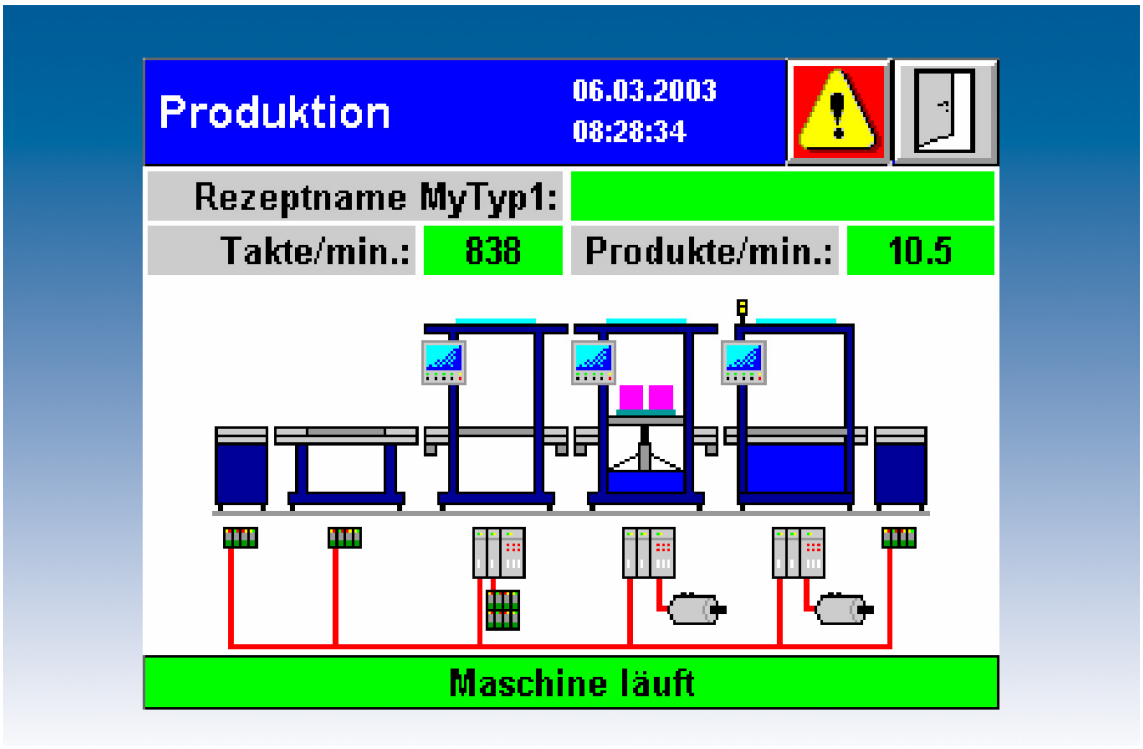
User Manual

Easy PageMachine (EPAM) V 3.40

Software



Document M001927-03
Edition 07/2009

micro innovation
Innovation in Automation

Manufacturer

Product Company
Micro Innovation AG
Spinnereistrasse 8-14
CH-9008 St. Gallen
Switzerland

Tel. +41 (0) 71 243 24 24
Fax +41 (0) 71 243 24 90
info@microinnovation.com
www.microinnovation.com

Sales Company
Micro Innovation GmbH
Nideggerstrasse 6-10
53115 Bonn
Germany

Tel. +49 (0) 228 602 2020
Fax +49 (0) 228 602 1713
info@microinnovation.com
www.microinnovation.com

Service/Repair Center
Micro Innovation GmbH
Carl-Benz-Strasse 19
78224 Singen
Germany

Tel. +49 (0) 7731 7896 110
Fax +49 (0) 7731 7896 101
servicecenter@microinnovation.com
www.microinnovation.com

Original language

German

Redaction

G.Fischbacher

Brand and product names

All brand and product names are trademarks or registered trademarks of the owner concerned.

Copyright

Subject to modifications

Symbols for warning texts

Warning of general hazard

Warning of electrical voltage

Components susceptible to damage from electrostatic charges. The opening of the housing or connections should only be carried out by trained personnel!

Notes

Contents

Page

1 Introduction

The ***Easy PageMachine*** (***EPAM***) visualization tool is specially designed for graphical operator guidance with touch panels, and enables ***visualization parameters to be configured without any extensive programming work required.*** EPAM provides objects such as buttons, switches, alphanumeric variables, bars, messages etc. for creating individual screen masks. These objects are configured in a structured ASCII file, linked with the PLC variables and combined to form complete screen pages. The different screen pages are combined together with links and can, for example, be called by clicking a button object.
The tabular and straightforward structure of this ASCII file (script file) enables project creation with a typical spreadsheet program such as Excel or similar. The transparent data format also enables the project documentation to be created virtually automatically.
EPAM also features an interpreter which allows the application to be tested on the PC it was created on. In this case, the visualization functions can be activated using the mouse. Modifications can thus be carried out and tested in seconds. The application is then loaded into the target system.

Requirements of the development system:

- IBM-compatible PC
- Windows 2000/XP/Vista
- Excel 2000 or newer
- Graphics program for creating images in PCX format, e.g. Paint Shop Pro (Windows demo version on CD), s.a. http://www.jasc.com
- FontBuilder/FontWindow for creating user-defined fonts. s.a. http://www.metagraphics.com

Target system requirements:

- Devices EP-300-Series (WindowsCE)
- Devices CPC-300/600-Series (WindowsCE)
- Devices SP-200 Series (WindowsCE)
- Devices HPG-200, 300 Series (VxWorks)
- Devices XVC-600, XCC-600 Series (VxWorks)
- Devices XV-100,XV-200,XVH-3xx/XV-4xx-XVM-400-Series (WindowsCE)
- PC with Windows2000/XP
- PocketPC with WindowCE (ARM Processor, Pocket Windows 2002/2003, e.g. HP iPAQ 5450)

1.1 New features of EPAM V3.40

V3.40

- New WindowsCE Targets: SP-200, EP300-07, XV-100, XVM-400, KeTop50
- Support Keyboardinput (Button-Option: Key=)
- PrintScreen Support for WindowsCE
- WebEPAM TrueColor Support (epamview.jar V1.0.04)
- Support DNS Settings via EPAM
- RAM-Drive increased for EP-300 (16MB), CPC-300/650 (32MB)
- New communiaction driver for ELAU- (MAX4, C-Series, P-Series), WAGO- (750-841), Parker-PLC (C3-Series), UDP (ASCII)

V3.30 SP2

- RemoteClient V1.0.2
- Support AT-S7 PLC control-functions: STOP, RUN, Reset,... (drvrs7.dll V1.4.2, rs7dll.dll V3.2.0)
- Support Windows Vista and Office 2007 (epamcom.dll V1.1.1 skip.exe V1.2.7)

V3.30 SP1

- New WindowsCE Target: XVH-2xx.
- Support UserColors

- Support online unit-system change (e.g. mm/inch).
- New communication driver for Siemens-MPI (requires ProfibusDP-Slave/MPI Option), AT-S7 (Step7 compatible Soft-PLC), Multiprog-PDD.
- New address column in UserVar for Siemens-MPI, AT-S7
- Support CoDeSys PLC control functions (STOP, RUN, Reset,...)
- Button Object output performance optimized
- New Option "Type=" for Alarm, Trend and Datalog Object for use of the same definition files for multiple objects.
- New macro "Project compare" for comparison of two EPAM projects
- Support for images with 16 Million colours (24 Bit) for WindowsCE/Windows

1.2 New features of EPAM V3.30

- New WindowsCE Targets: XVH-3xx, XV-4xx, EP-300, CPC-300, CPC-600, PocketPC
- New Windows2000/XP-Targets: PC (WinEPAM)
- Support 19“ Displays 1280x1024
- New object: DropDownList for selection of list elements
- New object: HTML-Browser to display HTML-pages (only Windows)
- New objects for database-connection: DBPasswd, DBTracer (only VxWorks, s.a. EPAM-DB-Extension)
- RemoteControl-object: supports login with password
- Recipe-object: support of cascading recipes (recipe1 loads recipe2)
- Recipe-object: new command save recipe from PLC
- Page-object: support of relative window positions
- Passwort-object: new option „SysPW=off“ deactivates the date based master password
- Passwort-object: new option „Bitwise=AND“ allows more flexible authorization handling
- Variablen-object: new option „CoseIfOk“ closes automatically the keyboard
- Recipelist-object: saves cursor position analog to textlist
- Signal-object: transparent background color
- Variable-object: transparent background color
- Trend-object: advanced functions for trend display (scale on/off, scale color, Y-lines, Datalog column)
- New button-action „PrintScreen“ (only Windows2000/XP) for default-printer
- New button-action: Reboot to restart the system
- New button-action: FileCopy(dst=path\file.ext src=path\file.ext)
- New button-action: EjectVolume(Drive #Page=eject_failed #Page=eject_ok) to check out removeable devices e.g. USB-Memorysticks (only Windows)
- New limit-actions: s_myvar=x, language=x, language=s_myvar
- Improved communication error handling (SymArti)
- Languages can be loaded individually at language change (EPAM.INI: LOAD_LANGUAGE=1) into RAMDrive (Default: load all languages into RAMDrive)
- EPAM-Wizard: Copy/Paste/Delete functions with Ctrl-C/V and Delete-keys
- New commandline-options for WinEPAM (Windowname, X,Y-Position)
- New system variable: s_alarm_tin_dt, s_alarm_tout_dt, s_alarm_tquit_dt for flexible formatting of Alarminfo
- New system variable s_myrecipe_cur_file and s_myrecipe_cur_name shows the current selected recipe within the recipe list
- New system variable: s_plcstate_<hostname> to display the state of (Remote)-controls
- Within column „Color“ it is possible to select color numbers (0 – 255)
- New environment variable USERCOLOR=Yes or =Image.pcx (Epam.ini) for support of custom defined color palettes (256 colors).
- New environment variable DRIVER=No (Epam.ini) to deactivate of communication on target
- New environment variable Kbd=off (Epam.ini) to suppress keyboard inputs
- Support of multi SymArti-connections for WindowsCE
- Access to RemoteServer with mode display only (no input)
- Release Mode EPAM_NOEXIT=yes (Epam.ini) avoids to exit EPAM also with keyboard (key “ESC”)

- Macro download checks also language dependent textfiles for textlist and alarm helpfiles and also images and icons; missing files will be displayed in a listbox

1.3 New features of EPAM V3.20

- For the Installation of Version 3.20 a product code is required. Without a product code the demo version will be installed. Applications build with the demo version, will terminate after of 1 hour!
- New Targets: MC-HPG200/300, MC-HPG200 Portrait (240x320), XVC600, XCC600
- Support for devics with resistive touch-screen (including calibration)
- Support of „Overlayed objects“ e.g. Button with LED, Bargraph with Value, etc.. This means objects can overlay each other and will than automatically updated. (see Demo „Overlayed objects”)
- Support of object on/off on background images (background will be restored)
- New macro „Zoom-Project“ for an easy convertion of projects for different screen resolutions
- New macro „Build Recipes“ to create recipe file within EXCEL
- Support to download recipes which are built within EXCEL
- Download settings will be stored within a project
- Download of project in Release-Mode with Ramdrive without reboot of the device
- Multiple page definitions will be checked
- Improved error report list in EXCEL
- User defined group settings in EXCEL will be stored
- Performance optimization of EXCEL macros
- EPAM-Wizard expanded for up to 700 objects/page . An error message will be displayed if this limit exceeds.
- New system variable s_recipelist_empty, can be used to remove “Load” button in a recipe list page (if recipe list is empty)
- New system variable s_toucherror to display touch errors (IR touch-screen only)
- New system variable s_irtouch for identification of devices with IR touch-screen
- New system variable s_remoteclient_connected to display a remote access
- New parameter Retry and DelayOnError in worksheet Hosts for networked devices
- Support for country specific keyboards and Windows charsets (e.g. kyrillic, etc.)
- New Button-Action „Close=Pagename“ closes window „Pagename“
- New Formats for Datalog for automatically definition of field width within the Logfile
- New option pos=left/right/center for objects Button, Switch, Radiobutton
- Diagnose-Signal with Limit-Actions: for change of alarm state from active -> inactive, the Limit1-Action will be performed, for change of alarm state from inactive -> active, the Limit2-Action will be performed
- New option DX=0 to display alarm list without a scroll bar. New option Coff to switch off the cursor.

1.4 New features of EPAM V3.10

- Easier, automatic Installation of EPAM-macros
- Windows-Version WinEPAM for Simulation under EXCEL incl. communication to the PLC
- New Fonthandling: Fonts can be defined within EPAM (macro „New Font“) and will be created automatically with macro „Build Fonts“ (s.a. chap 5, P.29)
- Display of the used fonts also in EPAM-Wizard (Option: Map Fonts)
- New Object: Sys2PLC for data exchange of system variables to the PLC (s.a. chap. 7.24 P.124)
- New Object: RemoteControl to control EPAM-Applications remote other the network (s.a. chap. 7.25 P.126)
- Support of transparent Images (PCX-Images): any color within a Image can be defines as a transparent color
- Textlist with formated flowtext: italic, bold and underline
- improved language support: Texts can be defined within the worksheet „Text“ for all used languages and will be assigned automatically in the project sheets with the macro „Build Language Texts“
- improved variable import for communication to PLCs within a network. It is possible to import variables of different PLCs (with Definition of different Hostnames and IP-Addresses for each PLC)

- Images and fonts can be placed in different directories, independent and separated from the project (environment variables: PATH_IMG=, PATH_FNT=)
- Global objects will be automatically inserted within the Initpage. The Initpage will be defined automatically.
- Display of the current PLC-state (Stop/RUN) with the systemvariable s_plcstate
- Export of the Alarmhistory as CSV (Action: AlarmExport=CSV)
- Support of Datantype IEC_DT (Date/Time-input, Timerfunctions)
- New option „Type=Password“ for invisible input of passwords with any font
- New system variable s_newpage, s_pageidx, s_pagename to change pages direct from the PLC or to display the current page in the PLC (in combination with object Sys2PLC)
- improved project download: Test of target connection, Target-ID and Diskfull
- Macro „Rebuild all“ tests if all used pages are defined

1.5 Notes to upgrade existing applications

1.5.1 Changes in EPAM V3.40

Action #PagePrev works ne was following:
Page changes to pages with the same dimension will be stored in a stack (ring buffer with the last 100 pages; the first page will not be overwritten, so it is always possible to move back to the first page). With #PagePrev it is possible to move back within this stack. #PrevPage can not be used for Pages with different size (Windows). Action Close or Close=Windowname has to be used for that.

1.5.2 Changes in EPAM V3.30

Notes for devices with WindowsCE-operating system
Directory-structure on WindowsCE devices:

- the name of the CompactFlash-card is „StorageCard“ (not „C:“)
- all EPAM specific files are within the directory \StorageCard\EPAM\
- the EPAM directory contains the following subdirectories:
  BACKUP …Backup directory (*.INI and *.DAT files)
  DATA …Data directory (*.DAT)
  FNT …Fonts (*.TTF)
  INI …*.INI files
  IMG …Images (*.PCX, *.ICO) optional
  PROJECT …EPAM project-files

Fonts
On WindowsCE full standard Windows True Type Fonts (*.TTF) will be used instead of language dependent fonts which contains only the Unicode-characters used within the project! So normally more space will be required for these font files (TTF, e.g. Arial Unicode MS → ca. 24MB!) → it may be necessary to use larger CF-cards or the Windows TTF-Fonts have to be modified manually with a standard Windows-Fonteditor to fit project specific requirement. (not recommended)
Fonts (*.TTF) will be stored global in directory FNT and download is optionally (Option: Download Image/Fonts). So fonts must not be downloaded every time. Bold Fonts will be displayed wider under WindowsCE as on Desktop. This is a „characteristic“ of WindowsCE. To get same results on the development PC and on the target the Fontstyle „SemiBold“ will be used.
Because of this differences there may be marginal divergence or modifications could be necessary when porting an application from VxWorks (HPG-200/300 XVC-600) to WindowsCE. If font modifications are necessary this can be done easily within the worksheet „Fontmap“.

RAM-Drive
The EPAM-Runtime System will be copied into directory \EPAM after power up and then started from there. So it is possible to update files on Compact Flash during operation of EPAM. (Note: Access to open files is not possible with WindowsCE)
If the option „Ramdrive“ is active, also the project will be copied to this directory.

Message-object
If the Message-Object is configured to display a variable value within the message, the message number must be a DWORD (32 Bit-Variable) on all WindowsCE devices!

RemoteControl
The EPAM-RemoteControl-Object supports at the moment only 256 colors (8Bit/Pixel). If an access is done to another WindowsCE device the RemoteControl-Server on this device should run in 256 color-mode or support this mode. The same is for the Password-functionality which also must be supported by the Server.

Unsigned Datatypes

Until now an overflow of an unsigned datatype (e.g. BYTE) from 0 to 255 (value = 0 and action SetVar-1) was not detected! (s.a. #bug175) So the limit-action of a defined limit1 of 0 was not performed. This error has been solved. So for value 0 and action SetVar=-1 the limit action1 will be performed now or if no limit-action is defined the value will remain 0.
If the overflow from 0 to 255 on SetVar-1 is wanted, the limit1-action SetVar=Limit2 must be defined!

Release Mode
Release Mode EPAM_NOEXIT=yes (Epam.ini) avoids to exit EPAM also with keyboard (key “ESC”) and with the pushbutton on XV-3xx/4xx devices!

1.5.3 Changes in EPAM V3.20

For the new function „overlayed objects“ it is necessary to limit the screen-output of an object to the object dimension DX, DY (clipping). Until V3.10 a PCX image within a signal object for example was displayed completely on screen even if the object dimension DX, DY was smaller than the width and height of the PCX image.
Now only the part of the PCX image within the object dimension DX, DY will be displayed. So it is possible that existing projects have to be modified.

The operation of the screen saver was modified (VarState and VarValue) and works now as documented.
For the bargraph object it is possible to define limits e.g. 0, 100 and display the bargraph with the option Fill=X. Now the bar will be filled from the middle to the left (values 50 to 0) and right (values 50 to 100).
The dimension of the meter-object was modified so that it is similar for Windows and on the target. The animation of Button, Switch, Radiobutton was modified. The interior of the Button will no longer be moved.

In the alarm history no longer the oldest alarm will be overwritten. New alarms are inserted in the following way:

1. if the oldest alarm is inactive, it will be overwritten by the new one
2. if the oldest alarm is active, the oldest inactive acknowledged alarm will be overwritten
3. if there is no inactive acknowledged alarm, the oldest inactive alarm will be overwritten
4. if there is no inactive alarm, the oldest alarm will be overwritten. (in this case alarm will be lost, that means the alarm is no longer displayed in the alarm list -> more than 512 active alarms!)

On devices with Infrared-touch screen a touch test will be performed on power up. If an error occurs a page with the bad light barriers will be displayed. (graticule). After a timeout the application will be started and the system variable s_toucherror will be set to indicate the touch error. (s.a. section system variables)

1.5.4 Changes in EPAM V3.10

Existing projects can be updated with the macro „Update Objects“. With “Update Objects” a new sheet „Fontmap“ will be inserted automatically. This sheet contains a number of predefined font definitions. If your application uses other fonts which are not already defined, than you have to define this fonts with the macro „New Font“. After this the project should be rebuilt with the macro “Rebuild all”.

Invisible password inputs with password font have to be defined with the new option „Type=Password“.

Improved language support:
The macro „Build language texts“ and the option „Insert undefined text“ inserts all existing texts into the worksheet „Text“. After this all texts in all languages of an application can be modified within this single worksheet. Changes inside the worksheet „Text“ can then be actualized in the project sheets with „Build language texts“ (option „Insert undefined text“ is inactive). In this case to all default texts in the whole project the corresponding language texts will be assigned within the language columns automatically.

2 Installation

2.1 Easy PageMachine EPAM

Insert the CD and select EPAM in the menu. EPAM will then be installed on the specified drive and path (current versions of EPAM are available via the Internet from www.microinnovation.com).
After the installation has been successfully completed, the following files and directories will be present:

EPAM\

FontBuilder-Unicode	...utility for creating user-defined fonts (only VxWorks)
Images	...contains PCX images
Samples	...contains EPAM demo projects
Target	...current runtime system (WinEPAM and PocketPC)

An executable demo application as well as a readme.txt file with up-to-date information is provided in the EPAM program group.

2.2 Settings in Excel

The following descriptions refer to Excel 2000. This is recommended for working with EPAM.

2.2.1 Installing EPAM macros

The EPAM macros will be installed in the directory \...\Microsoft Office\Office\XLStart and then automatically started with Excel.

2.2.2 Toolbars

In Excel2000, 2003 it is possible to configure the EPAM-Toolbars without any limits. In Excel 2007 this possibilities are limited.

With option quick acces for Toolbars it is possible to add the EPAM-Toolbars to the quick access menu. Select command „Add Ins“ and add the toolbar to quick access.
The order oft he EPAM-Toolbars can be changed by deleting the toolbar and exit/restart EXCEL. The last deleted toolbar will be at the end after restart of EXCEL.

Note!

It is recommended to save EPAM-Projects in XLS-Format (97-2003). Saving in standard new XLSX-Format (Excel 2007) reduces performance during project development (performance of macros is reduced).

2.3 Paint Shop Pro graphics program

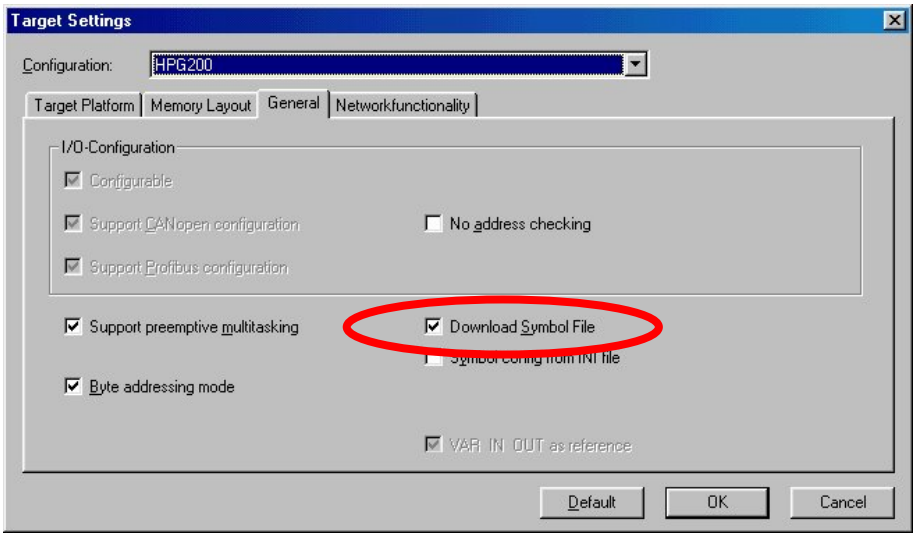
To install the **Paint Shop Pro** graphics program, select Paint Shop Pro on the CD. Remember that the PSP program is a Windows demo version with a limited period of validity.

2.4 Settings in the CoDeSys development environment

The following settings need to be made in the CoDeSys development environment in order to establish communication with the PLC runtime system.

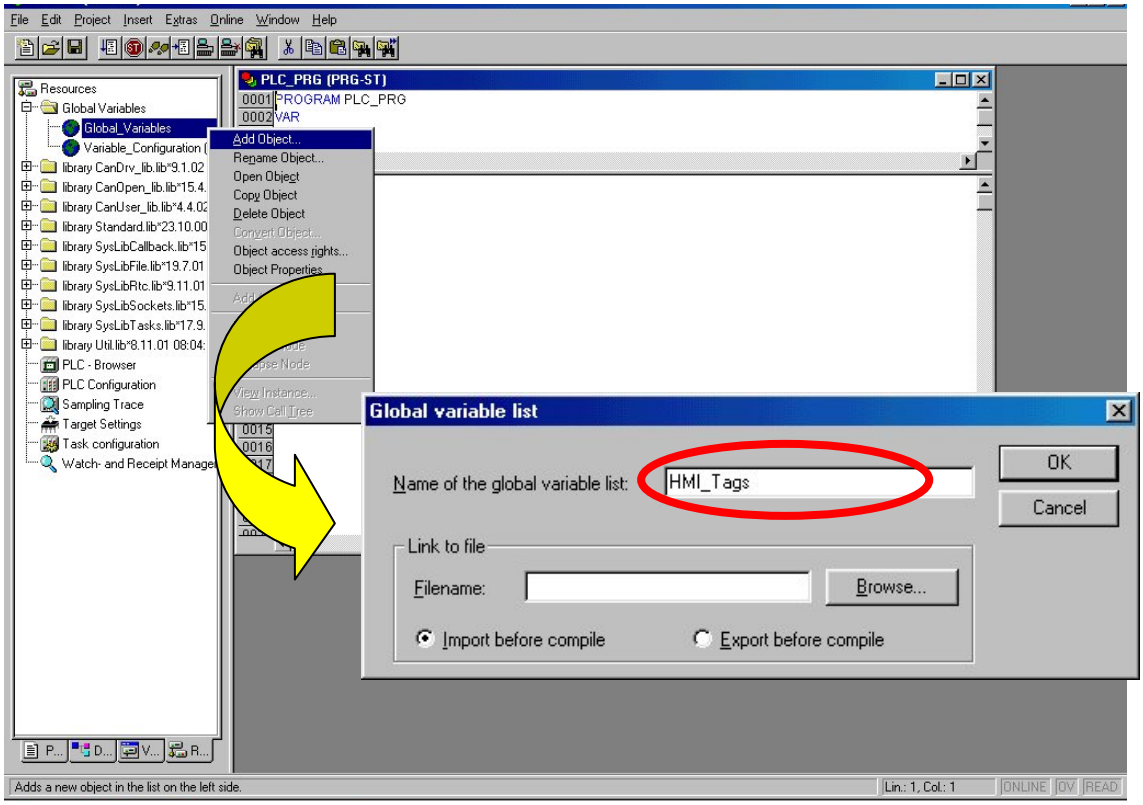
2.4.1 Target system settings

Select the appropriate target system, e.g. HPG-200 and activate the Download Symbol File check box.

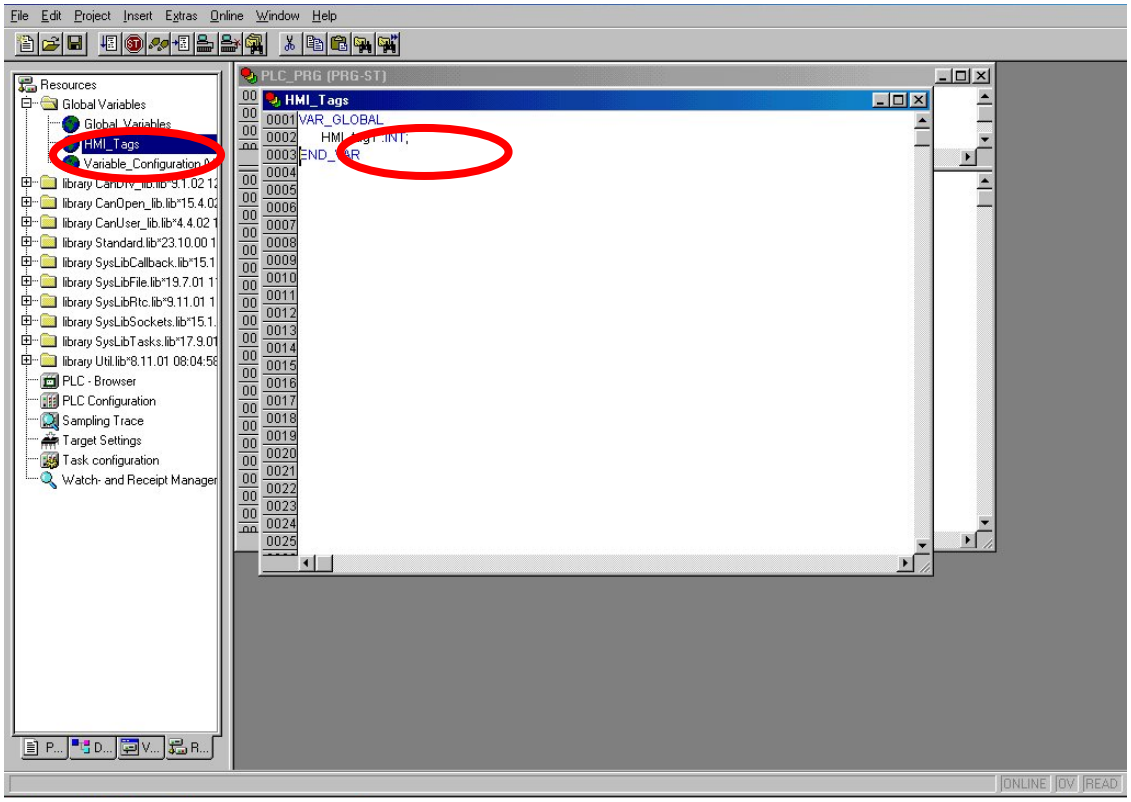


2.4.2 Defining global HMI variables

For performance reasons we recommend that only the global variables required for variable exchange with the visualization system are exported to the symbol file. A separate area should therefore be created for the global HMI variables via Resources - Global Variables - Add Object (right mouse button).

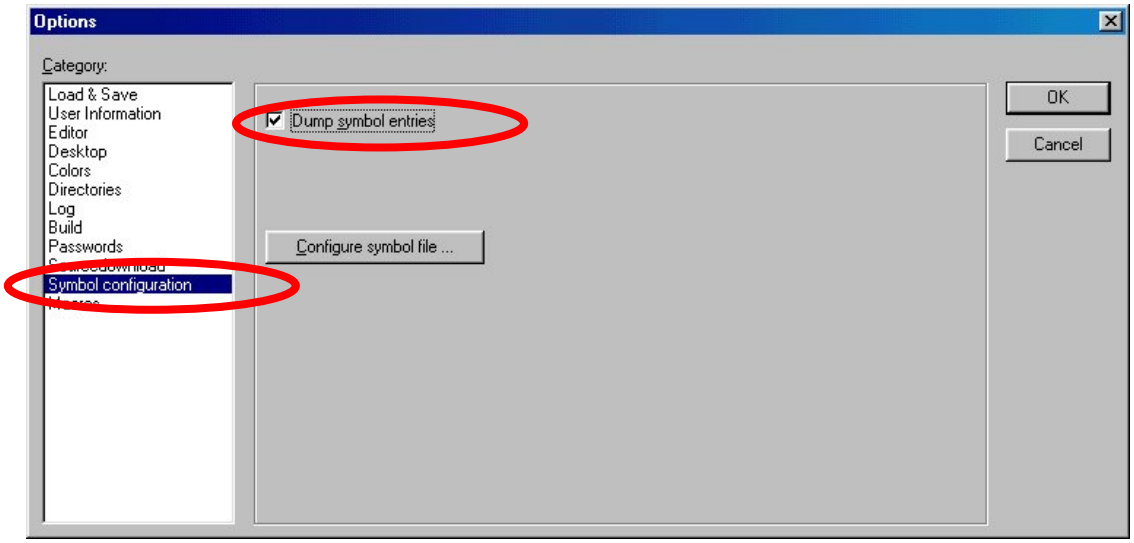


The global HMI tags can then be defined in the HMI Tags worksheet.



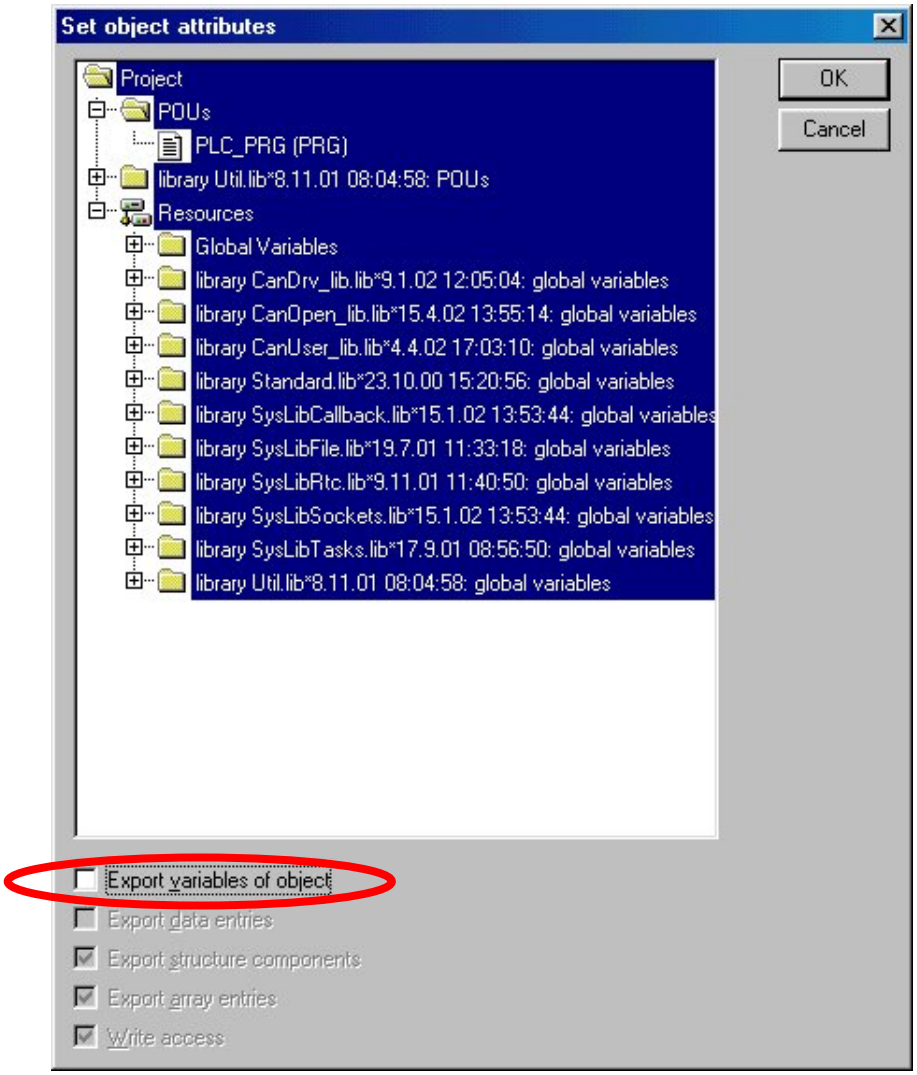
2.4.3 Project options

In order for the symbol file to be created, the Dump Symbol Entries check box must be activated via Project - Option - Symbol Configuration.

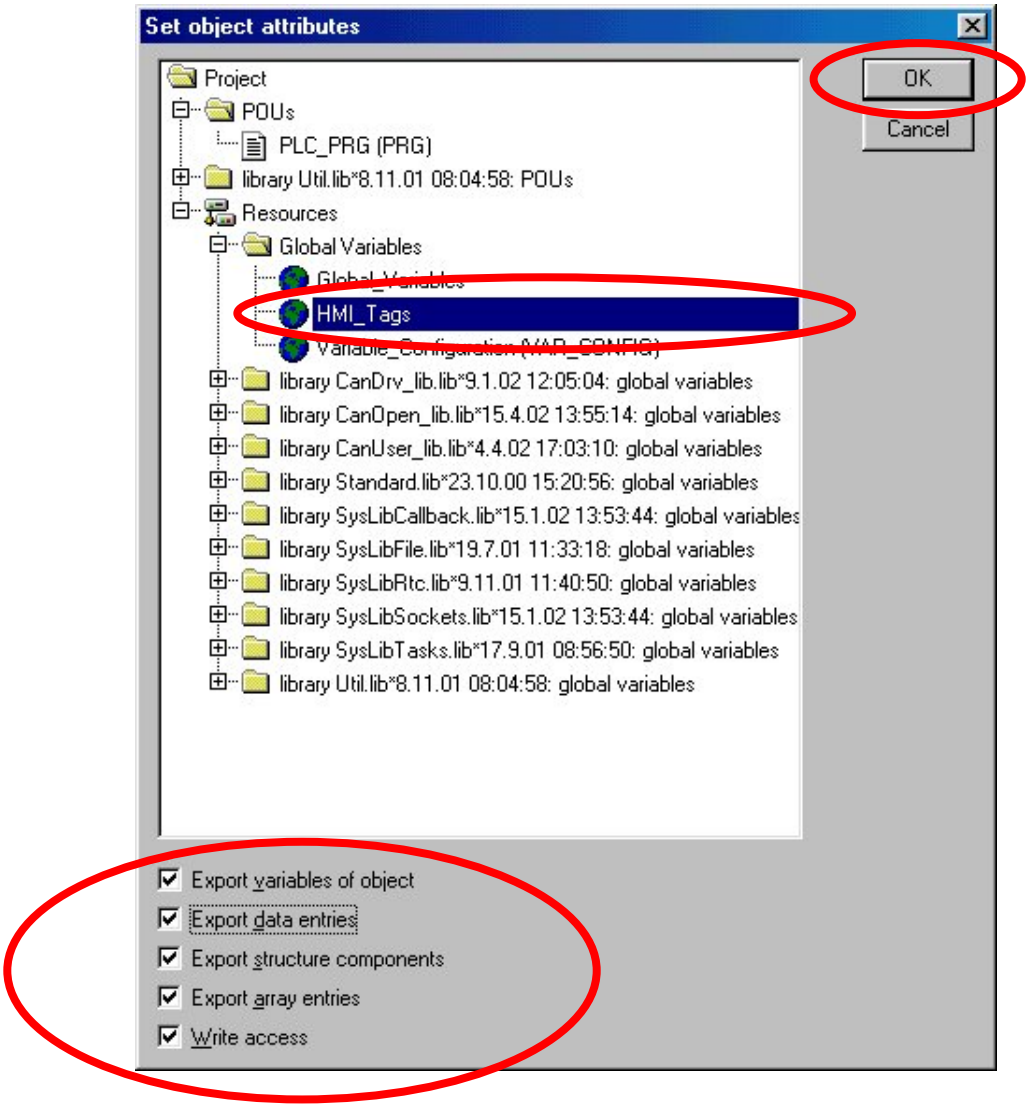


Note!
If the simulation function is activated in the Online menu, the Symbol Configuration entry will not be shown in Project - Options dialog ⇒ Deactivate the simulation function in the Online menu.

The Configure Symbol File function allows you to set which variables are to be exported to the symbol file. All objects are selected by default. Deactivate the Export Variables of Object check box.



Then select Global Variables - HMI Tags, activate the required options and click OK to close the menu.



Note!
The symbol file is not created every time there is a change in the PLC project. The project should therefore be re-created with every change in the HMI tags using the Clean All, Rebuild All function.

2.5 EPAM Runtime-System

The EPAM Runtime-System is normally already installed on the CompactFlash of the device. Following the most important features of the different RTS are described:

2.5.1 Runtime-System for devices with VxWorks

On devices with VxWorks (HPG-200/300, XVC/XCC-601) the CompactFlash is named as drive „C:“
Directory structure drive „C:“

C:\Backup	...EPAM Backup-directory (copy of INI-Files and recipes)
C:\Data	...EPAM Data- directory (recipes and Datalog-files)
C:\EPAM	...EPAM RTS and project
*_DNLD_*	...temporary Download- directory (will be removed after download)
\Project	...EPAM project
epam.out	...EPAM RTS
restore.out	...Utility to restore lost INI and recipe files
drvarti.out	...SymARTI-driver
rcs.out	...RemoteControl-Server
boxpc.out	…Display-configuration for XCC/XVC-601
EPAM.INI	…EPAM-settings (s.a. worksheet EPAM)
C:\INI	...EPAM INI-Files (sysvar.ini, alarm.ini)
Autoexec.INI	...Start EPAM-RTS and RemoteControl-Server
Config.ini	…will be used in EXCEL to check the connection during project-download
„Target“.SYS	...e.g. HPG200.sys will be used in EXCEL to check the target

The complete Runtime-System can be installed with the program „SetupTargetFirmware-Vx.x.exe“.

2.5.2 Runtime-System for devices with WindowsCE

On devices with WindowsCE (XVH-300, XV-4xx, EP-300, CPC-650) the CompactFlash is named as „StorageCard“
Directory structure „StorageCard“

StorageCard\EPAM\Backup	...EPAM Backup- directory (copy of INI-files and recipes)
StorageCard\EPAM\Data	...EPAM Data- directory (recipes and Datalog-files)
StorageCard\EPAM\FNT	...EPAM project specific Windows-TrueType Fonts
StorageCard\EPAM\IMG	…EPAM project specific images (opional, if PATH_IMG= in EPAM.INI is defined)
StorageCard\EPAM\INI	...EPAM INI-Files (sysvar.ini, alarm.ini)
StorageCard\EPAM	...EPAM RTS and project
*_DNLD_*	...temporary download-directory (will be removed after download)
\Project	...EPAM project
wceepam.exe	...EPAM RTS
drvarti.dll	...SymARTI-driver
cesysutl.dll	…HW-specific functions (e.g. Backlight, IP-Address, etc.)
EPAM.INI	…EPAM- settings (s.a. worksheet EPAM)
HMI.BAT	...Start EPAM-RTS

The complete Runtime-System can be installed with the program „SetupTargetFirmware-Vx.x.exe“.

The Windows Fonts (*.TTF) will be stored global in directory EPAM\FNT. So it is not necessary to download fonts every time. (s.a. Option: Download Image/Fonts)

Limitations of the WindowsCE Version
AlarmMail object is not supported.

