

KeTop T100
Handheld Terminal
User's Manual V 3.1



Automation by innovation.

Notes on This Manual

At various points in this manual you will see notes and precautionary warnings regarding possible hazards. The meaning of the symbols used is explained below.

DANGER

- **DANGER** indicates an imminently hazardous situation which, if not avoided, will result in death or serious injury.

WARNING

- **WARNING** indicates a potentially hazardous situation which, if not avoided, could result in death or serious injury.

CAUTION

- **CAUTION** indicates a potentially hazardous situation which, if not avoided, may result in minor or moderate injury.

CAUTION

- **CAUTION** used without the safety alert symbol indicates a potentially hazardous situation which, if not avoided, may result in property injury.



This symbol reminds you of the possible consequences of touching electrostatically sensitive components.

Note

Notes on use of equipment and useful practical tips are identified by the "Notice" symbol. Notices do not contain any information that draws attention to potentially dangerous or harmful functions.

© KEBA 2006

Specifications are subject to change due to further technical developments. Details presented may be subject to correction. All rights reserved.

Document: version 3.1 / material no.: 57448
Filename: t100_en.doc, last saving on: 25. 10. 2006

KEBA AG, Postfach 111, Gewerbepark Urfahr, A-4041 Linz
Tel.: ++43 / 732 / 70 90-0, Fax: ++43 / 732 / 73 09 10, E-Mail: keba@keba.com, www.keba.com

KEBA GmbH, Ulmer Straße 123, D-73037 Göppingen
Tel.: ++49 / 7161 / 97 41-0*, Fax: ++49 / 7161 / 97 41-40

KEBA Corp., 100 West Big Beaver Road, Troy, MI 48084
Tel. ++1 / 248 / 526 - 0561, Fax: ++1 / 248 / 526 - 0562, E-Mail: schr@us.keba.com

History

For the description of modification of versions see the document revision.doc.

Contents

1	Brief Description	7
	Target group of this manual.....	7
	General information.....	7
	Intended Use of the Handheld Terminal.....	7
	Construction.....	9
	Ergonomic Features.....	10
	Housing.....	10
	Operating and Display Panel.....	10
	Hardware.....	11
	Type Plate.....	12
2	Safety Precautions	13
3	General Instructions	14
	Waste disposal.....	14
	Handling of KeTop.....	14
4	Connection	16
	Cable Entrance Area.....	16
	Cable Routing in Cable Entrance Area.....	17
	Power Supply.....	18
	Emergency Stop Button and grey Stop Button.....	20
	Enabling Switch.....	22
	Foreseeable Misuse of Enabling Switch.....	28
	Ethernet.....	30
	RS-422-A.....	32
	Serial port female connector S2 for Debug Interface (RS-232-C).....	34
	PC Card Slot for PC Cards I, II, III.....	35
5	Membrane Keypad	38
	Standard.....	38
	Numbering of LEDs.....	44
6	Display	45
	Touch Screen.....	45
7	Software	46
	Windows CE.....	46
	Generation of Program for Windows CE.....	48
	KeTop API Design.....	49
	Functions.....	51
	Update API Design.....	65
	Initialising.....	66
	Program for Starting the Application and KeTop API.....	70

RDP – Connection via <u>R</u> emote <u>D</u> esktop <u>P</u> rotocol	73
KVC – KEBA Virtual Channel	80
Remote Software ActiveSync	90
Test tool (VC++ Demo).....	91
8 KeTop - Specific Operating Instructions.....	95
Setting of Date and Time.....	95
KeTop Configuration Tool (ConfigTool).....	96
Checking the Operating and Control Elements	103
Installation of Programs.....	104
Saving Files.....	104
Transferring Files.....	104
9 Options.....	107
Override Potentiometer	107
Electronic Handwheel.....	107
Illuminated Push-Button	108
Key Switch.....	108
Selector Switch.....	108
Joystick.....	108
10 Accessories	109
Wall bracket KeTop WB090 and KeTop WB095.....	109
Wall Bracket with Height Adjustment plate KeTop WBxxx	111
KeTop CB211 Connection Box.....	113
Connection Cable KeTop TTxxx.....	120
Intermediate Cable KeTop ICxxx.....	121
Download Cable KeTop XD040.....	122
11 Transport Conditions	123
12 Technical Data	124
13 CE Conformity, Directives and Standards	127
European Union Directives.....	127
Machinery Safety.....	128
Electromagnetic Compatibility	139
List of the appropriate EC directives and applied standards	150

1 Brief Description

Target group of this manual

This manual is directed at designing and project engineers which apply this product for their machines. For end users such as operators however, separate manuals must be provided by the machine and plant manufacturer.

General information

The handheld terminal KeTop T100 is a portable operating and display device with rugged design and Windows-CE compatible electronics.

Using a high-performance Intel StrongARM processor and providing a serial interface or Ethernet, the KeTop T100 is ideal for a great variety of applications (see next chapt. "Intended Use of the Handheld Terminal").

All tasks can be solved graphically and in color, and operation is intuitive using a touch screen.

Instead of rotating mass memories such as floppy disk and hard disk drives that are not suitable for rough environmental conditions, the KeTop T100 uses scalable FLASH and RAM banks.

The functionality of the handheld terminal can easily be expanded by using Compact Flash cards of the type I.

The KeTop T100 provides a Windows CE platform for applications generated with common visualization tools or with C#, Visual Basic.NET or Visual C++.

Moreover, the KeTop T100 can be connected as a client to a Win NT, Win 2000 or Windows XP server.

Through the use of optional operating and control elements, the KeTop T100 can easily be adapted to the specific application.

Intended Use of the Handheld Terminal

The intended use of the Handheld Terminal covers tasks like watching and parametrizing up to operating of machines e.g.:

- Injection moulding machine
- Robots
- Machine tools
- Textile machines

- Printing machines
- Theater backdrops
- and similar.

in normal operating modes

- Automatic

as well as

- Setting,
- Teaching,
- Testing,
- and similar.

in half automatic or manual mode.

Enabling switches and an emergency stop button are the safety elements of the device.

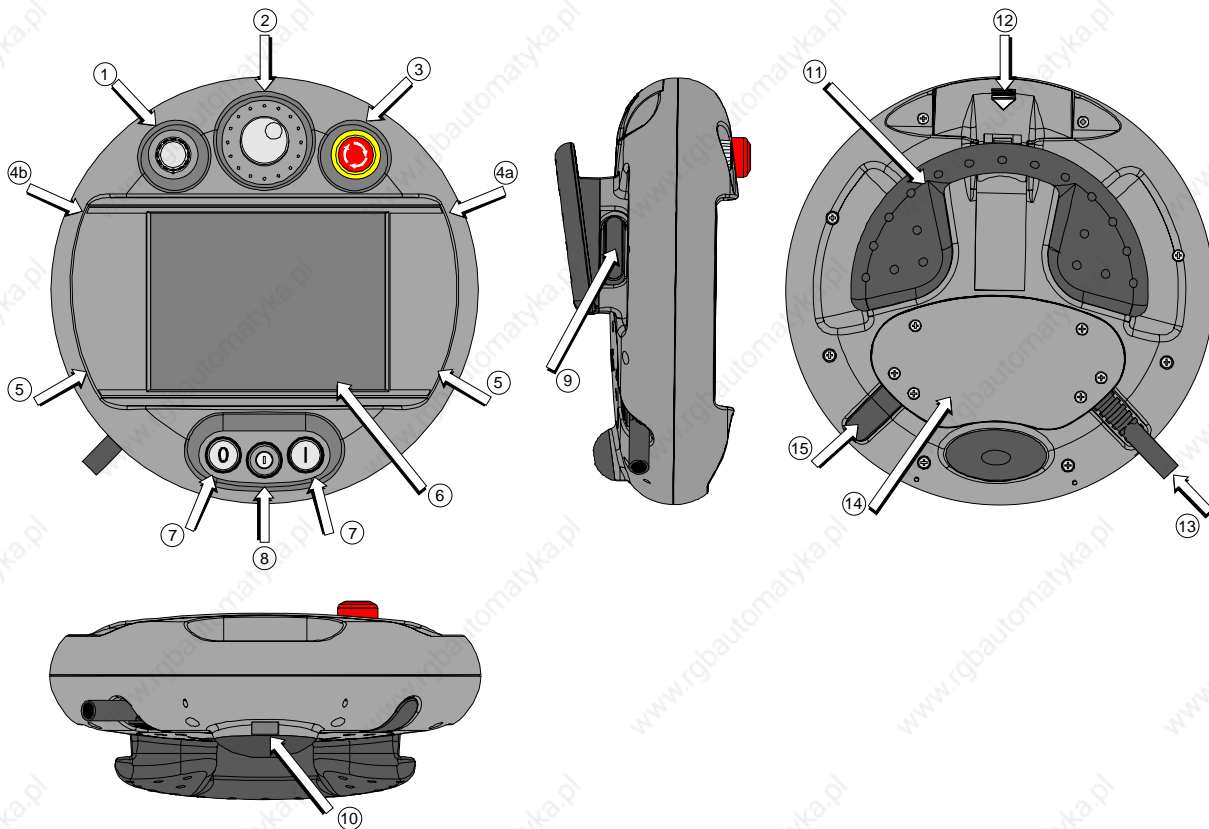
The Handheld Terminal KeTop T50-ADP is intended for fixed connection to a robot. Handheld Terminals for temporary connection must not have a red-yellow emergency stop button.

To meet the safety category 3 in accordance with EN 954-1, the safety functions are realized with 2 circuits.

The selection of the Handheld Terminal which is suitable for the machine and the projecting of the additional functions must obey the necessary hazard analyses and risk assessment bounded by law.

For intended use of the Handheld Terminal also please pay attention to chapter "CE Conformity, Directives and Standards".

Construction



- | | | | |
|----------|---|----------|---|
| 1 | override potentiometer (option) | 9 | two 3-position enabling switches (left and right), twin circuit |
| 2 | electronic handwheel or joystick (option) | 10 | multigrip handle |
| 3 | emergency stop switch (twin circuit) or stop switch (option) | 11 | PC card cover |
| 4a | 2 status LEDs (standard) | 12 | strain relief for connection cable (delivered with the cable) |
| 4b | further status LEDs (option) | 13 | cable entrance area |
| 5 | membrane keypad with tactile feedback | 14 | blind plug for cable outlet not used (to meet protection degree IP54) |
| 6 | color STN LC display with touch screen:
7.7" VGA resolution 640 x 480 or
8.4" SVGA resolution 800 x 600 (option) | | |
| 7 | 2 locations for (option):
illuminated push-button, momentary 0
illuminated push-button, momentary I
illuminated push-button, maintaining I/O | | |
| 8 | 1 location for (option):
illuminated push-button, momentary 0
illuminated push-button, momentary I
illuminated push-button, maintaining I/O
3-position key switch I-0-II / 0-I-II
3-position selector switch I-0-II / 0-I-II | | |

NOTICE: All optional operating elements are described in the chapter „Options" on page 107.

Construction of KeTop T100

Ergonomic Features

- Multigrip handle
- Round housing
- Different holding positions
- Operation by right-handed and left-handed people
- Desk top operation
- Operation in wall bracket
- The cable outlet can be on the left or right side of the housing.
- Easy-to-read display

Housing

- Vibration- and shock resistance
- Housing made of non-flammable material (UL 94-V0), impact-resistant, withstands water, cleaning agents (alcohol and tensides), oil, drilling oils, grease and lubricants
- Double-walled, extremely sturdy ABS housing. Drop-tested on industrial flooring from a height of 1.0 m (39.4 in).

Operating and Display Panel

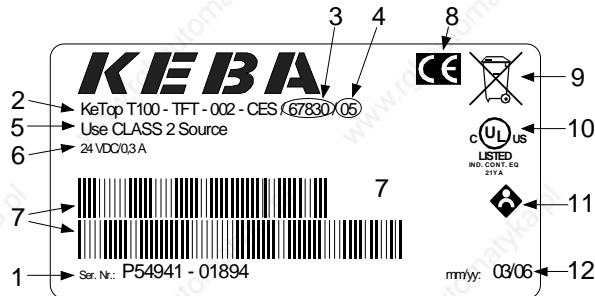
- Membrane keys with tactile feedback
- 2 status LEDs
- Buzzer in upper part of housing
- Resistive touch screen, to be operated with finger or stylus
- Backlit color STN LC display
7.7" VGA (640x480 pixels) or
8.4" SVGA (800x600 pixels)

Hardware

- CPU Intel StrongARM SA-1110/206 MHz
- Memory:
 - DRAM: 16 MB, FLASH: 32 MB or
 - DRAM: 64 MB, FLASH: 64 MB
- Interfaces:
 - Ethernet
 - Serial interface:
RS-422-A (option),
RS-232-C (debug interface in device)
 - PC card slot, with dust protection cover to guarantee IP54 when the device is closed:
For connection of various PC card components, type I-III (flash cards, network cards, etc.).

Type Plate

Sample of a KEBA type plate:



- 1 Serial number
- 2 Material name
- 3 Material number
- 4 Revision number of device
- 5 Further information (optional)
- 6 Technical data (here: power supply)
- 7 Bar code
- 8 CE marking
- 9 Advice for electronic scrap regulation
- 10 UL marking
- 11 NSBIV marking
- 12 Date of production (month/year)

Description of a KEBA type plate

2 Safety Precautions

The device was developed, manufactured, tested and documented in accordance with the applicable safety standards. If you follow the instructions regarding safety and use as described in this manual (see chap. „Intended Use of the Handheld Terminal“), the product will, in the normal case, neither cause personal injury nor damage to machinery and equipment.

The instructions contained in this manual must be precisely followed in all circumstances. Failure to do so could result in the creation of potential sources of danger or the disabling of safety features integrated in the handheld terminal.

Apart from the safety instructions given in this manual, the safety precautions and accident prevention measures appropriate to the situation in question must also be observed.

WARNING

- **For the right projecting of the Handheld Terminal the manufacturer must enforce a hazard and risk analysis. The following safety aspects must be considered:**
 - Right cable length for limitation of workspace.
 - Is an emergency stop button necessary and permissible?
 - Is the safety category for the application sufficient?
- **The device may be operated in faultless condition only and the operating instructions must be observed.**
- **The operator must have a sufficient educational level and must know details of intended use described in the user's manual.**
- **The safety advices in the following chapters must be considered absolutely.**
- **Further informations to safety and EMC are included in chapt. „CE Conformity, Directives and Standards“. They must be considered absolutely.**

3 General Instructions

Waste disposal

Observe the national regulations when disposing of electronic components!

Handling of KeTop

You have chosen a high-quality KeTop that is equipped with highly sensitive state-of-the-art electronics.

To avoid malfunctions or damage through improper handling, follow these instructions during operation.

CAUTION

- Turn off the power supply before opening the cable entrance area of the KeTop. Otherwise the components could be destroyed or undefined signals could occur.
- Make sure that nobody can fall over the cable to avoid that the device falls to ground.
- Take care not to squeeze and thus damage the cable with any object.
- Do not lay the cable over sharp edges to avoid damaging the cable sheath.
- If you do not use the device, hang it into the wall bracket KeTop WBxxx provided for storage.
- Do not lay down the device with the operating side facing down to avoid damaging the operating elements.
- Never lay the device onto unstable surfaces. It could fall to ground and thus be damaged.
- Never lay the device close to heat sources or into direct sunlight.
- Avoid exposing the device to mechanical vibrations, excessive dust, humidity or to strong magnetic fields.
- Never clean the device, operating panel and operating elements with solvents, scouring agent or scrubbing sponges. For cleaning the device, use a soft cloth and a bit of water or a mild cleaning agent.
- Make sure that no foreign objects or liquids can penetrate into the device. Check at regular intervals the protective covers of the device, if all housing screws are firmly tightened and if the housing or the cable entry is damaged.
- If the device shows any defect, please send it, including a detailed error description, to your supplier or the relevant after-sales service office.

- If the KeTop is equipped with a touch screen, then operate the touch screen with fingers or use a touch-pen. Never use sharp objects (e.g. screwdriver,...) for operating the touch screen. This could damage the touch screen.



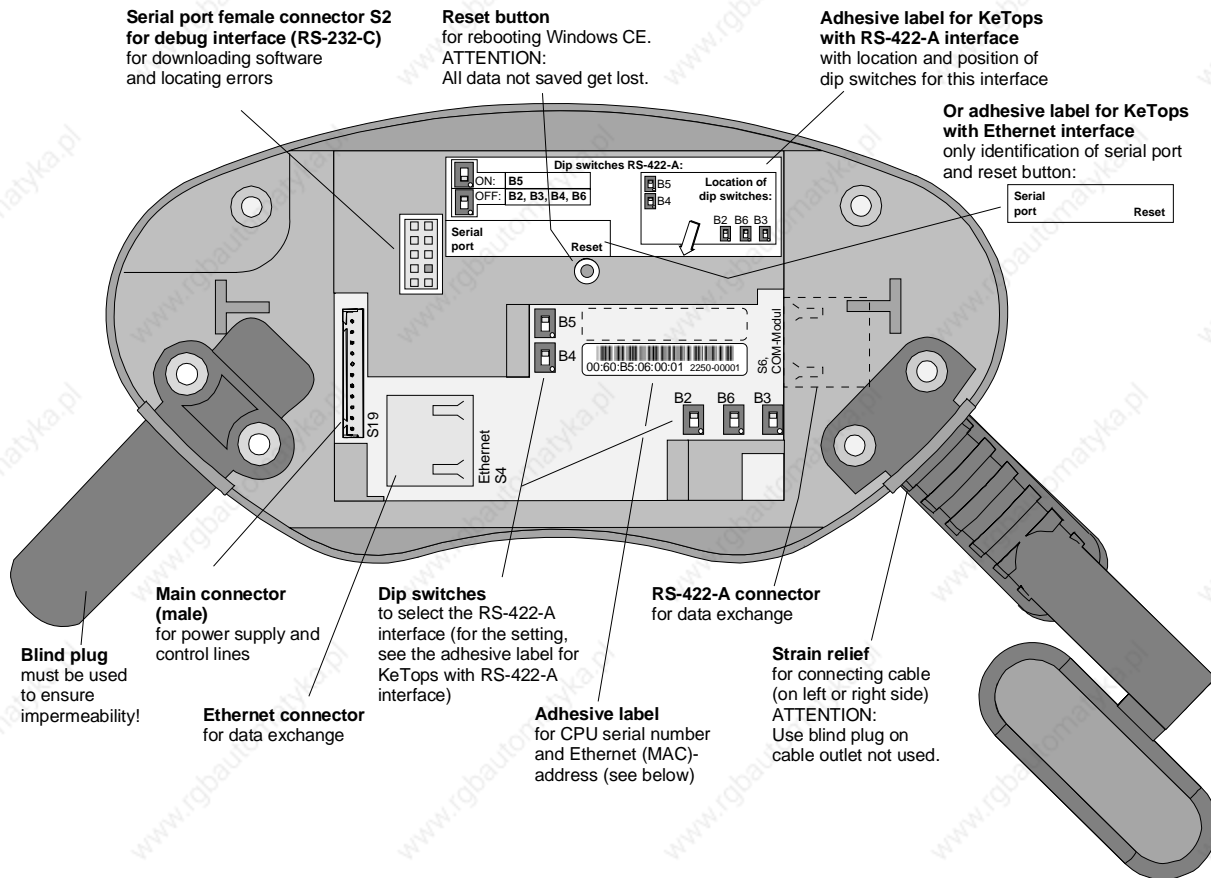
When the cable entrance area is open, the KeTop is sensitive to electrostatic discharge.

Note

- *Should the device fall to the ground, make sure that the PC card cover is correctly closed.*
- *If a PC card is used in the KeTop, make sure that the card is correctly seated in case of a severe shock, e.g. if the KeTop has been dropped. (Although the PC card cover remains closed after a shock, the card itself can become dislodged from its socket in the mounting slot, so that the electrical contacts are interrupted.)*

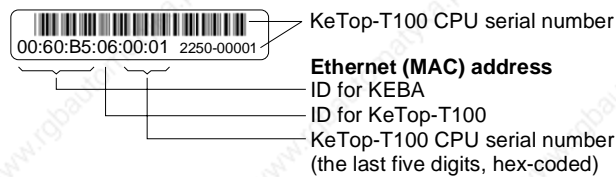
4 Connection

Cable Entrance Area



Cable entrance area of KeTop T100

Adhesive label for CPU serial number and Ethernet (MAC)-address



Adhesive label for CPU serial number and Ethernet-address

Cable Routing in Cable Entrance Area

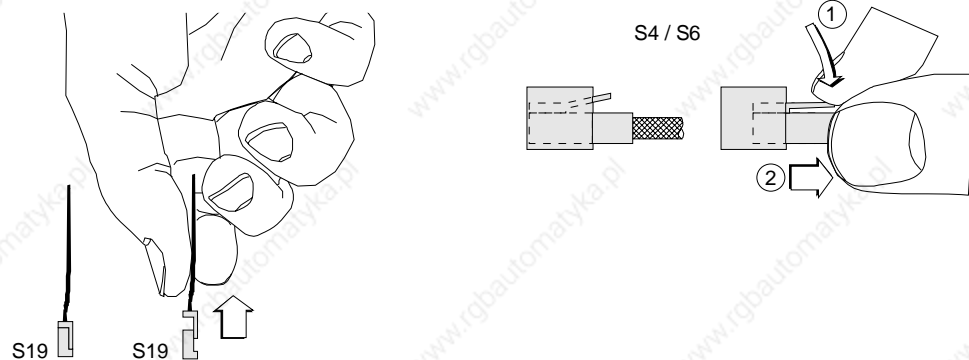
After opening the cable entrance area, the connecting lines can be routed as described in the following chapters. Before opening the KeTop please pay attention to the following safety instructions:

Instructions for opening the cable entrance area:

- Lay the KeTop with the display facing down onto a plane and clean table (preferable on ESD pad) and take care not to damage the KeTop and its operating elements.
- For opening and closing the cable entrance area use the following type of screwdriver: „Phillips size 2“.

Instructions for modifications in the cable entrance area:

- Unplug the main connector (S19) by pulling on its wires with your fingers. Do not use any sharp objects.
- For unplugging the RJ-45 jack (S4 / S6), actuate the locking lever:

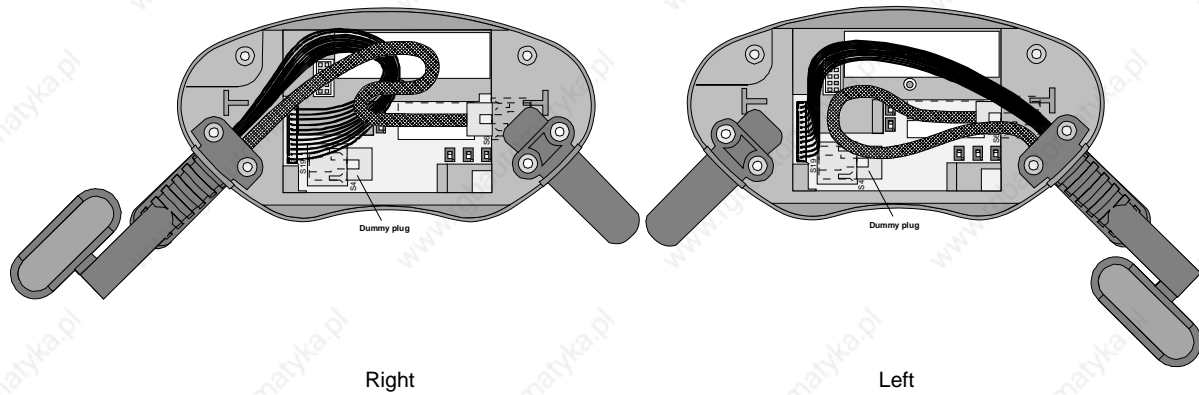


Make sure that the connectors S19 and S6 correctly snap in when you plug them in. Otherwise the emergency stop functionality (S19) or the correct shielding (S4/S6) might not be given any more.

Instructions for closing the cable entrance area: Make sure that

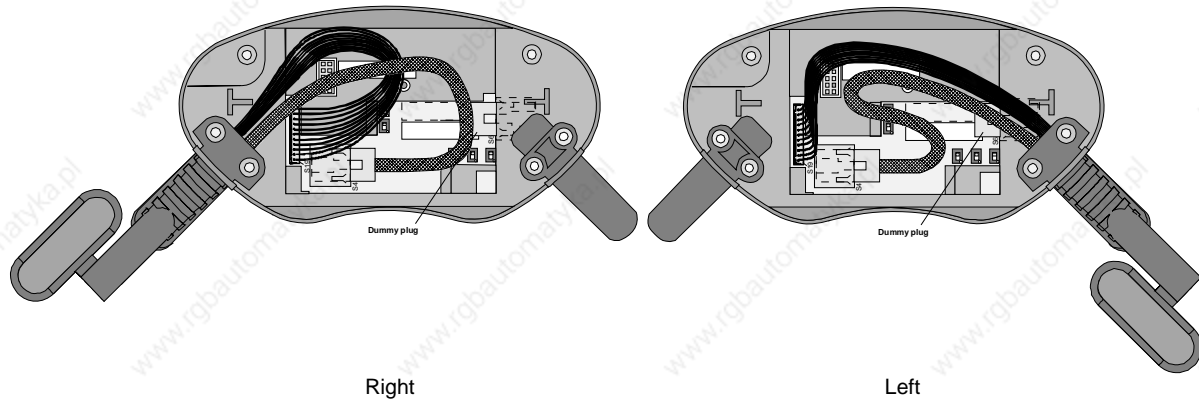
- the sealing is clean, not damaged and correctly positioned in the cable entrance area,
- no cables are squeezed in,
- the cover of the cable entrance area is attached again with all 6 screws (torque: 0.4 bis 0.5 Nm). Otherwise the protection degree cannot be guaranteed.

Cable outlet RS-422-A



Cable outlet on left and right side if the RS-422-A interface is used.

Cable outlet Ethernet



Cable outlet on left and right side if the Ethernet interface is used.

Power Supply

<p>⚠ WARNING</p>
<ul style="list-style-type: none"> • The device meets the safety class III in accordance with EN61131-2 and EN50178. The 24V power supply for the equipment must be guaranteed through safe isolation of the extra low-voltage circuits from dangerous-contact voltage circuits (e.g. by safety transformers or similar facilities).

CAUTION

- The power supply circuit must be protected with a fuse of a maximum of 3.15 A

Notice

- When planning the power supply, take into account the voltage drop on the KeTop TTxxx connection cable:

Specification of power supply lines in the KeTop TTxxx connection cable:

Cross section: AWG24 (0.24mm²)
Material: zinc-coated copper strand
Line resistance: ≤ 90 Ohm/km (≤ 145 Ohm/mile)

- The nominal supply voltage directly on the handheld terminal (without KeTop TTxxx connection cable) is:
+24 VDC (supply voltage range: 18-32 VDC).
- Power consumption:
7.2 W (400 mA at 18 V DC, 300 mA at 24 V DC)
- Maximum interruption time of supply voltage:
≤ 10 ms (lt. IEC 61131)

Emergency Stop Button and grey Stop Button

The emergency stop button respectively the grey stop button used on the KeTop features two circuits. The contacts are normally closed.

The red-yellow emergency stop button of the handheld terminal meets the requirements of the EN 418. It must be designed as an emergency stop of category 0 or category 1 (see EN 60204-1 chapter 9.2.5.4.2) on the basis of the risk assessment for the machine. The connection of the positive-break contacts to an appropriate monitoring system must meet the safety category which is defined by means of the risk assessment (in accordance with EN 954-1) of the machine.

Optionally the KeTop T50-ADP is also available with a grey stop button instead of the red-yellow emergency stop button. In principle the grey stop button has the same functionality as the red-yellow emergency stop button. If the handheld terminal is not plugged in, the grey color of the stop button should avoid the usage of the non-effective (emergency) stop button in dangerous situations.

The grey stop button fulfils also all mechanical aspects of the EN 418 and differs from the emergency stop button only by its color.

WARNING

- **Not fully functional emergency stop devices may have fatal consequences! Emergency stop switches which are red-yellow marked must be effective under all circumstances in all operating modes of a machine or plant.**

Store handheld terminals with not operational red-yellow emergency stop switches on a place where the operator cannot see it, so that he can not mistake the device.

Handheld terminals which are plugged in and out frequently for temporarily use, must not have a red-yellow emergency stop switch. Such devices must be equipped with a grey stop switch.

- **Resetting an activated emergency stop facility must not result in uncontrolled start-up of machines or installations.**
- **The emergency stop button does not replace other safety facilities.**
- **The emergency stop button on the handheld terminal does not replace the emergency stop buttons to be mounted directly on the machine**
- **Some mechanical errors in emergency stop switches can be recognized at operation only.**

Test the function of the emergency stop switch when the device had been exposed to mechanical shock (e.g. it had been fallen on the ground)

Note for maintenance:

Additionally the emergency stop switch must be tested cyclic (6 monthly).

Watch the machine stopping after the emergency stop switch had been pushed.

- For further informations to emergency stop switch observe chapt. "CE Conformity, Directives and Standards".

Connection values

- Connection voltage: 24 VDC
- Maximum current: 500 mA
- Minimum current: 10 mA

Enabling Switch

The KeTop is equipped with two enabling switches, one at the left and one at the right side of the device. This allows a left- and right-hand operation of the enabling switch. Both enabling switches are equivalent and parallel switched. So for enabling only one of both enabling switches must be activated.

The enabling switch consists of a 3-position operating element and an separated evaluation electronics. An essential feature are the continuous two-channel circuits beginning from the actuating elements up to the connecting terminals. For the evaluation circuits different technologies and circuits are used. Due to the electronic switching contacts, their lifetime does not depend on the load provided the nominal values of the load (ohmic, inductive and capacitive) are not exceeded.

Functioning

The actuating element consists of two symmetrically arranged slides. The position of these slides is detected by electrical switches and transmitted to the evaluation electronics.

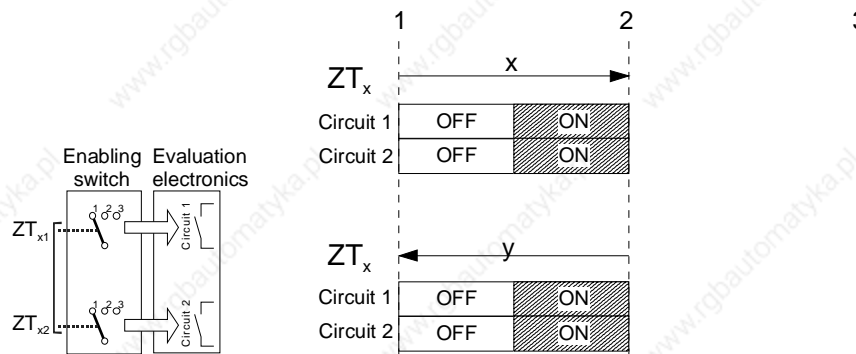
Positions of enabling switch:

Position	Function	Enabling switch	Contacts
1	home position	is not pressed	enabling outputs are open
2	enabling	is pressed	enabling outputs are closed
3	panic	is pressed strong	enabling outputs are open

For the enabling switch, the following switching sequences are possible:

Enabling

Home position \xrightarrow{x} enabling \xrightarrow{y} home position



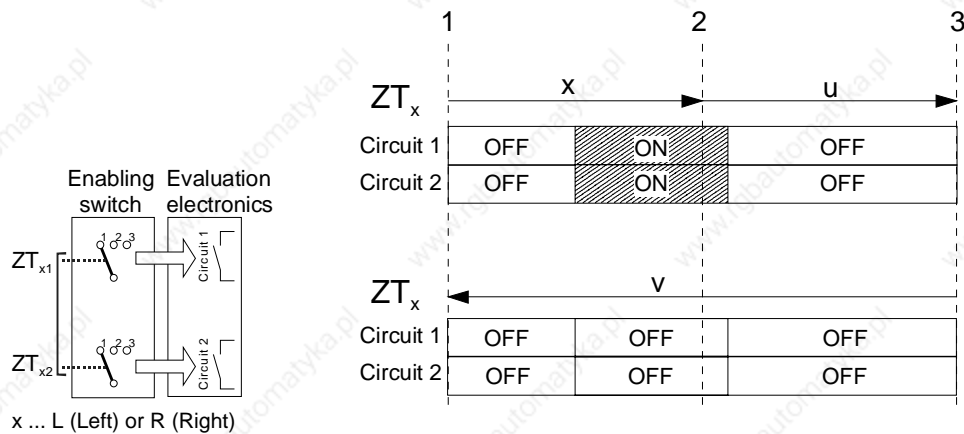
x ... L (Left) or R (Right)

Contact travel diagram for enabling

Panic

The pushing of the actuating elements into the panic position is evaluated in a way that the enabling position is skipped when the actuating elements are released.

Home position \boxed{x} enabling \boxed{u} panic \boxed{v} home position



x ... L (Left) or R (Right)

Contact travel diagram for panic

Notice

- At the KeTop, the enabling switches always feature two circuits.

To meet the safety category 3 in accordance with EN 954-1:1996, the enabling switch must be realized with 2 circuits.

The safety category 3 means, that one failure must not lead to the loss of the safety function, and whenever possible, the single failure is detected.

