Edition: 11/2024

a-Series basic compiler

Programming Manual

ABC

Made in Germany

ABC Programming Manual
for the following products

Family	Type
A	A3-2, A3, A4, A6, A8/300
A+	A2+, A4+, A4.3+, A6+, A8+
AXON	AXON 1, AXON 2
EOS	EOS1, EOS4
EOS2	EOS2, EOS5
HA	Hermes A2, Hermes A4, Hermes A5
HC	Hermes C6
HQ	HERMES Q2, HERMES Q4, HERMES Q4.3, HERMES Q6
H+	Hermes+2, Hermes+4, Hermes+ 4.3, Hermes+6
MACH 4	MACH 4
MACH 4S	MACH 4S, MACH 4.3S
PX	PX4, PX 4.3, PX6
PXQ	PX Q4, PX Q4.3, PX Q6
SQUIX	SQUIX 2, SQUIX 4, SQUIX 4.3, SQUIX 6, SQUIX 8
XC	XC4, XC6
XCQ	XC Q4, XC Q6
XD	XD4M, XD4T
XDQ	XD Q4, XD Q4.2

Edition: 11/2024 **Firmware version:** 5.45.2

Copyright

This documentation as well as translation hereof are property of cab Produkttechnik GmbH & Co. KG.

The replication, conversion, duplication or divulgement of the whole manual or parts of it for other intentions than its original intended purpose demand the previous written authorization by cab.

Trademark

Microsoft® is a registered trademark of the Microsoft Corporation.

Windows® is a registered trademark of the Microsoft Corporation.

TrueType™ is a registered trademark of the Apple Computer, Inc.

Editor

Regarding questions or comments please contact cab Produkttechnik GmbH & Co. KG.

Terms and conditions

Deliveries and performances are effected under the General conditions of sale of cab.

Germany
cab Produkttechnik GmbH & Co KG
Karlsruhe
Phone +49 721 6626 0
www.cab.de

USA
cab Technology, Inc.
Chelmsford, MA
Phone +1 978 250 8321
www.cab.de/us

Taiwan
cab Technology Co., Ltd.
Taipei
Phone +886 (02) 8227 3966
www.cab.de/tw

Singapore
cab Singapore Pte. Ltd.
Singapore
Phone +65 6931 9099
www.cab.de/en

France
cab technologies s.à.r.l.
Niedermodern
Phone +33 388 722 501
www.cab.fr

Mexico
cab Technology, Inc.
Juárez
Phone +52 656 682 4301
www.cab.de/es

China
cab (Shanghai) Trading Co., Ltd.
Shanghai
Phone +86 (021) 6236 3161
www.cab.de/cn

South Africa
cab Technology (Pty) Ltd.
Randburg
Phone +27 11 886 3580
www.cab.de/za

Representatives in other countries on request.

Table of contents

1 Introduction ... 4
1.1 Instructions ... 4
1.2 Overview ... 4
1.3 JScript vs abc ... 4
1.4 Requirements ... 5
1.5 Restrictions ... 5

2 Instruction types ... 6
2.1 Paths ... 6
2.2 Window-Handling ... 7
2.2.1 open window ... 7
2.2.2 window transfer to ... 8
2.2.3 window transfer from ... 8
2.2.4 window read from ... 9
2.2.5 window write to ... 9
2.2.6 clear window ... 10
2.2.7 close window ... 10
2.3 Peek variables ... 11
2.4 Poke variables ... 15
2.5 Streams ... 19
2.6 Graphical User Interface ... 22
2.6.1 Object creation ... 22
2.6.2 Object properties ... 24
2.6.3 Dialogs ... 26
2.6.4 Images ... 26
2.6.5 Events ... 27
2.7 Special commands ... 28
2.7.1 Erase ... 28
2.7.2 Exists ... 29
2.7.3 Flush ... 30
2.7.4 Font ... 31
2.7.5 On interrupt break ... 32
2.7.6 On interrupt continue ... 33
2.7.7 Sound ... 34

3 Examples ... 35
3.1 Ruler ... 35
3.2 Rotated text ... 35
3.3 Label distance measurement ... 36
3.4 Reading keyboard codes ... 36
3.5 Writing to serial port ... 36
3.6 Reading and parsing data ... 37
3.7 Usage of LCD and touch events ... 38
3.8 Database Connector ... 41
3.9 Testing the I/O commands with io.xin / io.xout ... 42
3.10 Last printed label as an image ... 44
3.11 GUI ... 45
3.12 HTTP server query ... 48
3.13 UDP server query ... 48
3.14 HTTP Client ... 49
3.15 OPC-UA ... 50

4 Library ... 51
4.1 GetPrinterModel$ ... 51
4.2 GetCPUType$ ... 51
4.3 OpenDisplay ... 52
4.4 ClearDisplay ... 52
4.5 CloseDisplay ... 53
4.6 DisplayText$ ... 53
4.7 CheckStatus ... 54
4.8 PromptText$ ... 55

1 Introduction

1.1 Instructions

Important information and instructions in this documentation are designated as follows:

Attention!
Draws attention to potential risks of property damage or loss of quality.

Note!
Advices to make work routine easier or on important steps to be carried out.

▶ Handling instruction

▷ Reference to section, position, illustration number or document.

∗ Option (accessories, peripheral equipment, special fittings).

Time Information in the display.

1.2 Overview

- abc is an internal basic compiler which has been implemented for applications which require more than "only" print commands.
- It is a command subset from a BASIC programming language called "Yabasic" (at the moment V2.722). The usage of abc requires good programming knowledge of the programming language BASIC
- Except the restrictions listed later, it is 100% compatible to Yabasic, so you can use the original binaries to test your programs under Windows or Linux.
  ▷ Downloads and documentation from http://www.yabasic.de
- The following description is based on the current firmware release.
  We highly recommend to update the firmware first before abc is used!

Attention!
Please always install the current firmware before using abc!!!!!
The current firmware release can be downloaded from http://www.cab.de.

- abc works internally with Unicode, so multilingual data processing is no problem for abc programs.
- Running programs can be stopped on the printer by pressing total cancel (pressing CANCEL for more than 3 seconds on front panel), this can be disabled by ON INTERRUPT command.
- abc can also handle chr$(0) within a string which is interpreted as string end in Yabasic.

1.3 JScript vs abc

- abc and JScript work with cooperative multitasking, i.e. a complex JScript command can delay abc commands and vice versa.
- The content of a file has priority over abc output to JScript.
  This way abc can e.g. send "`M l lbl;sample`" to JScript. However this means that when a file is executed from card, abc output is delayed until the file has been completely read and closed by JScript!

Note!
To avoid synchronization problems between abc and JScript, it is better and recommended to embed JScript commands completely into abc!

- To switch off the ESC command interpretation of JScript you can use `POKE "transparent", 0` or `1`. However all data which is already in the input buffer (64 kwords on Ax and X2, variable on others) has been filtered. So do not send data with ESC in it before the POKE command has been executed!

1.4 Requirements

- Running abc needs at least 300 KB of free memory to work smoothly. Parts of this memory are not being released after finishing the program, so restarting abc is faster.
- abc is supported on the following printer families:

Generation	Printer Models
Ax	A3, A4, A6, A8, Hermes A
X2	A+, MACH4, Hermes+, Hermes C, PX, XC, XD
X3	EOS1, EOS4
X4	SQUIX, MACH 4S, EOS2, EOS5, HERMES Q, PX Q, AXON, XC Q, XD Q

1.5 Restrictions

- No mouse functions (touch coordinates on touch printers are mapped to MOUSEX and MOUSEY commands)
- No PRINT AT.
- No COMPILE, no libraries.
- No BEEP and BELL.
- No CIRCLE command.
- No BITBLT, GETBIT$ and so on.
- No SYSTEM$() function.
- WINDOW ORIGIN is not supported, i.e. the origin 0,0 is always in the upper left corner.
- Window functions work differently (no single window on big screen, but mapping of window to LCD possible)

2.1 Paths

When accessing or using files, an optional path where the file is located can be used.
If the path is missing, the default location specified in printer setup will be used.

Syntax:

[/path/]filename.ext	
[/path/]	optional path name where the file is located
filename.ext	name and extension of the file

Possible paths:

Path name	Description	Compatibility			
		Ax	X2	X3	X4
card	Default memory slot specified in printer setup	■	■	■	■
cf	Compact Flash card	■	■	-	-
cfext	Compact Flash card in external control panel	■	■	-	-
iffs	Internal memory (Internal File Flash System)	-	■	■	■
pccard	PCMCIA card	■	■	-	-
pics	Used for GUI operations to use standard icons and pictures	-	-	-	■
sd	SD Card	-	-	-	■
temp	Temporary path. Files in this folder are deleted after printer restart	-	-	-	■
usbmem	USB Stick	-	■	■	■
webdav	WebDAV folder specified in printer setup	-	-	-	■

2.2 Window-Handling

abc uses a hidden window which can be (partially) mapped to the front panel LCD.

The printer handles the window as a bitmap with 8 bit indexed colors. So each dot can have a value of 0 (black) to 255 (white). During mapping to the LCD, each color is mapped according to its brightness which is predefined as grayscales, i.e. 128 to 255 gives white pixels, 0 to 127 black pixels.

The mapping can be changed with the POKE command to RGB colors which are useful if you want to write the graphic to the card.

2.2.1 open window

Generation	Ax	X2	X3	X4
Compatibility	■	■	■	■
LCD size	120x32	128x64	160x255	272x480 or 480x272

Syntax:

open window *width*, *height*	
width	width of the window to open (in pixels)
height	height of the window to open (in pixels)

Opens a window, only one is allowed.

As this window is stored internally in standard memory, define it only the size you really need.
E.g. a window 100,100 takes 10 KB memory.

There is only one font (16 dots high), variable width with support of Latin, Greek, Cyrillic, Hebrew and Arabic scripts. The origin is in the upper left corner of the first character's bounding box. For right-to-left writing countries, the origin is in the upper right corner.

When a window is opened, it should also be closed before exiting the program ▷ 2.2.7 page 10.

Example:

```
<ABC>
open window 128, 64
poke "lcd", 1
text 0, 0, "this is a test"
wait 3
close window
</ABC>
```

2.2.2 window transfer to

Generation	Ax	X2	X3	X4
Compatibility	■	■	■	■

Syntax:

window transfer to "*name*"	
name	name of the file

Transfers the window content to a JScript image "name" which can be used e.g. with the I command.