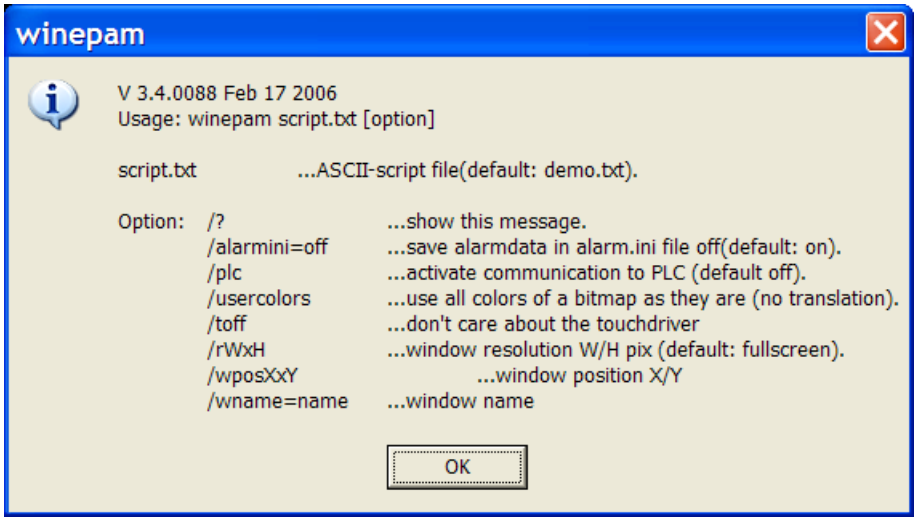
2.5.3 WinEPAM Runtime-System for PC/IPC

The WinEPAM Runtime-System consists of the following files:

Winepam.exe	...EPAM RTS
Drvarti.dll	...SymARTI-driver
Drvrs7.dll.dll	...AT-S7-driver
Drvmpi.dll	…MPI-driver (needs a Hilscher DP card)

These files are within the EPAM-directory on the development-PC. Project download is only possible to a directory. Default a subdirectory „Target“ will be used within the EPAM-project directory. Within this directory all necessary project files will be stored. (without winepam.exe and drvarti.dll).
Also a desktop link with the WinEPAM.exe call will be created. To install an EPAM-project on a PC/IPC the Target- directory, WinEPAM RTS and the destop link has to be copied. The link and the path-settings in EPAM.INI have to be modified if the directory structure is different on the PC/IPC.

WinEPAM commandline-parameter



WinEPAM should be started without the project file (script.txt) to use the settings from EPAM.INI file. With project file script.txt EPAM runs in simulation mode. (same as camera in EXCEL)

WinEPAM can be started multiple on a PC, but the window-name (/wname=) must be different for each window!

For the PC runtime-system a runtime licence is required!
Name of the item: WinEPAM Runtime-licence standard-PC

Limitations of the Windows Version
Systemfunctions like Touch-Beep, Backlight-setting, IP-address display/change are not supported in the Windows version. Communication to the control is done via TCP/IP, so a PC with Ethernet is required.
AlarmMail object is not supported.

2.5.4 Runtime-System for PocketPC (Target PocketPC-240x320)

The Runtime-System for PocketPC (e.g. iPAQ with PocketPC2002) is developed for PDAs with ARM-Processor and WindowsCE (PocketPC2002). The RTS has been tested on a HP-iPAQ 5450 with XScale Processor (PXA270), WindowsCE 4.2 (PocketPC2002) and WLAN.

In principle the RTS should run on every PDA with ARM and WindowsCE. But this can not be guaranteed and has to be tested for each individual case!

Runtime-System for PocketPC:

wceepam.exe	...EPAM RTS
drvarti.dll	...SymARTI-driver
cesysutl.dll	…HW-specific functions (dummy, not supported)

These files are within the EPAM-directory Target\PocketPC2002 on the development-PC. Project download is only possible to a directory. Default a subdirectory „Target“ will be used within the EPAM-project directory. Within this directory all necessary project files will be stored. (without RTS).
For the download of an EPAM-project and the RTS to a PocketPC it is recommended to use Microsoft Active Sync. Copy the Runtime-System with Windows-Exporer into the directory which is synchronized with the PocketPC (e.g. My files\My PocketPC Documents\EPAM). Afterwards select this directory also for the project-download in EXCEL and then synchronize with the PocketPC. (for that activate Microsoft Active Sync-Options-Sync Options Files)

On the PocketPC start EPAM by calling wceepam.exe within “My Documents\EPAM”.

For the PocketPC runtime-system a runtime licence is required!
Name of the item: EPAM Runtime-licence PocketPC

Limitations of the Pocket-Windows Version
System functions like Touch-Beep, Backlight-setting, IP-address display/change are not supported in the Pocket-PC version. Communication to the control is done via TCP/IP, so a Pocket-PC with Ethernet or WLAN is required. AlarmMail object is not supported.

2.5.5 EPAM-configuration on the target (EPAM.INI)

The EPAM-RTS is configured with EPAM.INI file (worksheet EPAM). Normally the settings will be made automatically in EXCEL. But is also possible to make the configuration manually within this file on the target. (Note! The file will be overwritten in case of a project-download!)

[ENVIRONMENT]	Comment
EPAM_VARLIST=_DRVVLST.TXT	Filename for Variable-List
EPAM_DRVPARAM=DRVPARAM.TXT	Filename for Driver Parameter
EPAM_PROJECT=Project.TXT	EPAM-Projectname
PATH_EPAM=\StorageCard\EPAM\PROJECT	EPAM-Projectpath
PATH_DATA=\StorageCard\EPAM\DATA	EPAM-Datapath
PATH_BACKUP=\StorageCard\EPAM\BACKUP	EPAM-Backuppath
PATH_INI=\StorageCard\EPAM\INI	EPAM-INI-Path
SHOW_INFO=	Debug-Info: SHOW_INFO=t ...Show Page build up time (Default: off)
EPAM_NOEXIT=NO	Disable Exit-Button in Dialog Box (EPAM-Error) (Default: No)
EPAM_RDONLY=NO	Disable all write to disk (Default: enable)
INIT_PICTURE=startup.PCX	Display Picture during startup
RUNMODE=0	Must be 0
EPAM2RAM=NO	Install RAMDrive (EPAM: or \EPAM)
EPAM_NOBEEP=NO	Disable Touch Beep
VIDEO_MODE=VESA640X480X256	Videomode (Default: VGA; VxWorks only)
PROJECTVERSION=V1.0	Project Version
PROJECTNAME=Project	Project Name
PROJECTPROGRAMMER=	Project Programmer
PROJECTTARGET=CPC600-10	Project Target
RAMDRV_SIZE_KB=8192	Size of RAM Drive in kB
PATH_LOG=\EPAM\LOG	Path for Datalog
LOCALHOST=xxx.xxx.xxx.xxx	IP of local host
PATH_IMG=	EPAM-Imagepath (absolut)
PATH_FNT=\StorageCard\EPAM\FNT	EPAM-Fonts (absolut)
ORIENTATION=Landscape	Orientation of Screen (Landscape or Portrait)
INPUT_DEVICE=Touch	Set Input Device: Touch/Mouse
LOAD_LANGUAGE=-1	Default: -1 = load all languages into RAMDrive; 1 = load only 1 language into RAMDrive
USERCOLORS=No	Yes = use full color palette (0-255) of PCX-images in project; USERCOLORS=name.PCX = use fixed color palette (0-255) of PCX-image name.PCX
DRIVER=Yes	No = disable communication driver
KBD=YES	No = disable keyboard input

All yellow marked entries will not be modified by EXCEL-Macros and can be changed manually.

3 Project implementation

EPAM was developed in order to allow graphical operator interfaces to be created as simply and quickly as possible. The project implementation procedure was therefore based on the **"fast prototyping"** method. In other words, a functional pattern is created and then tested immediately. This effectively supports and promotes professional project handling (specifications, concept, implementation, commissioning, testing etc.) by enabling the customer to check the specifications at an early stage using a functional sample.
We therefore recommend that projects are implemented in the following way:

3.1 Summary of requirements

The requirements of a graphical operator interface should be defined in the project specifications. The level of IT knowledge of the end users, dialog languages etc. should be given particular consideration.

3.2 Structuring the screen pages

This refers to the sorting of the different inputs/outputs on the different screen pages. At this stage, the different user profiles should be considered, e.g. operator profiles for production and setup parameters, or service profile for setting and machine parameters etc.
The optimum user-friendliness is achieved if the functions required are initiated with the least number of entries on the operator interface.

3.3 Defining the screen page layout

This stage provides the basis for creating the images, texts, and fonts if required. Experience has shown that a considerable amount of time is taken up with the creation of images for a visualization project, and modifications to the screen page layout often also require considerable modifications to the images created. It is therefore useful to work only with texts at this initial stage and test the design directly. Visual improvements can then always be made at a later stage.

3.4 Creating images

Images for EPAM (pictures and icons) are created using a standard graphics program such as Paint Shop Pro. Images for EPAM must be created in PCX format with 256 colors.

3.5 Implementation with Excel

At this stage you can now start with the implementation, and define and link your screen pages in Excel. See also chap. 6 Designing with Excel P.37.

3.6 Connection to the PLC

The visualization project is linked to the PLC by defining symbolic variable names in the VarValue, VarState, Limit1 and Limit2 columns. The variable names can be transferred by importing the symbol file from the CoDeSys programming environment.

Communication principle:
The communication between EPAM and PLC is implemented by means of Read/Write operations for individual variables and entire structures (records). In other words, the communication driver requests actual values from the PLC by means of variables. **Any modified setpoints are sent immediately to the PLC by means of individual variables, and then read back. In other words, a setpoint value can be reset by the PLC, which will then be displayed immediately in the visualization system**.
All variables are read and initialized when EPAM is started. Variables are otherwise interrogated cyclically and only the modified values are refreshed on screen. Only those variables that are required at the time are interrogated, i.e. the variables of all the screen pages (windows) that are opened at the same time.

 Reading and writing of variables is NOT synchronous to the PLC-cycle!

3.7 Documentation

The transparency of the ASCII data format means that project documentation is created virtually automatically at the same time as the project. Additional comments can be added via the Insert - Comment function in Excel. These comments can be added in any line apart from those with the object prefix '#'. They are only shown in the Excel file and therefore have no effect on the execution speed or memory on the target system.

 If additional comments are added in Excel via Insert - Comment in lines with the '#' object prefix, these comments will be deleted any time the project is updated using the Update Objects EPAM macro.

Ideally, the project should be completed at this stage and the operator interface should be ready to run on the target system. In practice, however, the process described will have to be run through several times, since requirements are modified or extended during project implementation as new information about the project is obtained. With EPAM, however, this does not present any problems since modifications and additions can be carried out simply and quickly thanks to the use of Excel.

3.8 Tips for touch screen applications

 Use light background colors if possible. This reduces the visibility of finger prints and improves legibility in a light environment.

 If possible keep to the basic colors red, green, blue, yellow, magenta, cyan, black and white. On flat-screen displays, only these colors ensure the optimum reading angle.

 Define your touch-activated zones as "finger-friendly" as possible (a finger is not a mouse pointer!).

 Use the options for showing and hiding objects, and, if possible, only provide the operator with those action fields that are required at that moment. This will ensure a more intuitive application and make it easier to use. This approach will also ensure optimum use of the benefits of touch screen technology.

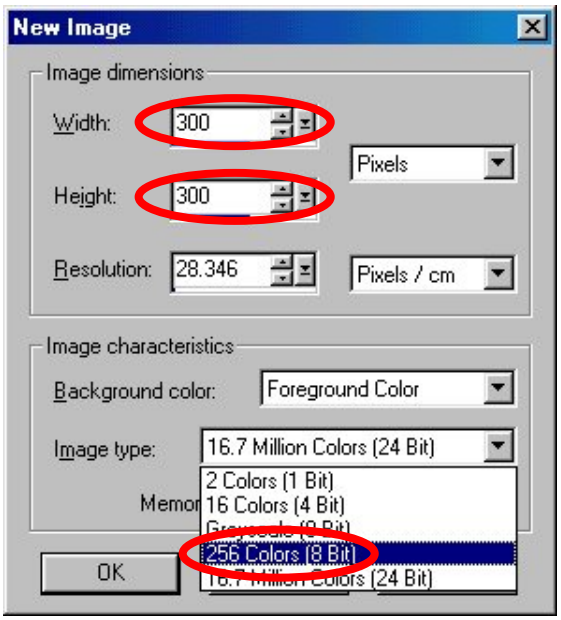
 Use „Beep" as acoustic feedback.

 Reduce possibilities of faulty operation by the use of e.g. Screensaver, additional acknowledge dialogs for critical functions etc.

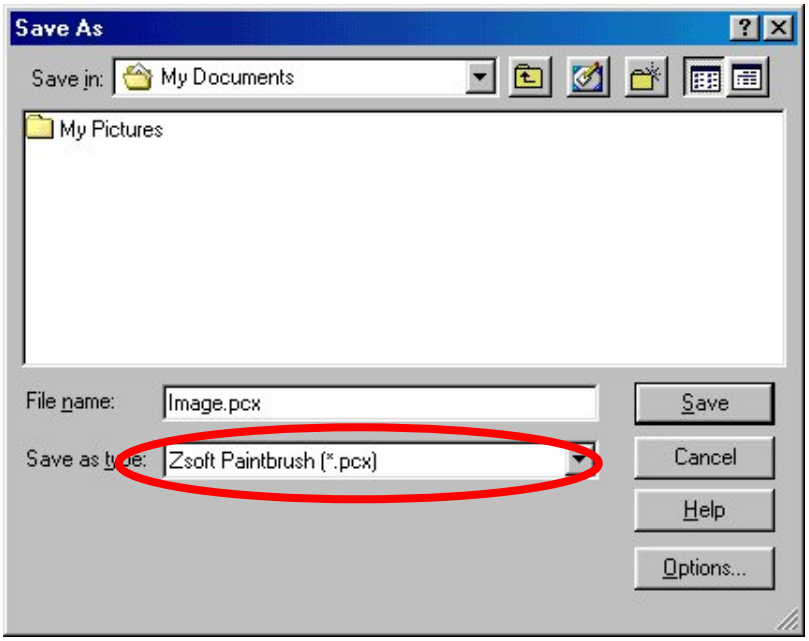
4 Creating images

4.1 Creating PCX images

Start Paint Shop Pro, choose File - New, and select the dimensions required and color depth corresponding to the used device (e.g. VxWorks 256 colors, WindowsCE 16 Million colors).



Create the image and save it in PCX format, Version 5 in the project directory by choosing File - Save As.



Notes for use of 256 color images with color palettes:

To ensure optimum performance all PCX images should be created in the same format and the same color palette **of 256 colors.** Otherwise the color palettes of the PCX images will be loaded during the runtime, and the PCX images will have to be converted to the correct format during the runtime.

The EPAM PCX Colortranslation macro converts 16-color images (e.g. icons) to 256-color images and adjusts the first 16 colors of 256-color images according to the EPAM color palette. This is the same as the 16 Windows colors.

User defined color palette
Alternative it is possible to use a userdefined color palette. In this case all 256 colors can be defined individually. With USERCOLOR=image.pcx in EPAM.INI the userdefined color palette will be loaded on EPAM start from image.pcx. In this case all images must have the same color palette! (USERCOLOR=Yes does the same, but the color palette will be loaded for each image within the project).
The worksheet UserColor can be used optionally to define the UserColors 0-255 and the corresponding RGB-values for the EPAM Wizard.

Color number/name	R (0-255)	G (0-255)	B (0-255)
0	0	0	0
1	128	0	0
...			

4.2 Creating Icons

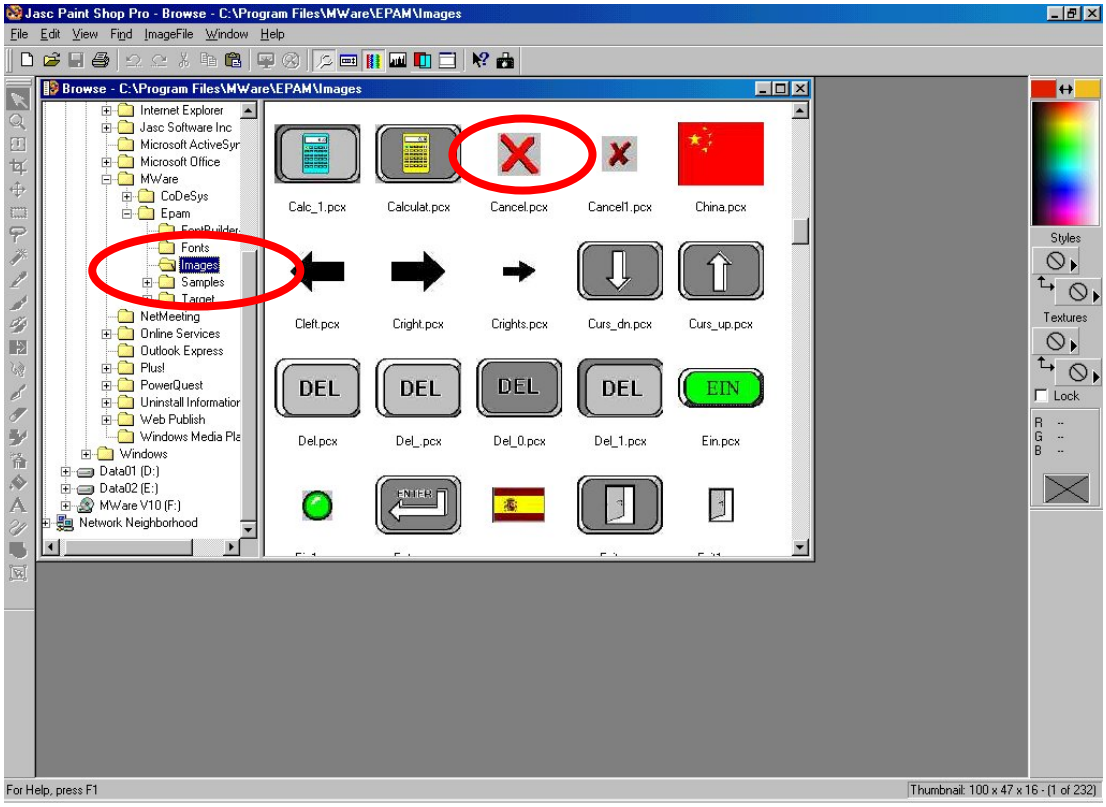
EPAM icons are saved in the same format, but using the file suffix MyIcon.ICO (the file suffix must be entered, otherwise the file will be saved as a *.PCX file.)

Notes for use of 256 color images with color palettes:

Icons should be created with 16 color and then converted into 256 color format by the use of the macro PCX-Colortranslation („256“). If Icons are created with 256 colors then all images of this page have to use the same 256 colors (same color palette).

4.3 Importing images

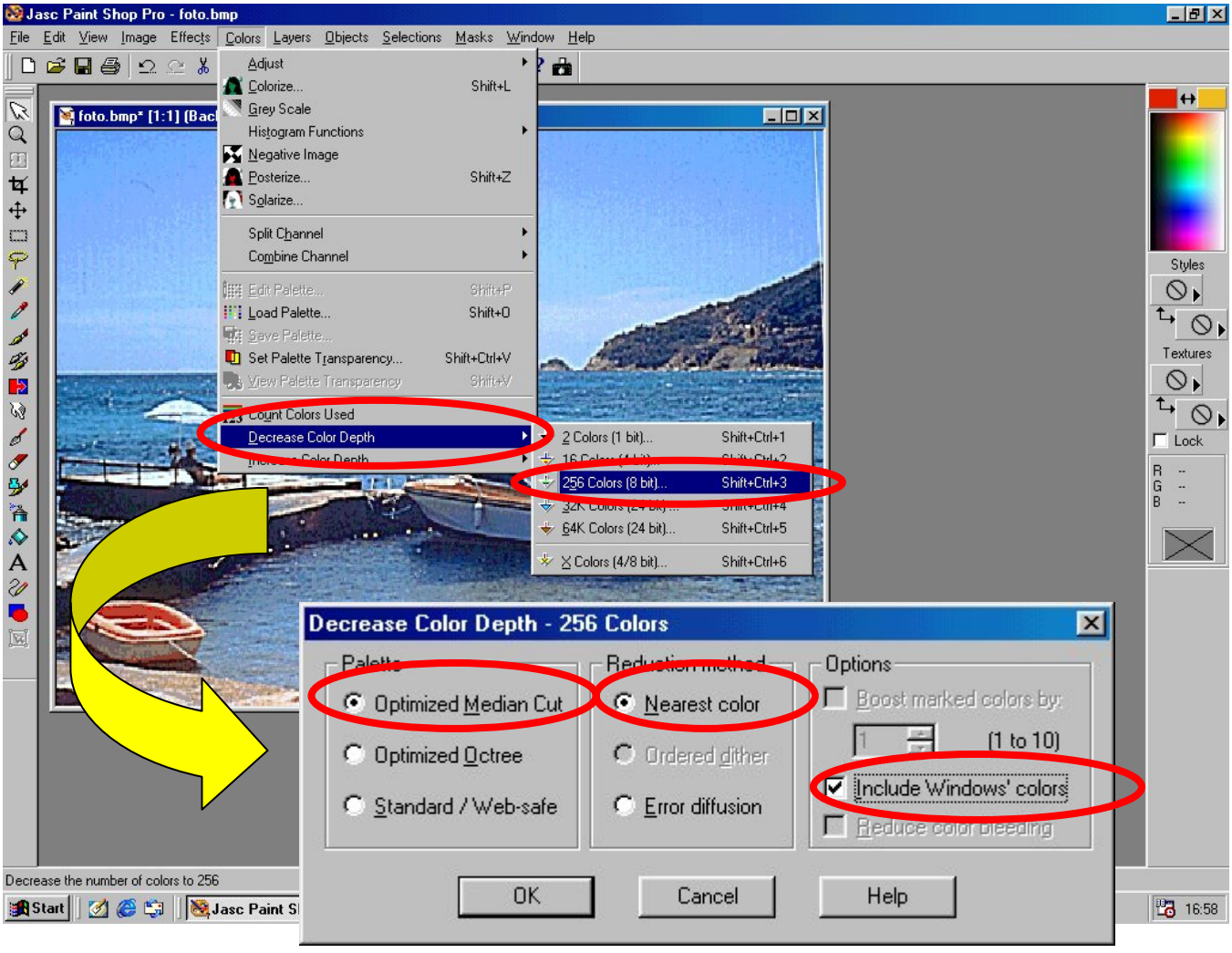
The File Browser function can be used to show all image formats supported by Paint Shop Pro. To do this, select the required directory. You can then select the images to be processed and double-click them in order to open them. Click File - Save As... to save the images in PCX format.



4.4 Images from digital cameras or scanned images

Notes for use of 256 color images with color palettes:

Photographic images can be reduced to 256 colors in Paint Shop Pro. To do this, choose Color - Decrease Color Depth - 256 Colors, set the Palette to Optimized Median Cut, the Reduction Method to Nearest Color, and activate the Include Windows' Colors check box in the Option field. All images created in this way must then be converted with the EPAM PCX Colortranslation macro. The color palette of all PCX images in the project directory is adapted so that the first 16 colors of the color palette are adapted to the Windows colors used in EPAM. This is necessary so that a button on this photo image is shown in the correct color. If there are 16-color images (e.g. icons) in the project directory, these are converted to a 256-color format, and the remaining 240 colors of the color palette are initialized with black. In this way photographic images can be mixed with icons.



Restriction

When a photographic image with 256 colors (full color palette) is displayed, the color palette of the appropriate image is loaded. In other words, when several images are shown on the same screen page, these images must use the same color palette. The required color palette for these images can be loaded in Paint Shop Pro via Colors - Load Palette. However, PCX images that only use the first 16 colors (e.g. icons) can be mixed with photographic images.

4.5 Transparent PCX-Images

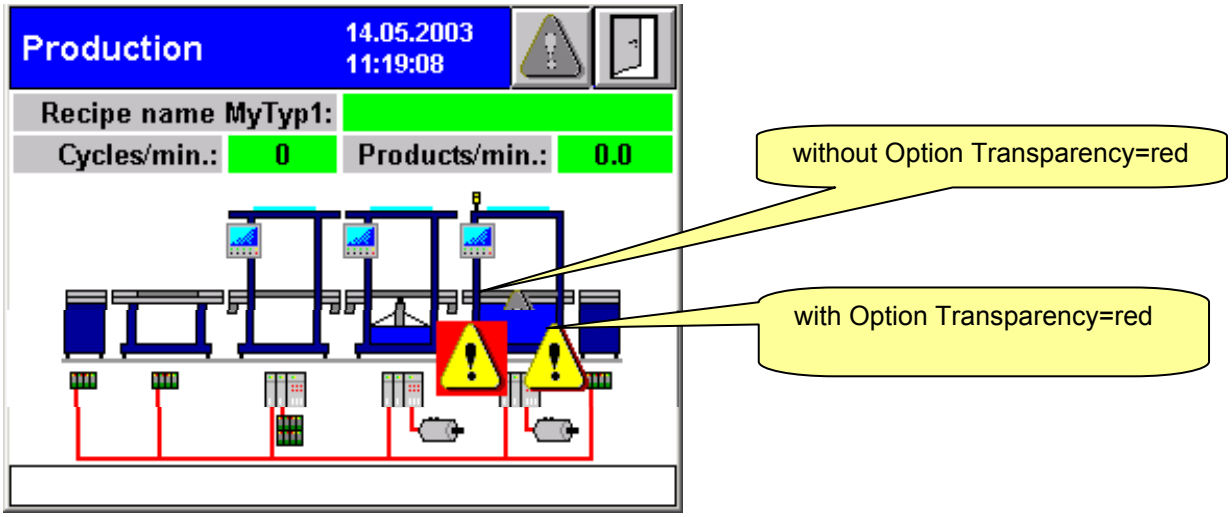
PCX-Images have normally a rectangle content. That means, any existing background within the image is replaced by the rectangle image. The option transparency is a possibility to declare one color inside the image as a transparent color. That means, instead of this transparent color the actual background will be displayed.

Example:

PCX-Image with yellow-black warning symbol and a red background

Definition of a PCX-Image with and without option Transparency=red

Object	Text/File	Font	X	Y	DX	DY	Color	Back-color	Format	Action	Limit1	Limit2	Action Limit1	Action Limit2	Var-Value	Var-Type	Var-State	Option
#Signal	AlarmOn.pcx		205	150	34	34	black	grey										
#Signal	AlarmOn.pcx		245	150	34	34	black	grey										Transparency=red



5 Creating Fonts

Fonts can be created within the Sheet „Fontmap“ by using the EPAM macros „New Font“ and „Build Fonts“ and the Utility „FontBuilder-Unicode“ (only VxWorks). The Fonts will be automatically converted from Windows True Type Fonts into a format which is useful for the target system.

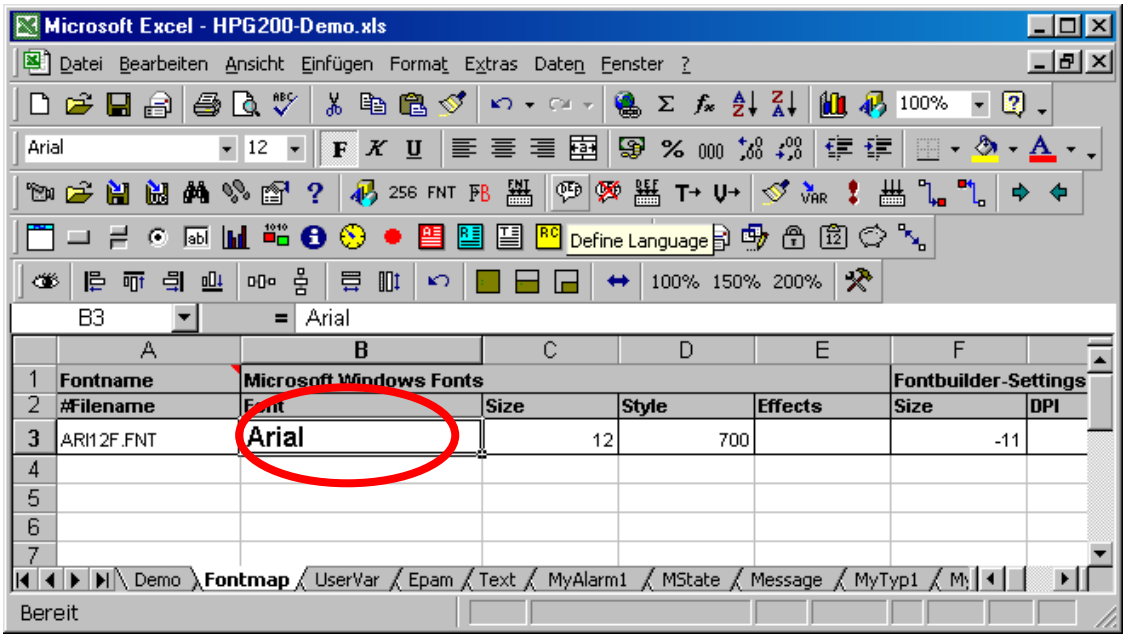
Note!

Only Windows TrueType Fonts can be used.

5.1 Definition of Fonts

Start EPAM-macro „New Font“. Enter a name (max. 8 characters) for the Font and close the menu with „OK“.

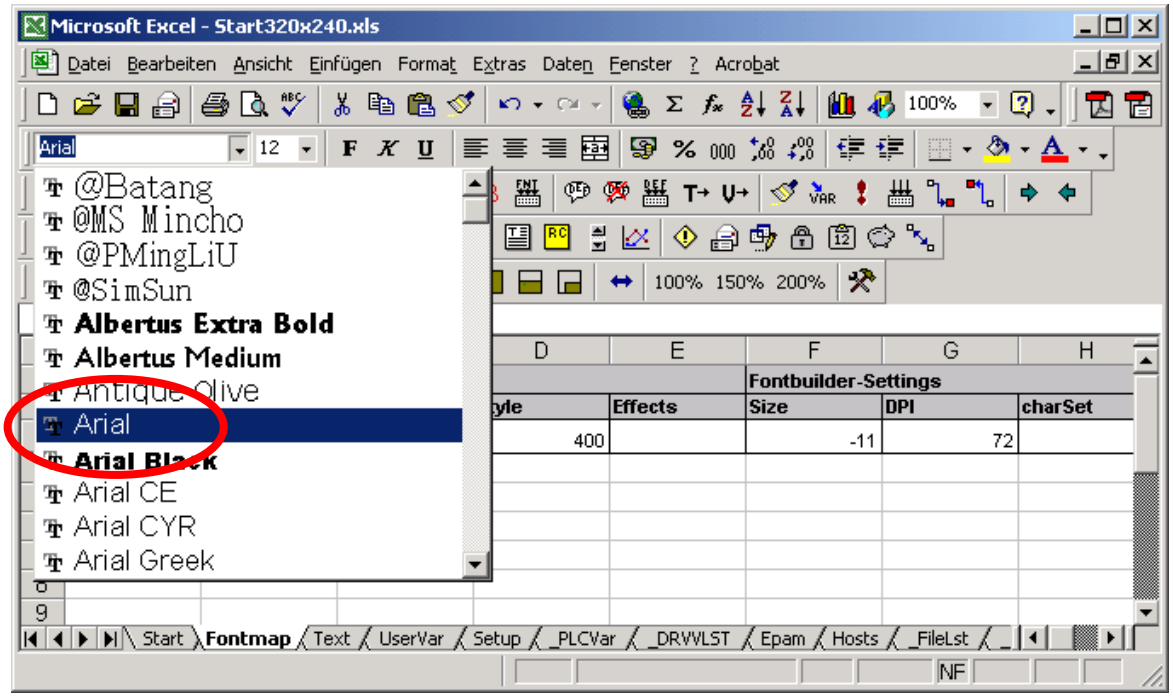
A new Fontdefinition will be inserted in sheet „Fontmap“ and the Fontfield will be selected...

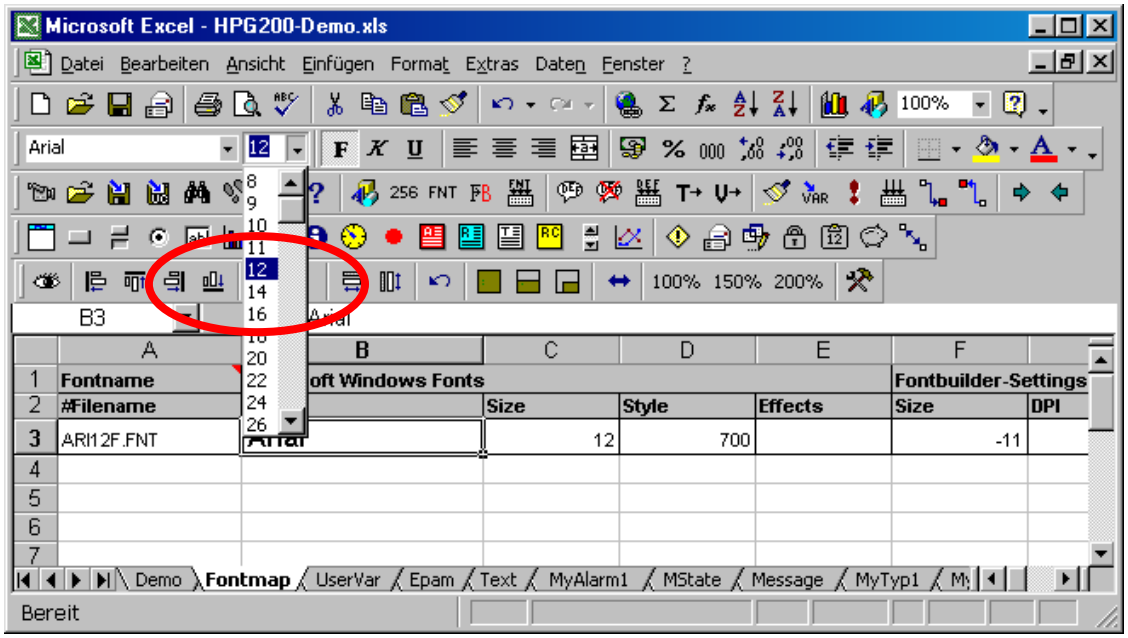


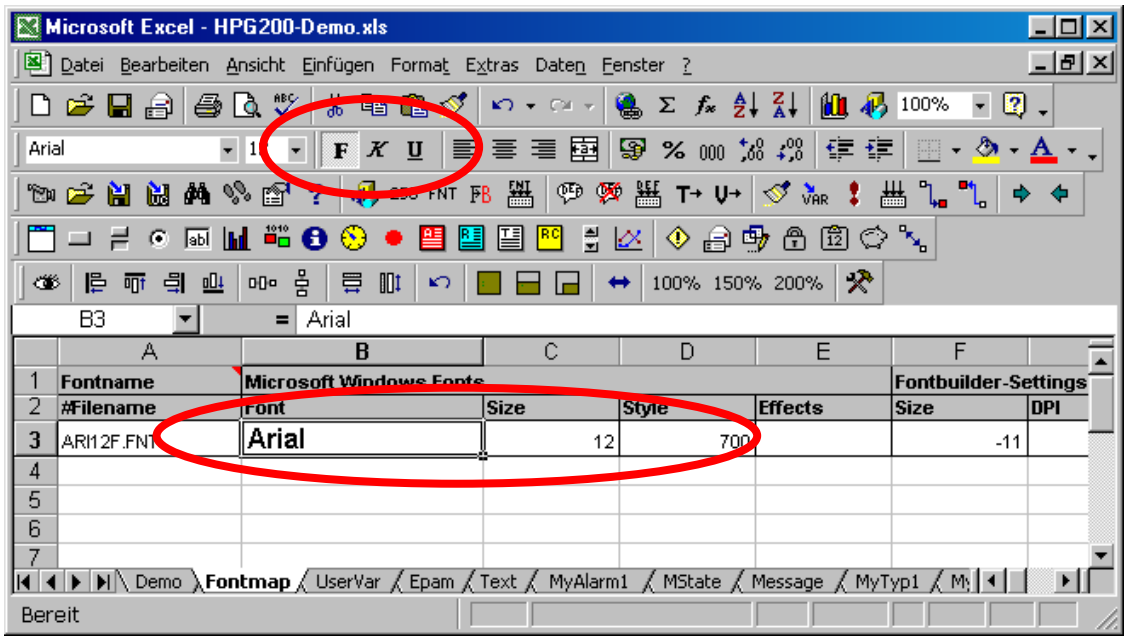
Note!

The text within the Font filed will automatically set to the name of the Windows True Type Font when EPAM-macro „Build Fonts“ is executed.

Now you can define the font attributes (TrueType Font, size, style) by the use of the normal Excel features...

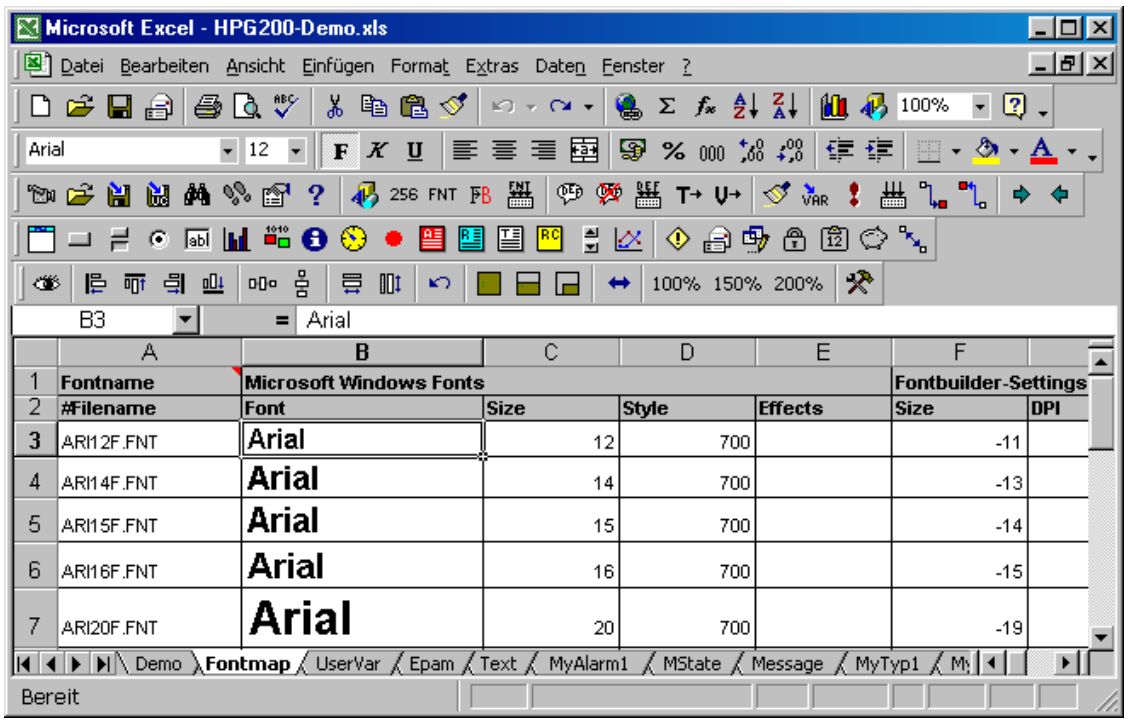






Note!
The fields Font, Size, Style, Effects,… should not be changed. This parameters will be automatically set by EPAM-macro „Build Fonts“.

Use the same procedure to define further fonts…



Note!
On devices with VxWorks the following Fonts are predefined and can not be changed: System72, Sysfnt72, Sys06x11, Arial12

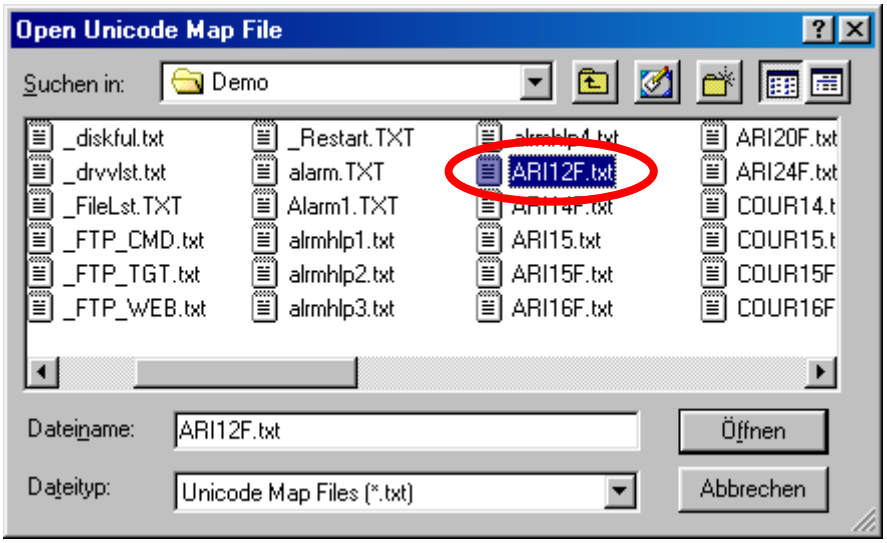
5.2 Building Fonts

This chapter is relevant only for devices with VxWorks operating system!
Start EPAM-macro „Build Fonts“ to build all defined Fonts. Now all Fontfiles (*.FNT) will be created and also a mapfile (same name as the Font, but with file extension .TXT).

