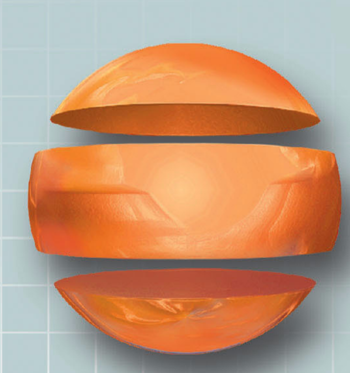
Polymath



User Manual

Rev 1.70

INDEX OF CHAPTER

Introduction **1**
What is Polymath? 1
What does Polymath do? 1
Which POLYMATH version? 1
What's new compared with VTWIN? 2
The Manual 2
Conventions used in the Manual 2
ESA Elettronica's Customer Care service 3

Installation **5**
Minimum requirements 5
Recommended requirements 6
Installing POLYMATH 7

Layout of menus **13**
Main menu 13
The Toolbar 39
Anchorable windows 40

Managing the project **43**
Creating a project in Wizard mode 43
47
Changing elements within a project 47
Changing a project's data 53
Saving a project 55
Opening a project 55
Network project 56
Creation of a network project 56
"ServerIT107" Project 59
"PC Client" Project " 61
"Network" Project 62
Compilation of the network project 64
Download the network project 65

Project Explorer 69
Operating on elements within the Project Explorer window 71
Setting the panel 72
Software Configuration 77
Variables 88
Pages 103
Popup pages 107
Images 109
Text list 113
114
114
Image list 114
Alarms 115
Recipes 124
Frames 128
Report 131
Pipelines 138
Scripts 140
GlobalScripts 142
Trend Buffers 142
145
DataLog 145
TrendBuffersXY 145
Remote Notifications 147
Weekly Tasks 150
153
Setting the device 153
MemoryAddresses 155
Conversion Tables 156

Properties Editor 159
Properties Editor 159
161
Events Editor 161
Managing a page 167
Predefined graphic elements 172
Simple Figures 172

Value fields 197
"Invert Function" Option 197
"Invert Function" option operation 198
199
"Soglie" (Thresholds) option functioning 199
Objects to which the "Soglie" (Thresholds) functionality can be applied
205
Simple Controls 243
Complex Controls 268
Movement properties of the objects 322
Operations on graphic elements 325

Other anchorable windows 341
POLYMATH Libraries 341
Errors Viewer 348
Compiler Output 349

Compiling, Downloading and Runtime....................................... 351
Project simulation 353
Downloading a project 356
Change Password 364
Download the IT OPERATING SYSTEM image 366
Set up an Ethernet connection 366
Downloading the image of the Operating System for VT CE 368
Establishing an Ethernet connection 368
Source project File and Backup transfer 373
Panel Reset 376

Scripts.. 379
The object ESAUSERMGR 389
The object ESAALARMMGR 389
The object ESARECIPEMGR 392
The object ESARECIPETYP 392
The object ESARECIPEARC 393
The object ESARECIPETRF 395
The object ESAPIPEMGR 397
The object ESATIMER 399

The object ESATRENDMGR 400
The object ESAPAGEMGR 405
The object ESAPAGE 409
The object ESACNTRL 410
object ESAPRN 464
Examples of Script use 467

Tutorial 477
Phase 1 - The Project and Hardware Configuration 477
Phase 2 - Software configuration 479
Phase 3 - Configuration of variables and Memory areas 482
Phase 4 - General configuration of the VT 488
Phase 5 - Defining the alarms 491
Phase 6 - Defining recipe types 493
Phase 7 - Loading Images 494
Phase 8 - Defining text and image lists 496
Phase 9 - Setting Pipelines 498
Phase 10 - Defining a Trend Buffer 499
Phase 11 - Graphic setting, drawing a Frame 500
Phase 12 - Creating pop-up pages 509
Phase 13 - Drawing Full Screen pages 514
Phase 14 - Using complex controls 523
Phase 15 - Defining the Trend graph 528
Phase 16 - Compilation and Download 533

Available functions for Remote connection from the PC 537
Remote Desktop 537
Installation and registration 537
"Remote Desktop" use 538
Enable and disable "FTP" 545

Panels network 549
Example creation of panel’s network 549
Download the network project 553

Appendix A - System Variables 555

Appendix B - Predefined functions 563

Appendix C - Status area 575
VT Status area 575
Keyboard status area 577
Status area of recipes - new style (non-compatible mode) 578
Status area of recipes - old style (compatible mode) 578

Appendix D - Command area 579
Command area for New Style (non compatible) recipes 584
Command area for Old style (compatible) recipes 584

Appendix E - VTxxxW Panels Management.............................. 585
587
587
587
587
Esplora Progetto 587

Appendix F - Update Operating System...................................... 589

[1]

1. Introduction

What is Polymath?

Polymath is the software that ESA Elettronica offers its customers to use to configure all its products that have Windows® CE as their operating system. The principal feature of the application is that it’s so easy to use, thanks to its user-friendly, intuitive interface.

What does Polymath do?

The concept behind Polymath is to be the switching-point between the customer and the terminal; in fact, it is the tool that allows the user to transfer his or her own ideas onto the panel creating projects at different levels of development. It is a universal software, that is, it can be used to program the behavior of ESA Windows® CE terminals, independent of their particular features and technical characteristics.
The work performed by Polymath produces a compiled project containing all the operative details of the package created. Once the project has been compiled without errors, it can be uploaded and installed on the panel, which is now ready to use. Polymath guides the user at every step of the development of the project: from its creation to editing, from compilation to its passage to the terminal.

Which POLYMATH version?

"Basic" only allows to program the VT family of products.

"Advanced" with all functionalities and for all families of products:

- VT text, graphic and touch operator terminals.
- IT terminals based on CE windows operational system
- VT CE terminals with CE open operational system
- Industrial PCs

Pass from the "Basic" mode to "Advanced" with the "Premium" upgrade.

Note: *For a better knowledge of the functions offered by a particular product, please consult the product's technical characteristics on www.esahmi.com*

What's new compared with VTWIN?

The most striking difference between Polymath and its predecessors is undoubtedly its improved, totally overhauled graphic interface. All operations are made simpler and more intuitive and can be achieved with just a few clicks.

There is now the possibility of creating projects by means of a guided procedure (Wizard) that makes it possible to work on a project just a few seconds after starting the software. In addition, easy-to-use operations have been included for managing Recipes and Alarms, automating operations that once could only be done manually.

Further on in this guide there will be a detailed description of all the new operative features and information will be supplied to help you use these in the most efficient way.

The Manual

This manual is designed to be a constant guide for ESA's customers, describing and explaining the different features that the software offers. It is aimed at the average user of ESA products, guiding both first-time users of ESA products and those already familiar with previous versions of the configurator.

The principal concepts and the method of use related to each topic and operative feature will be illustrated using appropriate examples and screenshots.

The information contained in this document is subject to change without prior notice and do not represent any obligation on the part of ESA elettronica S.P.A.

All products are trade names registered by their respective owners.

Conventions used in the Manual

To make it easier to consult the manual and make the topics dealt with simpler to understand, we will use symbols that it would be useful to learn from the outset.

The table below lists all the symbols that are used in the following chapters of this manual:

Tabella 1: List of conventions used

Symbol	Meaning
File->New	Indicates a navigational path; in this case it means the user should click consecutively on the File and then New buttons
💡	Indicates that there is a note; notes are often inserted to provide suggestions or clarify common doubt
⚠	Indicates particularly important points to be read with care to avoid falling into difficult situations
🧩	Indicates that there is a guide dedicated to explaining in detail how a particular operation should be carried out
☑	Indicates that within the description there are key ideas - ideal for rapid consultation of the guide in that they accompany the essential notions

ESA Elettronica's Customer Care service

In the event of any doubts about the use of POLYMATH or other ESA products, contact ESA Elettronica's Customer Care service (open Monday to Friday from 8.30 to 12.30 and 14.00 to 18.00).
Customer Care telephone number: 0039/031/757400
Fax: 0039/031/751777
E-mail: customer.care@esahmi.com

⚠ ***Important:*** *it is always a good idea to annotate the currently installed version of POLYMATH and keep it to hand every time you contact ESA's Customer Care service. The version of the software is shown in the main menu by clicking on Help->Information*

2. Installation

This chapter supplies information needed to be able to undertake the first steps towards using POLYMATH: ’installation. We set out the requirements a machine must have for the application to function correctly as well as the crucial steps that make up the installation phase.

Note: *POLYMATH is a programming utility for ESA panels ESA that use the Windows® CE operating system, but this configuration software can be installed on PCs using the Windows® 2000, XP or Vista operating system.*

Minimum requirements

Below are set out the minimum requirements necessary for using POLYMATH on one’s machine:

Tabella 1: Minimum requirements

Type	Requirement
Operating system	***Windows® 2000 with Service Pack 4*** ***Windows® XP with Service Pack 2*** ***Windows® Vista***
RAM	***256 MB RAM***
Processor	***Pentium IV or equivalent***
Screen resolution	***1024*768***
Space on Hard Disk	***750 MB***

Recommended requirements

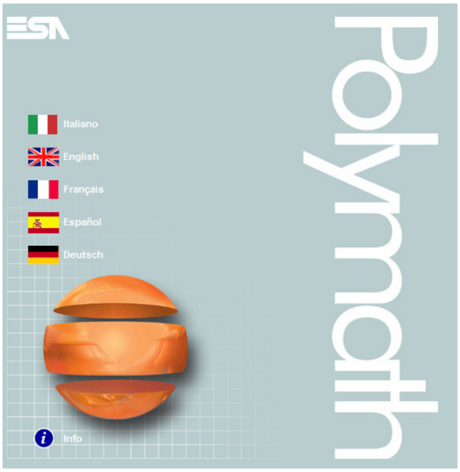
Below are set out the recommended requirements for being able to run POLYMATH better on one's machine:

Tabella 2: Recommended Requirements

Type	Requirement
Operating system	*Windows® 2000 with Service Pack 4 or better* *Windows® XP with Service Pack 2 or better*
RAM	*512 MB RAM or better*
Processor	*Pentium IV or better*
Screen resolution	*1280*1024 WIDESCREEN*
Space on Hard Disk	*750 MB*

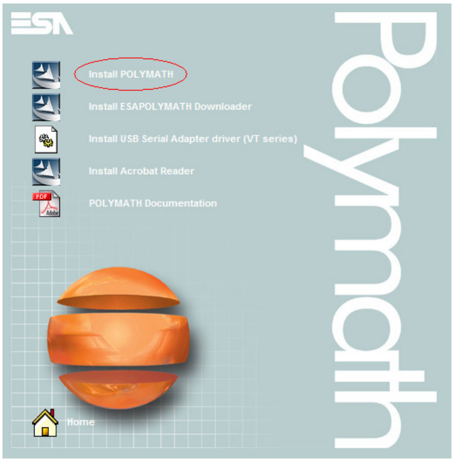
Installing POLYMATH

Once the presence of the minimum requisites have been checked on your mac-hine, it is possible to start the installation of POLYMATH.
Close or end any application active on the computer.
Introduce the POLYMATH program CD-ROM.
The following window is presented automatically :



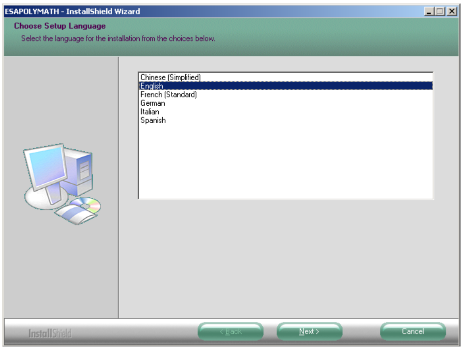
Select the desired language.

Start "Installazione POLYMATH" (POLYMATH Installation).


ESA
Polymath
Install POLYMATH
Install ESAPOLYMATH Downloader
Install USB Serial Adapter driver (VT series)
Install Acrobat Reader
POLYMATH Documentation
Home

The following window can also be activated by exploring the content of the CD-ROM and carrying out the \disk1\setup.exe. file.

Select the desired language.

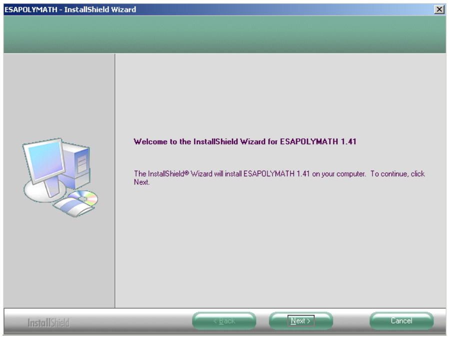


Choose Setup Language

Select the language for the installation from the choices below.

Chinese (Simplified)
English
French (Standard)
German
Italian
Spanish

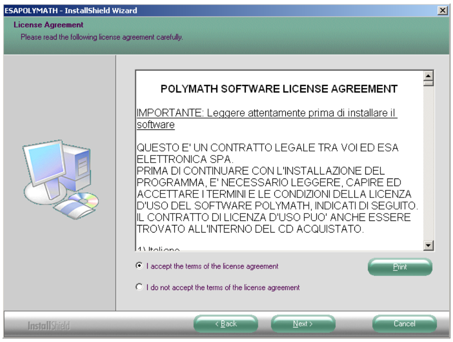
Select "Avanti" (Next).



Welcome to the InstallShield Wizard for ESAPOLYMATH 1.41

The InstallShield® Wizard will install ESAPOLYMATH 1.41 on your computer. To continue, click Next.

Read and accept the license terms and select "Avanti" (Next).



License Agreement

Please read the following license agreement carefully.

POLYMATH SOFTWARE LICENSE AGREEMENT

IMPORTANTE: Leggere attentamente prima di installare il software

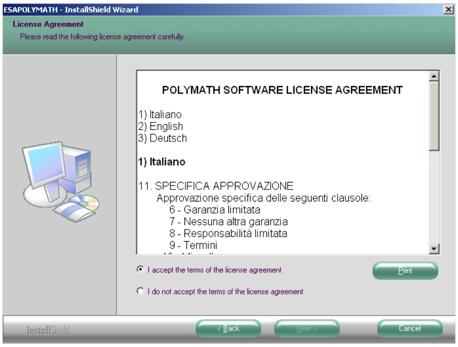
QUESTO E' UN CONTRATTO LEGALE TRA VOI ED ESA ELETTRONICA SPA.
PRIMA DI CONTINUARE CON L'INSTALLAZIONE DEL PROGRAMMA, E' NECESSARIO LEGGERE, CAPIRE ED ACCETTARE I TERMINI E LE CONDIZIONI DELLA LICENZA D'USO DEL SOFTWARE POLYMATH, INDICATI DI SEGUITO.
IL CONTRATTO DI LICENZA D'USO PUO' ANCHE ESSERE TROVATO ALL'INTERNO DEL CD ACQUISTATO.

1) Italiano

● I accept the terms of the license agreement

○ I do not accept the terms of the license agreement

Read and accept the license terms and select "Avanti" (Next).



ESAPOLYMATH - InstallShield Wizard

License Agreement

Please read the following license agreement carefully.

POLYMATH SOFTWARE LICENSE AGREEMENT

1) Italiano
2) English
3) Deutsch

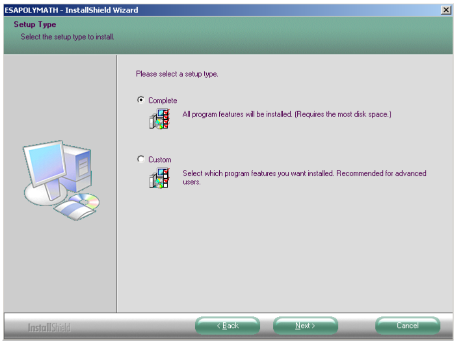
1) Italiano

11. SPECIFICA APPROVAZIONE
Approvazione specifica delle seguenti clausole:
6 - Garanzia limitata
7 - Nessuna altra garanzia
8 - Responsabilità limitata
9 - Termini

● I accept the terms of the license agreement

○ I do not accept the terms of the license agreement

Select the desired option.



ESAPOLYMATH - InstallShield Wizard

Setup Type

Select the setup type to install.

Please select a setup type.

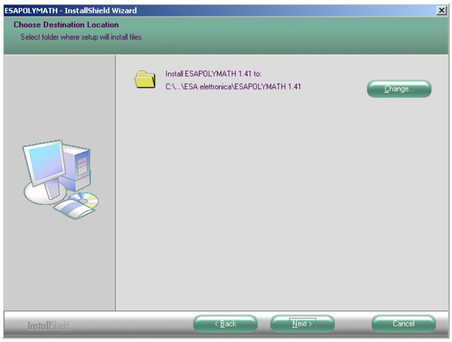
● Complete

All program features will be installed. (Requires the most disk space.)

○ Custom

Select which program features you want installed. Recommended for advanced users.

Select "Avanti" (Next) in order not to modify the default folder of the POLYMATH program (C:\Program Files\ESA elettronica\ESAPOLYMATH 1.xx) or "Cambia" (Change) to modify the pathway.



ESAPOLYMATH - InstallShield Wizard

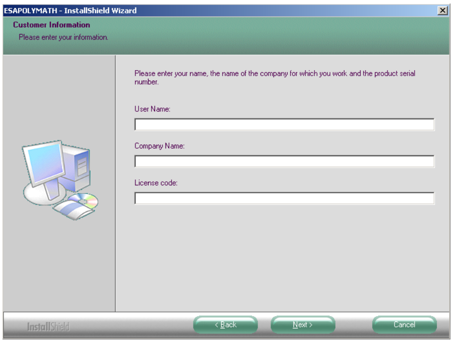
Choose Destination Location

Select folder where setup will install files.

Install ESAPOLYMATH 1.41 to:
C:\...\ESA elettronica\ESAPOLYMATH 1.41

Note*: as POLYMATH is a software in continuous development, with frequent issues of new versions, it is useful specify directories different to the default ones (e.g. ESAPOLYMATH_1.XX) in order to allow different versions to coexist on the same machine, if this necessity should arise.*

Introduce the following information regarding the user.



Customer Information

Please enter your information.

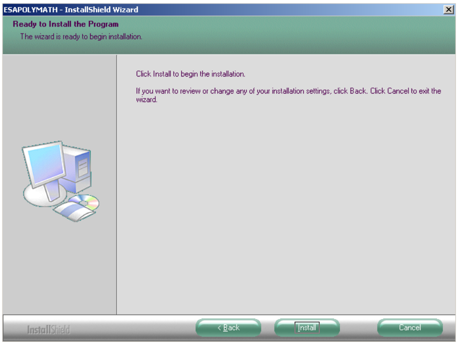
Please enter your name, the name of the company for which you work and the product serial number.

User Name:

Company Name:

License code:

Select "Installa" (Install).



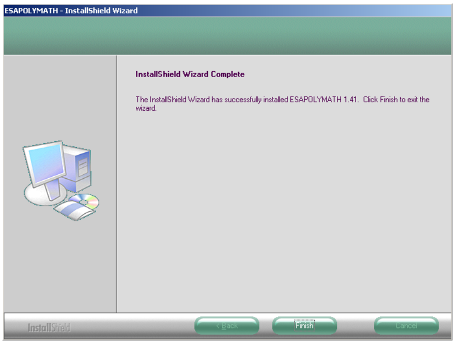
Ready to Install the Program

The wizard is ready to begin installation.

Click Install to begin the installation.

If you want to review or change any of your installation settings, click Back. Click Cancel to exit the wizard.

Select "Fine" (End).



ESAPOLYMATH - InstallShield Wizard

InstallShield Wizard Complete

The InstallShield Wizard has successfully installed ESAPOLYMATH 1.41. Click Finish to exit the wizard.

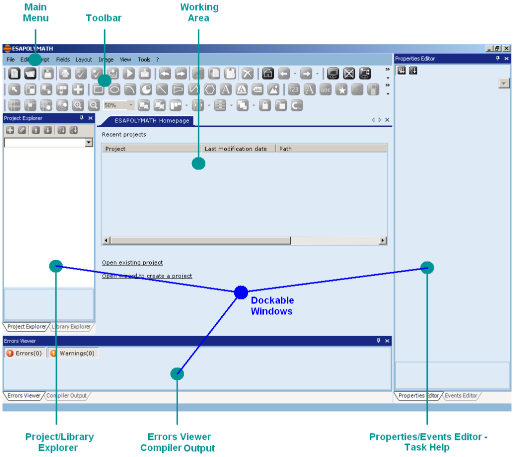
The POLYMATH installation procedure has ended.

3. Layout of menus

Before we can confidently operate the numerous features offered by POLYMATH we need to familiarise ourselves with the work environment and its various menus.
The layout of the application can be divided into the following areas:

- main menu
- toolbars
- work area
- a series of anchorable windows

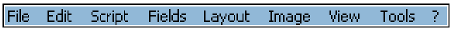
This chapter offers guidelines for making general software settings and will pay particular attention to the main menu and the toolbar which are the basic instruments for carrying out any operations within POLYMATH. We will also refer to the various anchorable windows which will be dealt with in greater detail in the course of the following chapters.



The functions offered by the toolbar can all be accessed via the main menu.

Main menu

The main menu is the tool that permits POLYMATH's main project and settings operations to be performed.



It is located in the top part of the program window and is a fixed element that cannot be repositioned within the framework of the page. There are also various scrollable submenus each offering different functions as set out in the paragraphs that follow.

File menu

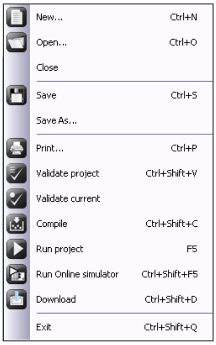


Table 1: Functions in the File menu

Icon	Path Menu	Description of function
📄	***File -> New***	Creates a new Wizard project (see chap. 4, “Managing the project” page 43)
📂	***File -> Open***	Opens an existing project
NA[1]	***File -> Close***	Closes the project
💾	***File -> Save***	Saves the project
NA[1]	***File -> Save as...***	Saves the project with a different name/path
🖨	***File -> Print***	Prints the project
☑	***File -> Validate project***	Validates all the project (see chap. 8, “Validation” page 351)

Table 1: Functions in the File menu

Icon	Path Menu	Description of function
✓	***File -> Validate current***	Validates the element currently selected
	File -> Compile	Compiles the project (see chap. 8, “Compiling, Downloading and Runtime” page 351)
▶	***File -> Run project***	Opens the project simulator (see chap. 8, “Compiling, Downloading and Runtime” page 351)
	File -> Perform online simulator	Open the project online simulator (see chap. 8, “Compiling, Downloading and Runtime” page 351)
⬇	***File -> Download***	Downloads the project onto the panel (see chap. 8, “Compiling, Downloading and Runtime” page 351)
NA[1]	***File -> Exit***	Exits from POLYMATH

[1] Icon Not Available

Edit menu

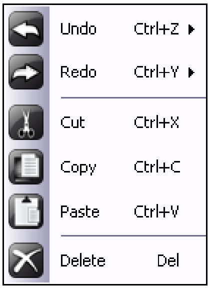


Table 2: Functions of the Edit menu

Icon	Path Menu	Description of function
↶	***Modify -> Annul***	Annuls the previous operation
↷	***Modify -> Repeat***	Repeats the following operation

Table 2: Functions of the Edit menu

Icon	Path Menu	Description of function
✂	***Edit -> Cut***	Cuts the object selected
📄	***Edit -> Copy***	Copies the object selected
📋	***Edit -> Paste***	Pastes the object that has been cut or copied
✕	***Edit -> Cancel***	Deletes the object selected

Script Menu

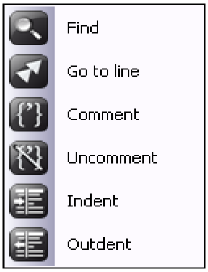


Table 3: Script menu functions

Icon	Path Menu	Description of function
🔍	***Script -> Find***	Finds a specific string in the script
➤	***Script -> Go to line***	Directs to a specific page in the script
{'}	***Script -> Comment***	Allows to insert a comment in the script
✕	***Script -> Uncomment***	Eliminates a comment from the script
☰	***Script -> Increase re-entry***	Increases the re-entry of the text in the script
☰	***Script -> Reduce re-entry***	Reduces the re-entry of the text in the script

Fields menu

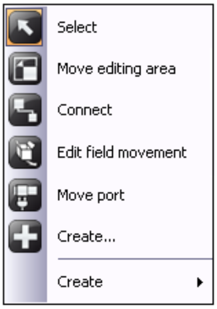
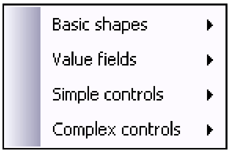


Table 4: Functions of the Fields menu

Icon	Path Menu	Description of function
↖	***Fields -> Select***	Selects the object clicked on after pressing
	Fields -> Move Editing area	Moves the Editing area selected (e.g. Popup Page)
	Fields -> Connect	Enables connection of devices and terminals (see chap. 4, “Managing the project” page 43)
	Fields -> Edits field movement	Displays the implemented movement of the object in the "Editor Proprietà" (Properties Editor)
	Fields -> Move Port	Moves a connection port
✚	***Fields -> Create***	Adds a device or a terminal to the project

The Create submenu can be reached via the Fields menu and this submenu can be used to add a large number of elements to the page (Fields -> Create).



The elements that can be added are grouped under the following headings:

- Simple figures
- Value fields
- Simple Controls
- Complex controls

The tables below give a description of the commands that can be launched from this submenu. Refer to the appropriate chapter for the characteristics peculiar to the elements that have been added.

Submenu: Simple figures

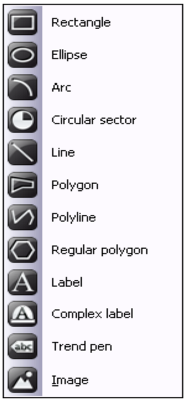


Table 5: Functions of the submenu: Fields -> Create -> Simple figures

Icon	Path Menu	Description of function
▭	***Simple figures -> Rectangle***	Adds a rectangle to the page (see chap. 6, “Simple Figures” page 172)
⬭	***Simple figures -> Ellipse***	Adds an ellipse to the page
◝	***Simple figures -> Arc***	Adds an arc to the page
◔	***Simple figures -> Circular sector***	Adds a circular sector to the page
╲	***Simple figures -> Line***	Adds a line to the page
⏢	***Simple figures -> Polygon***	Adds a polygon to the page
∿	***Simple figures -> Broken line***	Adds a broken line to the page
⬡	***Simple figures -> Regular polygon***	Adds a regular polygon to the page
A	***Simple figures -> Label***	Adds a label to the page
A	***Simple figures -> Complex label***	Adds a complex label to the page
abc	***Simple figures -> Trend pen***	Adds a trend pen to the page indicating the current value of the buffer
🖼	***Simple figures -> image***	Adds an image to the page

Submenu: Value fields

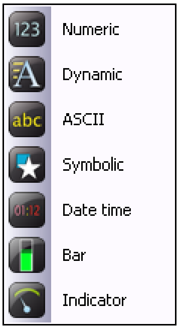


Table 6: Functions of the submenu: Fields -> Create -> Value fields

Icon	Path Menu	Description of function
123	***Value fields -> Numerical***	Adds a numerical field to the page (see chap. 6, “Value fields” page 197)
A	***Value fields -> Dynamic***	Adds a dynamic text to the page
abc	***Value fields -> ASCII***	Adds an ASCII field to the page
★	***Value fields -> Symbolic***	Adds a symbolic field to the page
01:12	***Value fields -> Date Time***	Adds a field relating to the date and time to the page
▮	***Value fields -> Bar***	Adds a bar to the page
◠	***Value fields -> Indicator***	Adds an indicator to the page

Submenu: Simple Controls

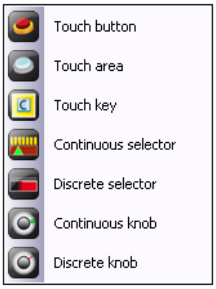


Table 7: Functions of the submenu: Fields -> Create -> Simple Controls

Icon	Path Menu	Description of function
●	***Simple Controls -> Touch Button***	Adds a touch button to the page (see chap. 6, “Simple Controls” page 243)
○	***Simple Controls -> Touch Area***	Adds a touch area to the page
C	***Simple controls -> Touch Keyboard Button***	Determines the keys and is used only during the configuration of the run time keyboard
▨	***Simple Controls -> Slide Potentiometer***	Adds a slide potentiometer (with no predefined values) to the page
▬	***Simple Controls -> Slide Selector***	Adds a slide selector (with predefined values) to the page
◎	***Simple Controls -> Potentiometer Knob***	Adds a knob potentiometer (without predefined values) to the page
◎	***Simple Controls -> Selector Knob***	Adds a selector knob (with predefined values) to the page

Submenu: Complex controls

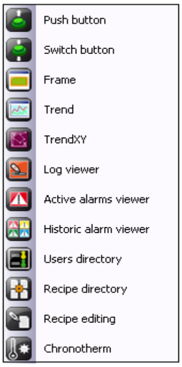


Table 8: Functions of the submenu: Fields -> Create -> Complex controls

Icon	Path Menu	Description of function
	Complex controls -> One-touch button	Adds a one-touch push-button to the page (see chap. 6, “Complex Controls” page 268)
	Complex controls -> Double-touch button	Adds a double-touch button to the page
	Complex controls -> Frame	Adds a frame to the page
	Complex controls -> Trend	Adds a trend to the page
	Complex controls -> TrendXY	Inserts a trendXY in the page
	Complex controls -> Logged on users displayed	Displays the users logged on and allows the password to be changed
	Complex controls -> Active alarm table	Adds a table of active alarms to the page
	Complex controls -> Alarm history table	Adds an alarm history table to the page

Table 8: Functions of the submenu: Fields -> Create -> Complex controls

Icon	Path Menu	Description of function
	Complex controls -> User list	Adds a table with a list of users to the page
	Complex controls -> Recipe list	Adds a table with a list of recipes to the page
	Complex controls -> Recipe editor	Adds a table with a recipe editor to the page
	Complex controls -> Chronothermostat	Inserts a chronothermostat in the page

Menu: Layout

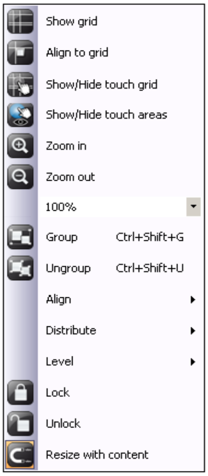


Table 9: Functions of the menu: Layout

Icon	Path Menu	Description of function
	Layout -> Show grid	Shows the grid in a page or in a Hardware configuration (see chap. 6, “Page properties” page 170)
	Layout -> Align grid	Aligns the selected element to the grid
	Layout -> Show/ Hide Touch Grid	Displays / hides the cells to b selected by the Grill on the Touch screen
	Layout -> Show/ Hide touch-sensitive areas	Displays / hides the pixels of the Area on the Touch screen
⊕	***Layout -> Enlarge***	Enlarges the page display
⊖	***Layout -> Reduce***	Reduces the page display
100%	***Layout -> Zoom***	Makes it possible to indicate the display percentage for the page
	Layout -> Group	Group two or more elements in the current selection (see chap. 6, “Grouping of two or more graphic elements” page 325)
	Layout -> Separate	Separates the elements of a group
🔒	***Layout -> Block***	Blocks the objects / pages
🔓	***Layout -> Un block***	Un blocks the objects / pages
	Layout -> Re-dimension with the control	Re-dimensions the elements collected, maintaining the aspect.