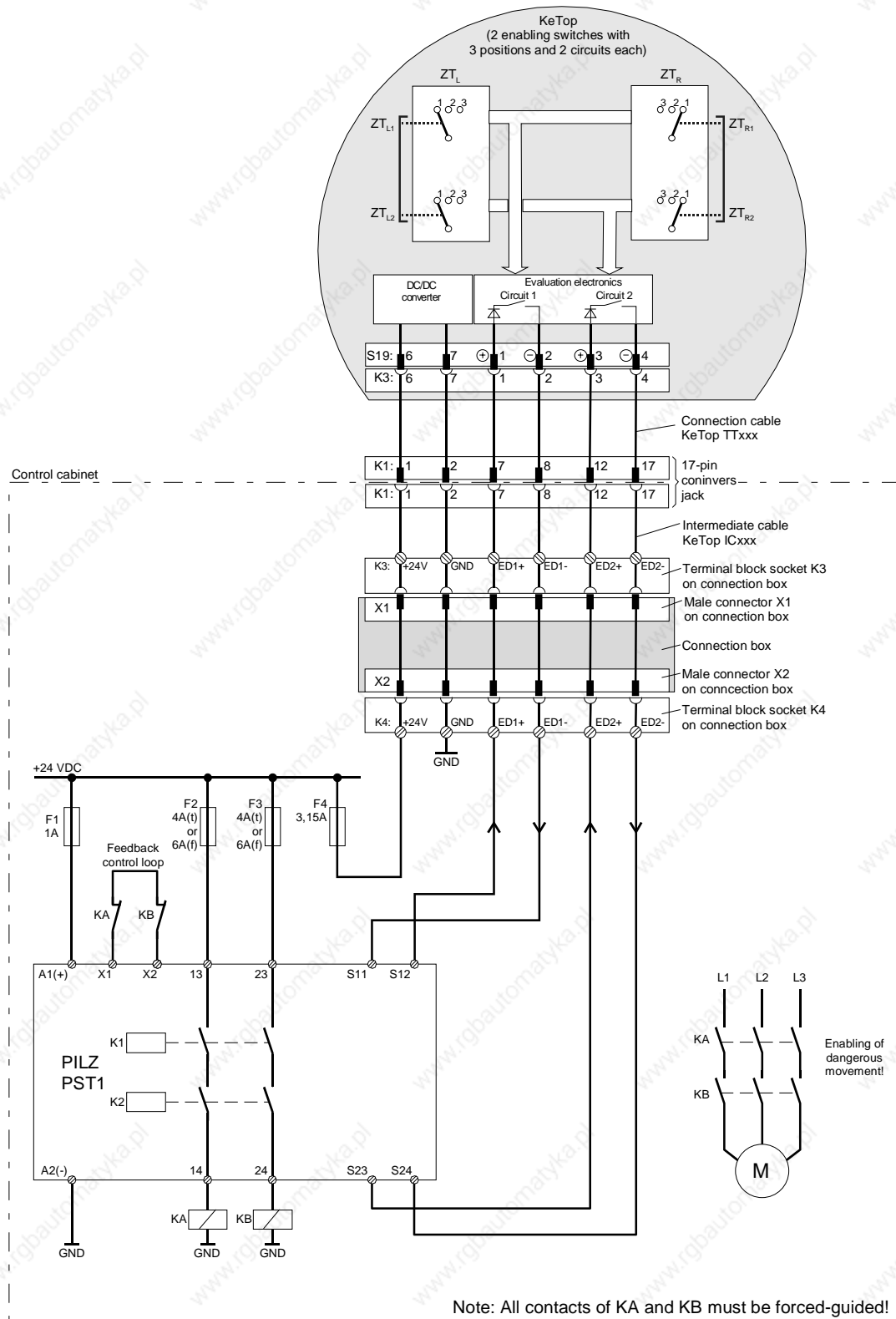
The chapters „Example of Connection with PILZ PST1 Control Relay“ and “Example of Connection with ELAN SRB-NA-R-C.27/S1 Control Relay” show how the **safety category 3** can be fulfilled with the KeTop and its safety-related parts. The entire concept of the machine must be laid out according to the principles of safety category 3.

The monitoring of the simultaneity by the control relay is necessary, because otherwise maybe a failure culmination would not be recognised and this would cause the loss of safeness:

If one circuit of the enabling device switches to the enabled state as a result of a failure and after some time the second circuit also switches to the enabled state as a result of an failure, then no de-energisation by the enabling device would be possible.

Further the EN 60204-1:1997 prescribes that the enabling switches shall be connected to a category 0 stop or a category 1 stop, that means the energy must be switched off.

Example of Connection with PILZ PST1 Control Relay



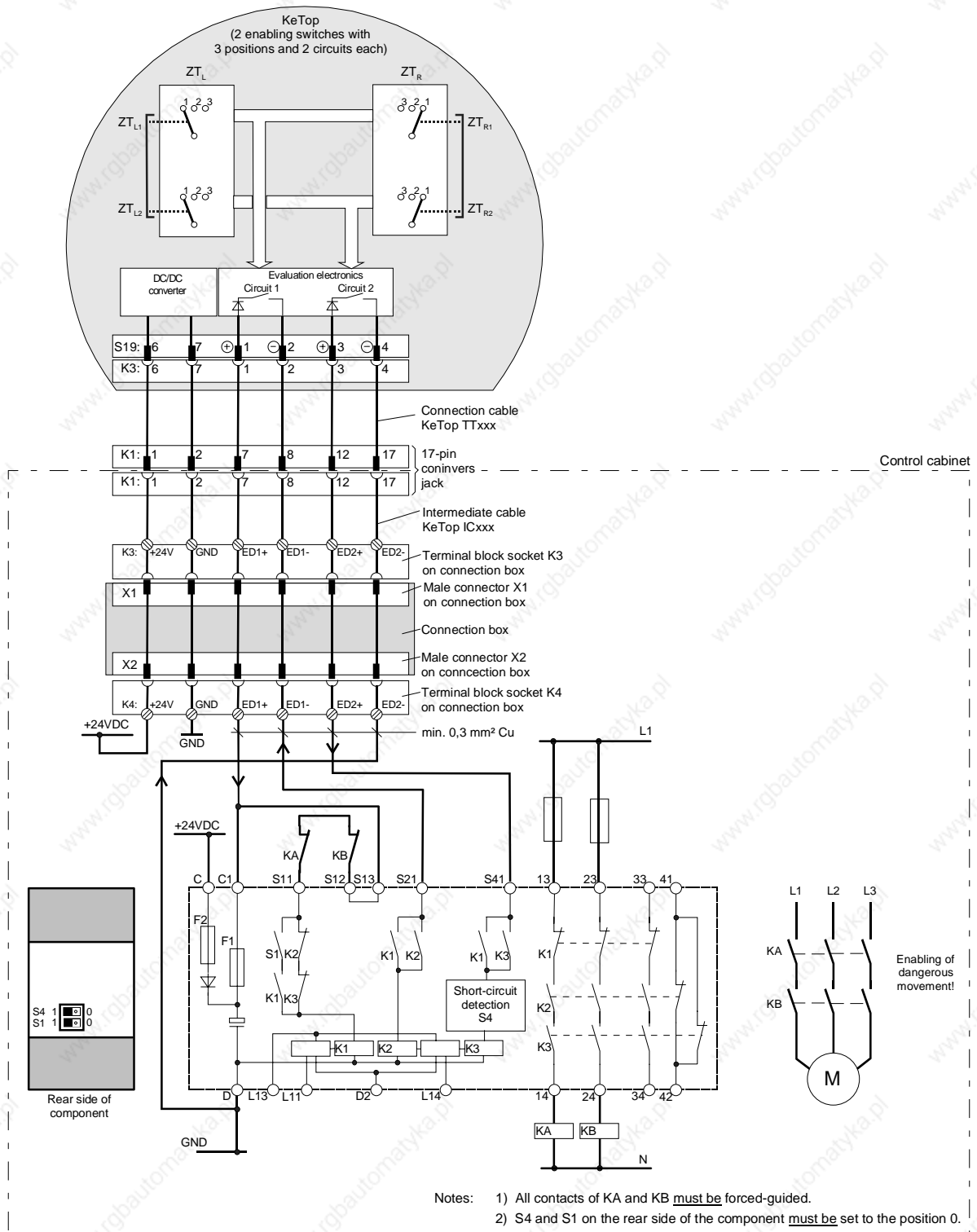
Suggested wiring of enabling switches to fulfill safety category 3 with PILZ control relay. Also follow the instructions described in the PILZ operating manual about the PST1 device.

Functional procedure:

- Only if both channels are activated „simultaneously“ (by pressing one of the enabling switches) both output relays K1 and K2 will energize and the output contacts 13-14 and 23-24 will close.
- The output relays K1 and K2 will not energize if
 - only one enabling channel is activated (in case of a defect),
 - the tolerance value for the simultaneity period is exceeded,
 - the feedback control loop X1-X2 is open.
- If one enabling channel is deactivated after being simultaneously activated (by releasing the enabling switch or in case of a defect), the output relays K1 and K2 will return to their initial position again. The forced-guided output contacts 13-14 and 23-24 will open. The output relays will energize again only after both enabling channels have been deactivated and simultaneously activated once again.

In this way the enabling switches avoid that one single error makes the safety function inoperational. A single error will be recognized at the next cycle at the latest.

Example of Connection with ELAN SRB-NA-R-C.27/S1 Control Relay



Suggested wiring of enabling switches for safety category 3 with the ELAN SRB-NA-R-C.27/S1 control relay. In addition follow the instructions of the operating manual about the SRB-NA-R-C.27/S1.

Functional procedure:

- Only if both channels are activated „simultaneously“ (by pressing one of the enabling switches) both output relays K2 and K3 will energize and the output contacts 13-14, 23-24 and 33-34 will close.
- The output relays K2 and K3 will not energize if
 - only one enabling channel is activated (in case of a defect),
 - the tolerance value for the simultaneity period is exceeded,
 - the feedback control loop S11-S12 is open.
- If one enabling channel is deactivated after being simultaneously activated (by releasing the enabling switch or in case of a defect), the output relays K2 and K3 will return to their initial position again. The forced-guided output contacts 13-14 and 23-24 will open. The output relays will energize again only after both enabling channels have been deactivated and simultaneously activated once again.

In this way the enabling switches avoid that one single error makes the safety function inoperational. A single error will be recognized at the next cycle at the latest.

Foreseeable Misuse of Enabling Switch

Foreseeable misuse means the not allowed fixing of the enabling switch in the enabling position. The foreseeable misuse of the enabling switch must be restricted. The following measures causing the stop of the machine in the manual mode are recommended:

- Inquiry of the enabling switch when turning on the machine/plant and inquiry of the enabling switch when changing the operating mode from automatic to manual (The enabling switch must not be in the enabling position.).
- The enabling switch must be released within a defined period and pushed into the enabling position again. The length of the period must be defined according to the activity.

Technical Data of Switching Elements of Enabling Switches

Nominal voltage	24 V DC (typ.) 32 V DC (max.)
Nominal current	500 mA (typ.)
Short-circuit current	circuit 1: max. 1,9 A circuit 2: max. 600 mA
Max. inductive load (at 500 mA)	circuit 1: max. 1H circuit 2: max. 320 mH
Max. capacitive load	circuit 1: no limit since the transistor is protected thermally circuit 2: max. 500 µF

The switching elements of the enabling switches are protected against reversed polarity. The outputs of both circuits are protected against short circuits and excess load.

Circuit 1: thermal protective circuit

Circuit 2: fold back line

WARNING

- **The enabling switch is only suitable as safety function if the operator activating the enabling switch recognizes the dangerous situation in time so that he can immediately take the necessary measures to avoid such situations. As additional measure reduced speed of the movement can be necessary. The allowed speed must be determined by means of a risk assessment.**
- **The enabling switch is only used to enable commands for performing dangerous movements. The commands themselves must be activated by a separate operating element (key on handheld terminal).**
- **Only the person who operates the enabling switch is allowed to work in the dangerous area.**
- **For further informations regarding the enabling switch please pay attention to chapter „CE Conformity, Directives and Standards“.**

Ethernet

The standard version of the KeTop is equipped with an Ethernet interface. This interface is based on the 10BaseT specification and suitable for the half-duplex mode.

The data communication for these interfaces takes place via the Ethernet connector S4 in the cable entrance area of the KeTop.

The following interface parameters are defined and cannot be changed:

- 10 Mbaud
- TCP/IP protocol

The Ethernet interface is selected under Windows CE as follows:

Start -> Settings -> Control Panel -> Network and Dial-up Connections:
Here you can select the Ethernet on-board interface SMSC91C9X.

WARNING

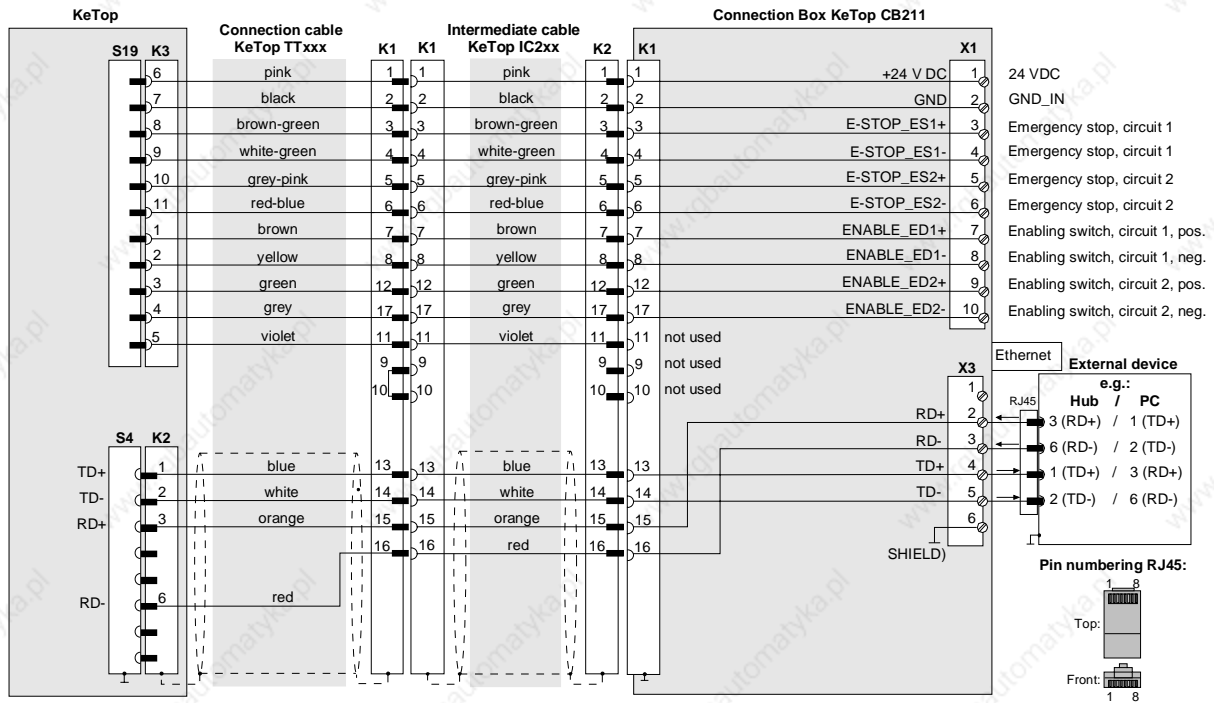
- If the KeTop and the control do not communicate via a point-to-point connection, it may happen that the keypad data, for example, are transmitted with a delay.

Therefore it is advisable to establish the connection between the control and the KeTop only via an Ethernet switch which enables a point-to-point connection.

If an Ethernet PC card is inserted into the PC card slot, the internal Ethernet interface must be deactivated.

The selection between the internal Ethernet or the PC card Ethernet interface is made in the „ConfigTool“. See chapter „KeTop Configuration Tool (ConfigTool“ on page 96).

The positions of the Dip switches in the cable entrance area are not relevant for this interface.



Wiring diagram: KeTop T100 with Ethernet via connection box KeTop CB211

RS-422-A

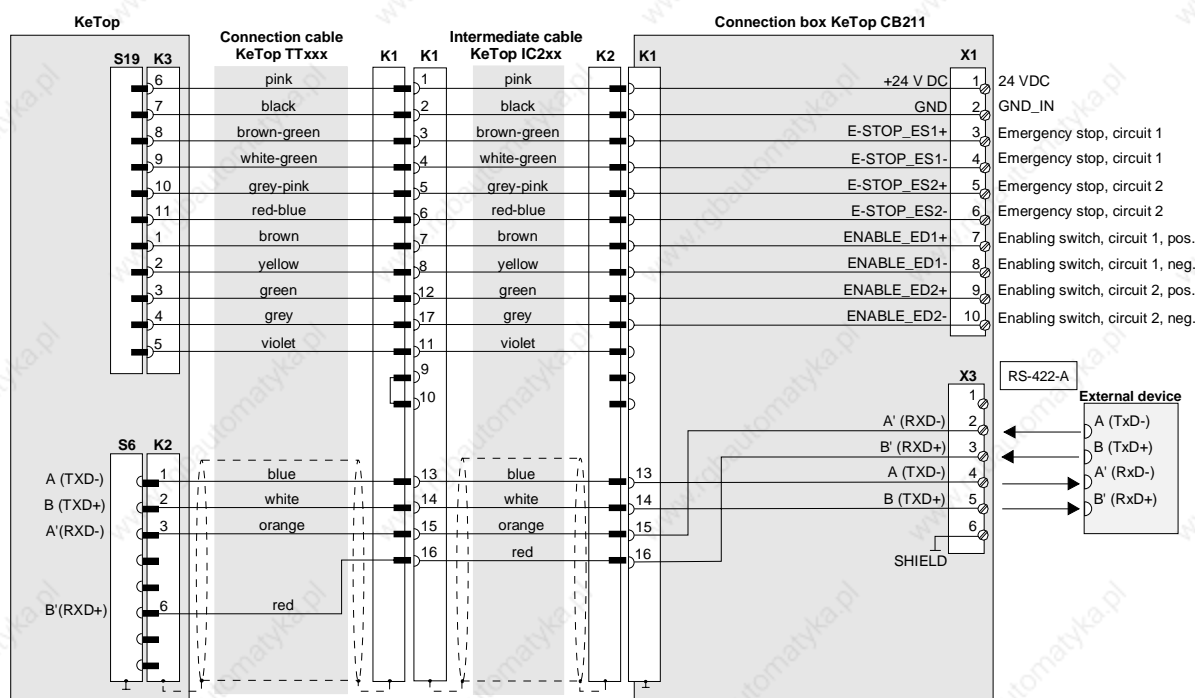
The serial data communication for this interface requires the installation of an optional module in the KeTop.

The communication takes place via the COM module connector S6 in the cable entrance area of the KeTop.

Notice

- *The simultaneous use of the Ethernet interface and the RS-422-A interface is not possible.*

The RS-422-A interface assigned to the COM 5 interface port is in the software. The interface parameters are set via the WIN32API in the Windows operating system.



Dip switches for RS-422-A:

B2	B3	B4	B5	B6
OFF	OFF	OFF	ON	OFF

Wiring diagram: Connection of KeTop T100 with RS-422-A via connection box KeTop CB211

General information about the RS-422-A interface

- The A terminal of the generator shall be negative with respect to the B terminal for a binary 1 (MARK or OFF) state.
- The A terminal of the generator shall be positive with respect to the B terminal for a binary 0 (SPACE or ON) state.

To identify the lines, the voltage between the lines A and B can be measured by means of a voltmeter.

Serial port female connector S2 for Debug Interface (RS-232-C)

Using the „Boot-Loader“ software, the „serial port“ interface can be used for debugging and for downloading software. Using the remote software ActiveSync, it can be used for adjusting and transmitting data from and to a PC.

For that purpose, the download cable KeTop XD040 is available.

The following interface parameters are defined and cannot be changed:

- 38400 baud
- 8 data bits
- 1 stop bit
- No parity
- No handshake

The debug interface is assigned to the COM 1 interface port in the software.

The positions of the Dip switches in the cable entrance area are not relevant for this interface.

PC Card Slot for PC Cards I, II, III

The following list gives an overview about the tested PC cards and their manufacturers.

These PC cards are recommended for the use in the KeTop.

ATA flash cards:

- Manufacturer: Kingston, SanDisk, Kingmax, Viking
Type: All PC cards type II
Memory sizes: 8 MB and more

PC card adapter for CompactFlash (CF) cards:

- Manufacturer: Hama
Type: PC card adapter CF type II, Hama art. no. 56949
- Manufacturer: Kingston
Type: CF/ADP Compact Flash PC card adapter
- Manufacturer: Ultron
Type: UPA-150 PCMCIA adapter for compact flash card

Ethernet cards:

- Manufacturer: Socket (<http://www.socketcom.com>)
Name: LP-E Ethernet Card
Type: PC cards type II
Transmission rate: 10 Mbit/s
Special features: NE2000 compatible
- Manufacturer: Orinoco (<http://www.wavelan.com>)
Name: WaveLan Silber IEEE 802.11
Transmission rate: 11 Mbit/s
Special features: Wireless LAN-Card



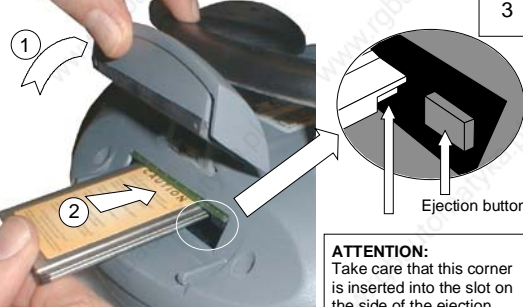


Modem cards:

- Manufacturer: TDK
(<http://www.tdksys.com/Products/5660.html>)
Name: Global Freedom 5660
Type: PC card type II


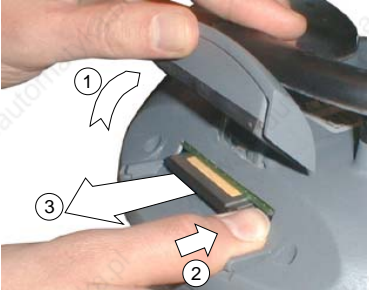
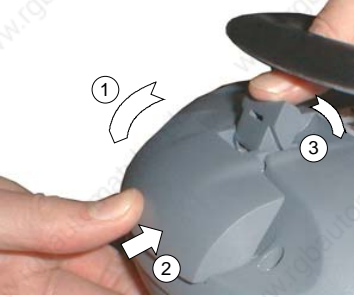

Notice

- *The KeTop does not support CardBus cards.*
- *PC cards which use 12 V programming voltage are not supported by the KeTop.*
- *SRAM cards are not supported by Windows CE.*

Inserting the PC Card

 <p>1</p> <p>Lay the KeTop with the display facing down onto a plane and clean table (preferably on ESD pad) and take care not to damage the KeTop and its operating elements.</p>	 <p>2</p> <p>Unlock the PC card cover as shown (until the locking lever is released).</p>
 <p>3</p> <p>1) Open the cover. 2) Insert the PC card as shown.</p> <p>ATTENTION: Take care that this corner is inserted into the slot on the side of the ejection button.</p> <p>Ejection button</p>	 <p>4</p> <p>ATTENTION: Check the condition and the position of the sealing before closing the PC card cover.</p> <p>Insert the PC card until it locks in and the ejection button jumps out.</p>
 <p>5</p> <p>1) Close the cover. 2) and 3) Lock the cover as shown.</p>	 <p>6</p> <p>Press down the cover until it snaps in completely to meet the protection degree IP54.</p> <p>Must snap in completely!</p>

Removing the PC Card

 <p>1</p> <p>Unlock the PC card cover as shown (until the locking lever is released).</p>	 <p>2</p> <p>1) Open the PC card cover. 2) Press the ejection button of the PC card slot. 3) Remove the PC card.</p>
 <p>3</p> <p>1) Close the cover. 2) and 3) Lock the cover as shown.</p>	 <p>4</p> <p>Must snap in completely!</p> <p>Press down the cover until it snaps in completely to meet the protection degree IP54.</p>

5 Membrane Keypad

Standard

General Membrane Keypad

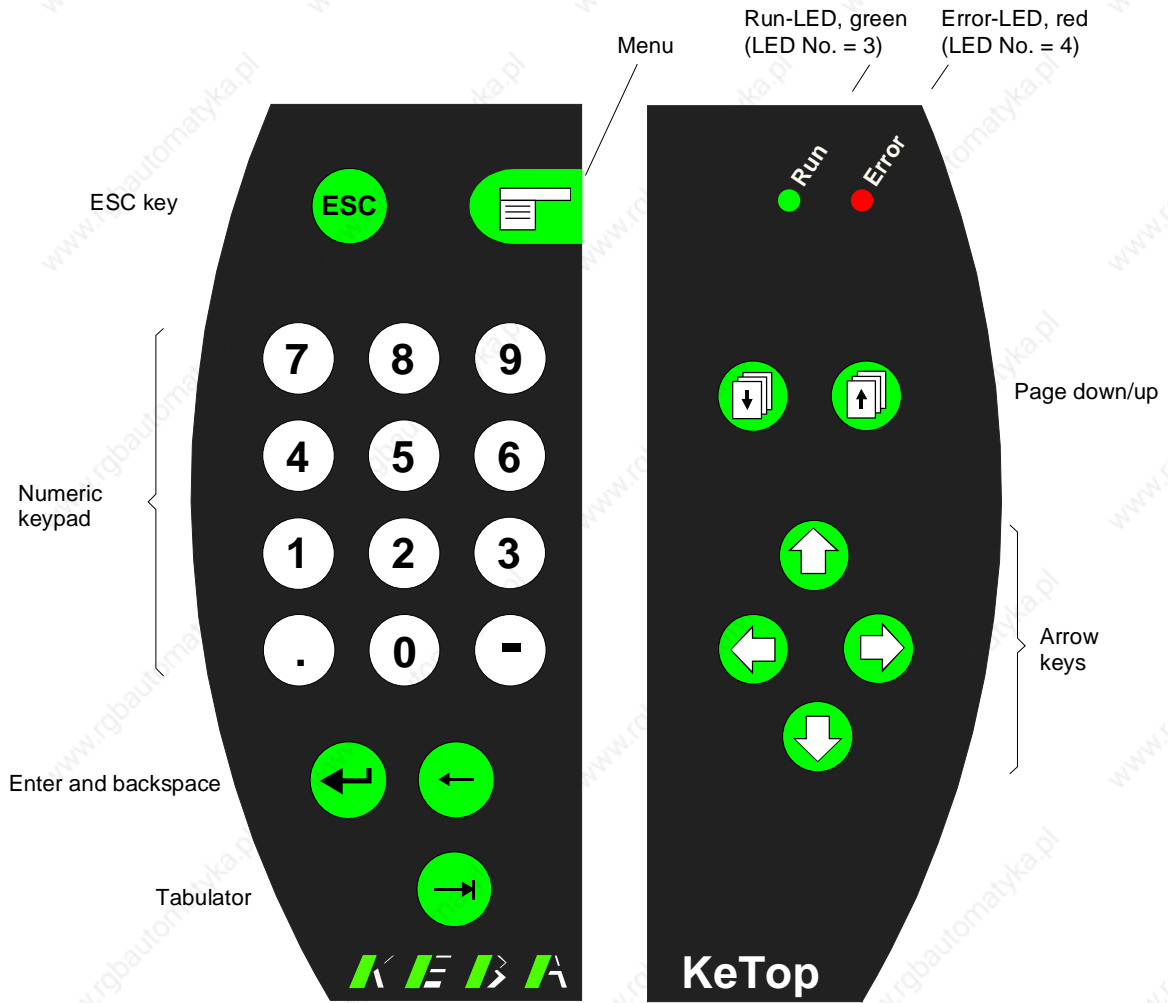
Keypad Assignment

The keypad assignment is stored in the registry and defined for the general membrane keypad as follows:

Key Labelling	Windows Virtual Key Code	Hex	Dez
ESC	VK_ESCAPE	0x1B	27
Menü	VK_MENU	0x12	18
Numpad figures 0-9	VK_0-9	0x30 - 0x39	48 - 57
Dot (.)	VK_PERIOD	0xBE	190
Minus (-)	VK_OEM_MINUS	0xBD	189
Enter	VK_RETURN	0x0D	13
Backspace	VK_BACK	0x08	8
Tabulator	VK_TAB	0x09	9
Page Down	VK_NEXT	0x22	34
Page Up	VK_PRIOR	0x21	33
Cursor Up	VK_UP	0x26	38
Cursor Left	VK_LEFT	0x25	37
Cursor Right	VK_RIGHT	0x27	39
Cursor Down	VK_DOWN	0x28	40
Illuminated push-button, left	VK_F13	0x7C	124
Key switch/selector switch, left	VK_F14	0x7D	125
Key switch/selector switch, right	VK_F15	0x7E	126
Illuminated push-button, right	VK_F16	0x7F	127

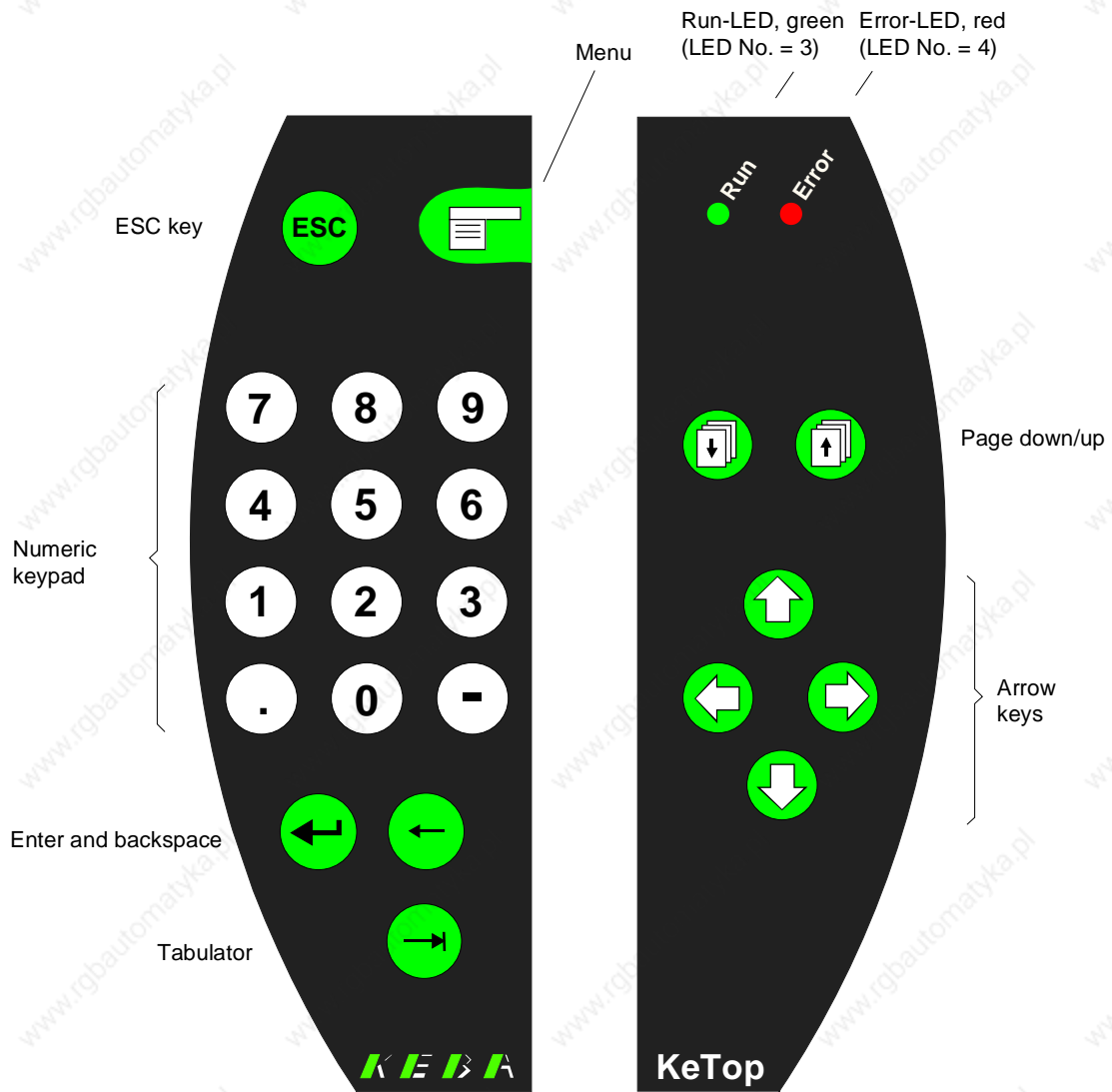
Depending on the size of the display, two general membrane keypads are available:

a) 7.7" display (640x480 pixels)



General membrane keypad for 7.7" display

b) 8.4" display (800x600 pixels)



General membrane keypad for 8.4" display

Robotics Membrane Keypad

Keypad Assignment

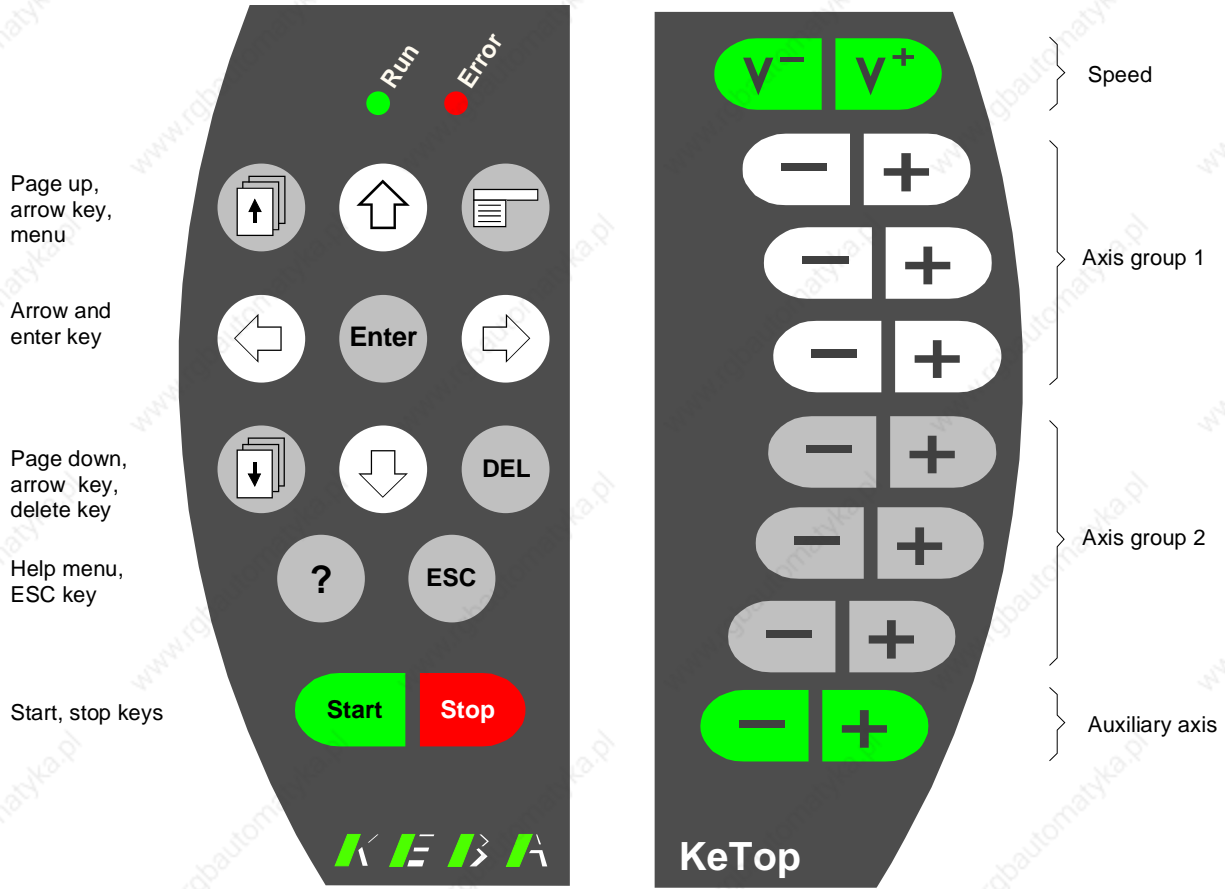
The keypad assignment is stored in the registry and defined for the robotics membrane keypad as follows:

