSI/VT100.
- The cursor only moves across as far as the right edge of the window; all other navigation commands in the same direction will be ignored.

Cursor left	ASCII: **ESC '[' (nn) 'D'**
	HEX: 1Bh 5Bh 43h

Parameter: (nn): Number of characters

- The cursor in the handle1 window moves nn lines to the left.
- The system expects the parameter (nn) to be formatted, in other words a character string made up of the digits (e.g. 23 characters: ASCII '2' '3' ; HEX 32h 33h). The character string can consist of either one or two digits.
- This command is compatible with ANSI/VT100.
- The cursor only moves across as far as the left edge of the window; all other navigation commands in the same direction will be ignored.

Cursor absolute	ASCII: **ESC '[' (yy) ';' (xx) 'H'**
	HEX: 1Bh 5Bh .... 3Bh 48h

Parameters: (yy): Line position
(xx): Column position

- The cursor in the handle1 window moves to the position (xx | yy). The origin of the coordinates (1 | 1) is located in the top left-hand corner of the window.
- The system expects the parameters (yy) and (xx) to be formatted, in other words a character string made up of the digits (e.g. position (3 | 4): ASCII: ESC '[' '4' ';' '3' 'H'; HEX: 1Bh 5Bh 34h 3Bh 33h 48h). The character string can consist of either one or two digits or be omitted completely. In the latter case, its value is assumed to be 1. You can thus position the cursor to the origin with the ESC '[' ';' 'H' command.
- This command is compatible with ANSI/VT100.
- If a coordinate value is so high that it is outside the window (e.g. line 5 in a 3-line window), the cursor only moves as far as the edge (i.e. to line 3).

Delete current line	ASCII: **ESC 'Y'**
	HEX: 1Bh 59h

- The line to which the character output cursor in the handle1 window is currently positioned is deleted.
- The cursor is then positioned to the beginning of this same line.

Clear window	ASCII: **ESC ESC 'd' hhigh hlow**
	HEX: 1Bh 64h ...

- The contents of window h are deleted.
- The window area is then the same color as the window background.
- This command has the same effect on both text and graphic windows.
- If a text window is cleared, the cursor is subsequently positioned to the top left-hand corner.

Set color of graphic output	ASCII: **ESC ESC '+' col**
	HEX: 1Bh 59h

- This command sets the color of all graphic elements which are output subsequently using handle G.
- It acts like a paintbrush on pixels, lines, rectangles and boxes.
- The color remains the same until another color is set.

Set pixel	ASCII: **ESC ESC 'P' xhigh xlow yhigh ylow**
	HEX: 1Bh 1Bh 50h ....

Parameters: x: Relative X-coordinate (column) (1...window width)
y: Relative Y-coordinate (line) (1...window height)

- A pixel is output in the window to which handle G is pointing. The specified coordinates are relative. They refer to the top left-hand corner of the window.
- If the coordinates are outside the width or height of the window, this command is ignored.

Draw line	ASCII: **ESC ESC 'L' x1 y1 x2 y2**
	HEX: 1Bh 1Bh 4Ch ....

Parameters: x1, y1: Relative coordinates of the start point of the line
x2, y2: Relative coordinates of the end point of the line

- A line (thickness: 1 pixel) is drawn in the window to which handle G is pointing. The coordinates of the start and end points are relative. They refer to the top left-hand corner of the window.
- The term "start and end points" does not necessarily mean that the coordinates of the start point must be smaller than the coordinates of the end point.

Draw rectangle	ASCII: **ESC ESC 'S' x1 y1 x2 y2**
	HEX: 1Bh 1Bh 53h ....

Parameters: x1, y1: Relative coordinates of corner 1
x2, y2: Relative coordinates of corner 2

- An orthogonal, filled rectangle (set pixel) which spans the diagonally opposite corners 1 and 2 is drawn (line thickness: 1 pixel). The coordinates of the corners are relative. They refer to the top left-hand corner of the window.

Draw bar	ASCII: **ESC ESC 'B' x1 y1 x2 y2**
	HEX: 1Bh 1Bh 42h ....

Parameters: x1, y1: Relative coordinates of corner 1
x2, y2: Relative coordinates of corner 2

- An orthogonal, filled rectangle (bar) which spans the diagonally opposite corners 1 and 2 is drawn. The coordinates of the corners are relative. They refer to the top left-hand corner of the window.

6.2 Key Codes in the EX TEC Protocol

The following table lists the key codes which are output by the terminal via the SER1 interface when keys are pressed in the EX TEC protocol.

Key	Name	Code (dec.)	Code (hex.)
CLR	Clear key	8	08h
	ENTER key	10	0Ah
-	Minus key	45	2Dh
.	Period / comma	46	2Eh
0	Number 0	48	30h
1	Number 1	49	31h
2	Number 2	50	32h
3	Number 3	51	33h
4	Number 4	52	34h
5	Number 5	53	35h
6	Number 6	54	36h
7	Number 7	55	37h
8	Number 8	56	38h
9	Number 9	57	39h
F1	Function key 1	128	80h
F2	Function key 2	129	81h
F3	Function key 3	130	82h
F4	Function key 4	131	83h
F5	Function key 5	132	84h
F6	Function key 6	133	85h
F7	Function key 7	134	86h
F8	Function key 8	135	87h
F9	Function key 9	136	88h
F10	Function key 10	137	89h
F11	Function key 11	138	8Ah
F12	Function key 12	139	8Bh
F13	Function key 13	140	8Ch
F14	Function key 14	141	8Dh
F15	Function key 15	142	8Eh
F16	Function key 16	143	8Fh
<⇧> 0	<Shift> 0	160	A0h
<⇧> 1	<Shift> 1	161	A1h
<⇧> 2	<Shift> 2	162	A2h
<⇧> 3	<Shift> 3	163	A3h
<⇧> 4	<Shift> 4	164	A4h
ACK, ***<⇧> 5***	Message acknowledgment key (<Shift> 5)	165	A4h
<⇧> 6	<Shift> 6	166	A6h
<⇧> 7	<Shift> 7	167	A7h
<⇧> 8	<Shift> 8	168	A8h
<⇧> 9	<Shift> 9	169	A9h
+	Increment key	178	B2h
-	Decrement key	179	B3h
S1	Special key 1	144	90h