Using the Layout menu you can also access all the functions for aligning and positioning the elements within the pages. This is done using the submenus: Align, Arrange and Level that are illustrated below.

Submenu: Align

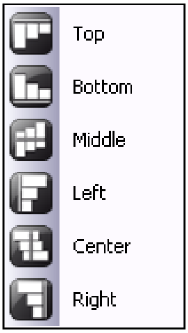


Table 10: Functions of the submenu: Layout -> Align

Icon	Path Menu	Description of function
	Align -> Top	Aligns the object in the selection with the top (see chap. 6, "Alignment of objects" page 330)
	Align -> Bottom	Aligns the object in the selection with the bottom
	Align -> Middle	Aligns the object in the selection with the middle
	Align -> Left	Aligns the object in the selection with the left
	Align -> Centre	Aligns the object in the selection with the centre
	Align -> Right	Aligns the object in the selection with the right

Submenu: Arrange

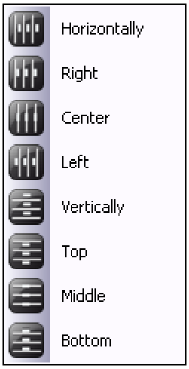


Table 11: Functions of the submenu: Layout -> Arrange

Icon	Path Menu	Description of function
▥	***Arrange -> Horizontally***	Arranges the object in the selection horizontally (see chap. 6, “Arrangement of objects” page 334)
▥	***Arrange -> Right***	Arranges the object in the selection to the right
▥	***Arrange -> Centre***	Arranges the object in the selection to the centre
▥	***Arrange -> Left***	Arranges the object in the selection to the left
☰	***Arrange -> Vertically***	Arranges the object in the selection vertically
☰	***Arrange -> Top***	Arranges the object in the selection to the top
☰	***Arrange -> Middle***	Arranges the object in the selection to the middle
☰	***Arrange -> Bottom***	Arranges the object in the selection to the bottom

Submenu: Level

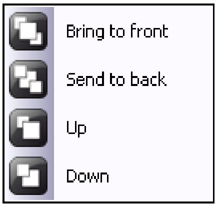


Table 12: Functions of the submenu: Layout -> Level

Icon	Path Menu	Description of function
⧉	***Level -> Foreground***	Places the object selected into the foreground (see chap. 6, “Depth order of objects” page 328)
⧉	***Level -> Background***	Places the object selected onto the background
⧉	***Level -> Up***	Raises the object selected by a level
⧉	***Level -> Down***	Lowers the object selected by a level

Menu: Image

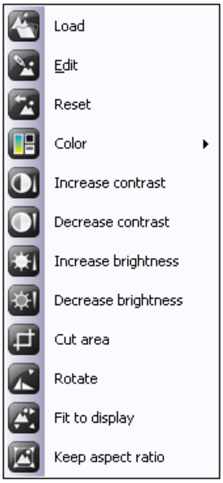


Table 13: Functions of the menu: Image

Icon	Path Menu	Description of function
🖼	***Image -> Load***	Load the current image (see chap. 5, “Operations performable on an image” page 111)
✎	***Image -> Edit***	Allows the image to be edited
↩	***Image -> Remove***	Remove the image loaded
▤	***Image -> Colour***	Makes it possible to choose the type of colouring between: Automatic, Tones of grey, White and Black
◐	***Image -> Increase contrast***	Increases the contrast of the image selected
◑	***Image -> Decrease contrast***	Decreases the contrast of the image selected
☀	***Image -> Increase brightness***	Increases the brightness of the image selected

Table 13: Functions of the menu: Image

Icon	Path Menu	Description of function
	Image -> Decrease brightness	Decreases the brightness of the image selected
	Image -> Cut area	Cuts the area selected
	Image -> Rotate	Rotates the image selected
	Image -> Adapt to screen	Adapts the selection to the display
	Image -> Maintain proportions	Maintains the proportions while the image size is changed

Menu: Display

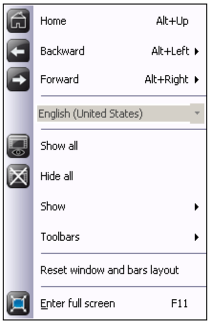


Table 14: Functions of the menu: Display

Icon	Path Menu	Description of function
	Display -> First page	Moves to POLYMATH Home Page
←	***Display -> Last***	Moves to last work page displayed
→	***Display -> Forward***	Moves to next work page displayed
English (United States)	***Display -> Project language***	Makes it possible to change the current project language

Table 14: Functions of the menu: Display

Icon	Path Menu	Description of function
▣	***Display -> Show all***	Shows all the anchorable windows (see chap. 3, “Anchorable windows” page 40)
☒	***Display -> Hide all***	Hides all the anchorable windows
NA[1]	***Display -> Show***	Allows to access the anchorable windows sub-menu
NA[1]	***Display -> Tools bar***	Allows to access the tools bar submenu
NA[1]	***Display -> Restores windows and bars position***	Allows to restore the POLYMATH windows and bars with the default position
⛶	***Display -> Full screen***	Allows to display the work window in "Schermo Intero" (Full Screen) mode

[1] Icona Non Disponibile.

Submenu: Show

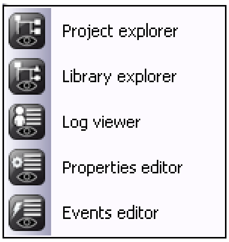


Table 15: Functions of the submenu: Display -> Show

Icon	Path Menu	Description of function
	Show -> Explore project	Shows the Explore Project window (see chap. 3, “Anchorable windows” page 40)
	Show -> Explore Library	Shows the Explore Library window
	Show -> Log List	Shows the Log List window
	Show -> Property Editor	Shows the Property Editor window
	Show -> Events Editor	Shows the Events Editor window

Submenu: Toolbar

This submenu lists the twelve groups of icons making up the toolbar. Using this menu the user can proceed to reintroduce into the application groups of icons that have been closed and that no longer appear in the POLYMATH screen. For further information about the way the toolbar works, please consult the next paragraph.

Menu: Tools

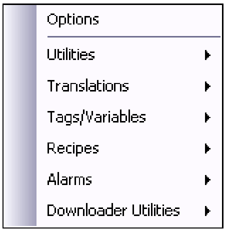


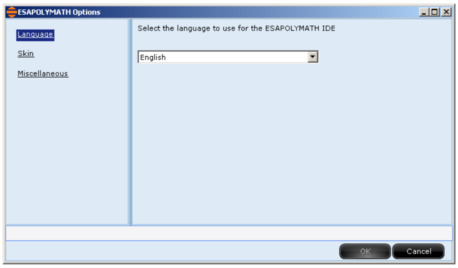
Table 16: Functions of the menu: Tools

Icon	Path Menu	Description of function
ND[1]	***Tools -> Options***	Makes it possible to configure the Options of POLYMATH
ND1	***Tools -> Utility***	Accesses POLYMATH utilities
ND[1]	***Tools -> Translations***	Manages the Translation of the project
ND[1]	***Tools -> Variables***	Manages the Tags/Variables of the project
ND[1]	***Tools -> Recipes***	Manages the project recipes
ND[1]	***Tools -> Alarms***	Manages the project alarms
ND[1]	***Tools -> Downloader Utility***	Accesses the Downloader functions

[1] Icon Not Available

Options Sub-menu

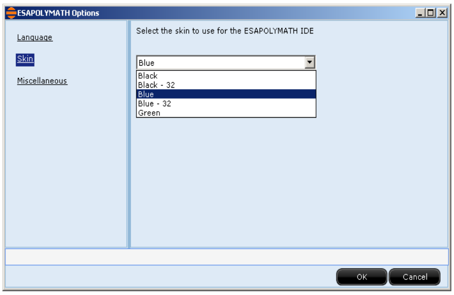
Click on the option Tools -> Options to access the mask for configuring the Options of POLYMATH.



Select the language to use for the ESAPOLYMATH IDE

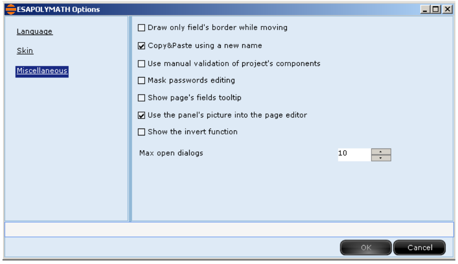
English

Use the Language menu to choose the language of the POLYMATH application. Once the language has been selected the application will need to be restarted to apply the changes.



Select the skin to use for the ESAPOLYMATH IDE

From the "Skin" menu, it is possible to select the skin to use with the POLYMATH interface.



☐ Draw only field's border while moving

☑ Copy&Paste using a new name

☐ Use manual validation of project's components

☐ Mask passwords editing

☐ Show page's fields tooltip

☑ Use the panel's picture into the page editor

☐ Show the invert function

Max open dialogs 10

Use the Various menu to proceed to configure the general Options of the application. The user may decide to view all the objects during the move or only their outline, to automatically provide a new name if using the cut/paste function , whether to validate the project manually (File -> Validate project) or in ‛real time’ automatically (see chap. 8, “Validation” page 351), to visualise the edit password screen, whether or not to view the ESA terminal frame on the page editor, whether to activate the "Invert" option or not and to set the maximum number of windows open at the same time in the POLYMATH Work area.

Note: *The options Manual validation and viewing only the outline while the objects are being dragged are advised for configuring particularly slow performing machines.*

Utility

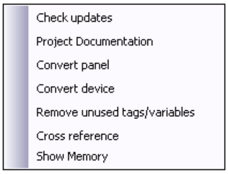


Table 17: Utility functions menu

Icon	Menu path	Function description
ND[1]	***Utility -> Updates control***	Allows to check the presence of new software issues of the POLYMATH program
ND[1]	***Utility -> Project documents***	Allows a document to be created with the specifics of the project
ND[1]	***Utility -> Panel converts***	Allows the conversion of a panel
ND[1]	***Utility -> Device converts***	Allows the conversion of a device
ND[1]	***Utility -> Remove Tags/Variables not used***	Checks if Tags/Variables not used are present in the project
ND[1]	***Utility -> Crossed reference***	Finds all components used inside the project. Configured only components , are not included in the research
ND[1]	***Utility ->Show Memory***	Displays the Tags occupied in the device memory

1.Icon Not Available

Translations Sub-menu

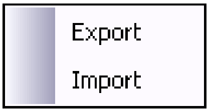


Tabella 18: Translations menu functions

Icon	Menu path	Function description
ND[1]	***Translations -> Export***	Exports Translations
ND[1]	***Translations -> Import***	Imports Translations

[1] Icon Not Available

Translations are converted in CSV. format easily transferable and convertible from each software.

Tags/Variables Sub-menu

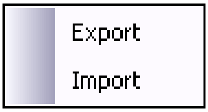


Tabella 19: Tags/Variables menu functions

Icon	Menu path	Function description
ND[1]	***Tags/Variables -> Export***	Exports Tags/Variables
ND[1]	***Tags/Variables -> Import***	Imports Tags/Variables

[1] Icon Not Available

Recipes Sub-menu

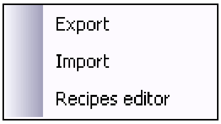


Tabella 20: Recipe menu functions

Icon	Menu path	Function description
ND[1]	***Recipes -> Export***	Exports Recipes
ND[1]	***Recipes -> Import***	Imports Recipes
ND[1]	***Recipes -> Recipe Editor***	Allows to manage the recipes

[1] Icon Not Available

Alarms Sub-menu

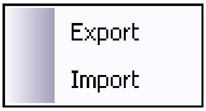


Tabella 21: Alarms menu functions

Icon	Menu path	Function description
ND[1]	Alarms -> Export	Exports Alarms
ND[1]	Alarms -> Import	Imports Alarms

[1] Icon Not Available

Downloader Utility Sub-menu

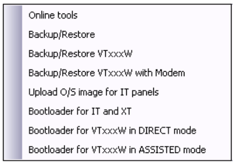

Online tools
Backup/Restore
Backup/Restore VTxxxW
Backup/Restore VTxxxW with Modem
Upload O/S image for IT panels
Bootloader for IT and XT
Bootloader for VTxxxW in DIRECT mode
Bootloader for VTxxxW in ASSISTED mode

Tabella 22: Utility downloader functions menu

Icon	Menu path	Function description
ND[1]	***Utility downloader ->Online Tools***	After having connected to the ESA panel, it allows to carry out the following operations: To transfer the project, to explore the panel, to compare the memory used with that which can be used, to compare the files which make up the project to be transferred with those already present on the ESA terminal, to fill in the project.
ND[1]	***Utility downloader ->Backup/Restore***	Perform backup or restore the project for CE / IT products
ND[1]	***Utility downloader ->Backup/Restore VTxxx***	Perform backup or restore the project for VTxxx products
ND[1]	***Utility downloader ->Backup/Restore VTxxxW with modem***	Perform backup or restore the project for VTxxx products by means of the modem
ND[1]	***Sends O/S image for the Windows IT panels***	Updates the image of the operating system for IT panels only
ND[1]	***Utility downloader ->Update Boot Windows CE for IT and XT***	Updates the boot Windows CE for IT and XT panels

Tabella 22: Utility downloader functions menu

Icon	Menu path	Function description
ND[1]	***Utility downloader ->Boot loader directly for WTxxx***	Updates the boot of the VTxxx terminal without the help of help messages
ND[1]	***Utility downloader ->Boot loader for WTxxx in assisted mode***	Updates the boot of the VTxxx terminal with the help of help messages

? Menu

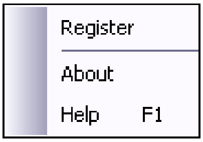


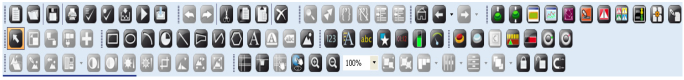
Table 23: ? menu functions

Icon	Menu path	Function description
ND[1]	***? -> Register***	Allows the user to register the installed polymath product
ND[1]	***? -> Information***	Allows the information regarding the version of the program to be seen
ND[1]	***? -> Help***	Allows to access the POLYMATH guide

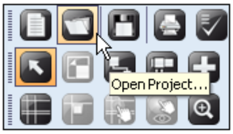
[1] Icon Not Available

The Toolbar

The Toolbar consists of buttons allowing the user to access all POLYMATH operations.



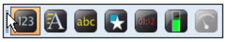
When the mouse is placed on one of the icons, its meaning is displayed, see below:



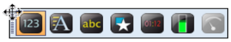
__Note:__ The Toolbar offers a shortcut to the same Functions that you can access from the main menu. To find out what a given icon means, consult the Table of Functions in the main menu (see chap. 3, "Main menu" page 13).

Editing the Toolbar

The Toolbar is organized into groups of icons, each of which can be managed individually.
To move or delete a group of objects, just drag up from the bar towards any area of the application. To start the drag, click on the left edge of the group.



Once you have clicked, the mouse pointer will change into the dragging cursor typical of Windows. It is now possible to insert the group wherever you want.



Release the mouse key to apply the move. The group can be left in any position on the screen or closed by clicking on the related ' ☒ '. Closed groups can be reinserted into the toolbar by clicking on the corresponding name in the main menu

(Display->toolbar). The changes to the layout of the toolbar are saved for the next time POLYMATH is used.

Anchorable windows

Besides the menu and the icons, the other fundamental component of POLYMATH is the Anchorable window.
Anchorable windows are:

- Project explorer (see chap. 5, “Project Explorer” page 69)
- Properties editor (see chap. 6, “Properties Editor” page 159)
- Events editor (see chap. 6, “Properties Editor” page 159)
- Library explorer (see chap. 7, “POLYMATH Libraries” page 341)
- “Errors Viewer” (see chap. 7, “Errors Viewer” page 348)
- “Compiler Output” (see chap. 7, “Compiler Output” page 349)

The Anchorable windows are described in detail in the following chapters together with their respective function. In this section we will simply explain how Anchorable windows are positioned and managed.

Displaying Anchorable windows

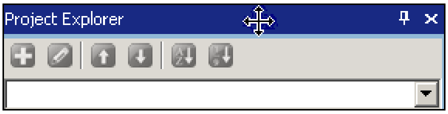
When the program is started up, all the Anchorable windows are displayed in the layout of the application, though the software layout can be changed to suit the user.
Each of these windows can be closed at any moment using the ☒ button and hidden using the 📌 button. Hidden windows remain on the sides of the screen in the form of clickable folders. To make a window appear again in its fixed position, click on the icon ⊣.

Note: *The Hide function is recommended where the resolution of the screen is poor and space needs to be reserved for the Work area.*

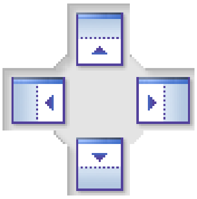
Once Anchorable windows have been closed, they can be re-introduced by clicking on the respective icon in the Tools menu or using the submenu: Display -> Show. Alternatively they can be re-introduced using the menu that appears after clicking with the right-hand key inside the toolbar (first activate the corresponding check).
By clicking on the icon 👁 (Display -> Show All) all the Anchorable windows are re-introduced, while the icon ☒ (Display -> Hide all) deletes them all without distinction.

Moving Anchorable windows

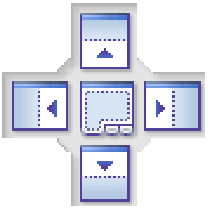
Anchorable windows can be moved within the POLYMATH as the user thinks fit. To move an Anchorable window, select it by clicking on the title bar of the window in question. See below:



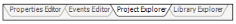
POLYMATH is organized into four virtual areas inside of which a window can be anchored. These virtual areas are situated respectively to the left, to the right, below and above the Work area.



To select which of the four areas to move the window to, use the mouse to place the grab (see picture) on one of the four arrows of the directional pointer (see picture) in the middle of the screen. When the mouse key is released, the window assumes its new position immediately, keeping it until the next operation. Each time the mouse reaches an area of the directional pointer, the corresponding destination area is highlighted.



If the window is dragged within another Anchorable window, a new directional pointer containing a fifth, central button appears. When the grab is dragged onto one of the arrows of the directional pointer and the mouse key is released, the window is simply set next to the existing one in the appropriate direction. While if it is dragged onto the “fifth” button of the pointer, the window is incorporated as a clickable folder as indicated in the figure below :



The windows that are in the form of a clickable folder can be moved by merely dragging the folder to a new position. The changes made to the program layout are saved for all future use of the software.

4. Managing the project

The user can completely program the behavior of the terminal by using POLYMATH which will produce at the end a project file.
The user can create a project file, edit it as he or she pleases (using the functions we will describe later on), save it and later reopen it for any further editing.
The aim of this chapter is precisely to furnish the information needed to create and manage the POLYMATH project files correctly.
The first step using POLYMATH is to create a brand-new project. The user is offered two ways of creating a new project: one guided (the Wizard) and one completely manual.

Creating a project in Wizard mode

Opening the Wizard

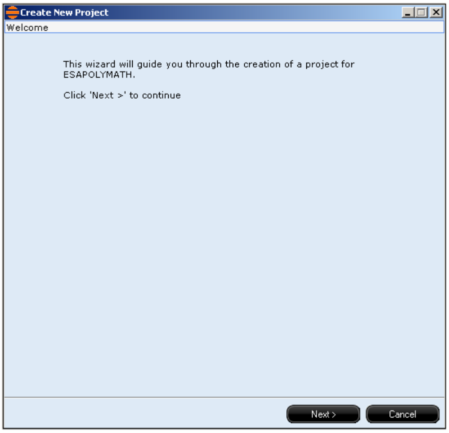
Wizard mode will guide you in creating your projects and in organizing the various hardware components.
There are three ways of activating the Wizard :

- Click on 📄
- File->New from the main menu
- Click on 🏠 to enter the Home Page of the program and Click on 'Open Wizard to create a project' to start up the guided project creation.

Using the Wizard

To create a project in "Wizard" mode 5 operations need to be carried out : choosing the "Type of project", choosing the "Panel", choosing the "Device", "Project information" and "Confirmation of the choices". While in any given window of the Wizard it will be possible to review the preceding step simply by clicking on the 'Preceding' key.

As soon as the guided project creation starts up, the Wizard welcomes you.



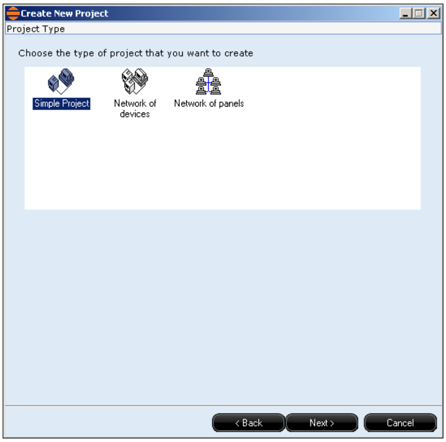
Create New Project

Welcome

This wizard will guide you through the creation of a project for ESAPOLYMATH.

Click 'Next >' to continue

To start creating the project just click on the ‘Forward’ button as highlighted in the figure.



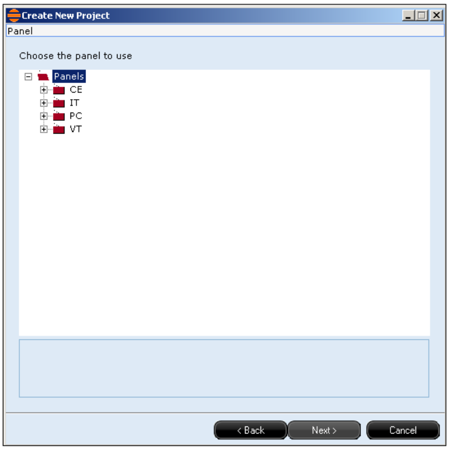
Create New Project

Project Type

Choose the type of project that you want to create

The first choice to make relates to the type of project to be created; the options available are Simple Project, Network of Devices or Network of Panels.

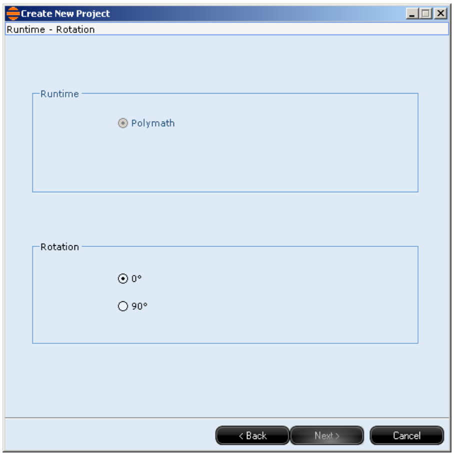
Once the choice has been made, click on ‘Continue’ to proceed.



Choose the panel to use

- Panels
  - CE
  - IT
  - PC
  - VT

Now state which ESA panel is to be used; once this choice has been made, click on ‘Continue’ to proceed.



Runtime

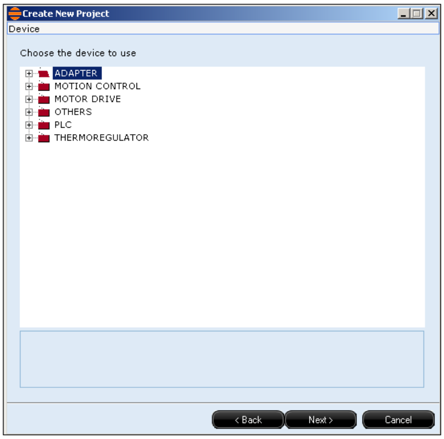
● Polymath

Rotation

● 0°

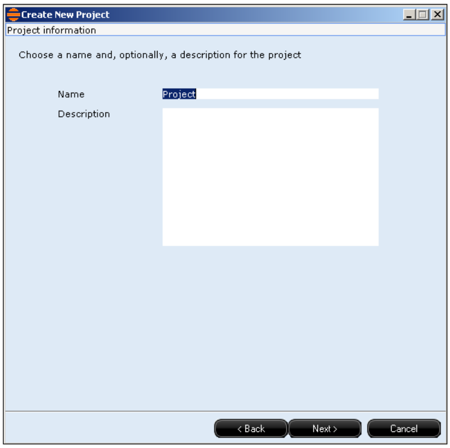
○ 90°

If a panel in the "IT" family is selected (excluding panels in the "IT105K" family), it is possible to have both the horizontal (0°) or vertical (90°) display options.



The device to be connected must be selected; different categories are supplied for the selection. For each category, the devices are divided by Manufacturer's. Once the selection has been made, click on "Avanti" (Next) to continue.

Note: *the categories proposed in the window differ according to the type of panel previously selected.*

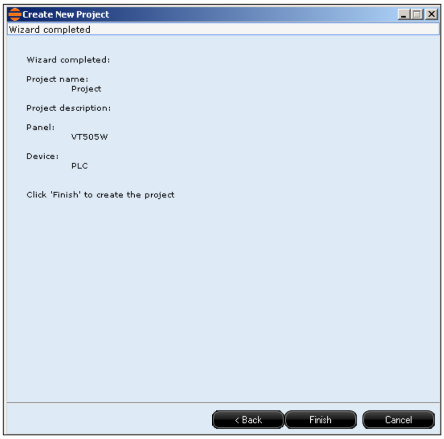


Choose a name and, optionally, a description for the project

Name: Project

Description

The next window contains a request to enter a name that is valid for your project and a description of the latter (this is optional, it serves only to identify that project within POLYMATH). Once this data has been entered, click on ‘Continue’ to proceed.



Wizard completed

Wizard completed:

Project name:
Project

Project description:

Panel:
VT505W

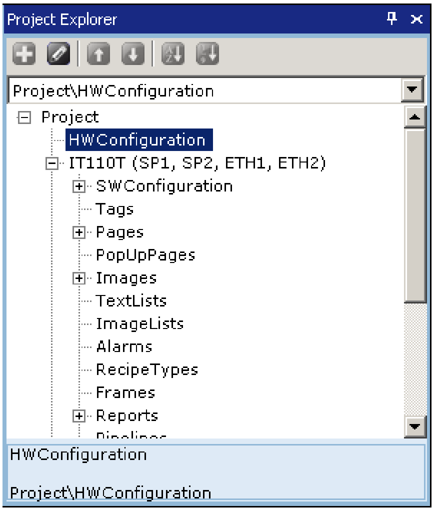
Device:
PLC

Click 'Finish' to create the project

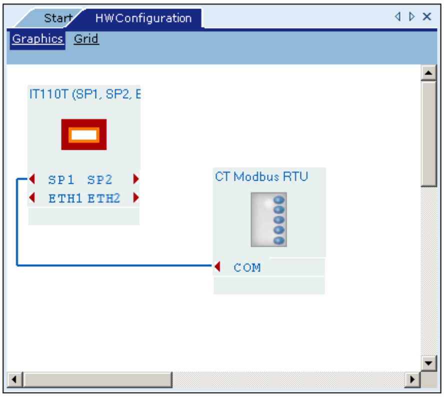
At this point all the data needed to create a project have been entered; the Wizard presents a summary of the choices made and asks for confirmation. If the choices are correct, you can confirm by clicking ‘End’, otherwise you can review your choices by clicking ‘Back’.
Once the choices have been confirmed, POLYMATH will compile the Hardware configuration of the project for you. At this point you can start the real editing of the project.

Changing elements within a project

At any given moment it is possible to add, change or cancel elements and connections which are part of project’s Hardware configuration; all you need to do is double-click on the option ‘Hardware Configuration’ in the ‘Explore project’ window (see chap. 5, “Project Explorer” page 69); the Hardware Configuration window will now appear and, using this, it will be possible to carry out the operations described below.

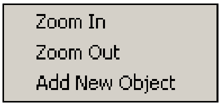


A window appears in which all project hardware elements are present.



There are three options for adding new devices to the project :

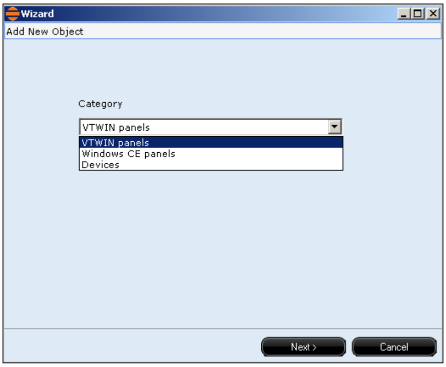
- Use the right mouse key to click inside the white configuration page and select "Aggiungi Nuovo Oggetto" (Add New Object) from the menu



- Click on the ➕ key present in the tools bar

- Select Fields -> ✚ Create...from the main menu

A dialogue window will open from where it is possible to select ESA devices and panels.



Category
VTWIN panels

Now the introduction procedure of the object selected, results identical to that described previously in the "Wizard".

Modification and connection of the project components

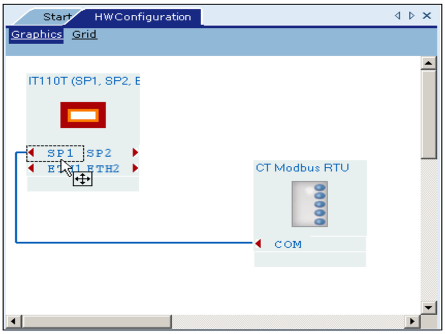
Once all of the useful elements for the realisation of the project have been introduced, they must be connected and the connection modes must be specified.
The "ConfigurazioneHW" (HW Configuration) window displays the VTs and previously-inserted devices. The ports available are indicated for every element (MSP,ASP,COM, etc..).

Moving a VT or device

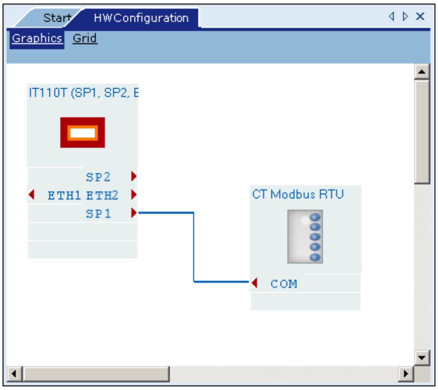
To move a VT or a device to the inside of the Configuration Window just click on ↖ and then on the element to be moved. At this point the element has been selected and you need only drag it (keeping the left mouse key pressed down) till it reaches the desired position; when the left mouse key is released the element will remain in the new position unless it is again moved. If elements containing connections are moved, POLYMATH will automatically update the position and the connections showing in the window.