Key Labelling	Windows Virtual Key Code	Hex	Dez
Page Down	VK_NEXT	0x22	34
Page Up	VK_PRIOR	0x21	33
Cursor Up	VK_UP	0x26	38
Cursor Left	VK_LEFT	0x25	37
Cursor Right	VK_RIGHT	0x27	39
Cursor Down	VK_DOWN	0x28	40
Menu	VK_MENU	0x12	18
Enter	VK_RETURN	0x0D	13
Delete	VK_DELETE	0x2E	46
ESC	VK_ESCAPE	0x1B	27
Help (?)	VK_SHIFT + VK_SLASH	0x10 + 0xBF	16 + 191
Start	VK_1	0x31	49
Stop	VK_0	0x30	48
V-	no VK Code defined	0xC1	193
V+	no VK Code defined	0xC2	194
A1-	no VK Code defined	0xC3	195
A1+	no VK Code defined	0xC4	196
A2-	no VK Code defined	0xC5	197
A2+	no VK Code defined	0xC6	198
A3-	no VK Code defined	0xC7	199
A3+	no VK Code defined	0xC8	200
A4-	no VK Code defined	0xC9	201
A4+	no VK Code defined	0xCA	202
A5-	no VK Code defined	0xCB	203
A5+	no VK Code defined	0xCC	204
A6-	no VK Code defined	0xCD	205
A6+	no VK Code defined	0xCE	206
Z+	no VK Code defined	0xCF	207
Z-	no VK Code defined	0xD0	208

Depending on the size of the display, two robotics membrane keypads are available:

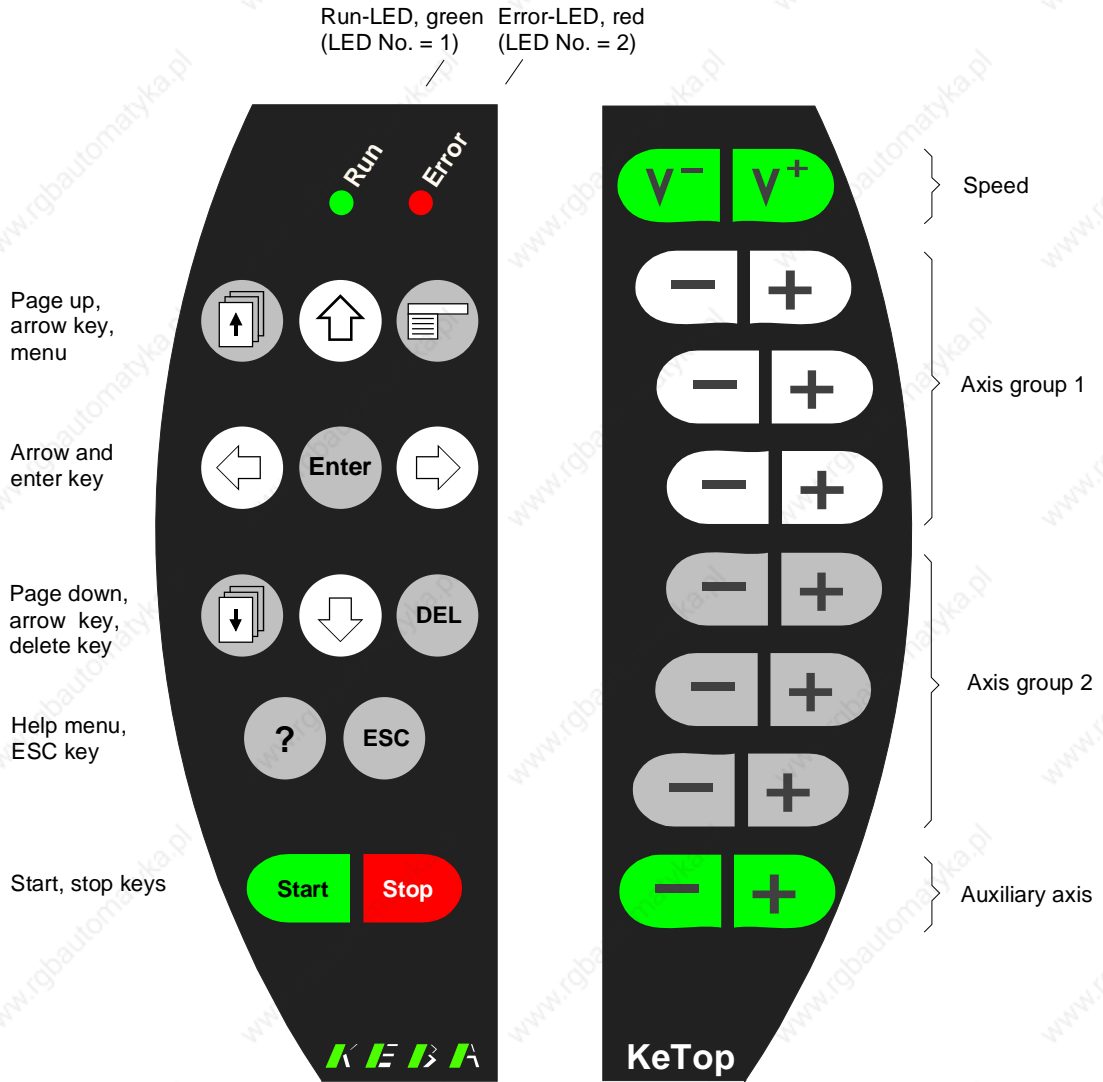
a) 7.7" display (640x480 pixels)

Run-LED, green (LED No. = 1) Error-LED, red (LED No. = 2)



Robotics membrane keypad for 7.7" display

b) 8.4" display (800x600 pixels)



Robotics membrane keypad for 8.4" display

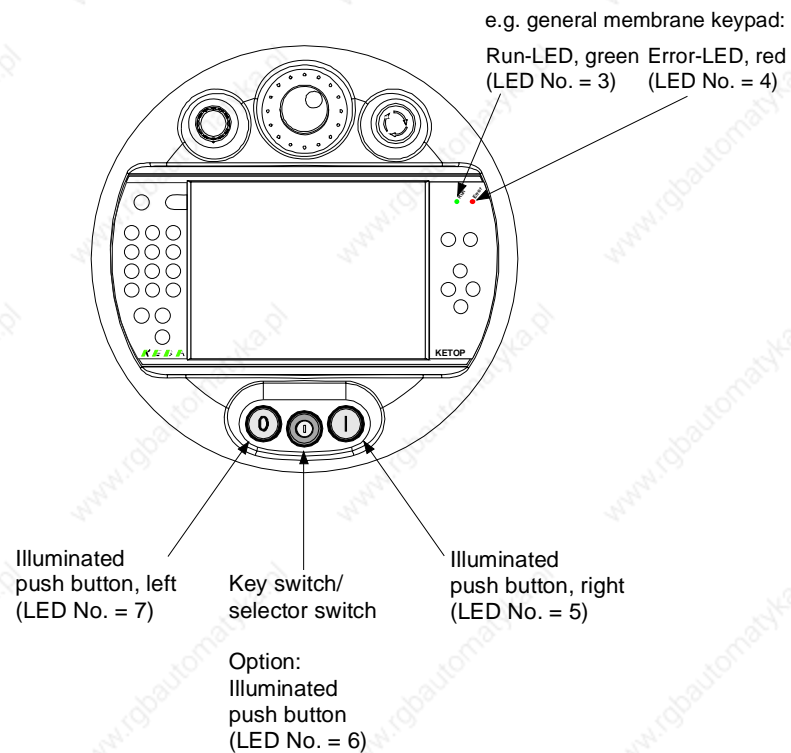
Numbering of LEDs

This chapter describes the numbering of the keyboard LEDs and push-button LEDs.

The LED number is needed for activating the LED.

A LED can be activated in two ways:

- Via KeTop API design
(see chapter „Fehler! Verweisquelle konnte nicht gefunden werden.“
-> „KtpSetKeyboardLed“ on page 57)
- Via KVC – KEBA Virtual Channel
(see chapter „Example: Interface on Server“ on page 84)



Numbering of LEDs on KeTop

6 Display

The KeTop is available with two displays which differ as to their resolution and screen diagonals:

	Display 7.7"	Display 8.4"
Type:	graphics-capable color STN LCD	graphics-capable b/w STN LCD
Size:	7.7" (160 x 120 mm)	8.4" (175 x 130 mm)
Resolution:	VGA 640 x 480 pixels	SVGA 800 x 600 pixels
Representation:	256 colors	256 colors
Background lighting:	CCFT cold cathode tube	CCFT cold cathode tube
Touch screen:	analog-resistive	analog-resistive

The lifetime of the background lighting (40,000 h) can be increased through the activation of a screen saver via the control panel.

Procedure:

*Start -> Settings -> Control Panel -> Display -> Folder Backlight
Select the function „Automatically turn off backlight while on external power“*

Here you can enter the turn-off time (30s to 30min).

Touch Screen

If the touch screen is not operated with the finger, we recommend using a touch stylus, for example touch styluses of PDAs by PALM or SONY.

The touch screen is already calibrated when the KeTop is delivered. No further calibration is required.

If a re-calibration is required for any reason (humidity of air, temperature, etc.), perform the calibration procedure under Windows CE as follows:

Start -> Settings -> Control Panel -> Stylus -> Calibration -> Recalibrate

For the calibration you can also use the ConifgTool. See chapter „Display and Touch Screen Settings“ on page 98.

7 Software

Windows CE

The KeTop T100 is delivered with pre-installed software that is stored in the flash of the device. This software is also called "Image" since it combines all software contents in one file. The image contains the Windows CE operating system core and possibly additional applications. The image is created by means of the MS Platformbuilder for Windows CE.

This description is valid for images from version 2.10.

(Get image version on KeTop as follows: Start -> Programs -> KeTop -> Version)

Software of Standard KeTop T100

Operating system Windows CE 4.2:

- System control
- Command Prompt
- Windows Explorer
- Pocket Internet Explorer

Flash File System:

- IPSM (Intel Persistent Storage Manager)

Pre-installed application programs:

- Pocket Notepad
- RDP-Client (Remote Desktop Connection)
- Pocket Registry Editor (\windows\regedit.exe)
- Active Sync

Pre-installed drivers:

- PC card wireless LAN card type Orinoco PC Card Silver 11Mbit/s IEEE 802.11b
- PC card for Windows CE standard modem

KeTop-specific software (see: Start -> Programs -> KeTop):

- KeTop Configuration Tool
- KebaVirtualChannel KVC
- Registry Backup

- SetTime, temporary time and date entry
- System check, testing tool for KeTop spezific functions
- TouchClean, Touch Cleaning Application
- Version, overview of versions
- Toggle sip (software keyboard)
- KeTop tools (extension in system control for configuration of contrast, background lighting, screensaver, touch screen and joystick calibration, onboard Ethernet on/off, clearing of registry, image update, ..)
- System check, test tool for KeTop spezific functions

Test tools

- TestSerial, test program for serial interfaces (\windows\testserial.exe)
- Reset (\windows\reset.exe)

Generation of Program for Windows CE

The user can easily generate programs for Windows CE. Programming is similar to an application for a standard MS Windows NT PC. Under Windows CE, only the number of available WIN32-APIs is limited.

Prerequisites for Application Programming

- **Microsoft eMbedded Visual C++ 4.0**
- SDK (Software Development Kit) by manufacturer of Windows CE device (KEBA).
The SDK provides to the programming environment the processor-dependent header and library files of the OEM adaptation.
- If special hardware-dependent functions of the KeTop should be used, the SDK of the KeTop must be installed subsequently since the delivered SDK for HPC (Handheld PC) and HPC Pro (Handheld PC Professional) does not take into account certain customer-specific adaptations.

KeTop API Design

The entire SDK is implemented in a single dynamic link library (DLL). All functions described in this document are exported from this DLL.

For Visual Basic, the module file KeTopAPI.bas must be imported into the Visual Basic project. This file and the corresponding SDK are contained on the CD „KeTop SK001“.

The following platform names are assigned to the KeTops:

KeTop T100:	KETOPT100
KeTop T50:	KETOPT50
KeTop T50VGA:	KETOPT50VGA
KeTop T41:	KETOPT41

- ▶ System files that are copied to \windows directory by Visual Studio remain stored permanently. For this reason, these files must be copied manually to the directory \IPSM\windows (must be created by the user if necessary).

In case of a restart of Windows CE, these files are automatically copied to the \windows directory. Therefore these files will be available for the operating system and after a restart.

Common data types

This section contains a detailed description of the common data types for the communication with the handheld terminal. For further information, see `TpuHwDataTypes.h`.

INT8	Signed 8 bit integer variable.
UINT8	Unsigned 8 bit integer variable.
BacklightStat	Enum, displays the backlight status
JoystickPos	Struct, for joystick data.
Status Struct	Describes the startup state of the device.
EventMsg	Enum, describes the event message received.
EventMsgDomains	Enum, describes the events a handler has been subscribed to.
eventCallback	Function pointer to callback function.

Handling of Errors

Rules

- All functions expecting any input parameter check if the parameter is located inside the range and if its data type is correct. If a parameter is located outside the range the function will return `INVALID_ARG_RANGE`.
- All functions expecting a pointer for output data as parameter check if the pointer is valid, i.e. the pointer must not be `NULL`. If the pointer is invalid, the function will return `INVALID_ARG_INVALID_PTR`.
- All functions having any string as parameter check if the pointer to the string is valid. If the pointer is invalid, the function will return `INVALID_ARG_INVALID_STR_PTR`.

Defines

SUCCESS	0
OK	0
FAIL	1
INVALID_ARG_RANGE	2
INVALID_ARG_PTR	3
INVALID_ARG_STR_PTR	4
INVALID_ARG_UNKNWN_COOKIE	5
INVALID_ARG_UNKNWN_DOMAIN	6
INVALID_NOT_CALIBRATED	7
INVALID_POTI_NOT_CALIBRATED	8
UNSAVE_DLL_TERMINATION	-100
API_NOT_INIT	-101

Functions

Functions for starting and closing

This chapter describes functions that are needed for starting and closing the KetopApi.dll.

KtpAPIInit

<i>Declaration</i>	UINT8 KtpAPIInit(void);
<i>Description</i>	This method initializes the KetopAPI.
<i>Arguments</i>	-

KtpAPIDeinit

<i>Declaration</i>	void KtpAPIDeinit(void);
<i>Description</i>	This method cancels all initializations of the KtpAPIInit.
<i>Arguments</i>	-

Functions for Configuration

This section describes the functions that are available for the configuration of the device. All functions return one of the above defines as error code.

KtpSetBrightness

<i>Declaration</i>	UINT8 KtpSetBrightness(/*[in]*/UINT8 u8_Brightness);
<i>Description</i>	This method sets the brightness of the LC display on the device.
<i>Arguments</i>	0-7, 0 = min, 7 = max

KtpSetBrightnessPercent

<i>Declaration</i>	UINT8 KtpSetBrightnessPercent(/*[in]*/UINT8u8_Brightness);
<i>Description</i>	This method sets the brightness of the LC display on the device in percent.
<i>Arguments</i>	0-100%

KtpSetContrast

<i>Declaration</i>	UINT8 KtpSetContrast(/*[in]*/UINT8 u8_Contrast);
<i>Description</i>	This method sets the contrast of the LC display on the device.
<i>Arguments</i>	0-31, 0 = min, 31 = max (Ketop T50: 0-63, 0 = min, 63 = max)

KtpSetContrastPercent

<i>Declaration</i>	UINT8 KtpSetContrastPercent(/*[in]*/UINT8u8_Contrast)
<i>Description</i>	This method sets the contrast of the LC display on the device in percent.
<i>Arguments</i>	0-100%

KtpSwitchBacklight

<i>Declaration</i>	UINT8 KtpSwitchBacklight(/*[in]*/BacklightStat backLight);
<i>Description</i>	Turns on/off the backlight of the LC display on the device.
<i>Arguments</i>	BACKLIGHT_ON, BACKLIGHT_OFF

KtpSetScreenSaverTimeOutMin

<i>Declaration</i>	UINT8 SetScreenSaverTimeOutMin(/*[in]*/UINT8 u8_ScreenSaverTO);
<i>Description</i>	Sets the timeout value of the screensaver in minutes.
<i>Arguments</i>	0-255, 0 = off, 255 = max

KtpSetScreenSaverTimeOutSec

<i>Declaration</i>	UINT8 SetScreenSaverTimeOut- Min(/*[in]*/UINT16u16_ScreenSaverTO);
<i>Description</i>	Sets the timeout value of the screensaver in seconds.
<i>Arguments</i>	0-65535, 0 = off, 65535 = max

KtpSetBuzzerVolume

<i>Declaration</i>	UINT8 KtpSetBuzzerVolume(/*[in]*/UINT8 u8_Volume);
<i>Description</i>	Sets the volume of the buzzer.
<i>Arguments</i>	0-16, 0 = off, 16 = max

Functions for Reading the Configuration

These functions return the current value of the configuration parameters. None of the functions need any parameter. These functions do not enable checking for errors since the return value of the function is the value of the configuration parameter.

KtpGetBrightness

<i>Declaration</i>	UINT8 KtpGetBrightness(void);
<i>Description</i>	Gets the current brightness value of the LC display.
<i>Arguments</i>	-

KtpGetBrightnessPercent

<i>Declaration</i>	UINT8 KtpGetBrightnessPercent(/*[out]*/UINT8 *u8_Brightness);
<i>Description</i>	Gets the current brightness value of the LC display in percent.
<i>Arguments</i>	UINT8 *u8_Brightness: brightness value 0-100%

KtpGetContrast

<i>Declaration</i>	UINT8 KtpGetContrast(void);
<i>Description</i>	Gets the current contrast value of the LC display.
<i>Arguments</i>	-

KtpGetContrastPercent

<i>Declaration</i>	UINT8 KtpSetContrastPercent(/*[out]*/UINT8 *u8_Contrast);
<i>Description</i>	Gets the current contrast value of the LC display in percent.
<i>Arguments</i>	UINT8 *u8_Contrast: contrast value 0-100%

KtpGetBacklight

<i>Declaration</i>	TKtpBacklightStat KtpGetBacklight(void);
<i>Description</i>	Gets the current status of the background lighting.
<i>Arguments</i>	-

KtpGetScreenSaverTimeoutMin

<i>Declaration</i>	UINT8 GetScreenSaverTimeOutMin(void);
<i>Description</i>	Gets the current timeout value of the screensaver in minutes.
<i>Arguments</i>	-

KtpGetScreenSaverTimeoutSec

<i>Declaration</i>	UINT8 GetScreenSaverTimeOutSec(void);
<i>Description</i>	Gets the current timeout value of the screensaver in seconds.
<i>Arguments</i>	-

KtpGetBuzzerVolume

<i>Declaration</i>	UINT8 KtpGetBuzzerVolume(void);
<i>Description</i>	Gets the current volume value of the buzzer.
<i>Arguments</i>	-

Peripheral Functions

KtpJoystickIsInstalled

<i>Declaration</i>	UINT8 KtpJoystickIsInstalled(void);
<i>Description</i>	Returns the number of joystick axes. If no joystick is installed on the device, 0 will be returned.
<i>Arguments</i>	-

KtpWheelsInstalled

<i>Declaration</i>	UINT8 KtpWheelIsInstalled(void);
<i>Description</i>	Returns 1 if an handwheel is installed on the device, otherwise 0.
<i>Arguments</i>	-

KtpPotIsInstalled

<i>Declaration</i>	UINT8 KtpPotIsInstalled(void);
<i>Description</i>	Returns 1 if an override potentiometer is installed on the device, otherwise 0.
<i>Arguments</i>	-

KtpGetJoystickPos

<i>Declaration</i>	UINT8 KtpGetJoystickPos (/*[out]*/TKtpJoystickPos *p_jPos);
<i>Description</i>	Gets the current joystick position.
<i>Arguments</i>	TKtpJoystickPos structure. Each component in the structure may only range between -15 and 15.
<i>Remarks</i>	Calling this function is only allowed if a joystick is installed on the device. If no joystick is installed the values of the components are undefined.

KtpGetJoystickPosEx

<i>Declaration</i>	UINT8 KtpGetJoystickPosEx(/*[out]*/int *posX, int *posY, int *posZ);
<i>Description</i>	Gets the current joystick position.
<i>Arguments</i>	Each component in the structure may only range between -15 and 15.
<i>Remarks</i>	Calling this function is only allowed if a joystick is installed on the device. If no joystick is installed the values of the components are undefined.

KtpGetJoystickPosRaw

<i>Declaration</i>	UINT8 KtpGetJoystickPosRaw (/*[out]*/TktJoystickPosRaw *p_jPos);
<i>Description</i>	Gets the actual raw data of the joystick.
<i>Arguments</i>	-
<i>Remarks</i>	Calling this function is only allowed if a joystick is installed on the device. If no joystick is installed the values of the components are undefined.

KtpGetJoystickPosRawEx

Declaration	<code>UINT8 KtpGetJoystickPosEx(/*out*/UINT16 *posX, UINT16 *posY, UINT16 *posZ);</code>
Description	Gets the actual raw data of the joystick.
Arguments	-
Remarks	Calling this function is only allowed if a joystick is installed on the device. If no joystick is installed the values of the components are undefined.

KtpSetJoystickCalibData

Declaration	<code>UINT8 KtpSetJoystickCalibData(/*in*/TKtpJoystickChannel ch, UINT16 rawMin, UINT16 rawCenter, UINT16 rawMax, UINT16 calibRange);</code>
Description	Calibrates the axis of the joystick.
Arguments	TKtpJoystickChannel ch: channel to be calibrated. UINT16 rawMin: value for smallest raw value UINT16 rawCenter: average value for raw data UINT16 rawMax: maximum value of raw data UINT16 calibRange: maximum range of joystick
Remarks	This function may only be called if a joystick is installed on the device. If no joystick is installed, the value of the components are undefined.

KtpGetOverridePoti

Declaration	<code>UINT8 KtpGetOverridePoti(/*out*/UINT8 *p_pos);</code>
Description	Gets the current value of the override potentiometers.
Arguments	Pointer to the variable containing the current value of the override potentiometer.
Remarks	This function may only be called if an override potentiometer is installed on the device. If no override potentiometer is installed, the value of the components are undefined.

KtpGetOverridePotiRaw

Declaration	<code>UINT8 KtpGetOverridePotiRaw(/*out*/UINT16 *p_pos);</code>
Description	Gets the uncalibrated value from the override potentiometer.
Arguments	Pointer to the variable containing the current value of the override potentiometer.
Remarks	This function may only be called if an override potentiometer is installed on the device. If no override potentiometer is installed, the value of the components are undefined.

KtpGetEnablingDevice

Declaration	<code>UINT8 KtpGetEnablingDevice(/*in*/TKtpEnablingDeviceCircuit circuit);</code>
Description	Circuit is the value of the enabling switch to be read out when it is pressed completely (panic position).
Arguments	Returns the current value of the enabling switch transferred in the circuit.
Remarks	This function may only be called if an enabling switch is installed on the device. If no enabling switch is installed, the value of the components are undefined.

KtpSetPotiCalibData

<i>Declaration</i>	UINT8 KtpSetPotiCalibData(<i>/*in*/</i> UINT16 rawMin, UINT16 rawMax, UINT16 calibRange);
<i>Description</i>	Calibration of override potentiometer. This function may only be called if a override potentiometer is installed on the device. If no override potentiometer is installed, the value of the components are undefined.
<i>Arguments</i>	UINT16 rawMin: value for smallest raw value UINT16 rawMax: maximum raw value UINT16 calibRange: maximum range of overridepoti

KtpGetWheelValue

<i>Declaration</i>	UINT8 KtpGetWheelValue(<i>/*out*/</i> UINT16 *p_val);
<i>Description</i>	Gets the actual value of the handwheel.
<i>Arguments</i>	Pointer to the variable containing the current value of the handwheel.
<i>Remarks</i>	This function may only be called if a handwheel is installed on the device. If no handwheel is installed, the value of the components are undefined.

KtpSetWheelValue

<i>Declaration</i>	UINT8 KtpSetWheelValue(<i>/*in*/</i> UINT16 val);
<i>Description</i>	Sets the current position of the handwheel to the value transferred in val.
<i>Arguments</i>	val is the value the handwheel should be set to.
<i>Remarks</i>	This function may only be called if a handwheel is installed on the device. If no handwheel is installed, the value of the components are undefined.

LED Functions

Because the SxTPU2 is not equipped with LEDs, following functions are not relevant.

KtpSetKeyboardLed

<i>Declaration</i>	UINT8 KtpSetKeyboardLed(<i>/*in*/</i> UINT8 ledNr, TLedState ledState);
<i>Description</i>	Switches the set LED to the desired state.
<i>Arguments</i>	UINT8 ledNr: number of LED to be set. UINT ledState: state, the LED is set to.

KtpGetKeyboardLed

<i>Declaration</i>	TLedState KtpGetKeyboardLed(<i>/*in*/</i> UINT8 ledNr);
<i>Description</i>	Returns the state of the desired LED.
<i>Arguments</i>	UINT8 ledNr: number of LED.

Other Functions

These functions perform various actions on the device.

KtpGetStatus

<i>Declaration</i>	UINT8 KtpGetStatus(/*[out]*/Status *p_tpuStatus);
<i>Description</i>	Gets the start-up state of the handheld terminal.
<i>Arguments</i>	Status *p_tpuStatus, pointer to memory area that will receive a copy of the status structure.

KtpWriteToFlash

<i>Declaration</i>	UINT8 WriteToFlash(void);
<i>Description</i>	Writes the registry of the device to the flash.
<i>Arguments</i>	-

KtpReset

<i>Declaration</i>	UINT8 KtpReset(void);
<i>Description</i>	Restarts the device new.
<i>Arguments</i>	-

KtpGetVersionString

<i>Declaration</i>	UINT8 KtpGetVersionString(/*out*/TCHAR *wszVersionString, unsigned int bufferSize);
<i>Description</i>	Returns the state of the KeTop during start-up.
<i>Arguments</i>	TCHAR *wszVersionString: pointer to buffer for the versions string unsigned int bufferSize: length of transferred buffer.

KtpGetEEPromData

<i>Declaration</i>	UINT8 KtpGetEEPromData(/*out*/TEEPromData *data);
<i>Description</i>	Read the data from the EEPROM
<i>Arguments</i>	TEEPromData data: data structure for the data contained in the EEPROM

KtpWriteByteToEEProm

<i>Declaration</i>	UINT8 KtpWriteByteToEEProm (/*in*/UINT16 addr, UINT8 data);
<i>Description</i>	Writes a byte to the EEPROM location transferred in addr.
<i>Arguments</i>	UINT16 addr: address of memory location UINT8 data: data for saving

KtpReadByteFromEEProm

<i>Declaration</i>	UINT8 KtpReadByteFromEEProm (/*in*/UINT16 addr, /*out*/ UINT8 *pData);
<i>Description</i>	Reads a byte from the EEPROM location transferred in addr.
<i>Arguments</i>	UINT16 addr: address of memory location UINT8 *pData: data from the EEPROM

KtpLaunchTouchScreenCalibApp

<i>Declaration</i>	UINT8 KtpLaunchTouchScreenCalibApp (void);
<i>Description</i>	Starts the touch-screen calibration tool.
<i>Arguments</i>	-

KtpGetTemperature

<i>Declaration</i>	UINT8 KtpGetTemperature (void);
<i>Description</i>	Returns the current temperature of the KeTop.
<i>Arguments</i>	-

KtpPlaySound

<i>Declaration</i>	UINT8 KtpPlaySound (/*in*/ UINT16 soundNr);
<i>Description</i>	Plays the system sound in soundNr.
<i>Arguments</i>	UINT16 soundNr: number of system sound.

KtpDoBeep

<i>Declaration</i>	UINT8 KtpDoBeep (/*in*/ UINT8 beepTime10ms);
<i>Description</i>	Is active for beepTime10ms * 10 ms
<i>Arguments</i>	UINT8 beepTime10ms: duration of beeps in n * 10ms

KtpGetVariantData

<i>Declaration</i>	UINT8 KtpGetVariantData (/*out*/ TKtpVariantData *data);
<i>Description</i>	Reads the device configuration from the EEPROM.
<i>Arguments</i>	TKtpVariantData data: data structure for VariantData.

Functions for Subscribing Events

These functions are used to subscribe/unsubscribe callback functions for different events. Joystick, override potentiometer, handwheel and keypad can be used as events.

KtpInstallWheelEventCallback

<i>Declaration</i>	UINT8 KtpInstallWheelEventCallback (*in*/TktpWheelEventCallback pWheelProc, int *cookie);
<i>Description</i>	Subscribes a callback function for the WheelEvent and returns an index (cookie) for the callback function.
<i>Arguments</i>	TKtpWheelEventCallback pWheelProc: callback function to be called when the event occurs. int cookie: The index for the callback function is required for removing the callback function.

KtpInstallWheelEventMessage

<i>Declaration</i>	UINT8 KtpInstallWheelEventMessage (HWND hWnd, int *cookie);
<i>Description</i>	Subscribes a WindowHandler for the WheelEvent and returns an index (cookie).
<i>Arguments</i>	HWND hWnd: WindowHandler, where the message is sent to. int cookie: Index, is required for removing the WindowHandler.

KtpRemoveWheelEventCallback

<i>Declaration</i>	UINT8 KtpRemoveWheelEventCallback (int cookie);
<i>Description</i>	Removes the WheelEventCallback function.
<i>Arguments</i>	int cookie: index for the callback function to be removed.

KtpRemoveWheelEventMessage

<i>Declaration</i>	UINT8 KtpRemoveWheelEventMessage (int cookie)
<i>Description</i>	Removes the WindowHandler.
<i>Arguments</i>	int cookie: index for the WindowHandler to be removed.

KtpInstallOvrEventCallback

<i>Declaration</i>	UINT8 KtpInstallOvrEventCallback (*in*/ TKtpOvrEventCall- back pOvrProc, /*out*/ int *cookie);
<i>Description</i>	Subscribes a callback function for the OverrideEvent and returns an index (cookie) for the callback function.
<i>Arguments</i>	TKtpOvrEventCallback pOvrProc: callback function to be called when the event occurs. int cookie: The index for the callback function is required for removing the callback function.

KtpInstallOvrEventMessage

<i>Declaration</i>	UINT8 KtpInstallOvrEventMessage (HWND hWnd, int *cookie);
<i>Description</i>	Subscribes a WindowHandler for the OverrideEvent and returns an index (cookie) for the WindowHandler.
<i>Arguments</i>	HWND hWnd: WindowHandler, where the message is sent to. int cookie: Index, is required for removing the WindowHandler.

KtpRemoveOvrEventCallback

<i>Declaration</i>	UINT8 KtpRemoveOvrEventCallback (int cookie);
<i>Description</i>	Removes the OverrideEventCallback function.
<i>Arguments</i>	int cookie: index for the callback function to be removed.

KtpRemoveOvrEventMessage

<i>Declaration</i>	UINT8 KtpRemoveOvrEventMessage (int cookie);
<i>Description</i>	Removes the WindowHandler.
<i>Arguments</i>	int cookie: index for the WindowHandler to be removed.

KtpInstallKbdEventCallback

<i>Declaration</i>	UINT8 KtpInstallKbdEventCallback (/*in*/ TktpKbdEventCallback pKbdProc, /*out*/ int *cookie);
<i>Description</i>	Subscribes a callback function for the KeyboardEvent and returns an index (cookie) for the callback function.
<i>Arguments</i>	TktpKbdEventCallback pKbdProc: callback function to be called when the event occurs. int cookie: The index for the callback function is required for removing the callback function.

KtpInstallKbdEventMessage

<i>Declaration</i>	UINT8 KtpInstallKbdEventMessage (HWND hWnd, int *cookie);
<i>Description</i>	Subscribes a WindowHandler for the KeyboardEvent and returns an index (cookie) for the WindowHandler.
<i>Arguments</i>	HWND hWnd: WindowHandler, where the message is sent to. int cookie: The index for removing the WindowHandler.