Example:

2.2.3 window transfer from

Generation	Ax	X2	X3	X4
Compatibility	■	■	■	■

Syntax:

window transfer from "*name*"	
name	name of the file

Loads the window with a JScript image. If the window and image size are not identical the result is clipped.

Example:

```
<ABC>
image$ = "BLUBBER"
print "M l IMG;/iffs/" + image$
wait 1
open window 128, 120
window transfer from image$
poke "lcd", 1
wait 5
close window
</ABC>
```

2.2.4 window read from

Generation	Ax	X2	X3	X4
Compatibility	■	■	■	■

Syntax:

window read from "*name*"	
name	name of the file

Loads a PNG file into the actual window. Path names are allowed here. ▷ 2.1 page 6
The window has to be big enough to hold the image, else loading will fail!
Supported formats are:

- grayscale 1 to 8 bits per pixel
- paletted images 8 bits per pixel.

Example:

```
<ABC>
open window 272, 480
poke "lcd", 1
window read from "cablogo"
pause 4
close window
</ABC>
```

2.2.5 window write to

Generation	Ax	X2	X3	X4
Compatibility	■	■	■	■

Syntax:

window write to "*name*"	
name	name of the file

Saves the actual window as a PNG file on the memory card.

Example:

```
<ABC>
open window 128, 64
poke "lcd", 1
text 0, 0, "this is a test"
window write to "/sd/test"
wait 3
close window
</ABC>
```

2.2.6 clear window

Generation	Ax	X2	X3	X4
Compatibility	■	■	■	■

Syntax:

```
clear window
```

Erases the content of the opened window.

Example:

```
<ABC>
open window 128, 64
poke "lcd", 1
text 0, 0, "this is a test"
wait 3
clear window
text 0, 0, "text erased"
close window
</ABC>
```

2.2.7 close window

Generation	Ax	X2	X3	X4
Compatibility	■	■	■	■

Syntax:

```
close window
```

Closes the window opened with the function open window.

Example:

```
<ABC>
open window 128, 64
poke "lcd", 1
text 0, 0, "this is a test"
wait 3
close window
</ABC>
```

2.3 Peek variables

The peek function is an output function which returns information about the printer, the printing process...
The return value can be a string, an integer or a float depending on the command.

Syntax:		
	peek("*command*")	if the result type is an integer or a float
	peek$("*command*")	if the result type is a string
	command	command from the list below

Command	Type	Description	Compatibility			
			Ax	X2	X3	X4
direction	int	Direction of paper movement **-1**: backward **0**: standing **1**: forward	■	■	■	■
firmware	string	Returns the firmware version of the machine. E.g. "V3.37 (Jul 10 2014)"	■	■	■	■
freememory	int	Returns the free main memory in bytes (available for abc or JScript)	■	■	■	■
imageheight:*name*	int	Returns the height of the image "name" in dots **0** if not known	■	■	-	-
imagewidth:*name*	int	Returns the width of the image "name" in dots **0** if not known	■	■	-	-
http.rc	int	Return code of HTTP request Negative return codes correspond to libcurl error codes preceded by a minus sign. Positive values correspond to the HTTP status codes sent by the server, e.g. 200 for OK or 404 for NOT FOUND.	-	-	-	■
io.xin	string	Returns the state of the inputs of I/O Interface. Responds string is on 10 digits, for example: **NNNYNNNNNNN** **N** = not activated **Y** = activated Signal order is following: START, REPRINT, STOP, RSTERR, JOBDEL, LBLREM, PAUSE, FSTLBL, LBLROT, LBLFEED START signal stays only activated for 40 ms	-	-	-	■
io.xout	string	Returns the state of the outputs of I/O Interface. Same as ESCxout with additional synchronization status (paper synchronized) Responds string is on 12 digits, for example: **NNNYNNNNNNNY** **N** = not activated **Y** = activated Signal order is following: READY, JOBRDY, FEEDON, ERROR, RIBWARN, PEELPOS, HOMEPOS, ENDPOS, LBLWARN, RIBERR, MEDERR, PAPERSYNC	-	-	-	■
iobox	int	Returns the input state of the I/O box on USB. Input data is binary ORed, values ranging from **1** for input 1 to **8** for input 4. **-1** if not available	■	■	■	■

Command	Type	Description	Compatibility			
			Ax	X2	X3	X4
`jphase`	int	Returns the phase of JScript-Interpreter: **0**: waiting for label definition **1**: in process of label definition **2**: during printing **3**: standby, waiting for new job or new data for old one	■	■	■	■
`lcd_orientation`	int	Returns the orientation (in degrees) of the printer's display.	-	-	-	■
`lcd_resolution`	string	Returns the resolution in pixel of the printer's display. E.g. "272x480" or "480x272" when rotated by 90 or 270°	-	-	-	■
`line`	int	Returns the number of the last printed label	■	■	-	■
`machine`	string	Returns the type and name of the printer. E.g. "A4+/300"	■	■	■	■
`manufacturer`	string	Returns the manufacturer of the machine. E.g. "cab"	■	■	■	■
`mlength`	float	Returns the measured length of the last label distance (in mm) **0** if not known	■	■	■	■
`opcua`**`x`**`:`**`y`**	int string	**x** slot value from 0 to 4 (0 if not specified) **y** node ID or node name Returns the value from node ID or node name as an integer or a string ▷ 3.15 page 50	-	-	-	■
`opcua`**`x`**`.rc`	int string	**x** slot value from 0 to 4 (0 if not specified) Returns the error code as an integer value Returns the error code as a string value List of error codes: **0** ok **100** invalid url **101** invalid object path **102** connect failure **140** node not found **150** failed to read **151** not a scalar value **152** value type unsupported **153** not a method node **154** method incompatible **155** access denied	-	-	-	■
`os`	string	Returns "cab A-Series" or "cab EOS" only for compatibility with Yabasic	■	■	■	■
`peelmodule.` `sensorstate`	int	Get the state of peel sensor. **1** label is in peel sensor **-1** not available	-	-	-	■
`peelpos`	int	Returns **1** if the label is in peel-off position	■	■	-	■
`peri`	string	Returns the name of the peripheral. Same as JScript "q p" command	■	■	■	■
`read_controls`	int	Returns the state of "`read_controls`" ▷ `read_controls` page 17			■	■
`resolution`	float	Returns the resolution of the printer (in dpi)	■	■	■	■

Command	Type	Description	Compatibility			
			Ax	X2	X3	X4
`rfid_rssi`	int	Returns the signal quality of a detected RFID tag. Range is 0 to 100.	■	■	-	-
`sec70`	int	Returns the time in unix format - i.e. seconds since Jan 1, 1970	■	■	■	■
`serial`	string	Returns the serial number of the PCB	■	■	■	■
`slength`	float	Returns the stored label distance (in mm). **0** if not known or invalid. This is effectively the distance of the last defined label before being switched off	■	■	■	■
`source`	string	Returns the name of last data source: "RS232" "RS422" "RS485" "IEEE1284" "RAWIP" "USB" "FTP" "LPD" "ABC" "SOAP" "BLUETOOTH" "UNKNOWN"	■	■	■	■
`status`	string	Returns the state of the printer. Same as `ESCs` answer string	■	■	■	■
`ticks`	int	Returns the timer tick since startup of printer in 1/128th seconds	■	■	■	■
`user`	string	Returns the content of the non-volatile user space	-	■	■	■
`version`	float	Returns the version of Yabasic	■	■	■	■
`width`	float	Returns the maximum print width (in mm)	■	■	■	■
`winf`	string	Returns the content of the WINF buffer. Same as `ESCi` command	■	■	■	■
`xinput`	int	Returns the status of the peripheral connector input pin (XSTART)	■	■	-	■
`xoutput`	int	Reads the actual peripheral control bits	■	■	-	■
`xstatus`	string	Returns the extended state of the printer. Same as `ESCz` answer string, but without CR	■	■	■	■

Example:

```
<ABC>
print "m m"
print "zO"
print "J"
print "S l1;0,0,48,51,90"
print "H 100,0,T,R0,B0"
print "O R,P"
print "T3,4,0,5,3,b,k;Peek samples"
print "T4,8,0,3,2.5,k;OS: ", peek$("os")
print "T4,12,0,3,2.5,k;Version: ", peek("version")
print "T4,16,0,3,2.5,k;Manufacturer: ", peek$("manufacturer")
print "T4,20,0,3,2.5,k;Machine: ", peek$("machine")
print "T4,24,0,3,2.5,k;Serial: ", peek$("serial")
print "T4,28,0,3,2.5,k;Firmware: ", peek$("firmware")
print "T4,32,0,3,2.5,k;Resolution: ", peek("resolution")
print "T4,36,0,3,2.5,k;Max width: ", peek("width")
print "T4,40,0,3,2.5,k;LCD orientation: ", peek("lcd_orientation")
print "T4,44,0,3,2.5,k;LCD resolution: ", peek$("lcd_resolution")
print "T45,8,0,3,2.5,k;Line: ", peek("line")
print "T45,12,0,3,2.5,k;Mlength: ", peek("mlength")
print "T45,16,0,3,2.5,k;Direction: ", peek("direction")
print "T45,20,0,3,2.5,k;Slength: ", peek("slength")
print "T45,24,0,3,2.5,k;Free memory: ", peek("freememory")
print "T45,28,0,3,2.5,k;Status: ", peek$("status")
print "T45,32,0,3,2.5,k;XStatus: ", peek$("xstatus")
print "T45,36,0,3,2.5,k;Source: ", peek$("source")
print "T45,40,0,3,2.5,k;Peripheral: ", peek$("peri")
print "T45,44,0,3,2.5,k;XInput: ", peek("xinput")
print "A 1"
</ABC>
```

2.4 Poke variables

The poke function is an input function which sets settings on the printer.