Moving a port

To move a port, click on ; at this point the icons representing the ports can be selected, as in the following example.



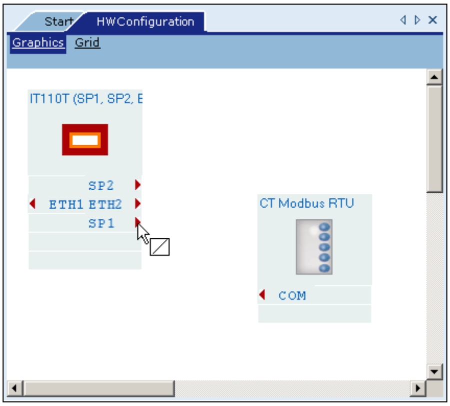
After clicking on the port to be moved just drag it (keeping the left mouse key pressed down) till it reaches the desired position; when the left mouse key is released the port will remain in the new position unless it is again moved.



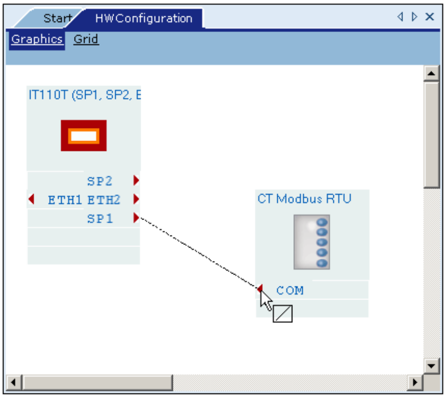
If ports that are references to existing connections are moved, POLYMATH will automatically update the position and the connections showing in the window without altering their nature.

Connecting two elements

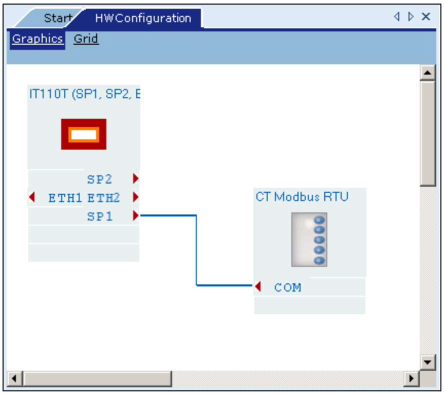
If the page contains at least one ESA panel and one device you will be able to specify the mode of the connection between them. If you want to add a connection you have first to click on and then go on to click inside a free port (one that is not already a reference to another connection). When the pointer nears an available port, a small rectangle will appear next to the pointer containing a connection thread as shown in the figure.



Without releasing the left mouse key, you can proceed to specify the connection path (a horizontal line appears).



To establish the second terminal of the connection release the mouse as soon as the black line reaches the port you wish to include in the connection. When the pointer nears an available port, a small rectangle will appear next to the pointer containing a connection thread.

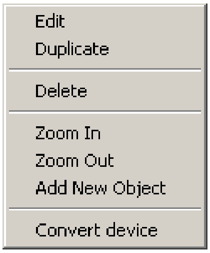


The connection will appear as a broken blue line between the two reference ports.

Important note: *In Simple Project mode it is not possible to create connections between two panels or between two devices; were such a need to arise, it would be necessary to create a network project.*

Operations on VTs and devices

To change a VT or a device in the Hardware Configuration window you need first to select it: click on ↖ and then on the element itself.
Once the object has been selected, just click with the right-hand key on the same to be able to access the following editing menu :



Using the 'Edit' option you can make changes to the properties of the object; 'Duplicate' creates within the Configuration Window an identical copy of the object that has been selected (all the properties of the first are copied into the second). The 'Cancel' option eliminates the element from the project, while the ‘Cut’, ‘Copy’ and ‘Paste’ keys have their usual functions, typical when operating in Windows. Besides these there are the Zoom options which allow you to edit the dimensions of the display of the objects.

Eliminating a VT or device

To eliminate a VT or a device from the Hardware Configuration window just click on ↖ and then on the element to be eliminated. To eliminate it, once the object has been selected, press the ‘Canc’ key of the keyboard or alternatively click with the right-hand key of the mouse on the element, then, using the drop-down menu that appears click on ‘Cancel’.

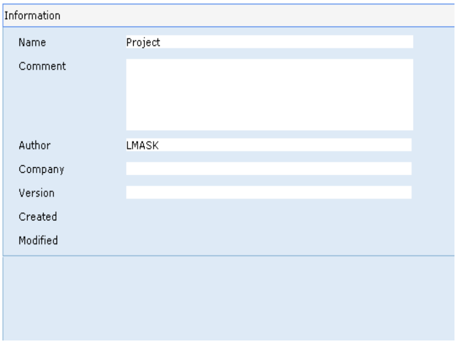
Eliminating a connection

To eliminate a connection from the Hardware Configuration window just click on ↖ and then on the connection (line) to be eliminated. To eliminate it, once the connection has been selected, press the ‘Canc’ key of the keyboard or alternatively click with the right-hand key of the mouse on the element, then, using the drop-down menu that appears click on ‘Cancel’.

Changing a project’s data

Changes to the general data of a project can be made at any moment throughout the project editing process (over and above changes to its components as seen in the last paragraph).
To access the editing menu of a project, double-click the ‘Project’ option within the ‘Explore project’ menu (see chap. 5, “Project Explorer” page 69). There are three editing masks: User Information, File Information and Components

User Information

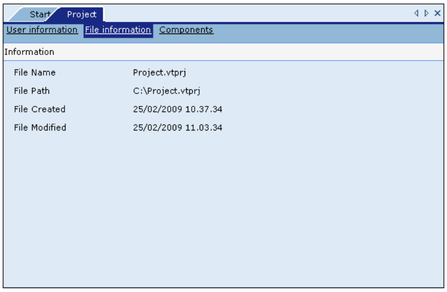


Information

Name	Project
Comment	
Author	LMASK
Company	
Version	
Created	
Modified	

Using the User Information mask you can edit general data relating to the project, such as Name, Comment (optional), Author, Company and Version. The data relating to the creation and editing of the project are not editable.

File Information



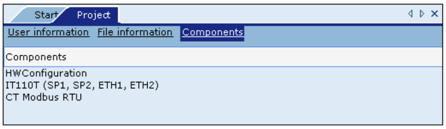
Information

File Name	Project.vtprj
File Path	C:\Project.vtprj
File Created	25/02/2009 10.37.34
File Modified	25/02/2009 11.03.34

The File Information mask contains the data relating to the current file in which the project is saved; such data contains information regarding the name of the file, the remote path in which the file is saved and the creation and editing dates of the file.

Note: *The name of the project and the name of the file are two quite distinct things: the name of the project is a project identifier used only within POLYMATH software while the name of the file serves to distinguish the file within the File System of the user's PC.*

Components



Components

HWConfiguration
IT110T (SP1, SP2, ETH1, ETH2)
CT Modbus RTU

The Components mask lists all the devices and ESA panels involved in the current project and added in the course of the creation of the project. By clicking on each element in the list you can access the corresponding editing mask.

Saving a project

At any point throughout the process of editing the project the user can save his or her work onto hard disk or a removable support.
There are three options for saving the project into a file:
- File -> Save from the main menu
- Press CTRL+S on the keyboard together
- Click on 💾

When the project file is overwritten, POLYMATH automatically creates a backup file with the extension *.vtprj.bak saving it into the folder the user is working in. In this way there is always a reserve copy of the original project; to use and edit the backup copy just rename the extension, changing it from *.vtprj.bak to *.vtprj and reopen the project in POLYMATH.

<u>Important note:</u> *When the Save command described above is used the currently open file is overwritten (or written onto a new file in the case of a new project); to maintain the original file you must choose File->'Save as...' from the main menu and supply a name or a different path.*

Opening a project

When the application is launched or in the course of the work on POLYMATH you can proceed to work on a project previously saved onto Hard Disk or onto a removable support.
There are three options for opening a new file:
- click on 📂
- File -> Open from the main menu
- click on 🏠 to go to the Home Page of the program. Then click on 'Open existing project'

In all these cases an exploration window opens that allows you to select project files (*.vtprj) from within your resources.

<u>Note:</u> *When you enter the Home page of the program by means of a click on 🏠 a list of recently opened files ordered chronologically according to their last editing date. This procedure is simplest and quickest if you often work with the same files.*

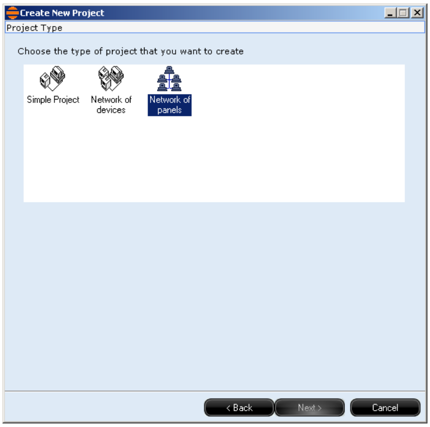
Network project

The network project makes it possible for several terminals to communicate, share and manage data simultaneously.
One's own project is present on each terminal, where Tags are shared and can be monitored by all of the network participants.

Creation of a network project

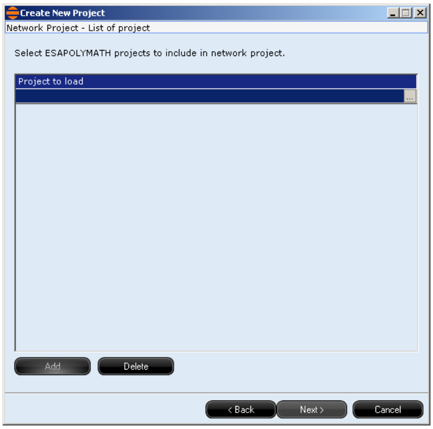
The sequence of operations to be carried out to create a "Network Project" will be shown in the following images.

To create a new project, having opened the "Wizard" mode, select the option "Panel network" and then press "Forward".



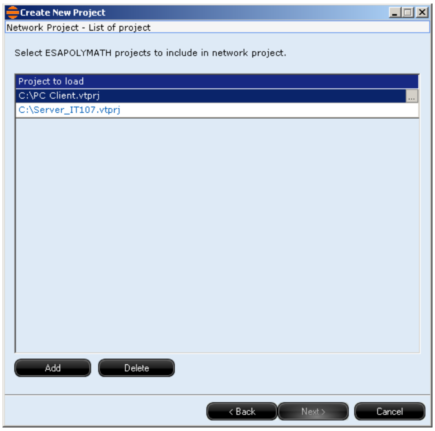
Choose the type of project that you want to create

Click "Add" to download the projects that make up the network, one at a time :



Select ESAPOLYMATH projects to include in network project.

Once the projects have been downloaded click "Forward" (in this example we will put two projects on the network, "Server_IT107.vtprj" and "PC Client.vtprj") :



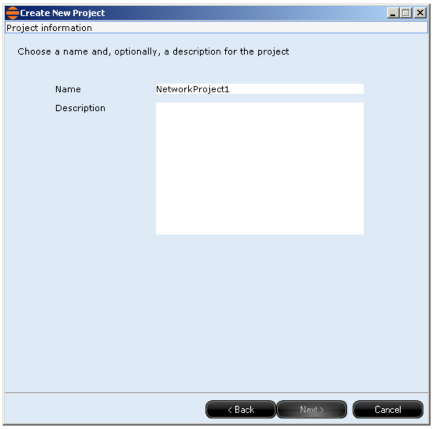
Create New Project

Network Project - List of project

Select ESAPOLYMATH projects to include in network project.

Project to load
C:\PC Client.vtprj
C:\Server_IT107.vtprj

Assign a name to the network project and click "Forward" :



Create New Project

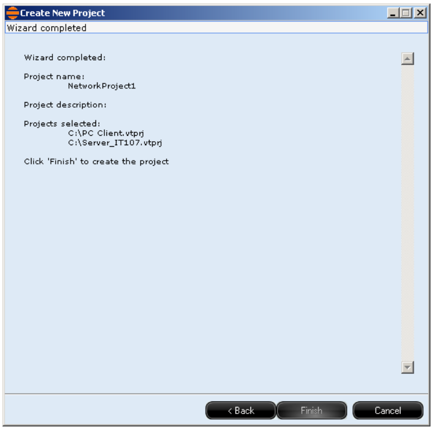
Project information

Choose a name and, optionally, a description for the project

Name: NetworkProject1

Description:

Then click "End".



Wizard completed:

Project name:
NetworkProject1

Project description:

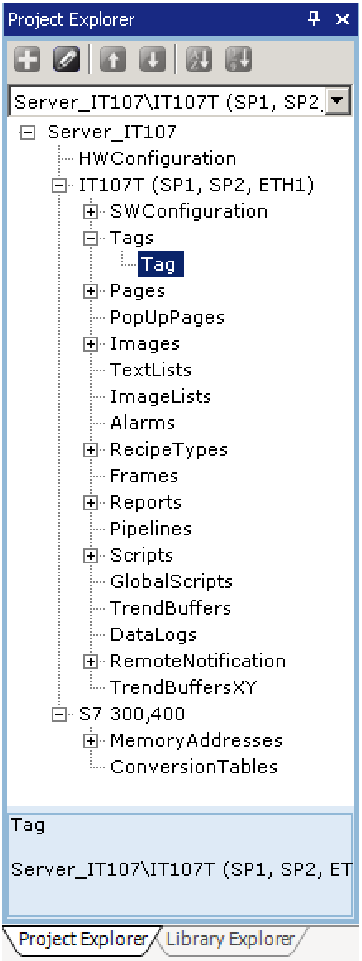
Projects selected:
C:\PC Client.vtprj
C:\Server_IT107.vtprj

Click 'Finish' to create the project

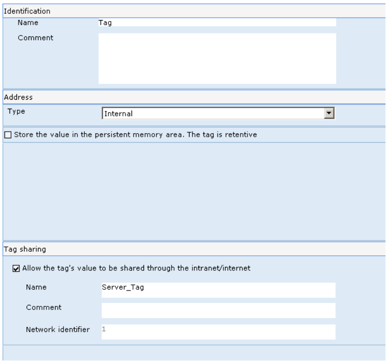
We will now examine the individual projects which make up the network project, in particular the part of the project referring to the shared variables.

"ServerIT107" Project

From "Explore project" double-click the voice "Tag" from the "Tags" option :



From the editing area, the features of the tag to be shared can be observed on the "General" mask. In our example, the tag will be "Internal" :



Identification

Name Tag

Comment

Address

Type Internal

☐ Store the value in the persistent memory area. The tag is retentive

Tag sharing

☑ Allow the tag's value to be shared through the intranet/internet

Name Server_Tag

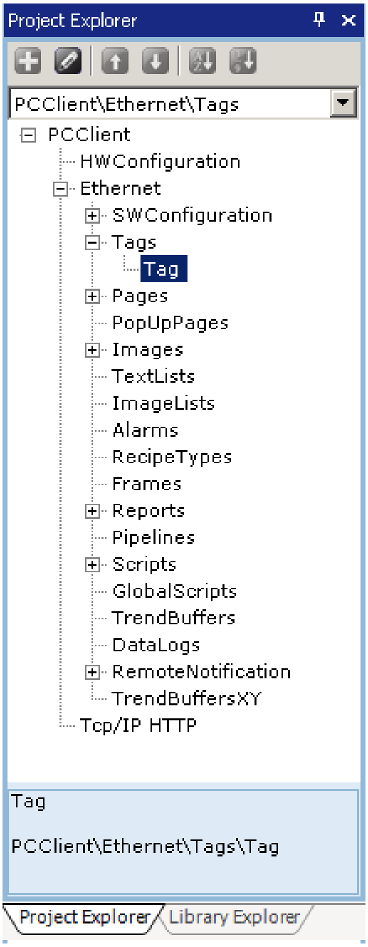
Comment

Network identifier 1

Select the ("Share Tag") option and assign a name (in this case "Server_Tag") so that it can be seen by the other participants.

"PC Client" Project "

From "Explore project" double-click the voice "Tag" from the "Tags" option :



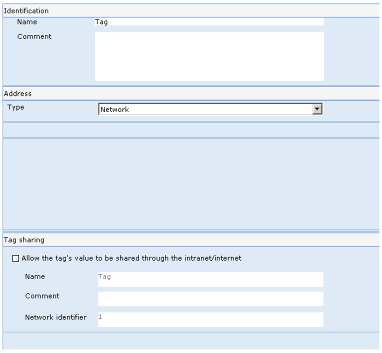
PCClient\Ethernet\Tags

- PCClient
  - HWConfiguration
  - Ethernet
    - SWConfiguration
    - Tags
      - Tag
    - Pages
    - PopUpPages
    - Images
    - TextLists
    - ImageLists
    - Alarms
    - RecipeTypes
    - Frames
    - Reports
    - Pipelines
    - Scripts
    - GlobalScripts
    - TrendBuffers
    - DataLogs
    - RemoteNotification
    - TrendBuffersXY
  - Tcp/IP HTTP

Tag

PCClient\Ethernet\Tags\Tag

From the editing area, the features of the tag to be monitored can be observed on the "General" mask. The tag must be the "Network" type :



Identification

Name: Tag

Comment:

Address

Type: Network

Tag sharing

☐ Allow the tag's value to be shared through the intranet/internet

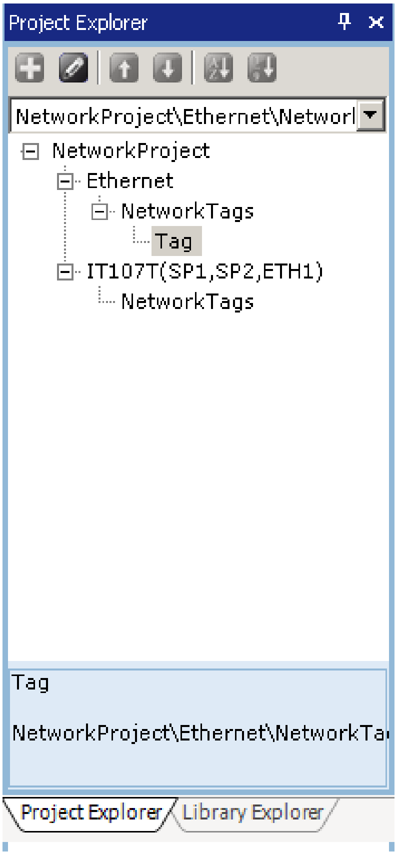
Name: Tag

Comment:

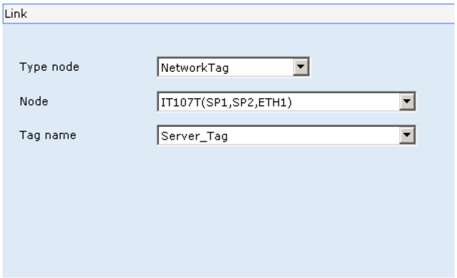
Network identifier: 1

"Network" Project

We will now examine the previously created network project which contains the two sub-projects just shown :



From '"Explore project", double-clicking the voice "Tag" from the "Network Tags" option, the editing area is accessed.
On the "Link" mask, the features of the tag to be monitored can be determined :


Link

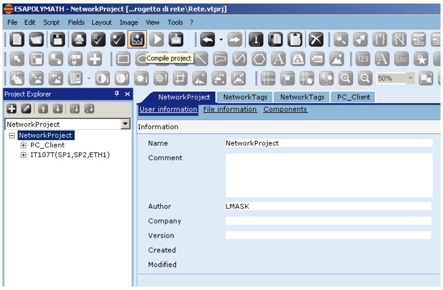
Type node	NetworkTag
Node	IT107T(SP1,SP2,ETH1)
Tag name	Server_Tag

The tag must have the following features :

- It must be the "Network" type.
- The node must be indicated. The node is the point where the tag is shared (in our example the shared tag is in the IT107T terminal).
- The name of the tag to be monitored must be determined (in our example the name of the tag to be monitored is Server_Tag).

Compilation of the network project

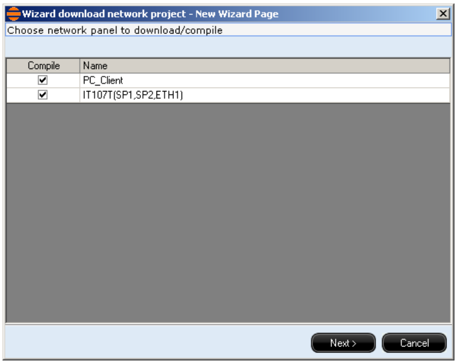
The compilation and the download of the projects that make up the network project must be carried out inside of the network project.
Click on the icon to fill out the project :



ESAPOLYMATH - NetworkProject [...rogetto di rete\Rete.vtprj]

Name: NetworkProject
Comment:
Author: LMASK
Company:
Version:
Created
Modified

The following mask will appear from which one can choose to fill out all the projects that make up the network or, if only one project has been varied, to fill out only the modified one :



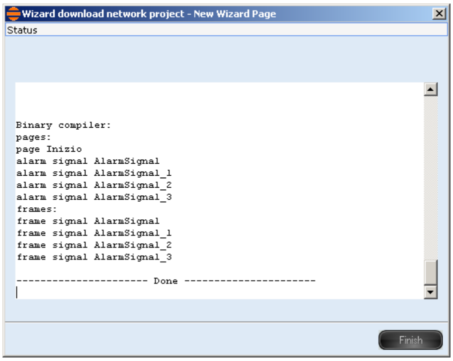
Wizard download network project - New Wizard Page

Choose network panel to download/compile

Compile	Name
☑	PC_Client
☑	IT107T(SP1,SP2,ETH1)

Click "forward".

At the end of the compilation, the following mask will appear :

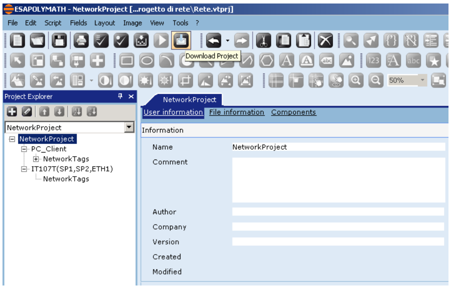


```
Binary compiler:
pages:
page Inizio
alarm signal AlarmSignal
alarm signal AlarmSignal_1
alarm signal AlarmSignal_2
alarm signal AlarmSignal_3
frames:
frame signal AlarmSignal
frame signal AlarmSignal_1
frame signal AlarmSignal_2
frame signal AlarmSignal_3

--------------------- Done ---------------------
```

At this point, by clicking "end", the projects that make up the network are ready to be transferred to their respective terminals.

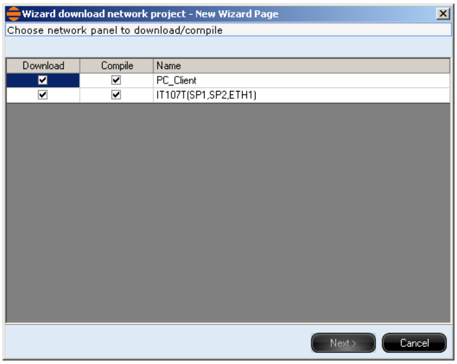
Download the network project

Click on the icon to transmit the projects to their respective terminals :



Name NetworkProject
Comment
Author
Company
Version
Created
Modified

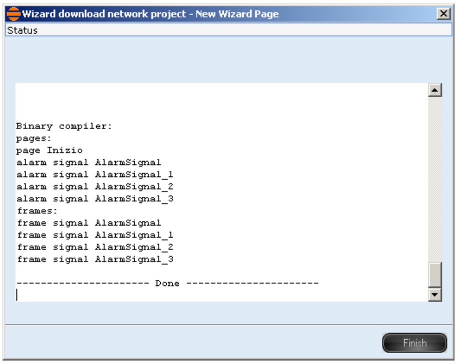
The following mask will appear from which one can choose to transfer all the projects that make up the network or, if only one project has been varied, to transfer only the modified one :



Choose network panel to download/compile

Download	Compile	Name
☑	☑	PC_Client
☑	☑	IT107T(SP1,SP2,ETH1)

Click "End" when the transfer is complete :

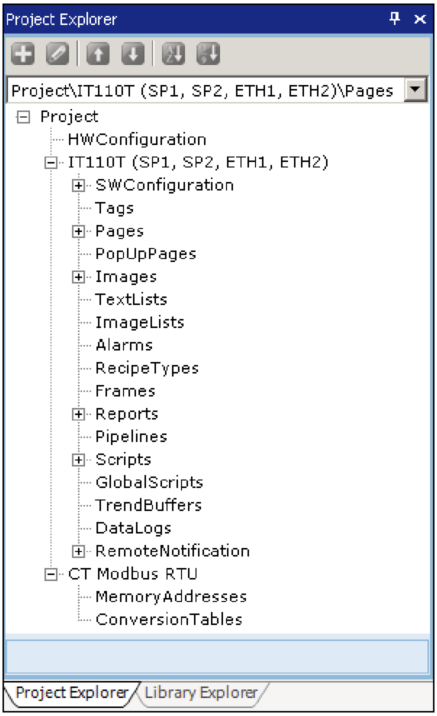


Status

```
Binary compiler:
pages:
page Inizio
alarm signal AlarmSignal
alarm signal AlarmSignal_1
alarm signal AlarmSignal_2
alarm signal AlarmSignal_3
frames:
frame signal AlarmSignal
frame signal AlarmSignal_1
frame signal AlarmSignal_2
frame signal AlarmSignal_3

---------------------- Done ----------------------
```

5. Project Explorer

The principal anchorable window in POLYMATH is the Project Explorer window from which the structure and operations of the project can be controlled. In this chapter we describe in detail all the characteristics that can be configured using Project Explorer.



```
Project\IT110T (SP1, SP2, ETH1, ETH2)\Pages
⊟ Project
    HWConfiguration
  ⊟ IT110T (SP1, SP2, ETH1, ETH2)
    ⊞ SWConfiguration
      Tags
    ⊞ Pages
      PopUpPages
    ⊞ Images
      TextLists
      ImageLists
      Alarms
      RecipeTypes
      Frames
    ⊞ Reports
      Pipelines
    ⊞ Scripts
      GlobalScripts
      TrendBuffers
      DataLogs
    ⊞ RemoteNotification
  ⊟ CT Modbus RTU
      MemoryAddresses
      ConversionTables
```

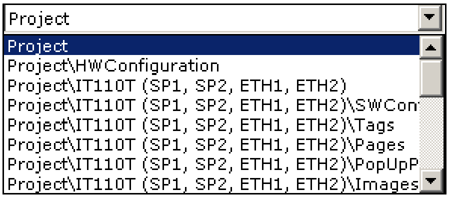
The Project Explorer window contains all the editable objects arranged as a tree diagram in which the parent element is always the project to which the Hardware configuration is anchored, the ESA terminals (with their attributable properties) and the connected devices (with their related settings).

Note: *A single click on an element in the tree selected it, while a double click allows you to edit.*

If Project Explorer should fail to appear on the screen because it has previously been closed, it can be brought back to the screen by clicking on the icon of the toolbar or using the main menu by clicking on Display->Show->Project Explorer. Like all anchorable windows, Project Explorer, too, can be moved, reduced to an icon or closed (see chap. 3, "Moving Anchorable windows" page 41).

There are six buttons present in the upper part of the window :

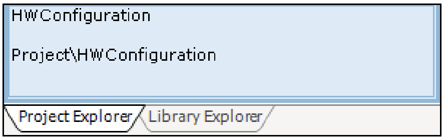
- The button is used to add one element to the category selected in the tree chart. If the entire project is selected, this key can be used to insert new VTs or devices.
- The button is used to enter editing mode for the element selected in the tree chart.
- The button is used to shift an element upwards.
- The button is used to shift an element downwards.
- The button is used to put the objects in order.
- The button is used to put the page numbers in order.



Project

Project
Project\HWConfiguration
Project\IT110T (SP1, SP2, ETH1, ETH2)
Project\IT110T (SP1, SP2, ETH1, ETH2)\SWCon
Project\IT110T (SP1, SP2, ETH1, ETH2)\Tags
Project\IT110T (SP1, SP2, ETH1, ETH2)\Pages
Project\IT110T (SP1, SP2, ETH1, ETH2)\PopUpP
Project\IT110T (SP1, SP2, ETH1, ETH2)\Images

There is also a drop-down menu from which any of the categories making up the menu can be selected.



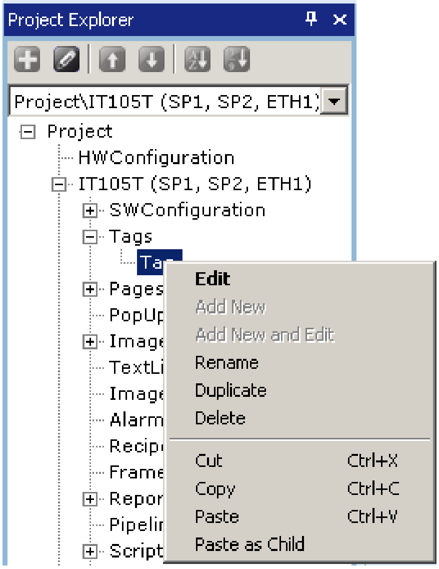
HWConfiguration

Project\HWConfiguration

Information relating to the element selected is displayed in the lower section of the window. Here you will find the name, the comment and the path of those objects chosen when creating the project.