S2	Special key 2	145	91h
S3	Special key 3	146	92h
S4	Special key 4	147	93h
S5	Special key 5	148	94h
S6	Special key 6	149	95h
S7	Special key 7	150	96h
S8	Special key 8	151	97h
S9	Special key 9	152	98h
S10	Special key 10	153	99h
	Message shift key (up)	183	B7h
	Message shift key (down)	184	B8h
	Set variable shift key (cursor up, 2D)	185	B9h
	Set variable shift key (cursor down, 2D)	186	BAh
	Set variable shift key (cursor left, 2D)	187	BBh
	Set variable shift key (cursor right, 2D)	188	BCh
	Info key	199	C7h

Note

- With certain keys, internal functions take priority over key code outputs. If an internal function is addressed, no codes are output via the interface:

Message shift keys	If user-defined messages exist in the loaded project or internal messages are activated in the setup, no key codes are output.
Increment / decrement keys	If set variables exist on the currently visible screen, no key codes are output.
Variable shift keys)	If set variables exist on the currently visible screen, no key codes are output.
Number keys, period / comma, minus key, clear key, ENTER key	If the keyboard input handle is pointing to a window, inputs will first be displayed in this window, then stored and finally sent all together after the input has been terminated with ENTER (refer to page 69).

6.3 Possible Settings

The list below shows all the possible settings that can be entered in the setup dialog or using the setup tool on the host computer. The settings that are actually available for each point and the setting limits for your particular firmware are indicated in the setup dialog (refer to page 11).

Setting	Description
hwar_TerminalType	Terminal type
hwar_TypeIdentifier	Type identifier of the unit
hwar_SerialNumber	Serial number
hwar_ManufacDate	Date of manufacture
hwar_FirstFirmwareVersion	Firmware version supplied
pass_Level2	Password for level 2
pass_Level3	Password for level 3
pass_Level4	Password for level 4
pass_Level5	Password for level 5
pass_Level6	Password for level 6
pass_Level7	Password for level 7
pass_Level8	Password for level 8
ser1_Aliasname	Alias name of SER1
ser1_Use	Use of SER1
ser1_Baudrate	Baud rate for SER1
ser1_Parity	Parity for SER1
ser1_Bits	Data bits for SER1
ser1_Stops	Stop bits for SER1
ser1_Baudrate_DLoad	Baud rate for SER1 in download mode
ser1_Parity_DLoad	Parity for SER1 in download mode
ser1_Bits_DLoad	Data bits for SER1 in download mode
ser1_Stops_DLoad	Stop bits for SER1 in download mode
ser2_Aliasname	Alias name of SER2
ser2_Use	Use of SER2
ser2_Baudrate	Baud rate for SER2
ser2_Parity	Parity for SER2
ser2_Bits	Data bits for SER2
ser2_Stops	Stop bits for SER2
ser3_Aliasname	Alias name of SER3
ser3_Use	Use of SER3
ser3_Baudrate	Baud rate for SER3
ser3_Parity	Parity for SER3
ser3_Bits	Data bits for SER3
ser3_Stops	Stop bits for SER3
ser4_Aliasname	Alias name of SER4
ser4_Use	Use of SER4
ser4_Baudrate	Baud rate for SER4
ser4_Parity	Parity for SER4
ser4_Bits	Data bits for SER4
ser4_Stops	Stop bits for SER4
ser5_Aliasname	Alias name of SER5
ser5_Use	Use of SER5
ser5_Baudrate	Baud rate for SER5
ser5_Parity	Parity for SER5

ser5_Bits	Data bits for SER5
ser5_Stops	Stop bits for SER5
ser6_Aliasname	Alias name of SER6
ser6_Use	Use of SER6
ser6_Baudrate	Baud rate for SER6
ser6_Parity	Parity for SER6
ser6_Bits	Data bits for SER6
ser6_Stops	Stop bits for SER6
ser7_Aliasname	Alias name of SER7
ser7_Use	Use of SER7
ser7_Baudrate	Baud rate for SER7
ser7_Parity	Parity for SER7
ser7_Bits	Data bits for SER7
ser7_Stops	Stop bits for SER7
ser8_Aliasname	Alias name of SER8
ser8_Use	Use of SER8
ser8_Baudrate	Baud rate for SER8
ser8_Parity	Parity for SER8
ser8_Bits	Data bits for SER8
ser8_Stops	Stop bits for SER8
gnrl_StartProject	Selected start project
gnrl_VariableAktTime	Variable update time
gnrl_MessageAktTime	Message update time
gnrl_KeyAutorepeat	Key auto-repeat yes/no
gnrl_KeyAutorepeatDelay	Key auto-repeat delay
gnrl_KeyAutorepeatRate	Key auto-repeat rate
gnrl_KeyAutorepeatRateAlt	Key auto-repeat in ALT mode
gnrl_AlphanumModifyTime	Time window for alphanumeric input
gnrl_CursorBlinkRate	Cursor blinking rate
gnrl_WindowBlinkRate	Window blinking rate
gnrl_AutoAckEnable	Auto-acknowledge enable yes/no
gnrl_StartupMessageTime	Startup message time
gnrl_StartupDelay	Startup delay
gnrl_DebugEnable	Debug enable yes/no
gnrl_ScannerCharfilter	Scanner character filter
gnrl_TransTimeVar	Transfer time/date yes/no
gnrl_SetupEntryLevel	Setup entry level
gnrl_ProjUploadEnable	Upload of stored project enabled or not
gnrl_MouseFunction	Function of integrated mouse
gnrl_MouseXSensitivity	Sensitivity of the mouse in x direction
gnrl_MouseYSensitivity	Sensitivity of the mouse in y direction
epca_Speed	EPCA speed
epca_HaltOnErrors	EPCA halt on errors yes/no
prot_HostProtocol	Host interface protocol
prot_ExtecSend	EX TEC protocol send instance
prot_ExtecReceive	EX TEC protocol receive instance
prot_HostXonXoff	XON/XOFF yes/no
prot_SiemensPlcType	Siemens PLC type
prot_SiemensPgmux	Multiplexer for AS511 yes/no
prot_3964rCoord1	Coordination flag1 for 3964R
prot_3964rCoord2	Coordination flag2 for 3964R
prot_AllenBradleyPlcType	Allen-Bradley PLC type
prot_AllenBradleyFileNr	File no. of Allen-Bradley PLC
prot_AllenBradleyPlcNetadr	Network address of Allen-Bradley PLC
prot_AllenBradleyTermNetadr	Network address of AB terminal