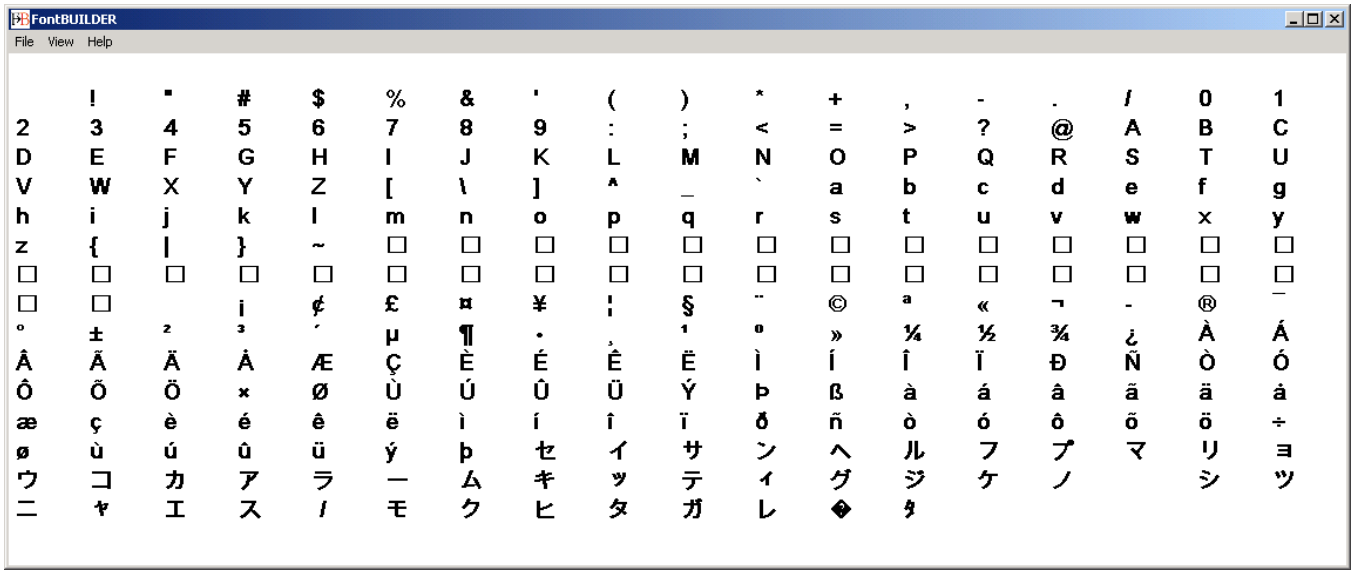
5.2.1 Limitations

The created fonts should be checked with the utility „FontBuilder-Unicode“ and the created mapfiles.

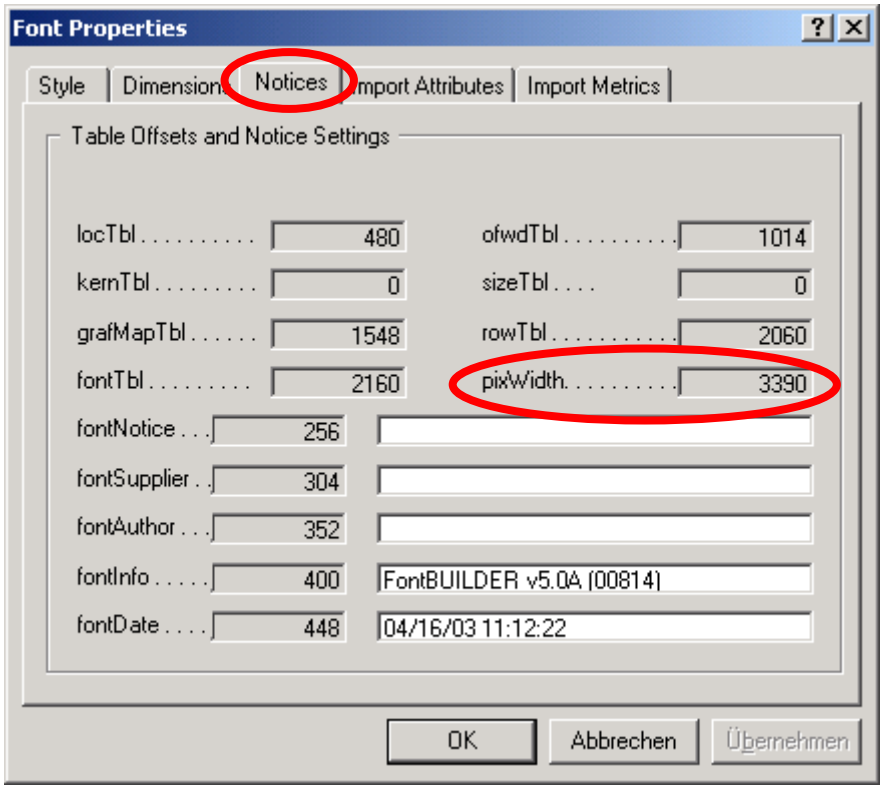
Start EPAM-macro „FontBuilder-Unicode” (F->B) and open „File-Mapfile“ for each font.



The Font will be displayed with all necessary characters:



Check now the following properties:



The entire width of the created font is shown in the pixWidth field in View - Properties. The font size for the font import must be selected so that this value <= 32767. In other words, with an average character width of 10 pixels, **the number of available characters per language is restricted to a maximum of approx. 3200!**

5.3 Unicode support

5.3.1 Function

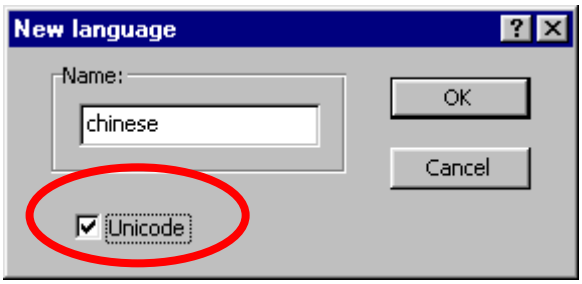
Other languages can be defined as Unicode languages. The texts are entered in Word via Insert - Symbol and by using the Unicode fonts supplied in Office 2000 (e.g. Arial Unicode MS).
All text files containing Unicode texts (EPAM language files, messages, alarms, etc.) are saved in Unicode text file format (EPAM Save as Unicode Text macro or directly in Word using Save As and Save as type: Encoded Text).
When calling EPAM using the Start EPAM macro, all Unicode text files are converted back into normal ASCII file format. A map file fb_map.txt is created that contains all the Unicode characters required for this application. This is used by EPAM-macro “Build Fonts” to create an EPAM compatible font from a Windows TrueType font e.g. Arial Unicode MS. (only VxWorks)
On devices with WindowsCE the Windows-Fonts (*.TTF) will be used.

Note!

Input of foreign language texts in EXCEL (also Unicode-characters) can be done by the help of the office tool „Visual-Keyboard“ and installation of the required keyboard drivers. The Visual-Keyboard (VkeyInst.EXE) is available on the Microsoft-Homepage:
http://www.microsoft.com/downloads Search: “Visual Keyboard”

5.3.2 Definition of a Unicode language

In Excel a foreign language (EPAM Define Language macro) can be defined as a Unicode language:



The default language (column B) cannot be defined as a Unicode language!
Fonts for different languages must have a unique name.

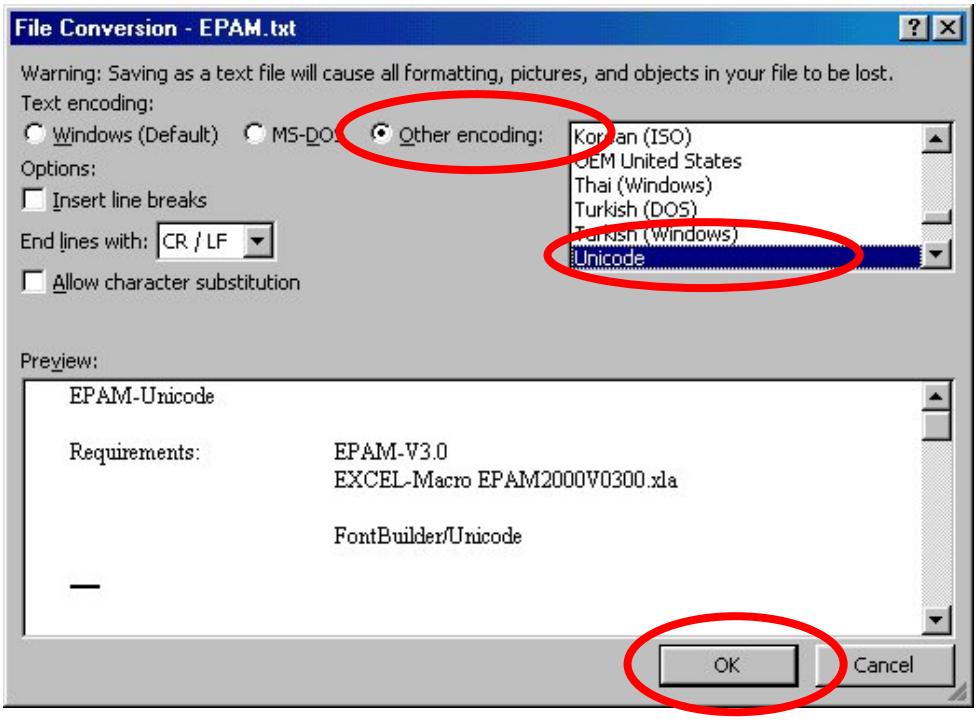
Two new columns for Text/File, Font and a subdirectory with the language name are then created at the end of the file. The Font column is marked with the comment LanguageUC (this must not be deleted).

5.3.3 Entering the Unicode texts

Menu texts
Unicode texts can then be displayed in the language column e.g. with the Arial Unicode MS font. The texts are entered easily with Microsoft-Officetool „Visual-Keyboard“ or in Word by choosing Insert - Symbol and using an Unicode font e.g. Arial Unicode MS. The texts can then be transferred to the Excel spreadsheet via Copy - Paste.

Messages, alarms
Message and alarm texts can also be defined in several languages. The texts are defined in the appropriate worksheets in the same way as the menu texts. When the Start EPAM macro is called, the alarm and message texts are automatically created as separate files in the corresponding language subdirectories. This operation can also be carried out manually with the EPAM Save as Unicode Text macro. Only the currently selected Excel worksheet is saved as a Unicode text file (2-byte code).

Text files for Text list object
Normal text files that are to be displayed with the Text list object, are best created directly in Word and then saved under File - Save As, Save as type: Encoded text with the following option.



5.3.4 Start EPAM

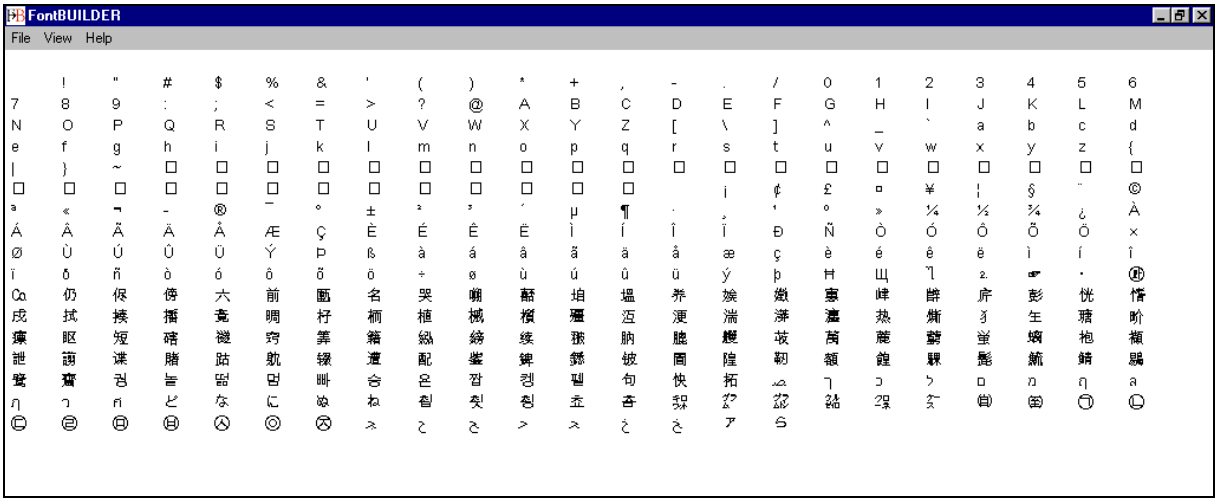
When EPAM is called, all Unicode text files are automatically converted back to normal text files with all Unicode characters shown as special characters in \xHHHH format (HHHH = Hexcode). These characters are inserted in the font from position 257. A map file fb_map.txt is generated at the same time, which can then be used with a font converter (FontBuilder/Unicode) from a Windows TrueType font (e.g. Arial Unicode MS to create an EPAM-compatible font that contains all the characters required). (only VxWorks)
On devices with WindowsCE the Windows-Fonts (*.TTF) will be used.

5.3.5 Creating a Unicode font

This chapter is relevant only for devices with VxWorks operating system!

Creating of a unicode font is done in the same way as creating „normal“ fonts (see chap. 5.1 Definition of Fonts P.29). The Windows True Type font must be a Unicode fonts which contains the necessary Unicode characters e.g. Arial Unicode MS (it may be necessary to install international language support for MS-Office). The created font should be checked with the Utility FontBuilder-Unicode (macro: F->B) by opening the corresponding mapfile e.g. UNI14.txt (Import – Mapfile).

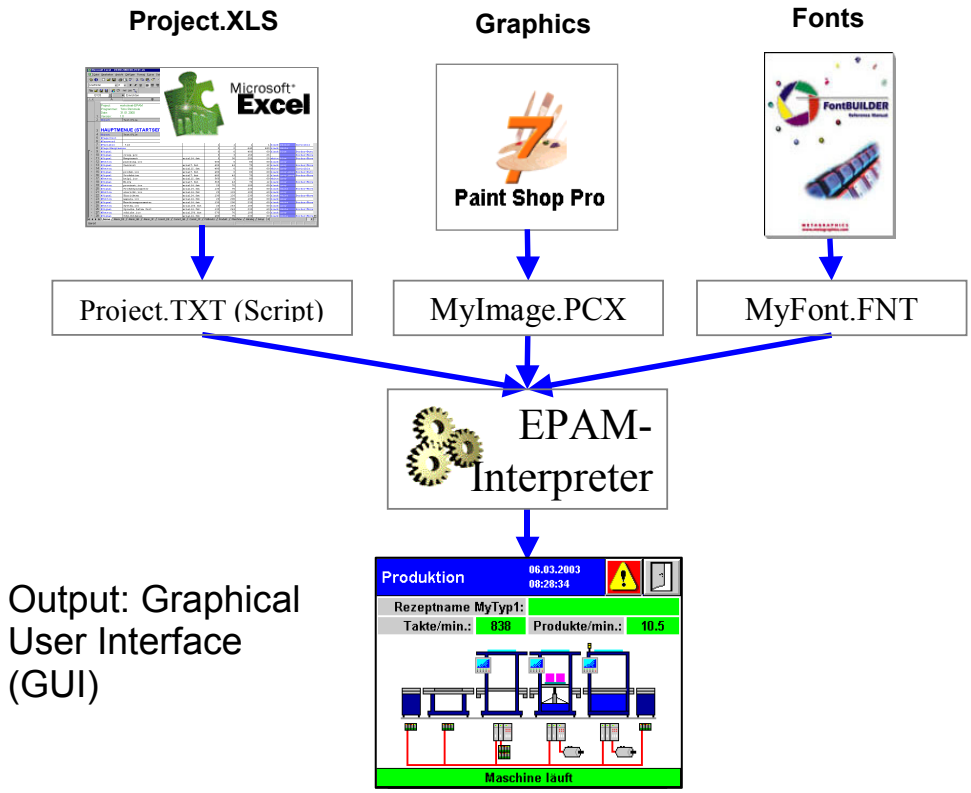
The new font created is now shown with all the characters required:



6 Designing with Excel

6.1 Operating principle

EPAM is an interpreter, i.e. the objects and screen pages are defined in a structured and tabular ASCII file (so-called script file), and are converted by EPAM for graphical display on the screen (similar to an Internet browser). The script file contains the definitions of the individual screen pages and their objects, and is created in Excel. Graphics are created as PCX image files with a standard graphics program and are referenced by their file names in the script file. The same principle applies to the font files.



When EPAM (Start EPAM macro in EPAM) is launched, the Excel worksheet is saved as an ASCII text file. This enables the project data to be independent of the Excel version used.

Advantages of this concept

- Simple screen page creation and object definition with Excel
- Transparent, readable data base
- Supports a wide range of hardware platforms
  (¼ VGA 320x240 with 16 greyscales/colors up to 1280x1024 with 16 million colors)
- Integrated communication to CoDeSys PLC environment using symbolic names
- Freely selectable colors and fonts
- Online language selection, also Unicode (e.g. Chinese characters)
- No expensive Windows development environment

In visualization applications, a large number of lists always have to be managed. It is therefore advisable to make use of a standard software package like Excel for these types of tasks.

Advantages of Excel:

- Straightforward screen display and printout (project documentation)
- Existing objects and complete screen pages can be copied easily
- Formulae and automatic filling of cells possible by dragging
- Simple and fast modifications possible
- The use of Excel macros enables applications to be designed to customer requirements and further developed as required
- Input tools and project testing tools directly accessible from Excel, by simply clicking pre-defined icons

The Project.XLS file contains an empty project template. This file can be adapted to your requirements and stored as a sample template in the MSOffice TEMPLATE directory. You can then start a new project with your default settings in Excel via File - New.

Launching EPAM:
You can launch EPAM directly from the Excel user interface with EPAM's Start EPAM macro and then you can exit the simulation at any time via the ESC key.

6.2 Structure of the Excel spreadsheet

Objects are provided for defining the screen pages. An object can be defined for each line in the Excel spreadsheet. A screen page starts with the object **#Page=name** and ends with a blank line or with a line that does not start with the object prefix '#'. The page object defines the screen page (position and dimension) in which all the objects below it are placed. An object starts with the #Object keyword. All lines that do not have the object prefix in the first column are comment lines.

The columns of the Excel spreadsheet contain the properties of the object concerned and have the following meaning:

Column	Meaning
Object	Object name, e.g. #Page=Name
Text/File	Name of a PCX image or ASCII text file and a text string for the default language
Font	Optional object-specific font for the default language
X,Y,DX,DY	X, Y position, width and height of the object in pixels (in relation to top left!)
Color	Foreground color (0-15 or color name)
Backcolor	Background color (0-15 or color name)
Format	Object-specific format definitions
Action	Action on touch actuation
Limit1	Lower limit value: Constant value, PLC variable or system variable for limit value
Limit2	Upper limit value: Constant value, PLC variable or system variable for limit value
Action Limit1	Action on undershooting of Limit1 (**Variable value < Limit1**)
Action Limit2	Action on overshooting Limit2 (**Variable value > Limit2**)
VarValue	PLC variable and system variable for object value
VarType	Variable type
VarState	PLC variable and system variable for object status
Option	Object-specific options
C function	Optional user C function

6.2.1 Object column

EPAM objects are defined in the Object column. The following objects can be defined:

Object	Meaning	Designing
#Alarm	Alarm monitoring with History function (512 alarm messages), time stamp for Come, Go and Acknowledge alarm	Global
#Alarmlist	Output of alarm events within a rectangular area	
#Alarmmail	Email notification as a result of alarm events	Global
#Bar	Display of value in a rectangular bar	
#Button	Non-latching, touch-activated area	
#DataLog	Recording of PLC data/variables in a DataLog file	Global
#DBPasswd	central User/Password-handling via mySQL-Database1	Global
#DBTrace	DataLog-object for recording data in a mySQL-Database1	Global
#DiagSig	Diagnostics signal, display of flashing points (e.g. machine image) on alarms for diagnostics	
#DropDownList	touch-activated area, selection of one of several options in a listbox	
#HTMLBrowser	Display of HTML-files within a rectangular area (requires Internet Explorer)	
#Message	Output of messages in text or image information	
#Meter	Display of a value in a semicircle/circle/user-defined segment	
#Page	Screen page dimensions	
#Password	Password management	Global
#RadioButton	Touch-sensitive area, selection of one of several options	
#Recipe	Recipe management	Global
#RecipeList	Output of a recipe list within a rectangular area	
#ScreenSaver	Screen saver	Global
#Scrollist	Scroll list, display of objects as a scroll list, e.g. parameter list	
#Signal	Display of states or static images and texts	
#Switch	Latching, touch-sensitive area	
#Textlist	Output of an ASCII text file within a rectangular area	
#Trend	Display of the DataLog file as a trend graph	
#Variable	Display of a numeric/alphanumeric variable	
#Sys2Plc	Synchronisation of EPAM-system variables and PLC variables (e.g. active Page)	Global
#RemoteControl	Remote control of different HPGs via Ethernet (like PC-anyware)	

6.2.2 Text/File column

The Text/File column contains the text of an object, the file name of a PCX image/icon or the name of a text file (additional worksheet) with object-specific settings. Texts can be selected from the text list (Text worksheet) and added to the Text worksheet with EPAM's Add Text macro.
EPAM's Open File macro can be used for selecting and entering PCX, ICO and TXT files.

6.2.3 Font column

The Font column contains the file name of a font file (*.FNT) that is used for displaying the text. The pulldown menu can be used for selecting and entering FNT files.

1 Requires option „EPAM-DB-Extension“ on target (at the moment only available for VxWorks) and the EPAM-DB-Server running within the Network

6.2.4 X,Y,DX,DY columns

The X, Y, DX, DY columns define the position and the dimension of the object in pixels in relation to the origin position of the current screen page. The reference point (origin point) is the top left.

Automatic positioning
The *Font, X, Y, DX, DY, Color and Backcolor* fields can also be kept empty. These fields will automatically take on the values from the previous object. In other words, *Font, DX, DY, Color* and *Backcolor* only have to be entered for changes. If the X,Y fields are left empty, the subsequent objects are automatically positioned in succession. The direction in which objects are positioned is in the direction of X, at intervals equal to the value of the current width (DX). If this exceeds the current screen page width setting, there will be a line break to the value of the current object height (DY). If different objects have to be positioned underneath each other within a column, only the X co-ordinate has to be set to the same value for all objects, the Y co-ordinate is then defined automatically.

EPAM Wizard
The EPAM Wizard allows you to change the *X, Y* position values and the *DX, DY* dimensions of objects quickly and easily by mouse.

6.2.5 Color, Backcolor column

The *Color and Backcolor* fields as well as the actions *Color=x and Backcolor=x* can contain the numeric values or the following color names. The standard color palette is the same as the palette of the Paint Shop Pro graphics program (Standard Windows VGA color palette) and is shown below:

Index	Color name
0	black
1	dark red
2	dark green
3	brown
4	dark blue
5	dark magenta
6	dark cyan
7	grey
8	dark grey
9	red
10	green
11	yellow
12	blue
13	magenta
14	cyan
15	white

6.2.6 Format column

The Format column contains object-specific display versions.

6.2.7 Action column

The Action column contains the definition of the object-specific action that is executed when the object is selected with a touch operation or with the mouse. Possible actions include screen changes, changes to variable values etc.

Multiple actions
The '&' character allows several actions (Action column) to be triggered at the same time.
Example: SetVar=1 & #Page=*name* ...Set variable to 1 and then change to screen page *name*

#Page=name should be the last action!

Standard keyboard table
The action Key=key code can be defined with normal ASCII characters, with one of the following key names and the corresponding key code.

F1	...Key F1 corresponds to key code \x3b00
F2	...Key F2 corresponds to key code\x3c00
F3	...Key F3 corresponds to key code \x3d00
F4	...Key F4 corresponds to key code \x3e00
F5	...Key F5 corresponds to key code \x3f00
F6	...Key F6 corresponds to key code \x4000
F7	...Key F7 corresponds to key code \x4100
F8	...Key F8 corresponds to key code \x4200
F9	...Key F9 corresponds to key code \x4300
F10	...Key F10 corresponds to key code \x4400
F11	...Key F11 corresponds to key code \x4500
F12	...Key F12 corresponds to key code \x4600
ESC	...Key ESC corresponds to key code \x1b
CursorUp or CUp	...Cursor up key corresponds to key code \x4800
CursorDown or CDown	...Cursor down key corresponds to key code \x5000
CursorLeft or Cleft	...Cursor left key corresponds to key code \x4b00
CursorRight or CRight	...Cursor right key corresponds to key code \x4d00
PageUp or PgUp	...Page up key corresponds to key code \x4900
PageDown or PgDn	...Page down key corresponds to key code \x5100
Home	...Home key corresponds to key code \x4700
End	...End key corresponds to key code \x4f00
Insert	...Insert key corresponds to key code \x5200
Backspace	...Backspace key corresponds to key code \x08
Return or Enter	...Return/Enter key corresponds to key code \x0d
Delete or Del	...Delete key corresponds to key code \x5300

6.2.8 Limit1, Limit2 column

The Limit1 and Limit2 columns define the object-specific lower and upper limit values of the variable. The limit value can be defined as a constant, a system variable or as a PLC variable. Limit value variables must be of the same type as the VarValue variable.

Limit values
The Limit1 and Limit2 limit values are part of the value range. In other words, the limit value is out of range if the **value is less or greater than** Limit1 or Limit2 respectively.

6.2.9 Limit1 Action, Limit2 Action column

The Limit1 Action and Limit2 Action columns define the actions to be executed when the value range defined by **Limit1 and Limit2 is undershot or overshot**. Possible actions include color change, screen change etc.

6.2.10 VarValue column

The VarValue column contains the name of a PLC variable or a system variable. System variables are global variables in EPAM that cannot be used for communication with the PLC. PLC variables are defined in the following syntax:

[[/Communication driver name/]Host name/]Variable name

Example:	/ARTI/PLC/HMIVar1	...Variable HMIVar1 of PLC with communication driver ARTI
	PLC/HMIVar1	...Variable HMIVar1 of PLC with default communication driver (=ARTI)
	HMIVar1	...Variable HMIVar1 of local PLC with default communication driver

Driver name and host name are optional and must be specified if variables of a different PLC are to be read.

The drop-down list box can be used to select variables from the UserVar list. When the project is compiled, EPAM checks whether all the variables used in EPAM are defined in the UserVar list and whether the data type of the variable matches the object data type.

Indexed variable addressing
A system variable such as 's_index' and the SetIndex=x button action allows variable names to be modified during the runtime and read as an index. In this case the variable name must be defined in the following way (VarValue column):

MyVariable***%s_index%***xy

Spaceholder for index (name of a system variable of type: WORD)

When the screen page is being generated, the current value of the index variable is inserted instead of the spaceholder %s_index% and this variable is interrogated.
Example: MyVariable1xy …when s_index holds the value 1

The range for the index variable can be defined in worksheet „UserVar" in columns Limit1 and Limit2.

Application:
The indexed variable addressing option, combined with the Scrollist object, enables parameter lists to be created very effectively, for example, for any number of motion controls. This allows the parameters of several motion controls, for example, to be entered **with only one** screen page.

If the index variable is modified on the current screen page, the screen page must be generated again: SetIndex=x & #Page=CurrentPage

6.2.11 VarType column

The VarType column defines the object data type and shows which variable types can be assigned to an object. When the project is compiled, EPAM checks whether the variable type (UserVar) matches the object data type. The following basic data types of IEC61131 are supported:

BOOL, BYTE, DINT, DWORD, INT, REAL, SINT, STRING:[xx], UDINT, UINT, USINT and WORD

The IEC_TIME data type is used in EPAM to display and enter time values and is interpreted in the PLC as a TIME data type.

The datatype IEC_DT is used in EPAM to display and enter date and time values DT and is interpreted in the PLC as a DT data type.

The TIME data type is reserved in EPAM for displaying time/date variables.

Complex data types such as structures or arrays are handled in EPAM as STRING variables with the corresponding length STRING:xx. xx stands for the length of the data type in bytes.

Array and structure elements can also be read and written, in addition to simple variables.

6.2.12 VarState column

The VarState column contains a variable name for the object status. The variable must be of type WORD, INT or UINT. The object status enables any object in EPAM to assume one of the following states:

Object status = 0	...Object is visible and active, i.e. the area X, Y, DX, DY is shown with the object
Object status = 1	...Object is invisible and inactive (off), i.e. the area X, Y, DX, DY is cleared with the background color of the current screen page. **Limits will NOT be checked in this case.**
Object status = 2	...Object is visible but inactive (disable), i.e. the area X, Y, DX, DY is hatched
Object status = 4	...Object flashes at app. 1Hz, i.e. the area X, Y, DX, DY is cleared with the current background color of the screen page and is then displayed again.
Object status = 8	...Object flashes at app. 2Hz

The object status is changed via the object status variable (VarState) by setting the appropriate value.

Object status on screen change
After a screen change, all the objects provided with an object status variable are initialized with the object status not visible and inactive (off). The object is not displayed according to its status until the current object status is read.
This procedure prevents actions from being started accidentally whilst the screen on the target system is being generated! (during simulation on the development environment, all objects are always shown!)

6.2.13 Option column

The Option column is used to define the object-specific options. Possible settings are for example, DX=, Scroll, Pos=, etc.

Multiple options
The ',' character is used to configure several options.
Example: Pos=Center,PWL=1,Scroll ...Positions the object in the center, Object is assigned password level and the object can be scrolled.

6.3 Excel worksheets

The first sheet contains the definitions of the different screen pages, objects, actions etc. and the assigned variables. There are also other worksheets with additional information such as on objects. These worksheets are created automatically if required.
EPAM provides the following types of worksheet:

Worksheet type	Meaning	Number
Project	All screen pages and their objects are contained in the Project worksheet. This worksheet **MUST** be the first worksheet!	1
Text	The Text worksheet is used for managing all project-related texts. All texts that were defined in the Text worksheet can be selected via the drop-down list in the Text/File column.	1
UserVar	The UserVar worksheet is used for defining all variables. Variables can be imported from the CoDeSys programming environment into the UserVar list using the PLC Variable Import macro. This operation will delete the existing variables and re-create the list. A check is also made whether all variables used in the Project worksheet are also defined in the UserVar worksheet, and whether their data type matches.	1
UserColor	Colour definitions (colour number/name, R, G, B) for correct display of user defined colour palette (UserColor) within the EPAM Wizard	1 optional
Epam	The Epam worksheet contains the Epam settings for the target system concerned and should not be changed.	1
Setup	The Setup worksheet contains different settings and should not be changed.	1
DRVParam	The DRVParam worksheet contains the settings for communication between EPAM and CoDeSys and should not be changed.	1
Hosts	The Hosts worksheet contains the settings for communication with different PLCs via Ethernet (TCP/IP).	1
Alarm	The Alarm worksheet contains the alarm definitions of the Alarm object.	1
Alarmmail	The Alarmmail worksheet contains the e-mail definitions of the alarm object.	1
Datalog	The DataLog worksheet contains the variable definitions for the DataLog object. A DataLog worksheet is created for each DataLog object and is referenced via the sheet name.	1 per DataLog object
Message	The Message worksheet contains the definitions of the Message object. A Message worksheet is created for each message object and is referenced via the sheet name. However, several message objects can also use the same message worksheet.	1 per message object
Recipe	The Recipe worksheet contains the variable definitions for a recipe type. Several different recipe types can be defined in a project (such as product and machine parameters). The relevant variable definitions are referenced via the sheet name.	1 per recipe object
Trend	The Trend worksheet contains the definitions for the trend object. A Trend worksheet is created for each trend object and referenced via the sheet name. Several trend objects can use the same Trend worksheet.	1 per trend object
_FileLst	System worksheet with all files used in EPAM.	1
Sys2Plc	The worksheet „Sys2Plc“ contains the variable definitions for a sys2plc object. A sys2plc worksheet is created for each sys2plc object and referenced via the sheet name.	1 per sys2plc-object
Fontmap	The worksheet „Fontmap“ contains the Font-Definitions (Name, Type, Style...)	1

The worksheet type is written as a comment in the first cell (A1) and must not be changed!

6.4 Password protection for EPAM-projects

An EPAM-project (*.xls) can be password protected via Excel. Select „Save as - Extras-general options“ and define a read/write password.

6.5 EPAM macros

The EPAM toolbars will appear after the EPAM Add-Ins have been installed.

- Easy PageMachine
- EPAM Objects
- EPAM Wizard
- EPAM-DB

These EPAM toolbars contain different additional functions in the form of Excel macros.

6.5.1 Easy PageMachine EPAM toolbar

Start EPAM macro

- Saves the current project (entire Excel spreadsheet). The first worksheet and all language-dependent columns are saved in the Text (tab delimited) format. The Windows simulation (EPAM.EXE) is then started with the current project.

EPAM Open File macro

- Enables the entry of file names (*.PCX, *.ICO, *.TXT) using a user-friendly file selection dialog. The selected file is copied to the current project directory and the file name transferred to the selected Excel cell.

EPAM Save Worksheet as *.TXT macro

- Saves the current worksheet in the Text (tab delimited) format. The file name is formed from the name of the worksheet and the file suffix.TXT.

EPAM Save as Unicode Textfile macro

- Saves the current worksheet in the Unicodetext. *.txt format. The file name is formed from the name of the worksheet and the file suffix.TXT.

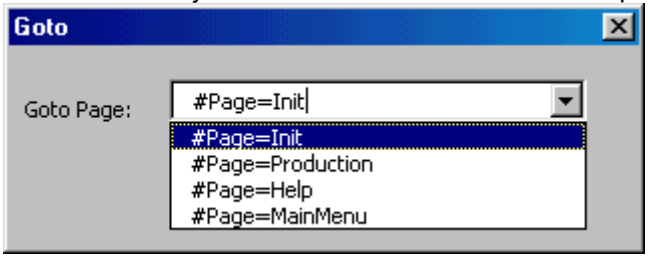
Requirement: Excel 2000 with international language support.

EPAM Search macro

- Searches for references of the selected cell in the current worksheet.

EPAM Goto macro

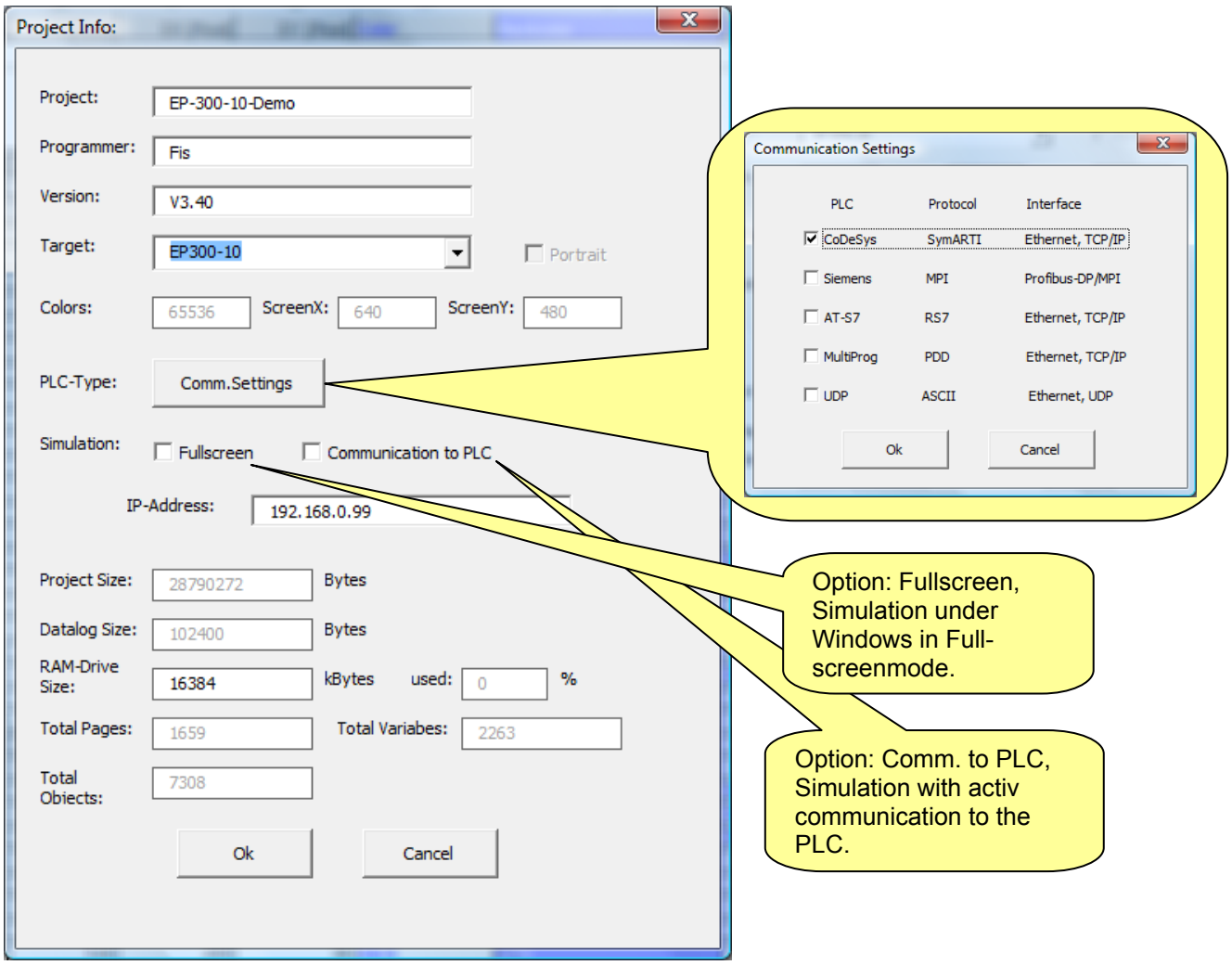
- Enables the jump in the Excel Project worksheet to a selected screen page #Page=*name*.



EPAM Project Settings macro

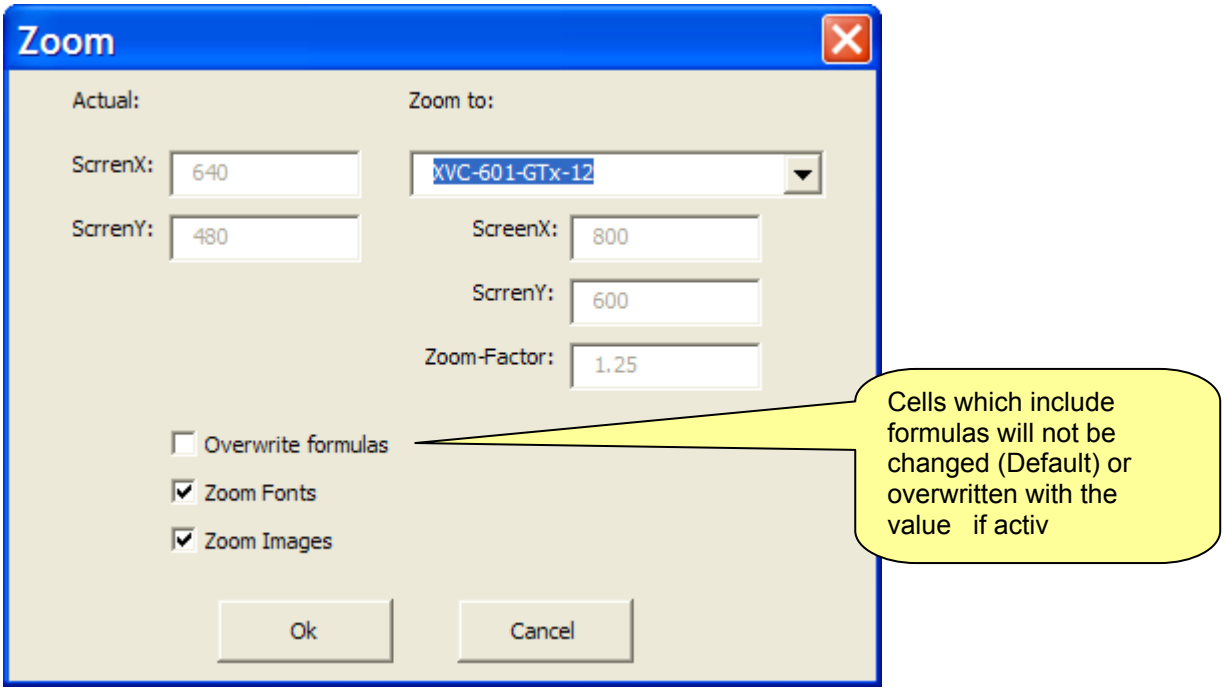
- Enables the entry of project information such as project names, project programmer and project version.
- Selection of the target system
- Sets the number of colors used and available memory of the RAM drive on the basis of the selected target system
- Output of project information such as the number of screen pages configured, number of PLC variables used, project size and the size of the DataLog file used, and checks the available memory in the RAM drive.
    - DataLog size and project size < = RAM drive size
      ⇒ Data logging and project can be run from the RAM drive
      (see also Download Project macro)
    - DataLog size < RAM drive size but DataLog size and Project size > RAM drive size
      ⇒ Only Data logging can be run from the RAM drive
      (see also Download Project macro)
    - DataLog size > RAM drive size
      ⇒ Neither data logging nor project can be run from the RAM drive
      (see also Download Project macro)

- Selection of PLC type, default: Codesys
- IP address of the Target (required if communication to PLC is activ)



EPAM-Macro „Zoom Project“

- Converts a project for different screen resolutions including Fonts and Images (optional).



EPAM-Macro „Compare project“

- Compares two EPAM projects

EPAM version macro

- Version information on the currently used EPAM macro. Input of the product code.