Syntax:

poke "*command*", *params*	
command	command from the list below
params	optional parameter(s) depending on the used command

Command	Type	Description	Compatibility			
			Ax	X2	X3	X4
abort	int	Simulates pressing CANCEL/ABORT Stops abc program	-	-	-	■
backlight	int	Controls the back-light of the LCD if "lcd" is 1. **0** off **1** on **2** controlled by JScript (default)	■	■	-	-
bcolor	int	Sets the background color for abc window operations	■	■	■	■
bypass	int	**0** prevents data from interfaces to go directly to JScript **1** allows data from interfaces to go directly to JScript	■	■	■	■
cancel	int	Deletes actual print job (same as ESCc command)	-	-	-	■
color#**x**	int	Sets the RGB value for color #**x**. **x** is valid from 1 to 254. Color 0 (black) and 255 (white) cannot be modified	■	■	■	■
fcolor	int	Sets the foreground color for abc window operations	■	■	■	■
feed	int	Simulates pressing FEED	-	-	-	■
gui.peributton	int	Simulates pressing peripheral button	-	-	-	■
http.auth	string	Authentication mode for HTTP connection Valid values are: **Basic**, **Digest** or empty string (no authentication method)	-	-	-	■
http.header	int	Response stream should contain the response headers@ Valid values: **0** or **1**	-	-	-	■
http.header	string	Specify additional header for request **<header_field>: <header_field_value>** **empty string** use standard header	-	-	-	■
http.method	string	HTTP request method Valid values: **GET**, **POST**, **PUT**, **DELETE**	-	-	-	■
http.proxy	string	Specify a proxy server for the HTTP connection **<host>:<port>**	-	-	-	■
http.store	string	Write received data to the specified file instead of making it available via the input stream. Data is written in binary, i.e. without any character conversion. **[/path/]filename.ext**	-	-	-	■
http.userpwd	string	If http.auth is set, this field must contain the username and password **<username>:<password>**	-	-	-	■

Command	Type	Description	Compatibility			
			Ax	X2	X3	X4
`httpswap`	string	Can be used to swap the normal root directory and the memory card on the webserver. E.g. `poke "httpswap", "/secret"` moves the applet to /secret/index.htm and /card/index.htm to /index.htm	■	■	-	-
`io.xin`	string	Sets input signals of I/O Interface. Same tokens as `ESCxin` **FSTLBL** print first label only when Cycle sequence = Apply-Print **JOBDEL** cancel print job **LBLFEED** feeds a label **LBLREM** label removed **REPRINT** reprints the last label **RSTERR** resets the error state of the printer **START** start signal only when Print on demand = On **STOP** stop signal to interrupt operation **PAUSE=x** x = **0**: pause off x = **1**: pause on **LBLROT=x** x = **0**: labelling with primary orientation e.g. 0° x = **1**: labelling with secondary orientation e.g. 90° only with applicators with variable labelling orientation	-	-	-	■
`iobox`	int	Sets the output state of the I/O box on USB. Returns an error if not available. Output data is binary ORed, values ranging from 1 for output 1 to 8 for output 4	■	■	■	■
`key`	int	Puts a character into the key buffer. E.g. `poke "key", dec("F001")` simulates pressing the MODE key.				
		dec("F001") MODE (Ax) or MENU (X2) key	■ arrow left	■ arrow left	-	-
		dec("F002") FEED key	■ arrow down	■ arrow right	-	-
		dec("F003") CANCEL key	■ arrow right	■ arrow down	-	-
		dec("F004") PAUSE key	■ arrow up	■ arrow up	-	-
		dec("F090") CANCEL longer than 3 seconds (total cancel)	■ arrow right	■ arrow right	-	-
		dec("F100") ENTER key	-	■	-	-
		dec("F101") ENTER key longer than 2 seconds	-	■	-	-
`lcd`	int	Controls the source for the LCD. **0** standard, JScript content **1** abc window	■	■	-	-
`lcdx,lcdy`	int	Offset for the LCD in the abc window. Works only if the window is bigger than the LCD.	■	■	■	■

Command	Type	Description	Compatibility			
			Ax	X2	X3	X4
`led`	int	Controls the state of the front panel LEDs, if "lcd" is 1. Bit coded: **1** Cancel (Ax), Menu (X2) **2** Mode (Ax), **4** Feed **8** Pause **16** Arrows (Ax), Enter (X2) **32** Up arrow (X2) **64** Left arrow (X2) **128** Right arrow (X2) **256** Down arrow (X2)	■	■	-	-
`ledmask`	int	Masks the LEDs to be lit. Independent of "lcd"-value. Same bit coding as "`led`". **0** Masks the respective LED	■	■	-	-
`nice`	int	Sets the multitasking priority of abc vs. JScript. Ranges from 1 (JScript fast) to 20 (abc fast). Default is 10.	■	■	■	■
`opcua`**`x`**`:`**`y`**	int string	**x** slot value from 0 to 4 (0 if not specified) **y** node ID or node name Writes a value to node ID or node name. Value can be an integer or a string ▷ 3.15 page 50	-	-	-	■
`opcua`**`x`**`.rc`	int	**x** slot value from 0 to 4 (0 if not specified) Error code handling. If not activated, OPC-UA peek functions with errors will cause the script to fail. When activated the result of the operation is received as a result code that can be processed. **0** disable result code handling **1** enable result code handling	-	-	-	■
`opcua`**`x`**`.url`	string	**x** slot value from 0 to 4 (0 if not specified) Specify OPC url and port **<host>:<port>**	-	-	-	■
`opcua`**`x`**`.userpwd`	string	**x** slot value from 0 to 4 (0 if not specified) Specify OPC access parameters **<user>:<password>**	-	-	-	■
`pause`	int	Simulates pressing PAUSE **0** Pause OFF **1** Pause ON	-	-	-	■
`print_with_verify`	int	Controls the usage of a barcode scanner by the print engine of an enabled machine. Set to 1 for the print engine to wait for "scanresult" after each label	■	■	-	-
`read_controls`	int	Value: 0 or 1. **1** allows control characters to pass threw INPUT or INKEY$. All characters are passed to abc, including the character terminating the input line (e.g. CR). (This CR can be removed e.g. with TRIM$)	■	■	■	■

Command	Type	Description	Compatibility			
			Ax	X2	X3	X4
`scanresult`	int	Sets the result of the barcode verification scan: **1** Good, apply the label **2** Bad, display error (depending on user decision on front panel reprint will occur or not) **3** Bad, keep label on liner (reprint will occur) **4** Bad, put label in recycle position (if hardware available, reprint will occur) **5** Bad, put label on product (reprint will occur) **3+8** Bad, keep label on liner (no reprint) **4+8** Bad, put label in recycle position (if hardware available, no reprint) **5+8** Bad, put label on product (no reprint)	■	■	■	■
`stdout`	string	Outputs to systemlog	-	-	-	■
`syserror`	string	Puts the first character of the string into the error message buffer. Allowed characters are the same as in the `ESCs` response. Message will be shown when motor is not moving	■	■	-	■
`transparent`	int	**0** switches ON ESC-command interpretation **1** switches OFF ESC-command interpretation	■	■	■	■
`user`	string	Writes a value into the non-volatile user space. Max. 31 UTF-8 characters are allowed	-	-	■	■
`usererror`	string	Similar to `syserror` but with custom error string	-	-	-	■
`wakeup`	int	Wakes the printer resp. prevents it from falling asleep	■	■	-	■
`widget`	string	Puts text into abc debug widget. Up to 4 printable characters. Only digits and upper case letters are allowed	-	■	-	-
`winf`	string	Writes a value into the "winf" buffer	■	■	■	■
`xinput`	int	Triggers the print of label (analog to start input signal) on supported hardware	■	■	-	■
`xoutput`	int	Status of the peripheral connector control bits (output) Note: you have to set the peripheral mask to 0 (`x m` command) before!	■	■	-	■
`xstart`	int	Same as `xinput`	■	■	-	■

Example:

```
<ABC>
poke("cancel"),1
</ABC>
```

2.5 Streams

- Writing to an interface (e.g. `/dev/rs232`) will fail if the printer cannot send the data. There's a timeout of 10 seconds.
- Opening an interface as file stops ESC interpretation on this device.
- abc has an additional command which enables you to clear the input buffer of streams in read mode ▷ 2.7.3 page 30
- abc has an additional command to erase files ▷ 2.7.1 page 28
- `/dev/keyboard` works only if a window is opened and displayed, some keycodes have changed compared to old printers
- No random writing within a file, only append or overwriting,
- According to the filename extension the files are automatically sorted into the appropriate directories (i.e. /images, /labels, /fonts and /misc) on the memory

Syntax:

```
open id, "streamname", "mode"
id = open("streamname", "mode")
```

`id`	unique identifier (integer value) The identifier can be passed as a parameter or an available ID can be returned by the function
`streamname`	command from the list below
`mode`	mode from the list below

Stream name	Type	Description	Possible values	Compatibility Ax	X2	X3	X4
`/dev/rs232:` *`baud,`* *`handshake,`* *`parity,`* *`stopbits`*	I/O 8 bits	Serial, RS-232 interface *`baud`* *`handshake`* *`parity`* *`Stopbits`*	 1200-230400 ---, RTS/CTS, XON/XOFF N, E, O 1, 2	■	■	■	■
`/dev/ieee1284`	I/O 8 bits	Bidirectional parallel interface		■	■	-	-
`/dev/rs422:` *`baud,`* *`handshake`*	I/O 8 bits	RS-422 interface *`baud`* *`handshake`*	 1200-230400 ---, RTS/CTS, XON/XOFF	■	■	-	-
`/dev/rs485:` *`baud,`* *`address`*	I/O 8 bits	RS-485 interface *`baud`* *`address`*	 1200-230400 A-Z	■	■	-	-
`/dev/usb`	I/O 8 bits	USB slave		■	■	■	■
`/dev/rawip`	I/O 8 bits	Raw-IP Socket Server		■	■	■	■
`/dev/lpr`	I 8 bits	LPD Server		■	■	■	■
`/dev/panel`	I 16 bits	Input from front panel keys, key values are: **$F001** Mode **$F002** Form feed **$F003** Cancel **$F004** Pause **$F090** Cancel longer than 3 seconds **$F100** Enter **$F101** Enter longer than 2 seconds		 ■ ■ ■ ■ ■ - -	 ■ ■ ■ ■ ■ ■ ■	 - - - - - - -	 - - - - - - -