prot_AllenBradleyRespTimeout	Response timeout for AB
prot_AllenBradleyMaxCycletime	Max. cycle time of AB PLC
prot_ModbusSlaveAddress	Terminal slave address for MODBUS interface
prot_DbAddr	Data block address
prot_DbGeneralDataExchange	General data exchange in DB yes/no
prot_DbMessageExchange	Message exchange in DB yes/no
mess_DispMode	Message display mode
mess_DbUse	Use of data block
mess_IntMessagesEnable	Internal messages yes/no
mess_ErrorDispEnable	Error display yes/no
mess_ErrorDispPri	Error display priority
mess_WarningDispEnable	Warning display yes/no
mess_WarningDispPri	Warning display priority
mess_HintDispEnable	Hint display yes/no
mess_HintDispPri	Hint display priority
mess_StopDelTimeDay	Number of days until deletion from Message_event.list
mess_StopDelTimeHour	Number of hours until deletion from Message_event.list
scal_WmApprovable	Mettler scale approvable yes/no
scal_MettlerOrdWaittime	Mettler command wait time
prnt_InitSeq	Printer initialization sequence
prnt_LinefeedSeq	Printer line feed sequence
prnt_FormfeedSeq	Printer form feed sequence
prnt_EndSeq	Printer end sequence

Rules for the setup level

Settings	**Read level**	**Write level**
hwar_	1	10
pass_	Depends on the level to which the password variable applies	Depends on the level to which the password variable applies
serX_ and gnrl_	1	2
gnrl_SetupEntryLevel	1	8

6.4 Character Sets / Character Codes / Control Characters

TERMEX 750 is supplied with six character sets in various sizes.

Character set no.	Pixel resolution x * y	Max. characters / line	Max. lines
1	8 x 15	80	32
2	10 x 20	64	24
3	16 x 30	40	16
4	20 x 40	32	12
5	32 x 60	20	8
6	40 x 80	16	6

Printable characters are coded in ASCII. The following characters are available:

Decimal	Hexa-decimal	Character	Decimal	Hexa-decimal	Character	Decimal	Hexa-decimal	Character
32	20 h	' '	-	-		96	60 h	`
33	21 h	!	65	41 h	A	97	61 h	a
34	22 h	"	66	42 h	B	98	62 h	b
35	23 h	#	67	43 h	C	99	63 h	c
36	24 h	$	68	44 h	D	100	64 h	d
37	25 h	%	69	45 h	E	101	65 h	e
38	26 h	&	70	46 h	F	102	66 h	f
39	27 h	'	71	47 h	G	103	67 h	g
40	28 h	(	72	48 h	H	104	68 h	h
41	29 h	)	73	49 h	I	105	69 h	i
42	2A h	*	74	4A h	J	106	6A h	j
43	2B h	+	75	4B h	K	107	6B h	k
44	2C h	,	76	4C h	L	108	6C h	l
45	2D h	-	77	4D h	M	109	6D h	m
46	2E h	.	78	4E h	N	110	6E h	n
47	2F h	/	79	4F h	O	111	6F h	o
48	30 h	0	80	50 h	P	112	70 h	p
49	31 h	1	81	51 h	Q	113	71 h	q
50	32 h	2	82	52 h	R	114	72 h	r
51	33 h	3	83	53 h	S	115	73 h	s
52	34 h	4	84	54 h	T	116	74 h	t
53	35 h	5	85	55 h	U	117	75 h	u
54	36 h	6	86	56 h	V	118	76 h	v
55	37 h	7	87	57 h	W	119	77 h	w
56	38 h	8	88	58 h	X	120	78 h	x
57	39 h	9	89	59 h	Y	121	79 h	y
58	3A h	:	90	5A h	Z	122	7A h	z
59	3B h	;	91	5B h	[	123	7B h	{
60	3C h	<	92	5C h	\	124	7C h	|
61	3D h	=	93	5D h	]	125	7D h	}
62	3E h	>	94	5E h	^	126	7E h	~
63	3F h	?	95	5F h	_	-	-	

The following special characters can be activated using both the DOS and the Windows codes:

DOS		Windows		Character
Decimal	Hexa-decimal	Decimal	Hexa-decimal	
142	8E h	196	C4 h	**Ä**
153	99 h	214	D6 h	**Ö**
154	9A h	220	DC h	**Ü**
132	84 h	228	E4 h	**ä**
148	94 h	246	F6 h	**ö**
129	81 h	252	FC h	**ü**
225	E1 h	223	DF h	**ß**

ASCII control characters

The control characters used in this manual are printed in bold face.

Decimal	Hexa-decimal	Character
0	00 h	NUL
1	01 h	SOH
2	**02 h**	**STX**
3	**03 h**	**ETX**
4	04 h	EOT
5	05 h	ENQ
6	**06 h**	**ACK**
7	07 h	BEL
8	**08 h**	**BS**
9	09 h	HT
10	**0A h**	**LF**
11	0B h	VT
12	0C h	FF
13	**0D h**	**CR**
14	0E h	SO
15	0F h	IS
16	10 h	DLE
17	**11 h**	**DC1 (XON)**
18	12 h	DC2
19	**13 h**	**DC3 (XOFF)**
20	14 h	DC4
21	15 h	NAK
22	16 h	SYN
23	17 h	ETB
24	18 h	CAN
25	19 h	EM
26	1A h	SUB
27	**1B h**	**ESC**
28	1C h	FS
29	1D h	GS
30	1E h	RS
31	1F h	US

6.5 System of Coordinates

The following system of coordinates (x/y) is valid for all information entered using coordinates (both EX TEC protocol commands and TERMEX PRO):

1/1		640/1
1/480		640/480

6.6 Colors

The following table lists the standard colors that are available when the unit is started up. You can change these colors in TERMEX PRO. Colors 0 to 221 can be used for bitmaps.