KtpRemoveKbdEventCallback

<i>Declaration</i>	UINT8 KtpRemoveKbdEventCallback (int cookie);
<i>Description</i>	Removes the WheelEventCallback function.
<i>Arguments</i>	int cookie: index for the callback function to be removed

KtpRemoveKbdEventMessage

<i>Declaration</i>	UINT8 KtpRemoveKbdEventMessage (int cookie);
<i>Description</i>	Removes the WindowHandler.
<i>Arguments</i>	int cookie: index for the WindowHandler to be removed

KtpInstallJoyEventCallback

<i>Declaration</i>	UINT8 KtpInstallJoyEventCallback (<i>/*in*/</i> TktpJoyEventCallback pJoyProc, <i>/*out*/</i> int *cookie);
<i>Description</i>	Subscribes a callback function for the JoystickEvent and returns an index (cookie) for the callback function, if a joystick is installed.
<i>Arguments</i>	TktpJoyEventCallback pJoyProc: callback function to be called when the event occurs. int cookie The index for the callback function is required for removing the callback function.

KtpInstallJoyEventMessage

<i>Declaration</i>	UINT8 KtpInstallJoyEventMessage (HWND hWnd, int *cookie);
<i>Description</i>	Subscribes a WindowHandler for the JoystickEvent and returns an index (cookie) for the WindowHandler.
<i>Arguments</i>	HWND hWnd: WindowHandler, where the message is send to. int cookie: index for the WindowHandler to be removed

KtpRemoveJoyEventCallback

<i>Declaration</i>	UINT8 KtpRemoveJoyEventCallback (int cookie);
<i>Description</i>	Removes the JoyEventCallback function, if a joystick is installed.
<i>Arguments</i>	int cookie: index for the callback function.

KtpRemoveJoyEventMessage

<i>Declaration</i>	UINT8 KtpRemoveJoyEventMessage (int cookie);
<i>Description</i>	Removes the WindowHandler.
<i>Arguments</i>	int cookie: index for the WindowHandler to be removed

KtpLaunchJoystickCalibApp

<i>Declaration</i>	UINT8 KtpLaunchJoystickCalibApp(HANDLE *pProcHandle);
<i>Description</i>	Starts the joystick calibration tool.
<i>Arguments</i>	HANDLE *pProcHandle: Handle for the started program.

KtpGetDispalyRotation

<i>Declaration</i>	TKtpDispalyRot KtpGetDisplayRotation (void)
<i>Description</i>	Returns the state for the DisplayRotation (for left or right handers).
<i>Arguments</i>	-

KtpSetDisplayRotation

<i>Declaration</i>	UINT8 KtpSetDisplayRotation(TKtpDisplayRot dispRot);
<i>Description</i>	Changes the design of the display for left or right handers. Maybe for some devices no changes.
<i>Arguments</i>	TKtpDisplayRot dispRot: eKtpDisplayLeft for left handers, eKtpDisplayRight for right handers

KtpEraseRegistry

<i>Declaration</i>	UINT8 KtpEraseRegistry(void);
<i>Description</i>	Deletes the registry of the device during next startup.
<i>Arguments</i>	-

KtpErasePSM

<i>Declaration</i>	UINT8 KtpErasePSM(void);
<i>Description</i>	Deletes the flash file system of the device during next startup.
<i>Arguments</i>	-

KtpGetPowerFailState

<i>Declaration</i>	UINT8 KtpGetPowerFailState(void);
<i>Description</i>	Returns the actually state of the powerFail input.
<i>Arguments</i>	-

KtpForcePressedHardbuttons

<i>Declaration</i>	UINT8 KtpForcePressedHardbuttons (void);
<i>Description</i>	A WM_KEYDOWN message will be send for each pressed keyswitch and the KeyboardCallback function will be activated. This function will be activated also for the initializing of the KetopApi and for the installation of the KeyboardCallback function.
<i>Arguments</i>	-

KtpIsJoystickCalibrated

<i>Declaration</i>	UINT8 KtpIsJoystickCalibrated (void);
<i>Description</i>	Returns OK (0) if all joystick axes are calibrated. If one or more axes are not calibrated, it returns INVALID_NOT_CALIBRATED (7). If no joystick exists, it returns INVALID_NOT_SUPPORTED (6).
<i>Arguments</i>	-

KtpIsPotiCalibrated

<i>Declaration</i>	UINT8 KtpIsPotiCalibrated (void);
<i>Description</i>	Returns OK (0) if the override potentiometer is calibrated. Returns INVALID_NOT_CALIBRATED (7), if it is not calibrated. Returns INVALID_NOT_SUPPORTED (6) if no override potentiometer is installed.
<i>Arguments</i>	-

KtpRestoreMemorySettings

<i>Declaration</i>	UINT8 KtpRestoreMemorySettings(void);
<i>Description</i>	Saves the registry-value of the path "System\MemorySettings\StorePages" as storage memory. If the registry-value is in a range where memory for storage respectively for program memory is reserved, these settings wont be changed and INVALID_ARG_RANGE will be returned. If it is not allowed to save the memory settings ("System\MemorySettings\MemorySaveEnable"=0), INVALID_NOT_SUPPORTED will be returned.
<i>Arguments</i>	-

KtpStoreCurrentMemorySettings

<i>Declaration</i>	UINT8 KtpStoreCurrentMemorySettings(void);
<i>Description</i>	Saves the memory settings of the registry ("System\MemorySettings\StorePages"), if it is allowed. If it is not allowed to save, INVALID_NOT_SUPPORTED will be returned.
<i>Arguments</i>	-

KtpShowInputPanel

<i>Declaration</i>	UINT8 KtpShwoInputPanel(UINT8 show);
<i>Description</i>	Opens (show=1) or closes (show=0) the input panel on the display and returuns OK or FAIL.
<i>Arguments</i>	-

Update API Design

All required methods for an image update are implemented in one single dynamic link library (update.dll). All functions described in this document are exported from this DLL.

For running the update.dll the appropriate KetopAPI.dll will be necessary. Both DLL files must exist in the image and correspond with each other.

Handling of Errors

Rules

- All functions expecting any input parameter check if the parameter is located inside the range and if its data type is correct. If a parameter is located outside the range the function will return ERROR_INVALID_RANGE.

Defines

OK	0
SUCCESS	0
FAIL	1
ERROR_INVALIDE_RANGE	100
ERROR_IMGUPD_INIT	101
ERROR_IMGUPD_FILEOPEN	102
ERROR_IMGUPD_WRONGIMG	103
ERROR_IMGUPD_PROGRAMERROR	104
ERROR_IMGUPD_TOMUCHUPD	105
ERROR_PROGRAM_FLASH_BURN	106
ERROR_PROGRAM_FLASH_ERASE	107
ERROR_IMGUPD_NOTEQUAL	108
ERROR_IMGUPD_FILEWRITE	109
ERROR_IMGUPD_FILEREAD	110
ERROR_WRONG_FILE_HEADER	111

Initialising

All required initialisation will be carried out by starting respectively by loading the update.dll.

Functions

UpdStartImageUpdate

Declaration	
	BOOL UpdStartImageUpdate (LPCTSTR fileName, BOOL eraseRegPSM, TUpdProgressCallback pCallback)
Description	<p>With this method the file handed over by <code>fileName</code> will be stored to the FLASH memory of the device. If <code>eraseRegPSM</code> is set, the registry respectively the PSM will be deleted, after the FLASH has been written successfully.</p> <p>If a method is handed over with <code>pCallback</code>, the method will be executed depending of the program progress. The method returns OK or the corresponding error message.</p>
Arguments	<p>LPCTSTR <code>fileName</code>: Name of the image file</p> <p>BOOL <code>eraseRegPSM</code>: Flag for deleting the registry respectively the PSM</p> <p>TUpdProgressCallback <code>pCallback</code>: Callback function for program progress or 0</p>

UpdPartialImageUpdate

Declaration	
	BOOL UpdPartialImageUpdate (LPCTSTR fileName, unsigned long from, unsigned long to, BOOL eraseRegPSM, TUpdProgressCallback pCallback)
Description	<p>With this method the file handed over by <code>fileName</code> will be stored from the address <code>from</code> to the address <code>to</code> to the FLASH memory of the device.</p> <p>If <code>eraseRegPSM</code> is set, the registry respectively the PSM will be deleted, after the FLASH has been written successfully.</p> <p>If a method is handed over with <code>pCallback</code>, the method will be executed depending of the program progress. The method returns OK or the corresponding error message.</p>
Arguments	<p>LPCTSTR <code>fileName</code>: Name of image file</p> <p>unsigned long <code>from</code>: First address for overwriting</p> <p>unsigned long <code>to</code>: Address, out of overwriting</p> <p>BOOL <code>eraseRegPSM</code>: Flag for deleting the registry respectively the PSM</p> <p>TUpdProgressCallback <code>pCallback</code>: Callback function for program progress or 0</p>

UpdCheckFile

Declaration	BOOL UpdCheckFile(LPCTSTR fileName, TupdProgressCallback pCallback)
Visual C: Visual Basic:	UINT8 KtpSetBrightness(/*[in]*/UINT8 u8_Brightness); KtpSetBrightness(ByVal brightness As Byte) As Byte
Description	UpdCheckFile verifies the image file and if the image file corresponds to the device. If a method is handed over with pCallback, the method will be executed depending of the program progress. The method returns OK or the corresponding error message.
Arguments	LPCTSTR fileName: Name of the image file TupdProgressCallback pCallback: Callback function for program progress or 0

UpdCompareFile

Declaration	BOOL UpdCompareFile(LPCTSTR fileName, TupdProgressCallback pCallback)
Description	UpdCompareFile compares the handed over image file with the entries of the FLASH memory. If a method is handed over with pCallback, the method will be executed depending of the program progress. The method returns OK or the corresponding error message.
Arguments	LPCTSTR fileName: Name of the image file TupdProgressCallback pCallback: Callback function for program progress or 0.

UpdSetFlashLock

Declaration	BOOL UpdSetFlashLock(long from, long to)
Description	This method sets the lock bits of the FLASH chips from the address from to the address to. So, this address area can't be overwritten. After an image update, the lock bits are reset. One lock bit locks one whole FLASH block. So the saved area can differ from the handed over addresses.
Arguments	long from: First address for locking bits. long to: Address as far as the lock bits are set.

UpdResetFlashLock

Declaration	BOOL UpdResetFlashLock(long from, long to)
Description	This method resets the lock bits of the FLASH chips from the address from to the address to, One lock bit locks one whole FLASH block. So the unsaved area can differ from the handed over addresses.
Arguments	long from: First address for reset locking bits. long to: Address as far as the lock bits are reset.

UpdGetImage

<i>Declaration</i>	<code>BOOL UpdGetImage(LPCTSTR fileName, unsigned long from, unsigned long to, TUpdProgressCallback pCallback);</code>
<i>Description</i>	UpdGetImage creates a copy of the device-image from the address from to the address to and saves it in the file fileName. If a method is handed over with pCallback, the method will be executed depending of the program progress. The method returns OK or the corresponding error message.
<i>Arguments</i>	LPCTSTR fileName: Name of the image file unsigned long from: Starting address stored in the file unsigned long to: Address not stored TUpdProgressCallback pCallback: Callback funktion for program progress or 0.

UpdEraseRegistry

<i>Declaration</i>	<code>BOOL UpdEraseRegistry();</code>
<i>Description</i>	The method UpdEraseRegistry clears the registry of the device after the next update.

UpdErasePSM

<i>Declaration</i>	<code>BOOL UpdErasePSM();</code>
<i>Description</i>	the method UpdErasePSM clears the PSM of the device after the next update.

UpdResetDevice

<i>Declaration</i>	<code>BOOL UpdResetDevice();</code>
<i>Description</i>	The method UpdResetDevice resets the device.

UpdGetFileVersion

<i>Declaration</i>	<code>BOOL UpdGetFileVersion(LPCTSTR fileName, LPTSTR version);</code>
<i>Description</i>	UpdGetFileVersion returns the version information stored in the image file in the string version. The number of characters returned can be defined in VERSION_STRING_LEN. Therefore version has to exceed VERSION_STRING_LEN.
<i>Arguments</i>	LPCTSTR fileName: Name of the image file LPTSTR version: String for version information

UpdGetImageVersion

<i>Declaration</i>	<code>BOOL UpdGetImageVersion(LPTSTR version);</code>
<i>Description</i>	UpdGetImageVersion returns the in the registry stored version information in the string version. The number of characters returned is defined in VERSION_STRING_LEN. Therefore version has to exceed VERSION_STRING_LEN.
<i>Arguments</i>	LPTSTR version: String for version information

TupdProgressCallback

<i>Declaration</i>	<code>typedef void (* TUpdProgressCallback) (int percent);</code>
<i>Description</i>	TUpdProgressCallback is the prototype of the callback functions, which can be called from the Update API.
<i>Arguments</i>	<code>int percent</code> : Number between 0 and 100 for the progress

Program for Starting the Application and KeTop API

This chapter describes the program for starting the application and the KeTop API. The program is contained as a StartAPI.exe in the Windows directory.

Functional description

The StartAPI program is started by an entry in the Startup directory or by an entry in the registry under [HKEY_LOCAL_MACHINE\init] when the KeTop is started.

When the program is started, the KeTop API will be initialized first and then the programs entered under [HKEY_LOCAL_MACHINE\Autostart] will be started.

The program remains invisible in the memory and handles different input devices, e.g. override potentiometer and handwheel.

Registry entries

[HKEY_LOCAL_MACHINE\Autostart] contains the entries that are read out by the program.

The following entries are possible:

- **Startx**
string identifying the program to be started. x is a number between 1 and 255.
- **Paramsx**
string containing the parameters for the program identified in Startx. x must have the same value as in Startx.
- **Delayx**
DWORD containing the waiting time in milliseconds until the next program is started. x must have the same value as in Startx.
- **DependStartx**
array of 10 bytes containing the numbers of the programs which must have been started so that the program x may start.
If there is no such entry or all bytes are 0 then the program x starts and no other programs must have been started.
- **DependEndx**
array of 10 bytes containing the numbers of the programs which must have been stopped before the program x starts.
If there is no such entry or all bytes are 0 then the program x starts without waiting of any other programs.
- **StartTypex**
DWORD affecting the startup of the application.

StartTypex = 0: The application starts automatically
 StartTypex = 1: The application wont be started
 StartTypex = 2: A window with the message „start programm <Yes> / <No>“ appears. The user can decide, if the application should be started or not.
 x must have the same value as **Startx**.

Example

Example of a possible configuration in the registry:

```
[HKEY_LOCAL_MACHINE \ Autostart]
Start1      = "cmd.exe"
Params1    = "/c copy \ipsm\windows\*. * \windows"
Delay1     = dword:0x000003e8 (1000)
```

```
Start10    = "pvbload.exe"
Params10   = "\ipsm\vb\TestProjectVB1.vb"
Delay10    = dword:0x00002710 (10000)
```

```
Start20    = "SetTime.exe"
Params20   = ""
Delay20    = dword:0
```

```
Start30    = "ReadCorrectTime.exe"
Params30   = ""
Delay30    = dword:0
DependStart30 = hex: a, 0, 0, 0, 0, 0, 0, 0, 0, 0
DependEnd30  = hex:14, 0, 0, 0, 0, 0, 0, 0, 0, 0
```

When the StartAPI is started, the KeTop API will be initialized first and then the program cmd.exe will be called with the parameters "/c copy \ipsm\windows*. * \windows".

```
cmd.exe /c copy \ipsm\windows\*. * \windows
```

This program copies the data from the directory \ipsm\windows to the normal Windows directory.
 After the call of cmd.exe, the system waits for 1000 milliseconds (Delay1) until the next program is started even if the cmd.exe has not been completed before.

After the delay, the program pvbload.exe will be started with the parameters "\ipsm\TestProjectVB1.vb".

```
pvbload.exe \ipsm\TestProjectVB1.vb
```

This program starts the VisualBasicScript TestProjectVB1.vb.

After a waiting time of 10000 milliseconds (Delay10), the program with the next higher entry will be started.

SetTime starts a program for setting the time. This program will be started without parameter and delay.

Afterwards it will be checked if the program 10 (DependStart30 a,..) has been started and the SetTime program (DependEnd30 14,..) has been finished already. If all conditions are met the program ReadCorrectTime will be started.

If no more entries are available, the starting of the applications will be finished.

Programs started by the StartAPI will not be finished.

RDP – Connection via Remote Desktop Protocol

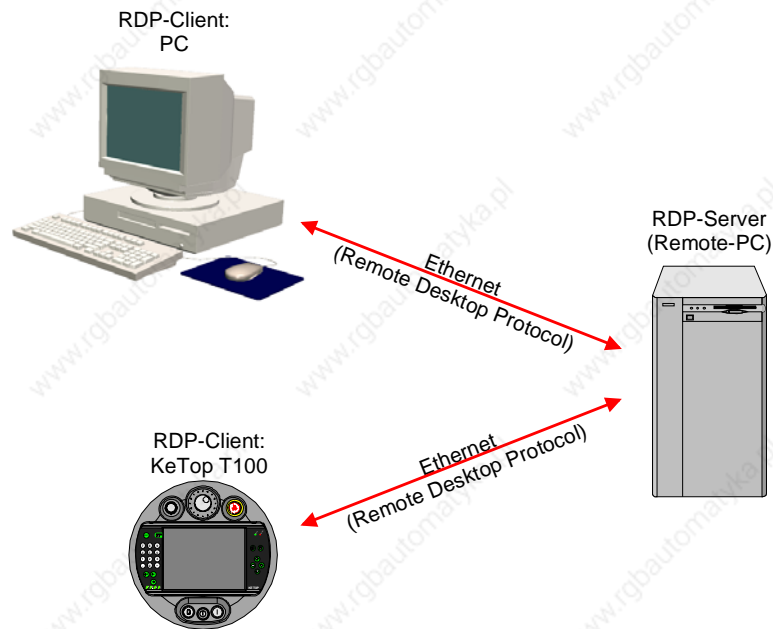
The Remote Desktop Protocol (Abbreviated as "RDP") is a specification by Microsoft for remote control of applications.

Microsoft has replaced the earlier name "Microsoft Terminal Server Client" (Abbreviated as "MSTSC") by the term RDP-Client.

A detailed description of this term can be found on Microsoft's website.

Likewise, it can also be found in the Windows-Help feature using the search criteria "mstsc" and "rdp".

Operating principle (highly simplified):



RDP-Client <-> RDP-Server

Programs and data are stored on the remote computer. This is where the program is executed and the data is accessed. This remote computer (Remote PC) is identified as "RDP-Server."

However, data is entered and displayed on the monitor of the local "RDP-Client."

The exchange of data between the client and the server takes place as per the "Remote Desktop Protocol."

RDP-Server (Remote-PC)

The RDP-Server (Remote-PC) requires one of the following operating systems:

- Windows NT4 Terminal server
- Windows 2000 Server
- Windows Server 2003
- Windows XP Professional (Not Windows XP Home !)
- Windows XP embedded (The Server-functionality is integrable.)

Note

When using Windows XP Professional, the RDP-access on the PC in use, must be enabled using the path below: "Control Panel -> System -> Remote" (The option "(x) Remotedesktop" must be activated.)

RDP-Client (PC)

To start the RDP-Client on the PC, the file <mstsc.exe> must be invoked. In Windows XP, this can be found in the Windows directory or under "Start -> All Programs -> Accessories -> Communication -> Remote Desktop Connection."

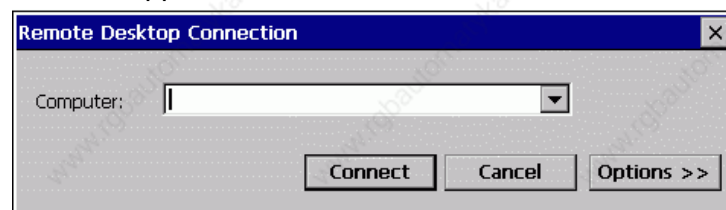
For other operating systems such as Windows 95/98/NT, the appropriate files can be obtained from the Microsoft website.

RDP-Client (KeTop)

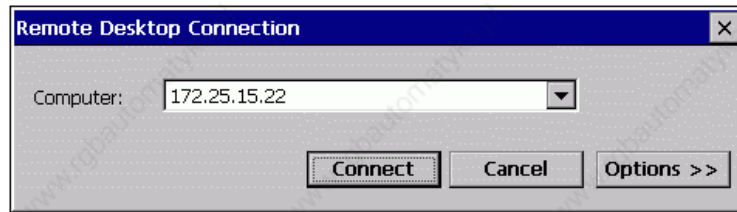
When starting the client on the KeTop, the file `cetasc.exe` must be invoked. This is located in the Windows directory and can be accessed using the path "Start-> Programs -> Remote Desktop Connection."

Starting a RDP-Connection manually

- ▶ Start the RDP-client using the file `cetasc.exe` and the login window shown below, appears:



- ▶ Enter the IP-Address of the destination computer in the field “Computer.”



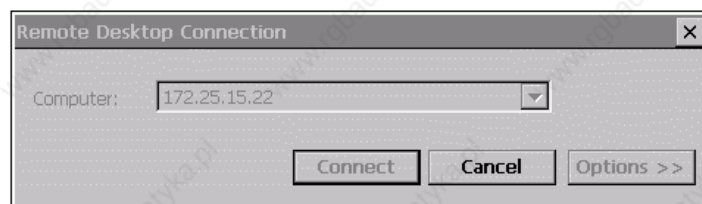
In case the network name is to be used in place of the IP-Address, then resolution of the name must be ensured either via the DNS-server or by an entry in the file “hosts.ini.”

Note

The “Input panel” in this case must be enabled on the KeTop, as access to this feature is blocked during the logging process:

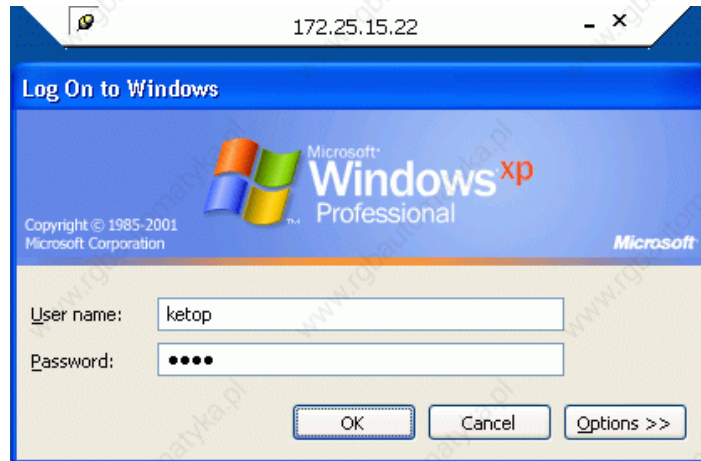


- ▶ Click the button “Connect” to establish connection with the destination computer.



Clicking the “Cancel” button disrupts the connection set up.

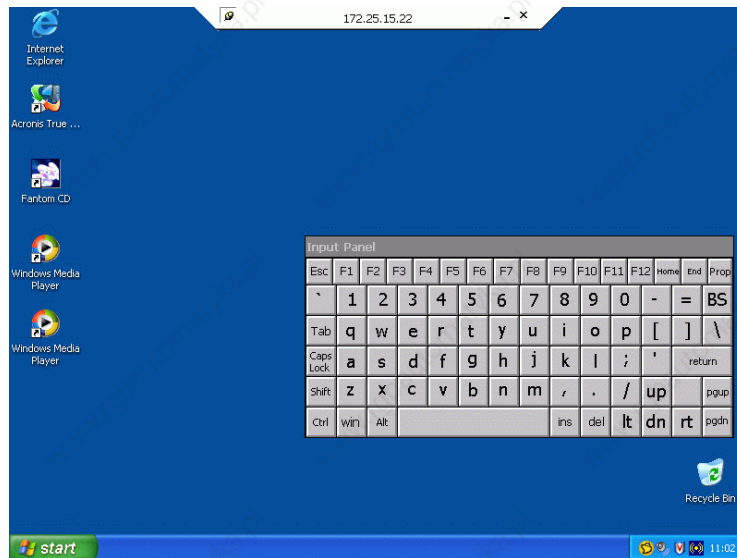
Once the connection has been established, the login window of the destination computer is displayed with a trapezoidal title bar that is treated as an identifier of the RDP-Connection.



- ▶ Enter the username and the password and confirm by clicking OK.



The Windows screen of the destination computer is displayed once login is successful.

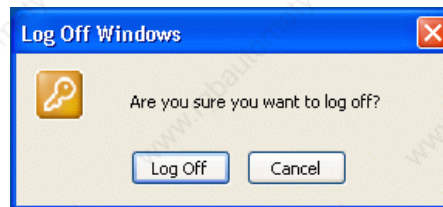


If the trapezoidal title bar shows any errors when being used with certain applications, it can be deactivated when logging in the next time, by simply clicking the symbol at the top left.

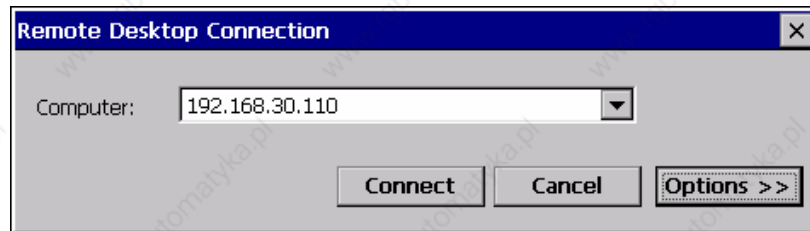


If this symbol is displayed, the title bar automatically disappears after the next login.

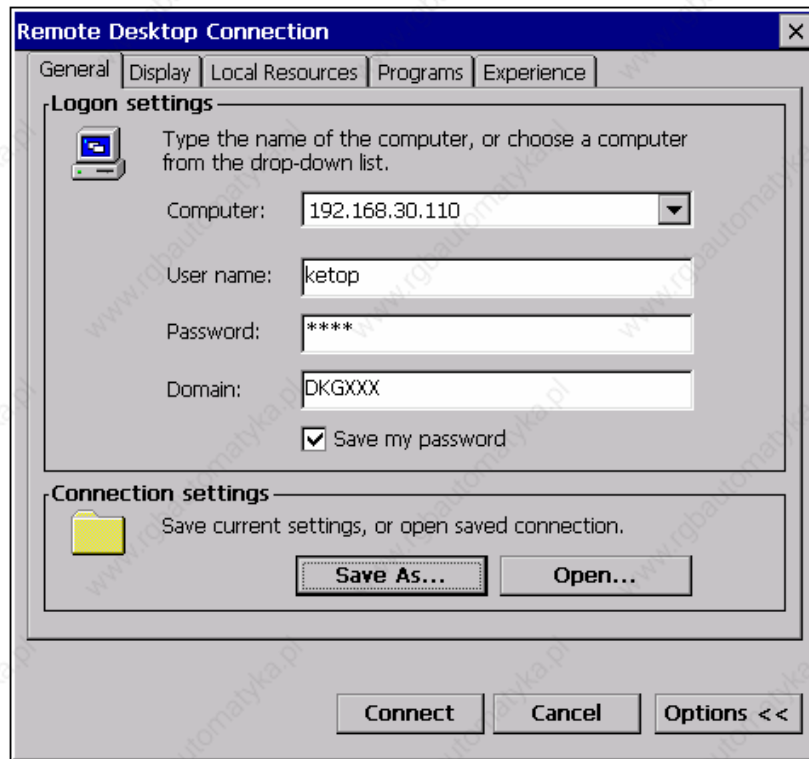
By invoking “Start->Logout,” you can logout of the destination computer.



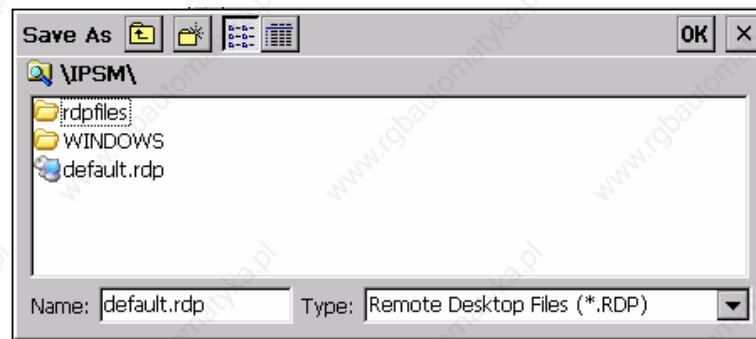
Saving the settings of a RDP-Connection



- ▶ Clicking the button “Options >>” on the login window of the RDP-Client displays an expanded window with the current settings.



- ▶ Click the button “Save As...” and save all these settings in a file of the type “rdp”



Starting the RDP-Connection automatically via an entry in the registry

Connection with the destination compute can be established automatically on starting the KeTop with the help of an entry in the registry under [HKEY_LOCAL_MACHINE\Autostart]. (Also refer the chapter “Program for starting the application and the KeTop API”)

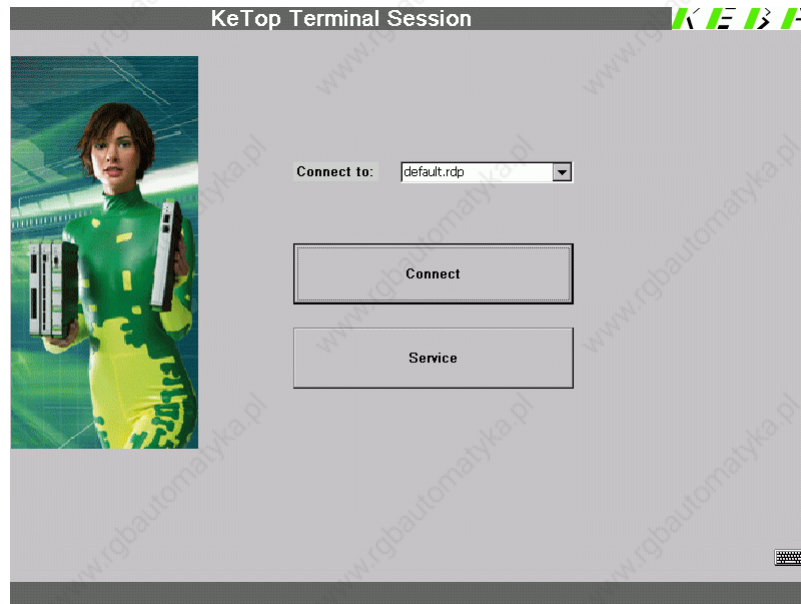
Startx = "\windows\cetsc.exe "
 Paramsx = "\ipsm\default.rdp"

While doing so, the precise RDP-settings must be saved in the file “\ipsm\default.rdp.”

Starting a RDP-Connection via TSCDialog

KEBA has produced TSCDialog.exe, an upgraded version of the RDP-Client with further enhancements.

This program can be used to regulate the operator's access to Windows CE. Moreover, the login screen can be designed as desired.



The following entry must exist in the registry in order to invoke the TSCDialog.exe when the KeTop is turned on:

```
HKEY_LOCAL_MACHINE\Autostart\Startx = "\windows\tscdialog.exe"
```

While doing so, the precise RDP-settings must be saved in the file "\ipsm\rdpfiles\default.rdp."

To customize the login screen as required, individual elements of the login screen may be replaced by customer-specific elements.

The customization is controlled using the file
 \ipsm\rdpfiles\resources\TSCDialog.ini

TakeUserLogo:i:1	
ProgTitle:s:title	(Title)
LogoFileName:s:logo.bmp	(Logo right top, in place of "KEBA")
ImgFileName:s:bitmap.bmp	(Image to the left, in place of the woman in green)

Likewise, the files "logo.bmp" and "bitmap.bmp" must also be entered in the directory "\ipsm\rdpfiles\resources\".

KVC – KEBA Virtual Channel

The protocol "KEBA Virtual Channel" (KVC) is used to transmit control and operating element data between a control and one or more KeTop handheld terminals.

The following data can be transmitted via the KVC:

Data	Direction ST ↔ HT	Value range	Size [bytes]	Transmission
Override potentiometer	↔	0..127	1	Event in case of modification and upon request of the control
Electronic handwheel	↔	-32000 .. 32000	2	Event in case of modification, upon request of the control, and as set command for adjusting
Joystick	↔	2x -63..63	2	Event in case of 0-Pos and ≠ 0-Pos, then request of control
Lighting for button below display LEDs on keypad	↔	on flashing off	2	As command from the control and as request from the client to the control.
Contrast, brightness	↔	0..255	2	As command from the control and as request from the client to the control
Time for screensaver	↔	0..255	1	As command from the control and as request from the client to the control
State of screensaver	↔	0..1	1	Event or as request from the client to the control
Volume	↔	0..255	1	As command from the control and as request from the client to the control
Background lighting	↔	0..1	1	As command from the control and as request from the client to the control
WriteToFlash	→	-	-	Command
PlaySound	→	0..255	1	Command
KeepAlive	↔	0..65535	2	Command and KeepAliveEvent if parameterized by the server

The data transmission between the control and the handheld terminals is based on an Ethernet connection (TCP/IP protocol, Listening Port **0xCEBA**). All devices must be identified by their IP address.

Several handheld terminals may be connected to one control, but **one** handheld terminal can only be connected to **one** control.

The KVC protocol is an event-driven protocol, i.e. each station can send data at any time without request.

To monitor the functioning of the KeTop T100 in the control, the KVC offers the possibility to use KeepAlive data packages that are periodically sent from the client to the server. The intervals at which the data packages are sent can be set in milliseconds.

Events Causing the Client to Send Data

- Modification of value of override potentiometer
- Modification of handwheel value
- Modification of key status of a key
- The joystick or the space mouse reaches the 0 position.
- The joystick or space mouse leaves the 0 position.
- Data inquiry of server

With each event, the client sends to the server a data package containing the information about the type of event as well as the current data of all operating elements. If the client produces several events one after the other, and the server cannot process all of them at once (e.g. quick rotation of handwheel and simultaneous movement of joystick), the server will send one message containing all events.

Events of Server

By sending a package to the client, the server can influence the status of the LEDs on the client, set the current value of the handwheel, or request the current values (position of joystick, handwheel and override potentiometer, as well as state of keys). The server can also read the current states of the values which can be modified by the server.

Data Transmission

Each of the two socket connections between the client and the server constitutes a channel for the data transmission in each direction. The data structures relevant for the corresponding data direction are described in this chapter.