EPAM Open Drawing Program macro

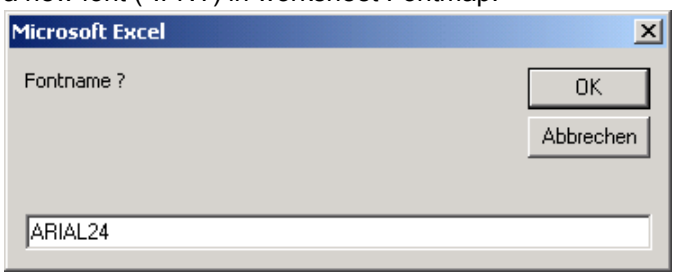
- Starts the graphics program with the selected PCX image. When started for the first time, the path of the graphics program must be entered using the Open File dialog. The current path is then saved in the Setup worksheet.

EPAM PCX Colortranslation macro

- Converts all PCX images and icons (*.ICO) in the project directory. Images with 16 colors are converted to 256-color images in which the remaining 240 colors of the color palette are set to black. The first 16 colors of 256-color images are converted to the colors in the EPAM standard color palette.

EPAM New Font macro

- Definition of a new font (*.FNT) in worksheet Fontmap.



Start FontBuilder macro

- Starting the FontBuilder. When started for the first time, the path of the program must be entered using the Open File dialog. The current path is then saved in the Setup worksheet.

EPAM-macro „Build-Fonts”

- Creates all Font files defined in the worksheet „Fontmap“.

EPAM Define Language macro

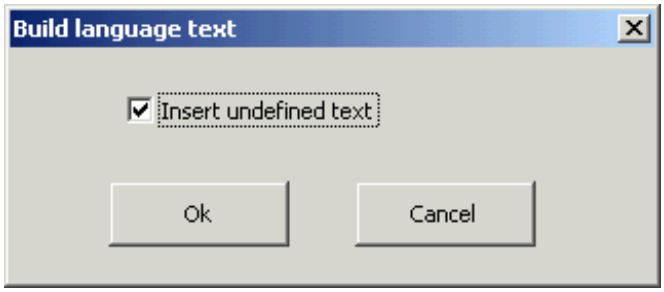
- Defines a new language in the EPAM application. Two additional language columns for Text/File and Font are added to all language-dependent worksheets. Language-dependent worksheets are message, alarm and project. An additional subdirectory with the relevant language name is defined in the current project directory. All language-dependent files (*.TXT, *.PCX, *.ICO, *.FNT) are saved in this subdirectory for the language concerned.

EPAM Delete Language macro

- Deletes a language defined with Define Language and its language-dependent subdirectory (prompt appears) from the EPAM application.

EPAM-macro „Build Language Texts”

- Automatical „translation function“ for multilingual applications. The language texts defined in worksheet „Text“ will be assigned automatically in all language dependent worksheets („Project“, „Alarm“, „Message“) in the columns Text/File of the corresponding language.



Option „Insert undefined text“
With activ option „Insert undefined text“, all undefined texts in the sheets „Project“, „Alarm“ and „Message“ (Column Text/File) will be inserted into the „Text” worksheet.

EPAM Add Text macro

- Adds the text of the current cell to the Text worksheet. The text can then be selected in the Text/File column using the pull-down field.

EPAM Add UserVar macro

- Adds the text of the current cell to the UserVar worksheet as a variable. The variable can then be selected in the VarValue, Limit1, Limit2 and VarState columns using the pull-down field.

EPAM-Macro „Build Recipes”

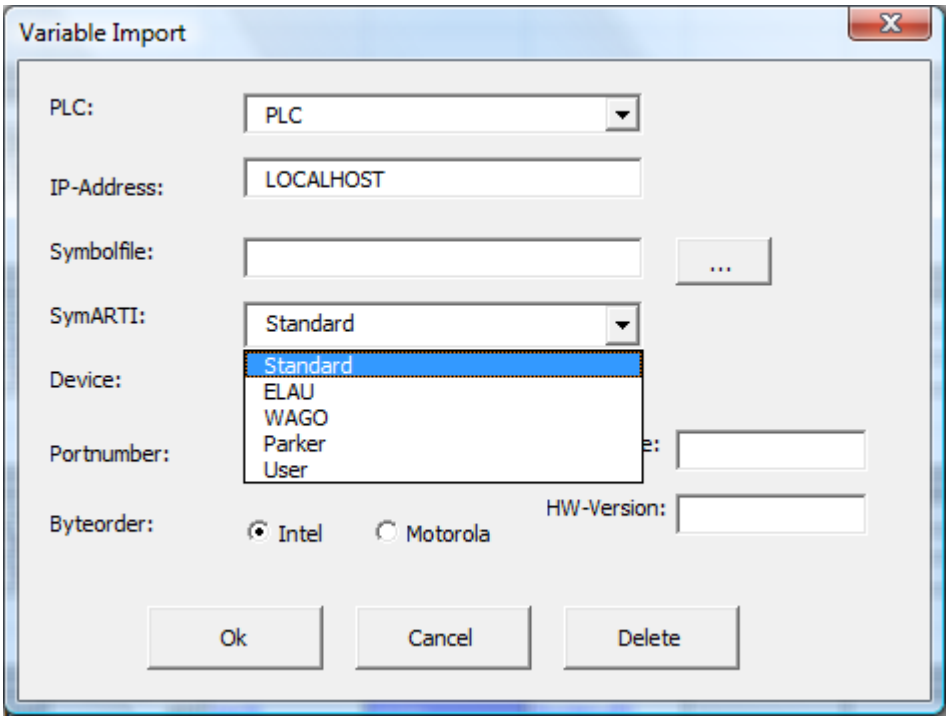
- Builds user defined recipes corresponding to the definition in sheet „Recipe“ column D and following columns.

EPAM Update Objects macro

- Updates the object properties of existing EPAM projects with the current pull-down fields and options.

EPAM PLC Variable Import macro

- Imports variable definitions from the CoDeSys project (symbol file *.SYM) to the current EPAM project. This will delete all the variables of the selected PLC in the UserVar worksheet and initiate a new import. All the variables in the EPAM project are then checked. Variables that are not defined, and type conflicts are displayed.



Within the field PLC it is possible to define additional names for remote PLCs and the corresponding IP addresses within a network. After this the symbol file of this PLC can be selected and the variables will be imported (Definitions of remote PLCs will be stored in worksheet „Hosts“).

Option „SymARTI“
Default Standard for all Grossenbacher devices. Additional parameters for the communication via SymARTI-Protokoll to 3rd party devices like ELAU-, WAGO-, Parker-PLCs

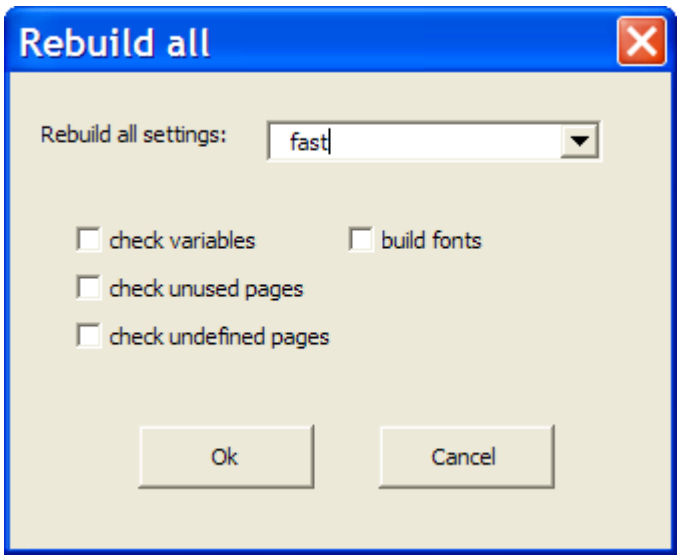
EPAM Build VarList macro

- Creates a textfile _DRVVLST.TXT with a list of all the variables used in the project. This list is required by the communication driver. When EPAM is started on the target system, an image of all configured variables is generated and all of them are read.

EPAM Rebuild Project macro

In Dialog „Rebuild all“ the following options can be selected:

- Fast: saves only all worksheets
- Complete: performs a complete Rebuild including Build Fonts
- Check variables: checks variable definitions and data types (only necessary after variable changes)
- Build Fonts: builds all fonts (only necessary after font changes)
- Check unused pages: check project for unused pages
- Check undefined pages: check project for undefined pages

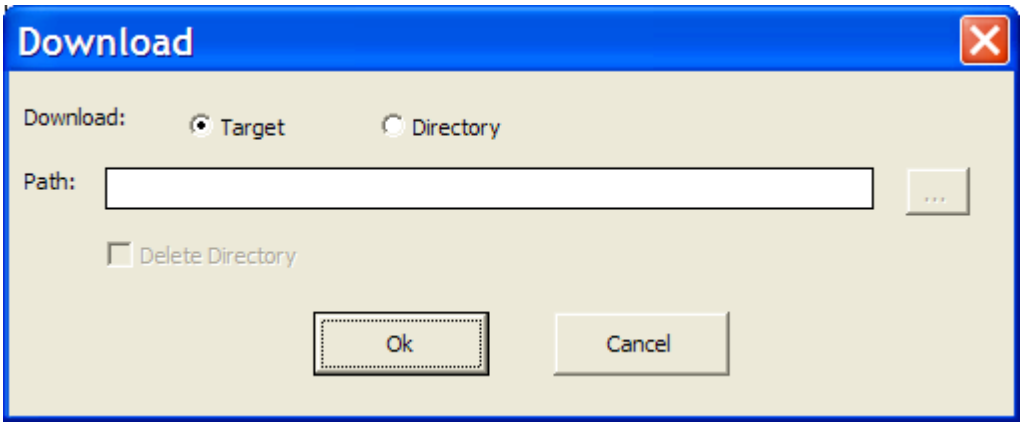


EPAM-Macro „Rebuild Project”:

- Compiles the entire project and saves all worksheets in Text (tab delimited) format.
- Saves all languages
- Checks whether all the files used are present (images, fonts, text files)
- Creates the variables list (macro: Build VarList)
- Checks the DataLog size used and the RAM drive size available
  - DataLog size > RAM drive size

- Starts the project download (EPAM macro: Download Project)

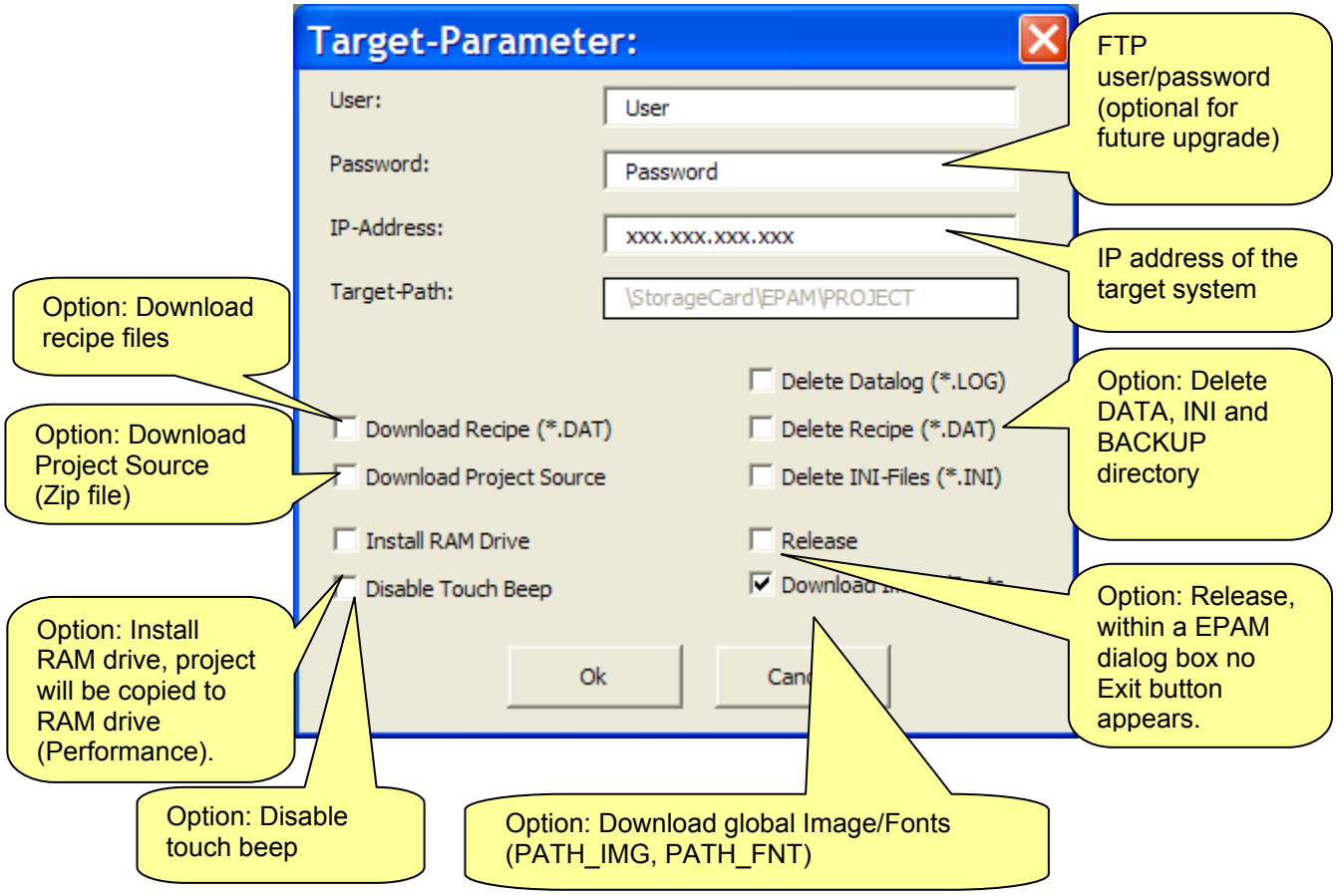
EPAM Download Project macro



The project can be downloaded to the target or copied into a directory. In setting „Directory“ a path can be selected with the Button „...“ (Default: subdirectory “Target“ within the current project directory).
Bei den Zielsystemen PC bzw. PocketPC ist nur ein Download in ein Verzeichnis möglich!

Download to Target

- Creates a list of all files (*.PCX, *.ICO, *.TXT, *.FNT) used in the project in the _FileLst system worksheet and transfers them via Windows FTP to the target system. For this, the target system must be connected to the development computer via Ethernet. The download parameters can be entered in the dialog box below:

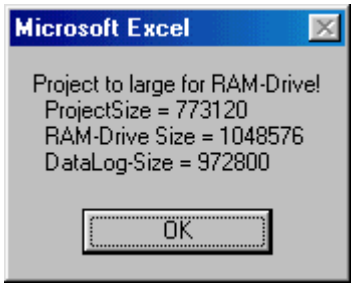


Project directory
The project directory name on the target system (target path) is predefined as “PROJECT”. The last project loaded is started. Existing projects will be overwritten.

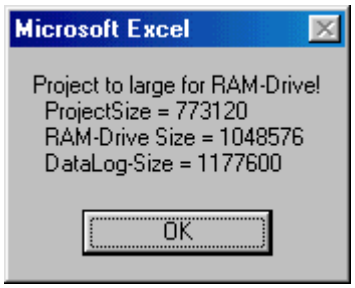
Project download
The project is loaded in the directory ..\EPAM_DNLD_, EPAM is closed automatically, the project directory is deleted and the download directory is renamed to “PROJECT”. A project restart is then carried out and EPAM is started with the new application. So **during download of a project the required space on the CompactFlash is two times of the project size.**

Ram drive option
The Install RAM Drive option can be activated for speed optimization depending on the DataLog and project size:

- DataLog size and project size < = RAM drive size
  ⇒ Data logging and project can be operated from the RAM drive
- DataLog size < RAM drive size but DataLog size and Project size > RAM drive size
  ⇒ Only data logging can be run from the RAM drive



- DataLog size > RAM drive size
  ⇒ Neither data logging nor project can be run from the RAM drive



Download Image/Fonts
Images and fonts can be stored global (project independend) in a separate directory. These directories can be defined with PATH_IMG= and PATH_FNT= within worksheet EPAM (EPAM.INI). If at least one path is defined, the option Download Image/Fonts appears within the download dialog. If the option is inactive, global images and/or fonts will not be downloaded to the target!
Note!
All global files will not be loaded into RAM-Drive! Performance is therefore slower!

Delete INI files option
The Delete INI files option should be activated if a new project is loaded onto the target system. Otherwise the INI files may not match the INI files of the new project. In this case, the INI files in the EPAM backup directory are also automatically deleted.
INI files contain values of system variables and the Alarm history.

EPAM Upload Project macro

- Upload of the entire project (*.PCX, *.ICO, *.TXT, *.FNT)

During a project upload, a request will appear to saver the project. This saving of a file in any directory is simply for determining the directory in which the required project is then to be saved. A project upload can only be carried out if a project download has already been executed with the Download Project Source option and the Zip file is present on the target system.

EPAM Grouping macro

- Standard Excel Group command is used to organize the Excel spreadsheet. In EPAM projects, for example, this command can combine all the lines of a screen page. The screen page can then be shown or hidden on the left next to the spreadsheet.

The EPAM Update Objects macro undoes groupings

EPAM Ungroup macro

- Standard Excel Ungroup command

6.5.2 EPAM Objects toolbar

EPAM NewPage object

- Creates a new page object and inserts it into the current line. The name of the screen page can be entered in the displayed dialog box. The name of a screen page must be unique and must not contain any special characters apart from "_".

EPAM NewButton macro

- Creates a new button object and inserts it into the current line.

EPAM NewSwitch macro

- Creates a new switch object and inserts it into the current line.

EPAM-„NewDropDownList" macro

- Creates a new DropDownList object and inserts it into the current line.

EPAM NewRadioButton macro

- Creates a new radio button object and inserts it into the current line.

EPAM NewVariable macro

- Creates a new variable object and inserts it into the current line.

EPAM NewBar macro

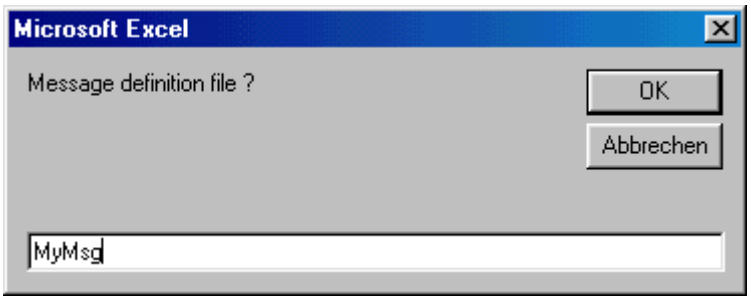
- Creates a new bar object and inserts it into the current line.

EPAM NewSignal macro

- Creates a new signal object and inserts it into the current line.

EPAM NewMessage macro

- Creates a new message object and inserts it into the current line. The name of the message definition file can be entered in the displayed dialog box. The name of the message definition file must not contain any special characters apart from "_", and must not exceed 8 characters in length (ISO 9660, 8.3 with restricted font). A Message worksheet will then be created with the entered name. The messages can then be defined in this worksheet. It is possible to define different message objects which use the same definition file. In this case no new message worksheet will be created and the existing one will be used.



EPAM NewMeter macro

- Creates a new meter object and inserts it into the current line.

EPAM NewDiagnoseSignal macro

- Creates a new Diagnose signal object and inserts it into the current line.

EPAM NewAlarmList macro

- Creates a new Alarm list object and inserts it into the current line.

EPAM NewRecipeList macro

- Creates a new recipe list object and inserts it into the current line.

EPAM NewTextList macro

- Creates a new text list object and inserts it into the current line. The ASCII text file to be displayed can be selected using the displayed Open File dialog. This file is always copied to the current project directory.

EPAM „NewHTMLBrowser"macro

- Creates a new HTML-Browser-Object object and inserts it into the current line.
(only Windows/WindowsCE)

EPAM „NewRemoteControl" macro

- Creates a new RemoteControl object and inserts it into the current line.

EPAM NewScrollList macro

- Creates a new Scroll list object and inserts it into the current line.

EPAM NewTrend macro

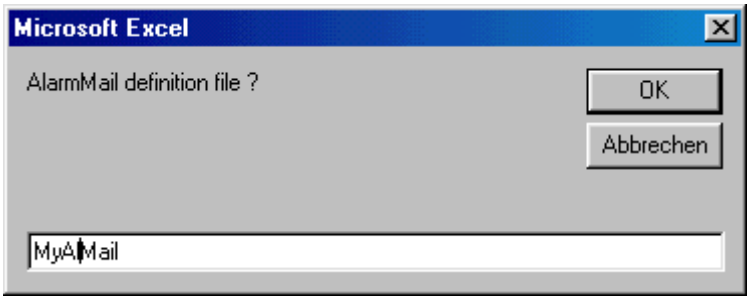
- Creates a new trend object and inserts it into the current line. The name of the trend parameter file can be entered in the displayed dialog box. The name of the trend parameter file must not contain any special characters apart from "_", and must not exceed 8 characters in length (ISO 9660, 8.3 with restricted font). A Trend worksheet will then be created with the entered name. The trend parameters can then be defined in this worksheet. Different trend objects can share the same trend parameter file.

EPAM NewAlarm macro

- Creates a new alarm object and inserts it into the current line. The name of the alarm definition file can be entered in the displayed dialog box. The name of the alarm definition file must be unique, must not contain any special characters apart from "_", and must not exceed 8 characters in length (ISO 9660, 8.3 with restricted font). An Alarm worksheet will then be created with the entered name. The alarm messages can then be defined in this worksheet.

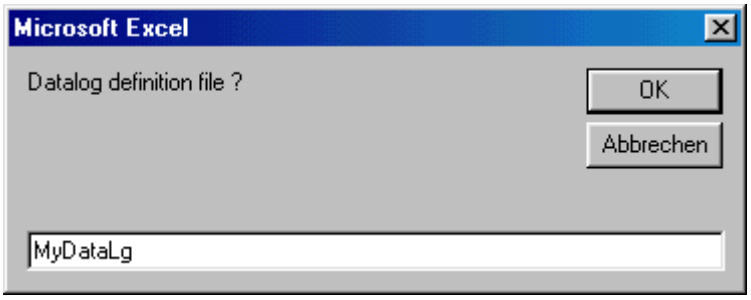
EPAM NewAlarmMail macro

- Creates a new alarm mail object and inserts it into the current line. The name of the alarm mail definition file can be entered in the displayed dialog box. The name of the alarm mail definition file must be unique, must not contain any special characters apart from "_", and must not exceed 8 characters in length (ISO 9660, 8.3 with restricted font). An AlarmMail worksheet will then be created with the entered name. The alarm mail parameters can then be defined in this worksheet.



EPAM NewDataLog macro

- Creates a new DataLog object and inserts it into the current line. The name of the DataLog definition file can be entered in the displayed dialog box. The name of the DataLog definition file must be unique, must not contain any special characters apart from "_", and must not exceed 8 characters in length (ISO 9660, 8.3 with restricted font). A DataLog worksheet will then be created with the entered name. The DataLog parameters can then be defined in this worksheet.

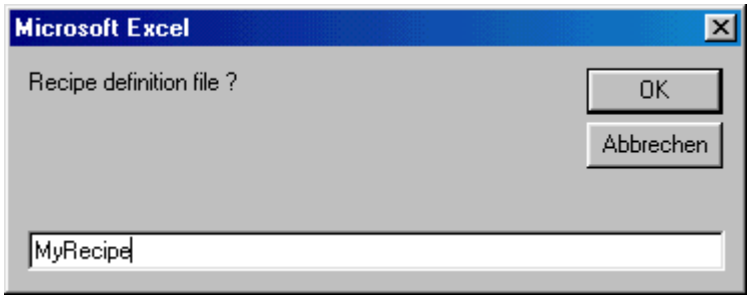


EPAM NewPassword macro

- Creates a new password object and inserts it into the current line.

EPAM NewRecipe macro

- Creates a new recipe object and inserts it into the current line. The name of the recipe definition file can be entered in the displayed dialog box. The name of the recipe definition file must be unique, must not contain any special characters apart from "_", and must not exceed 8 characters in length (ISO 9660, 8.3 with restricted font). A recipe worksheet will then be created with the entered name. The recipe variables can then be defined in this worksheet.

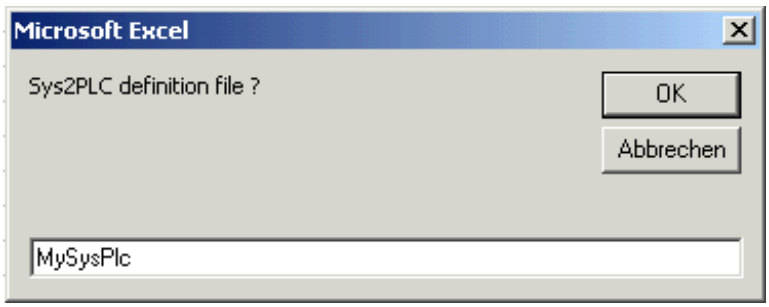


EPAM NewScreenSaver macro

- Creates a new screensaver object and inserts it into the current line.

EPAM „NewSys2PLC" macro

- Creates a new Sys2Plc object and inserts it into the current line. The name of the Sys2Plc definition file can be entered in the displayed dialog box. The name of the Sys2Plc definition file must be unique, must not contain any special characters apart from "_", and must not exceed 8 characters in length (ISO 9660, 8.3 with restricted font). A Sys2Plc worksheet will then be created with the entered name. The Sys2Plc variables can then be defined in this worksheet.



EPAM demos
Some Excel spreadsheet samples are provided in the EPAM directory (EPAM):
EPAM\SAMPLES

Graphically displaying the Excel spreadsheet definitions
After EPAM is started on the development system, the definitions of the individual objects from the Excel spreadsheet can be displayed column by column in text format within the image using the key combination “Alt I”.

Pressing the Print Screen key enables a copy of the EPAM screen output to be printed in Windows.

The key combination “Alt p” can be used to create a screen shot of the current screen content in the form of a PCX image. The file name is created from the first 4 characters of the page name, and the remaining 4 characters form a consecutive number (e.g. STAR0000.PCX).

6.5.3 EPAM Wizard toolbar

The EPAM Wizard provides you with a user-friendly way of changing the positions *X, Y* and the dimensions *DX, DY* of objects.

EPAM Wizard Refresh macro

- Opening the EPAM Wizard. EPAM Wizard visualizes the active screen page in the EPAM Wizard window. Objects of this screen page can then be edited. Changes are made directly in the Excel spreadsheet. If you wish to visualize and edit other screen pages, move to the required screen page in the Excel spreadsheet and run the EPAM Wizard Refresh macro once more.

Changes which are carried out in the Excel spreadsheet with the EPAM Wizard active are not automatically carried out in the EPAM Wizard window. To refresh the EPAM Wizard window, run the EPAM Wizard Refresh macro once more.

EPAM Wizard Align Left macro

- Left-justifies objects in relation to the last selected object. Several objects can be selected with the Ctrl key held down.

EPAM Wizard Align Top macro

- Top-justifies objects in relation to the last selected object. Several objects can be selected with the Ctrl key held down.

EPAM Wizard Align Right macro

- Right-justifies objects in relation to the last selected object. Several objects can be selected with the Ctrl key held down.

EPAM Wizard Align Bottom macro

- Bottom-justifies objects in relation to the last selected object. Several objects can be selected with the Ctrl key held down.

EPAM Wizard Spacing Horizontal macro

- Arranges selected objects horizontally with equal spacing. Several objects can be selected with the Ctrl key held down.

EPAM Wizard Spacing Vertical macro

- Arranges selected objects vertically with equal spacing. Several objects can be selected with the Ctrl key held down.

EPAM Wizard Format Widths macro

- Changes the object width (DX) of selected objects on the basis of the last selected object. Several objects can be selected with the Ctrl key held down.

Format Heights macro

- Changes the object height (DY) of selected objects on the basis of the last selected object. Several objects can be selected with the Ctrl key held down.

EPAM Wizard Undo macro

- Undoes previous EPAM Wizard actions.

EPAM Wizard Full Screen macro

- EPAM Wizard window is shown in the full size (maximum height of the EPAM Wizard window corresponds to the height of the Excel spreadsheet area).

EPAM Wizard Half Screen macro

- EPAM Wizard window is shown in half size.

Small Screen macro

- EPAM Wizard window is shown in small size.

EPAM Wizard AutoSize macro

- EPAM Wizard window size is adapted automatically.

EPAM Wizard macro 100%

- Objects in the EPAM Wizard window are shown with a zoom factor of 100%.

EPAM Wizard macro 150%

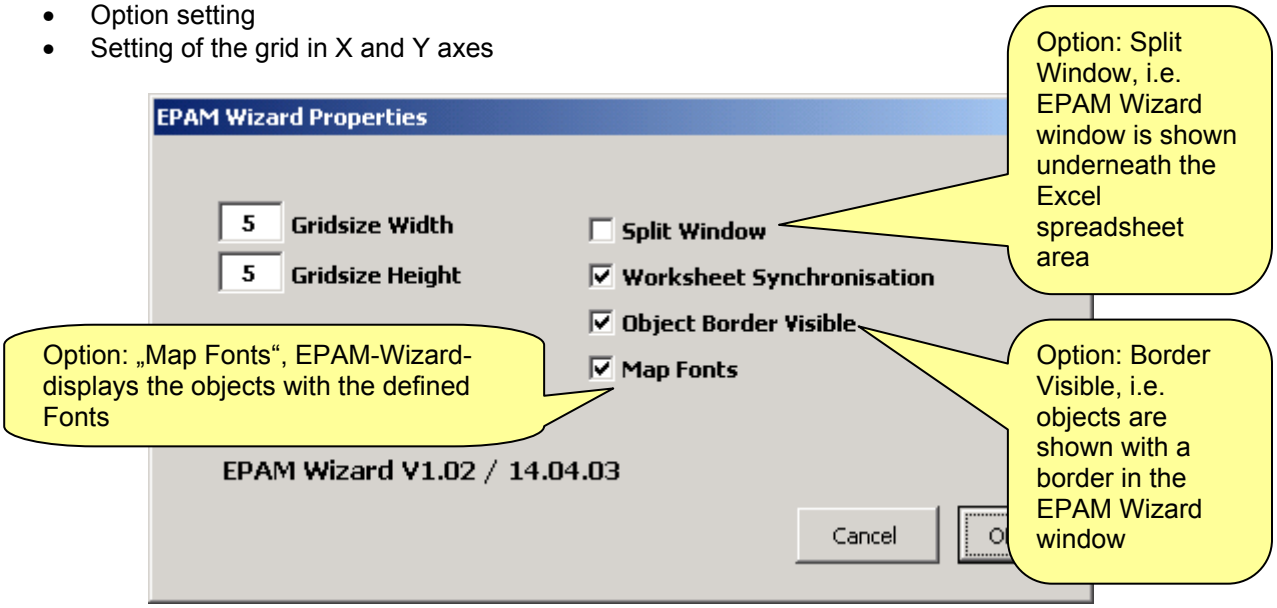
- Objects in the EPAM Wizard window are shown with a zoom factor of 150%.

EPAM Wizard macro 200%

- Objects in the EPAM Wizard window are shown with a zoom factor of 200%.

EPAM Wizard Properties macro

- Version information on the currently used EPAM Wizard macro.
- Option setting
- Setting of the grid in X and Y axes



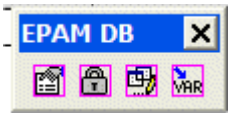
Worksheet synchronization
If the Worksheet Synchronization option is active, the EPAM Wizard window will transfer any modifications made directly to the Excel worksheet. (Default)

Copy and delete objects within Wizard
With Ctrl-C/Ctrl-V keys it is possible to copy the selected objects within the Wizard. The delete key removes the selected objects.

6.5.4 EPAM-toolbar „EPAM DB“

The EPAM-DB toolbar contain the following macros (s.a. documentation EPAM-DB-Server):

- DB-Setup ...Setup of EPAM-Database-Servers and Definition of VarLog-worksheet
- DBPasswd ...Object DBPasswd for central User/Password handling on EPAM DB-Server
- DBTracer ...Object DBTracer for logging of variables on EPAM DB-Server
- Import Varlog ...Import of variables into VarLog-List. All variables within the VarLog-List will be logged on the EPAM DB-Server, if the value is changed by the user.

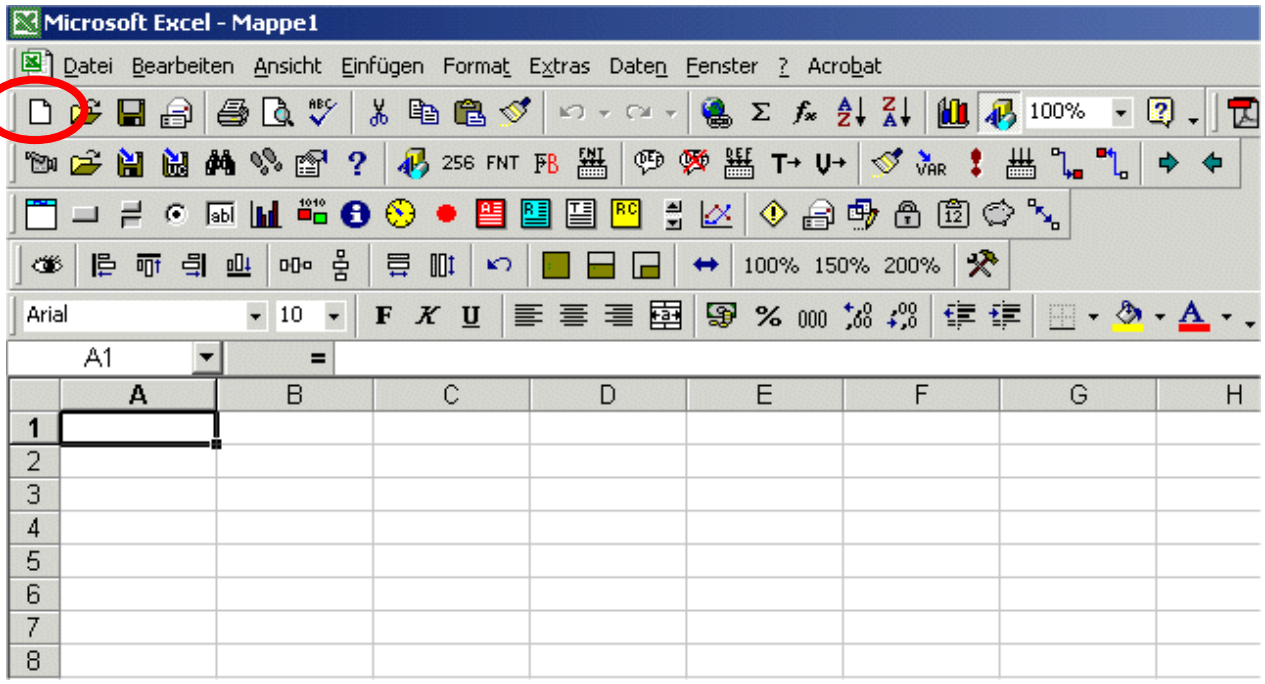


For details please see separate documentation to EPAM-DB-extension!
(at the moment only available for VxWorks)

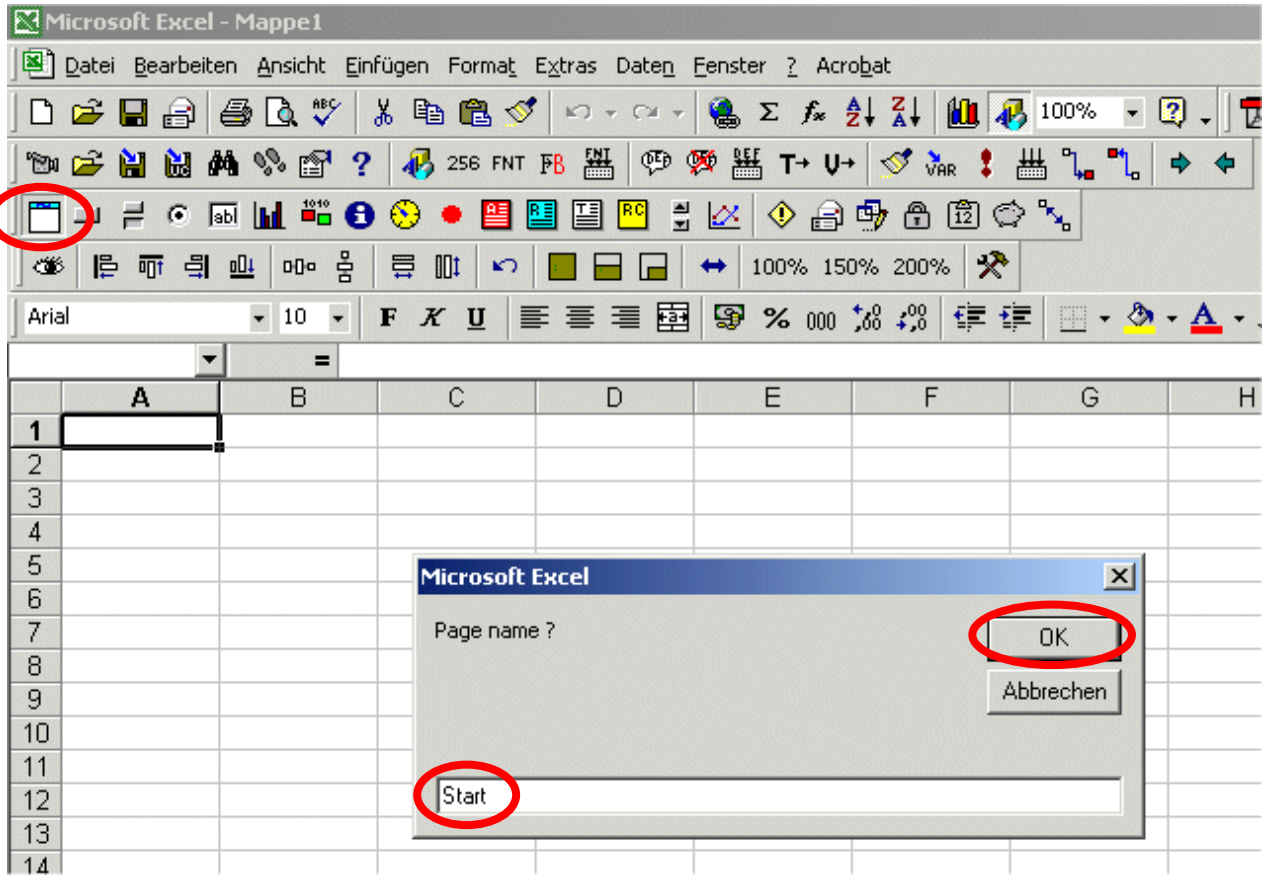
6.6 A little project from A-Z

The process to create an EPAM project is shown in the following steps:

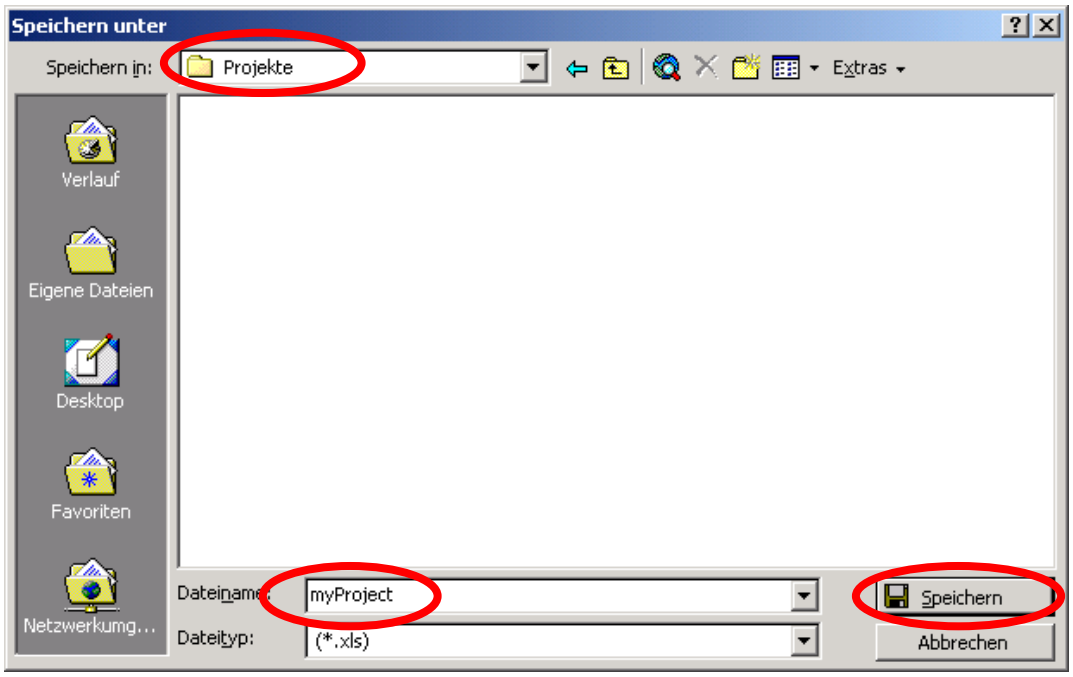
1. **Open a new project**: open a new EXCEL worksheet



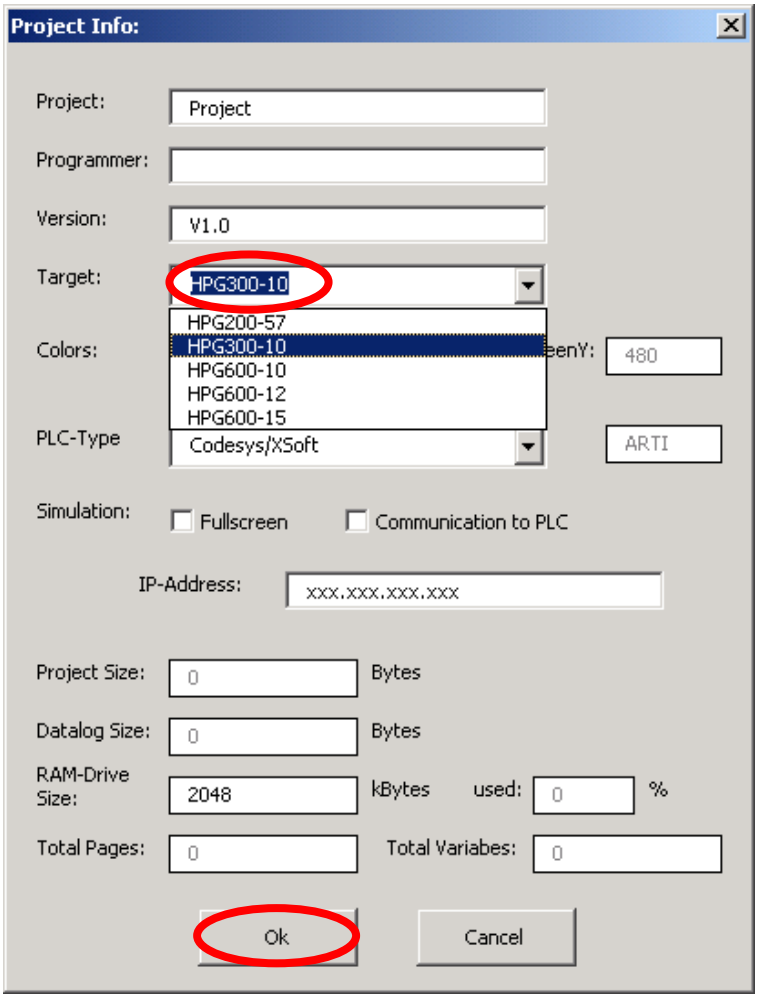
2. **Define pages and objects**: define a new page with „NewPage“ macro and enter a unique name for the page: e.g. „Start“



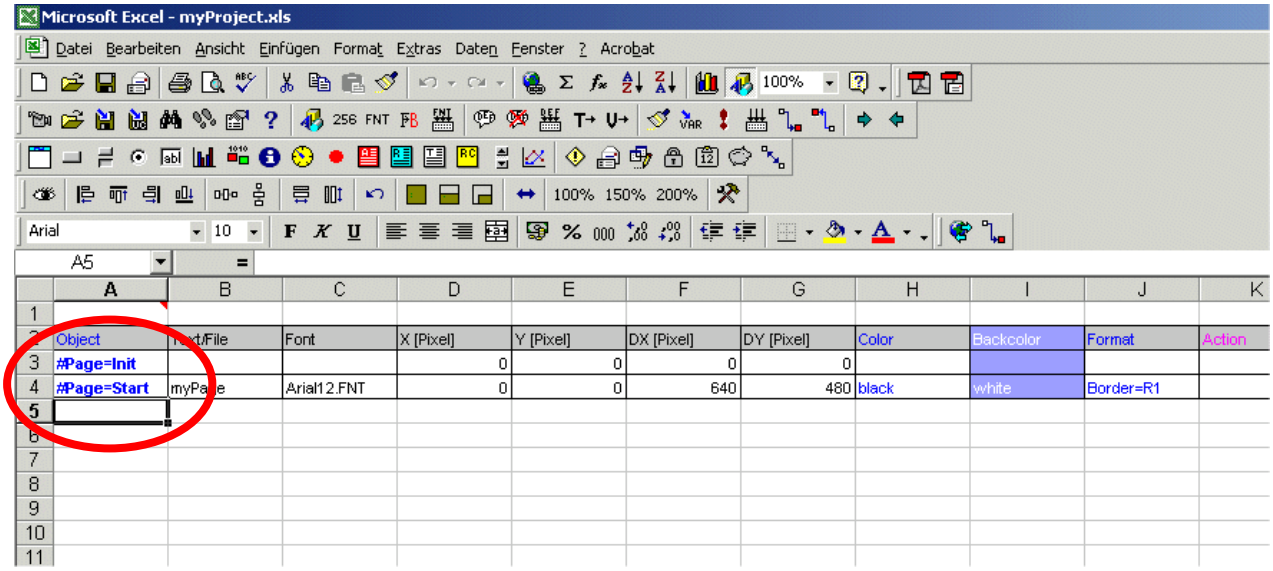
After this you will be asked to save the project (EXCEL-sheet). Select a project directory and enter a name for the EXCEL-sheet...