Color no.	Description
0...221	Used for bitmaps in conjunction with TERMEX PRO
222	Black
223	White
224	Light gray
225	Dark gray
226	Red
227	Dark red
228	Very dark red
229	Green
230	Dark green
231	Very dark green
232	Yellow
233	Dark yellow
234	Very dark yellow
235	Blue
236	Dark blue
237	Very dark blue
238	Magenta
239	Dark magenta
240	Very dark magenta
241	Turquoise
242	Dark turquoise
243	Very dark turquoise
244	Light orange
245	Dark orange
246	Green
247	Light brown
248	Pink
249	Skin-colored
250	Green
251	Dark brown
252	Dark red
253	Medium red
254	(Reserved, do not use !)
255	(Reserved, do not use !)

6.7 Variables

BCD1	**_123**	Data block 1 DW

BCD01	**0123**	Data block 1 DW

Up to four BCD-coded digits can be output with the BCD1 or BCD01 variable.
Leading zeros are only output with the BCD01 variable. The BCD1 variable outputs blanks instead.
The left digit corresponds to #3 and the right digit to #0. If fewer than four digits are used, #3, #2 and #1 are suppressed consecutively. A variable with two digits uses #1 and #0.

Bit no.	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DW x	BCD digit #3				BCD digit #2				BCD digit #1				BCD digit #0			

Each 4-bit value controls exactly one digit:

4-bit value	0h	1h	2h	3h	4h	5h	6h	7h	8h	9h	Ah	Bh	Ch	Dh	Eh	Fh
Output	**"0"**	**"1"**	**"2"**	**"3"**	**"4"**	**"5"**	**"6"**	**"7"**	**"8"**	**"9"**	":"	";"	"<"	"="	">"	"?"

Only the bold digits 0 to 9 are normally intended for output. If necessary, the other six characters can be used as well however.

MIN and MAX values can be preset in TERMEX PRO for use as a set variable. If an input is outside this preset range, it will be rejected and the old value displayed instead.

Examples:
BCD1 variable, 4 digits, DW x = 123h, output "_123"
BCD01 variable, 3 digits, DW x = 45h, output "045"

BCD2	**_1234567**	Data block 2 DW

BCD02	**01234567**	Data block 2 DW

Up to eight BCD-coded digits can be output with the BCD2 or BCD02 variable.
Leading zeros are only output with the BCD02 variable. The BCD2 variable outputs blanks instead.
The left digit corresponds to #7 and the right digit to #0. If fewer than eight digits are used, #7, #6, #5, etc. are suppressed consecutively. A variable with five digits uses #4, #3, #2, #1 and #0. The variable always requires 2 DW in the data block, however.

Bit no.	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DW x	BCD digit #7				BCD digit #6				BCD digit #5				BCD digit #4			
DW x+1	BCD digit #3				BCD digit #2				BCD digit #1				BCD digit #0			

Each 4-bit value controls exactly one digit:

4-bit value	0h	1h	2h	3h	4h	5h	6h	7h	8h	9h	Ah	Bh	Ch	Dh	Eh	Fh
Output	**"0"**	**"1"**	**"2"**	**"3"**	**"4"**	**"5"**	**"6"**	**"7"**	**"8"**	**"9"**	":"	";"	"<"	"="	">"	"?"

Only the bold digits 0 to 9 are normally intended for output. If necessary, the other six characters can be used as well however.

MIN and MAX values can be preset in TERMEX PRO for use as a set variable. If an input is outside this preset range, it will be rejected and the old value displayed instead.

Examples:
BCD2 variable, 8 digits, DW x = 123h, DW x+1 = 4567h ⇒ output "_1234567"
BCD02 variable, 5 digits, DW x = Fh, DW x+1 = 9876h ⇒ output "000?9876"

BINA "98765.50"

Data block
1 DW

VBINA "-98765.50"

Data block
1 DW

- Up to ten binary-coded digits can be output (plus sign for VBINA).
- The 16-bit binary number in the data block is scaled up to a 32-bit value using two MIN-MAX pairs.
- The 16-bit value in the data block is interpreted as an unsigned number with BINA. The number which is displayed is similarly always positive. With VBINA, the 16-bit value is considered to be a binary number in a complement of two. Depending on the scale factor, the displayed value can be either positive or negative.
- The number of integer and decimal positions for the variables can be preset. Fixed-point representation is used when they are output, in other words, the decimal point is output by the terminal at the selected position without the value in the data block needing to be changed (e.g. same value for 12.34 and 1234).
- The value range for BINA is 0...65535 in the data block and 0...4294967295 on the display.
- The value range for VBINA is -32768...32767 in the data block and -2147483648...2147483647 on the display.

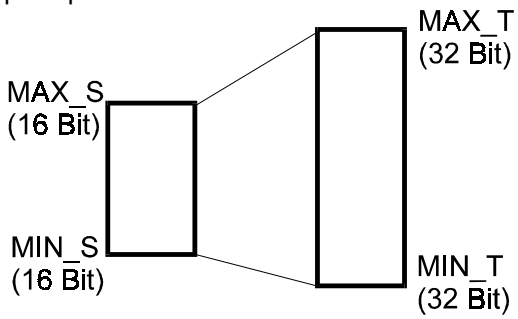
BINA

Bit no.	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DW x	16-bit binary number (unsigned integer)															

VBINA

Bit no.	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DW x	16-bit binary number (signed integer)															

The values in the data block are mapped to the values on the display according to the following principle:



The MIN_S limit is assigned to MIN_T and the MAX_S limit to MAX_T.

Example for BINA:
MIN_S (MIN PLC) = 0
MAX_S (MAX PLC) = 65535
MIN_T (MIN terminal) = 0
MAX_T (MAX terminal) = 1500000
6 integer positions, 1 decimal position
If the value 0 is entered in DW x in this example, the display reads "0.0".
If the value 65535 is entered, the display reads "150000.0".

Example for VBINA:
MIN_S (MIN PLC) = 0
MAX_S (MAX PLC) = 65535
MIN_T (MIN terminal) = -20
MAX_T (MAX terminal) = 15000
4 integer positions, 1 decimal position
If the value 0 is entered in DW x in this example, the display reads "-20.0".
If the value 65535 is entered, the display reads "1500.0".