Client → Server

```
typedef enum {
    eKVCJoystickIsZero      0x0001
    eKVCJoystickNotZero    0x0002
    eKVCSpaceMouseIsZero   0x0004
    eKVCSpaceMouseNotZero 0x0008
    eKVCHandWheelChanged   0x0010
    eKVCOverrideChanged    0x0020
    eKVCKeyPressed         0x0040
    eKVCKeyReleased        0x0080
    eKVCLEDValue           0x0100
    eKVCContrast           0x0200
    eKVCBrightness         0x0300
    eKVCVolume             0x0400
    eKVCScreensaverTime    0x0500
    eKVCScreensaverState   0x0600
    eKVCBacklightState     0x0700
    eKVCLED                 0x0800
    eKVCJoystickResp       0x0900
    eKVCSpaceMouseResp     0x0A00
    eKVCHandWheelResp      0x0B00
    eKVCOverrideResp       0x0C00
    eKVCALive               0xFE00
    eKVCClientDisconnect   0xFF00
} TKVCEvent;

typedef struct {
    UINT16 event;
    struct {
        UINT8          overrideVal;
        UINT8          keyVal;
        TKVCJostickData joystickVal;
        SINT16         handWheelVal;
        TKVCSpaceMouseData spaceMouseVal;
    } data;
} TKVCClientData;

typedef enum {
    eKtpKeyboardLedOff   = 1,
    eKtpKeyboardLedOn   = 2,
    eKtpKeyboardLedBlink = 3
} TKtpLedState;

typedef struct {
    char posX;
    char posY;
```

```
        char posZ;
    } TKVCJostickData;

typedef struct {
    UINT16 posX;
    UINT16 posY;
    UINT16 posZ;
} TKVCSpaceMouseData;

typedef struct {
    SINT16 absVal;
    SINT16 dynVal;
}TKVCHandWheelData;
```

Server → Client

```
typedef enum {
    eKVCSetWheelValue,
    eKVCSetLed
    eKVCSetContrast,
    eKVCSetBrightness,
    eKVCSetVolume,
    eKVCSetScreensaver,

    eKVCGetLed,
    eKVCGetContrast,
    eKVCGetBrightness,
    eKVCGetVolume,
    eKVCGetScreensaverTime,
    eKVCGetJostickValue,
    eKVCGetSpaceMouseValue,
    eKVCGetOverrideValue,
    eKVCGetWheelValue,

    eKVCSwitchBacklight,
    eKVCGetBacklightState,

    eKVCPlaySound,
    eKVCWriteFlash
    eKVCDisconnect
} TKVCCommand;

typedef struct {
    TKVCCommand command;
    SINT16 param;
} TKVCServerData;
```

Example: Interface on Server

On the server side, the KVC protocol is represented by two classes: `CKVCServer` and `CKVCCConnection`. An object of the `CKVCServer` class represents the actual server (the "listener") and an object of the `CKVCCConnection` class represents a connection to a client.

"Server class"

```
class CKVCServer {
public:
    virtual int Init();
    virtual int Exit();
    virtual CKVCCConnection* OnClientConnect(SOCKET socket,
                                              sockaddr_in
&sockAdr);
    virtual int OnClientDisconnect(CKVCCConnection *pConnection,
                                   TDisconInfo info);
    POSITION ConnectionListHeadPos();
    CKVCCConnection* ConnectionListGetNext(POSITION pos);
    int ConnectionListGetCount();
};
```

int Init();

Initializes the server and opens the port 0xCEBA for incoming connections.

int Exit();

Finishes all connections and closes the port 0xCEBA.

CKVCCConnection* OnClientConnect(SOCKET socket, sockaddr_in &sockAdr);

This method is always called when a teach pendant establishes a connection to the control. The parameters `socket` and `sockAdr` specify the connection parameters of the teach pendant. This function must return a pointer to an object of the class `CKVCCConnection`. A return value of 0 indicates that the control rejects the logon of the teach pendant.

int OnClientDisconnect(CKVCCConnection *pConnection, TDisconInfo info);

This method will be called if the server cannot reach the client any more. The cause of the logoff is specified in `info`.

POSITION ConnectionListHeadPos();

This method returns the position of the first entry in the `OpenConnection` list.

CKVCCConnection* ConnectionListGetNext(POSITION pos);

This method returns a pointer to the `ConnectionObject` that is stored on the position `pos` in the `OpenConnection` list.

int ConnectionListGetCount();

This method returns the number of connections that are stored in the `OpenConnection` list.

Connections

For details about the LED numbering (ledNr) needed for the following functions, refer to the chapter „Robotics Membrane Keypad“ on page 41.

```
class CKVConnection {
private:
    char *pIpAdr;
public:
    virtual int Init(CKVCServer *pServer, SOCKET socket,
                    SOCKADDR_IN &socketAdr);
    virtual int Exit();
    virtual int OnOverrideChange(SINT16 val);
    virtual int OnWheelChange(SINT16 wheelAbsVal);
    virtual int OnKeypadEvent(TKVCEvent keyEvent, UINT8 keyNum);
    virtual int OnJoystickEvent(TKVCEvent event, TKVCJoystickData *pJData);
    virtual int OnSpaceMouseEvent(TKVCEvent event,
                                    TKVCSpaceMouseData *pSMData);
    virtual int OnDisconnect();

    virtual int GetWheelVal(TKVCHandWheelData &hwData);
    virtual int GetOverrideVal(SINT8 &overrideVal);
    virtual int GetJoystickPos(TKVCJoystickData &jData);
    virtual int GetSpaceMousePos(TKVCSpaceMouseData &smData);
    virtual int GetLedState(UINT8 ledNum,UINT8 &state);
    virtual int GetContrast(UINT8 &contrast);
    virtual int GetBrightness(UINT8 &brightness);
    virtual int GetVolume(UINT8 &volume);
    virtual int GetScreensaverTime(UINT16 &time);
    virtual int GetScreensaverState(UINT8 &state);

    virtual int SetWheelVal(SINT16 val);
    virtual int SetLed(UINT8 ledNum, TKVCLedMode mode);
    virtual int SetContrast(UINT8 contrast);
    virtual int SetBrightness(UINT8 brightness);
    virtual int SetVolume(UINT8 volume);
    virtual int SetScreenSaver(UINT16 screenSaverTime);
    virtual int SwitchBacklight(UINT8 backlightOnOff);

    virtual int WriteToFlash();
    virtual int PlaySound(UINT16 soundNr);
    sockaddr_in GetSocketAdr();
    SOCKET GetSocket();
};
```

The methods **OnOverrideChange**, **OnWheelChange**, **OnKeypadEvent**, **OnJoystickEvent** and **OnSpaceMouseEvent** will be called if an event has occurred at the corresponding operating element on the client.

```
int CKVConnection::OnOverrideChange(SINT16 val);
```

This method will be called if the value of the override potentiometer has changed on the client. The current value is specified in the parameter val.

```
int CKVConnection::OnWheelChange(SINT16 wheelAbsVal);
```

The method **OnWheelChange** will be called if the value of the handwheel has changed. The current value is transferred as an absolute value in the parameter wheelAbsVal.

```
virtual int OnKeypadEvent(TKVCEvent keyEvent, int keyNum);
```

The method `OnKeypadEvent` will be called if a key has been pressed/released. The key number is specified in `keyNum`, the state of the key (make, break) in `keyEvent`.

```
int CKVCCConnection::OnJoystickEvent(TKVCEvent event,  
                                     TKVCJoystickData *pJData);
```

The method `OnJoystickEvent` is called when the joystick is moved from the 0 position and reaches the 0 position. The current position is transferred in the parameter `event`, and the current values in the parameter `pJData`.

```
int CKVCCConnection::OnSpacemouseEvent(TKVCEvent event,  
                                       TKVCSpaceMouseData *pSMData);
```

The method `OnSpacemouseEvent` is called when the space mouse is moved from the 0 position and reaches the 0 position. The current position is transferred in the parameter `event`, and the current values in the parameter `pSMData`.

```
int CKVCCConnection::OnDisconnect(int val);
```

The method `OnDisconnect` will be called if the client terminates the connection with the disconnect message.

```
int CKVCCConnection::GetWheelVal(TKVCHandWheelData &hwData);
```

The method `GetWheelVal` returns the current position value of the wheel in `hwData.absVal`, and the modification value since the last call in `hwData.dynVal`.

```
int CKVCCConnection::GetOverrideValue(SINT16 &overrideVal);
```

The method `GetOverrideValue` returns the current value of the override potentiometers in the variable `overrideVal`.

```
int CKVCCConnection::GetJoystickPos(TKVCJoystickData &jData);
```

The method `GetJoystickPos` returns the current joystick position in the variable `jData`.

```
int CKVCCConnection::GetSpaceMousePos(TKVCSpaceMouseData &smData);
```

The method `GetSpaceMousePos` returns the current space mouse position in the variable `smData`.

```
int CKVCCConnection::GetLedState(UINT8 ledNr , UINT8 &state);
```

The method `GetLedState` is used to request the current state of the LED transferred in `ledNr`. The result is returned in `state`.

```
int CKVCCConnection::GetContrast(UINT8 &contrast);
```

The method `GetContrast` returns the current value of the contrast setting in the variable `contrast`.

```
int CKVCCConnection::GetBrightness(UINT8 &brightness);
```

The method `GetBrightness` returns the current value of the brightness setting in the variable `brightness`.

```
int CKVCCConnection::GetVolume(UINT8 &volume);
```

The method `GetVolume` returns the current value of the volume setting in the variable `volume`.

```
int CKVCCConnection::GetScreensaverTime(UINT16 &screensaverTime);
```

The method `GetScreensaverTime` returns the current value of the screensaver setting in the variable `screensaverTime`.

```
int CKVCCConnection::GetScreensaverState(UINT8 &state);
```

The method `GetScreensaverstate` returns the current state of the screensaver in the variable `state`.

```
int CKVCCConnection::GetBacklightState(UINT8 &state);
```

The method `GetBacklightState` returns the current state of the background lighting in the variable state.

```
int CKVConnection::SetWheel(SINT16 value);
```

This method sets the value of the handwheel to the value specified in `value` and returns the last value. This value is the initial value for the absolute value returned by `OnWheelChange`.

```
int CKVConnection::SetLed(UINT8 ledNum, TKVCLedeMode mode);
```

Calling the method `SetLed` sets the LED defined in `ledNum` to the mode transferred in `mode`.

```
int CKVConnection::SetContrast(UINT8 contrast);
```

Calling the method `SetContrast` changes the value of the contrast setting on the client.

```
int CKVConnection::SetBrightness(UINT8 brightness);
```

Calling the method `SetBrightness` changes the value of the brightness setting on the client.

```
int CKVConnection::SetVolume(UINT8 volume);
```

Calling the method `SetVolume` changes the value of the volume setting on the client.

```
int CKVConnection::SetScreensaver(UINT16 screensaverTime);
```

Calling the method `SetScreensaver` changes the response time of the screensaver on the client.

```
int CKVConnection::SwitchBacklight(UINT8 backlightOnOff);
```

Calling the method `SwitchBacklight` switches on and off the background lighting on the client (`backlightOnOff = 1 / backlightOnOff = 0`).

```
int CKVConnection::WriteToFlash();
```

Calling the method `WriteToFlash` saves the contents of the Client Windows Registry in the flash memory.

```
int CKVConnection::PlaySound(UINT16 soundNr);
```

Calling the method `PlaySound` starts the reproduction of the sound with the number transferred in `soundNr`.

```
sockaddr_in GetSocketAdr();
```

The method `GetSocketAdr` returns the features of the connection;

```
SOCKET GetSocket();
```

The method `GetSocket` returns the current socket of the connection with the client.

Server Implementation

The base of the server are the two classes `CKVCServer` and `CKVCCConnection`. The class `CKVCServer` establishes and manages the connections. The class `CKVCCConnection` constitutes the actual connection. To correctly implement a server, a derivation of the class `CKVCServer` is needed.

```
class CKVCTestServer: public CKVCServer{
public:
    CKVCTestServer();
    virtual ~CKVCTestServer();

    CKVCCConnection* OnClientConnect(SOCKET socket, sockaddr_in
                                     &sockAdr);
    int OnClientDisconnect(CKVCCConnection *pConnection,
                          TKVCDisconInfo info);
};
```

In this class, the method `OnClientConnect` must be overwritten.

In this method, a `CKVCCConnection` object must be created and initialized.

```
CKVCCConnection* CKVCTestServer::OnClientConnect(SOCKET
socket,
sockaddr_in &sockAdr){
    CKVCTestConnection *pConnect = 0;
    ...
    pConnect = new CKVCTestConnection();
    if (pConnect != 0){
        pConnect->Init(this, socket, sockAdr);
    }
    ...
    return pConnect;
}
```

The method `OnClientDisconnect` can be overwritten. But it must be guaranteed that the method `CKVCServer::OnClientDisconnect` is called in this derivation.

```
int CKVCTestServer::OnClientDisconnect(CKVCCConnection *pConnection,
                                       TKVCDisconInfo info){
    ...
    return CKVCServer::OnClientDisconnect(pConnection, info);
}
```

In the derivation of the class `CKVCCConnection`, only the `Event` and `Change` method must be derived. The `Get` methods can be derived, but it must be guaranteed that the method of the base class is called before the data are used.

```
class CKVCTestConnection: public CKVCCConnection{
public:
    CKVCTestConnection();
    virtual ~CKVCTestConnection();

    virtual int OnOverrideChange (SINT16 val);
    virtual int OnWheelChange (SINT16 wheelAbsVal);
    virtual int OnKeypadEvent (TKVCEvent keyEvent, UINT8 keyNum);
```



```
virtual int OnJoystickEvent (TKVCEvent event,
                             TKVCJoystickData *pJData);
virtual int OnSpaceMouseEvent (TKVCEvent event,
                               TKVCSpaceMouseData *pSMData);
virtual int OnAliveMsg      ();
virtual int OnDisconnect   ();
virtual int GetWheelVal    (TKVCHandWheelData &hwData);
virtual int GetOverrideVal (SINT8 &overrideVal);
virtual int GetJoystickPos (TKVCJoystickData &jData);
virtual int GetSpaceMousePos (TKVCSpaceMouseData &smData);
virtual int GetLedState(UINT8 ledNum, UINT8 &state);
virtual int GetContrast   (UINT8 &contrast);
virtual int GetBrightness (UINT8 &brightness);
virtual int GetVolume     (UINT8 &volume);
virtual int GetScreensaverTime(UINT16 &screeTime);
virtual int GetScreensaverState(UINT8 &state);
virtual int GetBacklightState (UINT8 &state);
};

int CKVCTestConnection::OnOverrideChange (SINT16 val){
    cout << "OnOverrideChange: " << (int)val << endl << flush;
    return true;
}

int CKVCTestConnection::GetOverrideVal (SINT8 &overrideVal){
    CKVCConnection::GetOverrideVal(overrideVal);
    cout << "GetOverrideVal: " << (int)overrideVal << endl << flush;
    return true;
}
```

Remote Software ActiveSync

This software is used as a debug interface and to adjust and transmit data from and to a PC.

The remote software ActiveSync is a product by Microsoft and can be downloaded free of charge from the following download address:

<http://www.microsoft.com/windowsmobile/downloads/activesync38.msp>

The data connection from the KeTop to the PC is established as follows:

- ▶ 1. Remove the cover of the cable entrance area on the KeTop.
- ▶ 2. Plug in the Download Cable KeTop XD 040
- ▶ 3. Start the ActiveSync software on the PC
 - a) select 'File' -> 'Delete Partnership' ...
 - b) deactivate '() Allow serial cable or ... ' -> <OK>
 - c) activate '(x) Allow serial cable or ... ' -> <OK>
- ▶ 4. Start the ActiveSync-Software on the KeTop:
Start -> Programs -> Communication -> Active Sync
- ▶ 5. At the PC:
 - Connect KeTop as a "Guest" (**Set Up a Partnership -> No**).

Now the data connection between the KeTop and the PC is established. In case of communication problems, check the following conditions:

- Max. time delay between step 3c) and step 4. = 30s.
- For further attempts repeat steps 3. to 5.
- The ActiveSync software must be started manually.
(Plugging in the connection cable does not start ActiveSync!)
- The factory-set baudrate for the remote connection is 115 kBaud.
In case of communication problems the baudrate can be reduced.
The baudrate can be changed as follows:

- Start -> Programs -> Communication -> Remote Networking
- select "com1_115k"
- File -> Properties
- SP1 on COM1: -> Configure -> Port Settings
(...change Baud Rate...)
- Close all Windows with <OK>

(optional: Start -> Programs -> KeTop -> Registry Backup -> <OK>)

Test tool (VC++ Demo)

Requirements

The reader must have knowledge of object-oriented programming, embedded C++ and MFC. The graphs are displayed in UML standard according to OMT.

Task of the test tool

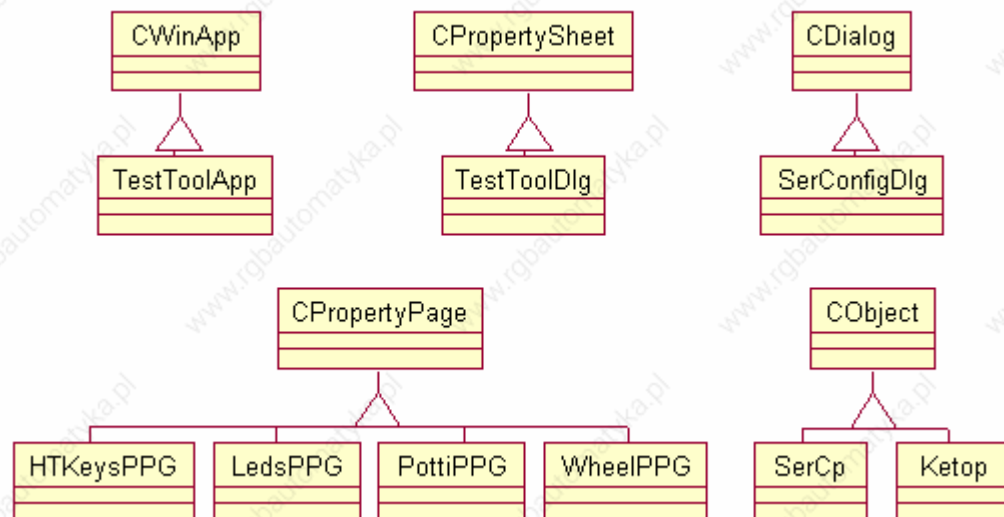
The test tool is a C++ demo and is used to display values that are made available by the KetopAPI, such as wheel values, potentiometer values, key states and LED states, on the screen.

These values are transmitted to the control by the SER protocol (CP001 data profile), immediately read out again from the control and also displayed on the screen.

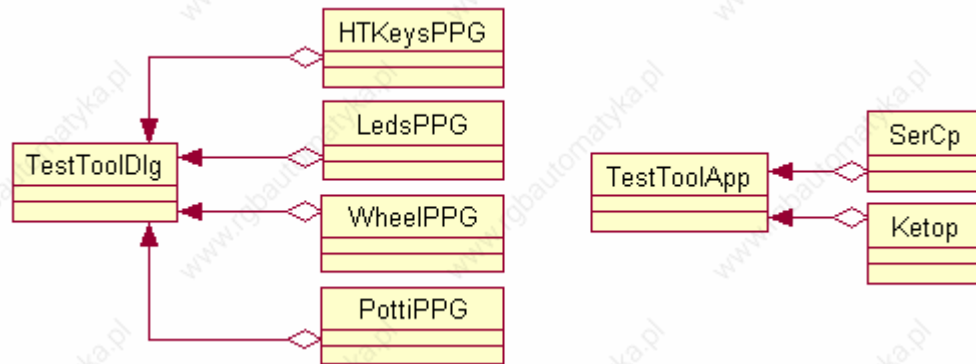
This test tool has been realized in embedded Visual C++ 3.0© with the application of the MFC© (for WinCe).

Description of the classes

Class hierarchies



Class relations

**TestToolApp**

This class is the actual application that is displayed on the screen. The Init method creates and administrates the dialog (represented by the class TestToolDlg). So that the application can be accessed in a simple way during the running time of the program, this class always contains a member variable at both the SerCp and Ketop levels of abstraction.

TestToolDlg

This dialog class contains the individual tabs that are realized by the PropertyPage classes.

HTKeysPPG

This class represents the HTKeys tab and is used for visualization of the keystrokes.

LedsPPG

This class is responsible for administration of the LEDs and is displayed as the LED tab.

PottiPPG

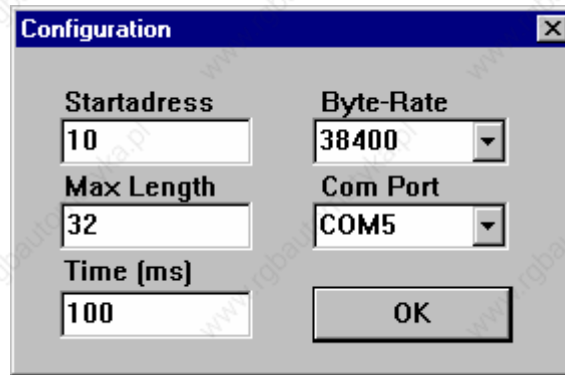
This class takes care of all values of the potentiometer and is visualized in the Potentiometer tab.

WheelPPG

This class displays all changes to the wheel in the Wheel tab.

SerConfigDlg

This class implements the functionality of the "configuration dialog" (see following figure) that is displayed on the screen when the application is started. This enables the user to enter his own port configuration.



Since the values of this dialog must already be available for the initialization of the port, the dialog is held by the SerCp class as, from a design point of view, it fits best here.

SerCp

This class forms a level of abstraction above the CP001 driver and should simplify communication with the control for the application.

The Constructor loads the SerCp001.dll. Moreover, function pointers to the method of the SerCp001.dll that is used are immediately created and the Callback function that is used for reading out values to the control, is hung up. The Destructor closes the port and releases the DLL. This class provides a "simplified" interface for communicating with the control.

Ketop

This class also forms a level of abstraction above the KetopAPI and, thus, should simplify access to it. It provides a somewhat "simplified" interface to the outside world.

The Constructor loads the KetopAPI.dll. Also in this case, function pointers to the required method are retrieved (however, in a somewhat different manner than in the SerCp class) and a Callback function, which is called automatically by the KetopAPI during modification, is hung up.

Procedure using the example of the test tool

The Init of the TestToolApp class firstly initializes the member variable (mSerCp) of the SerCp class that sees to the configuration of the driver. At the same time, the configuration dialog (SerConfigDlg) is called, the entered values are stored and, with them, the port to the control is opened and configured. The TestToolDialog is then called (in the TestToolApp class) and displayed on the screen. This starts with the start of a timer that issues a cyclical write command with the read command to the control. (Without the timer, the application would never be informed about modifications to the control because, in this test example, the control never becomes active by itself.)

If, for example, the wheel is now rotated, the Callback function of the Keto-pAPI is called automatically. There, a Windows message is sent, and received and processed by the test tool application which, in this case, is synonymous with a graphical update of the tab that is currently active. This modified value is then sent to the control by a write command.

The started timer cyclically informs the control that it would like to read. The control calls the Callback function registered with it and thus triggers a Window message. This, in turn, is received by the application. In the next step, a read command is sent to the control and the tab that is currently active is determined. The modified values are handed over to this tab and only this tab carries out an update of the screen.

General procedure for communicating with the control using the SerCp001.dll

1. Load the library (SerCp001.dll)
2. Retrieve the function pointers to the required method
3. Install the Callback function
4. Open and configure the port
5. Use the methods that are made available by the DLL
6. Close the port again
7. Release the library

8 KeTop - Specific Operating Instructions

This chapter describes the KeTop-specific settings and the differences to standard Windows CE devices.

Notice

- *The date and the time are not stored in the KeTop and, if needed, must be set anew after turning off/on. The date and the time may be very important for the log data for example.*

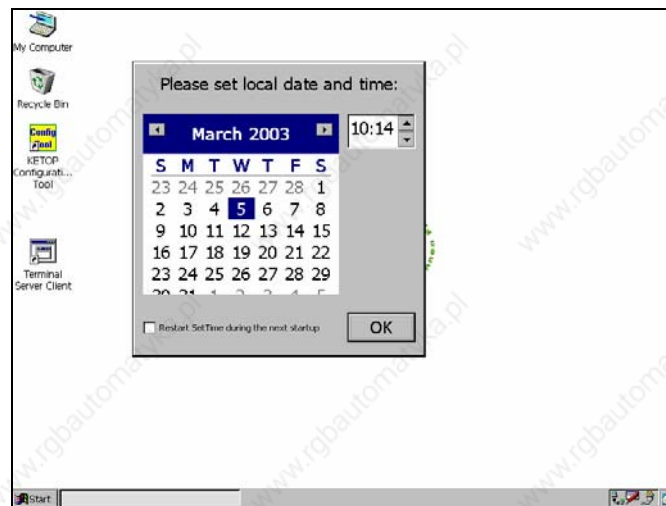
Notice

- *At settings that have not been made with the ConfigTool, the modifications must be saved as follows:*

*Start -> Programs -> KeTop -> Registry Backup
(or in the explorer with the command „\windows\regflush.exe“)*

Setting of Date and Time

During start-up the KeTop shows automatically the window for setting date and time:



Should the window for date and time not be shown, the checkbox has to be deactivated. This must be saved in the registry ("Start -> Programs -> KeTop -> Registry Backup").

The KeTop is not equipped with a real time clock. So, if you use the time it must be set during each star-up.

KeTop Configuration Tool (ConfigTool)

The ConfigTool is used to calibrate the operating elements, to control the functionality of the operating elements, to select the Ethernet interface and to set start-up functions.

To activate the tool, select the following item:

Start -> Programs -> KeTop -> KeTop Configuration Tool
(or by double-clicking on the file „Configuration_Tool.exe“ in the Windows directory).

Notice

- *The appearance of the Config tool depends on the operating elements installed in KeTop (the menus for override potentiometer, joystick and handwheel are only displayed when the operating elements are installed in the device).*

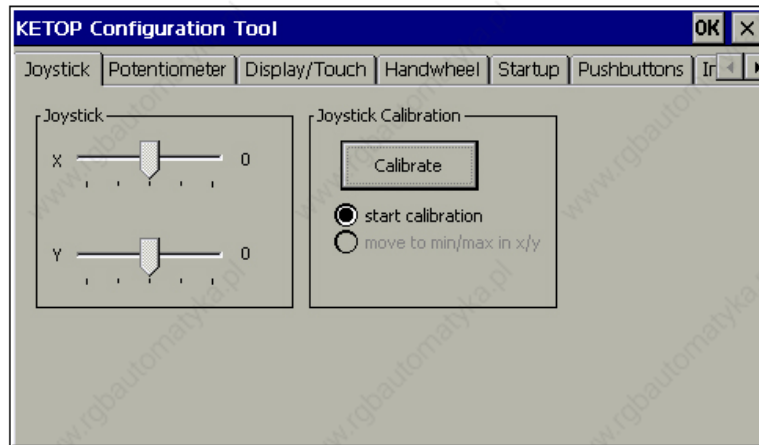
After carrying out modifications on the configuration and quitting the program, you can save all modifications in the registry. A window with the following message and options appears:

Data has changed! Do you want to write the registry to the flash? Yes/No

With „No“, the settings are contained in the DRAM and will not be stored therefore. They will get lost when the KeTop is turned off.

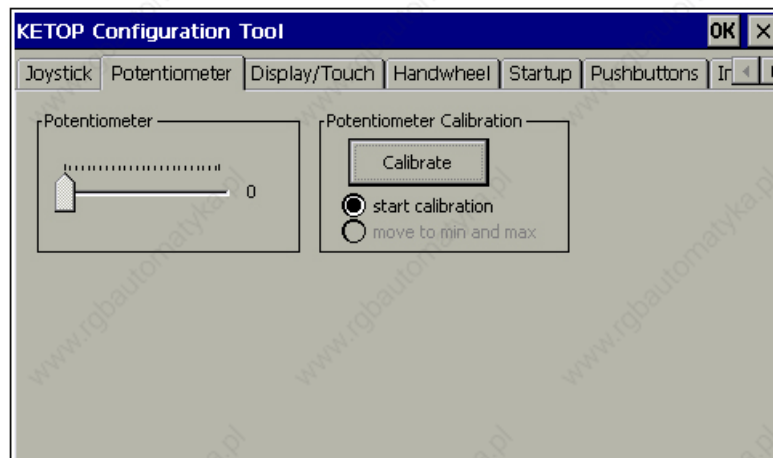
„Yes“ saves the calibration in the flash. The settings are preserved after a restart of the device. The device is blocked during the saving procedure (takes approx. 10 s).

Calibration of Joystick



A number is displayed next to the slide bar. During calibration, this number specifies the current value of the ADC (value range: 0..1024) and only controls the function. After calibration, the number specifies the current value of the joystick/potentiometer (value range: -15 .. 15 for each joystick axis). This value is also the value the KeTop API function *KtpGetJoystickPos* supplies to the application.

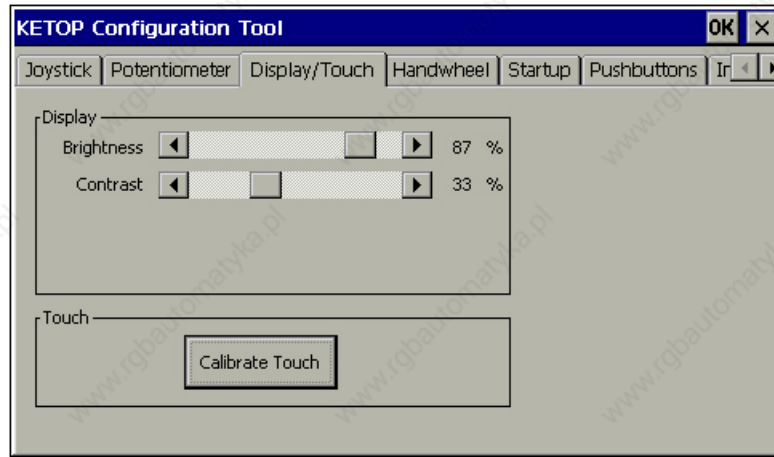
Calibration of Potentiometer



A number is displayed next to the slide bar. During calibration, this number specifies the current value of the ADC (value range: 0..1024) and only controls the function. After calibration, the number specifies the current value of the potentiometer (value range: 0 ...127). This value is also the value the KeTop API function *KtpGetOverridePoti* supplies to the application.

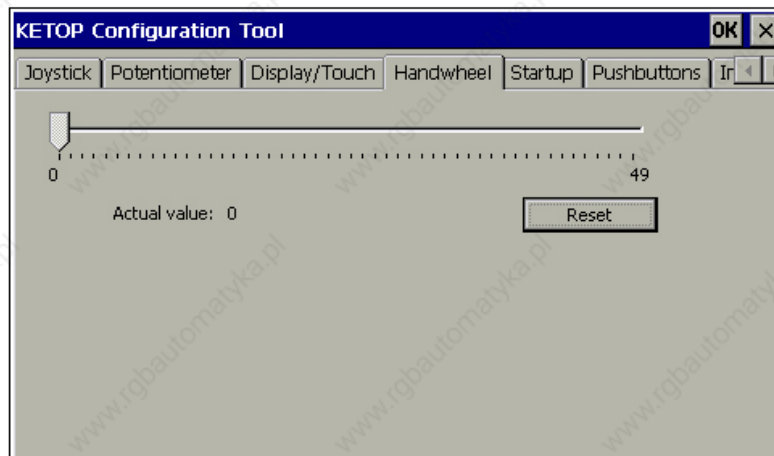
Display and Touch Screen Settings

In this window, the brightness and the contrast for the display are set.



In this window you can also calibrate the touch screen.

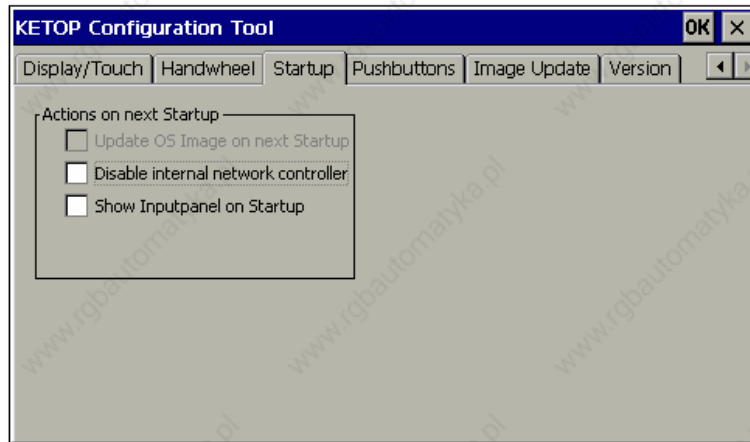
Setting the Handwheel to Zero



The handwheel outputs a 16-bit value which can be processed in the customer application as needed (e.g.: 0-65535, +-32767, ...). In the dialog box, the value from 0 to 65535 is displayed. When the device is turned on, the value of the handwheel will be set to 0. If the key „Set“ is pressed, the current value of the handwheel will be replaced by the start value.

Actions on next Startup

In this window, you can determine actions that are executed after a restart of the KeTop.



- **Update OS Image on next Startup:**
If you select this item, an OS(Operating System) image will automatically be loaded from the BOOTP Server via the network next time the KeTop is started.

Note

In this case, the Ethernet connection must have been established, the BOOTP Server must be correctly configured and an OS image file must be available on the server.

- **Disable internal network controller:**
If you select this option, the internal Ethernet interface „CELAN1: On-board Ethernet“ (see *Start -> Settings -> Control Panel -> Network -> Network Configuration*) will be deactivated during the next start-up.