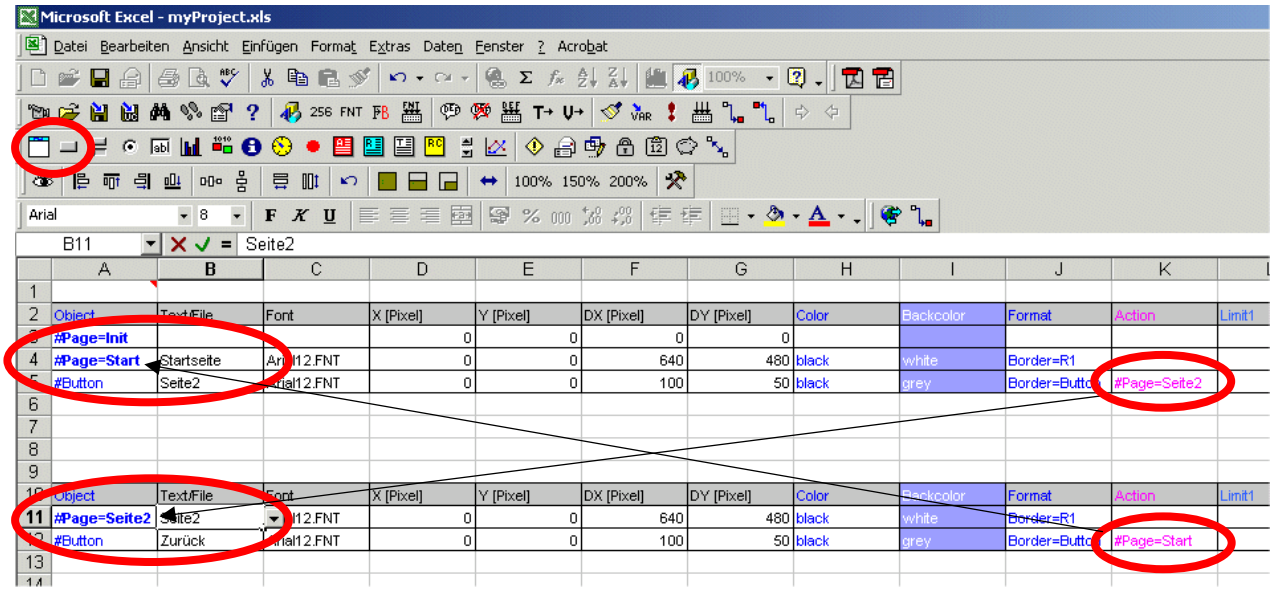
Then the „Project Info“ menu appears, where you can enter project information and select the target system: e.g. HPG300-10



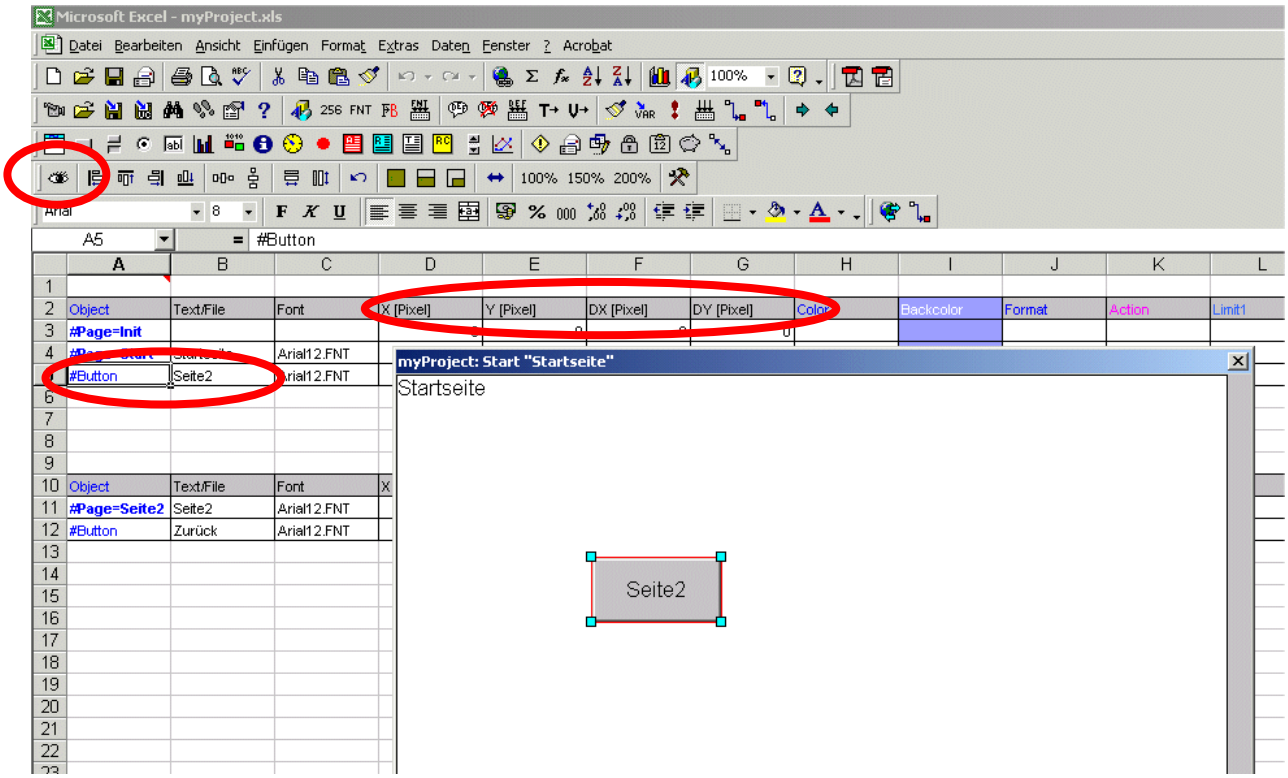
Now you can see two objects **#Page=Init** and **#Page=Start** in your Excel sheet. The cursor is set to the next new line.



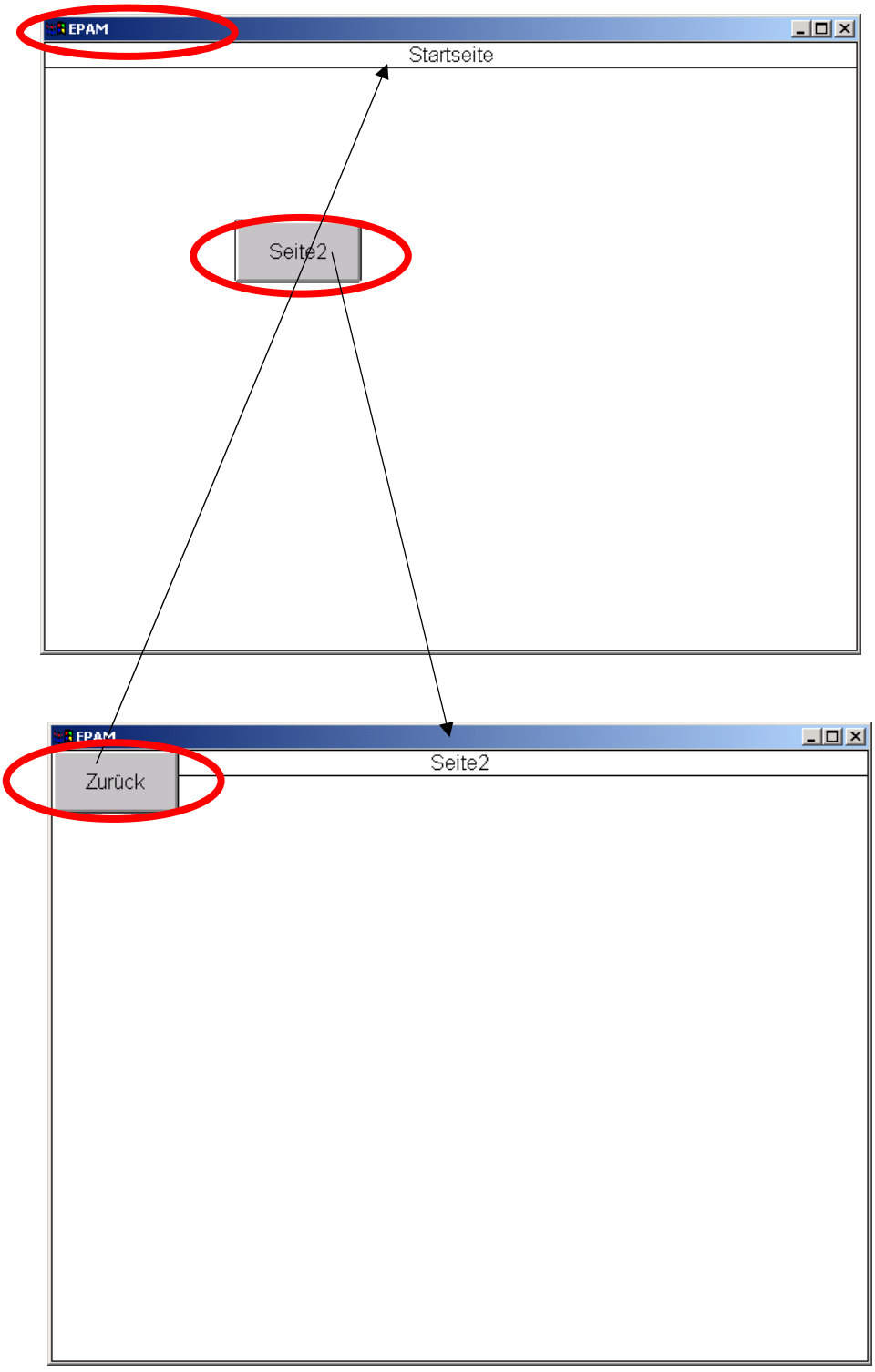
There you can define further objects for the page „Start“ with the macros „NewButton“, „NewVariable“, etc. To create a further page leave at least one line empty after the last object of page start and call macro “NewPage” again and enter a name for the new page e.g. “Page2”. The Button-Action #Page=name allows to change to Page2 and back to Page Start. Replace “name” with the name of your pages.



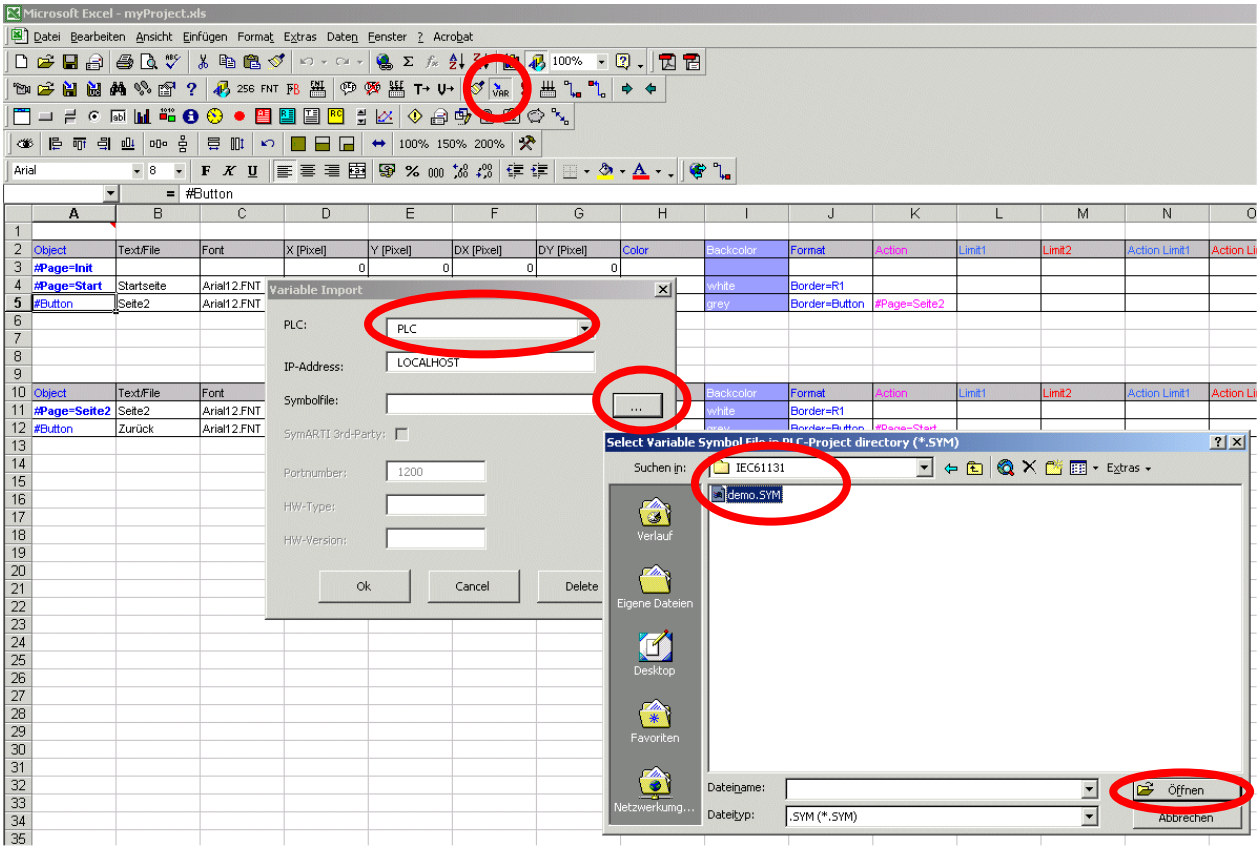
3. **Definition of object properties**: Within the columns you can change the different object properties with pulldown menus. To define the position and the size of the objects you can use the **EPAM-Wizard** („Refresh“ macro). Select a cell within the page you which to change. The selected page will be displayed within the Wizard and the objects can be moved and resized with the mouse. Changes will be done directly in the corresponding EXCEL columns (X,Y,DX,DY).



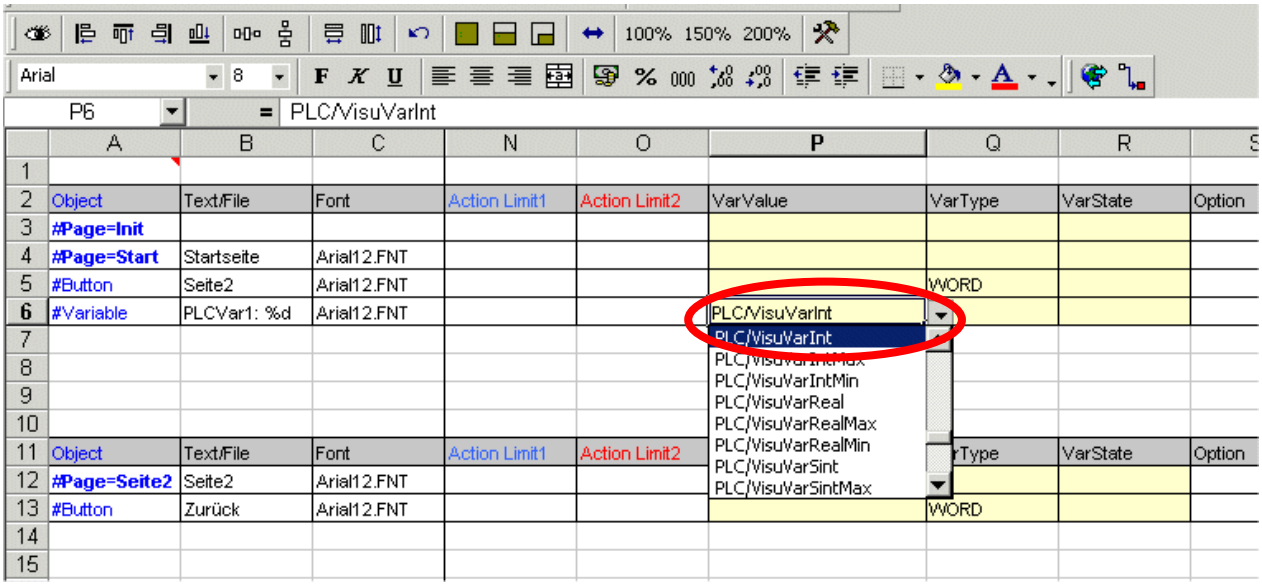
4. **Simulation of the application in development environment**: By the help of „Start EPAM“ macro you can start the simulation under Windows and test your application. It is also possible to activate the communication to the PLC (see “Project Info” option: communication to PLC) and test the application with a running communication. For that the PLC program has to be loaded to the target before.



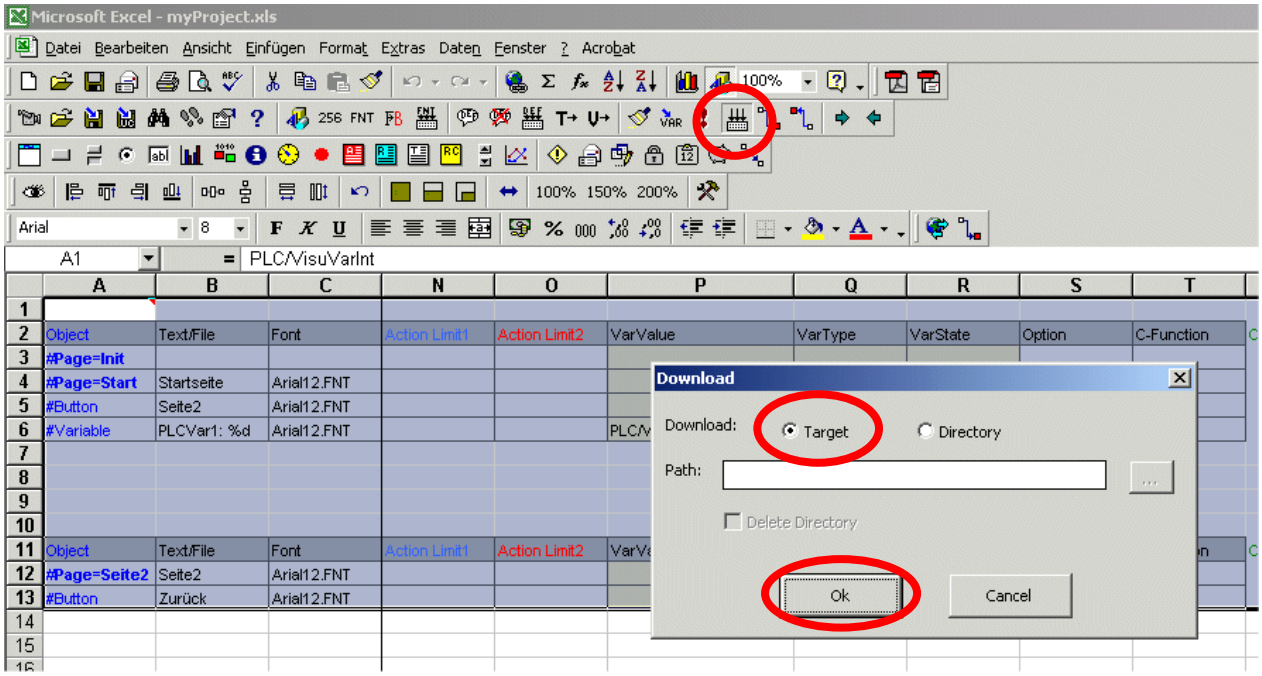
5. **Import of PLC-variable definitions**: The „PLC Variable Import“ macro allows to import variable definitions directly from the CoDeSys without entering the variable names twice. (see also chap. 2.4 Settings in the CoDeSys development environment, P.12). You will be asked to select a PLC (Default: the local PLC is defined, but it is also possible to define further PLCs which are connected within a Ethernet network) and the symbol file, which contains the variable definitions (*.SYM-File).



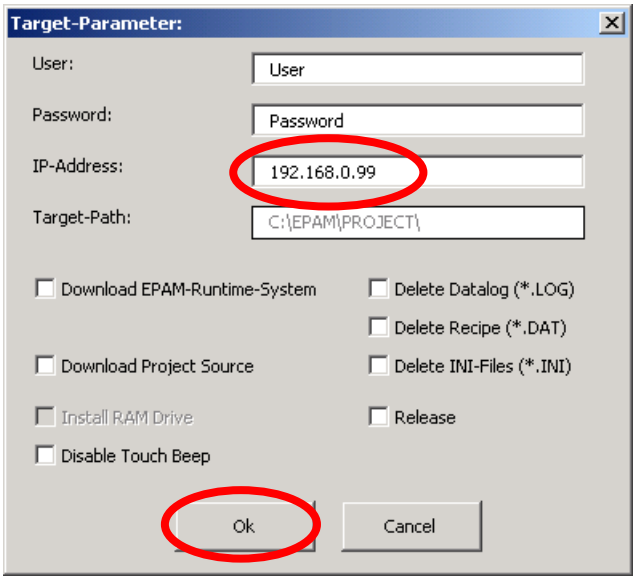
With „Ok“ the variables will be imported into the worksheet „UserVar“ and can than be selected and assigned to an object by the pulldown menus e.g. in column „VarValue“. Additionally the datatyp of the object has to be defined (Column VarType). The object datatype and the variable datatype must be the same. Typeconflicts will be tested during a project rebuild („Rebuild All“ macro).



6. **rebuild project and download to the target system:** the „Rebuild all“ macro creates and tests the whole project. After this you will be asked to where you want to download the project („Download“ menu). The project can be downloaded to the target (default) or into a local directory.



When Target (Default) is seleced the menu „Target Parameter“ appears. There you have to enter the IP-Address of the target system. Press „Ok“ to start the download. After a successful download the EPAM-Application will start automatically with the new project. Before the PLC-project should be loaded to the target, otherwise it is not possible to communicate with the PLC and display the variables which are defined wihtin the EPAM project.



If you have no connection to the target system, please chech the network settings. The development PC must have an IP-Address whithin the same subnet as the target system (first 3 numbers of the IP-Address are the same, the last number is different!).
You can check the network connection with the following command in the command line: ping xxx.xxx.xxx.xxx (xxx = IP-Address of the target system).

7 Object definition

The object attributes in the following tables are described in the first column to provide a better overview. Every object is described in a line, i.e. each attribute is shown in a separate column.

The length of a line is currently restricted to 512 characters per language! For a Unicode language the length is max 85 characters (=512/6)!

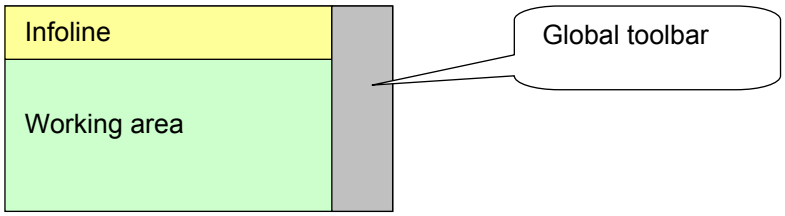
The notation used has the following meaning:

Text with grey background	...no or permanently pre-defined attributes. New features of the version 3.30 are blue
Text in bold type	...reserved words
Italic text	...are user entries

7.1 Page object

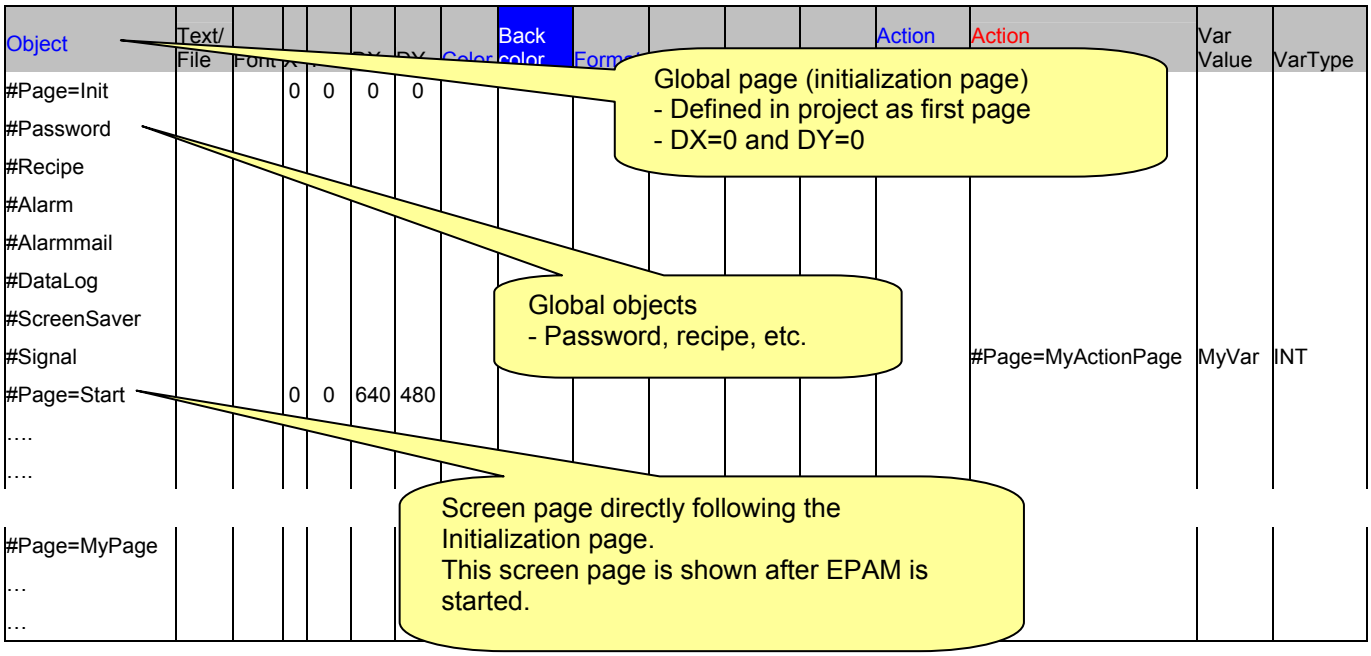
#Page=*Name*		**Freely definable, unique name of the screen page**
Text/File	*Image.***PCX**	• Name of a PCX image file for the background image
	Text	• Text string as page title (centered text display) Text can be divided up over several lines with line delimiter '|' (ASCII character 124 or 7CH). In this case the text is left-justified.
Font	*Font.***FNT**	• Optional font for the title
X,Y,DX,DY	*Integer values*	• Position, width and height of screen page (in relation to top left!)
Color	*0-15 or color name*	• Color of title text
Backcolor	*0-15 or color name*	• Background color of screen page
Format		• No entry means no border
	Border=Button	• Button border type
	Border=Input	• Input field border type
	Border=R*x*	• Border type, rectangle with width *x* pixels (1, 3, 5, etc.)
	Border=Shadow	• Border type, rectangle with shadow (3D effect)
	Border=Signal	• Signal field border type
Action		
Limit1		
Limit2		
Action Limit1		
Action Limit2		
VarValue		
VarType		
VarState		
Option		• Without input normal screen page
	ID=*x*	• Pagenumber which will be stored in system variable s_pageidx when page is activ (s.a. system variables)
	Page=Dialog	• Inputs only permissible in current screen page
	Transparency=*colorname*	• Name of the transparent color within an image
C function	*C function name*	• Name of C function (see Integration of C functions)

Frames
A screen page can be divided into different frames by defining several page objects without a newline. Screen contents in the individual frames can then be modified independently of each other (see also Demo: „#Page=ObjectMeter“)
So it is possible to define toolbars or infolines central one time within the project and page independend.
Example:



Window handling
Screen pages with different dimensions are stacked on top of each other and shown and updated on the screen at the same time (window handling). Objects that are covered in part or completely by superimposed screen pages are marked as invisible and no longer updated. All other objects remain fully accessible (Exception: Option Page=Dialog). A covering screen page (e.g. Help window) can only be closed via the Close action if the screen page below it is larger (DX or DY). The ‘close’ action is not effective if only one visible screen page is active.

7.1.1 Designing global objects



Global objects
Global objects must be designed **in the first screen page** in the Project worksheet (initialization page). This page must be defined with the dimensions DX=0 and DY=0. The page is thus defined as a global screen page and is always active. This enables screen page changes to be implemented also at any time using variable values and their action limits

7.2 Button object

#Button		Non-latching, touch-activated area
Text/File	*Up*.**PCX**,*Down*.**PCX**	• Name of PCX image files for states Not pressed, pressed, separated by ‘,’.
	Up.**ICO**,*Down*.**ICO**	• Name of PCX image files with the suffix .ICO for states Not pressed and pressed, separated by ‘,’. Icons are displayed in the center of the button.
	UpText,DownText	• Text strings for states Not pressed and pressed, separated by ‘,’. Can be divided over several lines with line delimiter ‘|’ (ASCII character 124 or 7CH).
	Icon.**ICO**	• PCX image with suffix .ICO • Is shown centered in the button
	Text	• Text string (text output is centered). Text can be divided up over several lines with line delimiter ‘|’ (ASCII character 124 or 7CH).
		• If neither image nor text are entered, it is a touch-active field, e.g. within a graphic; the field is inverted when pressed
Font	*Font*.**FNT**	• Optional font for the text
X,Y,DX,DY	*Integer values*	• Position, width and height of button (in relation to top left!)
Color	*0-15 or color name* *UpColor,Downcolor*	• Color of text for states Not pressed and pressed
Backcolor	*0-15 or color name* *UpBackColor,DownBackColor*	• Background color of the button for states Not pressed and Pressed
Format	**Border=Button**	• Button border type with text and icon output with fixed definition
	Invisible	• Invisible touch-active field (not inverted when pressed)
Action0, Action1, … or Action1 & Action2 & ...	**General actions**	
	#Page=*Name*	• Screen page change to screen page *Name*
	#PagePrev	• Screen page change to last active page
	#PageHome	• Screen page change to the first page (Startpage)
	Close	• Close screen page (window)
	Close=*Name*	• Close screen page (window) *Name*
	EjectVolume(Drive #Page=eject_failed #Page=eject_ok)	• Log off removable devices (e.g. USB-Memory-sticks; only Windows) Drive ...name of the removable device (e.g. StorageCard2 or F:) #Page=eject_failed …optional Page will be displayed if action failed (log off of removable devices is only possible if no program accesses the device!) #Page=eject_ok …optionale Page will be displayed if action was successful (requires also definition of #Page=eject_failed)
	Exit	• Exit program (back to operating system)
	FileCopy(dst=path\file.ext src=path\file.ext)	• copy file „src“ to „dst“
	Key=*key code*	• Simulation of a key code or key name (see also Keyboard table): ASCII characters or \xnnnn (HEX keyboard code)
	Language=default	• Online language selection for default language
	Language=*name*	• Online language selection for language *name*

	PLCcmd=[[/Driver/][Host]:Command	• Activate PLC-Command (depending on communication driver) • Commands: o CreateBootProject o ResetCold o ResetOriginal o ResetWarm o Start o Stop
	PrintScreen	• Printscreen to default printer (only Windows)
	Reboot	• Restart the system (system depending!)
	SetIndex=x	• Set index for indexed variable access
	SetVar=*x*	• Set variable value to x (strings must be defined with enclosing single apostrophe, e.g. 'String') x can also be the name of a system variable
	SetVar+*x*	• Increment variable value by x
	SetVar*x*	• Decrement variable value by x
	SetVar=NotVar	• Invert variable value (0/1)
	System=myprg.exe	• Start program (only Windows)
	Msg=*x*	• Output message with number x
	PWL=*x*	• Set (or Reset) password level to x
	TipVar=*x*	• Set variable value to x for as long as the button is pressed, the variable is then reset to 0
	Touch_calibrate	• Resistive-Touch-screen calibration
Action0, Action1, … or Action1 & Action2 & ...	**System actions**	
	Backlight=x	• Set backlight (0-100%)
	Backlight+x	• Increment backlight by x
	Backlight x	• Decrement backlight by x
	CFGINI=Read	• Read CONFIG.INI file (IP addresses)
	CFGINI=Write	• Write CONFIG.INI file (IP addresses)
	Contrast=*x*	• Set contrast (0-100%) (only passive LCD)
	Contrast+x	• Increment contrast by x (only passive LCD)
	Contrast-x	• Decrement contrast by x (only passive LCD)
	GetDT	• Update all RTC system variables **s_tm_day**, **s_tm_mon**, etc.
	Save=SysVar	• Save system variables in sysvar.ini
	SetDate	• Set system time (values are transferred from RTC system variables)
	SetTime	• Set system date (values are transferred from the RTC system variables)
	Object-specific actions in conjunction with Scroll list object	
	ScrollX=*x*	• Move objects horizontally in Scroll list by x pixel(s)
	ScrollY=*x*	• Move objects vertically in Scroll list by x pixel(s)
	Object-specific actions in conjunction with Alarm/Alarm list object	
	AlarmDelete	• Clear alarm history
	AlarmExport=CSV	• Export of Alarmhistory as CSV-File into EPAM-Data directory C:\DATA
	AlarmFilter=activ	• Set alarm filter: Display active alarms
	AlarmFilter=activ|notquit	• Set alarm filter: Display active or unacknowledged alarms
	AlarmFilter=activ+notquit	• Set alarm filter: Display active and unacknowledged alarms
	AlarmFilter=all	• Set alarm filter: Display all alarms
	AlarmFilter=notquit	• Set alarm filter: Display unacknowledged alarms

	AlarmInfo=1 or 2	• Call alarm information for selected alarm
	AlarmQuit	• Acknowledge selected alarm individually
	AlarmQuitall	• Acknowledge all alarms
	AlarmSort=FIFO	• Sort alarm in alarm list: Oldest alarm first
	AlarmSort=LIFO	• Sort alarm in alarm list: Latest alarm first
	AlarmSort=Priority	• Sort alarm in alarm list: Alarm with highest priority (=lowest alarm number) first
	AlarmType=*myalarmtype*	• Set actual Alarmtype for Alarmlist (necessary if more then one alarm object is defined)
	Object-specific actions in conjunction with Recipe/Recipe List object	
	Csave=list	• Save element from recipe list (e.g. myRecipeType), with prompt if file exists (see Recipe object)
	Csave=*myrecipetype*	• Save recipe type (e.g. myRecipeType) with prompt if file exists, the name is removed from the system variable 's_*myrecipetype*_file'
	Delete=list	• Delete element from recipe list (e.g. myRecipeType)
	Delete=*myrecipetype*	• Delete recipe type (e.g. myRecipeType) the file name is removed from the 's_*myrecipetype*_file' system variable
	Load_dat=*DR:*	• Load all *.DAT recipe files from drive DR:
	Load=list	• Load element from recipe list (e.g. myRecipeType)
	Load=*myrecipetype*	• Load recipe type (e.g. myRecipeType) the file name is removed from the 's_*myrecipetype*_file' system variable
Action0, Action1, … or Action1 & Action2 & ...	**Recipe=delete**	• Delete recipe, the file name is removed from the 's_*myrecipetype*_file' system variable
	Recipe=load	• Load recipe, the file name is removed from the 's_*myrecipetype*_file' system variable
	Recipe=save	• Save recipe, file name and recipe name are removed from the system variables 's_*myrecipetype*_file' or 's_*myrecipetype*_name'
	Save_dat=*DR:*	• Copy all *.DAT recipe files to drive DR:
	Save=list	• Save element from recipe list (e.g. myRecipeType) existing files are overwritten
	Save=*myrecipetype*	• Save recipe (e.g. myRecipeType), existing files are overwritten, file name and recipe name are removed from the system variables 's_*myrecipetype*_file' or 's_*myrecipetype*_name'
	Sort=File	• Sort recipe list by file name
	Sort=Name	• Sort recipe list by recipe name
	Sort=Number	• Sort recipe list by recipe name numerically
	Sort=Time	• Sort recipe list by time
	Sort=Type	• Sort recipe list by recipe type
	Type=*myrecipetype*	• Set recipe type (e.g. myRecipeType)
	Type=off	• Reset recipe type (all)
	Object-specific actions in conjunction with DataLog object	
	LogDelete=*mydatalog*	• Delete DataLog file in the LOG directory
	LogSave=*mydatalog*	• Save the DataLog file in the DATA directory
	Object-specific actions in conjunction with Trend object	
	Online	• Switch trend in Online mode
	ShiftCursor=x	• Scroll trend by x values
	ShiftGrid=*x*	• Scroll trend by x time units
	ShiftPage=*x*	• Scroll trend by x pages
	Zoom	• Zoom Trend (reduce resolution of time axis by one unit)
	Zoom+	• Zoom Trend (increase resolution of time axis by one unit)
	ZoomX-	• Zoom Trend (reduce resolution of X axis by one unit)
	ZoomX+	• Zoom Trend (increase resolution of X axis by one unit)
	ZoomY-	• Zoom Trend (reduce resolution of Y axis by one unit)

	ZoomY+	• Zoom Trend (increase resolution of Y axis by one unit)
Limit1	*Value/variable name*	• Lower limit value, PLC variable or system variable for limit value (the button is disabled if the current value is outside of the limit value)
Limit2	*Value/variable name*	• Upper limit value, PLC variable or system variable for limit value (the button is disabled if the current value is outside of the limit value)
Action Limit1 Action Limit2	**#Page=***Name*	• Screen page change to screen page *Name*
	Alarm=*x*	• Trigger alarm x (x is alarm number)
	Backcolor=*x*	• Change background color to x
	Backlight=*x*	• Set backlight (0-100%)
	Close	• Close screen page (window)
	Close=*Name*	• Close screen page (window) *Name*
	Color=*x*	• Change color to x
	Contrast=*x*	• Set contrast (0-100%) (only passive LCD)
	Exit	• Exit program (back to operating system)
	FastFlash	• Set object status to flash at 2 Hz
	Flash	• Set object status to flash at 1 Hz
	Language=*name*	• Online language selection to language *name*
	Language=*s_mysysvar*	• Online language selection to language in *s_mysysvar*
	Load=*x*	• Load recipe file with name x
	Msg=*x*	• Output message with number x
	SetVar=*x*	• Set variable value to x
	SetVar=Limit1	• Set variable value to Limit1
	SetVar=Limit2	• Set variable value to Limit2
	s_mysysvar=x	• Set system variable value to x
VarValue	*Variable name*	• PLC variable and system variable
VarType	**BOOL**	• Bool data type (8-bit)
	BYTE	• Byte data type (8-bit)
	DINT	• Double integer data type (32-bit)
	DWORD	• Double word data type (32-bit)
	IEC_DT	• Datatype IEC_DT (32 Bit)
	IEC_TIME	• Datatype IEC_TIME (32 Bit)
	INT	• Integer data type (16-bit)
	REAL	• Floating point data type (32-bit)
	SINT	• Short integer data type (8-bit)
	STRING	• String data type 80 bytes (without definition: default 80 bytes)
	STRING:xx	• String data type xx bytes
	UDINT	• Unsigned double integer data type (32-bit)
	UINT	• Unsigned integer data type (16-bit)
	USINT	• Unsigned short integer data type (8-bit)
	WORD	• Word data type (16-bit)
VarState	*Variable name*	• PLC variable and system variable for object status
Option	**Key=***key code*	• Key code or key name (see also Keyboard table) to execute the Button-Action: ASCII characters or \xnnnn (HEX keyboard code)
	NoBeep	• Disable Touch Beep for this object
	Pos=Center	• Centered text
	Pos=Left	• Left-justified text
	Pos=Right	• Right-justified text
	PWL=*x*	• Password level required for enabling
	Scroll	• Position of the object can be changed with the scroll object

	Switch=*x*	• x = value range for Action0,Action1,... e.g.: <0:1:2..5:>5 permissible values: Constant number e.g. 5 <Number ...Less than >Number ...Greater than Number..Number ...Range from to : ...Separator
	Timeout=*x*	• Timeout in seconds after which action is executed automatically (repetitive) • Timeout=0 executes the action one time
	Transparency=*colorname*	• Name of the transparent color within an image
	Type=dynamic	• Button-Action will be executed also on mouse move events
C function	*C function name*	• Name of C function (see integration of C functions)

Screen keyboard
Action Key=*key code* is used to create screen keyboards. The appropriate key code is generated by actuating the mouse or via a touch-activated field. The keyboard is simulated in this way (see also keyboard table with key designations).
Example:
Key=a ...simulates the key “a”

Multi-lingual applications
The Language=*name* or Language=Default action enables online language selection to any language. Other languages are stored in appropriate subdirectories. *name* designates the subdirectory concerned.
To create a multi-lingual application, define a new language with the EPAM Define Language macro. A directory with the entered name (max. 8 characters) is created, and two new columns for Text/File and Font are added to all language-dependent worksheets.