The values between these two points are computed according to the scale factor:

$$AKT\_T = MIN\_T + \frac{(MAX\_T - MIN\_T)}{(MAX\_S - MIN\_S)} * (AKT\_S - MIN\_S)$$

where AKT_S is the actual 16-bit value in the data block and AKT_T is the 32-bit output on the terminal display.

Note:
- The number range on the display need not necessarily be larger than the number range on the PLC; it can also be scaled down.
- The output may not be absolutely precise. The maximum error is 1 digit increment on the display (e.g. 99999 instead of 100000).
- MIN_T and MAX_T can also be used as limit values for a set variable. If an input is outside this preset range, it will be rejected and the old value displayed instead.

Colors:
BINA and VBINA variables can be configured so that their colors change dynamically according to the value.

BINB	**"112233.44"**	Data block 2 DW

VBINB	**"-112233.44"**	Data block 2 DW

- The 32-bit binary number which is entered in two data words in the data block is output on the display without scaling.
- The number is positive for BINB variables. With VBINB variables, the number is interpreted as a complement of two, in other words it may also be negative.
- Fixed-point representation is used when the variables are output, in other words, the decimal point is output by the terminal at the selected position without the value in the data block needing to be

changed (e.g. same value for 12.34 and 12345). The number of decimal and integer positions must be specified in TERMEX PRO.

- The value range for BINB variables is 0...4294967295.
  The value range for VBINB variables is -2147483648...2147483647.

.

BINB

Bit no.	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DW x	32-bit binary number (long unsigned integer, high 16 bits)															
DW x+1	32-bit binary number (long unsigned integer, low 16 bits)															

VBINB

Bit no.	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DW x	32-bit binary number (long signed integer, high 16 bits)															
DW x+1	32-bit binary number (long signed integer, low 16 bits)															

Examples:

BINB variable: 6 integer positions, 4 decimal positions, DW x = 1E2Ah, DW x+1 = 301Ch
⇒ output on display "50608.1308"

VBINB variable: 5 integer positions, 3 decimal positions, DW x = FF86h, DW x+1 = 229Dh
⇒ output on display "_7986.351"

Note:

- If you want to work with low numbers which only require 16 bits, you must nevertheless keep the two data words free in the data block. '0' is then written in DW x for BINB or a positive VBINB, while 'FFFFh' is written for a negative VBINB. DW x+1 contains the desired 16-bit value.
- The specified MIN and MAX values only function as limit values if they are used for a set variable. These values are not scaled. The binary value which is transferred is output without being converted (and therefore absolutely precisely). If you need a scale factor, you should use a BINA variable.

TEXT "OPEN"

Data block
1 bit

- Two predefined texts (variants) can be output for the TEXT variable type, depending on the status of a specific bit in the data block.
- One text is output if the bit status is 0 and the other text is output if the bit status is 1.
- Since any character set can be used for each variable, the TEXT variable can also be used to generate very simple graphic status displays. The open and closed valve conditions, for example, can each be generated as a character in a user-defined character set. The two texts then each consist of exactly the one character that corresponds to the respective valve condition.

Bit no.	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DW x													Bit			

Example:

Texts:

Text0:	ID=0	"closed"
Text1:	ID=1	"open"

Variable:

For bit=0:	Text0
For bit=1:	Text1

If the bit status is 0, the word "closed" will appear at the insertion point for the variable; if the bit status is 1, the word "open" will appear instead.

TEXT16 "Half throttle"

Data block
1 DW

- Up to 65536 predefined texts (variants) can theoretically be output for the TEXT16 variable type, depending on the binary value of a data word in the data block. Please note, however, that other limits which reduce the maximum number that is possible in practice may be reached before the maximum theoretical number, namely in particular the maximum number of blocks and the size of the project memory.
- Since any character set can be used for each variable, the TEXT16 variable can also be used to generate very simple graphic status displays. The open, half open and closed valve conditions, for example, can each be generated as a character in a user-defined character set. The three texts then each consist of exactly the one character that corresponds to the respective valve condition.

Bit no.	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DW x	16-bit binary number (integer)															

Example 1:

Texts:

Text group:	TEXT2	
Text0:	ID=0	"off"
Text1:	ID=1	"half throttle"
Text2:	ID=2	"full throttle"
Text3:	ID=3	"ultimate throttle"

Variable:

Text group:	TEXT2

If the binary value is between 0 and 3, the defined texts will appear; if the value is greater than 3, "----------" will appear instead. This serves to indicate that no variants exist for such values.

Example 2:

Texts:

Text group:	TEXT3	
Text0:	ID=0	"too low"
Text10000:	ID=10000	"correct"
Text12000:	ID=12000	"too high"
Text20000:	ID=20000	"much too high"
Text40000:	ID=40000	"much too high"

Variable:

Text group:	TEXT3

The variants can be defined for whole ranges. If value gaps are present in the variant definitions, the variant with the lower value continues to be used until the value of the next higher variant is reached. In this example:

0...9999:	"too low"
10000...11999:	"correct"
12000...19999:	"too high"
20000...40000:	"much too high"
40001...65535	"----------"

The second variant definition of "much too high" is necessary. If it is omitted, "----------" will be displayed for all values greater than or equal to 20001.

ASCII "xy123A"

Data block
x DW

- Any character strings can be output on the display by entering the ASCII codes in the data words of the ASCII variable.
- The high and low bytes of a data word are used alternately by a character code.
- The length of the character string must be specified when the ASCII variable is defined in TERMEX PRO.

Bit no.	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DW x	Character 1								Character 2							
DW x+1	Character 3								Character 4							
DW x+2	Character 5								Character 6							
DW x+3	Character 7								....							

Example:
6 characters, DW x = 7879h , DW x+1 = 3132h , DW x+2 = 3341h $\Rightarrow$ output "xy123A"

S5FLOAT "14.321"

Data block
2 DW

This variable outputs a 32-bit floating-point number (Siemens S5 format).

S7FLOAT "15.637"

Data block
2 DW

This variable outputs a 32-bit floating-point number (Siemens S7 format / IEEE-32).