Note

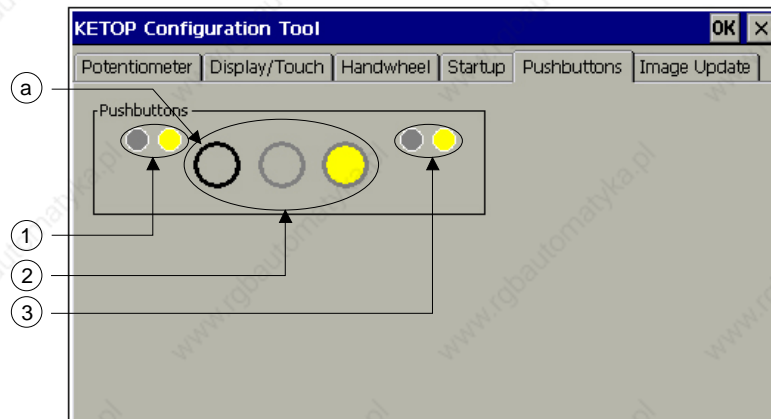
This function is needed, for instance, for the installation of an Ethernet interface in the PCMCIA slot.

The actual configuration of the Ethernet controllers must be performed in the system control.

- **Show Inputpanel on Startup:**
If you select this item, the softkeyboard will be launched at every startup.

Control of Buttons (Pushbuttons)

This toolbox enables a simple functional control of the buttons of the Ke-Top.



- 1 Symbols for the left LEDs of the membrane keypad (if available)
- 2 Symbols for operating elements
- 3 Symbols for the right LEDs of the membrane keypad (if available)
- a The black ring signalizes the actuation of an operating element

By clicking on one of the three symbols for operating elements (2), the corresponding LEDs of the operating elements can be switched on, off or to flashing. The first click switches the LED to flashing, the second click switches the LED on, and the third click switches the LED off again.

The same applies to the LEDs of the membrane keypad (1 and 3).

The actuation of an operating element is signalized by a black ring (a). This ring is grey if the operating element is not actuated.

Loading an Image File

This toolbox enables loading an image file into the KeTop.

Note

- *The image file may be copied to every directory excepting to directory \IPSM.*
- *Before each image update execute „Check file“ necessarily.*

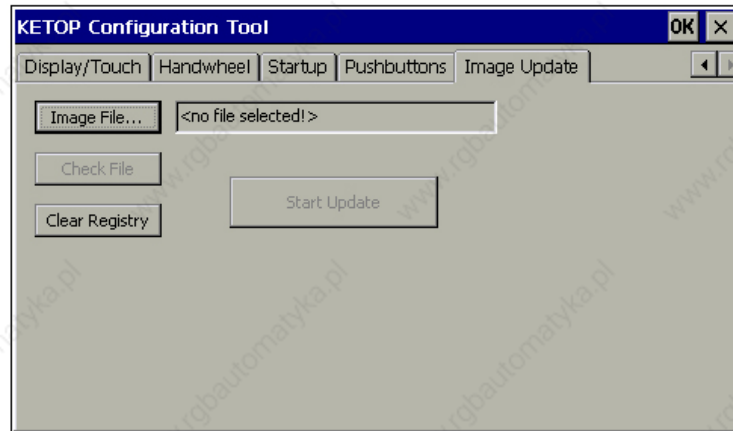


Image File

Clicking this button enables you to select a new WinCE image file in the file selection dialog.

Note

In the event of an image update, the registry will be deleted and the KeTop rebooted automatically. Following that, the KeTop reboots with the factory settings, and all settings (touch screen, joystick, override potentiometer, system control,...) must be stored anew.

Check File

The selected file can be checked for validity. "Check File" will only be active if a valid image file has been selected.

Start Update

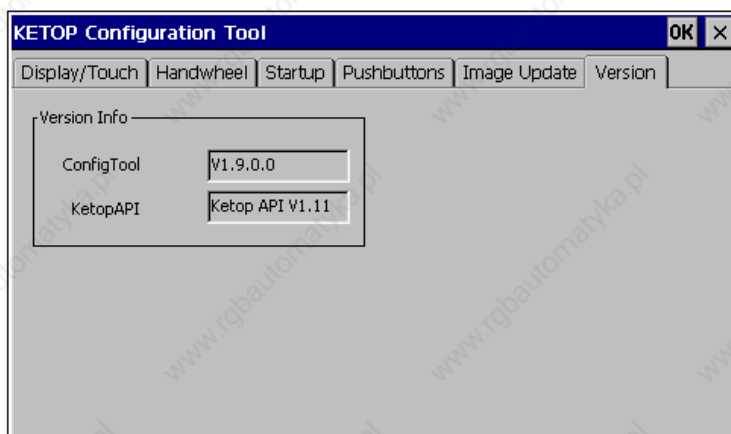
This button starts the update. The progress status is displayed in the bar below. "Start Update" will only be active if the selected image file has been successfully checked for validity (with "Check File").

Clear Registry

This item clears the registry and automatically reboots the KeTop. Following that, the KeTop reboots with the factory settings, and all settings (touch screen, joystick and override potentiometer, system control,...) must be stored anew.

Version Info

This window shows the actual version of the ConfigTool and KetopAPI of the KeTop.



Checking the Operating and Control Elements

For checking the functionality of the operating and control elements of the KeTop, the KeTop provides the software „Systemcheck“. This software is contained in the start menu and can be operated intuitively. It is called as follows:

Start -> Programs -> KeTop -> „System Check“

Provided they are exist on the device, the following operating and control elements can be checked:

- Override potentiometer
- Electronic handwheel
- Joystick
- Status LEDs
- Membrane keyboard
- Touch screen
- Display
- Illuminated push button
- Key switch
- Selector switch
- PC card interface (slot), if existing.

The following control elements CANNOT be checked since their internal evaluation via the electronics is not provided:

- Emergency stop switch
- Enabling switches

Installation of Programs

Programs can be only installed in the IPSM (Intel Persistent Storage Manager) directory. Since only this directory is stored in the flash, data installed in another way will get lost after turning off the KeTop.

Installation data entered in the registry can be saved with the command „\Windows\Regflush.exe“.

Notice

- *System files, which are copied to the directory \windows while installation, do not remain durably stored. For this reason these files must be copied manually into the directory \IPSM\windows (if necessary, this directory must be created by the user). With a restart of Windows CE these files are copied automatically into the directory \windows. So, these files are again available for the operating system and are again present also after a restart (KeTop T100: valid from OEM Build Vers. 1.4a; KeTop T50: valid from OEM Build Vers. 1.2).*

Saving Files

Files must be saved in the IPSM directory. The contents of other directories are not saved when the device is turned off.

Transferring Files

Transfer via external storage device (PC-card, CF-card, USB-Stick)

For further information on suitable storage devices please refer to the section "PC-card slot for PC-cards I, II, III" and/or "Compact Flash - Card Slot" in the chapter "Connection" or "USB-Stick".

- ▶ Insert the external storage device into a suitable PC.
- ▶ Copy the required files to the external storage device.
- ▶ Insert the external storage device into the KeTop.
- ▶ Copy the required files from "\storage card" or "\hard disk" to "\ipsm".

Transfer via network connection by SMB access from PC to KeTop

This kind of SMB access is only possible with Windows XP Professional and some Versions of Windows 2000.

For general information see the ", section "Ethernet" in the chapter "Connection".

The KeTop must have a valid Ethernet address.

In case there is no DHCP-server in the network, the IP-address can be set via "Start → Settings → Control Panel → Network and Dial-up Connections → CELAN1... → Properties..."

To save the IP-address go to "Start → Programs → KETOP → Registry Backup" and restart the KeTop.
(The Ethernet-connection can be checked with "Ping".)

Then enter the IP-address of the KeTop in the address line of the Windows Explorer at the PC. (e.g.: \\192.168.30.107)

The PC now has full access to the directory structure at the KeTop so that all required files can be copied from the PC to "ipsm" at the KeTop.

Transfer via network connection by access from KeTop to the PC

For general information see the section "Ethernet" in the chapter "Connection".

The KeTop must have a valid Ethernet address.

In case there is no DHCP-server in the network, the IP-address can be set via "Start → Settings → Control Panel → Network and Dial-up Connections → CELAN1... → Properties..."

To save the IP-address go to "Start → Programs → KETOP → Registry Backup" and restart the KeTop.
(The Ethernet-connection can be checked with "Ping".)

Then enter the IP-address of the KeTop in the address line of the Windows Explorer at the PC. (e.g.: \\192.168.30.107)

After start-up of the KeTop start the Windows Explorer at the KeTop. (Start → Programs → Windows Explorer)

The shared directory of the PC can now be addressed in the address line of the Windows Explorer at KeTop with "\\<pcname>\<share name>". (e.g.: \\tkg007\temp)

The directory at the PC must be shared and a user with the necessary rights to access must exist.

The files can now be copied from the PC to "\ipsm" at the KeTop.

Transfer via ActiveSync

Therefor see chap. „Remote Software ActiveSync“.

9 Options

This chapter describes the options the KeTop is available with.

Override Potentiometer

If available, the override potentiometer in the KeTop is evaluated by the software and can be read via the KeTop-API using a program.

The override potentiometer can be used for example for setting the number of spindle revolutions and the forward feed on machine tools.

- Resolution: 0 – 127, linear

Electronic Handwheel

If available, the electronic handwheel in the KeTop is evaluated by the software and can be read via the KeTop-API using a program.

50 pulses are counted per revolution. Turning the handwheel clockwise increments the count, turning the handwheel counter-clockwise decrements the count (-32768 to +32767, 16-bit value).

Main features

- 50 lock-in positions / revolution
- 1 pulse / lock-in position
-

Notice

- *If the KeTop falls to ground, control the position of the handwheel knob. If necessary press down the knob until it snaps in.*

Illuminated Push-Button

If available, the illuminated push-buttons in the KeTop are evaluated by the software and can be read via the KeTop-API (see page 49) using a program.

The illuminated push-buttons are available as „momentary“ type (labeled 0 or I) or as „maintaining“ type (labeled 0/I).

WARNING

- After removing the covers of the push-buttons (in case of changing the labelling) take care that the labelling plates are positioned again correctly referring to the functionality of the push buttons. (Otherwise the wrong push-button may be used for switching ON or OFF the motor.)

Key Switch

If available, the key switch is evaluated by the software in the KeTop and can be read via the KeTop-API using a program.

The key switch is available with 3 positions labeled „I-0-II“ or „0-I-II“.

Selector Switch

If available, the selector switch is evaluated by the software in the KeTop and can be read via the KeTop-API using a program.

The selector switch is available with 3 positions labeled „I-0-II“ or „0-I-II“.

Joystick

If available, the 2-axis joystick in the KeTop is evaluated by the software and can be read via the KeTop-API using a program.

To avoid that the joystick is damaged when the device falls to ground, a short joystick is used. The joystick enables moving robot axes for example.

- Value range: -15 to +15 per axis (31 increments)

In the KeTop there is only space for the optionally 3-axis joystick or a flash card slot.

10 Accessories

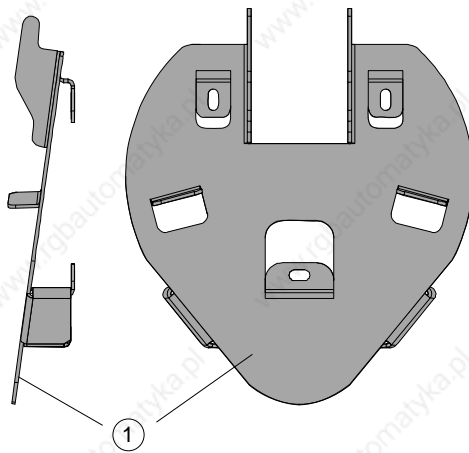
Wall bracket KeTop WB090 and KeTop WB095

The powder-coated black wall bracket is used for stationary operation or storage of the KeTop.

Two types of wall brackets are available:

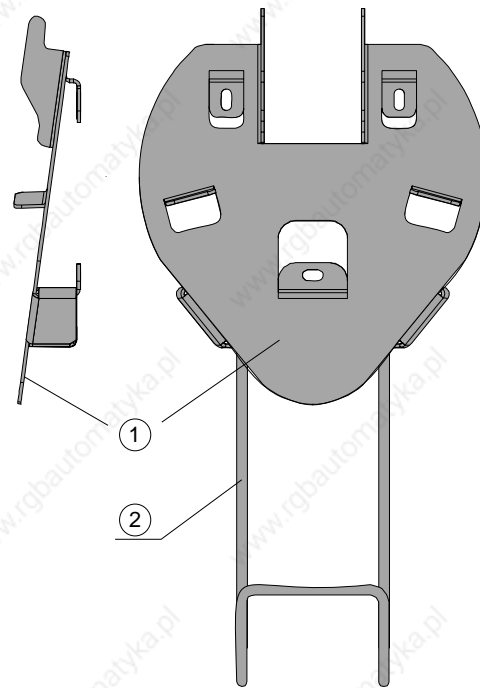
KeTop WB090

Wall bracket without cable suspension



KeTop WB095

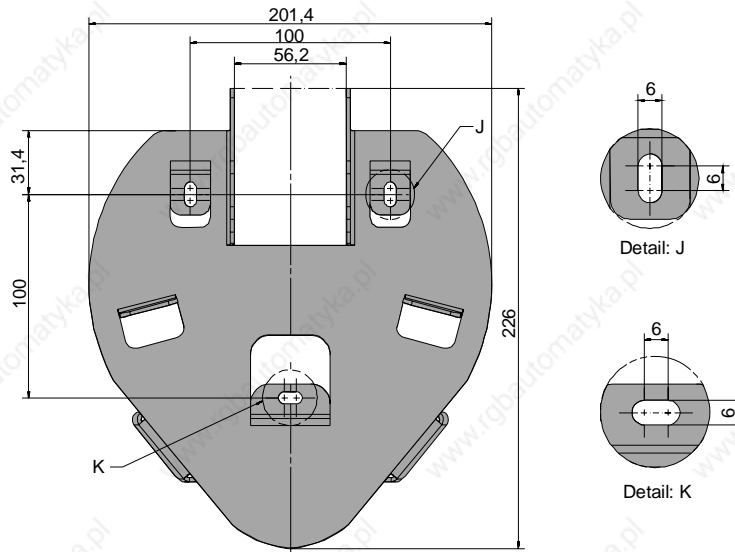
Wall bracket with cable suspension



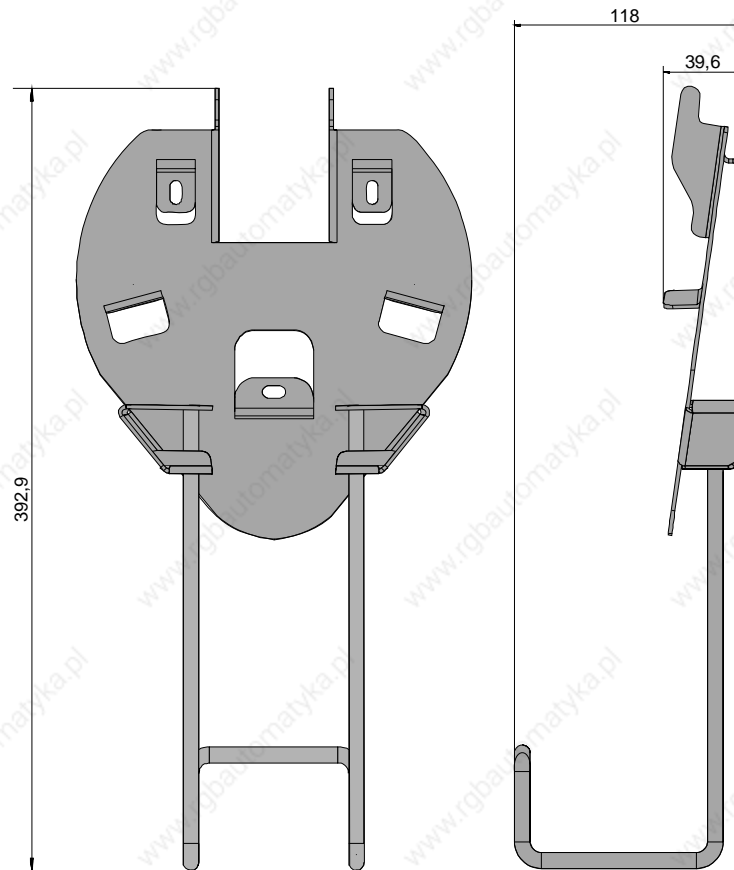
- 1.....Height adjustment plate
2.....Carrier

Wall bracket KeTop WB100 and KeTop WB110 with KeTop

Dimensions (mm)



Wall bracket KeTop WB090, front view



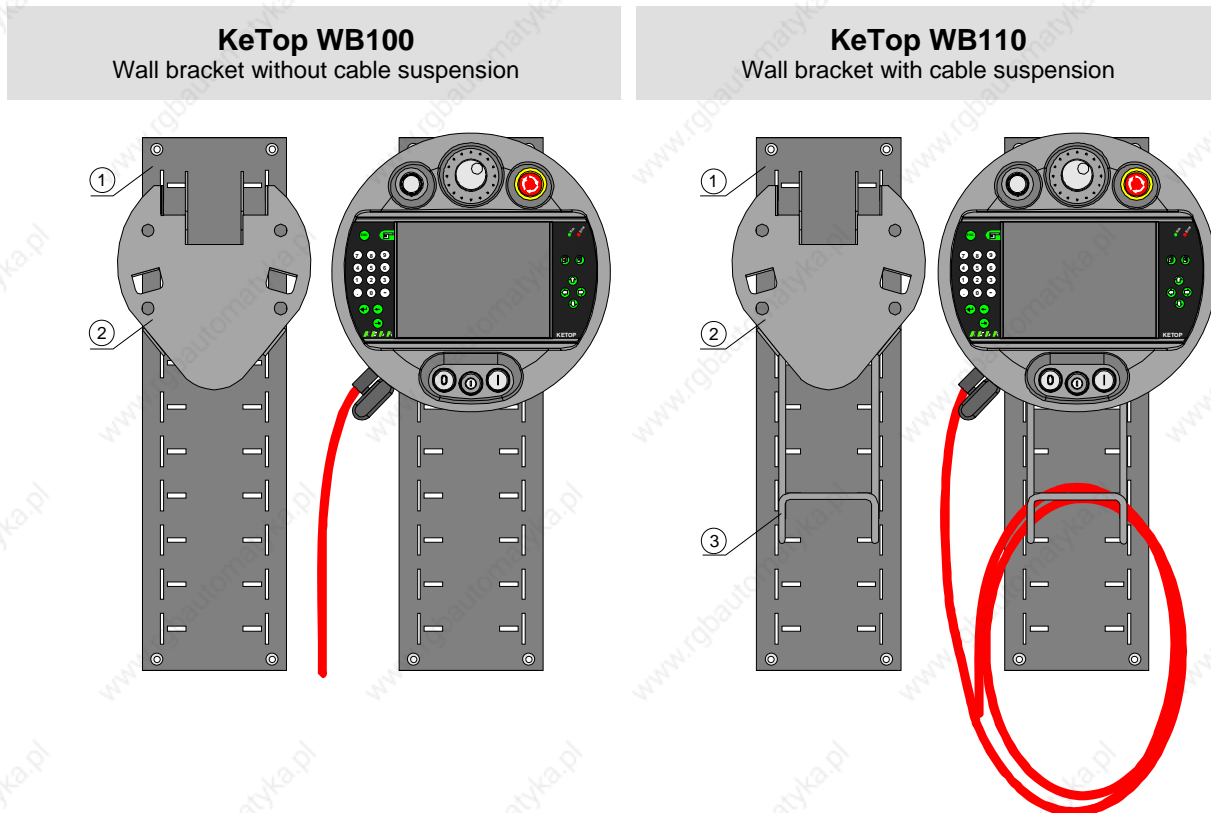
Wall bracket KeTop WB095, rear and side view

Wall Bracket with Height Adjustment plate KeTop WBxxx

The powder-coated black wall bracket is used for stationary operation or storage of the KeTop.

The carrier is adjustable in 8 positions over a height of 320 mm (12.6 in). Take care to hang up the carrier in all 4 points in the height adjustment plate. The cable suspension must be mounted on the carrier using the screws delivered with the device.

Two types of wall brackets are available:

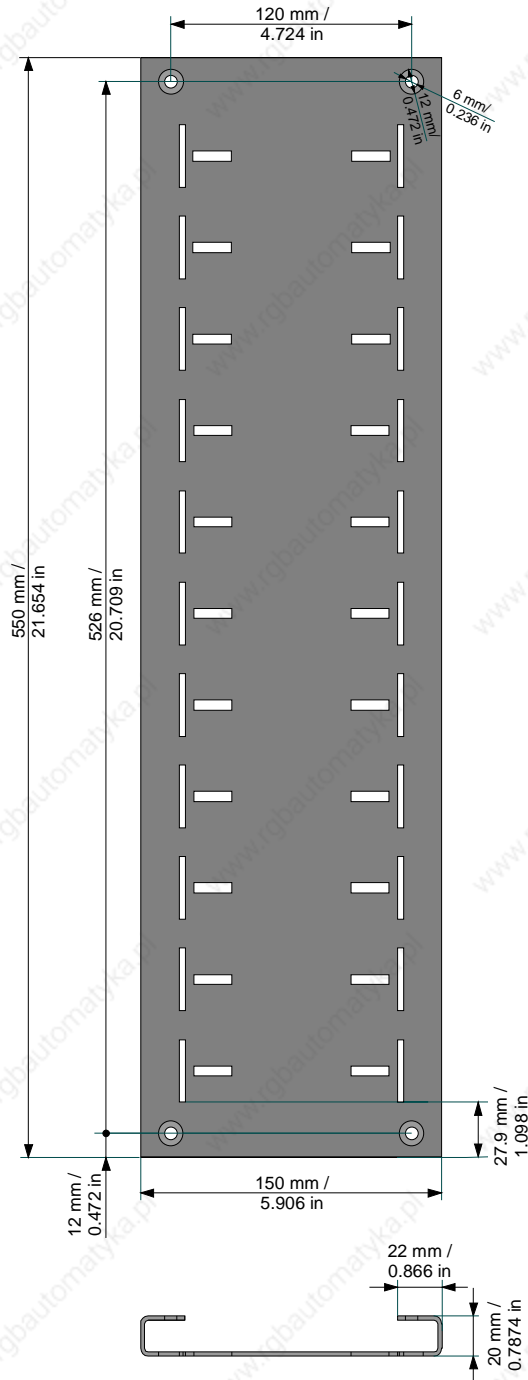


- 1Height adjustment plate
- 2Carrier
- 3Cable suspension

Wall bracket KeTop WB100 and KeTop WB110 with KeTop

Height adjustment plate

For mounting the height adjustment plate, use suitable screws (not part of delivery).

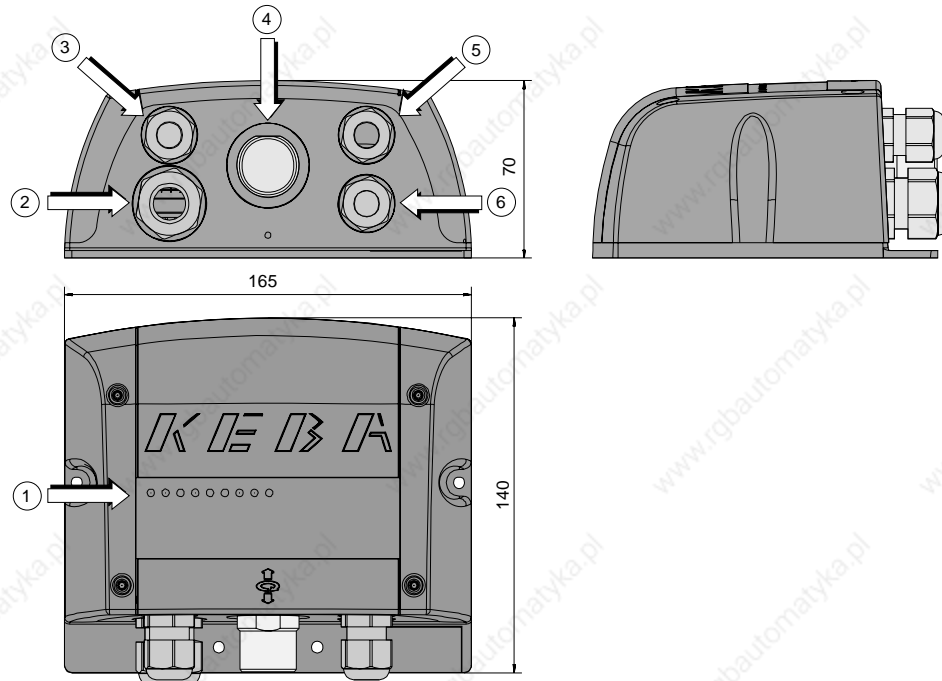


Height adjustment plate for wall bracket WB110

KeTop CB211 Connection Box

The KeTop CB211 connection box is used for integration of the KeTop in the machine/system. It is suitable for wall mounting and can also be mounted on a mounting rail through the use of the mounting rail assembly kit (KeTop DR200). The KeTop CB211 connection box has the following connections:

Construction



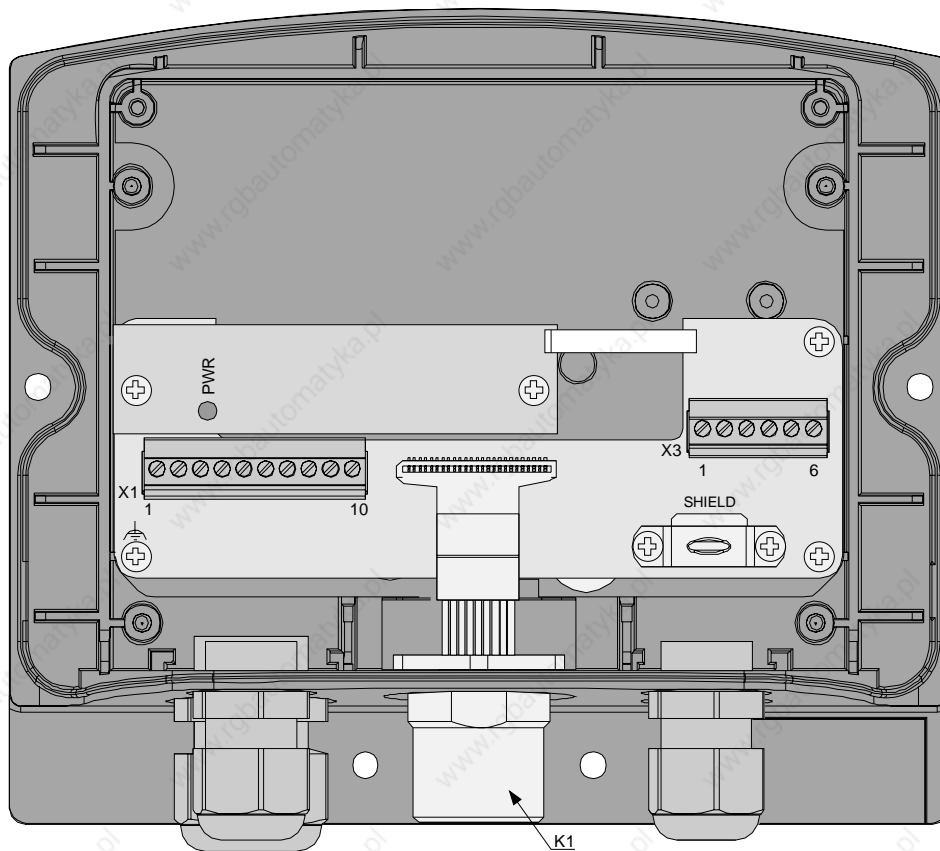
- 1.....Status and error LEDs
- 2.....PG gland (M20) for voltage supply, enabling switch and emergency stop
- 3.....PG gland (M16) for separate functional ground (status as supplied with dummy plugs)
- 4.....Coninvers female connector for KeTop T100, T50 and T40 - connection cable
- 5.....PG glands (M16) for data lines
- 6.....PG glands (M16) for data lines (status as supplied with dummy plugs)

View and device description of the Gateway box

⚠ WARNING

- **The connection box and the handheld terminal meet the safety class III in accordance with EN61131-2 and EN50178.**
When connecting the handheld terminal, make sure that all voltages connected to the handheld terminal are safety extra low voltages and isolated from the low voltage supply system by a safety transformer or a similar facility.

Interior view



- K1..... 17-pin female connector (Coninvers) for KeTop connection cable
 X1..... Terminal block for power and control lines (enabling switch and emergency stop)
 X3..... Terminal block for data lines
 SHIELD..... Cable shield clamp with connection surface for cable shield of data lines
 (not used for strain-relief of the cable!)

Interior view of connection box

Details regarding connection of the KeTop CB211 connection box see sub chapters "Ethernet" and "RS-422-A", both are parts of the main chapter "Connection".

Technical data of the connection terminals

The following technical data apply to the X1 and X3 connector terminal blocks already available in the Junction box:

Connection capacity:		
rigid / flexible / wire gages	[mm ²]/[mm ²]/AWG	0.14-1.5 / 0.14-1.5 / 28-16
flexible with wire end ferrules without / with plastic sleeve	[mm ²]	0.25-1.5 / 0.25-0.5
Grid dimension:		3.81
Insulation length:		[mm ²] 7
Tightening torque:		[Nm] 0.22-0.25

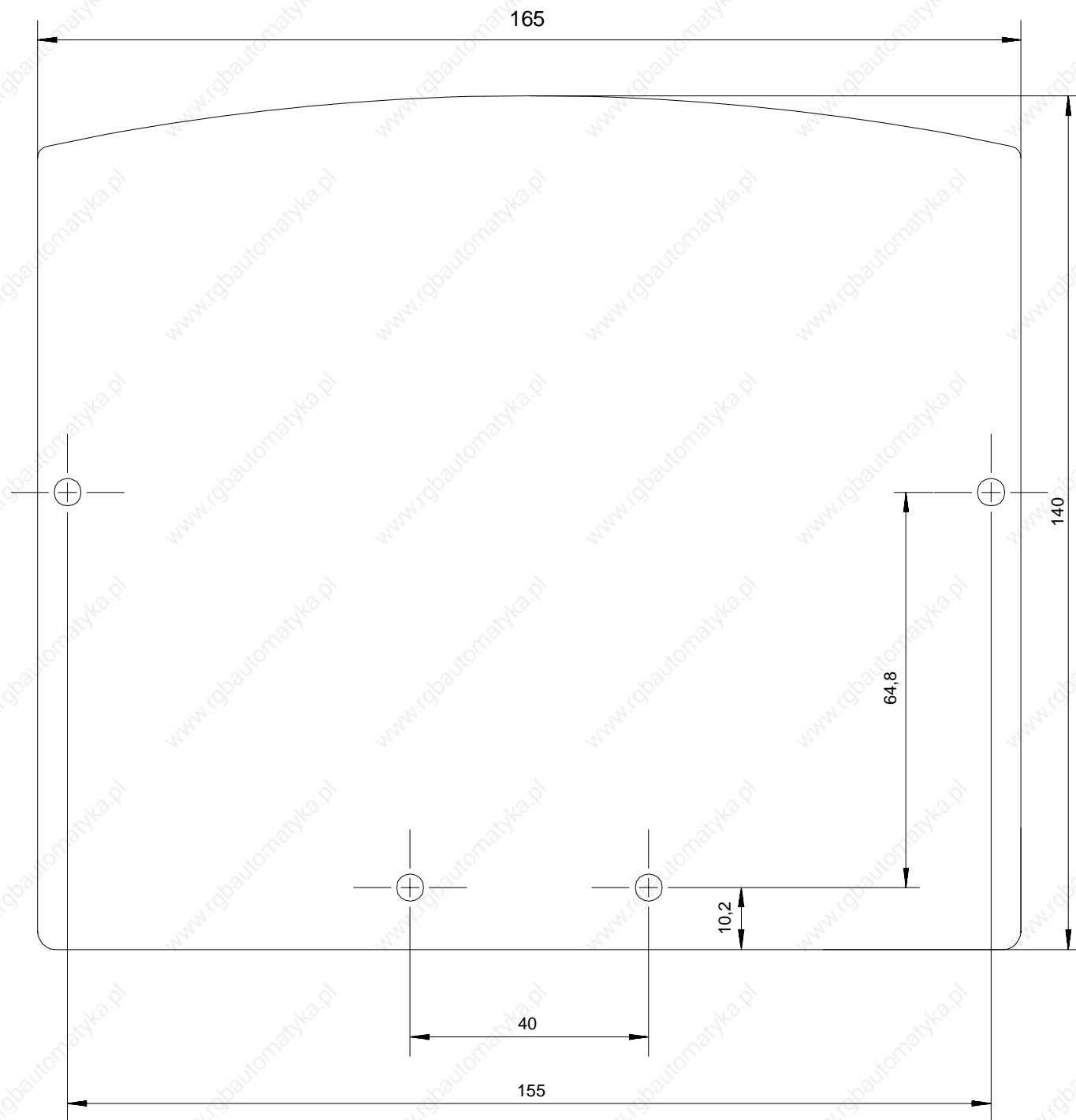
PHOENIX order data:

Gateway-terminal block	PHOENIX Type	Part no.
X1	MCVR 1.5/6-ST-3.81	1827169
X3	MCVR 1.5/7-ST-3.81	1827172
X4, X4B	MCVR 1.5/10-ST-3.81	1827208

Notice

- Consider the connection capacity of the terminal blocks when selecting the connection cable.
- Use the following screwdriver to connect the wires to the terminal blocks:
Blade: 0.4 x 2.5 x 80 mm,
Length: 160 mm
- Multi-line connections (2 wires in one terminal) are not allowed. Use the X4B terminal block for continuing the field bus.