Stream name	Type	Description Possible values	Compatibility			
			Ax	X2	X3	X4
`/dev/keyboard`	I 16 bits	Input from external keyboard There are too many key codes to list them here. You can use following example ▷ 3.4 page 36	■	■	■	■
`/dev/jscript`	I 16 bits	JScript interpreter - needed for reading back answers	■	■	-	-
[/path/] `filename.ext`	I/O* 8/16 bits	File from memory. *[/path/]* Optional path ▷ 2.1 page 6 *filename.ext* Name of the file	■	■	■	■
`bitmap:`	O 8 bits	Last printed label as PNG file Note: a label must have been previously printed after the printer was powered on.	-	-	-	■
`http://`*url[:port]* `https://`*url[:port]*	I/O 16 bits	HTTP/HTTPS request *url* Address of website *[:port]* Optional port used on website	-	-	-	■
`mailto:address`	O 8 bits	Writes an email to the specified address. A SMTP-Server address and a return address have to be set in the setup! The subject is the first line printed into the stream.	■	■	■	■
`sql:`*ip,port*	I/O 16 bits	Database Connector, always Unicode *ip* IP address of the DBC server *port* Port used for DBC (default: 1001) You have to open two streams, one for reading, one for writing. After printing the SQL query, you have to input the result, even if you don't need them, e.g. after INSERT. The query is sent at the moment to do the first INPUT on the reading stream.	■	■	■	■
`sqlite:`*[/path/]* *filename.ext*	I/O 16 bits	Same as Database Connector but for SQLite and a local database. *[/path/]* Optional path ▷ 2.1 page 6 *filename.ext* SQLite Database filename	-	-	-	■
`tcp:`*ip,port*	I/O 16 bits	TCP client only send and receive. Only binary data. Since TCP is a stream, data is also read in the same way. There is no synchronization between the streams such as with the SQL Connector (=> flush when reading). However, only one socket is used for the connection. This means that only one connection is possible to an end point at a time. ▷ 3.12 page 48	-	-	-	■
`udp:`*ip,port*	I/O 16 bits	UDP for sending and receiving data. To enable bidirectional communication, a client socket (for writing) and a server socket (for reading) are implemented. Each API call creates its own UDP datagram. Therefore, the print function without a semicolon may result in undesirable behavior (the text and the automatically inserted newline are sent in two separate datagrams) ▷ 3.13 page 48	-	-	-	■

Mode	Description	Compatibility			
		Ax	X2	X3	X4
r	Read Opens the stream for reading File reading and writing automatically transforms Unicode to ASCII and vice versa according to selected codepage, reading an Unicode or ASCII file is automatically detected	■	■	■	■
w	Write Opens the stream for writing File reading and writing automatically transforms Unicode to ASCII and vice versa according to selected codepage, reading an Unicode or ASCII file is automatically detected	■	■	■	■
a	Append Opens the stream for appending	■	■	■	■
rb	Read Binary Opens the stream for reading without transforming. File reading and writing uses only low-byte of e.g. string	■	■	■	■
wb	Write Binary Opens the stream for writing without transforming File reading and writing uses only low-byte of e.g. string	■	■	■	■
ab	Append Binary Opens the stream for appending without transforming	■	■	■	■
wu	Write Unicode Opens the stream for writing using Unicode	■	■	■	■
au	Append Unicode Opens the stream for appending using Unicode	■	■	■	■

Example:

```
<ABC>
a$ = "Hello " + chr$(dec("20AC"))
open 1,"test.dat","w"
print #1 a$
close 1
open 1,"testu.dat","wu"
print #1 a$
close 1
open 1,"testb.dat","wb"
print #1 a$
close 1
</ABC>
```

This example demonstrates the differences for file handling. Connect the memory card/USB drive to your computer and use a hex editor to see the different results.

2.6 Graphical User Interface

Generation	Ax	X2	X3	X4
Compatibility	-	-	-	■

The following graphical user interface (GUI) elements are supported:

- Text buttons
- Image buttons
- Labels (for texts or images)
- Line edits
- Combo boxes
- Check boxes
- Default buttons (in the design of the printer)
- Movies

In addition, ready-made cab editors can be used as dialogs in full-screen mode such as:

- fileopen
- numeric
- dates
- time
- info
- confirm

2.6.1 Object creation

To create GUI elements in abc, a poke command is issued with the following syntax for name and value:

Syntax:

poke "gui.*Objecttype*.add", "*Objectname*,*Objectparams*"	
Objecttype	Type of GUI element from the list below
Objectname	Unique name of the element
Objectparams	Parameters of the element from the list below

The object properties depend on the object type.

Note!

The object name can only contain lowercase letters a-z, digits 0-9 or underscore _.
Capital letters are not possible, since the object names are also used as part of peek and poke commands, which are always converted to lower case by abc.

Object type	Object parameter	Description
button	x-pos, y-pos, width, height, caption	Button which can be pressed on touch-display. Buttons can also host images. The image property can be set later. Note: in contrast to the label, the size of the image remains unchanged.

Object type	Object parameter	Description

`label`	`x-pos,`
	`y-pos,`
	`width,`
	`height,`
	`caption`
		Labels can also host images. To do this, leave the label blank (the comma must also be omitted in this case) and set the image property later.
		Note: the size of the displayed pixmap is adapted to the size of the label while maintaining the aspect ratio (smooth transformation, non-expanding).

`lineedit`	`x-pos,`
	`y-pos,`
	`width,`
	`height,`
	`content`
		Edit field which can be operated via keyboard.
		A virtual keyboard is displayed when you click on the lineedit.
		The (predefined) content is displayed by the lineedit.

`combobox`	`x-pos,`
	`y-pos,`
	`width,`
	`height,`
	`element1,`
	`element2,`
	`...`
	`elementX`
		The object of type combobox is filled with the given elements. Elements are separated by commas, i.e. the comma itself cannot appear in the text of an element (no escaping possible).

`checkbox`	`x-pos,`
	`y-pos,`
	`CheckState 0|1`
		Checkboxes have a default height, so there is no width and height. Also, checkboxes do not contain any text description. This must be done via a label object.

`pixbutton`	`x-pos,`
	`y-pos,`
	`size,`
	`type`
		Size refers to the width of the generated button.
		Possible values are: `tiny`, `small`, `medium` or `large`.
		Buttons can currently be of type: `back`, `cancel`, `feed`, `help`, `minus`, `next`, `ok`, `pause`, `plus`, `setup`, `start`, `stop`, `trigger`. The matching icons are available in every size and are loaded from the default location of the printer icons.
		If you want to use your own icons on buttons, you should switch to the general button type.

`movie`	`x-pos,`
	`y-pos,`
	`width,`
	`height`
		Movie object is an animated GIF file.
		The image is automatically scaled in `width` and `height`.

High-level GUI elements can no longer be deleted but are automatically discarded when the abc window is closed. The object name is then used for referencing during event processing or for reading or setting object properties.

2.6.2 Object properties

Object properties are referenced via the object name and are basically accessible via the syntax below.

To set a property:

Syntax:

```
poke "gui.Objectname.Property", IntegerValue
poke "gui.Objectname.Property", "StringValue"
```

Objectname	Unique name of the element
Property	Property from the list below
IntegerValue	Value of type integer
StringValue	Value of type string

To read a property:

Syntax:

```
val = peek("gui.Objectname.property")
val$ = peek$("gui.Objectname.property")
```

Objectname	unique name of the element
Property	Property from the list below

Typical for Basic, a distinction is made between string and integer properties. The available properties depend on the object type.

Property	Type	Mode	For object type	Value	Description
destroy	int	Write	combobox	**0**	Destroy the object
checked	int	Read / Write	checkbox	**0**	Checkbox is unchecked
				1	Checkbox is checked
enabled	int	Read / Write	all	**0**	Object is disabled
				1	Object is enabled
focus	int	Write	lineedit	**0**	Hide focus
				1	Set focus
image	string	Write	button label		Virtual path to an image on the file system, e.g. /IFFS/images/my.png On buttons, the image is placed in addition to the text
mask	string	Write	lineedit		Specification of an input mask, definition as for JScript input command
numeric	int	Write	lineedit	**0**	Display a standard keyboard
				1	Display a numeric keyboard only
pixmap	string	Write	pixbutton	**back** **cancel** **feed** **help** **minus** **next** **ok** **pause** **plus** **setup** **start** **stop** **trigger**	Change button type

Property	Type	Mode	For object type	Value	Description
`text`	string	Read / Write	button combobox label lineedit		Text assigned to the object type For a combo box: comma-separated string list which makes it possible to reinitialize the selection list
`visible`	int	Read / Write	all	**0**	Hide the object
				1	Show the object

The properties `checked`, `enabled`, `text` and `visible` can be queried from abc via peek.

Example:

```
poke "gui.my_checkbox.checked", 1
poke "gui.my_combobox.text", "a,new,choice"
poke "gui.my_combobox.text", "New"
```

2.6.3 Dialogs

The abc interface supports the possibility to open modal dialogs.

Syntax:	**poke**("gui.*Dialogclass*.show"), "*Caption,Dialogparams*"
Dialogclass	Type of dialog from the list below
Caption	Text displayed at the top of the dialog (title)
Dialogparams	Parameters of the dialog from the list below

Only one dialog can be opened at a time. A `DialogClosed` event is received via the event interface as soon as the dialog has been closed successfully or through cancellation.

The following dialog classes are available.

Dialog class	Dialog parameter	Description
`fileopen`	`subpath` `[,fileextension]`	Displays a file selection dialog. Same dialog as pressing *Menu* => *Storage* => *Load label* `subpath` is one of the following path: labels, images, fonts, misc. `fileextention` is the extension of the file If no file extension is specified, a wildcard (*) is set.

`numeric`	`Value,`
	`Min,`
	`Max,`
	`Step,`
	`Digits`
	`[,Unit]`
		Displays a numerical selection slider dialog
		(same as in *Menu* => *Setup* => *Printing* => *Print position X*)

`date`	`Day.Month.Year`	Displays a date calendar dialog (same as in *Menu* => *Setup* => *Time* => *Date*)
`time`	`Hour:Minute`	Displays a time dialog (same as in *Menu* => *Setup* => *Time* => *Time*)
`info`	`Displaytext`	Displays an information dialog (same as in *Menu* => *Information*)

`confirm`	`1. Buttontext,`
	`[2. Buttontext,]`
	`Dialogtext`
		Displays a confirmation dialog

2.6.4 Images

The pseudo-path `/PICS` can be used to display standard icons and pictures from printer.