Operating on elements within the Project Explorer window

There is a series of cumulative functions applicable to all the categories or elements of the Project Explorer window irrespective of their nature. These functions are contained in a menu called up by clicking with the right-hand key on the object in question as illustrated in the figure



The functions that can be selected are:

- Edit, to enter editing mode
- Add new, to add an element to a category
- Add new and Edit, to add an element to a category and directly access the editing page (in the Work area)
- Rename, to change the name of the object selected
- Duplicate, to create an exact copy of the element selected; the properties that must remain unique within the project (e.g. Name, Identifying Number, Description) are not copied but are automatically assigned a valid value
- Delete, to delete the element selected
- Cut, to eliminate the element selected and copy it into the clipboard
- Copy, to copy the element selected into the clipboard
- Paste, to paste in the element contained in the clipboard
- Paste as Child, to paste in the element contained in the clipboard as Child of the element selected
- Import texts from : to import texts inside of the project in the ".xls" or ".csv" format

- "Export texts to": exports project elements (texts, alarms, pages etc.) onto the Hard Disk or the USB storage device
- "Translations": displays all the project texts on a table simultaneously, to be able to edit/translate them to the desired languages at the same time
- "Unused Tags/Variables Removal": to remove the Tags and Variables not used in the project
- "Convert panel": to convert the panel with another
- "Convert device": to convert the device with another changing the communication protocol
- "VT Simulator": to simulate VT terminal project pages
- "Runtime Simulator": to simulate the IT terminal "real-time" operation
- "Crossed reference": to search for/verify the existence of a certain variable/page/script/function inside the project

Elements of Project Explorer

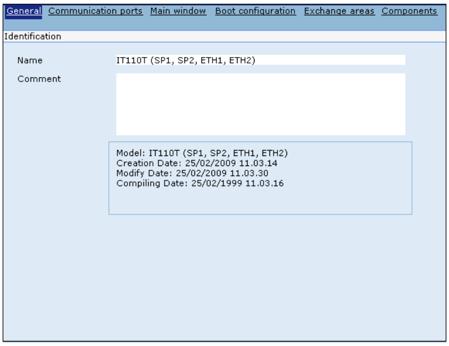
The tree-type structure of Project Explorer allows the user to access the configurator of all the components of a POLYMATH project (with the exception of the graphic elements that are configured by the Editor property); the Project (see chap. 4, “Changing a project’s data” page 53) and Hardware configuration editor (see chap. 4, “Managing the project” page 43) have already been described in the previous chapter while the other objects will be described in this chapter.
To access the editor of an element just double-click on it in Project Explorer; the corresponding editing window will appear in the work area.
We will start by describing the elements that can be associated with ESA terminals and then we will illustrate the settings of devices connected to these terminals.

Setting the panel

When the Project Explorer icon corresponding to the panel added to the project is double-clicked, the user is able to edit its characteristics.

Editing the panel is organized via 6 work windows: General, Communication ports, Main window, Configuration Boot, Exchange areas and Components. The user can move from one window to another at any time without losing any of the changes made.

General



Identification

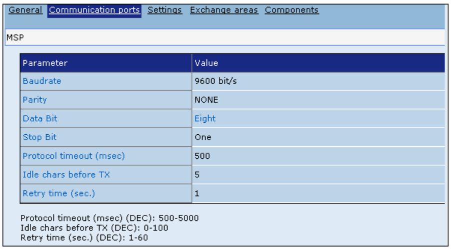
Name IT110T (SP1, SP2, ETH1, ETH2)

Comment

Model: IT110T (SP1, SP2, ETH1, ETH2)
Creation Date: 25/02/2009 11.03.14
Modify Date: 25/02/2009 11.03.30
Compiling Date: 25/02/1999 11.03.16

The work window 'General' is used to change the name of the panel in question and add comments within it to make it distinguishable in the programming phase with POLYMATH. The bottom of the window shows information on the date of creation, editing and compilation of the project.

Communication ports



MSP

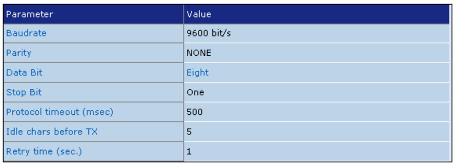
Parameter	Value
Baudrate	9600 bit/s
Parity	NONE
Data Bit	Eight
Stop Bit	One
Protocol timeout (msec)	500
Idle chars before TX	5
Retry time (sec.)	1

Protocol timeout (msec) (DEC): 500-5000
Idle chars before TX (DEC): 0-100
Retry time (sec.) (DEC): 1-60

In this window it is possible to configure the communication method between the panel and the device; the parameters can be configured in function of the connected panel and device.

The bottom of the window shows the range allowed by the protocol for each value inserted

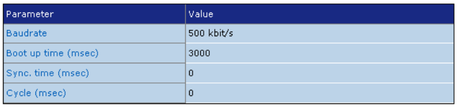
MSP/ASP/SP1/SP2



Parameter	Value
Baudrate	9600 bit/s
Parity	NONE
Data Bit	Eight
Stop Bit	One
Protocol timeout (msec)	500
Idle chars before TX	5
Retry time (sec.)	1

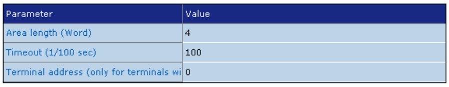
The first four parameters are always available in the configuration whilst the others vary according to the protocol used on the gate.

CAN



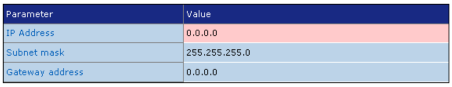
Parameter	Value
Baudrate	500 kbit/s
Boot up time (msec)	3000
Sync. time (msec)	0
Cycle (msec)	0

DP



Parameter	Value
Area length (Word)	4
Timeout (1/100 sec)	100
Terminal address (only for terminals wi	0

ETH1/ETH2

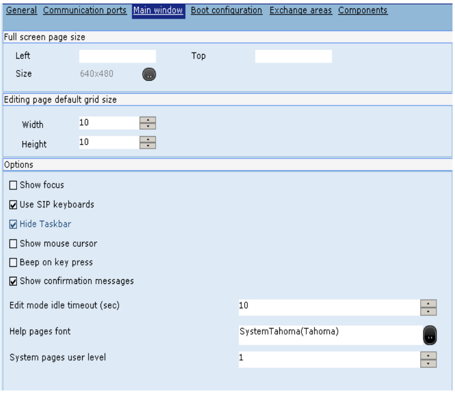


Parameter	Value
IP Address	0.0.0.0
Subnet mask	255.255.255.0
Gateway address	0.0.0.0

COM 0

For this communication gate, no parameters are foreseen because it can be set via script

Main window



Full screen page size

Left

Top

Size 640x480

Editing page default grid size

Width 10

Height 10

Options

☐ Show focus

☑ Use SIP keyboards

☑ Hide Taskbar

☐ Show mouse cursor

☐ Beep on key press

☑ Show confirmation messages

Edit mode idle timeout (sec) 10

Help pages font SystemTahoma(Tahoma)

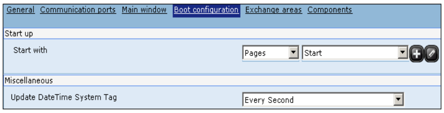
System pages user level 1

The Work window of the main window shows the dimensions in pixels of the page displayed on the panel; in general these dimensions are unchangeable and depend on the features of the panel hardware. On a PC, for example this one, character can be configured because it is not possible to determine the resolution of the screen. Nevertheless, it is possible to change the grid for arranging objects in the page (see chap. 6, “Managing a page” page 167); the default values for these dimensions are set at 10 pixels for the width and 10 pixels for the length. By reducing these values you have more freedom to add and reposition elements within the page (the grids in the work area will be denser); similarly, by increasing these values, the lines will become less dense and there will be less freedom to introduce objects.

Then, providing the operating system of the panel allows this (if not, they will appear disabled), a series of configurable options are available regarding the display of pages on the terminal; the "Show focus" option can be selected (100% zoom), the user can decider whether to display the title bar, the ‘Reduce to icon’ button, the window focus (practically speaking, the focus highlights the currently selected object or button), the on-screen keyboard for entering data and whether to hide the applications bar, whether the confirmation message is to be shown.

The last three options allow the user to set the time-out in the edit phase, the font for the Help pages and the password level for accessing the system pages (see chap. 5, “Password configuration” page 81).

Configuring the Boot



Start up

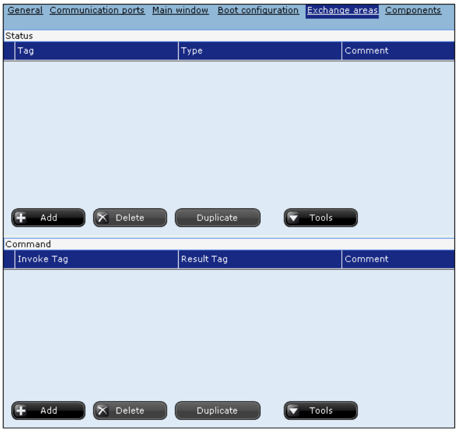
Start with: Pages, Start

Miscellaneous

Update DateTime System Tag: Every Second

This mask allows the user to set the page to be displayed when the project is opened. By clicking on the ✚ icon a new page can be added, while the✎ icon opens the editor of the page selected.
In addition, the Runtime refresh frequency of the DateAndTime system Tag (see “Appendix A - System Variables” page 555) can be defined; a refresh of once a second or once a minute can be set.

Exchange areas



Status

Tag	Type	Comment

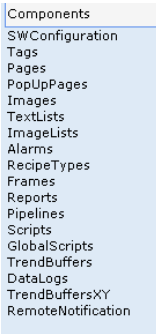
Command

Invoke Tag	Result Tag	Comment

ESA panels communicate with the field devices to which they are connected; to make this information exchange possible the panel and the device in question share memory areas from which data can be taken and into which it can be written. In reality, an exchange area is a tag-area (of one or more words) located in the field device.
The two main categories of exchange areas are the status areas and the command areas. The former are for the panel to write information regarding the working of the device connected while the second are read by the VT which then answers by running particular operations in relation the value read (in practice the device uses the command areas to send

automatic commands to the VT). From this mask it is possible to proceed to add (using the 'Add' key), delete (using the 'Delete' key) or duplicate (using the 'Duplicate' key) both exchange areas and command areas. Once an exchange area is added, an area-type variable must be assigned to it (see chap. 5, "Value" page 91) for reference. In the case of command areas it is also necessary to introduce a response tag (variable) to which the data relating to the outcome of the operation indicated is written.
This variable can also be newly created and edited by clicking on the adjacent icon; this can then naturally also be used inside the project or accessed using Scripts.
To be able to see in detail the list of activities that can be run using the status area and the command area, the reader is advised to consult the appropriate appendices (see "Appendix C - Status area" page 575 e see "Appendix D - Command area" page 579).

Components



This page offers only a summary of the components that can be assigned to ESA panels; by clicking on each of these the appropriate main editing page can be accessed.

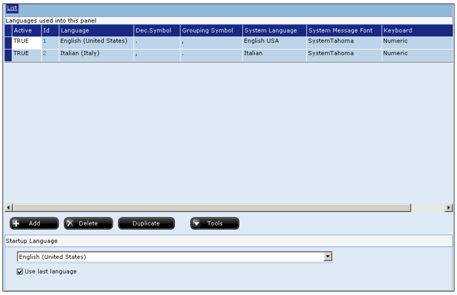
Software Configuration

The first option you find on the menu of the panel in Project Explorer is the one relating to the configuration Software. To this area there belong the setting windows for the following elements:

- Languages
- Fonts
- Password configuration
- SystemAlarms
- SystemMessages
- GlobalKeys
- Timers.

To access the general editor of each option just double-click on the appropriate name in Project Explorer. The following paragraphs will carry detailed information on the features that can be configured for each element.

Languages



Languages used into this panel

Active	Id	Language	Dec.Symbol	Grouping Symbol	System Language	System Message Font	Keyboard
TRUE	1	English (United States)	.	,	English USA	SystemTahoma	Numeric
TRUE	2	Italian (Italy)	,	.	Italian	SystemTahoma	Numeric

Startup Language

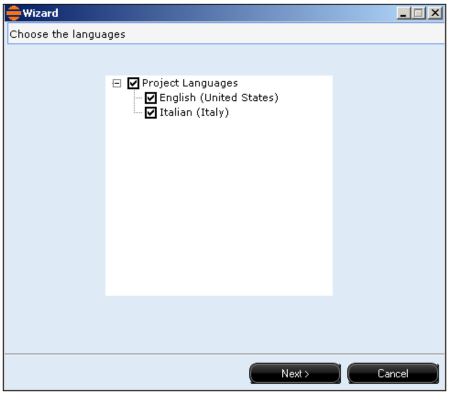
English (United States)

☑ Use last language

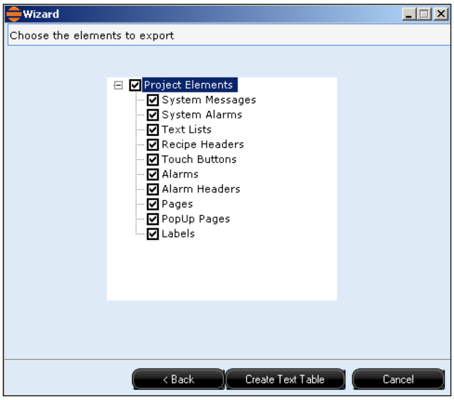
The configuration window for Languages allows the user to manage the project languages that can be displayed on the panel. Up to eight languages can be introduced at the programming level and at least one language always needs to be present. To introduce a new language to the project just click on ‘Add’.
For each language added a decimal and group needs to be indicated as well as a system language and a Font for the related system messages. Naturally you can delete languages present in the project (by pressing ‘Delete’), duplicate (by clicking on ‘Duplicate’) or change the settings of the existing ones (by clicking on the corresponding fields in the list table); in this window you can also indicate the language to be used when the project starts up on the terminal.

The "Strumenti" (Tools) key allows to access two windows :

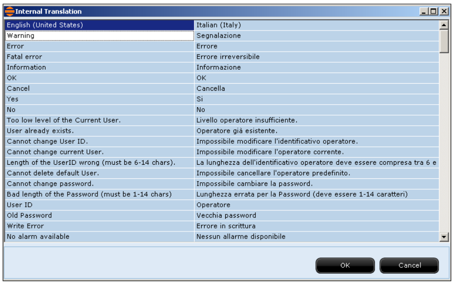
- "Traduzioni" (Translations); if a multilanguage project is created, every time a text is introduced, the possibility of translating it into all languages is given. In this way a Wizard will start that will guide the user through the translation process.
- "Configurazione colonne" (Columns configuration) for modifying the structures of the columns at will.



Select the desired languages and select "Avanti" (Next).



Select the elements to be exported and click on the "Crea tabella" (Create table).



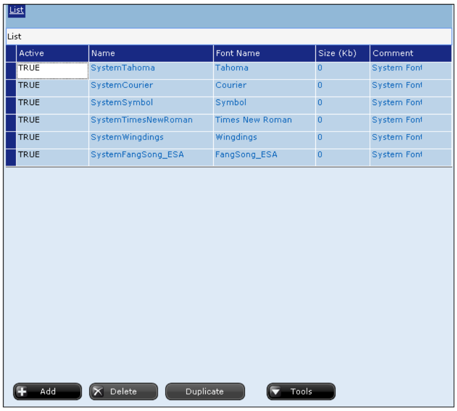
English (United States)	Italian (Italy)
Warning	Segnalazione
Error	Errore
Fatal error	Errore irreversibile
Information	Informazione
OK	OK
Cancel	Cancella
Yes	Si
No	No
Too low level of the Current User.	Livello operatore insufficiente.
User already exists.	Operatore già esistente.
Cannot change User ID.	Impossibile modificare l'identificativo operatore.
Cannot change current User.	Impossibile modificare l'operatore corrente.
Length of the UserID wrong (must be 6-14 chars).	La lunghezza dell'identificativo operatore deve essere compresa tra 6 e
Cannot delete default User.	Impossibile cancellare l'operatore predefinito.
Cannot change password.	Impossibile cambiare la password.
Bad length of the Password (must be 1-14 chars)	Lunghezza errata per la Password (deve essere 1-14 caratteri)
User ID	Operatore
Old Password	Vecchia password
Write Error	Errore in scrittura
No alarm available	Nessun allarme disponibile

Once the texts have been added just click on OK to save the changes made or on Delete to delete them. There is no default translation but POLYMATH furnishes the same text (the one introduced for the main language) for all the languages.

Note: *There is no particular limit for the translation of secondary languages; their length may exceed that of the reference language.*

Note: *While programming with POLYMATH, the display language for the project elements (e.g. labels and buttons) can be changed simply by selecting the required language from the Display>project language menu or the option from the tools menu (if the field has been set to be present); in both cases the changes will be immediate and all the objects will be displayed in the required language.*

Character fonts



List

List

Active	Name	Font Name	Size (Kb)	Comment
TRUE	SystemTahoma	Tahoma	0	System Font
TRUE	SystemCourier	Courier	0	System Font
TRUE	SystemSymbol	Symbol	0	System Font
TRUE	SystemTimesNewRoman	Times New Roman	0	System Font
TRUE	SystemWingdings	Wingdings	0	System Font
TRUE	SystemFangSong_ESA	FangSong_ESA	0	System Font

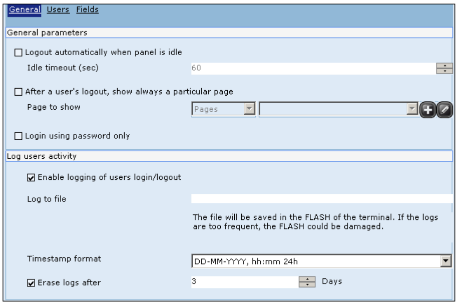
The window related to the fonts allows the user to manage (introduce, delete and edit the name or property) of all the character fonts used in the project. There is a series of default fonts present in the project (that cannot be cancelled), but new ones can be added by choosing from those installed on your PC. It is also possible to associate a comment to each font added to be displayed only within POLYMATH; for each font there is also the indication of the memory that to be occupied by installing the font in question.

For each project up to 8 fonts can be introduced in the programming phase (four default and four chosen by the user).

Password configuration

Within a project you can define authentication levels to maintain control of access to specific areas. The purpose of this feature is to distinguish and control the level of operational freedom for each user in the course of their work session. Using POLYMATH the programmer can proceed to establish access policies for particular features (e.g. access to buttons, pages, recipe management, etc.) thereby stopping operators without the proper credentials from accessing or editing data in an improper manner. Each operator, when using the panel, must be recognised by the system by entering an identifying name and a password for the appropriate level of access (logon operation). It is envisaged that only one operator can be logged on and use the panel at any given moment; each operator can logout at any time.
Up to ten access levels can be defined and the lowest level (typically level 1) is the one with the highest degree of operational freedom. Each user who has not gone through the login procedure will be treated by the system as a level 10 user (the lowest degree of freedom) and can access only the features available to that level. To run an operation of a level lower than ten, you will be asked to login again using a special Popup page predefined by the system.
Use POLYMATH to define the initial users' levels, that is the levels of those present at the start-up of the project. You can also add or edit users directly in runtime. To do this you can introduce into their pages a predefined check called User List (see chap. 6, "User List Table" page 304).
For security reasons each operator with access to the pages for changing User Passwords (using the User List check) can display and change the access credentials (name-password) only of users with the same or higher-numbered levels than his/her own (e.g. an operator on level 5 can see and change the password of levels 5,6,7,8,9 and 10).
The password configuration of the access levels is made up of three edit masks: General, Users and Fields grid.
By means of the "Mask Password edit" it is possible to determine if the password is to be displayed or hidden with asterixes during configuration.

General



General parameters

☐ Logout automatically when panel is idle

Idle timeout (sec) 60

☐ After a user's logout, show always a particular page

Page to show Pages

☐ Login using password only

Log users activity

☑ Enable logging of users login/logout

Log to file

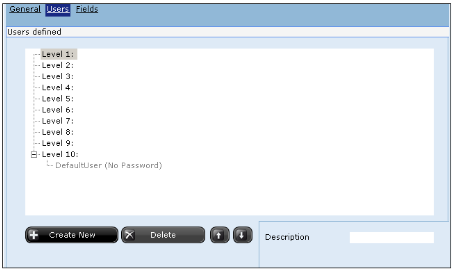
The file will be saved in the FLASH of the terminal. If the logs are too frequent, the FLASH could be damaged.

Timestamp format DD-MM-YYYY, hh:mm 24h

☑ Erase logs after 3 Days

The General mask is used to configure the panel such that it executes the logout automatically after a certain period of inactivity; you can also define which page to go to see once a user has completed the logout.
This window can also be used to set the procedures for recording the user login/logout operations; this function is particularly useful where it is important to be able to maintain a history file of accesses. The files in which the data is saved (a valid file name must be given when working in a Windows) are editable, as is the format of the date-time and whether to program the logs after a certain period of time. The log file is saved in text format in the folder \log (see chap. 8, “Transferring data” page 362).

Users



Users defined

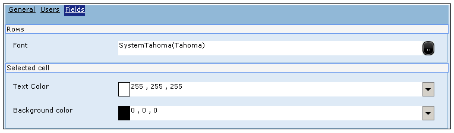
Level 1:
Level 2:
Level 3:
Level 4:
Level 5:
Level 6:
Level 7:
Level 8:
Level 9:
Level 10:
DefaultUser (No Password)

Description

The Users window is the one used to show the participating users and the corresponding passwords. Up to 19 participating users can be introduced. To create a new participant, just click the appropriate level and then Create

new; for each participant created it is essential to indicate a user name and a password (minimum 6 alphanumeric characters, maximum 14).
In addition, each level can be supplied with a comment visible only within POLYMATH in the programming phase. To introduce a comment just click on the level (not the user) and enter the text in the corresponding field.
Once a participation has been registered and selected using the ⬆ and ⬇ icons it can be transferred to a lower or higher level as required. If, however, 'Delete' is pressed, the selected participation is cancelled.

Fields grid



Rows

Font: SystemTahoma(Tahoma)

Selected cell

Text Color: □ 255 , 255 , 255

Background color: ■ 0 , 0 , 0

The Fields grid window is used to set the graphic properties of the cells of the user list table (see chap. 6, "User List Table" page 304). The Font choice box allows you to decide to assign a font to the user list table; by clicking on the ● icon a window appears for specifying the font and using this each project language can have a font assigned to it. In addition, this window can be used to define various properties of the font for the table like dimensions and graphic effects.
In addition you can specify a background and text colour for the cell selected currently. The colour can be selected using RGB values or the colour palette obtainable by clicking on the rectangle of the colour 🟧 or on the selection arrow ▼; the classic Windows colour selection window appears and using this even customized colours can be defined.

SystemAlarms

The system alarms are alarms that are displayed to the operator whenever certain conditions of anomaly occur.
In this section it is possible to access a table containing all the system alarms that are displayed by the panel in particular situations. Alarm messages are displayed for each project language entered. Some messages are unchangeable by the programmer while others are contained in editable fields. In any event, it is always possible to delete the changes made to the translation by clicking on the appropriate button ('Clear Translation'). Use the Project language button to access the

editable list of languages already described in this paragraph (see chap. 5, "Languages" page 78).

Warning: *when editing the texts of the systemalarms (and messages), be careful not to introduce special characters reserved for the system (e.g. '%').*

SystemMessages

System messages are messages displayed to the operator at various points when the panel is in use.
In this section it is possible to access a table containing all the system messages that are displayed. Messages are displayed for each project language entered. Some messages are unchangeable by the programmer while others are contained in editable fields. When editing these strings, be careful not to introduce special characters reserved for the system (e.g.`%'). In any event, it is always possible to delete the changes made to the translation by clicking on the appropriate button ('Clear Translation').
Use the Project language button to access the editable list of languages already described in this paragraph (see chap. 5, "Languages" page 78).

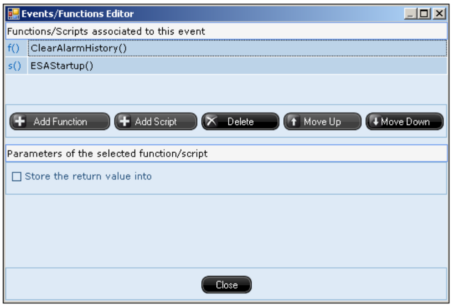
Note: *In ESA terminology, system messages differ from alarms in as much as the former are simple messages set into Dialog Boxes or masks for entering information, while the latter are connected to events correlated to system variables (e.g. flat battery, insufficient space on disk, etc...).*

GlobalKeys

This mask allows the user to define a global mode of behaviour for all F keys (of a virtual or physical keyboard).

Note: *By global mode of behaviour we mean that the key will make it possible to effect the configured function independently of the page being displayed on the panel, while by local mode of behaviour we mean the execution of the function only in the context of the current page (see chap. 5, "F keys" page 106).*

A predefined function or a user Script can be associated with any key simply by double-clicking on the table in the corresponding row or by selecting and clicking on 'Put'; should you wish to delete an already existing association, click on 'Remove' after having made the selection. If you choose to introduce an association with a key, the following dialog window opens :


Events/Functions Editor

Functions/Scripts associated to this event

f()	ClearAlarmHistory()
s()	ESAStartup()

Parameters of the selected function/script

☐ Store the return value into

To add a function just click on 'Add Function' and choose the function required from the list which appears, by clicking on the line just created three times or on the ▾ key, similarly by clicking on 'Add Script' the Script to be associated can be chosen. Up to 2 functions/Scripts can be introduced for each key and these will be executed in the order indicated; to change the order of the functions just move them with the 'Move Up' and 'Move Down' keys. To delete a function just select it and click on the 'Delete' button

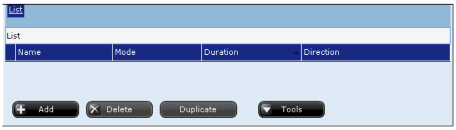
Should a predefined function be chosen to associate with the global key, the lower part of the window can be used to indicate the data related to a correct execution of this (e.g. file name, name of objects, etc..).

Should a Script be chosen to associate with the global key, it will be possible to choose to save the value returned by this Script (if the Script is set to return a value) in a variable.

For details regarding the functions that can be associated and regarding Scripts the reader is advised to consult the sections of the manual devoted to these topics (see "Appendix B - Predefined functions" page 563 e see chap. 9, "Scripts" page 379).

Timers

Timers are tools put at the operator's disposal for programming the execution of certain activities in line with temporal variables calculated directly by the terminal.



The Timers can be used in accordance with the needs of the project, simply by entrusting functions or Scripts to their start, suspend or end count events (see chap. 6, "Events related to Timers" page 167).
Using the general table relating to the Timers you can introduce, delete and duplicate Timers. In relation to each element you can specify the operational mode, the duration and the direction of the count.
There are different modes of operation:

- One-run: the timer starts, allows a certain period of time to elapse, then goes off and stops (one run)
- Normal: the timer works periodically, that is, when it goes off it resets itself and then another cycle starts, indefinitely (continuous run)
- Single alarm: the timer goes off at the date and time specified and then stops
- Alarm time: the timer goes off at the specified time then resets and another cycle starts (continuous run)

Warning: *Irrespective of the type of Timer used, it is always necessary for the Timer to be activated in runtime by the related Start function called up by the button or Script (see "Appendix B - Predefined functions" page 563 and see chap. 9, "The object ESATIMER" page 399), otherwise the related count or control will not be initialised.*

The duration attribute also takes on various meanings depending on the operational mode specified:
If the Mode is One-run or Normal: it represents the trigger time in tenths of a second (0 disables the Timer)
If the Mode is Single alarm: it represents the date-trigger time in ANSI-C format: number of seconds from time 0:0:0 of the

1-January-1970 (the data can be selected in POLYMATH using an convenient calendar window).
If the Mode is alarm time: it represents the trigger time in seconds after midnight; assigning a inadmissible value disables the Timer and is flagged to the operator by means of an error dialog box.
The value of direction, indicates the counting mode of the Timer; this may be arrived at by increasing the count variable or decreasing it (this choice has no operative consequences on the working of the Timer but merely on the internal count value).

Warning: *These types of Timer are software timers, so it is preferable to avoid using them as clocks.*

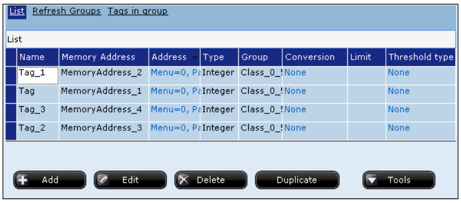
Keyboards

Keyboards can be customized to enter data having the desired form, colour and content, so that they can be used for projects in any language (using Cyrillic, Greek, German, American and Asian characters). Keyboards can be created and saved in the library to be used in further projects.
You can associate a customized default keyboard to any language.
Double-click on keyboard to access the list of keyboards entered by default; select the keyboard you want to change and click on edit. In the Properties Editor, click on Background Image and choose or add an image to use as a background for the keyboard. Arrange the sensitive areas with the relative Key Code so that they correspond to the new keys. To customize the numeric or hexadecimal keyboard, change the Keyboard of the 123 key if you are linking a numeric keyboard to an ASCII keyboard, or of the ABC key if you are linking an ASCII keyboard to a numeric keyboard.
Go to the language selection window and select the "keyboard" column; use the drop-down list to associate the default keyboard to the language. The default keyboard will appear every time you edit a field. Click on ABC or 123 to pass from one keyboard to another.

Variables

Variables are fundamental elements for creating a POLYMATH project ; they allow the programmer to store and arrange data to permit dialog between panel and device. An indefinite number of variables can be created, the limits depending on the memory available on the device.

List



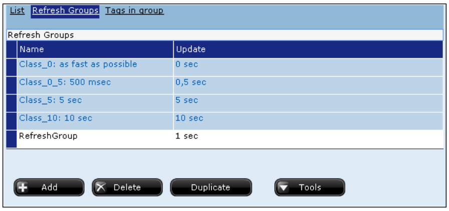
Name	Memory Address	Address	Type	Group	Conversion	Limit	Threshold type
Tag_1	MemoryAddress_2	Menu=0, Pa	Integer	Class_0_5	None		None
Tag	MemoryAddress_1	Menu=0, Pa	Integer	Class_0_5	None		None
Tag_3	MemoryAddress_4	Menu=0, Pa	Integer	Class_0_5	None		None
Tag_2	MemoryAddress_3	Menu=0, Pa	Integer	Class_0_5	None		None

After double-clicking on Project Explorer, you access a table of variables, whose list and classes of update (described in the next subsection) can be managed. Using the list, you can not only introduce new variables, delete them and duplicate them but also edit certain properties (name, memory and Type; the meaning of these properties will be described in the next section).
One alternative method of creating a variable is to click Add in the menu arising from right clicking on Tags in Project Explorer or clicking on ➕ in all those properties to which a variable can be associated. Once a variable is created, it (with its valid name assigned by POLYMATH) will appear under the Tags option of the tree-form diagram; to enter edit mode for this just double-click on it.
If you wish to get to know the list and the meaning of the events that can be associated to a variable, you are advised to consult the next chapter (see chap. 6, “Events related to variables” page 163).

__Note:__ By dragging a variable from Project Explorer onto a page in the work area, POLYMATH automatically creates a data field (numerical or ASCII) associated to the variable within that page.

__Note:__ The duplication of a variable provokes the creation of a new variable with a new MemoryAddress (see chap. 5, “MemoryAddresses” page 155) with the same value (address) as the MemoryAddress of the original variable.