Important!
The designation in the first line of this column corresponds to the directory name and is stored as a Language comment (designation of the language columns!) These designations must not be removed!

You can now define all language-dependent definitions in the appropriate language columns (the language-dependent definitions can be left empty, these are then configured automatically from the default language). You can also define other fonts if required.
All language-dependent files (*.TXT, *.PCX, *.ICO, *.FNT) must be saved in the appropriate subdirectory.
The next time that the Start EPAM macro is called in order to launch the interpreter, all language files will be created automatically.

Variable action
Several actions can be defined with a comma to separate them. In this case the current variable value (0, 1, 2, ...) determines the appropriate action to be executed. In this way, for example, different screen pages can be selected depending on the variable value concerned.

Example:
#Page=Value0,#Page=Value1
...Change to screen page Value1, if the object value is 1, otherwise to the screen page Value0

The ‘switch=’ option can be used to define value ranges for the individual actions.

Example:
#Page=Range0,#Page=Range1,#Page=Range3 Option:switch=<0:0..5:>5
...Change to the Range0 screen page if the object value < 0,
...Change to Range1 screen page, if the object value is in the range 0 to 5
...Change to the Range2 screen page if the object value is > 5.

7.3 Switch object

#Switch		Latching, touch-sensitive area
Text/File	*Image0*.**PCX**,*Image1*.**PCX**,…	• Name of PCX image files for states 0,1, separated by ‘,’.
	Icon0.**ICO**,*Icon1*.**ICO**,…	• Name of PCX image files with the suffix .ICO for states 0,1, separated by ‘,’, are shown as icons that are centered in the button.
	Text0,Text1,...	• Text strings for states 0,1 , separated by ‘,’. Can be divided over several lines with line delimiter ‘|’ (ASCII character 124 or 7CH).
Font	*Font.***FNT**	• Optional font for the text
X,Y,DX,DY	*Integer values*	• Position, width and height of switch (in relation to top left!)
Color	*0-15 or color name Color0,Color1,…*	• Color of text for states 0,1,...
Backcolor	*0-15 or color name BackColor0,BackColor1,…*	• Background color of switch for states 0,1,...
Format	**Border=Button**	• Button border type with text and icon output with fixed definition
Action	**SetVar+1**	• When the Switch object is pressed the variable value is automatically incremented by 1 and the relevant image/text information is displayed. If the current value is greater/less than the number of defined states (1), so the value 0 is set
Limit1	*Value/variable name*	• Lower limit value, PLC variable or system variable for limit value
Limit2	*Value/variable name*	• Upper limit value, PLC variable or system variable for limit value
Action Limit1 Action Limit2	**#Page=***Name*	• Screen page change to screen page *Name*
	Alarm=*x*	• Trigger alarm x (x is alarm number)
	Backcolor=*x*	• Change background color to x
	Backlight=*x*	• Set backlight (0-100%)
	Close	• Close screen page (window)
	Close=*Name*	• Close screen page (window) *Name*
	Color=*x*	• Change color to x
	Contrast=*x*	• Set contrast (0-100%) (only passive LCD)
	Exit	• Exit program (back to operating system)
	FastFlash	• Set object status to flash at 2 Hz
	Flash	• Set object status to flash at 1 Hz
	Language=*name*	• Online language selection to language *name*
	Language=*s_mysysvar*	• Online language selection to language in *s_mysysvar*
	Load=*x*	• Load recipe file with name x
	Msg=*x*	• Output message with number x
	SetVar=*x*	• Set variable value to x
	SetVar=Limit1	• Set variable value to Limit1
	SetVar=Limit2	• Set variable value to Limit2
	s_mysysvar=x	• Set system variable value to x
VarValue	*Variable name*	• PLC variable or system variable
VarType	**BOOL**	• Bool data type (8-bit)
	BYTE	• Byte data type (8-bit)
	INT	• Integer data type (16-bit)
	UINT	• Unsigned integer data type (16-bit)
	WORD	• Word data type (16-bit)

VarState	*Variable name*	• PLC variable and system variable for object status
Option	**Pos=Center**	• Centered text
	Pos=Left	• Left-justified text
	Pos=Right	• Right-justified text
	PWL=*x*	• Password level required for enabling
	Scroll	• Position of the object can be changed with the scroll object
	Transparency=*colorname*	• Name of the transparent color within an image
C function	*C function name*	• Name of C function (see Integration of C functions)

7.4 Object DropDownList

#DropDownList		Selection of one element out of a static DropDown-List
Text/File	*Image0*.**PCX**,*Image1*.**PCX**,…	• Name of PCX-image files for states 0,1,... separated by ‘,’
	Icon0.**ICO**,*Icon1*.**ICO**,…	• Name of PCX-image files with file extension .ICO for states 0,1,... separated by‘,’ are shown as icons centered in the object
	Text0,Text1,...	• Textstrings for states 0,1,... separated by ‘,’ Text can be devided over several lines with line delimiter ‘|’ (ASCII-character 124 or 7CH)
Font	*Font.***FNT**	• optionaler Font for text
X,Y,DX,DY	*Integerwerte*	• Position, width and height of object (in relation to top left!)
Color	*Farbname oder Farbnummer*	• Color of text
Backcolor	*Farbname oder Farbnummer*	• Background color
Format		• No entry means no border
	Border=Button	• Button border type
	Border=Input	• Input field border type
	Border=R*x*	• Border type, rectangle with width *x* pixels (1, 3, 5, etc.)
	Border=Shadow	• Border type, rectangle with shadow (3D effect)
	Border=Signal	• Signal field border type
Action		• When DropDownList object is pressed the variable value is automatically set to the corresponding image/text state value. (e.g. selection of Image1.pcx → value = 1)
Limit1	*Wert/Variablenname*	• Lower limit value, PLC variable or system variable for limit value
Limit2	*Wert/Variablenname*	• Upper limit value, PLC variable or system variable for limit value
Action Limit1 Action Limit2	**#Page=***Name*	• Screen page change to screen page *Name*
	Alarm=*x*	• Trigger alarm x (x is alarm number)
	Backcolor=*x*	• Change background color to x
	Backlight=*x*	• Set backlight (0-100%)
	Close	• Close screen page (window)
	Close=*Name*	• Close screen page (window) *Name*
	Color=*x*	• Change color to x
	Contrast=*x*	• Set contrast (0-100%) (only passive LCD)
	Exit	• Exit program (back to operating system)
	FastFlash	• Set object status to flash at 2 Hz
	Flash	• Set object status to flash at 1 Hz
	Language=*name*	• Online language selection for language *name*
	Language=*s_mysysvar*	• Online language selection for language in *s_mysysvar*
	Load=*x*	• Load recipe file with name x
	Msg=*x*	• Output message with number x
	SetVar=*x*	• Set variable value to x
	SetVar=Limit1	• Set variable value to Limit1
	SetVar=Limit2	• Set variable value to Limit2
	s_mysysvar=x	• Set system variable value to x
VarValue	*Variablenname*	• PLC-variable or system variable
VarType	**BOOL**	• Bool data type (8-bit)
	BYTE	• Byte data type (8-bit)
	SINT	• Short integer data type (8-bit)
	USINT	• Unsigned short integer data type (8-bit)

	INT	• Integer data type (16-bit)
	UINT	• Unsigned integer data type (16-bit)
	WORD	• Word data type (16-bit)
VarState	*Variablenname*	• PLC variable and system variable for object status
Option	**Coff**	• Display dropdownlist without cursor
	DX=0	• Display without dropdown icon on the right side
	LineHeight=x	• linehight of the dropdownlist (Default: character hight)
	Maxlines=x	• Number of lines within the dropdownlist (Default: all elements)
	Open=Down	• dropdownlist opens down (Default)
	Open=Up	• dropdownlist opens up
	Pos=Center	• Centered text
	Pos=Left	• Left-justified text
	Pos=Right	• Right-justified text
	PWL=*x*	• Password level required for enabling
	Scroll	• Position of the object can be changed with the scroll object
	Transparency=*Farbname*	• Name of the transparent color within an image
C-Function	*C function name*	• Name of C function (see Integration of C functions)

7.5 Radio button object

#RadioButton		Touch-sensitive area, selection of one of several options
Text/File	*Image0*.**PCX**,*Image1*.**PCX**,…	• Name of PCX image files for states Inactive and Active, separated by ‘,’.
	Icon0.**ICO**,*Icon1*.**ICO**	• Name of PCX image files with the suffix .ICO for states Inactive and Active, separated by ‘,’, are shown as icons that are centered in the button.
	Text0,Text1	• Text strings for states Inactive and Active, separated by ‘,’. Can be divided over several lines with line delimiter ‘|’ (ASCII character 124 or 7CH).
Font	*Font.***FNT**	• Optional font for the text
X,Y,DX,DY	*Integer values*	• Position, width and height of radio button (in relation to top left!)
Color	*0-15 or color name Color0,Color1,…*	• Color of text for states Inactive and Active
Backcolor	*0-15 or Color name BackColor0,BackColor1*	• Background color of radio button for states Inactive and Active
Format	**Border=Button**	• Button border type with text and icon output with fixed definition
Action1 & Action2	**#Page=***Name*	• Screen page change to screen page *Name*
	SetIndex=x	• Set index for indexed variable access
	SetVar=*x*	• Set variable value to x
Limit1	*Value/variable name*	• Lower limit value, PLC variable or system variable for limit value
Limit2	*Value/variable name*	• Upper limit value, PLC variable or system variable for limit value
Action Limit1 Action Limit2	**#Page=***Name*	• Screen page change to screen page *Name*
	Alarm=*x*	• Trigger alarm x (x is alarm number)
	Backcolor=*x*	• Change background color to x
	Backlight=x	• Set backlight (0-100%)
	Close	• Close screen page (window)
	Close=*Name*	• Close screen page (window) *Name*
	Color=*x*	• Change color to x
	Contrast=*x*	• Set contrast (0-100%) (only passive LCD)
	Exit	• Exit program (back to operating system)
	FastFlash	• Set object status to flash at 2 Hz
	Flash	• Set object status to flash at 1 Hz
	Language=*name*	• Online language selection to language *name*
	Language=*s_mysysvar*	• Online language selection to language in *s_mysysvar*
	Load=*x*	• Load recipe file with name x
	Msg=*x*	• Output message with number x
	SetVar=*x*	• Set variable value to x
	SetVar=Limit1	• Set variable value to Limit1
	SetVar=Limit2	• Set variable value to Limit2
	s_mysysvar=x	• Set system variable value to x
VarValue	*Variable name*	• PLC variable and system variable
VarType	**BOOL**	• Bool data type (8-bit)
	BYTE	• Byte data type (8-bit)
	INT	• Integer data type (16-bit)
	SINT	• Short integer data type (8-bit)
	UINT	• Unsigned integer data type (16-bit)

	USINT	• Unsigned short integer data type (8-bit)
	WORD	• Word data type (16-bit)
VarState	*Variable name*	• PLC variable and system variable for object status
Option	**Pos=Center**	• Centered text
	Pos=Left	• Left-justified text
	Pos=Right	• Right-justified text
	PWL=*x*	• Password level required for enabling
	Scroll	• Position of the object can be changed with the scroll object
	Transparency=*colorname*	• Name of the transparent color within an image
C function	*C function name*	• Name of C function (see Integration of C functions)

Function of the radio button

The radio button object compares the current object value with the setpoint of the action SetVar=x. If the value equals the setpoint, the radio button concerned is displayed as active. Otherwise the radio button is inactive. If the radio button is pressed, the appropriate setpoint is set.

Several radio button objects can be defined for selecting several elements. The individual selection elements are linked by assigning them with the same PLC variables

7.6 Variable object

#Variable		Display of a numeric/alphanumeric variable
Text/File	*Text* ***%[Flags][Width].[Prec][Type]*** *Text* *%[Format-Unit0],[Format-Unit1]*	• Text string with format definition (in relation to top left) Exmpl. Speed %5.2d rpm Valid format definitions after %: *Flags:* + ...Optional output always with sign 0 ...Optional output with preceding zeros *Width:* Number ...Optional number of preceding digits *Prec:* Number ...Optional number of decimal places *Type:* b ...Binary representation (word) lb ...Binary representation (double word) d ...Integer data format (word) ld ...Double accuracy (double word) u ...Unsigned integer data format (word) lu ...Unsigned integer data format (double word) e ...Exponential representation f ...Floating point representation x ...Hexadecimal representation (word) lx ...Hexadecimal representation (double word) c ...Representation as character s ...String (without length, 80 characters) Time/date representation English (VarType TIME) %a ...Abbreviated weekday %A ...Full weekday %b ...Abbreviated month %B ...Full month (English) %c ...Local representation of date and time %d ...Day of the month (01-31) %H ...Hour (00-23) %I ...Hour (01-12) %j ...Day in year (001-366) %m ...Month (01-12) %M ...Minute (00-59) %p ...Local equivalent of AM or PM %S ...Second (00-59) %U ...Week in year (00-53) (Sunday is the first weekday) %w ...Weekday (0-6) (Sunday is 0) %W ...Week in year (00-53) (Monday is the first weekday) %x ...Local representation of date %X ...Local representation of time %y ...Year without century (00-99) %Y ...Year with century %Z ...Name of time zone if present (depending on hardware)

Text/File	*Text* **%[***HHH:MM:SS:MSMSMS***]T** *Text*	• Time representation (VarType IEC_TIME) Example. Cycle %[M:SS:MSMSMS]T HHH ...Hours with 3 digits 0-9 MM ...Minutes with 2 digits 0-9 SS ...Seconds with 2 digits 0-9 MSMSMS ...Milliseconds with 3 digits 0-9 : ...Separator • Inputs / outputs are converted directly • Constants for limits are defined in IEC61131 format T#10H5M2S100MS
	Text **%[***dd.mm.YYYY HH:MM:SS***]DT** *Text*	• Time and Date representation (VarType IEC_DT) Example. %[dd.mm.YYYY HH:MM:SS]DT dd ...day with 2 digits mm ...month with 2 digits YYYY ...Jahr with 4 digits HH ...Stunden with 2 digits MM ...Minuten with 2 digits SS ...Sekunden with 2 digits : ...Separator • Inputs / outputs are converted directly • Constants for limits are defined in IEC61131 format DT#04d04m2003Y12H30M03S
Font	*Font.***FNT**	• Optional font for the text
X,Y,DX,DY	*Integer values*	• Position, width and height of the variable (in relation to top left!)
Color	*0-15 or color name*	• Color of variable
Backcolor	0-15 or color name	• Background color of variable
Format		• No entry means no border
	Border=Button	• Button border type
	Border=Input	• Input field border type
	Border=R*x*	• Border type, rectangle with width *x* pixels (1, 3, 5, etc.)
	Border=Shadow	• Border type, rectangle with shadow (3D effect)
	Border=Signal	• Display field border type
	Invisible	• Object invisible
Action	**#Page=***Name*	• Screen page change to input screen page *Name*
	#Page=*Name%s_language%*	• Screen page change to input screen page *Name+Value of s_xy*. Page name will be created dynamically. This can be used to create different keyboards for different languages.
	Edit	• A variable set to Input mode via the Set_Focus option can be edited directly
	SetVar=*x*	• Set variable value to x (strings must be defined with enclosed single apostrophe, e.g. ‘String’)
	SetVar+*x*	• Increment variable value by x
	SetVar-*x*	• Decrement variable value by x
	SetVar=NotVar	• Invert variable value (0/1)
Limit1	*Value/variable name*	• Lower limit value, PLC variable or system variable for limit value
Limit2	*Value/variable name*	• Upper limit value, PLC variable or system variable for limit value

Action Limit1 Action Limit2	**#Page=***Name*	• Screen page change to screen page *Name*
	Alarm=*x*	• Trigger alarm x (x is alarm number)
	Backcolor=*x*	• Change background color to x
	Backlight=x	• Set backlight (0-100%)
	Close	• Close screen page (window)
	Close=*Name*	• Close screen page (window) *Name*
	Color=*x*	• Change color to x
	Contrast=*x*	• Set contrast (0-100%) (only passive LCD)
	Exit	• Exit program (back to operating system)
	FastFlash	• Set object status to flash at 2 Hz
	Flash	• Set object status to flash at 1 Hz
	Language=*name*	• Online language selection to language *name*
	Language=*s_mysysvar*	• Online language selection to language in *s_mysysvar*
	Load=*x*	• Load recipe file with name x
	Msg=*x*	• Output message with number x
	SetVar=*x*	• Set variable value to x
	SetVar=Limit1	• Set variable value to Limit1
	SetVar=Limit2	• Set variable value to Limit2
	s_mysysvar=x	• Set system variable value to x
	Unit=x	• Change unit system
VarValue	*Variable name*	• PLC variable and system variable
VarType	**BOOL**	• Bool data type (8-bit)
	BYTE	• Byte data type (8-bit)
	DINT	• Double integer data type (32-bit)
	DWORD	• Double word data type (32-bit)
	IEC_DT	• Datatype IEC_DT (32 Bit)
	IEC_TIME	• IEC_TIME data type (32-bit)
	INT	• Integer data type (16-bit)
	REAL	• Floating point data type (32-bit)
	SINT	• Short integer data type (8-bit)
	STRING	• String data type, 80 bytes (without definition: default 80 bytes)
	STRING:xx	• String data type xx bytes
	TIME	• Time data type, representation of time/date variables
	UDINT	• Unsigned double integer data type (32-bit)
	UINT	• Unsigned integer data type (16-bit)
	USINT	• Unsigned short integer data type (8-bit)
	WORD	• Word data type (16-bit)
VarState	*Variable name*	• PLC variable and system variable for object status
Option	**HelpText**=x	• When the input is started, the system variable s_helptext variable is set to x (see also System variables)
	Pos=Center	• Centered variable
	Pos=Left	• Left-justified variable
	Pos=Right	• Right-justified variable
	PWL=*x*	• Password level required for enabling
	Scroll	• Position of the object can be changed with the scroll object
	Set_Focus	• Sets the variable to input mode (only one variable per screen page possible)
	Type=Password	• Inputs/outputs of values icht in Klartext sondern mit dem Charakter „*“ (Findet Verwendung bei der Eingabe und Ausgabe von Passworten)

C function	*C function name* *Unit-conversion function*	• Name of C function (see Integration of C functions) • predefined conversion function or free define able factor e.g. [,mm_inch] oder [,*2.54] • [[Function-Unit0],Function-Unit1]

Integer values with decimal point
Integer values can be shown on screen with a decimal point. A value of 1000 grammes can be shown and entered as the unit kg 1.000. In most cases, this saves the PLC from using a time consuming floating point arithmetic function (speed!).

Time/date functions
The Time variable type enables the time/date to be displayed. The current time is updated once every second. Special system variables are available for entering the time and date (see also System variables).

Current time on PLC
Variable objects support the download of time to the PLC (once per second). To do this the object must be assigned with a PLC variable with the following structure:

Global variable:

```
VAR_GLOBAL
    EpamTime        :    EpamTimeType;
END_VAR
```

PLC data type:

```
TYPE EpamTimeType:
    STRUCT
        tm_time     :    DT;     (* Time since 1st January 1970, 00:00:00 *)
        tm_sec      :    WORD;   (* Seconds after the minute - [0,59] *)
        tm_min      :    WORD;   (* Minutes after the hour - [0,59] *)
        tm_hour     :    WORD;   (* Hours since midnight - [0,23] *)
        tm_day      :    WORD;   (* Day of the month - [1,31] *)
        tm_mon      :    WORD;   (* Months of the year [1,12] *)
        tm_year     :    WORD;   (* Year since 1900 *)
        tm_wday     :    WORD;   (* Days since Sunday - [0,6] *)
        tm_yday     :    WORD;   (* Days since 1st January - [0,365] *)
        tm_isdst    :    WORD;   (* Daylight saving time flag 0 = off, 1 = on, -1 = not avail *)
    END_STRUCT
END_TYPE
```

System variables
The following system variables are defined with the current values when the input is started, and can be displayed, for example, on the screen keyboard page:

s_edit_val	…Last value before input of data type STRING
s_input_val	…Current value of input of data type STRING
s_limit1	...Lower limit value of data type STRING
s_limit2	...Upper limit value of data type STRING
s_helptext	...Contains the current Help text number Data type WORD

Input screen page

Input screen pages defined with variable action „#Page=name“ has to be defined as window! That means the screen size of the page with the variable-object and the keyboard page has to be different.

7.7 Bar object

#Bar		Representation of a value as a rectangular bar
Text/File		
Font		
X,Y,DX,DY	*Integer values*	• Position, width and height of bar (in relation to top left!)
Color	*0-15 or color name*	• Color of bar (fill color)
Backcolor	0-15 or color name	• Background color of bar (delete color)
Format		• No entry means no border
	Border=Button	• Button border type
	Border=Input	• Input field border type
	Border=R*x*	• Border type, rectangle with width *x* pixels (1, 3, 5, etc.)
	Border=Shadow	• Border type, rectangle with shadow (3D effect)
	Border=Signal	• Signal field border type
Action	**SetVar=***x*	• Set variable value to x
	SetVar+*x*	• Increment variable value by x
	SetVar-*x*	• Decrement variable value by x
Limit1	*Value/variable name*	• Lower limit value, PLC variable or system variable for limit value
Limit2	*Value/variable name*	• Upper limit value, PLC variable or system variable for limit value
Action Limit1 Action Limit2	**#Page=***Name*	• Screen page change to screen page *Name*
	Alarm=*x*	• Trigger alarm x (x is alarm number)
	Backcolor=*x*	• Change background color to x
	Backlight=*x*	• Set backlight (0-100%)
	Close	• Close screen page (window)
	Close=*Name*	• Close screen page (window) *Name*
	Color=*x*	• Change color to x
	Contrast=*x*	• Set contrast (0-100%) (only passive LCD)
	Exit	• Exit program (back to operating system)
	FastFlash	• Set object status to flash at 2 Hz
	Flash	• Set object status to flash at 1 Hz
	Language=*name*	• Online language selection to language *name*
	Language=*s_mysysvar*	• Online language selection to language in *s_mysysvar*
	Load=*x*	• Load recipe file with name x
	Msg=*x*	• Output message with number x
	SetVar=*x*	• Set variable value to x
	SetVar=Limit1	• Set variable value to Limit1
	SetVar=Limit2	• Set variable value to Limit2
	s_mysysvar=x	• Set system variable value to x
VarValue	*Variable name*	• PLC variable and system variable
VarType	**BYTE**	• Byte data type (8-bit)
	DINT	• Double integer data type (32-bit)
	DWORD	• Double word data type (32-bit)
	INT	• Integer data type (16-bit)
	REAL	• Floating point data type (32-bit)
	SINT	• Short integer data type (8-bit)
	UDINT	• Unsigned double integer data type (32-bit)
	UINT	• Unsigned integer data type (16-bit)
	USINT	• Unsigned short integer data type (8-bit)
	WORD	• Word data type (16-bit)

VarState	*Variable name*	• PLC variable and system variable for object status
Option	**Fill=Down**	• Fill direction from top to bottom
	Fill=Left	• Fill direction from right to left
	Fill=Right	• Fill direction from left to right
	Fill=Up	• Fill direction from bottom to top
	Fill=x	• Fill direction left and right (Bar center = (Limit1 + Limit2) / 2)
	Fill=y	• Fill direction bottom and top (Bar center = (Limit1 + Limit2) / 2)
	PWL=*x*	• Password level required for enabling
	Scroll	• Position of the object can be changed with the scroll object
C function	*C function name*	• Name of C function (see Integration of C functions)

The limit action (e.g. color change) is triggered when the limit values are undershot or overshot. A limit action, for example, at 80 % of the value, is not possible. For that two bargraphs can be defined (s.a. Demo “overlayed objects”).

7.8 Signal object

#Signal		Display of states or static images and texts
Text/File	*Image0*.**PCX**,*Image1*.**PCX**,…	• Name of PCX image files for states 0,1, separated by ‘,’.
	Icon0.**ICO**,*Icon1*.**ICO**,…	• Name of PCX image files with the suffix .ICO for states 0,1, separated by ‘,’, are shown as icons that are centered in the button.
	Text0,Text1,...	• Text strings for states 0,1 , separated by ‘,’. Can be divided over several lines with line delimiter ‘|’ (ASCII character 124 or 7CH).
		• If no text is entered, the signal is output without text
Font	*Font*.**FNT**	• Optional font for the text
X,Y,DX,DY	*Integer values*	• Position, width and height of signal (in relation to top left!)
Color	*0-15 or color name Color0,Color1,…*	• Color of text for states 0,1,...
Backcolor	*0-15 or color name BackColor0,BackColor1,…*	• Background color of signal for states 0,1,...
Format		• No entry means no border
	Border=Button	• Button border type
	Border=Input	• Input field border type
	Border=R*x*	• Border type, rectangle with width *x* pixels (1, 3, 5, etc.)
	Border=Shadow	• Border type, rectangle with shadow (3D effect)
	Border=Signal	• Signal field border type
	Frame=*x*	• Border type frame with width *x* pixels (1, 3, 5, etc.) and the corresponding background color for the states 0,1,... The area inside the frame will not be erased.
Action		• Only display, no inputs
Limit1	*Value/variable name*	• Lower limit value, PLC variable or system variable for limit value
Limit2	*Value/variable name*	• Upper limit value, PLC variable or system variable for limit value
Action Limit1 Action Limit2	**#Page=***Name*	• Screen page change to screen page *Name*
	Alarm=*x*	• Trigger alarm x (x is alarm number)
	Backcolor=*x*	• Change background color to x
	Backlight=x	• Set backlight (0-100%)
	Close	• Close screen page (window)
	Close=*Name*	• Close screen page (window) *Name*
	Color=*x*	• Change color to x
	Contrast=*x*	• Set contrast (0-100%) (only passive LCD)
	Exit	• Exit program (back to operating system)
	FastFlash	• Set object status to flash at 2 Hz
	Flash	• Set object status to flash at 1 Hz
	Language=*name*	• Online language selection to language *name*
	Language=*s_mysysvar*	• Online language selection to language in *s_mysysvar*
	Load=*x*	• Load recipe file with name x
	Msg=*x*	• Output message with number x
	SetVar=*x*	• Set variable value to x
	SetVar=Limit1	• Set variable value to Limit1
	SetVar=Limit2	• Set variable value to Limit2
	s_mysysvar=x	• Set system variable value to x
VarValue	*Variable name*	• PLC variable and system variable
VarType	**BOOL**	• Bool data type (8-bit)

	BYTE	• Byte data type (8-bit)
	INT	• Integer data type (16-bit)
	SINT	• Short integer data type (8-bit)
	UINT	• Unsigned integer data type (16-bit)
	USINT	• Unsigned short integer data type (8-bit)
	WORD	• Word data type (16-bit)
VarState	*Variable name*	• PLC variable and system variable for object status
Option		• Default: Signal centered
	Pos=Center	• Signal centered
	Pos=Left	• Signal left-justified
	Pos=Right	• Signal right-justified
	Scroll	• Position of the object can be changed with the scroll object
	Switch=*x*	• x = value range for Action0,Action1,... e.g.: <0:1:2..5:>5 permissible values: Constant number e.g. 5 <Number ...less than >Number ...greater than Number..Number ...range from to : ...Separator
	Transparency=*colorname*	• Name of the transparent color within an image
C function	*C function name*	• Name of C function (see Integration of C functions)

Triggering screen page changes in the PLC
The signal object can also be used to trigger screen page changes on the basis of variable values by defining limit values (Limit1/2) and the action #Page=*name*. In this case, the Text/File column is empty. The object must be defined globally in order for this screen page change to be executed at any location (see also Global objects).

Signal state
It is possible to define multiple signal states separated by comma. In this case depending to the actual variable value (0,1,2,…) the coresponding text or PCX-image file will be displayed.

Example:
PCX-image files Icon0.ico,Icon1.ico,Icon2.ico
...change to PCX-image file „Icon0.ico“, if variable value equals 0
...change to PCX-image file „Icon1.ico“, if variable value equals 1
...change to PCX-image file „Icon2.ico“, if variable value equals 2

The **Option ‚switch=’** allows to define value ranges for the individual states.

Example:
PCX-image files Icon0.ico,Icon1.ico,Icon2.ico with option:switch=<0:0..5:>5
...change to PCX-image file „Icon0.ico“, if variable value is less than < 0
...change to PCX-image file „Icon1.ico“, if variable value is within the range of 0 to 5
...change to PCX-image file „Icon2.ico“, if variable value is greater than> 5

Important!
If image names (*.ICO and *.PCX) are used it is not allowed to use Spaces between the commas, because they will be seen as part of the filename!
To define a colour change for the same Text, the Text has to be defined multiple.

7.9 Message object

#Message		Output of messages as text or images
Text/File	Message.**TXT**	• Name of the message definition file with the message number and the message texts
Font	*Font.***FNT**	• Optional font for the text
X,Y,DX,DY	*Integer values*	• Position, width and height of message (in relation to top left!)
Color	*0-15 or color name*	• Color of the text of the message window
Backcolor	0-15 or color name	• Background color of the message window
Format		• No entry means no border
	Border=Button	• Button border type
	Border=Input	• Input field border type
	Border=R*x*	• Border type, rectangle with width *x* pixels (1, 3, 5, etc.)
	Border=Shadow	• Border type, rectangle with shadow (3D effect)
	Border=Signal	• Signal field border type
Action	**SetVar=***x*	• Set variable value to x
Limit1		•
Limit2		•
Action Limit1 Action Limit2		•
VarValue	*Variable name*	• PLC variable and system variable
VarType	**SINT**	• Short integer data type (8-bit)
	INT	• Integer data type (16-bit)
	STRING:xx	• String data type xx bytes
	UINT	• Unsigned integer data type (16-bit)
	WORD	• Word data type (16-bit)
VarState	*Variable name*	• PLC variable and system variable for object status
Option		• Default: Message left-justified
	Pos=Center	• Message centered
	Pos=Left	• Message left-justified
	Pos=Right	• Message right-justified
	PWL=*x*	• Password level required for enabling
	Scroll	• Position of the object can be changed with the scroll object
	Transparency=*colorname*	• Name of the transparent color within an image
C function	*C function name*	• Name of C function (see Integration of C functions)

Message texts with variable values
A message text can contain additional format definitions (e.g.%d, see also Variable object). In this case the variable type (VarType) must be defined as a String with the length of the message number (2 bytes) and the length of the variable parameter (with %d 2 bytes more, i.e. in this case with String:4). The length of the string is determined according to the largest format definition in the message file. When a message with parameters is called in the PLC, the variable arguments must be set beforehand according to the format definition.

Example: In addition to the relevant message number, the message is assigned variable values (INT,REAL)

Global variables:

```
VAR_GLOBAL
                VisuMessage     AT %MB200  :    ARRAY[0..5] OF BYTE
                MessageNumber   AT %MW200  :    INT;
                MessageReal     AT %MD202  :    REAL;
                MessageINT      AT %MW202  :    INT;
END_VAR
```

Important!
If the Message-object contains a variable value within the message, the message number has to be of type DWORD (32Bits) on all WindowsCE defices!

7.9.1 Message worksheet

The Message worksheet is where the message texts and the message-specific properties are assigned to the message numbers. The Message worksheet has the following structure:

Number	Text/File	Font	Color	Backcolor
1	Any message 1	Arial8.FNT	white	yellow
2	Any message 2	Arial8.FNT	white	red
3	Any message 3 with variable %3d	Arial8.FNT	white	red
4	Any message 4 with variable %2.2f	Arial8.FNT	white	red
...	...	...	...	...
10	myMsg.PCX			
...	...	...	...	...

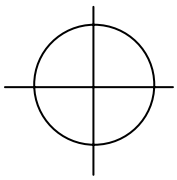
Message-specific font and color definition (optional)

The list of messages must be sorted by message number in ascending order, starting with the lowest message number!

Deleting messages from the PLC:
To do this, define an empty message that does not have any text. In this case only the rectangular area is deleted.

Message texts must be defined for every language. If no text is defined for a message an „empty“ message will be displayed. The text of the default language will not be displayed!

7.10 Meter object

#Meter		Display of a value in a semicircle/circle/user-defined segment
Text/File	*Image.***PCX**	• Name of a PCX image file for the background image. The background and the scale are not drawn (only the pointer)
	Text	• Text string (in the ¾ circle the text is shown in the empty ¼, in upwards/downwards facing semicircles, below/above the object)
Font	*Font.***FNT**	• Optional font for the text
X,Y,DX,DY	*Integer values*	• Position, width and height of rectangle including scale (in relation to top left!) If the Up/Down or Left/Right format option is defined, the meter object is aligned to the center line inside the rectangle. In other words, if two meter objects such as Up and Down are defined with the same co-ordinates, these are positioned symmetrically around the center line.
Color	*0-15 or color name*	• Color of scale and pointer
Backcolor	0-15 or color name	• Background color (color of semicircle/circle/segment)
Format 	**Angle=***Begin,Angle,Scalenum*	• User-defined segment, segment start, segment size (as angle), number of scale sections ...Negative segment size (angle) => Clockwise ...Positive segment size (angle) => Counter-clockwise) Scale sections are divided into 5 subsections, display without scale possible (Scale = 0). The scale can be displayed with an accuracy of one decimal place.
	Circle	• ¾ circle (default) (minimum value at 225°)
	Down	• Semicircle down (min. at 180°)
	Left	• Semicircle to left (min. at 270°)
	Right	• Semicircle to right (min. at 270°)
	Up	• Semi-circle up (min. at 180°)
Action	**SetVar=***x*	• Set variable value to x
	SetVar+*x*	• Increment variable value by x
	SetVar-*x*	• Decrement variable value by x
Limit1	*Value/variable name*	• Lower limit value, PLC variable or system variable for limit value
Limit2	*Value/variable name*	• Upper limit value, PLC variable or system variable for limit value
Action Limit1 Action Limit2	**#Page=***Name*	• Screen page change to screen page *Name*
	Alarm=*x*	• Trigger alarm x (x is alarm number)
	Backcolor=*x*	• Change background color to x
	Backlight=x	• Set backlight (0-100%)
	Close	• Close screen page (window)
	Close=*Name*	• Close screen page (window) *Name*
	Color=*x*	• Change color to x
	Contrast=*x*	• Set contrast (0-100%) (only passive LCD)
	Exit	• Exit program (back to operating system)
	FastFlash	• Set object status to flash at 2 Hz
	Flash	• Set object status to flash at 1 Hz
	Language=*name*	• Online language selection to language *name*
	Language=*s_mysysvar*	• Online language selection to language in *s_mysysvar*
	Load=*x*	• Load recipe file with name x
	Msg=*x*	• Output message with number x

	SetVar=*x*	• Set variable value to x
	SetVar=Limit1	• Set variable value to Limit1
	SetVar=Limit2	• Set variable value to Limit2
	s_mysysvar=x	• Set system variable value to x
VarValue	*Variable name*	• PLC variable and system variable
VarType	**BYTE**	• Byte data type (8-bit)
	DINT	• Double integer data type (32-bit)
	DWORD	• Double word data type (32-bit)
	INT	• Integer data type (16-bit)
	REAL	• Floating point data type (32-bit)
	SINT	• Short integer data type (8-bit)
	UDINT	• Unsigned double integer data type (32-bit)
	UINT	• Unsigned integer data type (16-bit)
	USINT	• Unsigned short integer data type (8-bit)
	WORD	• Word data type (16-bit)
VarState	*Variable name*	• PLC variable and system variable for object status
Option	**Scroll**	• Position of the object can be changed with the scroll object
	Transparency=*colorname*	• Name of the transparent color within an image
C function	*C function name*	• Name of C function (see Integration of C functions)

The limit action (e.g. color change) is triggered when the limit values are undershot or overshot. A limit action, for example, at 80% of the value, is not possible.

7.11 Text list object

#Textlist		Output of an ASCII text file within a rectangular area
Text/File	*Text.***TXT**	• Name of ASCII text file
	s_alarm_txtinfo	• Variable with the name of the ASCII text file with the alarm text information
	*Text.**CSV***	• Display CSV-file in chart form
Font	*Font.***FNT**	• Optional font for the text
X,Y,DX,DY	*Integer values*	• Position, width and height of the text list (in relation to top left!)
Color	*0-15 or color name*	• Color of the text of the text list
Backcolor	0-15 or color name	• Background color of text list
Format		• No entry means no border
	Border=Button	• Button border type
	Border=Input	• Input field border type
	Border=R*x*	• Border type, rectangle with width *x* pixels (1, 3, 5, etc.)
	Border=Shadow	• Border type, rectangle with shadow (3D effect)
	Border=Signal	• Signal field border type
Action		•
Limit1		•
Limit2		•
Action Limit1 Action Limit2		•
VarValue	*Variable name*	• STRING system variable containing the file name or System variable which defines the jump line number for the text list or saves the current line number
VarType	**INT**	• Integer data type (16-bit)
	STRING	• String data type 80 bytes (without definition: default 80 bytes)
	UINT	• Unsigned integer data type (16-bit)
	WORD	• Word data type (16-bit)
VarState		•
Option	**DX=***x*	• Width of the scroll bar in pixels (default: 25 pixels)
	Format	• Displays textfile with formated flowtext
C function	*C function name*	• Name of C function (see Integration of C functions)

Display of formated flow text
With option Format it is possible to dispay textfile with formated flowtext. In this case the text will automatically spiltted into different lines corresponding to the defined font, size, etc. and the dimension of the textwindow. Newlines will be inserted automatically between words. Normal newlines <CRLF> will be ignored.

Manuel new line:
Text **Bold**: **myText**
Text Underline: <u>myText</u>
Text Italic: *myText*

Display of different textfiles with the same Textlist object
To display different textfiles in one page with a textlist object you can use the button action SetVar='MyTextFile.txt' and a the textlist object with a system variable of type STRING. In this case the system variable contains the filename of the textfile.

Example:

Definition of object textlist in page *Help:*

Object	Text/File	Font	X	Y	DX	DY	Color	Back-color	Format	Action	Action Limit1	Action Limit2	Var-Value	Var-Type	Var-State
#Page=Help			10	10	300	220	black	grey							
#Textlist		COUR16F.FNT	5	50	290	165	black	grey					S_Helpfile	STRING	

Change to page *Help* and setting variable s_helpfile to help1.txt:

Object	Text/File	Font	X	Y	DX	DY	Color	Back-color	Format	Action	Action Limit1	Action Limit2	Var-Value	Var-Type	Var-State
#Page=Seite1			10	10	300	220	black	grey							
#Button	Help.ico		200	0	40	40	black	grey		setVar='help1.txt' & #Page=Help			S_Helpfile	STRING	

Change to page *Help* and setting variable s_helpfile to help2.txt:

Object	Text/File	Font	X	Y	DX	DY	Color	Back-color	Format	Action	Action Limit1	Action Limit2	Var-Value	Var-Type	Var-State
#Page=Seite2			10	10	300	220	black	grey							
#Button	Help.ico		200	0	40	40	black	grey		setVar='help2.txt' & #Page=Help			S_Helpfile	STRING	

Display of CSV-files in chart form
Optionally it is possible to specify a format header as the first line within the CSV-file (TAB-seperated). Without this header the CSV-file will be displayed with columns oft he same width and lext left alignment.