Drilling template for wall mounting



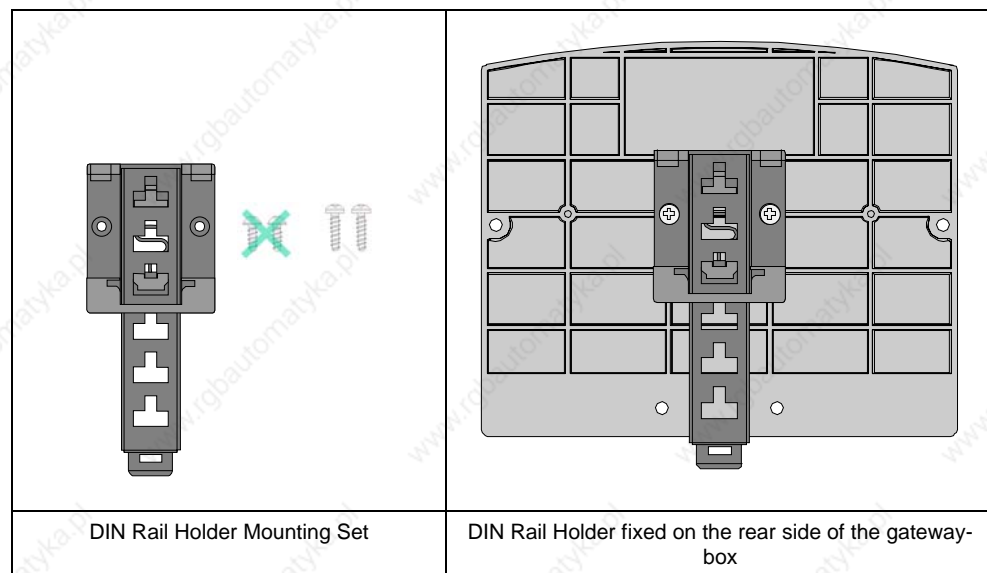
Drilling template for the Gateway box, scale 1:1

For wall mounting, we recommend the following screws and materials:

- Chipboard screw: \varnothing 4 x 40mm
Head form: flat head
Max. head diameter: \varnothing 9mm
- Recommended rawlplug: \varnothing 6 x 30mm

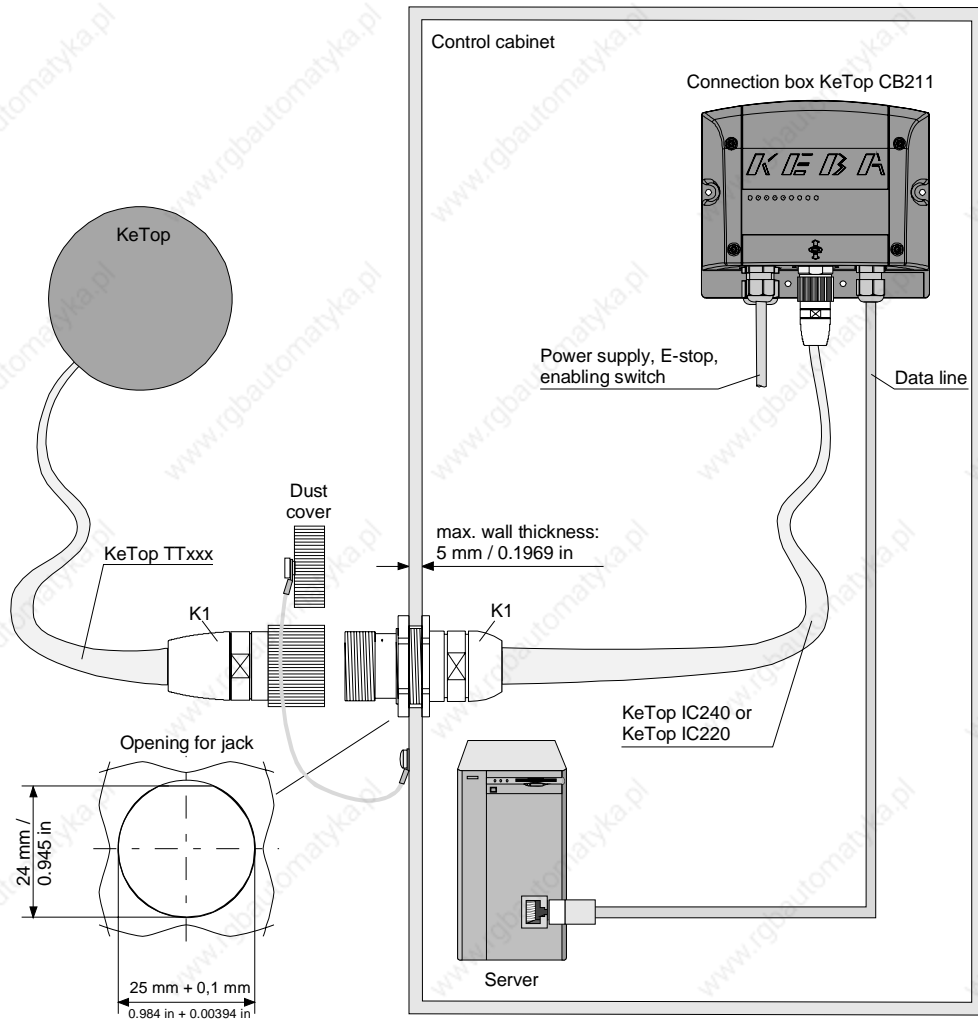
DIN Rail Holder Mounting Set KeTop DR200

The DIN rail holder KeTop DR200 is available as accessory and will be mounted on the rear side of a KeTop CB2xx gatewaybox. So the gatewaybox can be easily snapped onto a DIN rail.



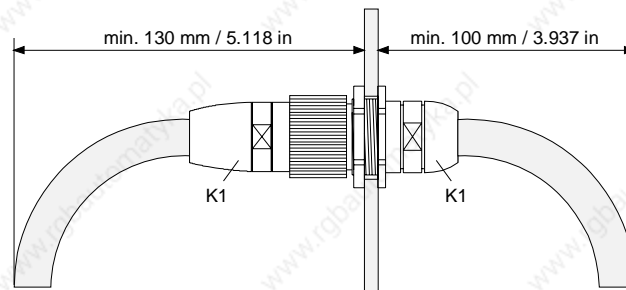
DIN Rail Holder Mounting Set for gatewaybox

Use of Connection Box



Connection box KeTop CB211 in control cabinet

Minimum Bending Radius of Cable



Required minimum distance outside and inside the control cabinet

Technical Data of Connection Box KeTop CB211

General data

Nominal supply voltage:	24 V DC (Safety extra low voltage)
Supply voltage range:	18 V DC to 32 V DC
Maximum interruption time of supply voltage:	≤ 10 ms (lt. IEC 31131)
Power consumption:	
with KeTop:	3.6 W (200 mA at 18 V DC, 150 mA at 24 V DC)
without KeTop:	10.8 W (600 mA at 18 V DC, 450 mA at 24 VDC)
Inrush current:	max. 5.6 A (with limitation of current)
Safety class:	III in accordance with EN 61131-2 and EN 50178

Environmental conditions

Operating temperature:	0 °C to 50 °C (32 °F to 122 °F)
Storage temperature:	-20 °C to +70 °C (-4 °F to 158 °F)
Relative humidity (non-condensing):	5 % to 95 %
Vibration resistance (operation):	(IEC 60068-2-6)
	5 ≤ f < 9 Hz 7 mm
	9 ≤ f ≤ 150 Hz 2g (0.0044 pound)
Shock resistance (operation):	15 g (0.033 pound) / 11 ms (IEC 60068-2-27)

Housing

Construction	Double-walled ABS housing Withstands grease, oil, lubricants, alcohol, etc.
Flammability class:	UL94-V0
Dimensions:	
Width:	160 mm
Height:	140 mm
Depth:	70 mm
Protection degree:	IP65
Weight:	500 g (1.1 pound)
Display:	Status LEDs

Accessories

Intermediate cable	Connection box to connection cable
KeTop IC220:	2 m / 6.56 ft
KeTop IC240:	4 m / 13.12 ft
Download cable	
KeTop XD040:	4 m / 13.12 ft. For downloading software and for debugging via S2.
DIN Rail Holder Mounting Set	
KeTop DR200:	For mounting on the rearside of a KeTop CB2xx gatewaybox.

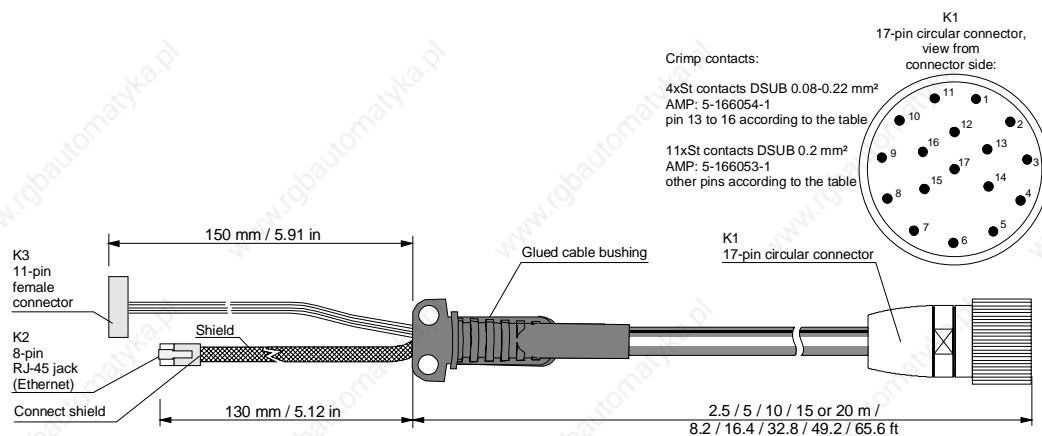
Connection Cable KeTop TTxxx

The standard KeTop handheld terminals are available with the following cables:

- KeTop TT025 (2,5 m)
- KeTop TT050 (5 m)
- KeTop TT100 (10 m)
- KeTop TT150 (15 m)
- KeTop TT200 (20 m)

The KeTop connection cable withstands water, cleaning agents (alcohol and tensides), oil, drilling oils, grease and lubricants.

Description of signal	K3, 11-pin female connector to S19 on the KeTop	K2, 8-pin RJ-45 jack (S6, RS-422-A)	Connection cable KeTop TTxxx, color of wires		K1, 17-pin male connector, pin No.:
24 VDC	6	-	pink	->	1
GND_IN	7	-	black	->	2
E-stop, circuit 1	8	-	brown-green	->	3
E-stop, circuit 1	9	-	white-green	->	4
E-stop, circuit 2	10	-	grey-pink	->	5
E-stop, circuit 2	11	-	red-blue	->	6
enabling switch, circuit 1, pos.	1	-	brown	->	7
enabling switch, circuit 1, neg.	2	-	yellow	->	8
enabling switch, circuit 2, pos.	3	-	green	->	12
enabling switch, circuit 2, neg.	4	-	grey	->	17
not used	n.c.	-	-	-	9
not used	n.c.	-	-	-	10
not used	5	-	violet	->	11
TD+ CAN+	-	1	blue	->	13
TD- CAN-	-	2	white	->	14
RD+ SGND	-	3	orange	->	15
RD- not used	-	6	red	->	16



Connection cable KeTop TTxxx

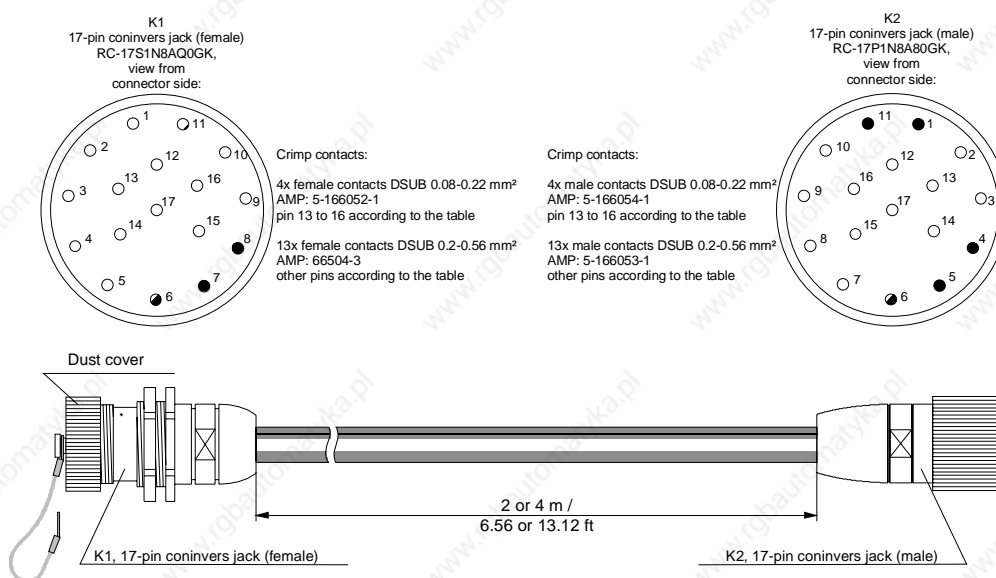
Intermediate Cable KeTop ICxxx

The KeTop intermediate cable is used to connect the connection box and the Coninvers jack in the wall of the control cabinet. Cables with the following lengths are available:

- KeTop IC020 (2 m)
- KeTop IC040 (4 m)

Description of signal	K1, 17-pin female connector, pin No.:	Intermediate cable ICxxx, color of wires		K2, 17-pin male connector, pin No.:
Enabling switch, circuit 1, pos.	7	brown	<-	7
Enabling switch, circuit 1, neg.	8	yellow	<-	8
Enabling switch, circuit 2, pos.	12	green	<-	12
Enabling switch, circuit 2, neg.	17	grey	<-	17
Not used	10	-		10
24 V DC (+/- 10%)	1	red	<-	1
Not used	9	-		9
Emergency stop, circuit 1	3	green/brown	<-	3
Emergency stop, circuit 1 GND	4	white/grey	<-	4
Emergency stop, circuit 2	5	grey/pink	<-	5
Emergency stop, circuit 2 GND	6	red/blue	<-	6
GND_IN	2	black	<-	2
TD+ (transmit)	13	blue	<-	13
TD- (transmit)	14	white	<-	14
RD+ (receive)	15	orange	<-	15
RD- (receive)	16	red	<-	16
Powerfail	11	violet	<-	11

Shielded signals: TD+, TD-, RD+, RD-



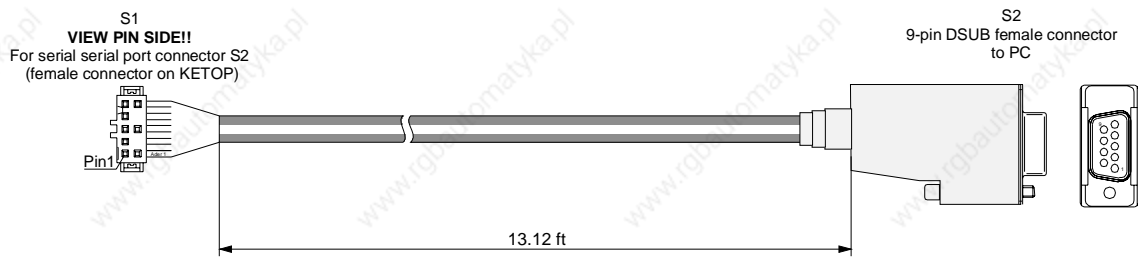
Intermediate cable KeTop ICxxx

Download Cable KeTop XD040

The download cable XD040 is available with a length of 4 m (13.12 ft) and plugged into the serial port connector S2 in the cable entrance area of the KeTop or into the X6 of the KeTop CB23x Gatewaybox. The cable is used for downloading software and for debugging.

Handheld Terminal KeTop xxxx		Gateway box KeTop CB23x		Download cable KeTop XD040			PC
Signals	S2 Serial port connector	Signals	X6 Serial port connector	S1 male connector	Description of signals	S2, 9-pin DSUB female connector	Signals (COMx)
-	1*	n.c.	1	1	< - >	1	(ActiveSync)
-	2*	n.c.	2	2	< - >	6	(ActiveSync)
TXD	3	TXD	3	3	< - >	2	RXD
CTS	4	n.c.	4	4	< - >	-	-
RXD	5	RXD	5	5	< - >	3	TXD
RTS	6	n.c.	6	6**	< - >	-	-
-	7*	n.c.	7	7	< - >	4	(ActiveSync)
n.c.	8	n.c.	8	8**	< - >	9	-
GND	9	GND	9	9	< - >	5	GND
GND	10	n.c.	10	10	< - >	-	not used
						7	not used
						8	not used

*)..... The pins 1, 2 and 7 of the serial port connector S2 are short circuited directly on the CPU board of the KeTop.
 If you produce the serial download cable yourself you will need these three wires in the cable for the ActiveSync signal.
 **) One pin has been removed from the male connector (prevents from incorrect plugging in).



Download cable KeTop XD040

11 Transport Conditions

To avoid damaging the device during further or return transport, the following transport conditions must be fulfilled:

- Always use the original packaging for the transport.
- The environmental conditions for the device (see chapter „Technical Data“) must also be fulfilled during transport.

12 Technical Data

General data

Nominal supply voltage:	24 V DC (Safety extra low voltage)
Supply voltage range:	18 V DC to 32 V DC
Maximum interruption time of supply voltage:	≤ 10 ms (EN 61131)
Power consumption:	7,2 W (400 mA at 18 V DC, 300 mA at 24 V DC)
Inrush current:	max. 5.6 A (with limitation of current)
Safety class:	III in accordance with EN 61131-2 and EN 50178

Environmental conditions

Operating temperature:	0 °C to 50 °C (32 °F to 122 °F)
Storage temperature:	-20 °C to +70 °C (-4 °F to 158 °F)
Relative humidity (non-condensing):	5 % to 95 %
Vibration resistance (operation):	(IEC 60068-2-6) 10 Hz ≤ f < 57 Hz with 0.15 mm (0.0059 in) 57 Hz ≤ f < 150 Hz with 2 g (0.0044 pound)
Shock resistance (operation):	25 g (0.055 pound) / 11 ms (IEC 60068-2-27)

Housing

Construction	Double-walled ABS housing Withstands grease, oil, lubricants, alcohol, etc.
Flammability class	UL 94-V0
Dimensions:	
Diameter:	290 mm / 11.417 in
Total height incl. handle:	130 mm / 5.118 in
Protection degree:	IP54
Weight:	1800 g / 3.968 pound (with handwheel, without Override, without cable) + 100 g / 0.22 pound for PCMCIA extension + 20 g / 0.044 pound for RS-422-A submodul
Display 7.7"	
Type:	Graphics-capable color STN LC display
Size:	7,7" (160 x 120 mm)
Resolution:	VGA 640 x 480 pixels
Representation:	256 colors
Background lighting:	CCFT cold cathode tube (min. lifetime 40,000 hours)
Touch screen:	Analog-resistive
Display 8.4"	
Type:	Graphics-capable color STN LC display
Size:	8,4" (175 x 130 mm)
Resolution:	SVGA 800 x 600 pixels
Representation:	256 colors
Background lighting:	CCFT cold cathode tube (min. lifetime 40,000 hours)
Touch screen:	Analog-resistive
Keypad:	
	- Membrane keypad with tactile feedback
	- Keys laid out for thumb operation
	- Left / right-hand operation
	- Customer-specific keypad possible
	- 2 status LEDs
Operating elements:	Two 3-position enabling switches, twin circuit

Options:	
	Override potentiometer
	Electronic handwheel
	Joystick
	Emergency stop switch
2 locations for:	Illuminated push-button, momentary 0
	Illuminated push-button, momentary I
	Illuminated push-button, maintaining I/O
1 location for:	Illuminated push-button, momentary 0
	Illuminated push-button, momentary I
	Illuminated push-button, maintaining I/O
	Key switch, 3 positions: I-0-II / 0-I-II
	Selector switch, 3 positions: I-0-II / 0-I-II

Processor and interfaces

Processor:	Intel StrongARM SA-1110/206 MHz
Memory:	DRAM: max. 128 MB, FLASH: 64 MB or
Interfaces:	Ethernet
	RS-422-A
	RS-232-C (debug interface in the device)
	PC card slot (PC card type I, II, III)
Operating system:	Windows CE 4.2

Accessories

Wall bracket	For stationary operation or storage of the KeTop.
KeTop WB090:	Wall bracket without cable suspension.
KeTop WB095:	Wall bracket with cable suspension.
KeTop WB100:	Wall bracket with height adjustment and without cable suspension.
KeTop WB110:	Wall bracket with height adjustment and with cable suspension.
Connection cable	Resistant to twisting, bending and foot traffic, with 20-pin push pull connector or 17-pin circular connector
KeTop TT025:	2.5 m / 8.2 ft
KeTop TT050:	5 m / 16.4 ft
KeTop TT100:	10 m / 32.8 ft
KeTop TT150:	15 m / 49.2 ft
KeTop TT200:	20 m / 65.6 ft
Intermediate cable	Connection box to connection cable
KeTop IC220:	2 m / 6.56 ft
KeTop IC240:	4 m / 13.12 ft
Download cable	(not for KeTop T30)
KeTop XD040:	4 m / 13.12 ft. For downloading software and for debugging via S2 (KeTop) or via X6 (KeTop CB23x).
Connection / Gateway boxes	(KeTop CB23x not for KeTop T30)
KeTop CB211:	Connection box for separating the data and control lines.
KeTop CB230:	Gateway box for connecting to CAN via the RS-422-A interface.
KeTop CB232:	Gateway box for connecting to CANopen via the RS-422-A interface.
KeTop CB233:	Gateway box for connecting to DeviceNET via the RS-422-A interface.
KeTop CB234:	Gateway box for connecting to InterBus via the RS-422-A interface.
KeTop CB235:	Gateway box for connecting to PROFIBUS-DP via the RS-422-A.
KeTop CB236:	Gateway box for connecting to PROFIBUS-MPI via RS-422-A.
Visualization software	Only for KeTop T100 and KeTop T50
KeTop PT001:	zenOn 5.50
Start Kit	
KeTop SK001:	Manuals English/German on CD
	SDK for Windows CE
	Programming tool KeTop PS040
	Demoapplications zenOn 5.50
	Demoapplication PLC

Spare parts

Touch styluses	
KeTop E-TP001:	3 original KeTop touch styluses
Service lid	
KeTop E-SD100:	Service lid of cable entrance area with seal and attachment material.
PC card lid	
KeTop E-PC100:	PC card lid with seal and attachment material.

13CE Conformity, Directives and Standards

European Union Directives

It is a fundamental goal of the European Union to create an internal market within Europe and, linked to this, the dismantling of trade barriers.

In order to achieve this goal, the European Treaties guarantee “four freedoms”:

- Free trade in goods
- Right of establishment
- Free exchange of goods and services
- Free movement of capital

Free trade in goods means that quantitative import restrictions on goods are prohibited between the member states.

Goods that are hazardous to the safety of people or the environment are excluded. Such products can be disallowed by member states in their sovereign territory.

In order that free trade may also be guaranteed for these products, the national safety regulations of the member states are harmonised by means of European Union Directives.

These Directives exist for a range of product classes, such as machinery, medical products and toys. However, Directives have also been compiled for further common product safety aspects, such as protection from electricity and explosion, and electromagnetic compatibility.

The Directives are aimed at the member states, whose task is to convert them into national law. The Directives therefore have legal character.

CE marking certifies that the manufacturer fulfils all obligations in relation to the product on the basis of the EC Directives.

The CE mark is the “passport” within the EC and is defined on behalf of the supervisory authorities.

Additionally independent, accredited notified bodies can be commissioned to carry out an EC type-examination and draw up a certificate.

It is not, however, a safety mark or mark of conformity, which can only be awarded by independent testing agencies.

Both the Electromagnetic Compatibility Directive (EMC 89/336/EC) and the Machinery Directive (MD 98/37/EC) are applicable to the handheld terminals.

Machinery Safety

Additional measures must be taken wherever faults arising in the machinery may cause personal injury or significant material damage. These measures must also guarantee a safe operating condition for the whole system in the event of a fault.

Although the handheld terminal is not, strictly speaking, a machine, it does, however, perform important tasks to guarantee the safety functions of a machinery to which it is attached.

The handheld unit has, for example, the "Emergency Stop" safety function and an enabling device for use in special operating modes. It is, as a result, a "Safety Component" in the sense of the Machinery Directive.

Safety components, or parts whose failure or faulty operation put the safety of people within the hazard area of the machine in danger, fall expressly within the range of application of the Machinery Directive.

The fundamental requirements that the Machinery Directive places on the manufacturer are as follows:

- To carry out a hazard and risk analysis
- To comply with the integration of safety principles
- To compile and keep a technical construction file
- To provide solutions in accordance with the latest state of the art
- To recognise conformity by means of harmonised Standards
- To apply CE marking

The same basic requirements apply to safety components. In their particular case, it must be proven that failure or malfunction are not possible, or that malfunction does not lead to a hazardous situation.

"Hazard and Risk Analysis"

The manufacturer of a machine is required to analyse its machine throughout its operating life and in all modes of operation, and to document all hazards that may possibly arise.

The next step is to formulate a goal for protection against each identified hazard and subsequently to define one or more protective measures to achieve the protection goal.

Further details about the procedure for carrying out the hazard and risk analysis and lists of commonly occurring hazards can be found in the following Standards:

- EN 292-1 and EN 292-2 “General Design Principles for Machinery“
- EN 1050 “Guidelines for the Risk Analysis of Machinery“

“Principles for the Integration of Safety”

In Appendix I, Chapter 1.1.2 of the Machinery Directive 98/37/EC there is a clear procedure and sequence for the selection of protective measures:

Eliminating or minimising the hazards

This takes place at the design stage of the machine. These measures include, for example:

- A reduction in the use of energy (power, revolutions, voltages etc.) as far as this is possible
- The avoidance of unnecessary sharp points or edges
- The avoidance of human errors by means of the ergonomic and logical design of operating devices
- The avoidance of hazardous materials and commodities

Taking protective measures against hazards that cannot be eliminated

These measures include, for example:

- Guards, railings, housings
- Protective devices (light barrier for hazard elimination)
- Protective control equipment (enabling devices, two-hand controls, speed monitoring etc.)

User information about residual hazards

This last of the three options is used if residual hazards remain after the first two options have been applied. These measures include, for example:

- Warning notices
- Training and organisational measures
- The use of personal protective equipment

“Technical Construction File”

The technical construction file contains all the documents that are required to prove the safety of the machinery / safety component. These are, for example:

- A complete drawing of the machinery or safety component including control circuit diagrams
- Hazard and risk analysis

- Calculations
- Research and test results
- A list of the basic safety requirements of the Machinery Directive applicable to the machine and a description of the solutions
- Applied Standards
- Operating instructions
- A list of the quality assurance measures in the procedure

The technical construction file must be retained for a minimum of 10 years after the supply of the last product, and must be presented within a period of a few days in the event of a claim for damages.

“State of the Art”

This means technical possibilities at a certain point of time that are based on certain scientific and technical knowledge. The state of the art also means something that is commercially viable, that is it can be realised by the majority in the industrial sector concerned.

The state of the art is defined as the state of development of advanced procedures, equipment or operating methods, that makes the practical applicability of the measure appear assured overall in respect of the targeted goals (e.g. the goals of protection of work, protection of the environment, safety of third parties and operating efficiency: namely to achieve a generally high level overall in relation to the aspects under consideration).

The state of the art can develop further irrespective of the Standards.

“Recognition of Conformity by means of Harmonised Standards”

The European Directives mainly contain general requirements for the safety of products; however they do not contain details of how to carry them out.

The European Standards Institutes are responsible for this. They provide implementation proposals for real safety problems or specific product classes. Standards that are assumed to meet and correctly interpret the requirements of the Directives are known as “Harmonised Standards”. Most of the available Standards, however, are not harmonised.

By applying and implementing harmonised Standards, a manufacturer can claim conformity for the respective product. However, the Standards, in contrast to the Directives, are not legally binding. This means that the manufacturer may also take into consideration other solutions that are not described in the Standards; but these solutions must attain at least the same safety level as the relevant Standards and satisfy the requirements of the appropriate Directives.

Selection of Safety Categories in accordance with EN 954-1

The Machinery Directive demands that a fault in control circuit logic, or interference or damage thereto, shall not lead to a hazardous situation.

This general approach is substantiated in EN 954-1 "Safety of Machinery, Safety Related Parts of Control Systems", which defines safety categories (B, 1, 2, 3, 4) for control parts that are relevant to safety.

These categories apply irrespective of the technology employed, for example to electrical, electro-mechanical or pneumatical systems.

The categories place qualitative requirements on the probability of failure, the detection of faults and the performance of the controller in the event of a fault.

The manufacturer of the machine selects the category dependent on the actual hazard potential, which is determined from the hazard and risk analysis.

For hazards that can cause irreversible injury or death, safety category 3 or 4 is usually required as a minimum. These categories require single fault safety, which usually applies to multiple circuit technology in conventional manner. It is also important in this context that individual faults are detected in time to avoid a build up of faults, which may finally lead to loss of safety. Faults that must be detected on electrical and electronic systems, for example, are: cross circuits, cut-outs and sticking contacts.

Special certified safety control devices are often used to detect faults in the individual safety circuits. The safety category quoted for these devices is only attained, however, if also the whole machine circuit under review lies within the scope of the respective safety category. The safety categories must always be considered in relation to a complete safety function and not as applied to individual components or parts.

The proof of attainment of a safety category can take place with the aid of an FMEA (Failure Mode and Effects Analysis), in which all the faults that could possibly arise are simulated, either theoretically or in practice, and it is demonstrated that the requirements of the category are fulfilled.

Application of Handheld Terminals in Special Operating Modes

For the manual control of machines in special operating modes, where safety depends on the timely reaction of the operating staff, it is absolutely essential that the operator can overlook the operating area.

The handheld terminal has the advantage that the operator can get very close to the control panel.

At the same time, the danger of misuse increases with mobility since, in remote locations where it is not possible to observe the operating area,

machine movements can also be set in motion with the handheld terminal, knowingly or unknowingly.

The machine operator, therefore, has to find the right compromise between necessary flexibility and a reasonable limitation of the working range when selecting the corresponding cable length for the handheld terminal's.

It is not possible for the working range of radio-operated handheld terminal's to be limited by means of the cable; therefore additional technical solutions are required for these handheld terminal's.

If the machine or equipment is operated with the handheld terminal, care must be taken at this time to ensure that operation can only be controlled by the handheld terminal and cannot be operated from any other point on the equipment.

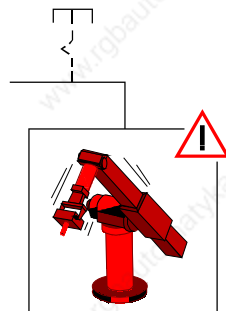
The hazard zone may only be entered by the person who is operating the handheld terminal. If it should be necessary for more than one person to work in the hazard zone at the same time, each person present requires an enabling device and machinery movement may only be allowed after all the enabling devices have been activated.

Information about the Emergency Stop Button

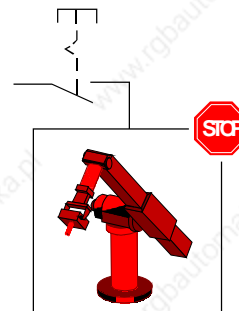
In theory, a perfectly designed machine should not require an emergency stop button, since the Machinery Directive demands that a machine must be safe throughout its lifetime and in all operating modes.

In practice, however, one is aware that unforeseen situations arise, in spite of all precautions. In order to enable fast shutdown of the machine in these cases, or to avert danger, an emergency stop button is provided on most machines.

As can be seen from the following illustration, a machine is permanently in the hazard state from the viewpoint of the emergency stop button, for as long as this is not activated.



Neutral position (not activated)



Emergency stop button pressed

Emergency stop button functions

The emergency stop button may not, therefore, serve as the primary safety device but is provided exclusively to cover any residual risks.

Instead of this and depending on the mode of operation, other methods should be employed as the primary safety device, for example guards, light barriers and two-hand control devices or enabling switches.

When the emergency stop button is activated, the entire machine or all of the machines that are switched together in a plant must be lead to a safe state e.g. by stopping and switching off the power to the endangering drive mechanisms (stop categories 0 or 1 in accordance with EN 60204-1).

The release of the emergency stop button must not cause any uncontrolled restart of the machine.

Irrespective of whether a handheld terminal has an emergency stop button or not, permanently installed, universally identifiable and easily accessible emergency stop buttons must be provided at selected points around the machine in every case.

Application of Emergency Stop Button and Stop Button on the Handheld Terminal

The Machinery Directive does not specify any detailed requirements in respect to the colouring of the operating elements for emergency stop. The requirement states: "The device must have clearly identifiable, clearly visible and quickly accessible controls".

Emergency stop buttons are usually identified by red/yellow. The specific design should have a signalling function whose effect is that any person, and especially untrained people, can quickly recognise the control as the device to eliminate a hazard in the case of an emergency.

One imperative requirement for equipment identified in this way is, therefore, that it must be possible to trigger a safe state at any time and in any operating mode, without further knowledge of the machine (see also EN 418).

It is not acceptable for an emergency stop button to be enabled temporarily, since this can lead to faulty operation and life threatening loss of time in panic situations.

If one considers handheld terminals in relation to these requirements it is evident that a differentiation must be made as to whether an emergency stop button on the handheld terminal may be used or not, depending on specific criteria and applications.

Permanently installed handheld terminals: Emergency stop button

Permanently installed handheld terminals are provided with a cable by means of which the handheld terminal is connected to the designated machine. The handheld terminal is usually connected to or disconnected from

the machine with the machine switched off. This takes place during the installation or deinstallation process. The handheld terminal's are not designed to be connected or disconnected whilst the machine is operating.

On many simple machines the handheld terminal is also the only operating option, so that, without the handheld terminal, the machine cannot be put into operation at all. Nonetheless, if the connector is unplugged from the machine during operation the emergency stop circuit is broken and the emergency stop function of the machine is activated, causing the machine to stop.

If a handheld terminal is uninstalled from a machine and is not reinstalled immediately, the equipment must be locked out in order to avoid confusion with a functioning handheld terminal. The machine can only be put back into operation again after a new handheld terminal has been installed.

This procedure must be described in the operating instructions for the machine and is the responsibility of the operator.

Due to the fact that plugging and unplugging takes place only rarely and that the machine is out of operation when unplugged, the risk of an accident as the result of a non-operational emergency stop button is considered to be very low and the use of red/yellow identification is therefore permissible.