RefreshGroups

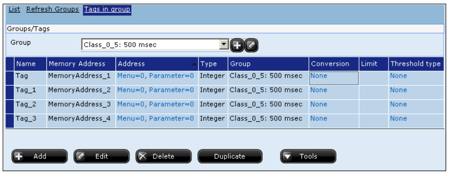


Refresh Groups

Name	Update
Class_0: as fast as possible	0 sec
Class_0_5: 500 msec	0,5 sec
Class_5: 5 sec	5 sec
Class_10: 10 sec	10 sec
RefreshGroup	1 sec

The second window in the "Tags" menu allows to specify the "Gruppi di Rinfresco" (Refresh Groups) present in the project. These classes allow to distinguish the updating frequency of the values of the relative tags. This function is useful when different degrees of mutability are envisioned for field tags. It is possible to introduce, eliminate and duplicate update classes. An identification name and a refresh value indicated in seconds can be inserted for each of these. The "Strumenti" (Tools) key allows to modify the structure of the columns at will.

Tags in the groups



Groups/Tags

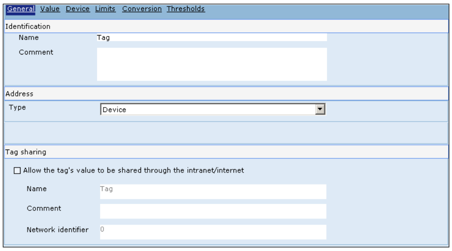
Group: Class_0_5: 500 msec

Name	Memory Address	Address	Type	Group	Conversion	Limit	Threshold type
Tag	MemoryAddress_1	Menu=0, Parameter=0	Integer	Class_0_5: 500 msec	None		None
Tag_1	MemoryAddress_2	Menu=0, Parameter=0	Integer	Class_0_5: 500 msec	None		None
Tag_2	MemoryAddress_3	Menu=0, Parameter=0	Integer	Class_0_5: 500 msec	None		None
Tag_3	MemoryAddress_4	Menu=0, Parameter=0	Integer	Class_0_5: 500 msec	None		None

This window shows the list of project tags and it is possible to modify the relative class associated. It is possible to filter the display of the elements, limiting it just to the tags of the class selected. The distinctive features are supplied for every tag in the list.

Under "Tags" in the tree chart, find the tags just created. Double click on these to enter the editing window of the individual tag.

General



Identification

Name: Tag

Comment:

Address

Type: Device

Tag sharing

□ Allow the tag's value to be shared through the intranet/internet

Name: Tag

Comment:

Network identifier: 0

The editable elements in the General mask are the identifying properties of the variable like name and comment; the name of a variable must be unique, that is other variables cannot exist bearing the same name. The comment is a string (max. 255 characters) that is displayed only within POLYMATH and it identifies the variable.

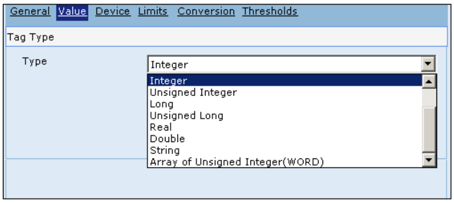
The name of a variable can contain alphanumeric characters (from ‘A’ to ‘Z’, ‘a’ to ‘z’ and from ‘0’ to ‘9’) or the character underscore (‘_’). The maximum length of the string cannot exceed 32 characters and the first digit must be alphabetical.

It is necessary to specify the type of variable you are editing; the variables are divided into: device, internal system and networks variables.

The device variables are shared with the connected equipment and constitute the two-way data exchange medium; the internal variables, by contrast, are used as a deposit for local data or results of operations or Scripts and their value is not read by the device; in this case, it is possible to specify whether the value should be retentive by activating the option appearing in the page when the variable is internal (‘Save the value in a persistent memory. Tags are retentive’). If the variable is retentive, the value is conserved when the terminal is switched off.

The system variables (whose names begin obligatorily with the prefix ‘SYS_’) are variables predefined by POLYMATH that contain special information relating to the working of the project and of the system. They are not editable by the operator but can be displayed and used by the panel. The type of system variable can be selected from the options on a drop-down menu; the characteristics of each variable appear in the lower part of the mask and are also illustrated in the related appendix of this manual (see “Appendix A - System Variables” page 555). The network tags are variables that can be used by all the terminals that make up the network, in the case of a "panel network" project.

Value



Tag Type

Type: Integer

The mask relating to value can be used to configure the type of data that the variable is supposed to contain. The types of data possible are those represented in the following table:

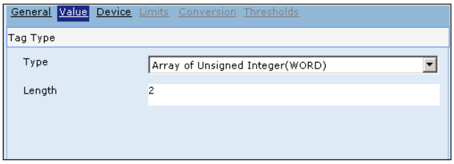
Tabella 1: Types of variable

Type	Description	Range
Char	8-bit signed integer	-127 to 128
Byte	8-bit unsigned integer	0 to 255
Boolean	Single bit	True (1) or False(0)
Integer	16-bit signed integer	-32.768 to 32.767
Unsigned Integer	16-bit unsigned integer	0 to 0xFFFF
Long	32-bit signed integer	-2,147,483,648 to 2,147,483,647
Unsigned Long	32-bit unsigned integer	0 to 0xFFFFFFFF
Real	Floating point IEEE 32-bit single precision	-3.402823E+38 to -1.401298E-45 for negative values; 1.401298E-45 to 3.402823E38 for positive values

Tabella 1: Types of variable

Type	Description	Range
Double	Floating point IEEE 64-bit double precision	-1.7976931348623E308 to -4.9406564584124E-324 for negative values; 4.9406564584124E-324 to 1.7976931348623E308 for positive values
String	UNICODE string in BSTRING format	String of characters in BSTR format (maximum length 0x7FFF characters)
Array of Unsigned Integer (WORD)	Whole value string without sign	1 to 255

For each variable you can introduce an initialization value that is assumed at the start of the project. In the case of a String-type of data, its maximum length can also be indicated.



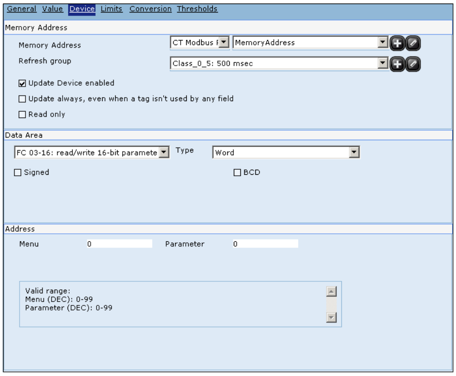
Tag Type

Type: Array of Unsigned Integer(WORD)

Length: 2

If the variable is a device variable, Array-type data will also be present; this in substance is a data area whose dimensions can be set by POLYMATH; as indicated in the figure, a table will also appear which enables you to introduce the initialization values of each portion of the area.

Device



Memory Address

Memory Address: CT Modbus F | MemoryAddress

Refresh group: Class_0_5: 500 msec

☑ Update Device enabled

□ Update always, even when a tag isn't used by any field

□ Read only

Data Area

FC 03-16: read/write 16-bit paramete | Type: Word

□ Signed □ BCD

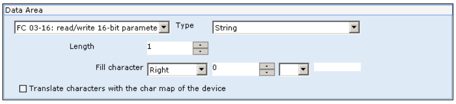
Address

Menu: 0 Parameter: 0

Valid range:
Menu (DEC): 0-99
Parameter (DEC): 0-99

In the case of device variables it is necessary to proceed to specify the destination memory areas for the values. POLYMATH guides the user by furnishing indications regarding the valid memory ranges calculating them automatically in relation to the device chosen in the project.
First of all it is necessary to introduce the destination device, the related memory addresses (see chap. 5, “MemoryAddresses” page 155) and the update class (see chap. 5, “RefreshGroups” page 89). If these last two components are lacking or incorrect, they can be introduced again by clicking on the ✚ icon or they can be edited by clicking on ✎.
In addition, you can decide whether to enable the updated or not.
Even when a tag is not being used by any field, the option ‘Keep updating’, indicates, if it is activated, that the variable will be updated even when its value is not shown in the page currently being displayed on the panel; this option is indispensable whenever, you want, for example, to access the value of this variable via Script. In the event that the variable is part of Alarms, Pipelines, Trends or Recipes this setting is ignored and the variable is monitored all the same.
In conclusion, a one-shot read can be requested when the variable is used in a data field.
It is also necessary to indicate the type of memory to reserve, whether Bit, Byte, Word, DWord or String; if it is not String, you can indicate whether the memory is to be considered as Signed (for relative values) or BCD.

Note: *Binary-coded decimal (BCD) is a commonly used format for representing the decimal digits in binary code. In this format each digit of a number is represented by a 4-bit binary code, whose value is between 0 (0000) and 9 (1001). For example the number 127 is represented in BCD as 0001, 0010, 0111.*



Data Area

FC 03-16: read/write 16-bit paramete ▾ Type String ▾

Length 1

Fill character Right ▾ 0

☐ Translate characters with the char map of the device

If the type of area is String, you can define the length, the type and the gap characters. The types of gap characters available are left, right or none. It is also possible to define the gap character by entering the appropriate ASCII code or choosing an option from the drop-down menu. In both cases, the right-hand side of the mask will display a preview of the gaps.

The lower check box can be used to specify whether to allow the translation of the characters with the map of correspondences present in the device (see chap. 5, “Character translation” page 157).

You can use the lower part of the mask to introduce the destination memory addresses (see chap. 5, “MemoryAddresses” page 155); the values entered must be coherent with the range displayed at the foot of the mask.

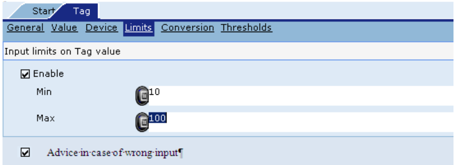
Limits

Validity ranges can be defined for the tag just created (if this restriction has sense in relation to the type of data). It is possible to assign these limits to the values of the tag and/or to the device. If limits are assigned to the Tag (i.e. on the tags of the terminal) the limit will have effect in the editing phase: if, for example, a maximum limit is at 100 and the operator inserts a higher value in an editing field, the field will automatically be taken to 100 (maximum limit). However, this limit does not prevent a greater value being written in the device memory by a device side process.

If this is not the case, by assigning device limits a value will be read on the terminal within the range set also when the tag on the device assumes values outside of the interval.

Once the relative box has been enabled, it is possible to manually insert the limit values or assign them dynamically by combining them with those of tags. This last option can be performed by clicking on and selecting the tag from the drop-down menu that will occur as a consequence. It will always be possible to access the creation-modification of the tags directly from this mask.

Another option that can be found in the "Limiti" (Limits) option is the possibility that the user is warned when an incorrect value is attributed to the Tag. This option is activated
by selecting the "Avvisa se il valore immesso è errato" (Warn if the value introduced is incorrect) box. When the option is activated, a warning will appear under the form of a "Popup" page every time that the value attributed to the Tag is greater or smaller than the previously-set limits.
After having enabled and set the minimum and maximum limit and having selected the "Avvisa se il valore immesso è errato" (Warn if the value introduced is incorrect) option (see following image):



Input limits on Tag value

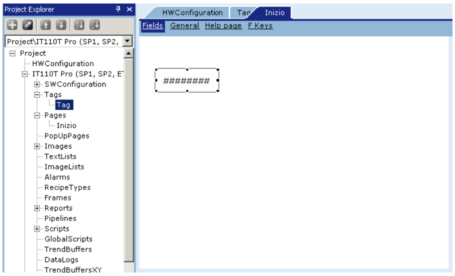
☑ Enable

Min 10

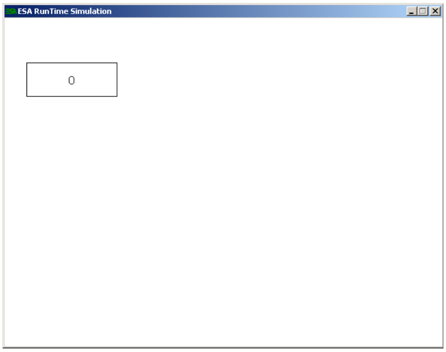
Max 100

☑ Advice·in·case·of·wrong·input¶

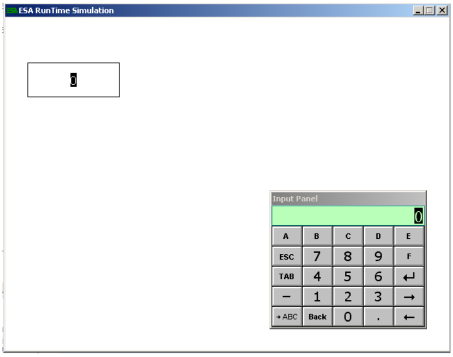
From "Esplora Progetto" (Project Explore) click twice on the main page and then use the mouse to drag the Tag inside the page :



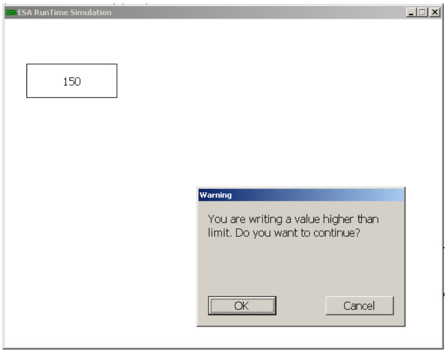
At this point click on the "Esegui Progetto" (Perform Project) icon ▶ ; the following image will appear :



By clicking on the Tag the editing keyboard will appear, from where a value can be assigned to the tag itself :

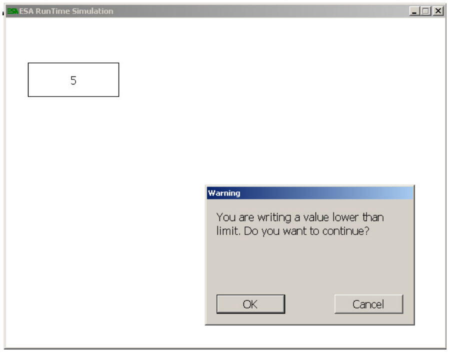


For example, by entering "150" on the editing keyboard and confirming using "Enter" also on the editing keyboard, a warning will automatically appear under the form of a "Popup" page, where the user is warned of the fact that the value being introduced is a higher value that the maximum limit set (which remember is 100) :



You are writing a value higher than limit. Do you want to continue?

At this point the user can decide whether to continue (by clicking on the "OK" key, or to annul the introduction of the data, which must be re-set.
If the user decides to continue by clicking "OK" as just seen, Polymath will automatically attribute the maximum limit value (100).
The same will occur when trying to insert a value below the minimum set, e.g. if a value equal to "5" is set when the minimum limit is "10", a warning message will appear under the form of a "Popup" page, as shown below:

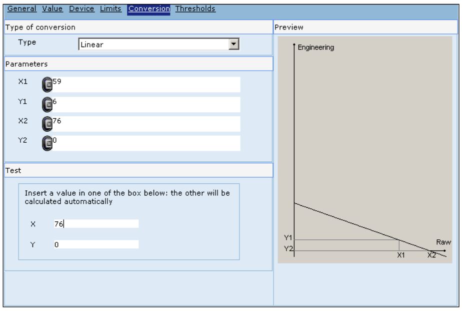


You are writing a value lower than limit. Do you want to continue?

Also in this case, the user can decide whether to continue (by clicking on the "OK" key, or to annul the introduction of the data, which must be re-set).
If the user decides to continue by clicking "OK" as just seen, Polymath will automatically attribute the minimum limit value (10).

Conversion

The value of the numerical external variable is always calculated by the system based on the rough value.
Often, apart from the standard conversions, it is necessary to carry out a calculation, because the units of measurement in which the rough value is expressed are different from those required for the value of the variable.
For example, it occurs very frequently that the rough value is expressed as an integer value within the range of a digital-analog converter, while the value of the variable is expressed in engineering units.
Using this mask you can determine the type of conversion to be adopted for the variable; the conversions that can be selected are: none, linear, quadratic or defined by the user.



Type of conversion

Type: Linear

Parameters

X1: 59
Y1: 6
X2: 76
Y2: 0

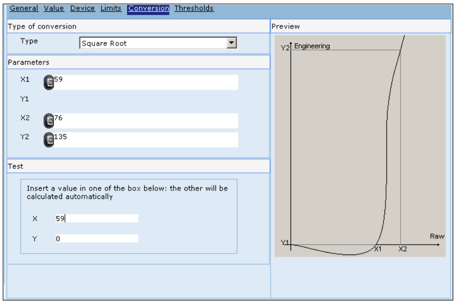
Test

Insert a value in one of the box below: the other will be calculated automatically

X: 76
Y: 0

Linear conversion implies the definition of two pairs of values, each formed of the value of the variable and the corresponding rough value:
P1 (x1, y1)
P2 (x2, y2)
where xn are rough values and yn the corresponding 'engineering'.
The rough value x and the corresponding value y of the variable in the conversion are related by the following equation :

$$\frac{y - y_1}{x - x_1} = \frac{y_2 - y_1}{x_2 - x_1}$$



Type of conversion

Type: Square Root

Parameters

X1: 59

Y1

X2: 76

Y2: 135

Test

Insert a value in one of the box below: the other will be calculated automatically

X: 59

Y: 0

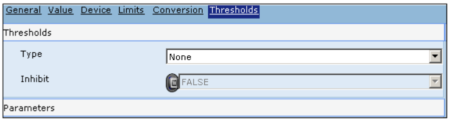
Preview

The quadratic conversion needs the same values with the exception of Y1; in the quadratic transformation the equation that connects the rough value x and the y value of the variable is as follows :

$$\frac{y^2}{x \;\; - x_1} = \frac{{y_2}^2}{x_2 - x_1}$$

In both cases the window situated in the left part of the mask furnishes a graphic representation of how the conversion of the values will take place.
In addition you can carry out an immediate test of the conversion after entering the necessary values; a value can be entered in the appropriate fields and POLYMATH displays its conversion instantly.
The conversion defined by the user envisages the association of a Script with the events that can be associated to the variable (see chap. 6, “Events related to variables” page 163).

Thresholds



Thresholds

Type: None

Inhibit: FALSE

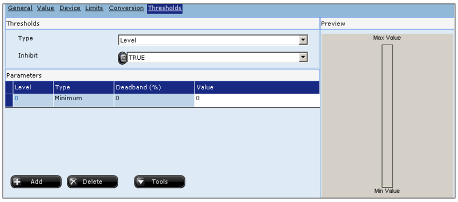
Parameters

The developer can enable the generation of various types of event associated to a numerical variable. The events are generated when the variable assumes particular values (or when there is a rapid change in the value itself), called threshold values or simply thresholds. The user can make use of the defined thresholds by assigning a function or a Script to the threshold events that can then be associated to a variable (see chap. 6, “Events related to variables” page 163).

There are three types of thresholds: level thresholds, deviation thresholds and variation speed thresholds.

In this mask, select the type of threshold required and decide whether to value activating the event should be dynamic or not (Boolean).

The type of threshold is represented graphically to the right of the mask.



Thresholds

Type: Level

Inhibit: TRUE

Parameters

Level	Type	Deadband (%)	Value
0	Minimum	0	0

The first type of threshold is the Level type. Up to eight Level thresholds can be defined, each of which can be enabled independently of the others.

For each of the above mentioned thresholds the developer wants to enable, he/she must specify whether the threshold is a maximum or minimum, independently of which event is generated: thus, if all eight of the possible Level events are generated, eight different thresholds need to be specified. It is also necessary to specify a dead zone value (indicated as a percentage of the reference value), to be used exclusively to check the re-entry of the event (hysteresis). The dead zone indicates a time interval within which the event must not be raised so as to be able to make the slight value oscillations negligible.

Alternatively, the Dead zone and Value attributes can be assigned to another tag simply by clicking on inside the field in question.

The functioning of the Level thresholds can be summed up as follows:

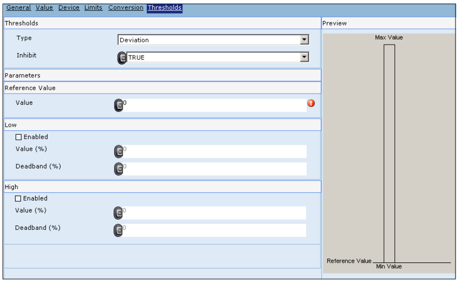
- minimum thresholds: if the value of the variable is falling, the event is activated the moment the variable falls below the reference value; if the value of the variable is rising, the event is activated the moment the variable rises above the reference value increased by the value of the dead zone.
- maximum thresholds: if the value of the variable is falling, the event is activated the moment the variable

falls below the reference value diminished by the value of the dead zone; if the value of the variable is rising, the event is activated the moment the variable rises above the reference value.

Let us consider the example in which we put a reference value of 30 without a dead zone. In this case, if the threshold is defined as a minimum threshold, the events will be activated as soon as the value arrives at the value 30 (when the value is rising) and as soon as it goes below (when the value is falling). If the threshold is defined as a maximum threshold, the events will be activated as soon as the value exceeds the value 30 (when the value is rising) and as soon as it returns to a value equal to or less than 30 (when falling).

If we add to our example a dead zone equal to 10% of the reference value (10% of 30 = 3) the behaviour will be:

if the threshold is defined as a minimum threshold, the events will be activated as soon as the value arrives at the value 30 (when the value is rising) and as soon as it goes below the value 33 (when the value is falling). If the threshold is defined as a maximum threshold, the events will be activated as soon as the value exceeds the value 30 (when the value is rising) and as soon as it returns to a value equal to or less than 27 (when falling).



Thresholds

Type: Deviation

Inhibit: TRUE

Parameters

Reference Value

Value: 0

Low

□ Enabled

Value (%): 0

Deadband (%): 0

High

□ Enabled

Value (%): 0

Deadband (%): 0

The second type of threshold the Deviation threshold; there are two types of this threshold and these can be enabled separately:

- DLO= lower deviation
- DHI= higher deviation

Deviation thresholds are relative to a reference value and indicate how much can be deviated from this value.

For each of the above mentioned thresholds the developer wants to enable, he/she must specify the deviation value (expressed as a percentage of the reference value), independently of which event is generated: thus, if both possible events are enabled, two thresholds different from one another must be defined. The values can be fixed or can refer

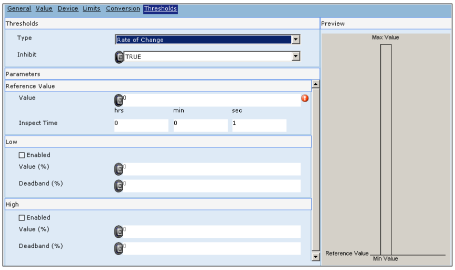
to other tags. There must be a dead zone value for each threshold, expressed as a percentage value of the level referred to; the attributes dead zone and value can be associated to another tag.
Let us now seek to clarify how deviation thresholds work by using an example. To make it easier to understand we will avoid using the dead zone the concept of which has already been expressed in the course of the explanation of the Level threshold concept.
Let us set 30 as a reference value. We shall activate (by clicking on the appropriate box) the low Level threshold, assigning 10% as the value. This means that the event onThdDevLo (see chap. 6, “Events related to variables” page 163) is launched each time there is a breach (whether rising or falling) of the value given by:

- reference value - % low threshold value

In our case, 30 - (10% of 30)= 30 - 3 = 27 thus the event will be activated when the value 27 is crossed.
We operate in the same way to define a high deviation threshold; by clicking on the appropriate box we activate the high-level threshold, assigning 50% as the value. This means that the event onThdDevHi (see chap. 6, “Events related to variables” page 163) is launched each time there is a breach (whether going up or down) of the value given by:

- reference value + % high threshold value

In our case, 30 + (50% of 30)= 30 + 15 = 45 thus the event will be activated when the value 45 is crossed.



Thresholds

Type: Rate of Change
Inhibit: TRUE

Parameters

Reference Value

Value: 0

Inspect Time: hrs 0, min 0, sec 1

Low

□ Enabled
Value (%): 0
Deadband (%): 0

High

□ Enabled
Value (%): 0
Deadband (%): 0

The third group of thresholds is the Variation speed group; the idea is to be able to carry out checks on variation times used by a variable. It may, for example, be useful to manage a situation in which a temperature plunges or soars too rapidly.
We can define two Variation speed thresholds that can be enabled independently of each other:

- RLO= low variation (decrease in the value)
- RHI= high variation (increase in the value)

The Variation speed thresholds are relative to a reference value. It is necessary to specify the time in seconds below which the variation in value should take place such that the event is launched relative to the threshold. Return from the threshold occurs when there is an increase/decrease lower than the value specified within the threshold period.

There must also be a dead zone value for each threshold, expressed as a percentage value of the level referred to; the attributes dead zone and value can be associated to another variable.

Let us now seek to clarify how Variation speed thresholds work by using an example. To make it easier to understand we will avoid using the dead zone, the concept of which has already been expressed in the course of the explanation of the Level threshold concept.

Let us set 30 as a reference value and 5 seconds as the checking time. Clicking on the appropriate box we shall activate the low Level threshold, assigning 10% as the value. This means that the event onThdDevLo (see chap. 6, "Events related to variables" page 163) is launched each time there is a decrease of at least 3 (10% of 30) in an interval of less than 5 seconds. In the same way, the event returns to rest when there is a decrease of less than 3 seconds in a time interval of less than 5 seconds.

Similarly if we set a high threshold: we will leave the reference values unchanged (30) and the checking time remains 5 seconds.

Clicking on the appropriate box we shall activate the high Level threshold, assigning 50% as the value. This means that the event onThdDevHi (see chap. 6, "Events related to variables" page 163) is launched each time the value of the variable increases by at least 15 (50% of 30) in an interval of less than 5 seconds. In the same way, the event returns to rest when there is an increase of less than 15 seconds in an interval of 5 seconds.

Pages

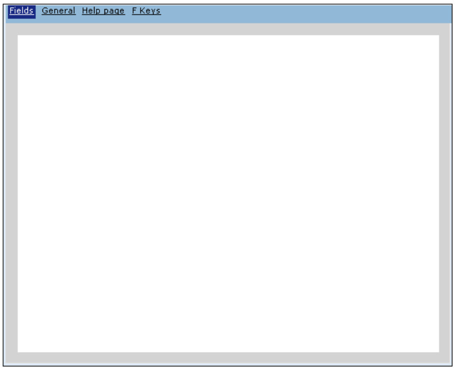
Pages are fundamental for the creation of a project; they are the real interface between operator and terminal. The editing of pages must be based on information accessible to the user and the access policy (restrictions on users) and navigation procedures (links between the pages).

An enormous quantity of objects that can be selected from a list furnished by POLYMATH - which will be described in detail in the next chapter - can be introduced into these pages.

After double-clicking the Pages icon of Project Explorer, the work area will display a list of the pages introduced into the project. Using this list you can introduce new pages and duplicate or delete existing ones. In addition, certain

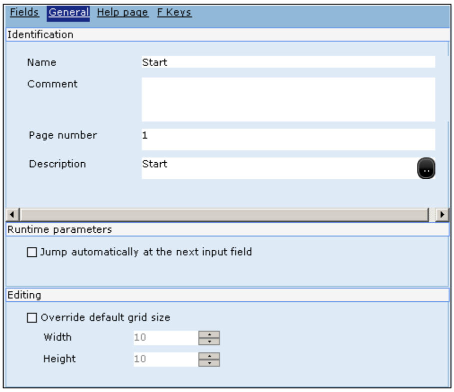
attributes like Page number, Description and Comment can be edited simply by clicking inside the appropriate fields of the table and new texts can be introduced.
Once a page has been created (using Project Explorer or the list), double-clicking on it in the tree-diagram makes it possible to edit it in the work area. The page editor is organized in the following sections: Fields, General, Help page and F keys. The subsections below offer a description for each mask.
The properties and the events that can be assigned to the Page will be dealt with in the next chapter; we advise readers to consult the relevant section for a list of them and their meanings (see chap. 6, “Page properties” page 170 and see chap. 6, “Events related to Pages” page 171).

Fields



The Fields mask shows graphically how the page will appear once the project has been installed in the terminal. To introduce an object simply click on the relevant icon and immediately afterwards draw the outline of the area that will contain it in the page in the desired position.
The next chapter will illustrate all the procedures for introducing graphic objects and the relevant meanings and tools (see chap. 6, “Managing a page” page 167).

General



Identification

Name	Start
Comment	
Page number	1
Description	Start

Runtime parameters

□ Jump automatically at the next input field

Editing

□ Override default grid size

Width	10
Height	10

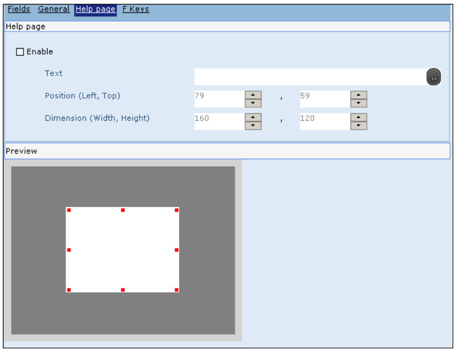
Using this mask you can introduce the identifying attributes of the Page like Name, Comment, Number and Description. The Name, the Number and the Description of the page are unique properties within the project that is there cannot be other different pages that have one of these attributes in common. For this reason, whenever a page is pasted in or duplicated POLYMATH sees to it that these properties are edited so that they satisfy the requirement of uniqueness.
The page number can be a whole number greater than 0, its maximum value depending on the capacity of the terminal's memory; the Comment is a Unicode string whose maximum length is 255 characters and it is visible only within POLYMATH.
The Description is a UNICODE multilingual string with a maximum length of 32 characters.

Warning: *when entering the Description string, be very careful not to introduce control or punctuation characters. Control characters are those between 0x0000 and 0x001F (inclusive) that can be introduced using a keyboard by pressing the sequences ALT+000 up to ALT+031. This rule applies in general for all the objects containing the property Description.*

This mask can also be used to furnish operational settings for the page; you can decide have the cursor jump automatically to the next input field in runtime. If this option is activated, each time the operator enters a value into a numeric field and presses the Enter key, the focus of the application (that is, the

selection) moves to the next field (the order of the selection passage between fields is defined by the TabIndex attribute to be assigned to the page objects, (see chap. 6, “Properties of the Numerical Field” page 216, for example).
It is also possible to overwrite the default grid dimensions for the current page by specifying the desired dimensions. This option is useful when a different degree of precision is required during the editing phase of the page. When low values are entered the grid becomes denser and there is greater freedom in positioning objects within the page; by entering a high value, the grid becomes less dense and the freedom in positioning objects in the page becomes more limited.