Available pictures/icons are: `background`, `logo`, `back`, `next`, `home`.

2.6.5 Events

Events of the GUI objects can be polled with the help of the `peek` gui.event command.

Syntax: `peek$("gui.event")`

If no events are available at the time of the call, the peek command returns with an empty string.

Alternatively, a string of the form `objectname:eventclass:eventproperties` is returned.
The object name is the name given when the element was instantiated. The possible event class depends on the object type.

Event class	Event parameter	For object type	Description
`Clicked`	`-`	button pixbutton	Triggered when the button is pressed
`TextChanged`	`NewText`	combobox lineedit	Triggered when the text changed
`Checked`	`0` `1`	checkbox	Triggered when the checkbox is checked/unchecked
`ReturnPressed`	`-`	lineedit	Triggered when the return key is pressed in the keyboard dialog

`OnScreenKeyboardVisible`	`0:cancel`
	`0:ok`
	`1`
		lineedit	

`DialogClosed`	`ok` `fail`	dialogs	Triggered when the dialog is closed

2.7 Special commands

2.7.1 Erase

Generation	Ax	X2	X3	X4
Compatibility	■	■	■	■

Syntax:

erase "*[path]name.ext*"	
[path]	Optional parameter to select the pathname where the files are located ▷ 2.1 page 6
name	File name of the file on memory card
ext	Extension of file

Deletes a file on a memory card.

The path is optional, if it is not specified the default memory selected in the printer setup menu will be used to search for the file.

If the file is not found, an error message will be displayed.

Example:

```
<ABC>
erase "Etiq1.lbl"
erase "/iffs/Etiq2.lbl"
</ABC>
```

2.7.2 Exists

Generation	Ax	X2	X3	X4
Compatibility	■	■	■	■

Syntax:

exists "*[/path/]*name.ext"	
[/path/]	Optional parameter to select the pathname where the files are located ▷ 2.1 page 6
name	Filename of the file on memory card or stream name
ext	Extension of file (if a filename is specified)

Checks for the existence of files on a memory card or devices available on printer.

The path is optional, if it is not specified the default memory selected in the printer setup menu will be used to search for the file.

If the file is not found, an error message will be displayed.

Example:

```
<ABC>
open window 272, 480
poke "lcd", 1
if exists("/dev/rawip") then
  text 0, 0, "RAWIP exists!"
else
  text 0, 0, "RAWIP not found!"
endif
if exists("/sd/label1.lbl") then
  text 0, 20, "Label1 exists!"
else
  text 0, 20, "Label1 not found!"
endif
wait 3
poke "lcd", 0
close window
</ABC>
```

2.7.3 Flush

Generation	Ax	X2	X3	X4
Compatibility	■	■	■	■

Syntax:

flush *#id*	
id	unique identifier (integer value) of the stream to clear

Clears the input buffer of /dev-streams in read mode.
flush #0 clears standard input.

Example:

```
<ABC>
open 1,"/dev/jscript","r"
open 2,"/dev/rs232","w"
print "qm"
line input #1 a$
print #2 a$
close 2
close 1
flush #0
print "f"
</ABC>
```

2.7.4 Font

Generation	Ax	X2	X3	X4
Compatibility	-	-	■	■

Syntax:

`font` "*`name,size`*"	
`name`	name of the font
`size`	size of the font (in pixels)

Changes the font used to display texts on the printer's display.

Example:

```
<ABC>
open window 272, 480
font "Swiss,10"
text 0,0,"Swiss"
font "Swiss,20"
text 0,20,"Swiss"
font "Swiss,30"
text 0,50,"Swiss"
font "Swiss,40"
text 0,90,"Swiss"
font "Swiss Bold,40"
text 0,130,"Swiss"
font "Monospace,15"
text 80,20,"Monospace"
font "Monospace,25"
text 80,40,"Monospace"
font "Default"
text 80,0,"Default"
poke "lcd", 1
pause 5
poke "lcd", 0
close window
</ABC>
```

2.7.5 On interrupt break

Generation	Ax	X2	X3	X4
Compatibility	-	-	-	■

Syntax:

```
on interrupt break
```

Example:

```
<ABC>
on interrupt continue
poke "bypass", 1
open 1,"/dev/keyboard","r"
open window 120,50
poke("lcd"),1
do
  do
    k = peek(#1)
    if k <> -1 then
      on interrupt break
      poke("abort"), 1
    endif

    a$ = jget$
    if a$ <> "" break
  loop

  clear fill rectangle 10,10 to 20,60
  text 10,10,a$
  b$ = "<" + a$ + ">"
  jput b$
  pause 1
loop
</ABC>
J
S e;0,0,30,32,100
H 50,0,T
T:text;10,10,0,3,5;[SER:1]
T 10,20,0,3,5;[ABC:text]
A 5
```

2.7.6 On interrupt continue

Generation	Ax	X2	X3	X4
Compatibility	-	-	-	■

Syntax:

```
on interrupt continue
```

Example:

```
<ABC>
on interrupt continue
poke "bypass", 1
open 1,"/dev/keyboard","r"
open window 120,50
poke("lcd"),1
do
  do
    k = peek(#1)
    if k <> -1 then
      on interrupt break
      poke("abort"), 1
    endif

    a$ = jget$
    if a$ <> "" break
  loop

  clear fill rectangle 10,10 to 20,60
  text 10,10,a$
  b$ = "<" + a$ + ">"
  jput b$
  pause 1
loop
</ABC>
J
S e;0,0,30,32,100
H 50,0,T
T:text;10,10,0,3,5;[SER:1]
T 10,20,0,3,5;[ABC:text]
A 5
```

2.7.7 Sound

Generation	Ax	X2	X3	X4
Compatibility	-	■	■	-

Syntax:

sound "soundname"	
soundname	name of the sound to play Possible values: **Beep** **Bicycle** **Chime** **Comic** **Ding** **Frog** **Rooster** **Synthesizer** **Tribble** **Weep** **Whistle**

Plays a sound using the printer buzzer.

Example:

```
<ABC>
sound("Beep")
sound("Rooster")
</ABC>
```

3.1 Ruler

Small program to print a 100 mm long ruler with 1 mm markings on a label of size 104x68 mm.

```
<ABC>
' Test label for ruler
print "m m"
print "J"
print "S l1;0,0,68,71,104"
print "G 0,10,0;L:100,.1"
for x = 0 to 100
    if mod(x,10) = 0 then
        print "G ",x,",10,270;L:4,.1"
    else
        print "G ",x,",10,270;L:2,.1"
    endif
next x
print "A 1"
</ABC>
```

3.2 Rotated text

Small program to print a text in a circle.

```
<ABC>
' Test label for rotated text
print "m m"
print "J"
print "S l1;0,0,68,71,104"
a$ = "Rotated text with Euro sign: " + chr$(dec("20AC")) + " "
n = len(a$)
d = 360/n
for i = 1 to n
    w = ((i-1)*d)/180*pi
    x = 50-25*cos(w)
    y = 30-25*sin(w)
    r = 90-(i-1)*d
    if r < 0
        r = r + 360
    print "T ",x,",",y,",",r,",3,6,b;" + mid$(a$, i, 1)
next i
print "T 0,30,0,3,5;[J:c100]" + date$
print "T 0,38,0,3,5;[J:c100]" + time$
print "A 1"
</ABC>
```

3.3 Label distance measurement

Small program for measuring the distance between two label edges.

```
<ABC>
repeat
    'read measured length
    dy = peek("mlength")
    if dy > 0
        break
    print "f"
    wait 0.25
    'wait until standing again REPEAT
until(peek("direction")=0)
print "m m"
print "J"
print "O R"
print "S l1;0,0,",dy-2,",",dy,",100"
print "T 0,10,0,3,5;Measured distance: ", dy, " mm"
print "A 1"
</ABC>
```

3.4 Reading keyboard codes

This program reads keyboard codes and displays the values on the printer's display.

```
<ABC>
open 1, "/dev/keyboard", "r"
open window 120,32
poke "lcd", 1
do
    do
        x = peek(#1)
        if x <> -1
              break
    loop
    clear window
    text 0, 0, "Last character:"
    text 0, 16, "$" + hex$(x)+" = " + chr$(x)
loop
close window
</ABC>
```

3.5 Writing to serial port

This program writes some data on the RS-232 port.

```
<ABC>
a$ = "Hello " + chr$(dec("20AC"))
open 1, "/dev/rs232:57600,RTS/CTS", "w"
print #1 a$, chr$(13);
for i = 1 to 10
    print #1 i, chr$(13);
next i
close 1
</ABC>
```

3.6 Reading and parsing data

Simple program to show the capture of interface data, parsing it, extracting the data and sending it forward to the JScript interpreter.

```
<ABC>
print "m m"
print "J"
print "S l1;0,0,68,71,104"
print "T:t1;20,10,0,3,8;"
print "T:t2;20,20,0,3,8;"
print "T:t3;40,40,0,3,8;"
label start
line input a$
if left$(a$, 15) = "194300301480070" then
    print "R t2;", mid$(a$, 16)
endif
if left$(a$, 15) = "194300300580172" then
    print "R t3;", mid$(a$, 16)
endif
if left$(a$, 15) = "194300301970073" then
    print "R t1;", mid$(a$, 16)
endif
if a$ = "Q0001" then
    print "A 1"
endif
goto start
</ABC>
```

Here is the original Datamax DPL data stream sent from Easylabel:

```
M3000
<STX>d
<STX>e
<STX>f260
<STX>O0220
<STX>V0
<STX>L
D11
PA
SA
H10
z
194300301480070Rot
19430030058017248
194300301970073Bernd
W
Q0001
E
<STX>L
D11
PA
SA
H10
z
194300301480070gelb
19430030058017248
194300301970073Bertha
W
Q0001
E
```