The red/yellow emergency stop button must be connected into the emergency stop circuit of the machine in any case and must cause the power supply to the machine or set of machines in a plant to be switched off (stop category 0 or 1 in accordance with EN 60204-1).

Radio controlled handheld terminals: Stop button

Wireless handheld terminal's present a different case. These devices are not normally assigned to a specific machine but can be frequently logged on and off whilst the machine is in operation and can also be switched between different machines. For this reason the stop button is not always operational and the operating status is not clear to everyone.

Therefore, for stop functions in wireless controllers, EN 60204-1:1997 "Safety of Machinery. Electrical Equipment of Machines" (Chapter 9.2.7.3), demands:

"The operating means to initiate this stop function shall not be marked or labelled as an emergency stop device, even though the stop function initiated on the machine can be an emergency stop function."

For this reason, for radio controlled handheld terminal's KEBA uses a stop button that features all the mechanical characteristics of a normal emergency stop button but is of a neutral grey colour.

In contrast to the red/yellow emergency stop button there is no need to switch the stop output signals of the radio receiver into the machine's normal emergency stop circuit. It can also be used to stop individual safety zones of a machine or plant, where functions such as "Safe operational stop" can be triggered. This means that the drive components are safely maintained in controlled shutdown by means of active, powered drives (stop category 2 in accordance with EN 60204-1). This can prevent the loss of reference data and facilitates fast restart of the machines.

The trained handheld terminal or machine operator is aware of the function assigned to the button and knows the current operating mode in each case as well as the current assignment of the machine. For this reason the colouring is not disadvantageous to the machine operator.

The safety gain is that, in the case of inactive or unassigned equipment, no danger of confusion is presented to third parties in respect of functional emergency stop buttons.

Temporary plug-in handheld terminals: Stop button

Some cabled equipment is provided with a quick-release connector (e.g. bayonet) that makes it possible to plug in and unplug the handheld terminal quickly and ergonomically whilst the machine is in operation. Such handheld terminal's are provided for frequent alternation of operation between one or more machines and are needed there on a temporary basis, for commissioning or setting up purposes.

By means of several measures, such as bridging connectors or relay circuits, the emergency stop circuits of the machine are bridged when the handheld terminal is unplugged so that the machine can also continue to run in normal operation without the handheld terminal. Start-up and operation of the machine can then take place using an independent operating device.

These handheld terminals share the same problems as the radio terminals, since due to frequent plugging and unplugging it cannot be ruled out that disconnected handheld terminal's with ineffective stop buttons may be left lying or hanging temporarily in a machine shop or factory in the neighbourhood of working machines and may be mistaken for functional units in an emergency situation.

This special case is not clearly treated in the corresponding Standards. However, in the Product Standards for presses (EN 692:2004, EN 693:2001, EN 12622:2001, and EN 13736:2003) one finds the following clear statement:

"Any disconnectable control station shall not incorporate an emergency stop button if the press can be operated while this controls station is disconnected."

Further Product Standards are currently under review.

In several discussions with external notified bodies and technical committees it was also determined that temporary, plug-in handheld terminals should be treated as radio handheld terminal's.

The approach to constructively eliminate hazards by using clear colour coding, in preference to any organisational measures, also corresponds to the "Principles for the Integration of Safety" of the Machinery Directive and is therefore legally binding.

For this reason, temporary plug-in handheld terminals may likewise only be equipped with a grey stop button.

Since handheld terminals made by several other manufacturers are on the market with a grey stop button and have been certified by nominated test centres, the state of the art requirement is also met.

Information about Enabling Devices

Many machines have both a normal and special operating modes.

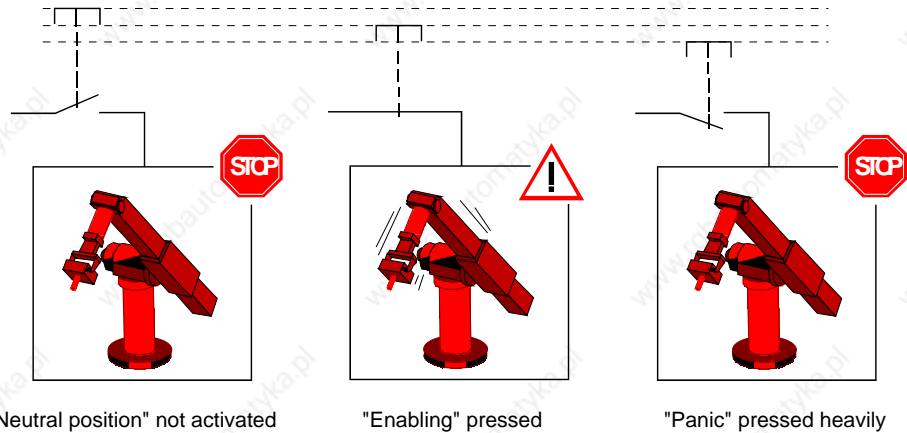
The machine fulfils its primary mission in normal (automatic) operation. In this mode of operation, safety is guaranteed by closed, guards and/or with protective devices.

A machine's special operating modes serve to maintain normal operation. In doing so, safety must be guaranteed in a way that is different to normal operation, since hazardous areas of the machine can now be entered and targeted machine movements have to be possible.

The enabling device plays a primary role here.

This application requires the enabling device to eliminate the hazards that are to be controlled in a timely manner, before any personal injury occurs. Additional safety measures may be necessary for this purpose, such as safely reducing the speed of drive mechanisms.

In contrast to the emergency stop button respectively the stop button, a machine in special operating mode with an enabling button is in a safe state as long as this enabling button is not activated (see illustration).



Enabling functions

The enabling function is also described in EN 60204-1:1997 and corresponds to the state of the art.

The panic position on 3 stage enabling switches was introduced because limbs often become cramped when a person is in a state of shock and as a result is not able to release the enabling switch. For this reason, firmly pushing down on the enabling switch also leads to switch off.

The controller must be designed in such a way that machine movement is not initiated directly when the enabling device is activated but only as a result of the additional activation of a control button. This may be via the handheld terminal's membrane keyboard or graphical software buttons on the touch screen.

An optionally available joystick can also be used to trigger the machine movement signal.

According to EN 60204, only stop categories 0 or 1 are allowed for the enabling function, i.e. stopping the machine with the enabling device must always be combined with switching off the power supply to the drive mechanisms.

In order to prevent incorrect, permanent activation of the enabling switch by mechanical fixing devices, it is recommended to limit the maximum duration of an accepted enablement. This must be achieved by means of a controller located downstream to the handheld terminal.

An enabling device is not a replacement for a two-hand control device, which is specified for some machines (e.g. presses) and must therefore not be confused.

In contrast to the enabling device, the two-hand control device forces the endangered limbs of the operator into a safe position whilst the machine is in motion.

Electromagnetic Compatibility

The European Union obligates its member states to harmonise their statutory provisions in respect of electromagnetic compatibility by means of Directive 89/336/EC (will be replaced by the new Directive 2004/108/EC from 20.7.2007). In the following text this Directive is abbreviated to the EMC Directive.

All electrical and electronic device that is put into circulation in Europe must, therefore, conform to the fundamental safety requirements of the EMC Directive. According to this Directive, electromagnetic compatibility is the ability of a device, plant or system to work satisfactorily in the electromagnetic environment, without itself causing electromagnetic interference that would be unacceptable to all devices, plant and systems present in this environment.

In addition to the legal requirements described above, the reliable functioning of an electrical device is also a fundamental quality characteristic of such a device.

As well as basic information concerning electromagnetic compatibility, the following pages describe the implementation of the EMC requirements in respect of the KeTop product range of handheld terminal devices.

The electromagnetic environment – interference sources, interference sinks and coupling paths

There are a number of artificial and also natural sources of interference in the electromagnetic environment that can affect electrical and electronic device.

The most well-known natural interference phenomenon is atmospheric discharge (lightning discharge).

Artificial sources of interference are, on the one hand, intentional, such as mobile radio, amateur radio, TV and radio transmitters, and, on the other hand, unintentional, such as microwave ovens, arc welding, ignition systems, high voltage device, electric motors, electronic device and also electrostatic discharge.

Atmospheric discharge (lightning)

Atmospheric discharge can take place directly or indirectly on power and communication lines. The consequence of such an impact if there is insufficient protection is the destruction of the electrical device.

Atmospheric discharge is simulated in accordance with the international test standard IEC 61000-4-5 (EN 61000-4-5 for Europe) – *Electromagnetic*

compatibility (EMC). Testing and measurement techniques. Surge immunity test.

The relevant international product standard for control systems, IEC61131-2 (EN61131-2 for Europe), divides the areas of application into zones. Higher or lower levels of interference can be expected depending on the zone in question. All Kemro-K2 control systems and also the products of the KeTop range are suitable for application in Zone B.

The product standard requires the testing of surge immunity in accordance with IEC 61000-4-5.

Electrostatic discharge (ESD)

Materials can be charged by contact followed by subsequent separation. This effect only arises when at least one of the two materials is a non-conductor. As a result, discharge may take place if a charged conductor, or one that is charged by the influence of an electronic field, comes near a metallic object.

After charging, the possible voltages between the charged bodies can reach over 10 kV.

The most frequent occurrence of electrostatic discharge takes place between people and metallic objects. Since one is practically unnoticed of discharges below 3500 V and yet electronic components are destroyed by low voltages, ESD damage to electronic components often goes unnoticed.

The international Standard IEC 61000-4-2 (EN 61000-4-2 for Europe) is used to simulate the measurement of electrostatic discharge. The international product standard IEC 61131-2 (EN 61131-2 in Europe) for programmable controls demands testing to IEC 61000-4-2 and also defines the severity level.

Technical systems as interference sources

Technical systems can act as sources of interference. In doing so, the interference may be intentional or unintentional. Electromechanical energy is often also used for material processing.

Periodically occurring interference:

- Ignition impulses of combustion engines
- Sparking of commutator motors
- Electromagnetic fields of induction furnaces, arc welding device, microwave device etc.
- Pulse currents from frequency converters and switching power supplies
- Electromagnetic fields of radio and telecommunications device

Randomly occurring interference:

- Ignition impulses of fluorescent lamps
- Switching procedures on inductive electrical circuits

- Contact bounces when closing or opening make-and-break contacts
- Voltage fluctuations on heavy load switching procedures

There is a series of test standards for the above listed sources of interference, intentionally or unintentionally caused by technical systems, which simulate this interference:

- IEC 61000-4-3 High Frequency Electromagnetic Field Immunity Test
- IEC 61000-4-4 Electrical Fast Transient / Burst Immunity Test
- IEC 61000-4-6 Test of Immunity to Conducted Disturbances, Induced by High Frequency Fields
- IEC 61000-4-8 Power Frequency Magnetic Field Immunity Test
- IEC 61000-4-11 Voltage Dips, Short Interruptions and Voltage Variations Immunity Test

All the international Standards listed here are also available as European Standards. The Product Standard IEC 61131-2 demands testing to these Standards and also defines the severity level.

Technical systems as interference sinks

In the case of functional interference, EMC problems first arise at interference sinks. The following interference sinks can be identified dependent on the degree of immunity to electromagnetic influences:

Immunity	Interference sinks
max	Transformers
	Circuit breakers, contactors
	Relays
	Power transistors
	Transistor circuits
min	Integrated switching circuits

Control systems without integrated switching circuits are unthinkable and would therefore not be sufficiently immune without suitable EMC measures.

Coupling paths

The transfer of interference signals from an interference source to an interference sink can take place via various coupling paths.

Coupling paths very often consist of two or more parallel lines running closely together. The coupling is a field coupling that takes place at low fre-

quencies either via the electrical field (capacitive coupling) or via the magnetic field (inductive coupling).

At high frequencies and with corresponding expansion of the parallel running lines one speaks about an electromagnetic coupling due to close linking of both field types.

Direct coupling can occur when electrical circuits from the interference source and electrical circuits from the interference sink have common line components.

If there is a large distance between an interference source and an interference sink then one speaks of a radiation coupling.

EMC measures

In principle, all devices should be designed in such a way as to function reliably in the planned environment and in doing so should not interfere with other systems. All products of the Kemro product range (K2 control systems and KeTop) meet these requirements and it is not necessary to use any of the EMC measures described below when the specified additional units, cables and wiring are used.

However, additional EMC measures may be necessary for various reasons.

The following text is designed to help the user to correctly implement any additional EMC measures.

Shielding

Electromagnetic emission problems and immunity problems often occur in products at the same time. Likewise, EMC measures are mostly effective in the case of both emission problems and immunity problems.

Shielding fulfils two main tasks. On the one hand, the penetration of electromagnetic fields into sensitive electronic parts is prevented and, on the other hand, the radiation of electromagnetic fields is also prevented.

A complete EMC shield consists of a shielded housing, which protects the sensitive electronics and prevents them from transmitting interference, and cable shields that shield the sensitive interface signals and prevent the device from transmitting interference via its interfaces.

The cable shield basically connects two shielded housings together and must therefore be connected directly to the shielded housings of the sensitive electronics (connection points).

Particular attention should be paid to the connection between the cable shield and the shielded housings of the device. In order that the shield effect is maintained even at higher frequencies, care must be taken to ensure that the cable shield connects to the largest possible surface area and

thereby acts as a continuation of the device's shielded housing. Pig tail connections are not suitable.

It may be necessary to earth cable shields and shielded housings for safety reasons but this is not an effective EMC measure.

Interference suppressors, filter components

Filtering is always necessary when unshielded signal and power supply lines are brought into shielded areas. Alongside the wanted signals, these lines often also carry interference signals that must not enter into shielded areas. Filters should therefore guarantee the interference immunity of the device but should also prevent the emission of interference from the device via unshielded lines.

Unshielded lines are usually used when the wanted signals that are carried are of a very low frequency. The normally high frequency interference signals are separated from the wanted signals by means of frequency selective filtering with the aid of low-pass filtering.

Low-pass filters must be sized in such a way that the lower frequency wanted signals can pass through and the higher frequency interference signals are filtered.

Multi-stage filters are often necessary for filtering. Nearly all filter combinations contain Y-capacitors, i.e. capacitors that are connected to the filter housing for the dissipation of interference currents. In order for these filters to function correctly, the housing must be connected to a stable reference potential.

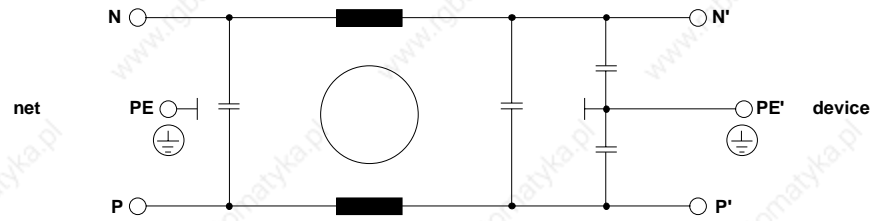
Since the interference signals must not enter into the shielded area, the potential of the shielded housing must also be the reference potential for the filter circuits.

Filter circuits or filter components must therefore be placed precisely where the lines enter the shielded area. If the filters are placed inside or outside the shield wall, this can lead to a field coupling between the filtered and unfiltered lines.

The connection to the reference potential must be as low impedance as possible and therefore it is necessary for there to be contact over a large surface area between the filter housing, which is usually metallic, and the shield wall of the device. Pig tails may not be used to make such a contact.

Line filters

Typical line filters have a metal housing that has to be connected to the earth wire (PE) for safety reasons and consist of a current-compensated choke (see illustration) consisting of Y-capacitors, which are connected to the metal housing on one side, and X-capacitors (connected between the phases or between phase and neutral).

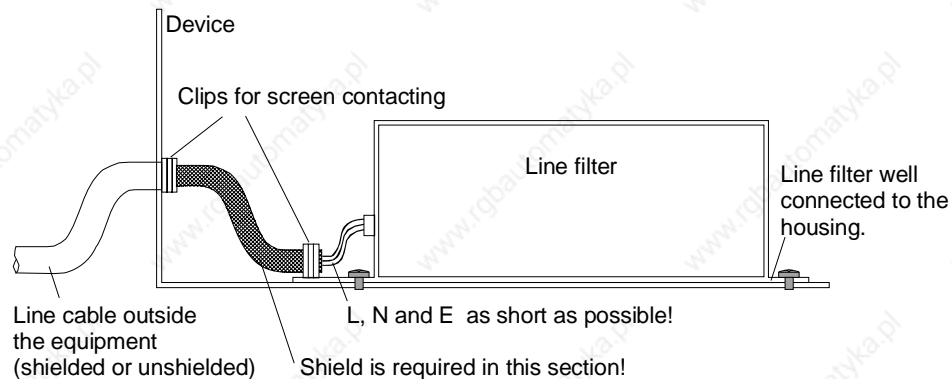


Line filter

The desired filter effect is basically governed by the correct contact between the filter housing and the shielded housing of the device. The line filters must therefore be placed directly at the inlet to the shielded housing of the device and a very good connection must be made between the metal housing of the filter and the shielded housing of the device.

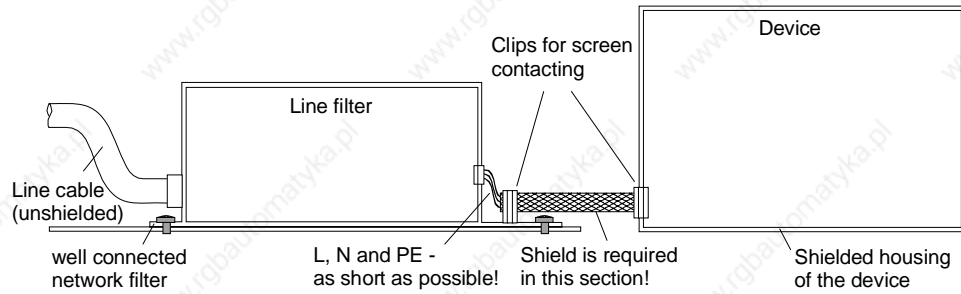
If placement at the inlet is not possible for reasons of space, partly shielded connection lines must be used. In doing so the following procedure is to be followed:

If the filter is placed inside the shielded housing of the device, the line cable must be shielded from the inlet through to the line filter and the cable shield contacted on both ends. This prevents the propagation of interference on the line cable inside the device. It also prevents any electromagnetic fields that may be present inside the device from penetrating the line cable.



Correct connection of a compact line filter

If the line filter is placed outside the shielded housing, the device connection line must be shielded from the line filter through to the shielded housing of the device and the cable shield must be contacted on both ends. This prevents interference produced by the device from being transmitted externally where it could penetrate other systems. It also prevents the penetration of interference signals back into connection wires that have already been filtered.



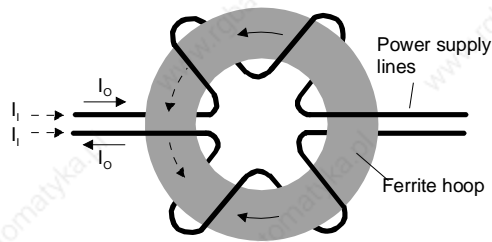
Correct connection of a compact line filter

When connecting line filters, attention should first be paid to the frequency ranges to be filtered. If frequencies in the MHz range are also to be filtered, such a line filter should have a metal housing. Line filters with plastic housings are normally not suitable for this purpose.

Current-compensating chokes

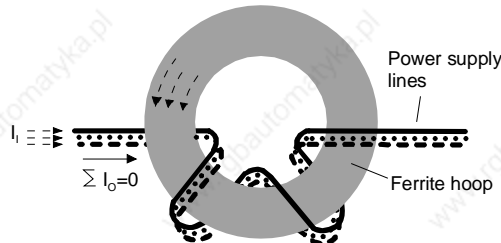
Device manufacturers sometimes specify the use of current-compensating chokes or the installation of ferrite loops in the power supply line. Many examples of this can be found in motor lines that are fed by frequency inverters.

Such chokes (see the following illustrations) suppress common-mode interference currents (I_s) and allow the operating currents (I_B) to pass unhindered. It should be noted that the operating currents in the core produce self-cancelling magnetic fields so that the operating current does not saturate the ferrite cores. Such throttles must also be placed at the inlet to the shielded housing of the device.



I_i ... Interference current
 I_o ... Operating current

Current flow of a current-compensating choke

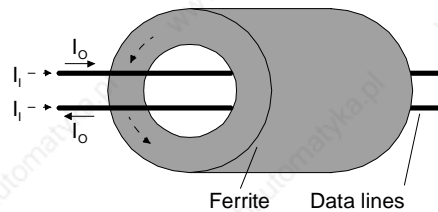


I_i ... Interference current
 I_o ... Operating current

Current flow of a current-compensating choke

Ferrites

Ferrites operating via data lines work in a similar way to current-compensating chokes. The wanted signals (see illustration) can pass unhindered and the interference signals are damped. A fundamental difference from current-compensating chokes is that in the case of ferrites the application range takes place at higher frequencies and therefore the material losses are responsible for the damping and the interference is not reflected, as in the case of inductance, but is converted into heat.



I_i ... Interference current
 I_o ... Operating current

Current flow for ferrites

Prevention of couplings

Coupling between interference sensitive signals (e.g. analogue signals) and the lines that carry the interference signals (e.g. motor lines) can be reduced by laying cables separately.

EMC measures on KeTop

The KeTop is designed for industrial applications. Considerable electromagnetic and electrostatic interference can occur in this environment. For this reason special emphasis was placed on interference immunity and data security in the design of the device. The EMC design implemented and described below considers all the above named sources of interference and the possible coupling paths.

- All shielding and filter measures (filtered power supply) in the KeTop are carried out directly on the printed circuit board.
- The special design of the KeTop cable guarantees interference immunity even for greater connection lengths, i.e. the data lines (communication signals) are shielded and so they are separated from the control lines (power supply, enabling button, emergency off, key-operated switch etc.) inside the KeTop cable.
- Power supply lines in the control cabinet are usually unshielded. They are therefore laid outside the KeTop cable shield in order to prevent a coupling with the sensitive data lines.
- The unshielded control and power supply lines are either filtered on entry into the shielded housing of the KeTop or separated from the electronics in such a way that any interference signals carried by these lines cannot affect the internal electronics of the KeTop.
- It is not necessary to lay the KeTop cable separately.

Shield connections

The cable shield of the KeTop cable can be considered to be an extension of the KeTop shielded housing (=printed circuit board) to the shielded housing of the communication partner (e.g.: PLC). From this it can be deduced that the shielded connections from the cable shield to the device shields make a fundamental contribution to the interference immunity of the KeTop. It is not necessary to wire shielded connections to earth.

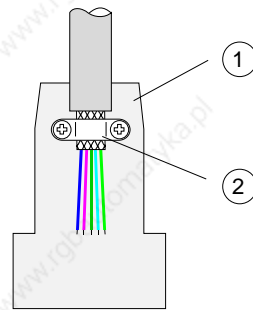
The shielding connection in the KeTop is made via the RJ-45 connector in the connection slot.

Connecting the KeTop via the connection box (e.g. CB211) guarantees safe shielding protection right through to the communication partner. It is therefore urgently recommended to use a suitable KEBA connection box.

All connection cables available from KEBA (KeTop TTxxx, KeTop ICxxx, KeTop XD040 and KeTop CD040) guarantee correct shielding connection.

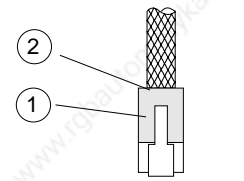
For self-made cables the following conditions must be met for shielding data lines:

- On every connector design (DSUB or RJ-45) the cable shield must be connected to as large a surface area of the connector housing as possible.



DSUB connector

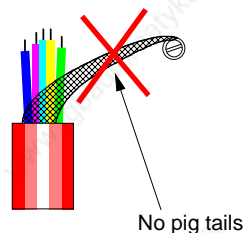
- 1..... Metallised or metallic housing
- 2..... Connect the shield to a large surface area



RJ-45 connector

Correct shield connection for DSUB and RJ-45 connectors

- The use of pig tails for contacting the cable shield via plug pins is unsuitable. The inductivity of such pig tails represents high impedance resistance for higher frequency interference; this means an apparent break in the cable shield. Interference is then no longer diverted and works directly on the inner lines.

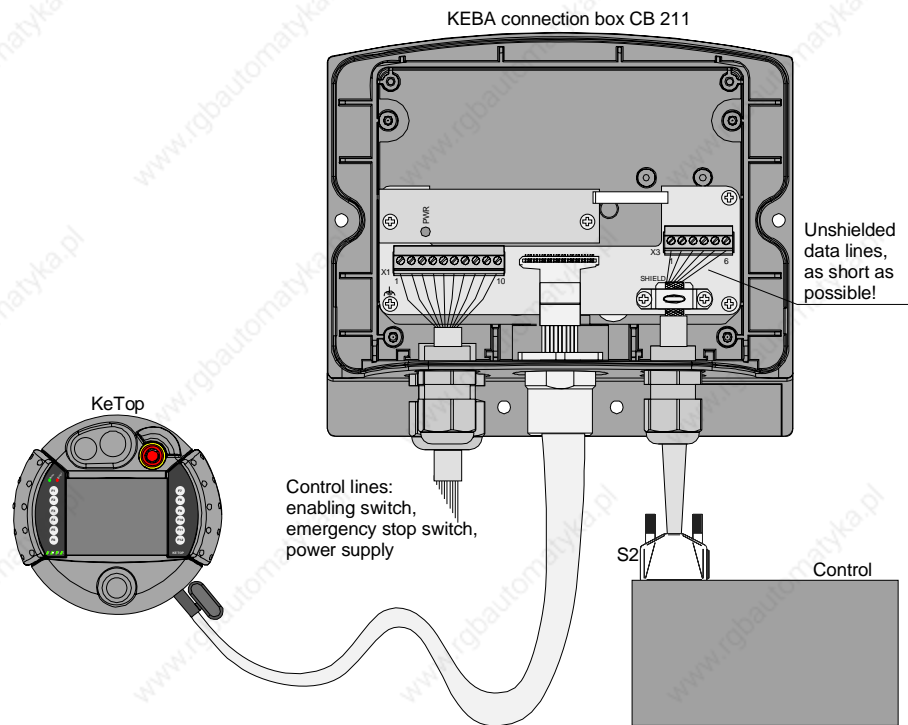


Insufficient contacting of a cable shield

Shielding inside the control cabinet

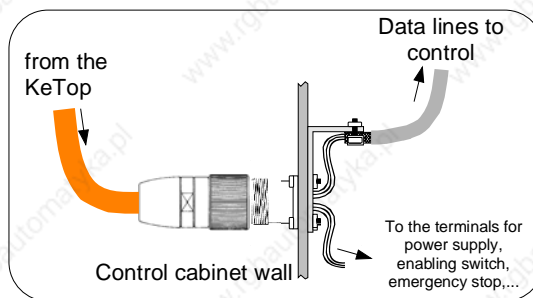
In many cases a range of interference sources, such as servo drive modules, transformers, contactors and relays, is present in the control cabinet. It is therefore necessary to run the cable shield from the connector housing (control cabinet) to the control (continuous connection from the handheld terminal to the control).

When using the appropriate connection box and a shielded cable for the data line from the connection box to the control, the continuous, high frequency compatible connection of the shield from the KeTop to the control is guaranteed.



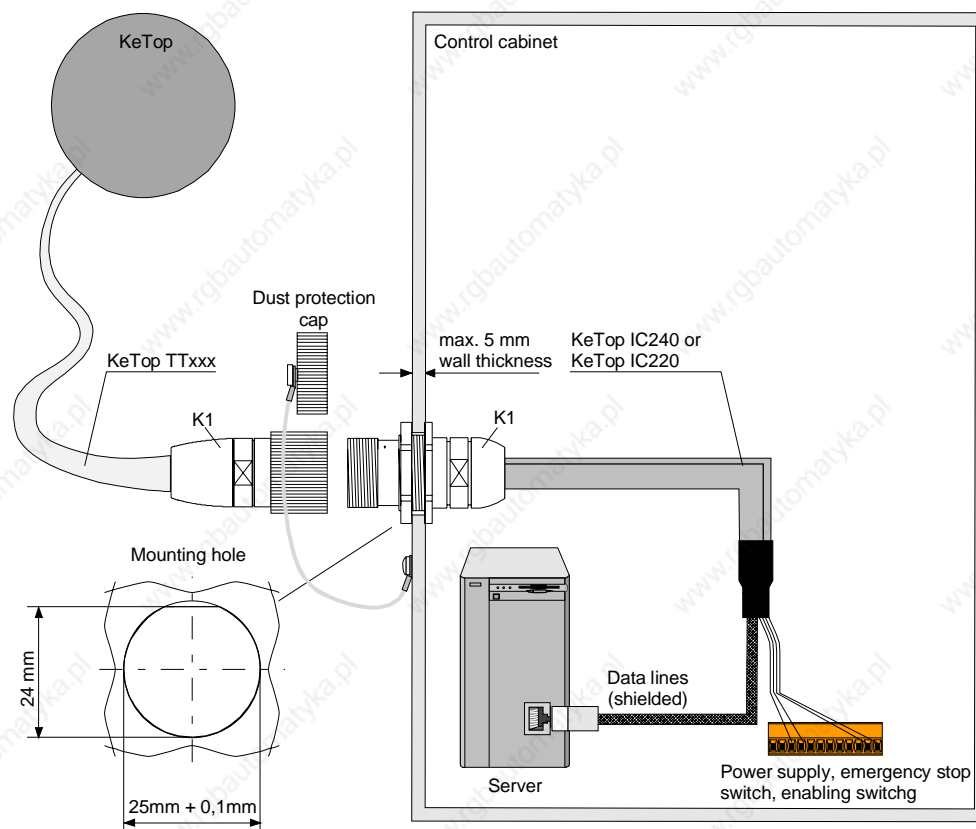
Example of a correct shield connection in a KEBA connection box

If for some reason a connection box cannot be used, the continuous connection of the shielding on less critical interface types such as RS-232-C takes place by contacting the connector housing with the control cabinet and inside the control cabinet by contacting the shielding with the control cabinet (using shielding clips). The smaller the distance between the contacting of the connector housing on the control cabinet to the cable clip in the control cabinet, the better the shield damping.



The interference immunity of the device with the connection type presented above is also decisively influenced by the satisfactory separation of the control signals and the data line signals. The better the separation of the two signal types and the shorter the shielding connection, the higher will be the interference immunity of the complete system.

If the Ethernet is used as the communications interface (KeTop T100 only) one of the two connection cables provided for the purpose of connecting to a control system must be used (IC020 or IC040). Both cables carry the Ethernet signals to a suitable connector (RJ-45) and thereby produce a continuous shielding connection and also the required characteristic impedance of the cable.



Connection cable IC020 or IC040 in the control cabinet

List of the appropriate EC directives and applied standards

EC directives

98/37/EC 89/336/EC	Directive for the safety of machinery with the application 98/79/EC EMC directives with the changes 91/263/EWG and 93/31/EWG
-----------------------	---

Standards

For examination the conformity of the KeTop regarding the directives the following legally not bounded european standards has been used:

Examination of the conformity regarding the directive of machinery

EN 418:1992	Safety of machinery - emergency stop equipment, functional aspects, principles for design
EN 954-1:1996	Safety of machinery - Safety-related parts of control systems - Part 1: General principles for design
EN 60204-1:1997, chap.9, 10	Electrical equipment of machines, general requirements

Examination of the conformity regarding the directive of EMC

EN 61131-2:2003 chap. 7, 8	Programmable Controllers Part 2: Equipment requirements and test
-------------------------------	---

So the accordance to following standards is also given:

EN 61000-6-2:2001	Electromagnetic compatibility (EMC): Generic standards - Emission standard for industrial environments
EN 61000-6-4:2001	Electromagnetic compatibility (EMC): Generic standards - Immunity for industrial environments

Other standards

For the design of the safety concept some parts of the following legally not bounded european standards has been used.

General procedure and safety principles

EN ISO 12100-1:2003	Safety of machinery - Basic concepts, general principles for design - Basic terminology, methodology
EN ISO 12100-2:2003	Safety of machinery - Basic concepts, general principles for design - Part 2: Technical principles

Enabling switch

EN 954-1:1996 (ISO 13849-1:1999)	Safety of machinery - Safety-related parts of control systems - Part 1: General principles for design
EN 60204-1:1997	Safety of machinery - Electrical equipment of machines - Part 1: General requirements
ISO 10218-1:2006	Manipulating Industrial Robots - Safety

Stop switch and emergency stop switch

EN 418:1992	Safety of machinery; emergency stop equipment, functional aspects; principles for design
EN 60204-1:1997 chap. 9, 10	Safety of machinery - Electrical equipment of machines - Part 1: General requirements

Ergonomics

EN 614-1:1995	Safety of machinery - Ergonomic design principles - Part 1: Terminology and general principles
EN 894-1:1997	Safety of machinery - Ergonomics requirements for the design of displays and control actuators-Part 1: General principles for human interactions with displays
EN 894-2:1997	Safety of machinery - Ergonomics requirements for the design of displays and control actuators - Part 2: Displays
EN 894-3:2000	Safety of machinery - Ergonomics requirements for the design of displays and control actuators - Part 3: Control actuators

Stability and impermeability of casing

EN 60529:1991	Protection degree of casing
EN 61131-2:2003 chap. 12	Programmable controllers - Part 2: Equipment requirements and tests

Electrical safeness and fire protection

EN 61131-2:2003 chap. 11	Programmable controllers - Part 2: Equipment requirements and tests
EN 50178:1997	Electronic equipment for use in power installations

Environmental Conditions

EN 61131-2:2003 chap. 4	Programmable controllers - Part 2: Equipment requirements and tests
EN 50178:1997	Electronic equipment for use in power installations

UL examination for robotic applications

UL 1740, 1998	Industrial Robots and Robotic Equipment (TETZ2, TETZ8)
---------------	--

UL examination for industrial control equipment

UL 508, 17 th edition	Industrial Control Equipment (NRAQ, NRAQ7)
----------------------------------	--