Help pages



Help page

□ Enable

Text

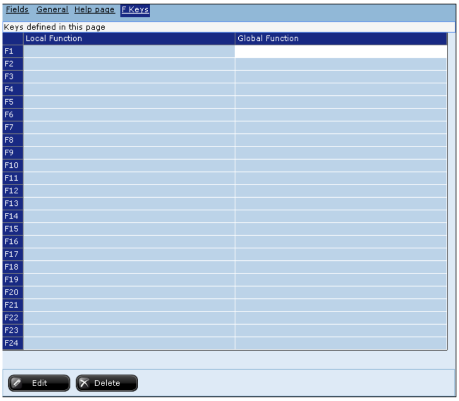
Position (Left, Top) 79 , 59

Dimension (Width, Height) 160 , 120

Preview

A each project page can have a Help page assigned to it, giving information relating to the working of the mother page. The Help page is essentially a window into which a text to guide the operator can be introduced. Apart from the text displayed, other properties like the position and the dimensions of the page can be defined. At the bottom of the mask you are offered a preview of how and where the Help page will appear in runtime. It is also possible to define the font and the dimension of the text of the Help pages during the phase of defining the general properties of the panel (see chap. 5, “” page 74). This page only becomes visible to the operator when it is expressly called (using the button assigned to this function introduced into a mother page, via the command area and function keys).

F keys



Keys defined in this page

	Local Function	Global Function
F1		
F2		
F3		
F4		
F5		
F6		
F7		
F8		
F9		
F10		
F11		
F12		
F13		
F14		
F15		
F16		
F17		
F18		
F19		
F20		
F21		
F22		
F23		
F24		

The last mask available for editing the page is the one relating to the F keys; it is possible to edit the behaviour of a particular function button within the page. Unlike the global keys, the functions set in this page are only effective when they are in the current page.

Warning: *The table present in this mask already indicates the global functions (see chap. 5, "GlobalKeys" page 84), so as to make any overwriting evident. In fact, if they were assigned to the same global and local key functions, only the local ones would carried out in runtime in the context of the page in question.*

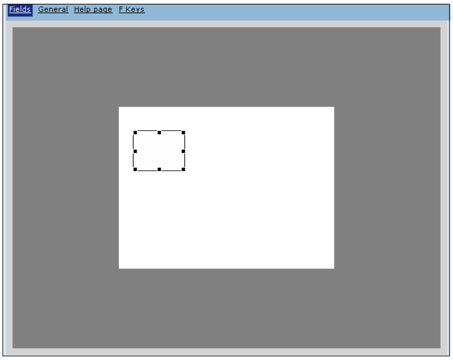
The functions or Scripts that can be associated to the local buttons are introduced in exactly the same way as already seen for the global keys, thus the same procedure should be followed (see chap. 5, "GlobalKeys" page 84).

Popup pages

Popup pages are pages that are only displayed following the occurrence of particular situations (these can be called using the command area and the button with an assigned function). After double-clicking the Popup pages icon in Project Explorer, a list of the pages introduced into the project will appear in the work area. This list can be used to add new Popup pages, duplicate them or delete existing ones. In addition, some attributes like the Page number, Description and Comment can be edited simply by clicking inside the fields relating to the table and new texts can be introduced.
Once the Popup page has been created (using Project Explorer or the list), you can double-click on it in the tree diagram to begin editing it in the work area. The page editor is organized

in the following sections: Fields, General, Help page and F keys as described in the paragraphs below.
For information regarding the properties and events that can be assigned to the Popup pages the reader is advised to read the relevant section in the next chapter (see chap. 6, “Properties of Popup pages” page 171 and see chap. 6, “Events related to Popup pages” page 171).

Fields



The Fields mask for the Popup pages is similar to that relating to the traditional pages (see chap. 5, “Fields” page 104).
The sole difference between the two masks consists in the dimensions of the Popup page. Naturally the Popup page is meant to be smaller than a standard page; it may be positioned in any part of the screen.
To change the dimensions of a Popup page, select it after pressing the key; at this point just take the cursor onto the edges of the page (red outline) to enlarge or reduce its dimensions.
To move a Popup page, select it after pressing the key; then simply drag it to the position you want it to appear in runtime.
The next chapter illustrates all the procedures for introducing graphic objects and their related meanings and tools (see chap. 6, “Managing a page” page 167).

General

The General mask for the Popup pages is identical to that relating to the traditional pages; thus, readers are advised to

consult the paragraph dealing with these (see chap. 5, "General" page 105) for details of the properties.
The sole difference consists in the possibility of expressing a preference in runtime: you can choose whether to display the page title bar or whether the Popup page should always appear in the foreground.

Help page

The Help page mask for the Popup pages is identical to that relating to the traditional pages; thus, readers are advised to consult the paragraph dealing with these (see chap. 5, "Help pages" page 106) for details of the properties.

F keys

The F keys mask for the Popup pages is identical to that relating to the traditional pages; thus, readers are advised to consult the paragraph dealing with these (see chap. 5, "F keys" page 106) for details of the properties.

Images

POLYMATH offers the possibility of importing into the project images that are in the programmer's PC; images in all the more common graphic formats can be introduced.
By double-clicking on the Images icon in Project Explorer, the list of images uploaded into the project can be accessed. Using this list you can see a preview of the figures, add them, duplicate and delete them. In addition this window makes information available regarding the dimensions (in pixels) of the image, its format and quality; it is also possible to edit the Comment relating to each figure.
When a new project is created POLYMATH introduces some images intended for specific uses as a default (display of alarms, Trend Pen, etc.). These can also be used inside for other purposes.
For the description of how to introduce and change images within a page you are advised to read the following chapter (see chap. 6, "Image Field" page 194). Below we describe the procedure for adding an image to a project.

Add an image

To add an image in POLYMATH you can operate directly on the image object (by clicking on Add) or using Project Explorer (by right-clicking on Images, then Add).
In both cases the Image mask is accessed (described in the next section); to browse the contents of your PC just click low down on the mask ('press here to upload a file'). At this point a window appears and this is used to add one of the personal

images that can be edited with the normal commands contained in the Image and General masks that we are about to describe in detail.
It is possible to insert images with extension type DWG or DXF type even if they are not available in the files list.
To insert such images in the project, select the image, select the type of file such as All files and open it.
Polymath will automatically convert the image in BPM, ready then to be used in the project.

Image mask

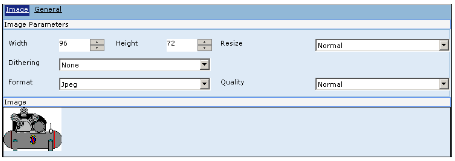


Image Parameters

Width 96 Height 72 Resize Normal

Dithering None

Format Jpeg Quality Normal

Image

This mask is used to edit the parameters of the image. Each time one of the properties of the image is changed, the changes made are immediately visible in the preview box at the foot of the mask.
Firstly, you can set the dimensions in pixels (width and height) of the image contained in the project. Should the original dimensions of the imported image be varied it will also be necessary to define how POLYMATH will have to effect the change in size (calculating the addition or removal of pixels).
The options available are:

- Normal
- Resample (also called Bilinear), a faster and less precise algorithm recommended for reducing images
- Bicubic, a more precise algorithm recommended for enlarging images

in addition you can define the type of filter for the images that contain colours not supported by the panel thanks to the dithering technique (substitution of pixels with colours not available with the interpolation method); you can choose from a list of the more common types of dithering algorithm the one you wish to use:

- None
- FloydStenberg
- Stucki
- Burkes
- Sierra
- StevensonArce

- Jarvis
- Ordered
- Clustered

Note: *For more details regarding the special characteristics of each dithering algorithm, the reader is advised to consult manuals specializing in digital graphics.*

Finally you have to specify the format in which the image is to be saved within the project (Bitmap or Jpeg); if the Jpeg format is chosen, the level of quality-compression desired will also have to be defined by choosing between the levels offered:

- Excellent quality
- Good quality
- Normal
- String compression
- High compression

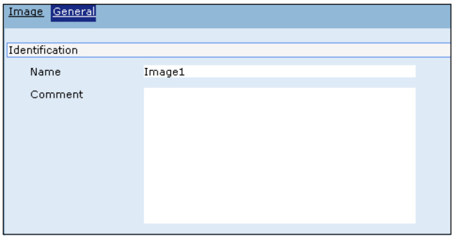
Operations performable on an image

When you are inside Image mask of an image, POLYMATH activates a series of icons for graphic purposes that are applicable to the image in question. These features are accessible via the toolbar or the image submenu of the main menu. To edit it is possible to use the image editor of the window otherwise Below we set out a list of POLYMATH utilities related to images:

- Load image: reached via icon or main menu (Image->Load image). Allows a new image present in your PC to be loaded (you can introduce images in the more common formats)
- "Modify": can be reached via the icon or by main menu (Image->Modify). Allows the image to be modified.
- Restore image: reached via icon or main menu (Image->Restore image). Undoes all changes made to the base image.
- Colour: reached via icon or main menu (Image->Colour). Allows the type of image colour to be selected from the following 3 types, namely Automatic, Grey tones or Black and white.
- Increase contrast: reached via icon or main menu (Image->Increase contrast). Increases the contrast of the image being edited.

- Decrease contrast: reached via icon or main menu (Image->Decrease contrast). Reduces the contrast of the image being edited.
- Increase brightness: reached via icon or main menu (Image->Increase brightness). Increases the brightness of the image being edited.
- Decrease brightness: reached via icon or main menu (Image->Decrease brightness). Reduces the brightness of the image being edited.
- Cut Area: reached via icon or main menu (Image->Cut Area). If this icon is pressed it will be possible to cut (and make visible) a portion of the imported image.
- Rotate: reached via icon or main menu (Image->Rotate). This function makes it possible to rotate the image anticlockwise; with each rotation POLYMATH automatically updates the Height and Width dimensions inverting them.
- Adapt to screen: reached via icon or main menu (Image->Adapt to screen). If this icon is pressed the image is adapted so that it occupies the work screen completely (in practice its dimensions coincide with the maximum screen dimensions of the VT).
- Maintain proportions: reached via icon or main menu (Image->Maintain proportions). If this icon is pressed the proportions of the original image are maintained, that is, to change the Height of the image, POLYMATH updates the Width and vice versa.

General



Identification

Name: Image1

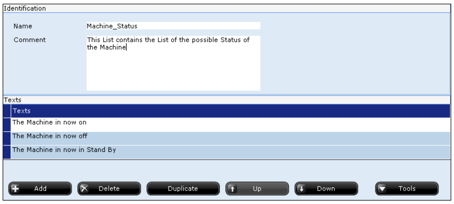
Comment:

The General mask can be used to set the identifying properties of the image. The Name of a image can contain alphanumeric characters (from 'A' to 'Z', 'a' to 'z' and from '0' to '9') or the character underscore ('_'). The maximum length of the string cannot exceed 32 characters and the first digit must be alphabetical.

The Name of an image is a unique attribute within any given project that is other different images with the same name cannot exist.
The Comment is a Unicode string with a maximum length of 255 characters and is visible only within POLYMATH.

Text list

In POLYMATH there are objects whose purpose it is to be text containers useful for creating value fields (see chap. 6, "Value fields" page 197). Each text list can contain an indefinite number of texts; the sole limits are those deriving from the Hardware configuration of the panel.
When you double-click on the Text list icon in Project Explorer the causes the table of text lists to appear in the work area; this list can be used to introduce, duplicate and delete the text lists or simply introduce or edit a related comment.
Once a Text list has been created, it can be double-clicked in Project Explorer to access the corresponding editing mask



Identification

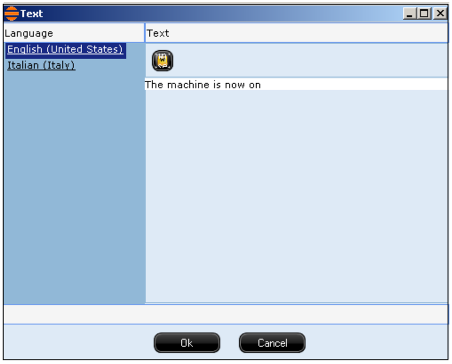
Name: Machine_Status

Comment: This List contains the List of the possible Status of the Machine

Texts

Texts
The Machine in now on
The Machine in now off
The Machine in now in Stand By

The upper part of the mask can be used to change the identifying properties of the list. The name of a list can contain alphanumeric characters (from 'A' to 'Z', 'a' to 'z' and from '0' to '9') or the character underscore ('_'). The maximum length of the string cannot exceed 32 characters and the first digit must be alphabetical.
The Name of a list is a unique attribute within any given project that is other different lists with the same name cannot exist.
The Comment is a Unicode string with a maximum length of 255 characters and is visible only within POLYMATH.
The lower part of the mask can be used to edit the text list itself; new texts can be added or existing ones deleted. To move a text just select it and click on the Up or Down keys according to the operation to be performed.
If there is more than one language in a project (see chap. 5, "Languages" page 78), you can go on to define the translation for each text in the list as shown in the following figure :

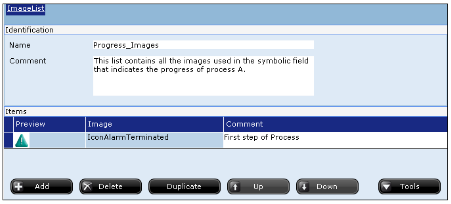


Language	Text
English (United States)	The machine is now on
Italian (Italy)	

The ▣ key allows to insert symbols into the description.

Image list

In POLYMATH there are objects whose purpose it is to be text containers useful for creating value fields (see chap. 6, “Value fields” page 197). Each Image list can contain an indefinite number of texts; the sole limits are those deriving from the Hardware configuration of the panel.
When you double-click on the Image list icon in Project Explorer the causes the table of Image lists to appear in the work area; this list can be used to introduce, duplicate and delete the text lists or simply introduce or edit a related comment.
Once an Image list has been created, it can be double-clicked in Project Explorer to access the corresponding editing mask.



Identification

Name: Progress_Images

Comment: This list contains all the images used in the symbolic field that indicates the progress of process A.

Items

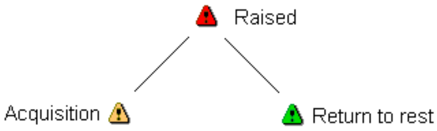
Preview	Image	Comment
⚠	IconAlarmTerminated	First step of Process

The upper part of the mask can be used to change the identifying properties of the list. The name of a list can contain alphanumeric characters (from ‘A’ to ‘Z’, ‘a’ to ‘z’ and from ‘0’ to ‘9’) or the character underscore (‘_’). The

maximum length of the string cannot exceed 32 characters and the first digit must be alphabetical.
The Name is a unique attribute within any given project that is other different lists with the same name cannot exist.
The Comment is a Unicode string with a maximum length of 255 characters and is visible only within POLYMATH.
The lower part of the mask can be used to edit the Image list itself; new images can be added or existing ones deleted. Add an image already in the project (see chap. 5, "Add an image" page 109) by using the relevant drop-down menu Image column. A preview and a comment can be displayed for each image belonging to the list.
To move an image just select it and click on the Up or Down keys according to the operation to be performed.

Alarms

Alarms are events that need immediate attention on the part of the operator; they are connected to signal anomalous conditions with respect to the plant or the terminal. The alarms usually have associated to them particular events of the following type :



- raised: the alarm condition is signalled on the device
- return to rest: refers to the end of the alarm state on the device
- acquisition: (often also identified as 'ack' - acknowledgement) an operator has recognized the alarm condition.

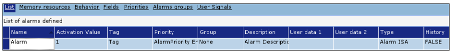
Using Project Explorer, double-click on Alarms to access the general setting windows of all the alarms. The masks available in this area are:

- List
- Memory resources
- Behaviour
- Fields
- Priority
- Alarm groups
- User signals

The next subsections will give a detailed account of the features accessible via each mask.

For a more thorough knowledge of the list and the meaning of the events that can be associated to an alarm the reader is advised to consult the next chapter (see chap. 6, "Events related to alarms" page 164) where there are also illustrations of the complex fields for displaying and managing alarms (see chap. 6, "Complex Controls" page 268).

List

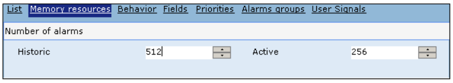


List of alarms defined

Name	Activation Value	Tag	Priority	Group	Description	User data 1	User data 2	Type	History
Alarm	1	Tag	AlarmPriority Er	None	Alarm Descriptic			Alarm ISA	FALSE

The Alarm list allows you to manage the table of alarms and their related properties; a summary of all the standard properties in editable fields is supplied. This mask is useful for giving an overall vision of all the alarms present in the project. New alarms can be added, or cancelled and those existing edited by means of the relevant buttons at the bottom of the mask.

Memory resources



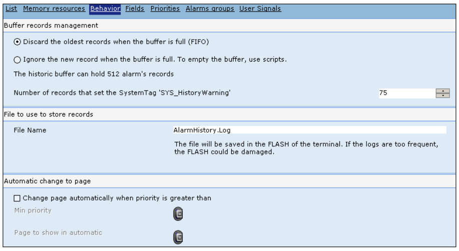
Number of alarms

Historic 512

Active 256

Use the Memory resources mask to define how much memory to reserve in the terminal for the management of the alarms; it is necessary to specify how many alarms can be managed by the history and how many active alarms to consider.

Behaviour



Buffer records management

● Discard the oldest records when the buffer is full (FIFO)

○ Ignore the new record when the buffer is full. To empty the buffer, use scripts.

The historic buffer can hold 512 alarm's records

Number of records that set the SystemTag 'SYS_HistoryWarning' 75

File to use to store records

File Name AlarmHistory.Log

The file will be saved in the FLASH of the terminal. If the logs are too frequent, the FLASH could be damaged.

Automatic change to page

□ Change page automatically when priority is greater than

Min priority

Page to show in automatic

Use the Behaviour mask to indicate the filling and emptying policy of the buffer when it has reached its maximum value.

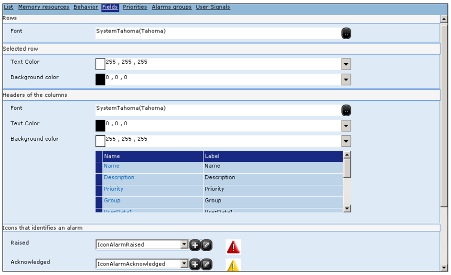
You can choose to substitute the least recent element (FIFO buffer) or ignore the new elements when the buffer is full. (The buffer can be emptied in runtime using a Script, a button or a command area.) You can also decide on the limit of alarms present in the history above which the system variable ‘SYS_HistoryWarning‘ will be activated (see “Appendix A - System Variables” page 555).
The name of the file in which the Alarm History is to be saved must be entered into the text field in the mask; the log file is saved in the folder \log (see chap. 8, “Transferring data” page 362).

Warning: *When entering file names, care must be taken that they are admissible names for a Windows environment. A file name, to be admissible, cannot contain the following characters \ / : * ? “ < > |*

Note: *The log file is a file used by the system to permanently save the data to be represented in the Alarm History. Being able to choose its name using POLYMATH is useful in that it allows the user to manage this file (e.g. copy or delete in the event that it is too big). Should you want to re-arrange the data in an Alarm History, however, it will have to be exported using a predefined function or Script (see “Appendix B - Predefined functions” page 563 and see chap. 9, “Scripts” page 379).*

If you decide that a Page should change automatically when an alarm is raised that is of higher priority than a threshold (specified using a drop-down menu) (see chap. 5, “Priorities” page 119), indicate which Page to display when the alarm is raised.

Fields



Rows

Font: SystemTahoma(Tahoma)

Selected row

Text Color: 255 , 255 , 255

Background color: 0 , 0 , 0

Headers of the columns

Font: SystemTahoma(Tahoma)

Text Color: 0 , 0 , 0

Background color: 255 , 255 , 255

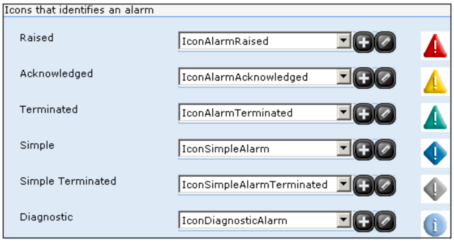
Name	Label
Name	Name
Description	Description
Priority	Priority
Group	Group
UserData1	UserData1

Icons that identifies an alarm

Raised: IconAlarmRaised

Acknowledged: IconAlarmAcknowledged

Use the Fields mask to define the character and the colours of the display tables of the alarms (see chap. 6, “Active Alarm View” page 298 and see chap. 6, “Alarm History View” page 302). You can specify the character of the rows, the colours of the cells selected, the characters and colours of the column headers. The way the editing fields relating to the fonts and colours work is identical to what has already been set out for the user table (see chap. 5, “Fields grid” page 83). The programmer can also make a choice in relation to labels to be assigned to the table headings. Each column can have associated to it a multilingual label; to access mutilingual editing just click on the .. icon adjacent to the editable text field (see chap. 5, “Languages” page 78).


Icons that identifies an alarm

Raised: IconAlarmRaised

Acknowledged: IconAlarmAcknowledged

Terminated: IconAlarmTerminated

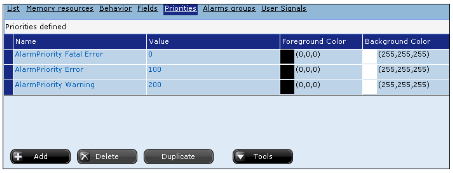
Simple: IconSimpleAlarm

Simple Terminated: IconSimpleAlarmTerminated

Diagnostic: IconDiagnosticAlarm

There is also the option of selecting the images to be associated to the state of the alarm; already at the project creation stage POLYMATH furnishes a set of default images that can be confirmed or substituted with an image added to the project (see chap. 5, “Add an image” page 109).
The alarm states to which an image should be attributed are: Active, Recognised, Returned, Simple and Diagnostic.

Priorities

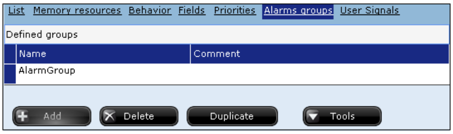


Priorities defined

Name	Value	Foreground Color	Background Color
AlarmPriority Fatal Error	0	■ (0,0,0)	□ (255,255,255)
AlarmPriority Error	100	■ (0,0,0)	□ (255,255,255)
AlarmPriority Warning	200	■ (0,0,0)	□ (255,255,255)

The Priorities mask gives you the possibility of managing the set of properties that can be assigned to an alarm. As default POLYMATH offers three priority levels to each of which there is a corresponding value: Advice (200), Error (100) and Fatal Error (0). The 'Add' and 'Delete' keys respectively allow you to add priority levels to and remove them from the list; the three initial levels predefined by POLYMATH cannot be removed. When a new priority is added, it has to be assigned a priority value that permits it to be classified relative to the other already existing levels. For example, if you wish to introduce a priority of a level lower than the three predefined ones, we will need to assign a value of 201 or above; if you wish to introduce a high priority, give a value between 1 and 99 (the predefined Fatal Error level is always the one with the highest priority).
You can distinguish the priority of the alarms in runtime by assigning them different colours in the Table of active alarms or in the history (see chap. 6, "Active Alarm View" page 298 and see chap. 6, "Alarm History View" page 302). Use this mask to indicate the background colour (with the RGB code or a palette) and text of the non-selected options in the table (otherwise the colours are those in the Fields mask).

Alarmgroups

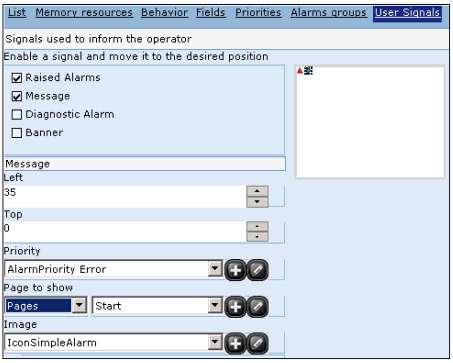


Defined groups

Name	Comment
AlarmGroup	

POLYMATH offers the possibility of organizing the alarms of a given project into Alarm groups; this could be useful where a considerable quantity of alarms is envisaged and the programmer wants to have at his/her disposal a cataloguing tool (for example, to speed up the acquisition of many a alarms at the same time). Using this mask new groups can be created by clicking on ‘Add’ or existing ones deleted by clicking on ‘Delete’); in addition, for each group a comment with a purely identificatory purpose for the programmer can be introduced that will be visible only within POLYMATH.

Usersignals



Signals used to inform the operator

Enable a signal and move it to the desired position

☑ Raised Alarms
☑ Message
☐ Diagnostic Alarm
☐ Banner

Message

Left: 35

Top: 0

Priority: AlarmPriority Error

Page to show: Pages | Start

Image: IconSimpleAlarm

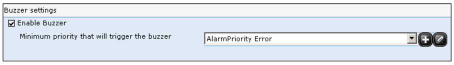
Use this mask to set the alarm signals that appear to the operator. The types of alarm messages displayed are:

- Raised alarms
- Simple messages
- Diagnostic alarms
- Banners

Once the type of alarm message has been selected (by clicking on the appropriate box), it is displayed in the preview page in the right-hand section of the mask. After clicking on the element introduced, it can be moved to the position you want the message to appear in.

For the first three types of messages the following must be defined for the appropriate icon to be displayed: a minimum level of priority; a destination page when the icon itself is pressed; and the image to be presented on screen (which can be selected from among those in the project).

If, on the other hand, the type of message is Banner, a background and text colour need to be defined as well as a rotation time expressed in seconds in case there should be more than one alarm/message.



Buzzer settings

☑ Enable Buzzer

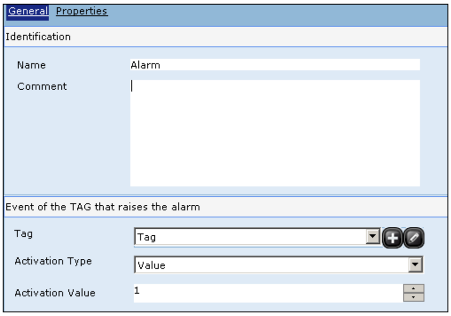
Minimum priority that will trigger the buzzer: AlarmPriority Error

At the bottom of the mask you can enable the reproduction of an alarm sound (‘Enable tone’); if this function is enabled, it will be necessary also to define a minimum level of priority of the alarm so that it triggers the reproduction of the tone (in the terminals with this feature).

Creating and changing an alarm

Once the general characteristics of the alarms have been defined within the project, you can begin to define the way the individual alarms should work. An alarm can be created directly from the alarm list (see chap. 5, “List” page 116) or using Project Explorer (click with the right-hand key on Alarms and then on Add).
In the editing phase, two masks, General and Property - that we shall go on to describe in detail below - are presented for each alarm in the project.

General



Identification

Name: Alarm
Comment:

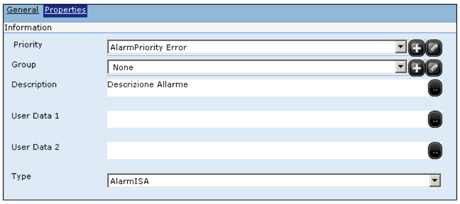
Event of the TAG that raises the alarm

Tag: Tag
Activation Type: Value
Activation Value: 1

The General mask can be used to set the identifying properties of the alarm like Name and Comment. The name is a unique attribute within any given project that is other different alarms with the same name cannot exist.
The name of an alarm can contain alphanumeric characters (from ‘A’ to ‘Z’, ‘a’ to ‘z’ and from ‘0’ to ‘9’) or the character underscore (‘_’). The maximum length of the string cannot exceed 32 characters and the first digit must be alphabetical.
The Comment is a Unicode string with a maximum length of 255 characters and is visible only within POLYMATH.
A variable must be assigned on which the checks relating to the alarm will be carried out; depending on the type of variable (see chap. 5, “Value” page 91) there will be different modes of checking which may be orientated to the bit or value of the variable. The text box asks you to enter the reference

value which when reached will generate the alarm (absolute value or bit number).

Properties



Priority	AlarmPriority Error
Group	None
Description	Descrizione Allarme
User Data 1	
User Data 2	
Type	AlarmISA

The specific characteristics of the individual alarm are defined in the Property mask.

First of all the programmer is asked to enter a membership group for the alarm (see chap. 5, "Alarmgroups" page 119) and a description representing the actual text that the operator will read on the panel when the alarm is triggered.

The description is a multilingual Unicode string (see chap. 5, "Languages" page 78) that cannot contain punctuation or control characters (da Alt+000 a Alt+031) and which cannot exceed 255 characters in length.

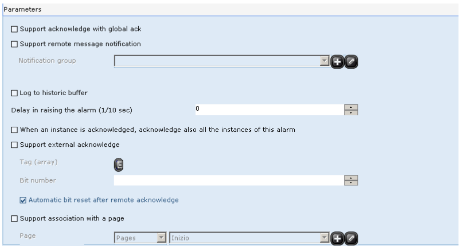
The attributes Datiuser 1 and Datiuser 2 are optional attributes indicating identifying multilingual strings of the alarm. The user can choose whether to employ them for personal purposes or to leave them unused. When they are used they appear in runtime within the alarm views if the appropriate column is present in the attribute Columns (see chap. 6, "Properties of the Active Alarm Grid" page 300 and see chap. 6, "Properties of the Alarm History Grid" page 304).

Note: *For the current value of a variable (for example one assigned to an alarm) to appear in the Description, Datiuser1 or Datiuser2 strings just put into the string the name of the variable in a sequence having the form* %{<name of the TAG>#<format>}%. the format follows ANSI-C specifications. For example, a Description containing the string "excessive temperature: %{TFORNO#%03d}%°C"; if the variable TFORNO has a value of 150, it will be displayed as: "excessive temperature: 150°C" .*

The Property mask also asks you to specify the type of alarm; the following table explains the types of alarm available.

Tabella 2: Types of alarm

Event type	Description
Simple event	simple event; this is not an alarm but an information message
ISA alarm	alarm event (requires acknowledgement on the part of an operator – triggers an ISA sequence)



Parameters

☐ Support acknowledge with global ack

☐ Support remote message notification

Notification group

☐ Log to historic buffer

Delay in raising the alarm (1/10 sec) 0

☐ When an instance is acknowledged, acknowledge also all the instances of this alarm

☐ Support external acknowledge

Tag (array)

Bit number

☑ Automatic bit reset after remote acknowledge

☐ Support association with a page

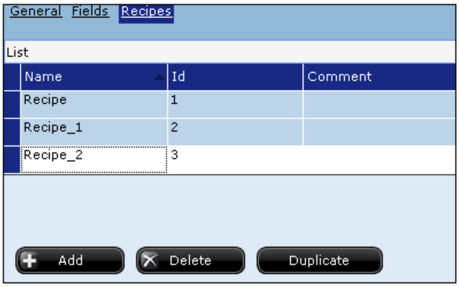
Page: Pages, Inizio

The lower part of the mask is used to set a series of parameters relating to the behaviour of the alarm.
It is possible to decide:

- to permit acknowledgement via global (cumulative)acquisition
- to include the alarm in the Alarm History
- to attribute a lag (in seconds) before the alarm is signalled to the user (in the tables or by means of messages). If the alarm is terminated within this interval it is not signalled.
- that acknowledgement of an alarm instance should provoke the acquisition of all the instances of this type of alarm
- to enable external acquisition via a project variable If so, it will be necessary to define a reference variable and the bit value to be checked. You can choose to have the bit reset automatically after its remote acquisition
- to assign to a page; if this preference is enabled, it will be necessary to define a reference page be assigned. To be able to exploit this function you will need to introduce the ‘Shows page’ button in the Alarm display tables (see

chap. 6, “Active Alarm View” page 298 and see chap. 6, “Alarm History View” page 302).