3.7 Usage of LCD and touch events

```
<ABC>
quan$ = eosnuminput$("Enter","Quantity","1","10")
'
sub eosnuminput$(line1$,line2$,minlen$,maxlen$)
  local inp$,x,y,delbut,backbut,cancelbut,okbut
'
  open window 272,480
  poke("lcd"),1
'
  ' Frames around input fields
  rectangle 8,41 to 262,439:rectangle 16,111 to 255,148
'
  ' Cancel and OK buttons
  rectangle 26,379 to 121,426:rectangle 149,379 to 244,426
'
  ' Boxes
  rectangle 17,170 to 93,214:rectangle 98,170 to 174,214:rectangle 179,170 to 255,214
  rectangle 17,216 to 93,260:rectangle 98,216 to 174,260:rectangle 179,216 to 255,260
  rectangle 17,262 to 93,306:rectangle 98,262 to 174,306:rectangle 179,262 to 255,306
  rectangle 17,308 to 93,352:rectangle 98,308 to 174,352:rectangle 179,308 to 255,352
'
  ' Texts
  font "Monospace, 30"
  text 46,172,"1":text 127,172,"2":text 208,172,"3"
  text 46,218,"4":text 127,218,"5":text 208,218,"6"
  text 46,264,"7":text 127,264,"8":text 208,264,"9"
  text 46,310,".":text 127,310,"0":text 208,310,chr$(8592)
  text 64,381,"X":text 180,381,"OK"
'
  ' Title
  font "Swiss, 16"
  text 17,50,line1$
  text 17,67,line2$
'
  ' Input field
  char$ = ""
  font "Monospace, 16"
  clear fill rectangle 18,114 to 253,146
  text 18,120,char$ + "_"
'
  do
    x = mousex
    y = mousey
    inp$ = ""
    delbut = 0
    backbut = 0
    cancelbut = 0
    okbut = 0
'
    if x >= 17 and x <= 93 and y >= 170 and y <= 214
      inp$ = "1"
    if x > 98 and x <= 174 and y >= 170 and y <= 214
      inp$ = "2"
    if x > 179 and x <= 255 and y >= 170 and y <= 214
        inp$ = "3"
    if x >= 17 and x <= 93 and y >= 216 and y <= 260
        inp$ = "4"
    if x > 98 and x <= 174 and y >= 216 and y <= 260
        inp$ = "5"
```

```
    if x > 179 and x <= 255 and y >= 216 and y <= 260
      inp$ = "6"
    if x >= 17 and x <=93 and y >= 262 and y <= 306
      inp$ = "7"
    if x > 98 and x <= 174 and y >= 262 and y <= 306
      inp$="8"
    if x > 179 and x <= 255 and y >= 262 and y <= 306
      inp$="9"
    if x >= 17 and x <= 93 and y >= 308 and y <= 352
      delbut = 1
    if x > 98 and x <= 174 and y >= 308 and y <= 352
      inp$ = "0"
    if x > 179 and x <= 255 and y >= 308 and y <= 352
      backbut = 1
    ' CANCEL and OK
    if x >= 26 and x <= 121 and y >= 379 and y <= 426
      cancelbut = 1
    if x > 149 and x <= 244 and y >= 379 and y <= 426
      okbut = 1
    if len(inp$) > 0 then
      do
        x = mousex
        y = mousey
'
        if x = -1 and y = -1
          break
        pause 0.01
      loop
      char$ = char$ + inp$
      clear fill rectangle 18,114 to 253,146
'
      if len(char$) <= 22 then
        text 18,120,char$ + "_"
      else
        text 18,120,right$(char$, 22) + "_"
      endif
    endif
'
    if backbut = 1 and len(char$) > 0 then
      do
        x = mousex
        y = mousey
'
        if x = -1 and y = -1
          break
        pause 0.01
      loop
      char$ = mid$(char$,1,len(char$)-1)
      clear fill rectangle 18,114 to 253,146
'
      if len(char$) <= 22 then
        text 18,120,char$ + "_"
      else
        text 18,120,right$(char$,22) + "_"
      endif
    endif
'
    if backbut = 1 and len(char$) > 0 then
      do
        x = mousex
        y = mousey
'
```

```
        if x = -1 and y = -1
          break
        pause 0.01
      loop
      char$ = mid$(char$,1,len(char$)-1)
      clear fill rectangle 18,114 to 253,146
'
      if len(char$) <= 22 then
        text 18,120,char$ + "_"
      else
        text 18,120,right$(char$,22) + "_"
      endif
    endif
    if okbut = 1 and len(char$) > 0 then
        do
          x = mousex
          y = mousey
'
          if x = -1 and y = -1
            break
          pause 0.01
        loop
    endif
    if cancelbut = 1 then
      do
        x = mousex
        y = mousey
'
        if x = -1 and y = -1
          break
        pause 0.01
      loop
      end
    endif
    if okbut = 1
      break
  loop
  close window
  poke("lcd"),0
  if okbut = 1
    return char$
end sub
</ABC>
```

3.8 Database Connector

Shows the usage of Database Connector from abc.

```
<ABC>
poke "bypass", 1
open 1, "sql:192.168.3.103,1001", "w"
open 2, "sql:192.168.3.103,1001", "r"
print #1, "SELECT * FROM Table1 WHERE ID='123'"
poke "read_controls", 1
line input #2 a$
poke "read_controls", 0
close #1
close #2
print "m m"
print "J"
print "S l1;0,0,68,70,100"
print "T 10,10,0,5,pt10;" + a$
print "A 1"
</ABC>
```

3.9 Testing the I/O commands with io.xin / io.xout

```
<ABC>
print "m m"
print "J"
print "O R,J"
print "P"
print "S l1;0,0,68,70,100"
print "T 10,10,0,5,pt10;TEST XIN/XOUT"
print "A 1"
do
  getxout()
  if (jobrdy)
    break
loop
pause 0.05
poke "io.xin", "START"
do
  getxout()
  if (peelpos)
    break
loop
poke "io.xin", "LBLREM"
do
  getxout()
  if (!peelpos)
    break
loop
do
  if peek("direction") = -1
    break
loop
do
  if peek("direction") = 0
    break
loop
'needed, because there is a gap in the printengine
pause 1
poke "io.xin", "REPRINT"
do
  getxout()
  if (jobrdy)
    break
loop
pause 0.05
poke "io.xin", "START"
do
  getxout()
  if (peelpos)
    break
loop
poke "io.xin", "LBLREM"
```

```
sub getxout()
  local xout$,tmp$
'
  xout$ = peek$("io.xout")
'
  for a = 1 to len(xout$)
    if mid$(xout$,a,1) = "Y" then
      tmp$ = tmp$ + "1"
    else
      tmp$ = tmp$ + "0"
    endif
  next a
'
  xout$ = tmp$
  ready      = val(mid$(xout$,1,1))
  jobrdy     = val(mid$(xout$,2,1))
  feedon     = val(mid$(xout$,3,1))
  perror     = val(mid$(xout$,4,1))
  ribwarn    = val(mid$(xout$,5,1))
  peelpos    = val(mid$(xout$,6,1))
  homepos    = val(mid$(xout$,7,1))
  endpos     = val(mid$(xout$,8,1))
end sub
</ABC>
```

3.10 Last printed label as an image

This program prints a label, saves it as a PNG file and displays it on the printer's display

```
m m
J
H 150,0
Se;0,0,20,22,40
O J,R
T 10,10,0,5,pt15;Hello World!
A1


<ABC>
open window 272,480
window read from "/IFFS/background.png"
font "Swiss,16"
poke "gui.label.add", "body,5,5,261,320,"
poke "gui.button.add", "exit,180,405,85,45,Exit"
poke("lcd"),1
'
if exists("bitmap:") then
   open 3,"bitmap:","rb"
   open 4,"/IFFS/bitmap.png","wb"
'
   do
      d = peek(#3)
      if d <> -1 then
         poke #4, chr$(d)
      else
         break
      endif
   loop
'
   close #3
   close #4
'
   poke "gui.body.image", "/IFFS/bitmap.png"
else
   poke "stdout", "No bitmap: no printed label"
endif
'
do
   var$ = peek$("gui.event")
   if (instr(var$, "exit:Click")) then
      break
   endif
loop
</ABC>
```