DATE "20 JAN 98"

Data block
3 DW

This variable can be used to output the terminal date in the data block and in a variable window. The following display options are available:

- DMY (day month year) or MDY (month day year), in other words the day of the month can be displayed either before the month or after it.
- Separator for the year, month and day values.
- Day of the week displayed (as text) or not.
- Month displayed as a number or as text.
- Year displayed either with or without the century.

Example:
Date: Monday, 20.05.1996

Variable type: DATE
Representation: DMY (day month year)
Separator "."
Display day of week
Display month as text
Display century

This definition causes the terminal date to be output in the data block in BCD format starting at data word 100. At the same time, the date is output in the selected format in the appropriate window on the display.

Output in the data block:

Data word	High byte		Low byte	
	Millenium	Century	Decade	Year
DW x	1	9	9	6
	Month, 1st digit	Month, 2nd digit	Day, 1st digit	Day, 2nd digit
DW x+1	0	5	2	0
	Day of week		Not used	
DW x+2	0x01			

Output on the display: "MON 20 MAY 1996"

Note:

- The date update option via the data block when a PLC is connected can be switched off in the setup. The date is then only output on the display. This enables the data transfer load to the control computer to be reduced in the event that this computer already has its own real-time clock.

Setting the date

The real-time clock of the terminal can be set by means of a DATE variable if this variable is defined as a set variable.
The input format is independent of the selected output format:

"DDMMYY"

Example: To set the date to November 9th 1999 Input: "091199"
Example: To set the date to February 20th 2000 Input: "200200"

Year 2000 compliance

- The terminal still displays the date correctly in the year 2000.
- There are no other malfunctions that are either directly or indirectly related to the year 2000 problem.
- The terminal also knows that the year 2000 is a leap year, in other words that there was a February 29th.
- The date will be displayed correctly up to and including the year 2089.
- If the year is entered as a two-digit number, all years in the 90s are interpreted as 199x, while years from the first decade of the century to the 80s are interpreted as 20xx.

TIME "08:45:15"

Data block
2 DW

This variable can be used to output the terminal time in the data block and in a window. The following display options are available:

- Hours displayed in 24-hour format or 12-hour format with AM for hours before midday and PM for hours after midday.
- Separator for the hours, minutes and seconds values.
- Seconds displayed or not.

Example:
Time: Morning, 8 hours, 45 minutes, 15 seconds

Variable type: TIME
Representation: 12-hour format
Separator ":"
Display seconds

This definition causes the terminal time to be output in the data block in BCD format starting at data word x (24-hour format). At the same time, the time of day is output in the selected format in the appropriate window on the display.

Output in the data block.

Data word	High byte		Low byte	
	Hours, 1st digit	Hours, 2nd digit	Minutes, 1st digit	Minutes, 2nd digit
DW x	0	8	4	5
	Seconds, 1st digit	Seconds, 2nd digit	Not used	
DW x+1	1	5		

Output on the display: "08:45:15 AM"

Note:
- The time update option via the data block when a PLC is connected can be switched off. The time is then only output on the display. This enables the data transfer load to the control computer to be reduced in the event that this computer already has its own real-time clock.

Setting the time
The real-time clock of the terminal can be set by means of a TIME variable if this variable is defined as a set variable.
The input format is independent of the selected output format:
"HHMMSS"

Example: To set the time to 9:31 hours Input: 20931002
Example: To set the time to 18:14:30 hours Input: 21814002

MSGFILTER "AnQ"

Data block
1 DW

This variable can be used to output the filter for displayed messages. The number of the selected filter is always displayed in the data block. Either the number or a short description of the filter can be displayed in the variable output window. The possible filters are listed in the description of the message system on page 22 ff.

Example:
Filter: Messages that are active but have not yet been acknowledged (AnQ / no. 0)

Variable type: MSGFILTER
Representation: Plain text

This definition causes the message filter to be entered in DW x in the data block. At the same time the filter is output in the selected format in the appropriate window on the display.

Output in the data block:

Data word	High byte	Low byte
DW x		0

Output on the display: "AnQ"

MSGLIST "Mess4"

This internal variable can be used to output message lists (refer also to the description of messages on page 22). The variable is not displayed in the data area.
Message lists allow you to output messages and other information relevant to them in list form. The output format, the filter for selecting which messages are output and the output order are freely definable. The number of lines in the window containing the MSGLIST variable determines the number of simultaneously visible messages. Message lists can also be scrolled using the increment and decrement keys.

Format:
The format is determined by a placeholder line consisting of placeholders and printable characters. The placeholders are each replaced by the corresponding message information, while the printable characters are output unchanged.
The placeholders always have the following format:

%x99...$

%: Start of a placeholder; all subsequent characters belong to thís placeholder
x: Placeholder identifier (1 letter)
nn: Length of the placeholder in characters (1 or 2 digits from 1 to 99).
The output is always limited to the specified length; any longer outputs will be truncated
...: Detailed specification of the placeholder (only possible and necessary for some placeholders)
$: End of the placeholder; all subsequent characters are interpreted as printable characters again

Placeholder (example)	Designation	Meaning
%n3$	Message number	Message ID; identifies the position of the message bit in the data area
%p3$	Priority	Priority defined for the message
%a19hh:mm:ss dd.oo.yyyy$	Message start time	Time at which the message was activated hh: hours mm: minutes ss: seconds dd: day oo: month yyyy: year The above-mentioned inner placeholders can be either used or omitted as necessary.
%e15hh:mm:ss dd.oo.$	Message end time	Time at which the message was deactivated. The same rules apply as for the message start time. If the message is still active, lines are output instead.
%q15hh:mm dd.oo.$	Message acknowledgment time	Time at which the message was acknowledged. If the message has not (yet) been acknowledged, lines are output instead.
%h2$	Message frequency	Frequency with which the message has occurred. If the message is activated again within the chaining time, it is combined in an event entry and the frequency is incremented by one.
%i8$	Message name	Name assigned to the message
%t20$	Message text	Text assigned to the message

Example:
Format:
```
%i8$  %t28$  Pri:%p3$  Start:%a19dd.oo.yyyy hh:mm:ss$
```
Output:
```
Mess4     Level too high                Pri:103  Start:29.02.2000 10:13:00
```

Filter:
The filter determines which message events should be output with the MSGLIST variable.
The filter string is made up of a series of entries that must be logically ANDed and true in order for a message event to be accepted.