Recipes



Name	Id	Comment
Recipe	1	
Recipe_1	2	
Recipe_2	3	

Recipes are a means of creating the setup of the plant or part of it) to carry out a given process.
This result can be obtained by writing appropriate values into a certain number of variables, typically set-points or regulating parameters and PLC memory cells.
POLYMATH allows you to define a number of types of recipes, that is, general data structures whose instances the operator will proceed to furnish in line with his needs; there are no limits to the number of types of recipes that the programmer can define using POLYMATH. The only limits my depend on the Hardware characteristics of the terminal.

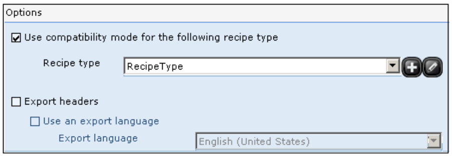
__Warning:__ *POLYMATH makes it possible to define types of recipes, that is, different structures identified by name and by the related variables; the recipes are created and managed in runtime and saved into the retentive memory of the panel. The types of recipes describe only the structure which all the recipes belonging to that type have.*

For further information regarding the list and the meanings of the events that can be associated to a Recipe type, the reader is advised to consult the next chapter (see chap. 6, “Events related to Recipes” page 165) where there is also a description of the display modes of the recipes using complex controls (see chap. 6, “Complex Controls” page 268) as well as the meanings of the transfer operations between the VT and the devices (see chap. 6, “” page 314).

__Recipe list__

After double-clicking on the Recipe element of Project Explorer, you access the list of types of recipes present in the project. Use this list to add new types (by clicking on the 'Add' key), duplicate (with the 'Duplicate' key) or delete (with 'Delete') existing ones.
For each recipe type the summary of the related characteristics is shown in editable fields. This mask is useful for gaining a complete view of all the recipes present in the project.

Modes of compatibility



Options

☑ Use compatibility mode for the following recipe type

Recipe type: RecipeType

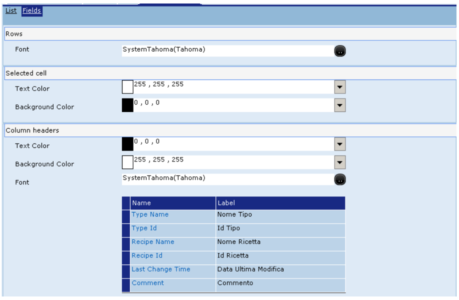
☐ Export headers

☐ Use an export language

Export language: English (United States)

Using the Recipe list mask you can specify for which type of recipe present in the project the mode of compatibility should be enabled (this option is applicable to one and only one Recipe type). By compatibility we mean a use of the exchange areas identical to the Mode of functioning of VTWIN-programmable ESA terminals . A compatible structure uses the command area of the project (see chap. 5, "Exchange areas" page 76) and accepts commands from the PLC only with a Recipe name not over 4 characters. On the other hand, a non compatible structure uses dedicated exchange areas (in this case the recipe can have a longer name).

Fields



Rows

Font: SystemTahoma(Tahoma)

Selected cell

Text Color: 255 , 255 , 255

Background Color: 0 , 0 , 0

Column headers

Text Color: 0 , 0 , 0

Background Color: 255 , 255 , 255

Font: SystemTahoma(Tahoma)

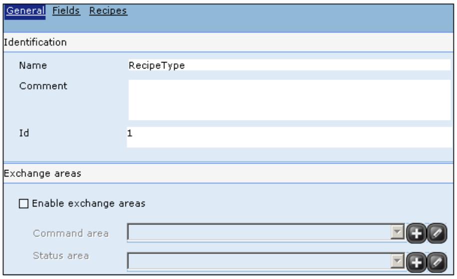
Name	Label
Type Name	Nome Tipo
Type Id	Id Tipo
Recipe Name	Nome Ricetta
Recipe Id	Id Ricetta
Last Change Time	Data Ultima Modifica
Comment	Commento

The Fields mask is used to define the character and colours of the Recipe display tables (see chap. 6, “Recipe List Table” page 307 and see chap. 6, “Recipe Editing Table” page 310). Here you can specify the character of the rows, the colours of the selected cells, the characters and the colours of the column headings. The way the editing fields relating to the font and the colours is identical to what has already been indicated for the User table (see chap. 5, “Fields grid” page 83).
It is up to the programmer to choose which labels to assign to the Table headings. Each column can have a multilingual label assigned to it. To access Multilanguage editing just click on the ● icon adjacent to the editable text field (active only if more than one language coexists in the project).

Creating and changing a Recipe type

Once the general characteristics of the recipes in the project have been established, you can start defining the actual characteristics of each Recipe type. A Recipe type can be created directly from the list of Recipe types (see chap. 5, “Recipe list” page 124) or using Project Explorer (click with right key on Recipes types and then on Add).
For each Recipe in the project, there are two editing masks, General and Property, which we shall describe in detail below.

General



Name: RecipeType

Comment:

Id: 1

□ Enable exchange areas

The General mask is used to define the identifying properties of a Recipe Type. The name of a recipe can contain alphanumeric characters (from ‘A’ to ‘Z’, ‘a’ to ‘z’ and from ‘0’ to ‘9’) or the character underscore (‘_’). The maximum length of the string cannot exceed 32 characters and the first digit must be alphabetical.

The Recipe Type ID is an identifying number within the data structure of the project; it is a whole number greater than zero.

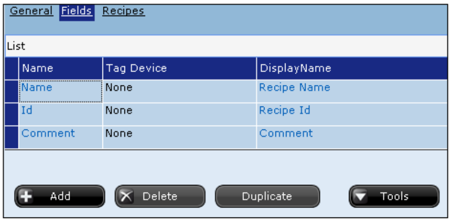
The Recipe type name and ID are unique attributes within the project that is other different Recipe types with the same name cannot exist.

The Comment is a Unicode string with a maximum length of 255 characters and is visible only within POLYMATH.

If the recipe being edited is not defined in a compatible mode, you can use the bottom of the mask to choose whether to enable the dedicated exchange areas for the recipe in question. If this option is activated it will also be necessary to indicate the command and status areas linked to the recipe (see chap. 5, “Exchange areas” page 76).

If, on the other hand, the recipe has been defined in a compatible mode, it will be possible to define the status area used (see chap. 5, “Exchange areas” page 76).

Recipe type fields



List

Name	Tag Device	DisplayName
Name	None	Recipe Name
Id	None	Recipe Id
Comment	None	Comment

The real structure of the Recipe type must be indicated in the Fields mask. Each recipe in the terminal must have the fields Name, ID and Comment while other fields can be introduced by the programmer. It is precisely the fields introduced by the operator that are the distinctive elements of each Recipe type. By clicking on the 'Add' key it is possible to introduce a new field to the Recipe type. After having clicked on the ✚ key, variables already present in the project or new variables can be assigned to the new field using the column relating to the variable by clicking on the field introduced. It is also possible to access the editor of the variable selected after clicking on ✎.
To remove a field in the Recipe type, simply select it and click on 'Delete'.

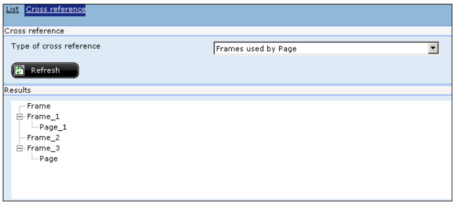
Frames

The purpose of the Frames is to edit synoptic diagrams parts to be used in more than one page. For example, if the project is supposed to contain twenty pages and ten of these have the same group of element (e.g. two numeric fields with a button), then simply define this portion once inside a frame and fetch it onto each page. Once a frame has been defined it can be introduced into a page simply by dragging it there (from Project Explorer to the page in the work area).
To learn more about the List and the meaning of the properties that can be assigned to the Frames, the reader is advised to consult the relevant part of the next chapter (see chap. 6, "Properties of Frames" page 171).

List

Double-clicking on the Frames option from within Project Explorer gives access to the list of Frames present in the project; this mask contains the name and the comment relating to each Frame and it is possible to introduce new Frames or delete or duplicate existing ones.

Cross references



Cross reference

Type of cross reference: Frames used by Page

Results

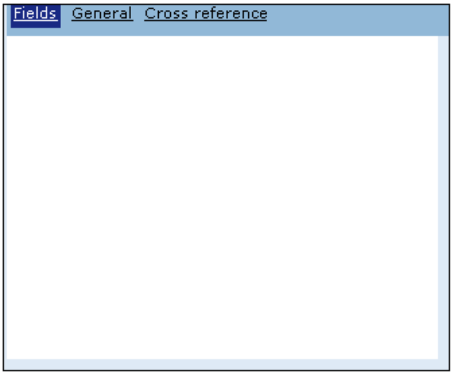
```
Frame
Frame_1
   Page_1
Frame_2
Frame_3
   Page
```

The mask relating to cross references allows you to see how the frames are used within the project; you can set the mask for displaying the list of pages using at least one frame or, as an alternative, the frames used by the pages. In both cases the results are displayed in a tree-diagram.
When the 'Update' key is pressed, POLYMATH recalculates the references to the Frames in real time.

Creating and managing Frames

A Frame can be created either by using the appropriate list (see chap. 5, "List" page 128) or directly by using Project Explorer (press right key on Frames, then Add).
Once a Frame has been added by double-clicking on it in Project Explorer Work area, in you access the editor subdivided into three pages: Fields, General and Cross References which will be dealt with in the following sections.

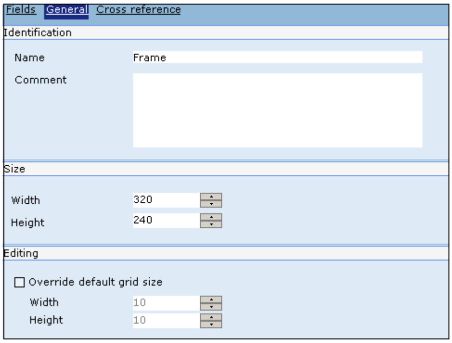
Fields



Using this mask you can edit the way a Frame will actually appear in the pages into which it is called; it is edited just like that for normal pages with various objects being introduced

and properties being set (see chap. 6, “Managing a page” page 167).
To introduce an object simply click on the respective icon and immediately after draw where in the page you wish the outline of the area to contain it to be placed.
The next chapter describes all the procedures for introducing the graphic objects together with their related meanings and tools.
Using this mask you can, however, set the dimensions of the Frame: click on the ◰ icon to select it and then move the cursor to one of the red corners by dragging it in line with the dimensions required. (This operation can also be performed by the General mask as set out in the next section).
You cannot use this mask to move the Frame, in that its final position is defined periodically in the destination page.

General



Identification

Name	Frame
Comment	

Size

Width	320
Height	240

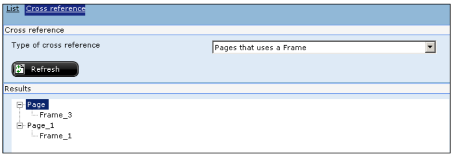
Editing

☐ Override default grid size

Width	10
Height	10

This mask can be used to introduce the identifying attributes of the alarm like Name and Comment. The name is a unique property within any given project that is other different frames with the same name cannot exist.
The name of a frame can contain alphanumeric characters (from ‘A’ to ‘Z’, ‘a’ to ‘z’ and from ‘0’ to ‘9’) or the character underscore (‘_’). The maximum length of the string cannot exceed 32 characters and the first digit must be alphabetical.
The Comment is a Unicode string with a maximum length of 255 characters and is visible only within POLYMATH.
As an alternative to the drawing of the dimensions as seen in the previous section, you can also manually set the width and height of the frame.
You can also overwrite the default dimensions of the grid by introducing new measurements.

Cross references



Cross reference

Type of cross reference: Pages that uses a Frame

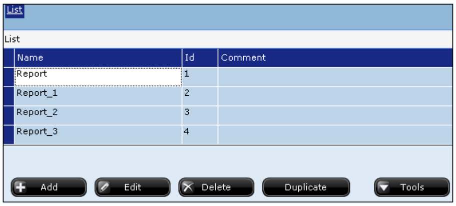
- Page
  - Frame_3
- Page_1
  - Frame_1

The mask relating to cross references offers the possibility of displaying the list of all the pages using the frame in question. This function is very useful for cataloguing the pages that are influenced by the changes made to the frame.

Report

Print reports are objects that make it possible to set out on paper information relating to the Runtime procedures. In POLYMATH different types of Report can be defined, each having an undefined number of pages. Each Report page can in turn contain all the objects found in a Page. The arrangement of these objects is independent and fixed.
in runtime, printing can be launched by pressing buttons to which predefined functions are associated or via User Script (see “Functions relating to printing” page 573 and see chap. 9, “ESAPAGEMGR methods accessible with Scripts” page 406). Naturally a compatible printer needs to be connected to the panel.
Using the Report element of Project Explorer you can define any number of Report types, Headers and Footers. Different pages, even ones belonging to the same Report, can have Different and customized Headers and Footers.

Report List



Name	Id	Comment
Report	1	
Report_1	2	
Report_2	3	
Report_3	4	

Double-clicking the Report element of Project Explorer accesses the List of Reports in the project. Using this list you can introduce Report Types by clicking on ‘Add’, duplicate existing ones by clicking on ‘Duplicate’ or delete them by

clicking on 'Delete'. In addition, existing ones can be edited by clicking on 'Edit'.
For each type of Report the summary of its characteristics is shown in editable fields (Name, ID and Comment). This mask is useful for gaining a complete view of all the recipes present in the project.

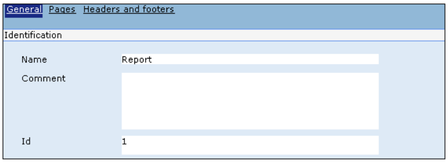
Definition of a Print Report

There are two ways of creating a Print Report:

- click on 'Add' in the Report list
- click on 'Add' or 'Add and change' on the menu appearing after clicking with the right key of the Reports element in the Project Editor

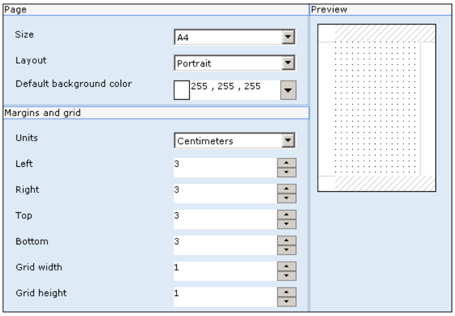
In both cases, the Report is edited by means of three tabs: General, Pages and Headers/Footers page.

General



Name: Report
Comment:
Id: 1

The upper part of the General mask can be used to introduce the general properties of the Report.
The name of a Report can contain alphanumeric characters (from 'A' to 'Z', 'a' to 'z' and from '0' to '9') or the character underscore ('_'). The maximum length of the string cannot exceed 32 characters and the first digit must be alphabetical.
The Report ID is an identifying number of the Report within the project; it is a whole number greater than zero.
The name and ID are unique attributes within the project that is other different Reports with the same name or the same ID number cannot exist.
The Comment is a Unicode string with a maximum length of 255 characters and is visible only within POLYMATH.



Page

Size: A4

Layout: Portrait

Default background color: 255 , 255 , 255

Margins and grid

Units: Centimeters

Left: 3

Right: 3

Top: 3

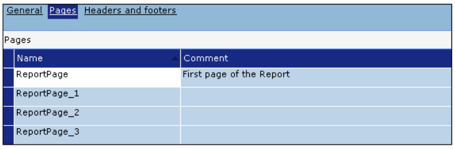
Bottom: 3

Grid width: 1

Grid height: 1

The lower part of the General mask can be used to define the common layout of each page belonging to the Report currently being edited. All the pages belonging to the same Report thus have the same layout.
Using the left side of the mask you can define the page settings while the right side shows you an updated preview of how the printed page will look.
You can define a format for the page (options are A4, A3, B5, Legal or Letter), an orientation (Portrait or Landscape) and a default Colour for the pages belonging to the Report.
Finally, after specifying a unit of measurement as a reference (options are: centimetres, pixel, inches or millimetres), you can proceed to define the margins between which to print the Report pages. You can define the left, right, top and bottom page margins. You can also define default values for the editing grid of all the pages belonging to the Report (then, if required, the grid can be edited for each individual page in the related General table).

Report page list



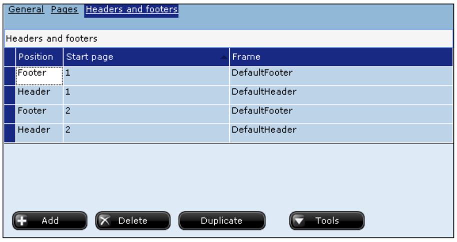
Pages

Name	Comment
ReportPage	First page of the Report
ReportPage_1	
ReportPage_2	
ReportPage_3	

The Report pages mask displays the list of the pages belonging to the Report. Use this list to add new types (by clicking on the ‘Add’ key), or duplicate (use the ‘Duplicate’ key) or delete (by using the ‘Delete’ key) the existent ones. In addition, existing ones can be edited by clicking on ‘Edit’.

For each type of Report the summary of its characteristics is shown in editable fields (Name and Comment). This mask is useful for gaining a complete view of all the Report pages.

Headers and Footers page



Headers and footers

Position	Start page	Frame
Footer	1	DefaultFooter
Header	1	DefaultHeader
Footer	2	DefaultFooter
Header	2	DefaultHeader

This Headers and Footers mask allows you to associate a Header or a Footer page (or both) to each Report page.
Just assign a Header/Footer object to the Report page ID and specify whether this is to be placed in the upper part (Headers) or the lower part (Footer).
In the following sections we will illustrate how personalized Headers and Footers can be defined.

Definition of a Report page

There are two ways of creating a Report page:

- click on 'Add' in the Report list of pages related to reports
- click on 'Add' or 'Add and change' on the menu appearing after clicking with the right key of the Reports element in the Project Editor

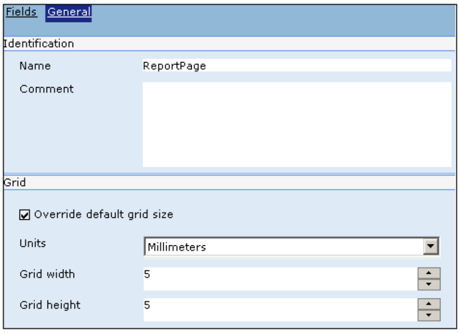
In both cases, the Report pages are edited by means of two tabs: Fields and General.

Fields

Using this mask you can define the way a Report page in question will actually appear; it is edited just like that for normal pages with various objects being introduced and properties being set (see chap. 6, “Managing a page” page 167).
The properties of the Report pages are the same as those of the Project pages (see chap. 6, “Page properties” page 170).
To introduce an object simply click on the respective icon and, immediately after, draw the outline of the area to contain it wherever you wish in the page .

The next chapter describes all the procedures for introducing the graphic objects together with their related meanings and tools.

General



Identification

Name: ReportPage

Comment:

Grid

☑ Override default grid size

Units: Millimeters

Grid width: 5

Grid height: 5

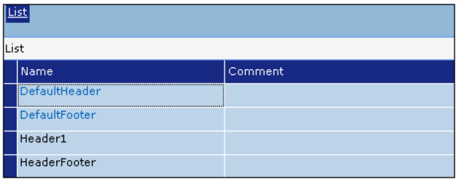
The General mask can be used to set the identifying properties of the Report page. The name can contain alphanumeric characters (from ‘A’ to ‘Z’, ‘a’ to ‘z’ and from ‘0’ to ‘9’) or the character underscore (‘_’). The maximum length of the string cannot exceed 32 characters and the first digit must be alphabetical.
The name is a unique attribute within the project that is other different Report pages with the same name cannot exist.
The Comment is a Unicode string with a maximum length of 255 characters and is visible only within POLYMATH.
Programmers can use the bottom of the mask to define their preferences regarding the editing of the page; by ticking the option required you can define whether to overwrite the default dimensions (established in the General mask of the Report) of the grid .

Definition of Header and Footer

In POLYMATH you can use, the default Headers and Footers after editing them or create an unlimited quantity of new ones. Editing for Headers/Footers is the same both for the default elements and for those introduced by the user.
After clicking on the ‘Edit’ option of the Headers/Footers list or on a Header or Footer in Project Explorer, you can proceed to the actual editing the object which is subdivided into two masks: Fields and General.

__Note:__ In this phase the objects are defined without distinguishing between Headers and Footers; thus these are created and edited in the same way and only at the moment of their being used within a Report is it specified whether they are to be placed at the top or the bottom of the page.

Header/Footer list



List

Name	Comment
DefaultHeader	
DefaultFooter	
Header1	
HeaderFooter	

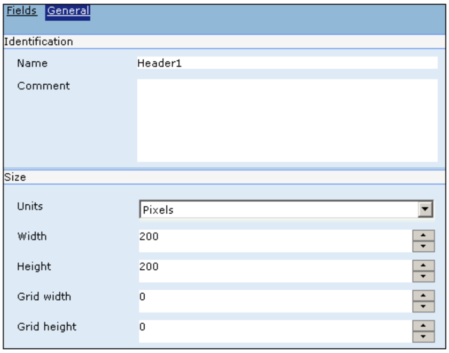
By clicking twice on the object Headers/Footers you can access the Headers/Footers list defined in the project for the Reports. As a default POLYMATH already contains two objects: Default Header and DefaultFooter; to edit these objects just click on the 'Edit' button. It is also possible to add new objects by clicking on 'Add', duplicate by clicking on 'Duplicate', or delete ones already present by clicking on 'Delete'.

Fields

Using this mask you can edit the way the Header/Footer will actually appear in the pages into which it is called; it is edited just like in the case of normal pages with various objects being introduced and properties being set (see chap. 6, "Managing a page" page 167).
The properties of the Header/Footer are the same as for Frames (see chap. 6, "Properties of Frames" page 171).
To introduce an object simply click on the respective icon and immediately after draw where in the page you wish the outline of the area to contain it to be placed.
The next chapter describes all the procedures for introducing the graphic objects together with their related meanings and tools.
Using this mask you can, however, set the dimensions of the Frame. Click on the icon to select it and then move the cursor to one of the red corners by dragging it in line with the desired dimensions (this operation can also be performed by the General mask as set out in the next section).

You cannot use this mask to move the object, in that its position could be at the top of the page (if used as a Header) or at the bottom (if used as a Footer).

General



Identification

Name: Header1

Comment:

Size

Units: Pixels

Width: 200

Height: 200

Grid width: 0

Grid height: 0

The General is used to set identifying properties of the Header/Footer page. The name can contain alphanumeric characters (from 'A' to 'Z', 'a' to 'z' and from '0' to '9') or the character underscore ('_'). The maximum length of the string cannot exceed 32 characters and the first digit must be alphabetical.
The name is a unique attribute within the project, that is, other different Headers/Footers with the same name cannot exist.
The Comment is a Unicode string with a maximum length of 255 characters and is visible only within POLYMATH.
At the bottom of the mask, you can define your preferences regarding editing the Headers/Footers. You can also specify a unit of measurement as a reference (options are: centimetres, pixel, inches or millimetres) and the height and breadth values of the part of the page occupied and the depth value of the grid for the edging phase.

Points relating to print formats: XML and Hardcopy

When the Print function is called using a predefined command or a Script you can decide to print the Report onto paper or onto file (or both). In the case of printing onto file, the Report specified by POLYMATH is saved onto a physical support of the panel in XML format so as to be able to be displayed on a browser and be in any case kept in a reconstructable digital format.

For more information on how to carry out this operation, the reader is advised to read the chapters illustrating this function (see "Functions relating to printing" page 573 and see chap. 9, "ESAPAGEMGR methods accessible with Scripts" page 406).
Hardcopy printout is an alternative mode for printing the Reports created in POLYMATH. With this you can print the entire content of the page displayed by the panel at the moment of the print command (adapting it to sheet format).
There are two types of Hardcopy printout:

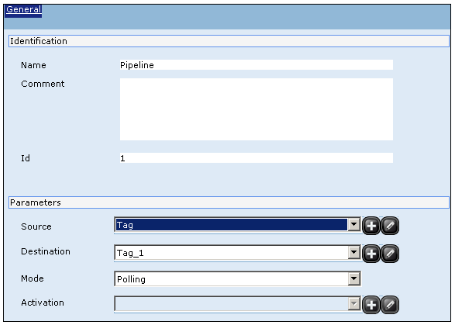
- Hardcopy page : print the current page excluding any popup
- Fullscreen hardcopy : print exactly what appears on the screen

Note: *There is also the possibility of managing the text print and values on rows in runtime exploiting the Scripting functions contained POLYMATH. Readers are advised to consult the section in this manual dealing with the Scripts to discover the potential of these functions (see chap. 9, "object ESAPRN" page 464).*

Pipelines

Pipelines are the active objects that update the value of one variable on the basis of the value of another variable. The most common application of Pipelines is for copying the value of one variable into another; this function is convenient for having the panel work as a bridge between two devices. The Pipelines created with POLYMATH are already activated at the start of the Runtime together with their particular functioning. By double-clicking on the object Pipelines in Project Explorer, the list of Pipelines in the project can be accessed. The principal characteristics of each Pipeline are entered in editable fields. Using this mask you can add new Pipelines, edit or delete existing ones. A new Pipeline can be edited after double-clicking on its Name in Project Explorer, thereby accessing the General mask described in the next section.
For more information on the table and the meaning of the events that can be associated to a Pipeline, readers are advised to consult the next chapter (see chap. 6, "Events related to Pipelines" page 166).

General



General

Identification

Name: Pipeline
Comment:
Id: 1

Parameters

Source: Tag
Destination: Tag_1
Mode: Polling
Activation:

Using the General mask you can set the identifying properties of the Pipeline. The name can contain alphanumeric characters (from 'A' to 'Z', 'a' to 'z' and from '0' to '9') or the character underscore ('_'). The maximum length of the string cannot exceed 32 characters and the first digit must be alphabetical.
The ID of the Pipeline is an identifying number of the data structure within the project; it is a whole number greater than zero.
The Pipeline name and ID are unique attributes within the project that is other different Pipelines with the same name and the same ID cannot exist.
The Comment is a Unicode string with a maximum length of 255 characters and is visible only within POLYMATH.
At the bottom of the mask you can enter the working characteristics that describe the Pipeline. First of all, it is necessary to indicate a source and destination variable.
The related sliding menu is used to select the Pipeline operating mode and this is chosen from those listed in the following table :

Tabella 3: Pipeline modes

Mode	Description
Polling	Each time a new value is read from the source variable, this value is assigned to the destination variable. The acquisition rhythm is governed by the refresh parameters of the source variable
Copy by Change	Similar to 'polling', only that the values acquired from the source variable are assigned to the destination variable only when the value of the source variable is changed
Copy by Command	The value is copied by command, that is, in line with the transition from FALSE to TRUE of the value of the auxiliary variable that can be entered into the next field (activation, must be Boolean)

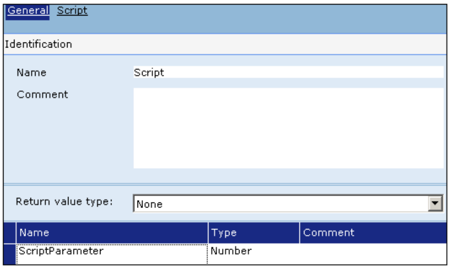
In the boxes for choosing variables you will also find the icons for adding variables ✚ and editing ✎ them.

Scripts

Scripts are an element enabling writing of functions to be customized, which is useful in that the predefined functions are not always sufficient for the user's needs. They can be written with true programming languages and be executed directly in runtime. For more information on writing Script codes readers are advised to consult the chapter dealing with these (see chap. 9, "Scripts" page 379).
In this section we shall limit ourselves to describing the how the Scripts are managed at the project level .
By double-clicking on the Script option in Project Explorer, the list of Scripts in the project (with their related comments) can be accessed. Using this list mask you can add new Scripts to the project (using the 'Add' key) or duplicate them (using 'Duplicate') and delete existing ones (using 'Delete'). A new Pipeline can be edited after double-clicking on its Name in Project Explorer, thereby accessing the General mask described in the next section.
To be able to enter the actual edit mode for a Script, double-click on it in Project Explorer; there are two masks for editing Scripts: General and Scripts. These are described in the next subsections.

For more information on the table and the meaning of the events that can be associated to a Script, readers are advised to consult the next chapter (see chap. 6, “Events related to Pipelines” page 166).

General



Identification

Name: Script

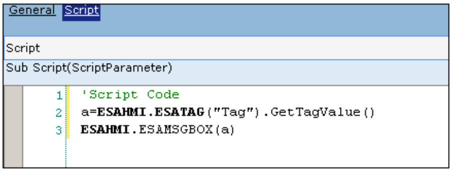
Comment:

Return value type: None

Name	Type	Comment
ScriptParameter	Number	

Using the General mask you can set the identifying properties of the Script. The Name of a Script can contain alphanumeric characters (from ‘A’ to ‘Z’, ‘a’ to ‘z’ and from ‘0’ to ‘9’) or the character underscore (‘_’). The maximum length of the string cannot exceed 32 characters and the first digit must be alphabetical.
The Name is a unique attribute within the project that is, other different Scripts with the same name cannot exist.
The Comment is a Unicode string with a maximum length of 255 characters and is visible only within POLYMATH.
It is also necessary to define whether the Script must return a value to the application and what type of value this must be (Number, String or Variant).
Use the table at the bottom of the mask to specify the input parameters to the function with their related Names, Types (Number, String or Variant) and Comments (visible only in POLYMATH).

Scripts



Script

Sub Script(ScriptParameter)

```
'Script Code
a=ESAHMI.ESATAG("Tag").GetTagValue()
ESAHMI.ESAMSGBOX(a)
```

The Script mask contains only one text input window, inside of which you enter the code relating to the functions the Script will have. For more information regarding the uses of Scripts we advise the reader to consult the relevant part of the manual (see chap. 9, “Scripts” page 379).

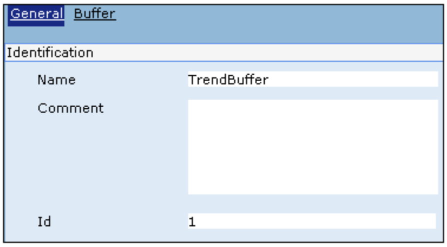
GlobalScripts

GlobalScripts function in the same way as the Scripts described in the preceding chapter; the real difference is that these types of Script cannot be associated to an event or a key but are activated with the start up of the Runtime.
They work and are edited in the same way as standard Scripts (see chap. 5, “Scripts” page 140), the only difference consists in the non configurability of the input parameters and the return values for these functions (as they cannot be called up inside the project).