3.11 GUI

```
<ABC>
open window 272,480
window read from "/IFFS/background.png"
'
poke("color#1"), dec("000000")
poke("color#2"), dec("ffffff")
poke("color#3"), dec("999999")
'
poke("fcolor"), 2
poke("bcolor"), 3
'
font "Swiss, 12"
poke "gui.label.add", "lbl,60,10,200,40,Test"
'
font "Swiss, 20"
poke "gui.label.add", "lnumeric,90,15,200,40,42"
poke "gui.label.add", "ldate,90,70,200,40"
poke "gui.label.add", "ltime,90,125,200,40"
'
poke "gui.checkbox.add", "cb,10,10,0"
'
poke("fcolor"), 1
poke "gui.combobox.add", "combo,10,60,200,40,Enabled,Disabled"
'
poke "gui.pixbutton.add", "right,145,330,large,next"
poke "gui.pixbutton.add", "left,15,330,large,back"
poke "gui.button.add", "remove,10,110,200,40,Remove"
poke "gui.left.enabled",0
'
poke "gui.combobox.add", "pcombo,10,210,200,40,labels,images"
poke "gui.button.add", "open,10,260,200,40,Open..."
combo_avail = 1
'
poke "gui.button.add", "bnumeric,210,15,48,48,"
poke "gui.bnumeric.image", "/IFFS/images/setup_cutting_normal.png"
'
poke "gui.button.add", "bdate,210,70,48,48,"
poke "gui.bdate.image", "/IFFS/images/setup_region_normal.png"
'
poke "gui.button.add", "btime,210,125,48,48,"
poke "gui.btime.image", "/IFFS/images/setup_time_normal.png"
'
poke "gui.button.add", "binfo,210,180,48,48,"
poke "gui.binfo.image", "/IFFS/images/short_status_normal.png"
'
poke "gui.bdate.visible", 0
poke "gui.btime.visible", 0
poke "gui.binfo.visible", 0
poke "gui.bnumeric.visible", 0
'
poke "gui.lnumeric.visible", 0
poke "gui.ldate.visible", 0
poke "gui.ltime.visible", 0
'
dim fields$(6)
a = split(date$,fields$(),"-")
poke "gui.ldate.text", fields$(3) + "." + fields$(2) + "." + fields$(4)
'
a = split(time$,fields$(),"-")
poke "gui.ltime.text", fields$(1) + ":" + fields$(2) + ":" + fields$(3)
'
```

```
poke("lcd"),1
'
current_label$ = ""
'
do
   var$ = peek$("gui.event")
'
   if (instr(var$, "right:Click")) then
      if (peek("gui.left.enabled") = 1) then
         break
      else
         poke "gui.right.pixmap", "cancel"
         poke "gui.left.enabled",1
'
         poke "gui.pcombo.visible", 0
         poke "gui.combo.visible", 0
         poke "gui.open.visible", 0
         poke "gui.cb.visible", 0
         poke "gui.lbl.visible", 0
         poke "gui.remove.visible", 0
'
         poke "gui.bdate.visible", 1
         poke "gui.btime.visible", 1
         poke "gui.binfo.visible", 1
         poke "gui.bnumeric.visible", 1
'
         poke "gui.lnumeric.visible", 1
         poke "gui.ldate.visible", 1
         poke "gui.ltime.visible", 1
      endif
   endif
'
   if (instr(var$, "left:Click")) then
      poke "gui.left.enabled", 0
      poke "gui.right.pixmap", "next"
'
      poke "gui.pcombo.visible", 1
      poke "gui.combo.visible", 1
      poke "gui.open.visible", 1
      poke "gui.cb.visible", 1
      poke "gui.lbl.visible", 1
      poke "gui.remove.visible", 1
'
      poke "gui.bdate.visible", 0
      poke "gui.btime.visible", 0
      poke "gui.binfo.visible", 0
      poke "gui.bnumeric.visible", 0
'
      poke "gui.lnumeric.visible", 0
      poke "gui.ldate.visible", 0
      poke "gui.ltime.visible", 0
   endif
'
   if (instr(var$, "bnumeric:Click")) then
      current_label$ = "lnumeric"
      val$ = peek$("gui.lnumeric.text")
      poke "gui.numeric.show", "Numeric dialog,"+val$+",0,100,1,0"
   endif
'
   if (instr(var$, "bdate:Click")) then
      current_label$ = "ldate"
      poke "gui.date.show", "New day?,10.10.2021"
   endif
```

```
'
   if (instr(var$, "btime:Click")) then
      current_label$ = "ltime"
      poke "gui.time.show", "Time change?,10:10"
   endif
'
   if (instr(var$, "binfo:Click")) then
      current_label$ = ""
      poke "gui.info.show", "Printer info,This is a great text that
describes the printer completely."
   endif
'
   if (instr(var$, "open:")) then
      current_label$ = "lbl"
      val2$ = peek$("gui.pcombo.text")
      poke "gui.fileopen.show","File selection," + val2$
   endif
'
   if (instr(var$, "DialogClosed:ok:")) then
      if (current_label$ <> "") then
         var2$ = right$(var$, len(var$)-17)
         poke "gui." + current_label$ + ".text", var2$
      endif
   endif
'
   if (instr(var$, "combo:") = 1) then
      if (instr(var$, "TextChanged:Enabled")) then
         poke ("gui.cb.enabled"),1
      else
         poke ("gui.cb.enabled"),0
      endif
   endif
'
   if (instr(var$, "remove:")) then
      if (combo_avail = 1) then
         combo_avail = 0
         poke ("gui.combo.destroy"),0
         poke "gui.remove.text", "Create"
      else
         combo_avail = 1
         poke ("gui.combobox.add"),"combo,10,60,200,40,Enabled,Disabled"
         poke "gui.remove.text", "Remove"
      endif
   endif
loop
</ABC>
```

3.12 HTTP server query

```
<ABC>
open 1, "tcp:192.168.200.71,80","wb"
open 2, "tcp:192.168.200.71,80","rb"
open 3, "/dev/rawip", "w"
'
print #1,"GET /cgi-bin/develop HTTP/1.1\r\n";
print #1,"Host: 192.168.200.71\r\n";
print #1,"Connection: close\r\n";
print #1,"\r\n";
'
do
   if eof(#2) > 0 then
      break
   endif
'
   line input #2 myline$
   print #3,myline$
loop
'
close #1
close #2
close #3
</ABC>
```

3.13 UDP server query

`eof` can be used to check whether the datagram has ended without triggering the reading of a new datagram. To do this, `eof(#2)` must be called before `peek(#2)`.

```
<ABC>
open 1, "udp:192.168.200.71,7777","wb"
open 2, "udp:192.168.200.71,7778","rb"
open 3, "/dev/rawip","w"
'
'Hello and World will be sent in one datagram
print #1, "Hello\nWorld\n";
'
do
   if eof(#2) then
       print #3 "FINISH"
       break
   else
       c = peek(#2)
'
       if c <> -1 then
           print #3 chr$(c);
       else
           print #3 "WAIT"
       endif
   endif
loop
'
close #1
close #2
close #3
</ABC>
```

3.14 HTTP Client

HTTP servers are connected via classic abc stream objects, i.e. data is sent to the server via an output stream and the response is read from the server via an input stream. While the URL is part of the `open` call, all other configuration parameters are sent using `poke` commands. In principle, only one simultaneous request is possible. This means that only one input and output stream can be open at a time. Closing the stream resets the configuration set via `poke`. After receiving `EOF` on the input stream, the HTTP operation is complete and the response code can be queried using a `peek` command. Only then can the streams be closed.

```
<ABC>
poke "http.method","POST"
poke "http.auth","digest"
poke "http.userpwd","admin:admin"
'
open 3,"http://192.168.200.71/cgi-bin/set","w"
open 4,"http://192.168.200.71/cgi-bin/set","r"
open 5,"/IFFS/drucker.txt","wb"
'
print #3 "cmd=2&id=ID_HEAT_LEVEL&value=2&tree_id=ID_SETUP"
'
do
   x = peek(#4)
   if x <> -1 then
      poke #5, chr$(x)
   else
      if eof(#4) then
         poke "stdout", "EOF"
         break
      endif
   endif
loop
'
close #3
close #4
close #5
</ABC>
```

When using `poke` http.store, the input stream cannot be read and doesn't deliver any data because the server's response data is sent directly to the specified file. The `eof` query remains valid and is required to complete the request.

```
<ABC>
poke "http.store", "/IFFS/gaga.png"
open 3,"https://127.0.0.1/gaga.png","rb"
'
do
   if eof(#3) then
      poke "stdout", "EOF"
      break
    endif
loop
'
rc = peek("http.rc")
v$ = "HTTP RC=" + str$(rc)
poke "stdout", v$
'
close #3
</ABC>
```

3.15 OPC-UA

This sample shows how to handle with OPC-UA functions and print a label with information from them.

```
<ABC>
poke "opcua0.userpwd","opcuser:opcpass"
poke "opcua0.url","192.168.16.116:4840"
poke "opcua0.rc",1
'
operatingtime$ = str$(peek("opcua0:2:DeviceSet,3:Printer,3:Statistics,3:Operating
Time") / 60)
operatingtime$ = left$(operatingtime$, instr(operatingtime$, ".") - 1)
'
print "m m"
print "zO"
print "J"
print "S l1;0,0,48,51,90"
print "H 100,0,T,R0,B0"
print "O R,P"
print "T3,4,0,5,3,b,k;abc OPC UA sample"
print "T4,8,0,3,2.5,k;Manufacturer: ",
peek$("opcua0:2:DeviceSet,3:Printer,2:Manufacturer")
print "T4,12,0,3,2.5,k;Machine: ", peek$("opcua0:2:DeviceSet,3:Printer,2:Model")
print "T4,16,0,3,2.5,k;Firmware: ",
peek$("opcua0:2:DeviceSet,3:Printer,2:SoftwareRevision")
print "T4,20,0,3,2.5,k;Serial: ",
peek$("opcua0:2:DeviceSet,3:Printer,2:SerialNumber")
print "T4,24,0,3,2.5,k;Total labels: ",
peek$("opcua0:2:DeviceSet,3:Printer,3:Statistics,3:Labels")
print "T4,28,0,3,2.5,k;Operating time: ", operatingtime$, " h"
print "T4,32,0,3,2.5,k;Thermal direct: ",
str$(peek("opcua0:2:DeviceSet,3:Printer,3:Statistics,3:Thermal Direct") / 1000), " m"
print "T4,36,0,3,2.5,k;Thermal transfer: ",
str$(peek("opcua0:2:DeviceSet,3:Printer,3:Statistics,3:Thermal Transfer") / 1000), " m"
print "T4,40,0,3,2.5,k;Printhead model: ",
peek$("opcua0:2:DeviceSet,3:Printer,2:SubDevices,3:TPH 1,2:Model")
print "T4,44,0,3,2.5,k;Printhead serial number: ",
peek$("opcua0:2:DeviceSet,3:Printer,2:SubDevices,3:TPH 1,2:SerialNumber")
print "T45,8,0,3,2.5,k;Status: ",
peek$("opcua0:2:DeviceSet,3:Printer,3:Interpreter,3:ESCs")
print "T45,12,0,3,2.5,k;XStatus: ",
peek$("opcua0:2:DeviceSet,3:Printer,3:Interpreter,3:ESCz")
print "T45,20,0,5,2.5,k;Setup settings: "
print "T45,24,0,3,2.5,k;Print speed: ", str$(peek("opcua0:ns=4;s=ID_PRINT_SPEED")), " mm/s"
print "T45,28,0,3,2.5,k;Heat level: ", peek$("opcua0:ns=4;s=ID_HEAT_LEVEL")
print "T45,32,0,3,2.5,k;Print position X: ",
str$(peek("opcua0:ns=4;s=ID_PRINT_POSITION_X"))
print "T45,36,0,3,2.5,k;Print position Y: ",
str$(peek("opcua0:ns=4;s=ID_PRINT_POSITION_Y"))
print "A 1"
'
' Set print speed and heat level
poke("opcua0:ns=4;s=ID_PRINT_SPEED"), 150
poke("opcua0:ns=4;s=ID_HEAT_LEVEL"), "5"
</ABC>
```

In this chapter you will find a list of functions frequently used when you develop abc programs.