Valid entries:
A Checks whether a message is active
Q Checks whether a message has been acknowledged
/ Negation

Possible combinations:

A	Active messages
Q	Acknowledged messages
/A	Messages which are no longer active
/Q	Messages which have not (yet) been acknowledged
AQ	Active, acknowledged messages
A/Q	Active messages which have not (yet) been acknowledged
/AQ	Acknowledged messages which are no longer active
/A/Q	Unacknowledged messages which are no longer active
(No filter)	All messages

A and *A/Q* are especially useful filters, since they allow you to display messages which are currently queued. "No filter" is helpful if you want to output a message history.

Sort string:
The sort string determines the order in which message events are output. Several different sorting orders are allowed (both ascending and descending).
The sort string has the following format:

x+y-z+

x : Primary sort key
y: Secondary sort key (only applies if the primary sort key does not produce a clear result)
z: Tertiary sort key (only applies if the primary and secondary sort keys do not produce a clear result) (more sort keys can be appended as necessary)
+,-: Ascending / descending sorting order (ascending = lowest value first; descending = highest value first)

Sort keys:

n	Sorted according to message number
p	Sorted according to message priority
a	Sorted according to message start time
e	Sorted according to message end time
q	Sorted according to message acknowledgement time
h	Sorted according to message frequency
i	Sorted according to message name
t	Sorted according to message text

p- is an especially useful sort string, since it allows you to display active messages with the highest priority at the top of the list. *a-* is helpful if you want to output a message history with the most recent entries at the top.

Note:
- TERMEX PRO offers various defaults for defining the MSGLIST variable. You can either accept these defaults or customize the settings according to your particular requirements. If you select one of the defaults, the strings are copied to the input fields and you can then edit them as necessary.
- Message lists can also be printed out, providing the SER2 interface is configured as a printer port (refer to page 53).

PLUGID "13"

Data block
1 DW

- Variables of the PLUGID type detect the switch position of a coding plug at the digital inputs (X8 interface) of the terminal. The seven digital inputs are set to a defined bit pattern by means of a coding plug. If a plug is not connected to the digital inputs, the value is automatically set to 0. In this case the display will read "NC" for "Not connected" instead of "0". If you did not specify a coding plug in the setup, the display will show lines. In order to stop intermediate positions from being displayed when a coding plug is connected or disconnected, the value of the digital inputs is not transferred from the variable until it has remained constant for at least one second.

Output in the data block:

Data word	High byte	Low byte
DW x		id

6.8 Error Messages

There are four types of error message that can be output by the terminal:

Startup error messages
These error messages are sometimes displayed immediately following the startup message. The cause is usually a (more or less serious) hardware problem (refer to page 8).

Load error messages
These error messages are sometimes displayed after a project has been loaded. The reason is that the project either contains an error or takes up more resources than the terminal can provide (refer to page 53).

Runtime error messages
These error messages are sometimes displayed while the terminal is operating. They are caused by errors in EPCA programs. These errors are fatal, in other words processing is aborted after they appear (refer to page 10).

Internal messages
These messages sometimes appear during operation. Their causes can normally be rectified (protocol problems, for example). The messages behave like user-defined messages, in other words they can be acknowledged and scrolled through.
Internal messages are subdivided into hints, warnings and errors.

6.9 Technical Data

The most important technical data of firmware Version VS1.09 is listed below.
Please refer to [TERMEX] for the hardware data.

Size of SER1 input buffer	16 KB
Size of SER2...SER8 input buffers	100 bytes
Max. number of screens per project	512
Max. number of open windows	80
Max. number of open bar graphs	16
Max. number of messages per project	512
Max. number of bitmaps / character sets per project	500
Max. number of colour palettes	512
Max. number of project frames	2000
Size of project memory	3 MB
Print buffer	8 KB
RAM for EPCA programs	25 KW = 50 KB
Max. number of loadable EPCA programs per project	600
Max. number of simultaneously executable EPCA tasks	70

7 History of Versions

VS1.00 (08/1999)
- The first version is published.

VS1.01 (10/1999)
- The initialization problem with global EPCA variables is solved.
- The number of variables is increased to 500.
- Bar graphs are now represented correctly.
- The colors used to represent BINA, VBINA, BINB and VBINB variables now change dynamically. Up to four limit values with dynamically changing colors can be defined.
- The TIME and DATE variables now function correctly. The clock can be set directly using TIME and DATE set variables.

VS1.02 (11/1999)
- The memory for the internal file system is enlarged.
- The memory for EPCA is enlarged (25 K words)
- The data area is extended from 256 data words to 512. The complete area can now be addressed with the 3964R / RK512 protocol by means of two consecutive data blocks. The same applies to the PROFIBUS interface. The complete length of the data area can be addressed directly with the EXTEC16 protocol.
- The problems with internal variables (e.g. TIME and DATE) are rectified.
- The interface settings for SER3 and SER4 (keyboard and mouse) can now only be changed on setup level 9 or higher. These crucial settings are now more difficult to alter by accident.

VS1.03 (11/1999)
- Minor patches are incorporated.

VS1.04 (12/1999)
- The special EPCA processing problem in conjunction with graphic handles is now solved.
- The EX TEC protocol now allows transmission rates up to 57600 baud.
- Projects can now be read back from the terminal again (i.e. uploaded).
- The MODBUS protocol is implemented.
- Bar graphs with dynamically changing colors are now supported. They can optionally have the same color as the variable assigned to them. The variable color may change according to the actual value.
- Extended system tests are performed when the terminal is started up, including a full RAM test and a project memory test.
- The initialization problem with global EPCA variables is solved.
- The maximum number of EPCA tasks is increased from 50 to 60.
- The current screen number is now displayed correctly in DW0 of the data area.
- A new setup option allows the mouse pointer to be switched off. In addition, an automatic function permits set variables to be selected with the left mouse button (without the EPCA program).
- The SER2 serial interface can now be used as a second host interface with the EXTEC16 protocol at up to 38400 baud.
- Acknowledged message numbers are now returned correctly in DW2 of the data area.
- A special download mode can now be activated while the startup message is displayed by pressing <F2>. This mode switches automatically to the EXTEC16 protocol and a high baud rate without any changes needing to be made in the setup. The default transmission rate is 38400 baud.