#header width=100 pos=center \t width=50 pos=rigth \t width=70 pos=left

7.12 Object HTMLBrowser

#HTMLBrowser		Output of an HTML file within a rectangular area (Browser window)
Text/File	URL	• URL of HTML-page local or in Intranet/Internet
Font		•
X,Y,DX,DY	*Integerwerte*	• Position, width and height of the text list (in relation to top left!)
Color		•
Backcolor		•
Format		•
Action		•
Limit1		•
Limit2		•
Action Limit1 Action Limit2		•
VarValue	*Variable name*	• PLC or system variable with URL
VarType	**STRING**	• Data type String 80 Bytes (default 80 Bytes)
VarState		•
Option		•
C function	*C function name*	• Name of C function (see Integration of C functions)

Requirements for HTML-Browser

The HTMLBrowser object is only available for Windows! On the target the InternetExplorer must be installed. If an Internet URL is defined (e.g. http://www.microinnovation.com/), an Internet connection via network is required on the target (e.g. via Gateway). Alternative it is possible to store HTML-pages also local on the device. In this case the full path of the HTML-pages is required. (e.g. C:\MyHTML\default.htm)

7.13 Alarm object

#Alarm		Alarm monitoring
Text/File	*MyAlarm.***TXT**	• Name of the alarm definition file with the alarm variables and the alarm message texts
Font		•
X,Y,DX,DY		•
Color		•
Backcolor		•
Format		•
Action	**#Page=***Name*	• Screen change to page *Name* on incoming alarms (“Come” alarms)
Limit1		•
Limit2		•
Action Limit1 Action Limit2		•
VarValue	*Variable name*	• PLC variable PLC variable specifying the alarm buffer
VarType	**STRING**	• Data type Array of Word (length according to number of alarms)
VarState		•
Option	**Type=**name	• Alarm type (allows the definition of different Alarm objects with the same Alarm definition file)
C function	*C function name*	• Name of C function (see Integration of C functions)

The alarm object will be designed globally in the first screen page in the Project worksheet (initialization page)!

7.13.1 Alarm worksheet

In the Alarm worksheet, the alarm variables are assigned with the alarm message texts, alarm-specific properties, actions and information. The Alarm worksheet has the following structure:

Variable	Alarm number	Text/File	Font	Color	Backcolor	Alarm Info action	Alarm Help text
[0].0		1 = Alarm(s) active (Direction: EPAM -> PLC)					
[0].1		1 = Delete alarm request from visualization (Direction: EPAM -> PLC)					
[0].2		1 = Delete alarm request from PLC (Direction: PLC -> EPAM)					
[0].3		0 = all Alarm(s) ack (Direction: EPAM -> PLC)					
[0].4		1 = Alarm.INI file (History) written (Direction: EPAM -> PLC)					
[0].5		1 = disable Acion Page=name (Direction: PLC -> EPAM)					
[0].6-[0].15		reserved					
[1].0	1	Any alarm text 1	Arial8.FNT	black	red	#Page=Diagnose, #Page=Photo1	alrmhlp1.txt
[1].1	2	Any alarm text 2	Arial8.FNT	black	red	#Page=Diagnose, #Page=Photo1	alrmhlp1.txt
[1].2	3	Any alarm text 3	Arial8.FNT	black	red	#Page=Diagnose, #Page=Photo1	alrmhlp1.txt
...	...	...	...	...	...	...	...
[1].15	16	Any alarm text 16	Arial8.FNT	black	yellow	#Page=Diagnose, #Page=Photo2	Alrmhlp2.txt
[2].0	17	Any alarm text 17	Arial8.FNT	black	yellow	#Page=Diagnose, #Page=Photo2	Alrmhlp3.txt
[2].1	18	Any alarm text 18	Arial8.FNT	black	yellow	#Page=Diagnose, #Page=Photo2	Alrmhlp3.txt
...	...	...	...	...	...	...	...
[2].14	31	Any alarm text 31	Arial8.FNT	black		#Page=Diagnose, #Page=Photo3	
[2].15	32	Any alarm text 32	Arial8.FNT		white		

Alarm-specific font and color definition (optional)

Action AlarmInfo: (optional)
Alarm info of the selected alarm,
e.g.:
AlarmInfo=1 ⇒ # Page=Diagnose
AlarmInfo=2 ⇒ # Page=Photo1

Alarm Help text: (optinal)
Name of the text file with alarm-specific text information

Alarm numbers must be unique!

Alarm specific Fonts (column Font) will not be supported at the moment!

Structure of the alarm buffer
Variable specifies the data word and data bit in the alarm buffer. The alarm buffer is structured in words and the start address is defined with the alarm object. Each bit from data word 1 represents an alarm. Up to 252*8 = 2016 alarms can be defined, the last 512 are stored in a ring buffer (History). The Alarm history and the current setting (sorting, filter) are saved retentively in the ALARM.INI file.

Definition in the PLC:

Global variables:

```
VAR_GLOBAL
    VisuAlarm           AT %MW1000    :   ARRAY[0..2]OF WORD
    AlarmFlagActive     AT %MX1000.0  :   BOOL;
    AlarmQuitFromVisu   AT %MX1000.1  :   BOOL;
    AlarmQuitFromPLC    AT %MX1000.2  :   BOOL;
    AlarmNoQuitActive   AT %MX1000.3  :   BOOL;
    AlarmIniWritten     AT %MX1000.4  :   BOOL;
    AlarmActionDisable  AT %MX1000.5  :   BOOL;
    Alarm1              AT %MX1002.0  :   BOOL;
    Alarm2              AT %MX1002.1  :   BOOL;
    Alarm3              AT %MX1002.2  :   BOOL;
    Alarm4              AT %MX1002.3  :   BOOL;
    …
    …
    Alarm26             AT %MX1005.1  :   BOOL;
    Alarm27             AT %MX1005.2  :   BOOL;
    Alarm28             AT %MX1005.3  :   BOOL;
    Alarm29             AT %MX1005.4  :   BOOL;
    Alarm30             AT %MX1005.5  :   BOOL;
    Alarm31             AT %MX1005.6  :   BOOL;
    Alarm32             AT %MX1005.7  :   BOOL;
END_VAR
```

7.13.2 Alarm handling procedure

EPAM cyclically checks the alarm variables and enters alarms according to status with a “Come” or “Go” time stamp in the alarm buffer. If required, incoming alarms (“Come” alarms) can also be provided with the screen page change action #Page=name. New alarms are inserted in the following way:

1. if the oldest alarm is inactive and acknowledged, it will be overwritten by the new one
2. if the oldest alarm is inactive and not acknowledged, it will be overwritten
3. if there is no inactive alarm, the oldest active and acknowledged alarm will be overwritten
4. if there is no inactive alarm, the oldest alarm will be overwritten. (in this case alarm will be lost, that means the alarm is no longer displayed in the alarm list -> more than 512 active alarms!)

7.13.3 Acknowledging alarms

Alarms can be acknowledged from both EPAM and from the PLC.

Acknowledging via EPAM:

Alarms can be acknowledged singly or altogether from EPAM, for which bit 1 in the status data word (alarm acknowledgement from visualization) is set and sent to the PLC. Bit 1 in the PLC must then be reset.

Acknowledging via PLC:

It is only possible to set all alarms from the PLC via bit 2 (alarm acknowledgement via PLC). When the alarm is acknowledged, the acknowledge time of the alarm or alarms concerned is set.

7.13.4 Alarm display

Alarm messages can be displayed on screen using the alarm list object. Several diagnostics options based on the alarm number are also available (see Alarm list object).

7.13.5 Export alarm history

The alarm history can be exported in CSV format with the button action **AlarmExport=CSV.** The CSV file will be written into EPAM-Data directory C:\DATA in the following format:

Export of alarmbuffer: myalarm1 @ 2003-04-29 15:03:29
Number of records: 3 sort = lifo
Nr;in[s];out[s];quit[s]
8;1051628587;0;0
3;1051628530;1051628533;1051628539
14;1051628178;1051628591;0

Note!
The time stamp is a value in seconds since 1.1.1970. With the formula =cell/86400+25569 the value can be displayed in normal date/time format in Excel.

7.14 Alarm list object

#Alarmlist		Output of alarm events within a rectangular area
Text/File	**no=%***[Width][Type]*, **tin=***Format*, **tout=***Format*, **tquit=***Format*, **sep=***key code*	• Format definition (in relation to top left) Keywords: no= ...Keyword for the alarm number tin= ...Keyword for “Come” alarm time tout= ...Keyword for “Go” alarm time tquit= ...Keyword for “Acknowledged” alarm time sep= ...Keyword for separator The individual keywords are separated by commas. The format definition comes after the keyword. **Keyword no=:** ...Without keyword no= Default output %5d of alarm number no= ...Only keyword no= No output of alarm number no=%... ...Keyword no= With format definition Output of alarm number according to format Valid format definitions after %: *Width:* Number ...Optional number of preceding digits *Type:* d ...Integer data format (word) **Keyword tin=, tout=, tquit=:** ...Without keywords tin=, tout=, tquit= Default output of date/time txxx= Only Keyword tin=, tout=, tquit= No output of date/time txxx=%... ...Keyword tin=,tout=,tquit= With format definition Output of date/time according to format Valid format definitions: %d ...Day of month (01-31) %H ...Hour (00-23) %I ...Hour (01-12) %j ...Day in year (001-366) %m ...Month (01-12) %M ...Minute (00-59) %S ...Second (00-59) %U ...Week in year (00-53) %w ...Weekday (0-6), %W ...Week in year (00-53) %y ...Year without century (00-99) %Y ...Year with century **Keyword sep=:** ...Without keywords sep= Default separator output sep= Only keyword sep= No output of separator sep=x ...Keyword sep= With key code Output of separator according to key code
Font	*Font.***FNT**	• Optional font for the text
X,Y,DX,DY	*Integer values*	• Position, width and height of the alarm list

		(in relation to top left!)
Color	*0-15* or color name	• Color of the text of the alarm list
Backcolor	0-15 or color name	• Background color of alarm list
Format		• No entry means no border
	Border=Button	• Button border type
	Border=Input	• Input field border type
	Border=R*x*	• Border type, rectangle with width *x* pixels (1, 3, 5, etc.)
	Border=Shadow	• Border type, rectangle with shadow (3D effect)
	Border=Signal	• Display field border type
Action		•
Limit1		•
Limit2		•
Action Limit1 Action Limit2		•
VarValue	*Variable name*	• System variable System variable that stores the currently selected alarm in the alarm list
VarType	**WORD**	• Word data type (16-bit)
VarState		•
Option	**AlarmFilter=activ**	• Set alarm filter: Display active alarms
	AlarmFilter=activ|notquit	• Set alarm filter: Display active or unacknowledged alarms
	AlarmFilter=activ+notquit	• Set alarm filter: Display active and unacknowledged alarms
	AlarmFilter=all	• Set alarm filter: Display all alarms
	AlarmFilter=notquit	• Set alarm filter: Display unacknowledged alarms
	AlarmSort=FIFO	• Sort alarm in alarm list: Oldest alarm first
	AlarmSort=LIFO	• Sort alarm in alarm list: Latest alarm first
	AlarmSort=Priority	• Sort alarm in alarm list: Alarm with highest priority (=lowest alarm number) first
	Coff	• Cursor off
	DX=*x*	• Width of the scroll bar in pixels (default: 25 pixels, • DX=0 display without scroll bar)
C function	*C function name*	• Name of C function (see Integration of C functions)

Format definition example:

Displaying alarm with date and acknowledge time:
"tin=%d-%m %H:%M,tquit=%H:%M,sep=|" => "13-09 08:34|08:40| 1|Alarm 1"

The Alarm list object supports the following displays. They can be selected/defined using Button actions and/or via fixed format definitions of the Alarm list in the Option column:

7.14.1 Alarm filter

- Display of all alarms
- Only active alarms
- Active and unacknowledged alarms
- Active or unacknowledged alarms
- Unacknowledged alarms

7.14.2 Alarm sorting

- Display by priority (low alarm number = High priority)
- Display by time: Latest alarm first (last in first out)
- Display by time: Oldest alarm first (first in first out)

7.14.3 Alarm diagnostics/Alarm system variables

The alarm message selected in the alarm list can be used via the AlarmInfo=1 or Alarminfo=2 action to jump to 2 screen pages configured in the Alarm worksheet in the Action Alarm Info column.
It is also possible to configure a text list with the s_alarm_txtinfo system variable entered in the Text/File column. Using the alarm message selected in the alarm list the screen page with the configured text list can be activated with the #page=name action. The text file configured in the Alarm Help text column of the Alarm worksheet is then displayed in this text list.

In conjunction with the alarm list, the alarm information of the last selected alarm is stored in the following system variables.

s_alarm_nr	…Alarm number, data type: WORD
s_alarm_text	…Alarm text, data type: STRING
s_alarm_tin	…Time Come alarm, data type: STRING
s_alarm_tout	…Time Go alarm, data type: STRING
s_alarm_tquit	…Time Acknowledged alarm, data type: STRING
s_alarm_tin	…Time Come alarm, data type: IEC_DT
s_alarm_tout	…Time Go alarm, data type: IEC_DT
s_alarm_tquit	…Time Acknowledged alarm, data type: IEC_DT
s_alarm_info	…Alarm info, data type: STRING
s_alarm_txtinfo	...Alarm Help text, data type: STRING

7.15 Alarm mail object

#Alarmmail		Email notification as a result of alarm events
Text/File	*MyAlarmmail*.**TXT**	• Name of alarm mail definition file (parameter file)
Font		•
X,Y,DX,DY		•
Color		•
Backcolor		•
Format		•
Action		•
Limit1		•
Limit2		•
Action Limit1 Action Limit2		•
VarValue		•
VarType		•
VarState		•
Option		•
C function		•

Function of alarm mail
The Alarm mail object enables alarm messages to be forwarded as e-mails **(only on the target system).** It is possible in the configuration to assign alarm numbers to specific e-mail addresses. **The functions, Weekday, From and To are currently not implemented!**

Requirements
Connection to an Internet service provider, e.g. via gateway (modem router or leased line). The gateway address can be defined on the target system. The service of a cell phone operator is required when forwarding e-mails as SMS messages. This service is provided in Switzerland by Diax or D2 in Germany.
The following environment variables are required (EPAM.INI):
SMTP_SERVER=name ...Name of the SMTP-Server
SMTP_FROM=email ...Email-Address of the sender

The alarm mail object will be designed globally in the first screen page in the Project worksheet (initialization page)

7.15.1 Alarm mail worksheet

All object parameters are defined in the Alarm mail worksheet. The Alarm mail worksheet has the following structure:

Alarmnumber	Day of Week	From	To	email:
1	reserved	reserved	reserved	1234567890@gsm.myprovider.com
1	reserved	reserved	reserved	name1@email.com
1	reserved	reserved	reserved	name2@email.com

7.16 DiagSig object

#DiagSig		Diagnose signal, flashing point display
Text/File	*Alarm number*	• Alarm number with Diagnose signal active
	Integer	• Integer value on which the Diagnose signal is active
	Text string	• String value on which the Diagnose signal is active
Font		•
X,Y,DX,DY	*Integer values*	• Position, width and height of the Diagnose signal (in relation to top left!)
Color	*0-15 or color name*	• Color of Diagnose signal
Backcolor		•
Format		•
Action	**#Page=***Name*	• Screen page change to screen page *Name*
	Close	• Close screen page (window)
	Close=*Name*	• Close screen page (window) *Name*
Limit1		•
Limit2		•
Action Limit1		• Action in case of alarm active -> inactive (e.g. alarm out)
	#Page=*Name*	• Screen page change to screen page *Name*
	Close	• Close screen page (window)
	Close=*Name*	• Close screen page (window) *Name*
Action Limit2		• Action in case of alarm inactive -> active (e.g. alarm in)
	#Page=*Name*	• Screen page change to screen page *Name*
	Close	• Close screen page (window)
	Close=*Name*	• Close screen page (window) *Name*
VarValue		• With no variable defined, a check is made whether the *Alarm number* configured under Text/File is active
	Variable name	• PLC variable and system variable
VarType		• With no variable defined, a check is made whether the *Alarm number* configured under Text/File is active
	INT	• Integer data type (16-bit)
	STRING	• String data type 80 bytes (without definition: default 80 bytes)
	UINT	• Unsigned integer data type (16-bit)
	WORD	• Word data type (16-bit)
VarState	*Variable name*	• PLC variable and system variable for object status
Option		• Static object status
	FastFlash	• Object status to flash at 2 Hz
	Flash	• Object status to flash at 1 Hz
C function	*C function name*	• Name of C function (see Integration of C functions)

Function of Diagnose signal
The Diagnose signal enables the cause of fault on the machine to be indicated in response to alarm messages. For example, the location of a fault can be indicated on a photograph of the machine (e.g. faulty limit switch). The Diagnose signal object checks the specified text string with the current variable value or whether the corresponding alarm number is active. It then activates the flashing point if the condition is fulfilled.

7.17 Recipe object

#Recipe		Recipe management
Text/File	*MyRecipe*.**TXT**	• Name of recipe definition file
Font		•
X,Y,DX,DY		•
Color		•
Backcolor		•
Format		•
Action	**#Page=***Name*	• Screen page change to screen page *Name* is activated with the csave=MyRecipeTyp button action for confirmation if a recipe file already exists
Limit1		•
Limit2		•
Action Limit1 Action Limit2		•
VarValue	*Variable name*	• PLC variable with the following function: 1 ...Download request from PLC (EPAM requested by PLC to reload the current recipe or the recipe values changed by the user and write them to the PLC) 2 ...Up/download from EPAM finished (Status of EPAM to PLC, that up/download finished) 3 ...Upload request from PLC (EPAM requested by PLC to read the recipe values from the PLC and save them in the ACTUAL.DAT file) 4 ...Download running (Status of EPAM to PLC that current recipe or the recipe values changed by the user are being loaded and written to the PLC) 5 ...Upload running (Status of EPAM to PLC that the recipe values are being read from the PLC and stored in the ACTUAL.DAT file) 6 ...save recipe to current filename (s_myrecipe_file) with current name (s_myrecipe_name) 7 Download Request from PLC (EPAM requested by PLC to load the last recipe file again and download the variables to PLC → undo function)
VarType	**INT**	• Integer data type (16-bit)
	UINT	• Unsigned integer data type (16-bit)
	WORD	• Word data type (16-bit)
VarState		•
Option		• Recipe values changed by the user in ACTUAL.DAT are written to the PLC when EPAM is started
	NoDownload	• Neither recipe values of the currently loaded recipe nor the recipe values changed by the user in ACTUAL.DAT are written to the PLC when EPAM is started **Note**: ACTUAL.DAT is not updated! (not required)

	NoActual	• Recipe values of the currently loaded recipe and not the recipe values changed by the user in ACTUAL.DAT are written to the PLC when EPAM is started **Note**: ACTUAL.DAT is not updated! (not required)
	Filename=Auto	• Create recipe files automatically (alpha numerical)
	Filename=Auto10	• Create recipe files automatically (numerical)
C function	*C function name*	• Name of C function (see Integration of C functions)

The recipe object will be designed globally in the first screen page in the Project worksheet (initialization page)

7.17.1 Recipe worksheet

The Recipe worksheet is used to assign all the recipe variables with a type and always with an initial value. The Recipe worksheet has the following structure:

Variable	VarType	Value
#Recipe=Default		Path=
MyRecipeVar1	WORD	123
MyRecipeVar2	WORD	345
…		
#Checksum=		

Default recipe type: no entry
Recipe type corresponds to recipe designation.
e.g. Type=MyRecipe

Default recipe directory: path=
The recipe is stored in the directory for the specific recipe type.
Example: C:\DATA\MyRecipe

System variables in recipes
System variables can be defined as recipe variables and also as values in the recipe. In this case the current value of the system variable is stored in the recipe and written to the PLC if necessary.

Consistency of recipe files
Recipe files are completed with a checksum in order to ensure their data consistency. The checksum is automatically generated when a recipe is saved and inserted into the last line. If these files are changed manually or created again with Excel, the user must close the file with the sequence**,#Checksum=<CRLF>**'. If this entry is not made, or **if the checksum is incorrect, the recipe file is invalid and is not loaded!**

Consistency of recipe values
Up- and download of recipes is not synchronous to the PLC cycle! The consistency over all recipe values must be checked in the PLC by the help of the recipe state. The download of a recipe is complete when the recipe state is 2.

Cascading recipes
With recipe data type RECIPE:mytype it is possible to define a recipe variable of type STRING which stores the name of a recipe of type “mytype”. So if such a recipe is loaded automatically the defined recipe of typ „mytype“ will be loaded.

7.17.2 Recipe management

The variables in the Recipe worksheet (File*MyRecipe.TXT)* define the process variables required for a recipe in the form of name, type and value. The default recipe *MyRecipe.TXT* contains the default values for the specified variables (Value=default value). The recipe variables are normally created as global objects (see also Global objects) and initialized with default values. Any change to a recipe variable is monitored, and the modified values are stored retentively in the ACTUAL.DAT file. The next time that EPAM is started, the current variable values are therefore reloaded and transferred to the PLC (default). This procedure can be modified using different options.

Several recipe objects with different recipe types (Type) can be defined. For example, recipes for product-specific settings and machine-specific configurations can be managed separately.

Creating the recipe directories
Recipes are stored in directories for specific recipe types. The specific directories for *MyRecipe* recipe types are automatically created in the EPAM data directory C:\DATA and EPAM backup directory C:\BACKUP.
Example:
Recipe type *MyTyp1* in directory C:\DATA\MyTyp1\ or C:\BACKUP\MyTyp1\
Recipe type *MyTyp2* in directory C:\DATA\MyTyp2\ or C:\BACKUP\MyTyp2\

7.17.3 Loading recipes

Recipes are saved with the file suffix *.DAT. The s_*myRecipeType*_file system variable can be used for entering the file name of an existing recipe file (*.DAT) (max. 8 characters without file suffix). The Recipe=load action is used to load the variable values of the recipe file defined by s_*myRecipeType*_file.

The recipe list object offers a more user-friendly option, by which a recipe can be selected and loaded from a list of existing recipe files using the load=list action (see also Recipe list object)

7.17.4 Saving recipes

The current values of the recipe variables are saved in a new recipe file (*.DAT) by specifying a file name in the s_ *myRecipeType*_file’ system variable, an optional recipe designation (system variable ‘s_ *myRecipeType*_name’), and by using the save=*myRecipeType* action. Existing files are overwritten.

Recipes can also be saved using the Recipe list object. The csave=*myRecipeType* action enables a prompt if the file already exists. In this case, the screen page is called that is defined in the recipe object concerned.

7.17.5 Build recipes in EXCEL

Within the worksheet „Recipe“ it is possible to create user defined recipes starting at column D. This user recipes can be built automatically with the macro „Build Recipes“. The following entries are necessary for that:
cell D1: Name of the recipe file without file extension (max. 8 chars)
cell D2: comment (optional)
cell D3: recipe name „#Recipe=myName“
cell D4 and following: recipe values corresponding to the defined recipe values (same as column Value)

7.18 Recipe list object

#RecipeList		Output of a recipe list within a rectangular area
Text/File	*Text* *%[Flags][Width][Type]* *Text* …	• Format string with format definition (in relation to top left) Example: %-9f %-16n %d-%m-%Y %H:%M without format definition: %12f %-12t %n Valid format definitions after %: *Flags:* - ...Optional output left-justified 0 ...Optional output with preceding zeros *Width:* Number ...Optional field width *Type:* f ...Optional display of file name n ...Optional display of recipe name ...(description) t ...Optional display of recipe type Time/date display English %a ...%a ...Abbreviated weekday %A ...Full weekday %b ...Abbreviated month %B ...Full month (English) %c ...Local display of date and time %d ...Day of the month (01-31) %H ...Hour (00-23) %I ...Hour (01-12) %j ...Day in year (001-366) %m ...Month (01-12) %M ...Minute (00-59) %p ...Local equivalent of AM or PM %S ...Second (00-59) %U ...Week in year (00-53) (Sunday is the first weekday) %w ...Weekday (0-6) (Sunday is 0) %W ...Week in year (00-53) (Monday is the first weekday) %x ...Local display of date %X ...Local display of time %y ...Year without century (00-99) %Y ...Year with century %Z ...Name of time zone if present (depending on hardware)
Font	*Font.***FNT**	• Optional font for the text
X,Y,DX,DY	*Integer values*	• Position, width and height of the recipe list (in relation to top left!)
Color	*0-15 or color name*	• Color of the text of the recipe list
Backcolor	0-15 or color name	• Background color of recipe list
Format		• No entry means no border
	Border=Button	• Button border type
	Border=Input	• Input field border type
	Border=R*x*	• Border type, rectangle with width *x* pixels (1, 3, 5, etc.)
	Border=Shadow	• Border type, rectangle with shadow (3D effect)
	Border=Signal	• Signal field border type

Action		•
Limit1		•
Limit2		•
Action Limit1 Action Limit2		•
VarValue	*Variable name*	• System variable System variable that stores the currently selected recipe line number in the recipe list
VarType	**WORD**	• Word data type (16-bit)
VarState		•
Option	**DX=***x*	• Width of the scroll bar in pixels (default: 25 pixels)
	Type=*myrecipetype*	• Set recipe type (e.g. myRecipeType)
	Type=off	• Reset recipe type (all)
C function	*C function name*	• Name of C function (see Integration of C functions)

Loading/saving/deleting recipes
The recipe list shows all the *.DAT files of the current recipe type. A file can be selected and a recipe loaded, saved or deleted using the button actions Load=*myRecipeType*/Load=list, Save=*myRecipeType*/Save=List and Delete=*myRecipeType*/Delete=list.

Changing between recipe types
The button action Type=*myRecipeType* can be used to change between different recipe types. This also changes the recipe directory automatically.

Changing the recipe list directory
The current path can also be changed by setting the s_recipe_path system variable (by means of a button action)
Example:
SetVar='A:' ...Load/save recipe from/to diskette
SetVar='' ...Back to current directory

Sorting the recipe list
The recipe list can be sorted by file name, recipe name, time and recipe type by setting the following button actions:
Sort=File ...Sort recipe list by file name
Sort=Name ...Sort recipe list by recipe name
Sort=Time ...Sort recipe list by file date
Sort=Type ...Sort recipe list by recipe type
Sort=Number ...Sort recipe list by recipe name numerically

7.19 Screen saver object

#ScreenSaver		Screen saver A text/image will be shown on the screen after a defined time, and the backlight will be dimmed
Text/File	*Image0*.**PCX**,*Image1*.**PCX**,…	• Name of a PCX image file Optional other names of PCX image files
	Text0,Text1,...	• Text string Other optional text strings Text can be divided up over several lines with line delimiter '|' (ASCII character 124 or 7CH).
		• If neither image nor text is entered, only the backlight is dimmed when the screen saver is activated
Font	*Font*.**FNT**	• Optional font for the text
X,Y,DX,DY		•
Color	*0-15 or color name*	• Color of the screen saver text
Backcolor	0-15 or color name	• Background color of the screen saver
Format		• No entry means default:random
	random	• Text or image is positioned randomly on screen
	move	• Text or image is moved on the screen
Action	**click**	• Screen saver is closed with the first click at any position on screen
	click=inside	• Screen saver is only closed with a click inside the text/image. If other texts/images are defined, these must be confirmed within 4 seconds in order to close the screen saver (prevention of accidental operation).
Limit1		•
Limit2		•
Action Limit1 Action Limit2		•
VarValue	*Variable name*	• PLC variable or system variable with the following function: 0 ...Do not close active screen saver 1 ...Close active screen saver
VarType	**INT**	• Integer data type (16-bit)
	UINT	• Unsigned integer data type (16-bit)
	WORD	• Word data type (16-bit)
VarState	*Variable name*	• PLC variable or system variable for object status with the following function: 0 ...Screen saver activated after *time1, time2* 1 ...Screen saver is deactivated
Option	**Timeout=***time1,time2*	• Time in minutes *time1* ...Time in minutes to activation of the screen saver and dimming of the backlight (50%) *time2* ...Optional time in minutes to switching off of the backlight (backlights 0%)
C function	*C function name*	• Name of C function (see Integration of C functions)

Deactivating the screen saver in the PLC
When the screen saver is active, only the Alarm, DataLog, and Recipe global objects are updated while the text or PCX image is shown. For important events, however, the screen saver can be deactivated from the PLC by setting the variable defined in the VarState.

Active screen saver on alarm event
If an alarm event occours during active screen saver (incoming Alarm, outgoing Alarm), the screen saver will be inactive.

Protection against operating errors
The entry of additional text or PCX images (separated by commas) ensures that the screen saver is deactivated over several stages. In other words, the first touch causes the display of the next text/image, which in turn must also be confirmed in order to terminate the screen saver. This can be used to virtually exclude the accidental triggering of functions via touch.

7.20 Password object

#Password		Password management
Text/File		•
Font		•
X,Y,DX,DY		•
Color		•
Backcolor		•
Format		•
Action		• Password-protected objects are visible but inactive (disabled)
	#Page=*Name*	• Screen page change to screen page *Name* if a password protected object will be selected (normally the page which contains the password input menu). If the password is valid the defined password action will not be executed. The password protected objects will be displayed normally (state: visible and active) and not as disabled
	off	• Password-protected objects are invisible and inactive (off)
Limit1		•
Limit2		•
Action Limit1	**#Page=***Name*	• Screen page change to screen page *Name* if the entered password is invalid (only in combination with action #Page=name)
Action Limit2		•
VarValue	**s_password**	• System variable
VarType	**STRING**	• String data type (without definition: default 80 bytes)
VarState		•
Option	**Bitwise=AND**	• PWL interpretation bitwise and not corresponding to the value of authorization level
	Master_PW=*x*	• Definition of the master password Password with highest authorization level
	SysPW=Off	• Disable system password (calculation of the password based on day and month) Default: enable
	Timeout=*time*	• Time in minutes without touch event until the current authorization is automatically reset (PWL=0)
	Keep_PWL	• Keep active authorization level after wrong password input
C function	*C function name*	• Name of C function (see Integration of C functions)

Function of password management
Password management can be used to implement up to 32767 authorization levels. Each object can be assigned a particular password level using the PWL=x option. If the current password level is lower than the one required, the object concerned is visible and inactive (object status disabled) or invisible and inactive (object status: off). The password level after the program start is 0. Entering the master password (Master_PW=x) sets the highest authorization level (32767). This level can also be reached by entering the value Day * Month + Day. Using the system variable s_password_1, s_password_2, ... of type STRING, passwords can be defined with the appropriate authorization levels 1, 2, ... (**consecutively, without gaps!**). The button action PWL=x can be used to reset the current authorization level, e.g. when leaving a screen page.
Access is allowed if current PWL >= object PWL (option PWL=x).

Option Bitwise=AND
With this option it is possible to use the 15 bit of the PWL for 15 authorization levels which can be configured to an access matrix.
With the system variables s_password_1, s_password_2, ... of type STRING, it is possible to define passwords of corresponding authorization levels 1 (Bit0=1), 2 (Bit1=1), ... (**consecutively, without gaps!**).
The combination of the current PWL and the object specific PWL is done by an boolean AND operation: Access allowed if: (current PWL AND Option PWL) > 0

Example:

PasswordBit	**…**	**Bit 3**	**Bit 2**	**Bit 1**	**Bit 0**		**EPAM-Project**
current PWL		8	4	2	1		Option-PWL
Function/Level		**Master**	**Service**	**Foreman**	**Operator**		
Page1		1	0	0	1	→	9 (=0x09)
Page2		1	0	1	0	→	10 (=0x0A)
Page3		1	1	0	1	→	13 (=0x0D)
...							

Service has access to Page3, but not to Page2 and 1.
Operator has access to Page1 and 3, bit not to Page2.
After Login the current PWL will be set corresponding to the password level (e.g. Service Bit2=1 → PWL=4). Now all objects (with PWL) are available which have a PWL option where the Bit 2 = 1.

The password object will be designed globally in the first screen page in the Project worksheet (initialization page)

7.21 Scroll list object

#Scrollist		Scroll list
Text/File		•
Font		•
X,Y,DX,DY	*Integer values*	• Position, width and height of the scroll list (in relation to top left!)
Color	*0-15 or color name*	• Color of the text of the scroll list
Backcolor	0-15 or color name	• Background color of scroll list
Format		• No entry means no border
	Border=Button	• Button border type
	Border=Input	• Input field border type
	Border=R*x*	• Border type, rectangle with width *x* pixels (1, 3, 5, etc.)
	Border=Shadow	• Border type, rectangle with shadow (3D effect)
	Border=Signal	• Signal field border type
Action	**Scrolly=***x*	• Move scroll list vertically by x pixel(s)
		•
Limit1		•
Limit2	*Integer value*	• Value must be calculated as follows: Number of entries in scroll list – Visible entries in scroll list + 1
Action Limit1 Action Limit2		•
VarValue	*Variable name*	• System variable saves the scroll position of the scroll list
VarType	**INT**	• Integer data type (16-bit)
	UINT	• Unsigned integer data type (16-bit)
	WORD	• Word data type (16-bit)
VarState		•
Option	**DX=***x*	• Width of the scroll bar in pixels (default: 25 pixels)
C function	*C function name*	• Name of C function (see Integration of C functions)

Function of the scroll list
The scroll list can be used to move in the X or Y direction all subsequent objects that are defined with the 'scroll' option. Object that are completely or partly outside of the scroll list area are not shown. The objects are positioned as if the screen had the required dimension (e.g. 800 pixels high)

The Scroll list object thus makes it possible to design parameter lists with standard objects in any form. **When combined with the indexed variable addressing option, parameters for e.g. several motion controls can be entered on one screen page.**

Within a page there can only be ONE Scroll-List object at the same time!

7.22 DataLog object

#DataLog		Recording of PLC data/variables in a DataLog file
Text/File	*MyDatalog.***TXT**	• Name of the DataLog definition file (parameter file) with the DataLog parameters
Font		•
X,Y,DX,DY		•
Color		•
Backcolor		•
Format		•
Action		•
Limit1		•
Limit2		•
Action Limit1 Action Limit2		•
VarValue	*Variable name*	• PLC variable PLC variable of the DataLog structure
VarType	**STRING**	• Data type Array of Byte (length according to the length of the DataLog structure)
VarState		•
Option	**Type=**name	• Definition of Datalog-Type. (for use of existing datalog-files from other objects)
C function		•

Function of the DataLog
The DataLog object is used for recording PLC data in a file. The entries are made in the form of ASCII text. The columns are delimited by a separator. Each entry uses one line and is supplemented by a time stamp.
The data interface to the PLC is any structure of elementary PLC variables. The first record field (Ctrl) must always be of DWORD type. The PLC can control the DataLog object by means of this variable. Each further structure element is logged if an appropriate entry exists in the parameter file.

Caution:
The entire length of this structure must not exceed 80 bytes
As the individual variables stored in the structure are word aligned (even addresses), the additional bytes must also be taken into account.
The DataLog object is not suitable for continuous logging of rapidly changing variables on CompactFlash cards. The write cycles of CompactFlash memories are limited (normally 100,000 write cycles, for details see specifications of CompactFlash used).
The DataLog object therefore normally logs the data in the RAM drive. Button actions can be used if required to save this data on the CompactFlash.

DataLog record field Ctrl with the following functions:

16#0000001 Trigger Ctrl bit
By setting the Trigger Ctrl bit, the PLC requests EPAM to start logging the DataLog entries (with #dt=x entry in the parameter file) or only one DataLog entry (without #dt=x entry in the parameter file).

16#0000002 Acknowledge trigger Ctrl bit
By setting the AcknowledgeTrigger Ctrl bit, EPAM informs the PLC that the DataLog entry was recorded (without #dt=x entry in the parameter file). The Acknowledge Trigger Ctrl bit must be reset by the PLC.

16#0000004 Reset Ctrl bit
By setting the Reset Ctrl bit, the PLC requests EPAM to delete the DataLog file in the EPAM log directory (EPAM RAM drive:).

16#0000008 Save Ctrl bit
By setting the Save Ctrl bit, the PLC requests EPAM to save the DataLog file in the EPAM data directory (C:\DATA). The Save Ctrl bit must be reset by the PLC.

16#0000010 HMI Reset Ctrl bit
By setting the HMI Reset Ctrl bit, EPAM informs the PLC that the DataLog file in the EPAM log directory (EPAM RAM drive:) was deleted with the logdelete=*MyDatalog* button action. The HMI Reset Ctrl bit must be reset by the PLC.

16#0000020 HMI Save Ctrl bit
By setting the HMI Save Ctrl bit, EPAM informs the PLC that the DataLog file was saved in the EPAM log directory (C:\DATA) with the logsave=*MyDatalog* button action. The HMI Save Ctrl bit must be reset by the PLC.

Triggering the log function

1. With entry #dt=x in the parameter file:
   - The DataLog operation is started with the Trigger Ctrl bit set in the PLC.
   - The DataLog operation is stopped by resetting the Trigger Ctrl bit in the PLC.

2. Without entry #dt=x in the parameter file:
   - One DataLog entry is saved by setting the Trigger Ctrl bit in the PLC.
   - If the DataLog entry was written, the Acknowledge Trigger Ctrl bit is then set by EPAM.
   - The Trigger Ctrl bit must then be reset with the PLC. The next DataLog entry can be saved with the next setting of the Trigger Ctrl bit with the PLC.

File handling

If no file name was specified (#file=), the object generates the file name automatically from the date (yymmdd.log). In this case a new file is created with each new calendar day.
If no file is present, a new file is created. If a file is already present, the DataLog lines are added at the end of the file. If the specified file size is reached, the oldest DataLog entries are overwritten (ring buffer).

Important!
All lines within a datalog-file must have the same line length! → Consider max. lenght of data types in format string definition!

The DataLog object will be designed globally in the first screen page in the Project worksheet (initialization page)

7.22.1 DataLog worksheet

All object parameters are defined in the DataLog worksheet. All variables to be registered are also specified with format and type. The ANSI-C notation (see Variable object) is used as the format string. The DataLog worksheet has the following structure:

Format	VarType	Comment
#size=100		Size of Log file in KB
#file=MyLog.log		Logfile name
#seperator=;		Separator between columns
#format=user		**format=user** ...fixed width user format, larger values will be truncated (e.g. Format %3d: Value 1000 -> 999) **format=auto** ...fixed width format corresponding to VarType (format will be defined automatically, so that the largest value for VarType can be stored) **format=V3.10** ...old compatible mode, user is responsible to define formats which are big enough to hold the largest value
#dt=5		Timescale[s]
#timeformat= %d.%m.%Y %H:%M:%S		Timeformat
%1u	BOOL	BOOL variable (unsigned decimal representation)
%1hu	BOOL	BOOL variable (hexadecimal representation)
%3u	BYTE	BYTE variable (unsigned decimal representation)
%2x	BYTE	BYTE variable (hexadecimal representation)
%6hu	WORD	WORD variable (unsigned decimal representation)
%4hx	WORD	WORD variable (hexadecimal representation)
%8lu	DWORD	DWORD variable (unsigned decimal representation)
%8lx	DWORD	DWORD variable (hexadecimal representation)
%4d	SINT	SINT variable (signed decimal representation)
%6d	INT	INT variable (decimal representation)
%12ld	DINT	DINT variable (decimal representation)
%3u	USINT	USINT variable (unsigned decimal representation)
%5hu	UINT	UINT variable (unsigned decimal representation)
%12lu	UDINT	UDINT variable (unsigned decimal representation)
%4.4f	REAL	REAL variable (floating point representation)
%e	REAL	REAL variable (exponential representation)

Variables of type STRING are not supported in the DataLog!

Example of a type definition (IEC61131):

To do this the object must be assigned with a PLC variable with the following structure:

Global variable:

```
VAR_GLOBAL
        Datalog         :   DatalogType;
END_VAR
```

PLC data type:

```
TYPE DatalogType:
    STRUCT
        Ctrl                :   DWORD;
        TimeStamp           :   DWORD;    …Time stamp of PLC (>0), Time stamp of EPAM (=0)
        VisuVarBoolDecimal  :   BOOL;
        VisuVarBoolHex      :   BOOL;
        VisuVarByteDecimal  :   BYTE;
        VisuVarByteHex      :   BYTE;
        VisuVarWordDecimal  :   WORD;
        VisuVarWordHex      :   WORD;
        VisuVarDwordDecimal :   DWORD
        VisuVarDwordHex     :   DWORD;
        VisuVarSintDecimal  :   SINT;
        Dummy1              :   SINT;     …Additional byte calculated (word aligned)
        VisuVarIntDecimal   :   INT;
        VisuVarDintDecimal  :   DINT;
        VisuVarUsintDecimal :   USINT;
        Dummy2              :   USINT;    …Additional byte calculated (word aligned)
        VisuVarUintDecimal  :   UINT;
        VisuVarUdintDecimal :   UDINT;
        VisuVarRealDecimal  :   REAL;
        VisuVarRealDecimal  :   REAL;
    END_STRUCT
END_TYPE
```

7.23 Trend object

#Trend		Display of the DataLog file as a trend graph
Text/File	*MyTrendDef.***TXT**	• Name of the Trend definition file (parameter file) with the Trend parameters
Font	*Font.***FNT**	• Optional font for the text
X,Y,DX,DY	*Integer values*	• Position, width and height of trend (in relation to top left!)
Color	*0-15 or color name*	• Color of the Trend text
Backcolor	0-15 or color name	• Background color of the trend
Format		•
Action		•
Limit1		•
Limit2		•
Action Limit1 Action Limit2		•
VarValue		•
VarType		•
VarState		•
Option	**Type=**name	• Definition of Trend-Type. (for use of existing trend-files from other objects)
C function		•

Function of Trend

The Trend object enables data recorded with the DataLog object to be displayed in the form of a trend graph (max. 4 curves in a trend). The values can be shown online and offline. The manipulation of the current section is carried out using button actions (see Button object).