4.1 GetPrinterModel$

```
'*****************************************************************************
'*  Function:             GetPrinterModel$()
'*  Author:               DS 19/04/2007
'*  Description:          Get the model of printer (A3, A4+...)
'*  Parameters:           -
'*  Result:               string containing the printer model
'*  Changes:              -
'*****************************************************************************
sub GetPrinterModel$()
  local Var1$
'
  Var1$ = peek$("machine")
  Var1$ = mid$(Var1$, 0, instr(Var1$, "/"))
'
  return Var1$
end sub
'*****************************************************************************
```

4.2 GetCPUType$

```
'*****************************************************************************
'*  Function:             GetCPUType$()
'*  Author:               DS 19/04/2007
'*  Description:          Get the CPU type (Ax, M4, X2, X3 or X4)
'*  Parameters:           -
'*  Result:               one of the following string:
'*                        X4L (for MACH 4S, HQ, PXQ) (90° turned display)
'*                        X4 (for SQUIX, EOS2/5)
'*                        X3 (for EOS)
'*                        X2 (for A+, H+, Mach, PX, XC, XD)
'*                        Ax (for A-Series, Hermes A)
'*                        M4
'*  Changes:              03/04/2017 added X4 (for SQUIX-Series)
'*                        19/12/2017 added X4L (for MACH 4S)
'*****************************************************************************
sub GetCPUType$()
  local PrinterModel$
  PrinterModel$ = GetPrinterModel$()
'
  if (instr(PrinterModel$,"MACH 4S")) or (instr(PrinterModel$,"HERMES Q"))
      or (instr(PrinterModel$,"PX Q"))then
     return "X4L"
  elsif (instr(PrinterModel$,"SQUIX")) or (instr(PrinterModel$,"EOS2"))
         or (instr(PrinterModel$,"EOS5")) then
     return "X4"
  elsif instr(PrinterModel$,"EOS") then
     return "X3"
  elsif ((instr(PrinterModel$,"+")) or (instr(PrinterModel$,"Mach")) or
         (instr(PrinterModel$,"PX")) or (instr(PrinterModel$,"XD")) or
         (instr(PrinterModel$,"XC")) or (instr(PrinterModel$,"Hermes C"))) then
     return "X2"
  elsif (PrinterModel$ = "M4") then
    return "M4"
  else
    return "Ax"
  endif
end sub
'*****************************************************************************
```

4.3 OpenDisplay

```
<ABC>
'*****************************************************************************
'*  Function:              OpenDisplay()
'*  Author:                DS 22/03/2010
'*  Description:           open the printer display
'*  Parameters:            -
'*  Result:                Boolean true or false
'*  Changes:               -
'*****************************************************************************
sub OpenDisplay()
  if isDisplayOpened
    return true
'
  CPUType$ = GetCPUType$()
'
  'according to the CPU the screen size is not the same
  if CPUType$ = "M4" then
    return false
  elsif CPUType$ = "Ax" then
    open window 120,32
  elsif CPUType$ = "X2" then
    open window 128,64
  elsif CPUType$ = "X3" then
    open window 160,255
  elsif CPUType$ = "X4" then
    open window 272,480
  elsif CPUType$ = "X4L" then
    open window 480,272
  else
    return false
  endif
'
  poke "lcd", 1
  isDisplayOpened = true
  return true
end sub
'*****************************************************************************
</ABC>
```

4.4 ClearDisplay

```
'*****************************************************************************
'*  Function:              ClearDisplay()
'*  Author:                DS 22/03/2010
'*  Description:           clear the content of the display
'*  Parameters:            -
'*  Result:                Boolean true or false
'*  Changes:               -
'*****************************************************************************
sub ClearDisplay()
  if isDisplayOpened then
    clear window
    return true
  endif
'
  return false
end sub
'*****************************************************************************
```

4.5 CloseDisplay

```
'*************************************************************************
'*  Function:              CloseDisplay()
'*  Author:                DS 22/03/2010
'*  Description:           close the printer's display
'*  Parameters:            -
'*  Result:                Boolean true or false
'*  Changes:               -
'*************************************************************************
sub CloseDisplay()
  if isDisplayOpened then
    isDisplayOpened = false
    close window
    poke "lcd",0
    return true
  endif
'
  return false
end sub
'*************************************************************************
```

4.6 DisplayText$

```
'*************************************************************************
'*  Function:              DisplayText$
'*  Author:                DS 04/10/2007
'*  Description:           display 4 text lines at the printer display during a given time
'*                         This function uses other functions
'*  Parameters:            line1$: string  => first text line
'*                         line2$: string
'*                         line3$: string
'*                         line4$: string
'*                         DisplayTime: integer => wait time before the display is released
'*  Result:                -
'*  Changes:               -
'*************************************************************************
sub DisplayText$(line1$, line2$, line3$, line4$, DisplayTime)
  OpenDisplay()
  if CPUType$ = "M4"
    return
'
  ' if you are using Ax or X2 CPU, please delete the following two lines
  if CPUType$ = "X3" or CPUType$ = "X4" or CPUType$ = "X4L"
    font "Monospace,20"
'
  text 0, 0, line1$
  text 0, 16, line2$
'
  ' we can display more than 2 lines only on newer printers
  if CPUType$ <> "Ax" then
    text 0, 32, line3$
    text 0, 48, line4$
  endif
'
  wait DisplayTime
  CloseDisplay()
end sub
'*************************************************************************
```

4.7 CheckStatus

```
'*************************************************************************
'*  Function:              CheckStatus()
'*  Author:                DS 23/03/2010
'*  Description:           check the printer status
'*  Parameters:            -
'*  Result:                Boolean true or false if the printer is in error
'*  Changes:               -
'*************************************************************************
sub CheckStatus()
  'no error or printhead opened without print job
  if (mid$(peek$("status"), 2, 1) = "-") or (mid$(peek$("status"), 1, 2) = "YD000000N") then
    return true
  else
    return false
  endif
end sub
'*************************************************************************
```

4.8 PromptText$

```
cstEnter        = 13
cstEscape       = 27
cst0            = 48
cst9            = 57
cstA            = 65
cstZ            = 90
cstaa           = 97
cstzz           = 122
cstEnd          = 61453
cstArrowLeft    = 61472
cstArrowRight   = 61473
cstArrowUp      = 61474
cstArrowDown    = 61475
cstNumeric      = 0
cstAlpha        = 1
cstAlphanum     = 3
'
' on X3 CPU, some keyboard codes are not the same
GetCPUType$()
if CPUType$ = "X3" then
  cstBackspace  = 8
else
  cstBackspace  = 61449
'
'
'*****************************************************************************
'  Function:              PromptText$
'  Author:                DS 07/03/2005
'  Description:           display a text on the printer's display, prompt a default value,
'                         limit the user input and returns the text entered by the user
'                         This function uses other functions like GetCPUType$
'  Parameters:            text1$: string => first line on display
'                         text2$: string => second line
'                         test3$: string => third line
'                         text4$: string => fourth line
'                         length: integer => max input length
'                         chartype: string => type of char the user can type in (numeric, alpha or alphanum)
'  Result:                -
'  Changes:               03/04/2017 added support for SQUIX-Series
'*****************************************************************************
sub PromptText$(text1$, text2$, text3$, text4$, length, chartype)
  local Var1$, x, charvalid
'
  open 1,"/dev/keyboard","r"
  OpenDisplay()
  ClearDisplay()
'
  ' if you are using Ax or X2 CPU, please delete the following two lines
  if CPUType$ = "X3" or CPUType$ = "X4"
    font "Monospace,20"
'
   text 0,0,text1$
   text 0,16,text2$
'
  ' we can display more than 2 lines only on newer printers
  if CPUType$ <> "Ax" and CPUType$ <> "M4" then
    text 0,32,text3$
    text 0,48,text4$
    Var1$ = text4$
```

```
  else
    Var1$ = text2$
  endif
'
 do
    do
      x = peek(#1)
      if x <> -1
        break
      wait 0.1
    loop
'
    'enter pressed => quit the loop
    if x = cstEnter
      break
'
    switch x
      'escape pressed => quit the program
      case cstEscape:
      case cstEnd:
        poke "lcd",0
        close 1
        return chr$(cstEscape)
        break
'
      'backspace pressed => we delete the last char
      case cstBackspace:
        Var1$ = left$(Var1$, len(Var1$) - 1)
        clear window
        text 0,0,text1$
'
        ' we can display more than 2 lines only on newer printers
        if CPUType$ <> "Ax" and CPUType$ <> "M4" then
          if (text3$ <> "") then
            text 0,16,text2$
            text 0,32,text3$
            text 0,48,Var1$
          else
            text 0,16,text2$
            text 0,32,Var1$
          endif
        else
          text 0,16,Var1$
        endif
        :break
'
      case cstArrowUp:
      case cstArrowDown:
      case cstArrowLeft:
      case cstArrowRight:
        break
'
      'another key was pressed => display the text
      default:
        if (len(Var1$) < length) then
          charvalid = false
'
```

```
          switch chartype
            case cstNumeric
              if (x >= cst0) and (x <= cst9)
                charvalid = true
              break
            case cstAlpha
              if ((x >= cstA) and (x <= cstZ)) or ((x >= cstaa) and (x <= cstzz))
                charvalid = true
              break
            default
              charvalid = true
              break
          end switch
'
          if charvalid then
            Var1$ = Var1$ + chr$(x)
'
            ' we can display more than 2 lines only on newer printers
            if CPUType$ <> "Ax" and CPUType$ <> "M4" then
              if (text3$ <> "") then
                text 0,16,text2$
                text 0,32,text3$
                text 0,48,Var1$
              else
                text 0,16,text2$
                text 0,32,Var1$
              endif
            else
              text 0,16,Var1$
            endif
          endif
        endif
        break
    end switch
    wait 0.1
  loop
'
  close 1
  CloseDisplay()
  return Var1$
end sub
'*****************************************************************************
```

Germany
cab Produkttechnik GmbH & Co KG
Karlsruhe
Tel. +49 721 6626 0
www.cab.de

France
cab Technologies S.à.r.l.
Niedermodern
Tel. +33 388 722501
www.cab.de/fr

USA
cab Technology, Inc.
Chelmsford, MA
Tel. +1 978 250 8321
www.cab.de/us

Mexico
cab Technology, Inc.
Juárez
Tel. +52 656 682 4301
www.cab.de/es

Taiwan
cab Technology Co., Ltd.
Taipei
Tel. +886 (02) 8227 3966
www.cab.de/tw

China
cab (Shanghai) Trading Co., Ltd.
Shanghai
Tel. +86 (021) 6236 3161
www.cab.de/cn

Singapore
cab Singapore Pte. Ltd.
Singapore
Tel. +65 6931 9099
www.cab.de/en

South Africa
cab Technology (Pty) Ltd.
Randburg
Tel. +27 11 886 3580
www.cab.de/za

cab // **820** distribution and service partners in more than **80** countries

we identify more

© cab