VS1.05 (12/1999)

- Special setup options allow the transmission cycle to be speeded up for PLC interfaces. The *prot_DbGeneralDataExchange* option determines whether or not the general data exchange area (DW 26 to 35) should also be transmitted in the cycle. The default setting is *Yes*. *No* means that two fewer message frames need to be transmitted in the cycle, making it considerably faster. *prot_DbMessageExchange* allows transmission of the message frame (DW 46 to 77) to be switched off. The default setting is *Yes*.
- After a set variable has been entered and confirmed by pressing the ENTER key, it is not possible to switch to another set variable until the data words for this variable have been transferred to the PLC. If the transmission cycle to the PLC is very slow, this prevents any set variables from being lost when they are sent to it.

VS1.06 (01/2000)

- Projects which are stored in the terminal can be read out and stored on the PC using the *TERMEX Load* tool.
- Project downloads now work at transmission rates up to 115200 baud. The time needed to load large projects in particular can be significantly reduced as a result.
- A progress bar now appears on the display when projects are uploaded or downloaded. These temporary operating modes can thus also be observed on the unit itself. No other operations are possible on the unit while a project is being uploaded or downloaded.
- Bitmap windows in particular are now built faster. The project must have been compiled using a recent version of TERMEX PRO (> V2.54).
- The maximum number of EPCA programs per project is increased from 300 to 600.
- The message frame transfer from the PLC is now restricted to those data words that are absolutely essential. The firmware takes the messages that actually occur in the project as a basis.
- Various patches are incorporated.

VS1.07 (03/2000)

- The problem displaying window borders is solved.
- The display initialization problem following cold starts is solved. In the past, this initialization procedure was unsuccessful and the display remained black if certain conditions applied. Another reset with <Shift> <ENTER> <9> (warm start) was necessary in such cases in order for the terminal to start up correctly.
- Message can now also be output in configurable message lists. These are suitable for displaying several simultaneously queued messages as well as message histories (refer to page 91).
- Printer support is implemented. A serial printer can be connected to SER2 for printing out message lists. The control sequences for initialization, line feed, form feed and end of printing can be specified in the setup (refer to page 53).
- The integrated setup is updated.
- Minor patches are incorporated.

VS1.08 (09/2000)

- The number of windows is increased from 80 to 100.
- The number of EPCA tasks running at a time is increased from 60 to 70.
- Projects can use up to 3 MB of memory (1 MB before).
- New type of variables S7_FLOAT for the input and output of floating point variables in the format IEEE-32.
- The number of palettes is increased from 100 to 512.
- The sensitivity ot the integrated mouse may be adjusted.
- The number of bitmaps in one project is increased from 255 to 500.
- Loading errors were output at the terminal.

VS1.09 (10/2000)

- The AS511 protocol is implemented (for Siemens S5 programming interface).
- The buffering of key state changes is suppressed during PLC communication errors .
- The trend plotter is available.
- The integrated setup is actualized.

8 Glossary

Bar graphs
Bar graphs allow variable values to be displayed graphically. A bar graph always has a fixed link to one variable. A bar with a variable length is generated according to the scale factor selected for the project. Bar graphs are displayed in separate windows.

Bitmaps
A bitmap is a graphic object consisting of individual pixels, with a fixed height and width. Any desired graphics can be generated. Bitmaps are displayed in windows.

Character sets
Character sets are needed to map text on the graphic display. TERMEX 750 is supplied with six character sets in various sizes. Additional character sets can be generated in TERMEX PRO and loaded onto the terminal for specific projects. Each character is represented by a (small) bitmap; its appearance can be modified by setting pixels.

Graphic elements
Each of these objects has one characteristic (pixel, line, rectangle, bar) and can be inserted in a graphic window on the TERMEX 7xx. Complete graphics can be generated by combining several such elements. The entire graphic can be deleted simply by clearing the window.

Handles
Handles are output pointers which indicate the destination of a particular data path. The handle always includes the number of the window in which the data must be output. If the value is 0, there is no output. Users only have to work with handles in conjunction with the EX TEC protocol. All other protocols merely use them internally.
There are handles for outputting characters (handle 1), key codes (handle 2), barcodes (handles 3 and 5) and graphic elements (handle G).

Peripherals
Peripherals are supplementary devices that can be connected to a terminal, such as barcode readers, scales and additional keyboards. They may be supplied by other manufacturers.

Programs
Programs enable terminal sequences to be controlled via the EPCA programming system. They can be used to support a connected control computer or for stand-alone applications.

Projects
A project consists of screens, messages, variable definitions, character sets and programs. It is generated in TERMEX PRO, then loaded onto the terminal via the SER1 control interface and stored there residently. It contains all the definitions that are known prior to runtime and that are not modified later.

Screens
Screens have a number that allows them to be switched. A project is usually subdivided into screens in order to separate the individual control steps. There may be an overview screen, for example, various detail screens and a parameter input screen.
A screen can include windows, texts, variables, bar graphs and graphic elements.

Softkeys
Softkeys are dynamically changing labels on the display either above or next to the function keys. They can consist of text or graphics. Their advantage as compared with insertable label strips is that they are much more flexible.

Tasks
The multitasking capability of the EPCA programming system permits several functions to be executed more or less simultaneously. These subprograms, which may only constitute a "proper" program when they are all run together, are referred to as tasks. The system can hence be designed in such a way that each key action is controlled by a separate task.

Texts
Texts are character strings that are output on the display. The appearance of a text depends on the character set which is used and on the position, size and style of the text window.

Trend plotters
Trend plotters are used to display variable values in the course of time in a diagram. The y axis shows the values, the x axis shows the time.

Variables
Variables are used to display quantities and states that may change. They are defined and placed when a project is created. The value of a variable is supplied to the terminal at runtime either by a connected control computer (actual variables) or as a user input on the terminal (set variables). Variables can be integrated in texts or output directly in a separate window.

Windows
Windows are special display areas which are used to output text, variables, bitmaps, graphic objets and bar graphs. The position and size of a window are fixed. Additional attributes such as inversion can be modified after the window has been opened. When a window is closed, the background on which it was superimposed at the time it was opened reappears. This attribute is particularly useful for messages.

9 Index

O

P

R

S

T

V

W