The Trend is normally displayed as Y/T-diagram. With option format_user=%ld it is possible to use the timestamp value (double word in Datalog) as a user formatted X value e.g. as integer value (1,2,3,...). So it is possible to display X/Y-diagrams e.g. of temperature values.

7.23.1 Trend worksheet

All object parameters are defined in the Trend worksheet. The Trend worksheet has the following structure:

Format		Comment
#Title=		...Diagrammtitle
#Seperator=;		...Seperator in datalogfile
#Orientation=horizontal		...Orientation horizontal or vertical
#Flow=Right2Left		...Flow of Trend Left2Right or Right2Left
#File=datalog.log		...[Path/]Name of the datafile (path optional, without path the PATH_LOG will be used) Filename my contain a system variable e.g. #File=dlog%s_idx%.log
#X=[t]		...Name of X-Axis
#format_time=%H:%M:%S		...Time/User Format of the X-axis
#Format_date=%d-%m-%y		...Date Format of the X-axis (should be empty if Format_user is selected)
#range=900		...Range of X-Axis in seconds/user
#dt=1		...Sampletime in seconds
#GridX=180		...Lettering Grid on X
#dtTolerance=0		...number of missing datapoints before break the curve
#ScaleX=On		...scale drawing off/on
#y=[C]		...1. Y-Curve (Name,color)
#Type=INT		...IEC Datatype (BYTE, SINT, USINT, WORD, INT, UINT, DWORD, DINT, UDINT, REAL)
#color=green		...curve-color
#ScaleColor=green		...scale-color
#max=150		...Ymax
#min=0		...Ymin
#GridY=15		...Lettering Grid on Trend 1
#Grid=Off		...on = draw horicontal grid lines
#Scale=On		...scale drawing off/on
#DataLogCol=1		...select data column within datalog file
#y=[C]		...2. Y-Curve (Name,color)
#Type=INT		...IEC Datatype (BYTE, SINT, USINT, WORD, INT, UINT, DWORD, DINT, UDINT, REAL)
#color=blue		...curve-color
#ScaleColor=blue		...scale-color
#max=20		...Ymax
#min=0		...Ymin
#GridY=2		...Lettering Grid on Trend 2
#Grid=Off		...on = draw horicontal grid lines
#Scale=On		...scale drawing off/on
#DataLogCol=2		...select data column within datalog file
#y=[bar]		...3. Y-Curve (Name,color)
#Type=REAL		...IEC Datatype (BYTE, SINT, USINT, WORD, INT, UINT, DWORD, DINT, UDINT, REAL)
#color=red		...curve-color
#ScaleColor=red		...scale-color
#max=10		...Ymax
#min=0		...Ymin
#GridY=1		...Lettering Grid on Trend 3
#Grid=Off		...on = draw horicontal grid lines
#Scale=On		...scale drawing off/on
#DataLogCol=3		...select data column within datalog file
#y=[bar]		...4. Y-Curve (Name,color)
#Type=REAL		...IEC Datatype (BYTE, SINT, USINT, WORD, INT, UINT, DWORD, DINT, UDINT, REAL)
#color=brown		...curve-color
#ScaleColor=brown		...scale-color
#max=3		...Ymax
#min=0		...Ymin
#GridY=1		...Lettering Grid on Trend 4
#Grid=Off		...on = draw horicontal grid lines
#Scale=On		...scale drawing off/on
#DataLogCol=4		...select data column within datalog file

7.24 Sys2Plc object

#Sys2Plc		Synchronisation of EPAM-system variables and PLC variables (two way communication)
Text/File	*MySysPlc.***TXT**	• Name of the Sys2Plc definition file
Font		•
X,Y,DX,DY		•
Color		•
Backcolor		•
Format		•
Action		•
Limit1		•
Limit2		•
Action Limit1 Action Limit2		•
VarValue		•
VarType		•
VarState		•
Option		•
C-Function		•

Function of Sys2Plc
The Sys2Plc object can be used to send information from EPAM system variables to PLC variables and vice versa. The Sys2Plc definition file contains the variable list of the corresponding variables.
How it works:
Startup: System variables will be written to the corresponding PLC-variables
Operation: after changes of system variables the corresponding PLC variables will be overwritten with the new value (this is based on events, e.g. input of a new value). If a PLC-variable has changed the new value will be written to the corresponding system variable. This is done in a cyclic way **(cycle time: 0.5s)**.

The Sys2PLC object will be designed globally in the first screen page in the Project worksheet (initialization page)

7.24.1 Sys2Plc worksheet

All variable pairs are defined in the Sys2Plc worksheet. The Sys2Plc worksheet has the following structure:

System-Variable	PLC-Variable	
s_mysysvar	PLC/myplcvar	
…	…	
…	…	

The EPAM system variable and the corresponding PLC variable (variable pair) must be of the same data type.

Triggering screen page changes in the PLC
The Sys2Plc object can also be used to trigger screen page changes on the basis of variable values by defining **s_newpage** and a PLC variable of type STRING. If the PLC variable e.g. PLC/NewPagename is set to a value like '#Page=*name'* the value will be written to the EPAM system variable s_newpage and this causes a page change to this page.

7.25 RemoteControl object

#RemoteControl		Remote control of EPAM-applications within a network of devices
Text/File	xxx.xxx.xxx.xxx	• IP-address
Font	*Font.***FNT**	• Optional font for the text
X,Y,DX,DY	*Integer values*	• Position, width and height of trend (in relation to top left!)
Color	*0-15 or color name*	• Color of the Trend text
Backcolor	0-15 or color name	• Background color of the trend
Format		•
Action		•
Limit1		•
Limit2		•
Action Limit1 Action Limit2		•
VarValue	*Variable name*	• PLC- or system variable wit actual IP-address of the Remote-System
VarType	**STRING**	• Data type STRING (IP-address: xxx.xxx.xxx.xxx)
VarState		•
Option		•
C-Function		•

Function of RemoteControl
The RemoteControl object can be used to control another EPAM-application on another device within an Ethernet network remote from a central station. The RemoteControl object is a window for the user to look to the screen of a remote station. Inputs e.g. by Touch within this window will be transfered to the remote station and will be handled like a local Input on the remote device. Vice versa changes on screen of the remote station will be transferred and displayed within the Remote Control window. (like PC-Anyware).

Remote control of devices with the same screen resolution
To control a remote device with the same screen resolution (fullscreen Remote Control object), it is possible to use an invisible-Button with the action #Page=xy and option timeout (e.g. lower right, DX, DY 1 Pixel). In this case the Remote-control object will be released automatically after the specified timeout without operation.
Otherwise the device which uses the RemoteControl object needs a higher screen resolution then the remote device. E.g. a VGA-device (640x480) can display a remote device with 1/4VGA screen (320x240).

System variables
Following system variables will be used together with RemoteControl-object:

s_rc_password	…Password for RemoteControl-Server (alternative to option Password=)
s_remoteclient_connected	…shows on the Remote-device (where the RemoteControl-server is running), if a client is connected (1 = RemoteControl-client connected)

8 Application Notes

8.1 Alarmhandling

8.1.1 Alarmdefinition

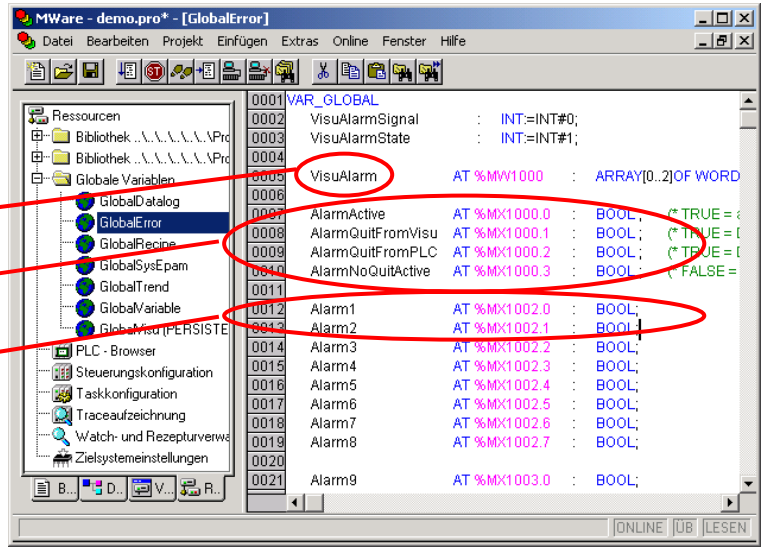
Global EPAM object #Alarm

Object	Text/File	Font	X [Pixel]	Y [Pixel]	DX [Pixel]	DY [Pixel]	Color	Backcolor	Format	VarValue	VarType
#Page=Init			0	0	0	0	Black	white			
#Alarm	MyAlarm1.TXT									PLC/VisuAlarm	STRING
#Page=Production			0	0	320	240	black	white			

Alarmdefinition in worksheet MyAlarm1

Variable	Alarm number	Text/File	Font	Color	Backcolor	Action Alarm Info	Alarm Helptext
[0].0		1 = Alarm(e) activ					
[0].1		1 = Delete alarm request from visualization					
[0].2		1 = Delete alarm request from PLC					
[0].3		0 = all Alarm(s)					
[0].4 [0].15		reserved					
[1].0	1	User Alarmtext 1	Arial8.FNT	black	red	#Page=Diagnose,#Page=Foto1	alrmhlp1.txt
[1].1	2	User Alarmtext 2	Arial8.FNT	black	red	#Page=Diagnose,#Page=Foto1	alrmhlp1.txt
...	...	...	...	...	...	...	...
[1].15	16	User Alarmtext 16	Arial8.FNT	black	yellow	#Page=Diagnose,#Page=Foto2	Alrmhlp2.txt
[2].0	17	User Alarmtext 17	Arial8.FNT	black	yellow	#Page=Diagnose,#Page=Foto2	Alrmhlp3.txt
...	...	...	...	...	...	...	...
[2].14	31	User Alarmtext 31	Arial8.FNT	black	white	#Page=Diagnose,#Page=Foto3	
[2].15	32	User Alarmtext 32	Arial8.FNT	black	white		

IEC61131 PLC-Alarmbuffer „VisuAlarm“



Alarm definition contains:

- Alarmbit → Alarmnumber
- Text, Font, color definition
- Pages for diagnosis
- Helptext

How it works

A global alarm object checks the PLC alarm buffer cyclically (IEC61131 variable VisuAlarm)

Coming Alarms

- New coming alarms [x].x = 1 (IEC61131 variables Alarm1..32) will be stored with a coming time stamp and inserted in the internal EPAM alarm buffer (alarm history 512 alarms)
- Status bit [0].0 (IEC61131 variable AlarmActive) and [0].3 (IEC61131 variable AlarmNoQuitActive) will be set to 1 by EPAM

Outgoing Alarms

- Outgoing alarms [x].x = 0 (IEC61131 variables Alarm1..32) will be stored with a outgoing time stamp in the internal EPAM alarm buffer

Acknowledging of alarms

- If alarms are acknowledged from EPAM, the acknowledge time stamp is written to the internal EPAM alarm buffer and the status bit (IEC61131 variable AlarmQuitFromVisu) will be set to 1 by EPAM
- To acknowledge alarm via PLC the status bit [0].2 (IEC61131 variable AlarmQuitFromPLC) in the PLC must be set to 1. Now for all active alarms within the internal EPAM alarm buffer the alarm acknowledge time stamp will be written.
- The status bit [0].3 (IEC61131 variable AlarmNoQuitActive) will stay active until all alarm are acknowledged.

8.1.2 Alarm display

Alarm definition in worksheet MyAlarm1

Variable	Alarm number	Text/File	Font	Color	Backcolor	Action Alarm Info	Alarm Helptext
[0].0		1 = Alarm(s) activ					
[0].1		1 = Delete alarm request from visualization					
[0].2		1 = Delete alarm request from PLC					
[0].3		0 = all Alarm(s)					
[0].4-[0].15		reserved					
[1].0	1	User Alarmtext 1	Arial8.FNT	black	red	#Page=Diagnose,#Page=Foto1	alrmhlp1.txt
[1].1	2	User Alarmtext 2	Arial8.FNT	black	red	#Page=Diagnose,#Page=Foto1	alrmhlp1
...	...	...	...	...	...	...	...
[1].15	16	User Alarmtext 16	Arial8.FNT	black	yellow	#Page=Diagnose,#Page=Foto2	Alrmhlp2.txt
[2].0	17	User Alarmtext 17	Arial8.FNT	black	yellow	#Page=Diagnose,#Page=Foto2	Alrmhlp3.txt
...	...	...	...	...	...	...	...
[2].14	31	User Alarmtext 31	Arial8.FNT	black	white	#Page=Diagnose,#Page=Foto3	
[2].15	32	User Alarmtext 32	Arial8.FNT	black	white		

IEC61131 PLC-Alarm buffer „VisuAlarm“
Alarm1 := TRUE
Alarm2 := TRUE

Alarm definition contains:
- Alarmbit → Alarmnumber
- Text, Font, color definition
- Pages for diagnosis
- Helptext

EPAM object #AlarmList

Object	Text/File	Font	X [Pixel]	Y [Pixel]	DX [Pixel]	DY [Pixel]	Color	Backcolor	Format	VarValue	VarType	Option
#Page=AlarmList			0	0	320	240	black	white				
#Alarmlist	No=%4d,sep	SYS08X16.FNT	0	0	320	160	red	white	...			DX=40,alarmfilter=activ|notquit,alarmsort=lifo, AlarmType=MyAlarm1

Format of the alarm line
Alarm Nr. Alarmtext

Format of the alarm list

How it works

The object AlarmList displays the internal EPAM alarm buffer on screen in from of a list

Display of alarm buffer in a list

- The selected alarm is displayed reverse
- The display options for the alarm list (sort, filter) can be predefined by options or changed by button actions
- The actual alarm information of the selected alarm are stored in system variables s_alarm_nr, s_alarm_text, s_alarm_tin, s_alarm_tout, s_alarm_tquit, s_alarm_info, s_alarm_txtinfo (for alarm diagnose)

Acknowledging of alarms

- Alarme can be acknowledged by button actions AlarmQuit one by one or all together with button action AlarmQuitAll

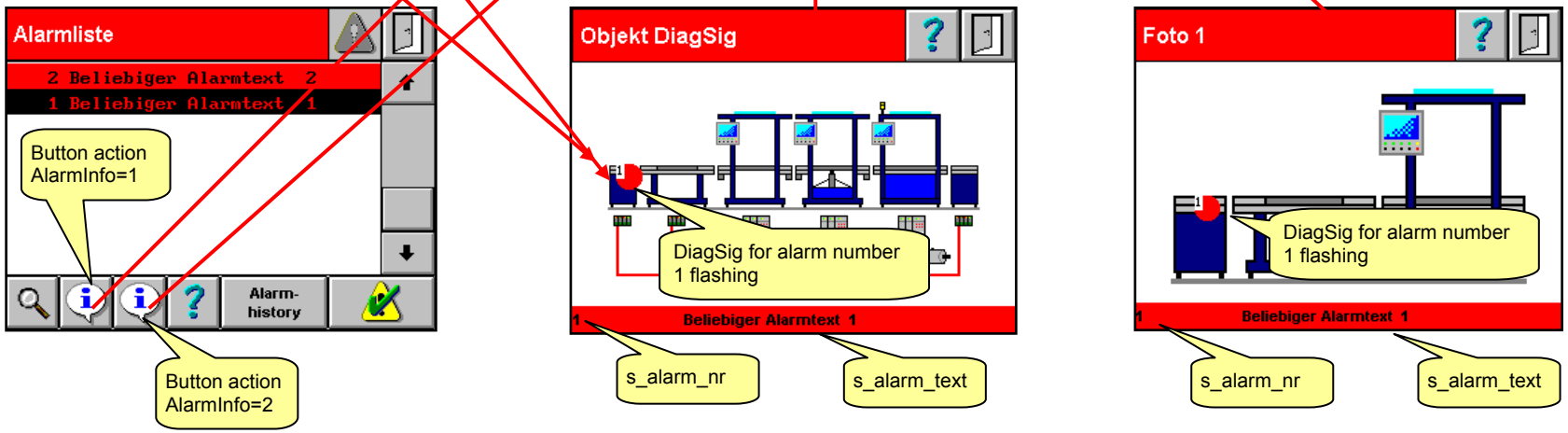
8.1.3 Alarmdiagnose

Alarm definition in worksheet MyAlarm1

Variable	Alarm number	Text/File	Font	Color	Backcolor	Action Alarm Info	Alarm Helptext
[0].0		1 = Alarm(s) activ					
[0].1		1 = Delete alarm request from visualization					
[0].2		1 = Delete alarm request from PLC					
[0].3		0 = all Alarm(s)					
[0].4-[0].15		reserved					
[1].0	1	User Alarmtext 1	Arial8.FNT	black	red	#Page=Diagnose,#Page=Foto1	alrmhlp1.txt
[1].1	2	User Alarmtext 2	Arial8.FNT	black	red	#Page=Diagnose,#Page=Foto1	alrmhlp1.txt
...	...	...	...	...	...	...	...
[1].15	16	User Alarmtext 16	Arial8.FNT	black	yellow	#Page=Diagnose,#Page=Foto2	Alrmhlp2.txt
[2].0	17	User Alarmtext 17	Arial8.FNT	black	yellow	#Page=Diagnose,#Page=Foto2	Alrmhlp3.txt
...	...	...	...	...	...	...	...
[2].14	31	User Alarmtext 31	Arial8.FNT	black	white	#Page=Diagnose,#Page=Foto3	
[2].15	32	User Alarmtext 32	Arial8.FNT	black	white		

EPAM object #DiagSig

Object	Text/File	Font	X [Pixel]	Y [Pixel]	DX [Pixel]	DY [Pixel]	Color	Backcolor	Format	VarValue	VarType
#Page=ObjectDiagSig			0	0	320	240	Black	white			
#Signal	Machine.pex		25	60	270	140	black	white	...		
#DiagSig	1		32	115	20	20	red		...	s_alarm_nr	STRING
#Variable	%4d	...	8	220	80	20	black	red	...	s_alarm_nr	WORD
#Variable	%s	...	80	220	240	20	black	red	...	s_alarm_text	WORD

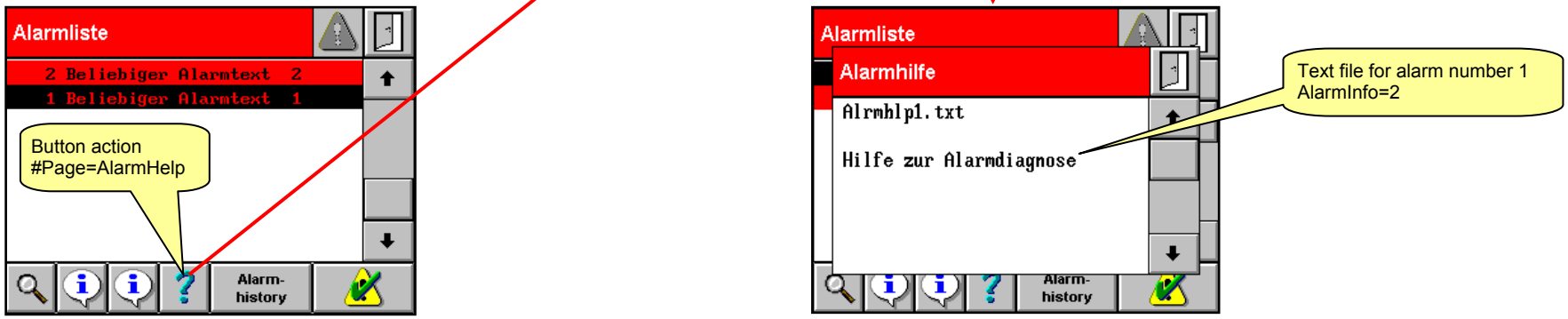


Alarm definition in worksheet MyAlarm1

Variable	Alarm number	Text/File	Font	Color	Backcolor	Action Alarm Info	Alarm Helptext
[0].0		1 = Alarm(s) activ					
[0].1		1 = Delete alarm request from visualization					
[0].2		1 = Delete alarm request from PLC					
[0].3		0 = all Alarm(s)					
[0].4-[0].15		reserved					
[1].0	1	User Alarmtext 1	Arial8.FNT	black	red	#Page=Diagnose,#Page=Foto1	Alrmhlp1.txt
[1].1	2	User Alarmtext 2	Arial8.FNT	black	red	#Page=Diagnose,#Page=Foto1	Alrmhlp1.txt
...	...	...	...	...	...	...	...
[1].15	16	User Alarmtext 16	Arial8.FNT	black	yellow	#Page=Diagnose,#Page=Foto2	Alrmhlp2.txt
[2].0	17	User Alarmtext 17	Arial8.FNT	black	yellow	#Page=Diagnose,#Page=Foto2	Alrmhlp3.txt
...	...	...	...	...	...	...	...
[2].14	31	User Alarmtext 31	Arial8.FNT	black	white	#Page=Diagnose,#Page=Foto3	
[2].15	32	User Alarmtext 32	Arial8.FNT	black	white		

EPAM object #TextList / Variable s_alarm_txtinfo

Object	Text/File	Font	X [Pixel]	Y [Pixel]	DX [Pixel]	DY [Pixel]	Color	Backcolor	Format	VarValue	VarType
#Page=AlarmHelp			15	30	290	180	Black	white	...		
...		...	...	...	...	...	...	...	...		...
#TextList	s_alarm_txtinfo	...	0	40	290	140	black	white	...		STRING



How it works

Button action AlarmInfo=1

- The action AlarmInfo=1 reads out the first page which is defined in the worksheet „MyAlarm1“ in column Action Alarm Info for the selected alarm within the alarm list and changes to this page.

Button action AlarmInfo=2

- The action AlarmInfo=2 reads out the second page which is defined in the worksheet „MyAlarm1“ in column Action Alarm Info for the selected alarm within the alarm list and changes to this page.

Text list with variable s_alarm_txtinfo

It is possible to define one page with a text list object to display all the alarm specific textfiles. For that the system variable s_alarm_txtinfo can be defined in the column Text/File. Now you can call the page with the text list e.g. as a alarm specific help page. The text list now displays the actual textfile for the alarm which is defined in worksheet “MyAlarm1” in column textfile. (the alarm list object with a selected alarm must be active -> page with text list object must be a window)

8.2 Recipe handling

8.2.1 Recipe definition

Global EPAM object #Recipe

Object	Text/File	Font	X [Pixel]	Y [Pixel]	DX [Pixel]	DY [Pixel]	Color	Backcolor	Format	Action	VarValue
#Page=Init			0	0	0	0	Black	white			
#Recipe	MyTyp1.TXT									#Page=MyTyp1Overwrite	PLC/VisuMyTyp1UpDnlo
#Page=Production			0	0	320	240	black	white			

Recipe definition in worksheet MyAlarm1

Variable	VarType	Value
#Recipe=Default		Path=
PLC/VisuAx1.mmiId	INT	
PLC/VisuAx1.mmiFunc	SINT	
PLC/VisuAx1.mmiNomVel	DINT	
PLC/VisuAx1.mmiNomAcc	DINT	
PLC/VisuAx1.mmiScaleN	INT	
PLC/VisuAx1.mmiScaleM	INT	
PLC/VisuAx1.mmiStopDec	INT	
PLC/VisuAx1.mmiNullPos	DINT	
PLC/VisuAx1.mmiVelKpRed	DINT	
PLC/VisuAx1.mmiCamTable	INT	
...	...	
'#Checksum=		

IEC61131 PLC Recipe state-variable:

1 ...Download-request from PLC
2 ...Up/Download finished
3 ...Upload request from PLC
4 ...Download in progress
5 ...Upload in progress

Confirmation page:

This page will be displayed if an action csave=***MyTyp1*** writes a recipe file which already exists. (on this page it is possible to display a confirmation question „Overwrite existing recipe file ?“)

Recipe options:

no option:	at start up actual recipe values will be downloaded form EPAM to the PLC (ACTUAL.DAT)
NoActual:	at start up EPAM downloads the last loaded recipe file
NoDownload:	no recipe values will be downloaded to the PLC

How it works

- A global recipe object with PLC recipe state variable (IEC61131 variable VisuMyTyp1UpDnload)
- The corresponding system variables **s_*MyTyp1*_file** and **s_*MyTyp1*_name** contains file names (ISO 9669, 8.3 with limited character set) and recipe names (80 characters) of a recipe.
- With button actions **load=*MyTyp1***, **save=*MyTyp1***, **csave=*MyTyp1*** and **delete=*MyTyp1*** the actual recipe defined in system variable **s_*MyTyp1*_file** will be loaded, written or with confirmation (#Page=MyTyp1RecipeOverwrite) overwritten or deleted.
- Recipes will be written as ASCII files xxxxxx**.DAT** (x = content of variable **s_*MyTyp1*_file**) into directory C:\DATA**MyTyp1**

8.2.2 *Recipe list*

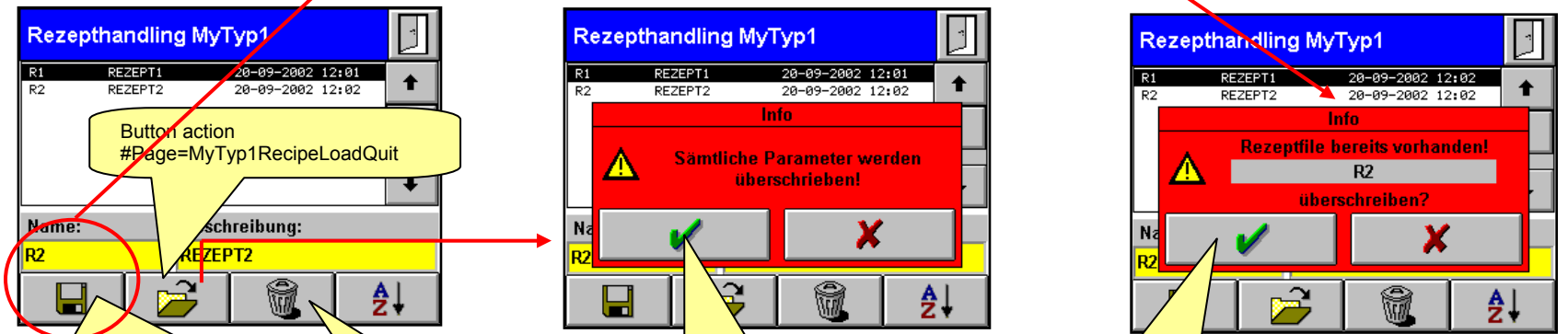
Global EPAM object #Recipe

Object	Text/File	Font	X [Pixel]	Y [Pixel]	DX [Pixel]	DY [Pixel]	Color	Backcolor	Format	Action	VarValue	VarType	VarState	Option
#Page=Init			0	0	0	0	Black	white						
#Recipe	MyTyp1.TXT									#Page=MyTyp1Overwrite	PLC/VisuMyTyp1UpDnload	INT		
#Page=Production			0	0	320	240	black	white						

EPAM object #RecipeList

Object	Text/File	Font	X [Pixel]	Y [Pixel]	DX [Pixel]	DY [Pixel]	Color	Backcolo	Format	Action	VarValue	VarType	Option
#Page=MyTyp1RecipeHandling			0	0	320	240	black	white					
#Recipelist	%-9[%-16n %d-%m-%Y %H:%M	SYS06X11.FNT	1	40	318	110	black	white					
#Signal	Name:	ARIAL10F.FNT	1	155	114	20	black	grey					
#Variable	%8s	ARIAL10F.FNT	1	175	114	25	black	yellow		#Page=KeybAlnum	S_MyTyp1_file	STRING	
#Signal	Beschreibung:	ARIAL10F.FNT	119	155	200	20	black	grey					
#Variable	%15s	ARIAL10F.FNT	119	175	199	25	black	yellow		#Page=KeybAlnum	S_MyTyp1_name	STRING	
#Button	save.ico		0	200	80	40	black	grey		Csave=MyTyp1			
#Button	open.ico		80	200	80	40	black	grey		#Page=MyTyp1RecipeLoadQuit			
#Button	trash.ico		160	200	80	40	black	grey		#Page=MyTyp1RecipeDeleteQuit			

Format of recipe list
File name,
Description,
Date



Button action
Csave=***MyTyp1*** checks if file R2.DAT already exists:
Not existing ⇒ save
existing ⇒ change to page #Page=MyTyp1Overwrite

Button action
Delete=List deletes the recipe file selected in recipe list (R1.DAT)

Button action
Load=list loads the recipe selected in recipe iste (R1.DAT)

Button action
Save=***MyTyp1*** saves the actual recipe values in file R2.DAT

9 System variables

System variables are global variables that are only required for visualization. System variables start with the prefix ‘s_’ and are saved retentively in the sysvar.ini file. The following system variables are predefined:

System variable	Meaning	Data type
System variable in for object Alarm/Alarmlist		
s_alarm_nr	Alarm number of the last selected alarm in the alarm list	WORD
s_alarm_text	Alarm text of the last selected alarm in the alarm list	STRING
s_alarm_tin	Time of "Come" alarm of the last selected alarm in the alarm list	STRING
s_alarm_tout	Time of "Go" alarm of the last selected alarm in the alarm list	STRING
s_alarm_tquit	Time of "Acknowledged" alarm of the last selected alarm in the alarm list	STRING
s_alarm_tin_dt	Time of "Come" alarm of the last selected alarm in the alarm list	IEC_DT
s_alarm_tout_dt	Time of "Go" alarm of the last selected alarm in the alarm list	IEC_DT
s_alarm_tquit_dt	Time of "Acknowledged" alarm of the last selected alarm in the alarm list	IEC_DT
s_alarm_info	Name of the configured screen page of the Alarminfo action of the last selected alarm in the alarm list	STRING
s_alarm_txtinfo	Variable with the name of the ASCII text file with the alarm text information (used with text list)	STRING
s_alarm_activ	Variable is set if alarm is active	INT
s_alarm_type	Variable with the name of alarm type (if more then one alarm objects used)	STRING
System variable for object Recipe/Recipelist		
s_*myrecipetype*_file	Recipe file name without extension for each defined recipe type	STRING
s_*myrecipetype*_name	Recipe name for each defined recipe type	STRING
s_*myrecipetype*_cur_file	Current selected recipe file name within recipe list without extension for each defined recipe type	STRING
s_*myrecipetype*_cur_name	Current selected recipe name within recipe list for each defined recipe type	STRING
s_*myrecipetype* _dnload_max	Number of recipe variables of the corresponding recipe type for download (*MyRecipetype*=user-defined recipe type) The system variable is created for each recipe file defined	WORD
s_*myrecipetype*_dnload_act	Current number of loaded recipe variables for download (progress display)	WORD
s_*myrecipetype*_upload_max	Number of recipe variables of the corresponding recipe type for upload (*Myrecipetype*=user-defined recipe type) The system variable is created for each recipe file defined	WORD
s_*myrecipetype*_upload_act	Current number of loaded recipe variables for upload (progress display with bar)	WORD
s_recipe_type	Currently selected recipe type	STRING
s_recipe_path	Current directory path of the recipes	STRING
s_recipelist_empty	1 if recipe list is empty, 0 if at least 1 recipe is in the list	INT
System variables for object Trend		
s_*mytrend*_c1	Current value at cursor position	as Trend
s_*mytrend*_c2	Current value at cursor position	as Trend
s_*mytrend*_c3	Current value at cursor position	as Trend
s_*mytrend*_c4	Current value at cursor position	as Trend
s_trend_t_sec	Time at cursor position (seconds)	DWORD
s_trend_t_min	Time at cursor position (minutes)	DWORD
s_trend_t_hour	Time at cursor position (hours)	DWORD
s_trend_t_mday	Time at cursor position (day)	DWORD
s_trend_t_mon	Time at cursor position (month)	DWORD

s_trend_t_year	Time at cursor position (year)	DWORD
s_trend_t_wday	Time at cursor position (weekday)	DWORD
s_trend_t_sec	Time at cursor position (minutes)	DWORD
s_trend_t	Unformatted Value of X-Position	DWORD
System variables for object Password		
s_password	Current password entry	STRING
s_password_x	Password for authorization level x	STRING
s_pwl	Current password level	WORD
s_user_x	Username for password level x	STRING
System variables for object Page		
s_newpage	new page name if s_newpage is set, EPAM changes to defined new page	STRING
s_pageidx	Actual page-ID (defined with option ID)	UINT
s_pagename	Name of actual page on screen	STRING
General system variables		
s_backlight	Current setting of the backlight (0-100%, Default: 100%)	WORD
s_contrast	Current setting of the contrast (0-100%, Default:50%) Only passive LCDs!	WORD
s_dns1_ip	DNS1 address of target system (Input in format xxx.xxx.xxx.xxx)	STRING
s_dns2_ip	DNS2 address of target system (Input in format xxx.xxx.xxx.xxx)	STRING
s_dhcp_mode	0 = DHCP diabled, static IP address 1 = DHCP enabled	INT
s_irtouch	1 if for devices with IR-Touch-screen; 0 for others	INT
s_rc_password	Passwort for RemoteControl-Server	STRING
s_remoteclient_connected	1 if a remote client is connected	INT
s_language	Current language	STRING
s_target_ip	Current IP address of the target system	STRING
s_toucherror	0 o.k. 1 Touchtest Error (IR-Touch only)	INT
s_gateway_ip	Current IP address of gateway	STRING
s_subnetmask	actual Subnet-Mask for the target (input in IP-Format xxx.xxx.xxx.xxx)	STRING
s_plcstate	Actual PLC state (0=Stop, 1=Run)	WORD
s_plcstate_<hostname>	current state of the (remote) control <hostname> 0=undefined 1=Run 2=Stop	WORD
s_unit_idx	current unit-system 0, 1, …	WORD
System variables for input of values		
s_helptext	Current Help text number	WORD
s_edit_val	Last value entered	STRING
s_input_val	Current entry value	STRING
s_limit1	Current lower limit value	STRING
s_limit2	Current upper limit value	STRING
System variables for Date/Time input and output		
s_tm_day	Day (1-31)	WORD
s_tm_mon	Month (1-12)	WORD
s_tm_year	Year (1980-2099)	WORD
s_tm_hour	Hours (00-23)	WORD
s_tm_min	Month (1-12)	WORD
s_tm_sec	Actual seconds (00-59) for display	WORD
s_tm_nsec	Nominal seconds (00-59) for entry	WORD
s_tm_wday	Weekday (0-6; 0 = Sunday)	WORD
s_tm_isdst	s_tm_isdst > 0 ...DST time s_tm_isdst 0 ...Normal time	WORD

	s_tm_isdst < 0 ...Information not available	
System variables for project information		
s_epam_version	Current EPAM version	STRING
s_epam_date	Current EPAM date (creation date)	STRING
s_projectname	Current project name	STRING
s_projectprogrammer	Current project programmer	STRING
s_projecttarget	Current project target system	STRING
s_projectversion	Current project version	STRING

Saving system variables
System variables are retentively saved in the SYSVAR.INI file with the following actions and reloaded the next time EPAM is started.

- Language selection
- Loading,saving,deleting recipes
- Button action: save=SysVar

10 Error messages

The following is a list of the possible EPAM error messages that may be output during the runtime. Error messages with the text “Error in Line xxx” refer to the relevant lines in the Excel file:

Error message	Meaning	Possible cause/solution
Exit program ???	EPAM should be closed	ESC key was pressed
EPAM Demoversion Will exit after 1 hour	EPAM demo version installed. The operation of the application is limited to one hour!	For a full version of EPAM a product code is required.
Error in *Line* File not found *Filename*	The file called *Filename* defined in *Line* could not be opened	File is already opened (e.g. by another application) File is not in the current project directory
Error in *Line* File read error *Filename*	File called *Filename* could not be read correctly	File incomplete or faulty
Error in *Line* File write error *Filename*	File called *Filename* could not be written correctly	Insufficient memory available on disk
No more dynamic memory (Heap) *Line*	No dynamic memory available in program line: *Line*	Insufficient memory available
Error in *Line* Unknown object *#Object*	*#Object* does not exist	Syntax errors
Error in *Line* Page not found *Page*	Screen page change to non-existent screen page	Syntax error or screen page not available
System error: Heap check	System errors: Dynamic memory no longer consistent	Restart computer, contact Technical Support if the error occurs again
No page in textfile	Script file contains no screen page	Empty or invalid script file
System error in *line* xxx: Null pointer	System error in program line *Line*: Null pointer	Note program line and contact Technical Support
Error in *Line* Action not allowed Action	Action action not allowed	Syntax error or action not supported
Error in *Line* No action defined	Action missing	Action column contains space(s)
Error in *Line* Limit not allowed	Invalid limit value	Syntax error or data type conflict
Error in *Line* Variable type not allowed *Vartyp*	Invalid variable type *Vartyp*	Syntax error
Error in *Line* Limit action not allowed *action*	Invalid Limit action *action*	Syntax error or action not supported
Error in *Line* SetVar action not allowed *action*	Invalid SetVar action *action*	Syntax error or action not supported
Error in *Line* Value not allowed *value*	Invalid variable value *value*	Syntax error or data type conflict, e.g. string instead of value
Error in *Line* Unknown key *Key*	Invalid key *Key*	See Standard keyboard table
Error in *Line* No key defined	Key missing	Key= action without key code
Error in *Line* Unknown color definition *color*	Invalid color or background color definition *color*	Color name or number invalid (see Standard color palette)
Too many system variables	Max. number of system variables exceeded	More than 1024 system variables defined
Not enough memory for system variables Name	Too little dynamic memory for system variable name	See No more dynamic memory (Heap)
Alarm number not allowed	Invalid alarm number	Alarm number must be an integer
Too many alarms defined	More than 1008 alarms defined	
Error in *Line* Unknown PCX format *Filename*	PCX file format not compatible with current video resolution	Example: display a 256-color PCX image in 16-color VGA mode
Inconsistent order in language file *Filename line* xxx	Language file and script file inconsistent	Line number does not match (objects were added or deleted in the script file or language file)

Inconsistent alarm data in file *name*	The data in the ALARM.INI file is inconsistent	File faulty, incomplete or no longer compatible with current EPAM version. Solution: Delete file
Inconsistent sysvar.ini file	The data in the SYSVAR.INI file is inconsistent	File faulty, incomplete or no longer compatible with current EPAM version. Solution: Delete file
Inconsistent recipe variable in file *name*	The data type of a recipe variable does not match the data type of an object that uses the same variable	Recipe variables and object variables must have the same data type
Invalid recipe file	Invalid recipe file Name	File is not a recipe file or syntax error in file
Too many recipe variables defined	Too many recipe variables defined	A maximum of 6553 recipe variables can be defined
Type conflict variable	Type conflict recipe variable name	Type in recipe variable and in current object do not match
Undefined variable in file varname	Error when downloading recipe files	Variable varname on PLC not defined
Error on DRV: <ARTI> Host: <PLC> There is no SDD assigned to the channel.	No symbol file present	Check Codesys project options
Failed to read variable Variable name	Variable not in symbol file	Old, incorrect PLC project on target
Error on DRV: <ARTI> Host: <PLC> No project on target	No PLC project on target	Transfer PLC project to target
Error on DRV: <ARTI> Host: <PLC> No symbol file on target	No file on the target or failed to open.	Transfer PLC symbol file to target
Error on DRV: <ARTI> Host: <PLC> SDD has changed	Load new project on target	Acknowledge with OK.
Timeout writing variable	Timeout during a variable write command	Communication error
Error on DRV: <ARTI> Host: <xxx> Host not running	Communication to Host <xxx> could not be established	No vonnection to PLC xxx or invalid IP-address
Login was refused by the target	Login to the control was not successful	Control is in a state, where access to variables is impossible (e.g. during boot sequence or during online-change) → timeout in DRVParam probably to small (should be >= Retry * DelayOnError in hosts worksheet!)
Language not found	The actual selected language is not available (sysvar.ini)	Sysvar.ini contains a language which is not available in the current project. Download project with option „Delete INI-Files“
Touch initialization failed	Error while initialising	Hardware problem with serial Touch interface, or wrong configuration in AUTOEXEC.INI or Touch error.
Invalid textfile	Invalid configuration file for object SYS2PLC	File is not a Sys2PLC-file or syntax error in file
Failed to register sysvar: varname	Error in SYS2PLC, system variable could not be created	To many system variables defined (max. 1536), or type conflict
Incomplete trend parameters	Invalid configuration file for TREND	Invalid trend file or syntax error in trend file
Invalid y-range on trend: value	Invalid scale of Y-axis	Invalid trend file or syntax error in trend file

Failed to create semaphor	Internal system error	
Unknown error	Unknown error	

11 Alphabetical index

F

G

I

K

L

M

N

O