Trend Buffers

in runtime the system supplies the support for the acquisition and accumulation of numerical values and for their graphic presentation in the form of a “trend curve”.
The accumulated data can be presented in real time or saved into the permanent memory and recalled to the screen at a later point.
By double-clicking on the Trend Buffers element in Project Explorer, the list of Trends held in the project can be accessed. This list also offers a summary of the principal characteristics of the Trends in editable fields. Using this list you can create new Trend buffers and duplicate or delete existing ones.
A new Pipeline can be edited after double-clicking on its Name in Project Explorer, thereby accessing the General mask described in the next section.
For more information on the table and the meaning of the events that can be associated to a Pipeline. For more information on the table and the meaning of the events that can be associated to a Trend buffer, readers are advised to consult the next chapter (see chap. 6, “Events related to Trend Buffers” page 167).
Once a new Trend has been created double-click on it in Project Explorer to be able to edit it. For this there are two pages General and Buffer as indicated in the following sections. Project Explorer is used only to define the operation of each Trend Buffer, while the way it is drawn is dealt with in the next chapter (see chap. 6, “Trend View” page 276).

General



Identification

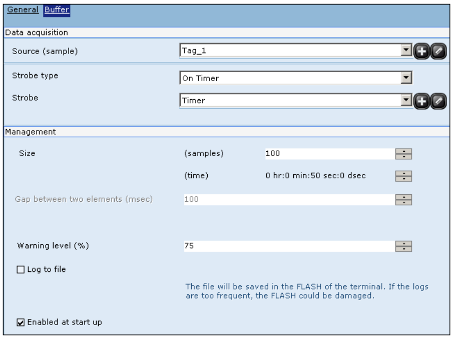
Name: TrendBuffer

Comment:

Id: 1

Using the General mask you can set the identifying properties of the Trends. The name can contain alphanumeric characters (from ‘A’ to ‘Z’, ‘a’ to ‘z’ and from ‘0’ to ‘9’) or the character underscore (‘_’). The maximum length of the string cannot exceed 32 characters and the first digit must be alphabetical.
The ID of the Trend is an identifying number of the data structure within the project; it is a whole number greater than zero.
The name and ID of a Trend are unique attributes within the project that is other different Trends with the same name and number cannot exist.
The Comment is a Unicode string with a maximum length of 255 characters and is visible only within POLYMATH.

Buffer



Data acquisition

Source (sample): Tag_1

Strobe type: On Timer

Strobe: Timer

Management

Size (samples): 100

(time): 0 hr:0 min:50 sec:0 dsec

Gap between two elements (msec): 100

Warning level (%): 75

☐ Log to file

The file will be saved in the FLASH of the terminal. If the logs are too frequent, the FLASH could be damaged.

☑ Enabled at start up

In this mask enter the operating characteristics of the Trend and of the related memory buffer.
First of all, a source variable to the object of the monitoring of the Trend must be specified.

You also need to indicate a sampling mode for the values. The types of sampling available are summed up in the following table:

Tabella 4: Types of Trend sampling

Sampling mode	Description
Time based	the sampling is done at regular intervals
On Strobe Raise	the sampling is done when the reference variable changes the value from FALSE to TRUE
On Strobe Fall	the sampling is done when the reference variable changes the value from TRUE to FALSE
On Command	the sampling is done on receipt of a command from a Script, function or command area

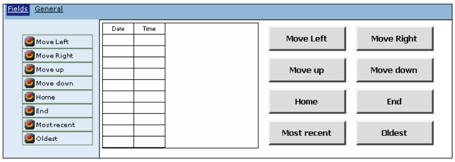
If the type of sampling is Time-based it will be necessary to enter a reference to a Timer specially configured so as to acquire sampling of the TrendBuffer (see chap. 5, “” page 85), while if the type of sampling is On Strobe Raise or On Strobe Fall it will be necessary to specify a Boolean variable (see chap. 5, “Value” page 91).
Setting the TrendBuffer also requires its dimension be indicated: the maximum number of samples to be saved can be defined, or, if the sampling frequency refers to a timer, the maximum duration of the buffer (in tenths of a second).
The system can manage the buffer either on a FIFO (first in first out: the least recent element is eliminated) or an ARRAY basis (when the buffer is full the new values are disregarded).
You can also set a Warning value, expressed as a percentage, beyond which the user must be advised that the Buffer is nearly full (this triggers an OnWarningLevel event).
The option ‘Save to File’ at the bottom of the mask indicates whether the elements of the TrendBuffer must be saved to file so as to be kept after the terminal is switched off (otherwise they are retained in the volatile memory). If this option is activated, a storage file name (containing characters supported by a Windows environment) will also have to be specified. The log file is memorized in the \log folder (see chap. 8, “Transferring data” page 362).
The last option relates to the possibility of enabling the Trend at the start-up of the project; if the Buffer is associated to a Timer, it will still be necessary to start the Timer (see chap. 6,

“Properties of the Password Grid” page 306) to begin the acquisition.

Note: *The log file is a file the system uses to permanently save the data to be represented in the TrendView. The fact that its name can be chosen using POLYMATH is useful in that this allows the user to manage the file (e.g. copy or delete if dimensions are too great). If, however, you want to manipulate the data of a TrendBuffer it will have to be exported using either a predefined function or Script (see “Appendix B - Predefined functions” page 563 and see chap. 9, “Scripts” page 379).*

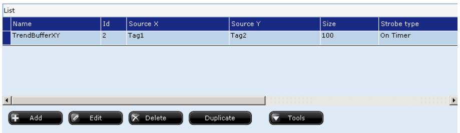
DataLog

The "DataLog" is a property similar to the "TrendBuffer". The biggest difference is that while the "TrendBuffer" is data displayed on a graphic, the "DataLog" is data displayed on a table.



TrendBuffersXY

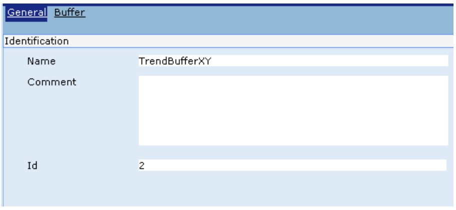
The display graphic of the "TrendBufferXY" property is the representation of two distinct variables, and not like in the "TrendBuffer" of a variable depending on time. Therefore, as shown in the following image, in the assignment phase, the variables must both be determined (Source X and Source Y).


List

Name	Id	Source X	Source Y	Size	Strobe type
TrendBufferXY	2	Tag1	Tag2	100	On Timer

Clicking "Modify" the "General" and "Buffer" masks can be accessed.

General



Identification

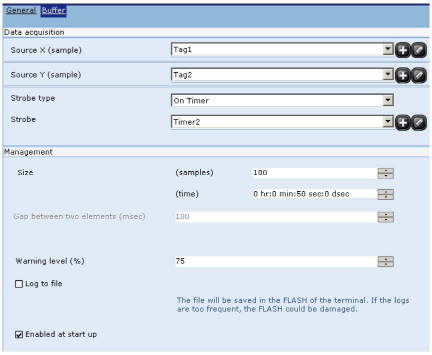
Name: TrendBufferXY

Comment:

Id: 2

The identification properties of "TrendXY" can be set on the "General" mask. The "Name" can contain alpha-numerical characters (from 'A' to 'Z', 'a' to 'z' and from '0' to '9') or the underscore character ('_').
The'"ID" of "TrendXY" is an identification number of the data structure inside of the project; it is a whole number greater than zero.
The "Name" and"ID" of a "TrendXY" are alone attributes inside of the project. Distinct "TrendXY"'s having the same name or ID number cannot exist.
The comment is a Unicode string visible only inside of POLYMATH.

Buffer



Data acquisition

Source X (sample): Tag1

Source Y (sample): Tag2

Strobe type: On Timer

Strobe: Timer2

Management

Size (samples): 100

(time): 0 hr:0 min:50 sec:0 dsec

Gap between two elements (msec): 100

Warning level (%): 75

☐ Log to file

The file will be saved in the FLASH of the terminal. If the logs are too frequent, the FLASH could be damaged.

☑ Enabled at start up

The functional features of the TrendXY and the relative memory buffer are indicated on this mask.
First of all, as can be seen, two distinct tag sources must be indicated (and not only one, as in "Trend", since the second

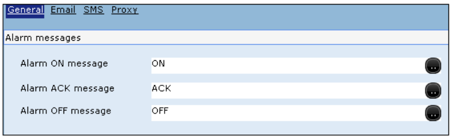
variable was the time) which will be the object of "TrendXY" monitoring.
As in the "Trend" function, a value sampling mode must be indicated. The available sampling types and their properties are identical to those on "Table 4" and in the successive descriptions previously shown.

Remote Notifications

The "Remote Notifications" function allows to send notification messages by e-mail to a previously created user list.
Associating the notification message to a given alarm present in the project, a message can be sent to one or more users, for example.
The message can be associated to one (or more than one) of the three alarm states: "Raised", "Acknowledged", "Acquired".
From "Explore Project", double-clicking "Remote Notifications", the editing area is accessed.

General

The text which will make up the message in the desired language can be typed on the "General" mask.

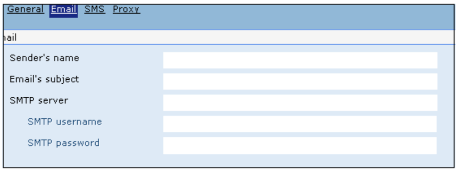


Alarm messages

Alarm ON message	ON
Alarm ACK message	ACK
Alarm OFF message	OFF

Email

From the "Email" mask it is possible to determine the settings required to send the "Notifiche Remote" (Remote Notifications) via E-mails.

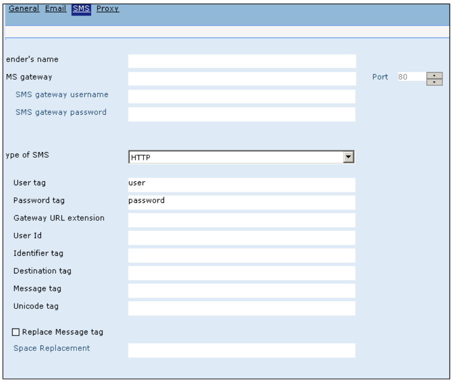


ail

Sender's name
Email's subject
SMTP server
SMTP username
SMTP password

SMS

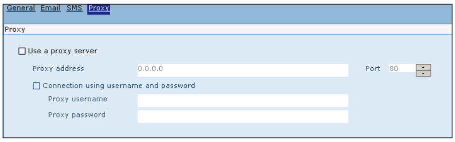
From the "SMS" mask it is possible to determine the settings required to send the "Notifiche Remote" (Remote Notifications) via SMS service.



ender's name			
MS gateway		Port	80
SMS gateway username			
SMS gateway password			
ype of SMS	HTTP		
User tag	user		
Password tag	password		
Gateway URL extension			
User Id			
Identifier tag			
Destination tag			
Message tag			
Unicode tag			
□ Replace Message tag			
Space Replacement			

Proxy

From the "Proxy" mask it is possible to determine the settings required to send the "Notifiche Remote" (Remote Notifications) via Proxy service.

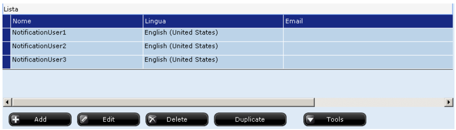


Proxy

□ Use a proxy server

Proxy address	0.0.0.0	Port	80
□ Connection using username and password			
Proxy username			
Proxy password			

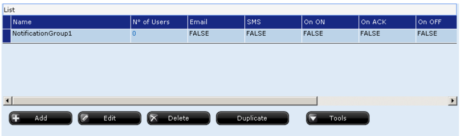
From "Explore Project", double-clicking "Notify User", the user list is accessed.
Clicking "Add", an unlimited number of different users can be added.
The name, e-mail address (needed to notify the message) and the language can be specified for each user.
These parameters can be edited modifying the corresponding fields :



Lista

Nome	Lingua	Email
NotificationUser1	English (United States)	
NotificationUser2	English (United States)	
NotificationUser3	English (United States)	

All users previously created can be put into one or more groups.
From "Explore Project", double-clicking "Notify User", the group list is accessed.
Clicking "Add", one or more groups can be added :

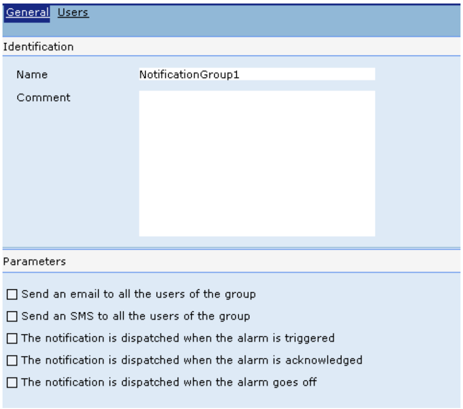


List

Name	N° of Users	Email	SMS	On ON	On ACK	On OFF
NotificationGroup1	0	FALSE	FALSE	FALSE	FALSE	FALSE

Clicking "Modify", the "General" and "User" masks are accessed.

General



Identification

Name NotificationGroup1

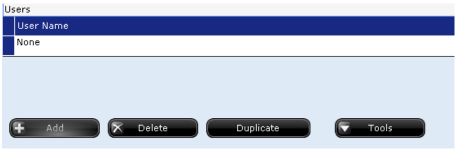
Comment

Parameters

□ Send an email to all the users of the group
□ Send an SMS to all the users of the group
□ The notification is dispatched when the alarm is triggered
□ The notification is dispatched when the alarm is acknowledged
□ The notification is dispatched when the alarm goes off

From the "General" mask, an e-mail can be sent to all the users in the group, choosing from the options of the "Parameter" voice.
The notification can be sent when the alarm is "Raised" or "Acknowledged" or when the alarm "Ends".

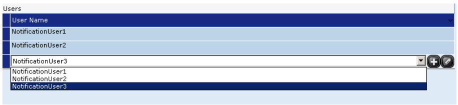
Users



Users

User Name
None

From the "Users" mask, clicking "Add", the users created previously can be added one at a time.
Double-clicking "None", the users on the list can be chosen.
To add other users to the same group, repeat the operation, clicking "Add" again, until the desired number of users is reached.



Users

User Name
NotificationUser1
NotificationUser2
NotificationUser3

NotificationUser1
NotificationUser2
NotificationUser3

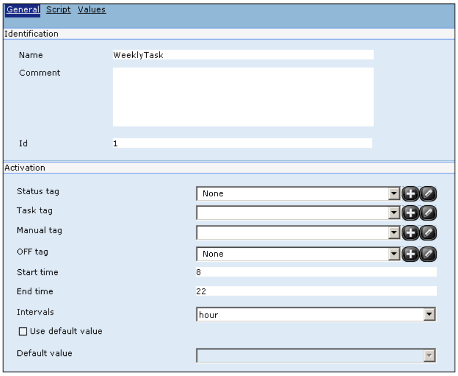
Weekly Tasks

The "TaskSettimanali" (Weekly Tasks) allow to set all functions that are necessary to create and edit a "Cronotermostato" (Chronothermostat).
After double clicking on the "TaskSettimanali" (Weekly Tasks) in the '"Esplora Progetto" (Project Explore), a list of "WeeklyTasks" inserted within the project will appear in the work area. From this list it is possible to insert new, duplicate or eliminate existing. The "Strumenti" (Tools) key allows to modify the structure of the columns at will.

Once a "WeeklyTask" has been created (from the "Esplora Progetto" (Project Explore) or list), by double clicking on it in the tree chart, it can be edited in the work area. The "WeeklyTask" editor is organised in the following sections: "Generale", "Script" e "Valori" (General, Script and Values). The description for each mask is supplied in the following subparagraphs.

The properties and events that can be associated to the "WeeklyTask" object will be treated in the next chapter. It is therefore recommended to consult the relative section for the list and meaning (chapter 8, "Cronotermostato" (Chronothermostat).

General



Identification

Name: WeeklyTask

Comment:

Id: 1

Activation

Status tag: None

Task tag:

Manual tag:

OFF tag: None

Start time: 8

End time: 22

Intervals: hour

☐ Use default value

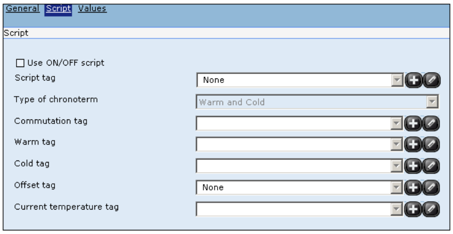
Default value:

The "Generale" (General) mask shows the data relative to the "Cronotermostato" (Chronothermostat) settings. It is possible to introduce the identification attributes of the page, such as "Nome", "Commento" e "Id" (Name, Comment and Id).
The different editing fields listed below can be found in the "Attivazione" (Activation) section:

- "Tag di stato" (State tag): it can be associated to a Boolean variable that indicates the switch-on state (1) or switch-off state (0) of the "Cronotermostato" (Chronothermostat) scheduling
- "Tag task": indicates the current temperature value
- "Tag manuale" (Manual tag):is the variable that memorises the temperature value that can be set in "Manuale" (Manual) mode.
- "Tag OFF": it can be associated to a Boolean variable that indicates the switch-on state (1) or switch-off state (0) of the "Cronotermostato" (Chronothermostat)

- "Tempo inizio" (Start time): indicates the minimum value relative to the time scale in the "Cronotermostato" (Chronothermostat) graphics
- "Tempo fine" (End time): indicates the maximum value relative to the time scale in the "Cronotermostato" (Chronothermostat) graphics
- "Intervalli" (Intervals): allows to set the time intervals in the graphics between "hour" and "half-hour"
- "Valore di default" (Default value): indicates the desired temperature value that can be activated by pressing the "Default" key in the "Cronotermostato" (Chronothermostat). The system exits the "Automatico" (Automatic) mode and passes to "Manuale" (Manual) mode with the Default value.

Script



General Script Values

Script

☐ Use ON/OFF script

Script tag: None

Type of chronoterm: Warm and Cold

Commutation tag:

Warm tag:

Cold tag:

Offset tag: None

Current temperature tag:

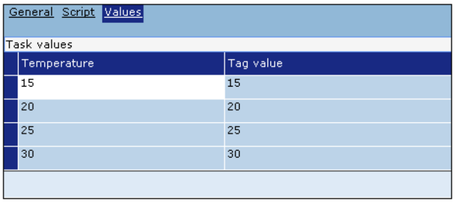
The "Scripts" that can be activated in this page, allow to define the behaviour of the "Cronotermostato" (Chronothermostat) in the "Riscaldamento" (Heating) or "Raffreddamento" (Cooling) modes or both.
"Tag script" is a service variable that memorises the activation state (1) or not (0) of the "Script" itself.
The "Tipo di cronotermostato" (Type of Chronothermostat) indicates which mode to make active in the system :

- "Solo riscaldamento" (Heating only): the system questions the values set and changes the values of the "Tag caldo" (Hot tag)
- "Solo raffreddamento" (Cooling only): the system questions the values set and changes the values of the "Tag freddo" (Cold tag)
- "Riscaldamento e Raffreddamento" (Heating and Cooling): the system questions the values set and changes the values of the "Tag caldo" (Hot tag) or "Tag freddo" (Cold tag) according to the system active. The passage of activation from one system to another is activated by

the "Estate" (Summer) or "Inverno" (Winter) control present in the "Cronotermostato" (Chronothermostat)

The "Tag di commutazione" (Switch-over tag) indicates the active state (1) or off state (0) of the "Riscaldamento e Raffreddamento" (Heating and Cooling) system
Differently, the "Tag caldo" (Hot tag) and "Tag Freddo" (Cold tag) indicate the temperature reference values for activation of the system.
The "Tag offset" can be used to set the offset value referring to the "Cronotermostato" (Chronothermostat) working temperature in a variable.
The actual temperature value is memorised in the "Tag temperatura corrente" (Current temperature tag).

Values



Task values

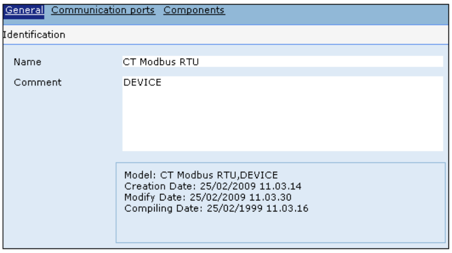
Temperature	Tag value
15	15
20	20
25	25
30	30

The "Valori" (Values) table contains the data relative to the temperature and corresponding tag values. It can be useful to assign a different tag value with respect to the temperature when the device, used in the realisation of the project, requires conversion of the data according to its programming features. The "Strumenti" (Tools) key allows to modify the structure of the columns at will.

Setting the device

Each device added to the project can have a series of properties and operational characteristics defined for it; for example, you can define the way the related memory and the conversion tables are to be used.
By double-clicking on the device using Project Explorer you can start to edit: editing is organized via three masks, General, Communication ports and Components.

General



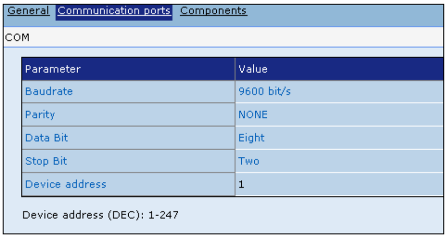
Name: CT Modbus RTU

Comment: DEVICE

Model: CT Modbus RTU,DEVICE
Creation Date: 25/02/2009 11.03.14
Modify Date: 25/02/2009 11.03.30
Compiling Date: 25/02/1999 11.03.16

Using this mask you can introduce the identifying attributes of the device like Name and Comment. The Name is a unique property in terms of the project, that is, there can be no other devices (though of the same Type, make and model) in the project with the same Name.
The Name of a device can contain alphanumeric characters (from ‘A’ to ‘Z’, ‘a’ to ‘z’ and from ‘0’ to ‘9’) or the character underscore (‘_’). The maximum length of the string cannot exceed 32 characters and the first digit must be alphabetical.
The Comment is a Unicode string with a maximum length of 255 characters and is visible only within POLYMATH.

Communication ports

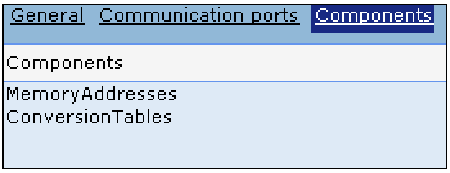


Parameter	Value
Baudrate	9600 bit/s
Parity	NONE
Data Bit	Eight
Stop Bit	Two
Device address	1

Device address (DEC): 1-247

This window is used to configure the mode of communication between panel and device. All the ports present on the element can be configured. Generically, you can set the dialog speed (baudrate) and the general parameters of the communication protocol (parity bits, stop bit) and the device address. At the bottom of the window you can enter the range permitted by the protocol for each value.

Components



Components

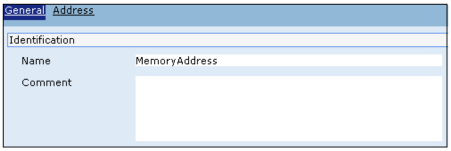
MemoryAddresses
ConversionTables

This page offers only a summary of the components that can be associated to the device; by clicking on each of these you access the related main editing page.

MemoryAddresses

Using POLYMATH you can set all the Memory addresses related to a device; using this section you prepare the memory areas that will then be used in the project (e.g. as a repository for the variables).
By double-clicking on the MemoryAddresses element in Project Explorer, the list of Memory areas already set in the project can be accessed. Using this list you can introduce new addresses (using the 'Add' key) and delete ('Delete' key) or duplicate ('Duplicate' key) existing ones.
Once a new Memory address has been created, by clicking in Project Explorer you can access the related editing area, organized in the masks General and Address.

General



Identification

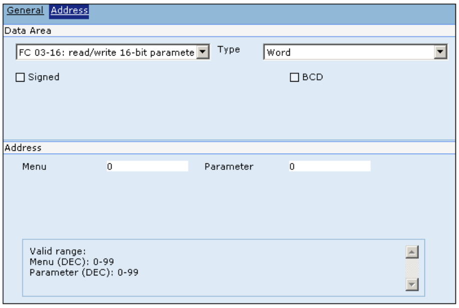
Name MemoryAddress

Comment

Using this mask you can introduce the identifying attributes of the Memory address: the Name and Comment. The name is a unique property within the project that is other different memory addresses with the same name cannot exist.
The name of a memory address can contain alphanumeric characters (from 'A' to 'Z', 'a' to 'z' and from '0' to '9') or the character underscore ('_'). The maximum length of the string cannot exceed 32 characters and the first digit must be alphabetical.

The Comment is a Unicode string with a maximum length of 255 characters and is visible only within POLYMATH.

Address



Data Area

FC 03-16: read/write 16-bit paramete

Type: Word

□ Signed

□ BCD

Address

Menu: 0

Parameter: 0

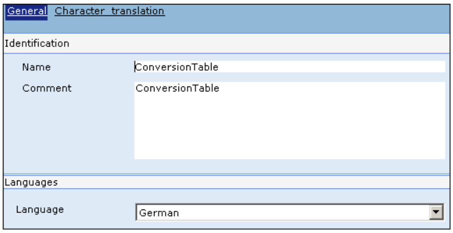
Valid range:
Menu (DEC): 0-99
Parameter (DEC): 0-99

This mask lets you edit the effective physical characteristics of the memory. You can choose the destination data area and the type of reading to be made in it; the drop-down menus relating to the data area are compiled dynamically by POLYMATH in line with the device in question. in addition you need to specify whether the type of data is Signed or in BCD (see chap. 5, “” page 92).
The bottom of the mask is used to enter the effective Memory addresses to which the current are must be assigned; POLYMATH shows the ranges of admissible values for the addresses to be introduced. It is the programmer’s job to avoid any unwanted superimpositions of addresses in different areas.

Conversion Tables

Conversion Tables are useful tools for supplying the translation of certain string characters read by the PLC in the context of a particular language. This feature is useful in those cases in which you need to display UNICODE characters each time a particular ASCII character is encountered.
By double-clicking on the Conversion Tables element in Project Explorer, the list of Conversion Tables already in the project can be accessed. For each element the general properties are displayed in editable fields. Using this list you can introduce new tables or delete or duplicate existing ones. Once a new table has been created, simply click twice on it in Project Explorer to be able to access the related editing feature organized in the masks General and Character translation.

General



Identification

Name: ConversionTable

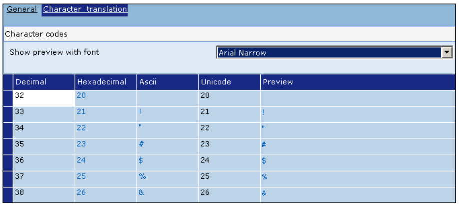
Comment: ConversionTable

Languages

Language: German

Using this mask you can introduce the identifying attributes of the Conversion table: the Name and Comment. The name is a unique property within the project that is other different memory addresses with the same name cannot exist.
The name of a memory address can contain alphanumeric characters (from 'A' to 'Z', 'a' to 'z' and from '0' to '9') or the character underscore ('_'). The maximum length of the string cannot exceed 32 characters and the first digit must be alphabetical.
The Comment is a Unicode string with a maximum length of 255 characters and is visible only within POLYMATH.
At the bottom of the mask a reference language for the Table of translations being edited must be defined.

Character translation



Character codes

Show preview with font: Arial Narrow

Decimal	Hexadecimal	Ascii	Unicode	Preview
32	20		20	
33	21	!	21	!
34	22	"	22	"
35	23	#	23	#
36	24	$	24	$
37	25	%	25	%
38	26	&	26	&

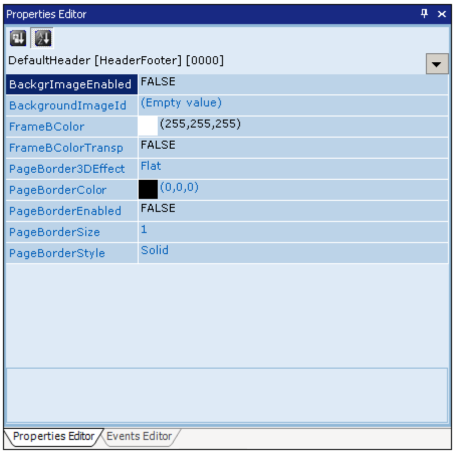
This mask is used to introduce the effective list of translations the have to be executed in runtime; A UNICODE translation must be supplied for each ASCII character (represented in decimal format). For each translation POLYMATH supplies a preview in UNICODE as well as a representation of the character in ASCII and hexadecimal.

To use this feature you have to enable the String-type variable (to which this change must be applied) in such a way as to permit the translation of the characters (see chap. 5, “” page 92).
Using this mask you can introduce new character translations, edit or delete existing ones; you can also choose from an appropriate pull-down menu a reference Font for the preview of the translation.

6. Properties Editor

The purpose of this chapter is to describe all those functions offered by POLYMATH for editing the graphics and the accessibility of the project applications. Our starting point is the concept that each executable operation, each visible data (modifiable or not), each link between the pages, each function button must appear to the operator inside a page displayed on the VT.
We shall start out giving some indications of the general organization of the pages and go on to give more detailed information on all the elements that can be introduced together with their characteristics. For each graphic element that can be introduced in a page (and for the pages themselves) a series of properties can be defined that determine the aspect that the object will assume in Runtime.
Furthermore, in the case of many objects, functions or scripts are applicable when particular events are triggered.
The reference windows for managing the properties and events are the Properties Editor and the Events Editor respectively.

Properties Editor



DefaultHeader [HeaderFooter] [0000]

BackgrImageEnabled	FALSE
BackgroundImageId	(Empty value)
FrameBColor	□ (255,255,255)
FrameBColorTransp	FALSE
PageBorder3DEffect	Flat
PageBorderColor	■ (0,0,0)
PageBorderEnabled	FALSE
PageBorderSize	1
PageBorderStyle	Solid

The Properties Editor is composed essentially of a list of properties and related values in editable fields. If the value fields are not editable it means that the current configuration of the element does not permit any change in its value; in these cas-

es, editing the fields in question is only possible when the correlated attributes allow it.
Changes in the graphic properties of an object cause an immediate redrawing of the object on the page which is noticeable to the programmer.
If the Properties Editor does not appear on the screen because it has already been closed, it can be recalled to the screen by clicking on the icon in the toolbar or, using the Main menu, by clicking on Display->Show->Properties Editor. Like all anchorable windows, the Properties Editor too can be moved, reduced to an icon or closed (see chap. 3, “Moving Anchorable windows” page 41).
Over the next paragraphs we will show the editable properties of each object and the meanings of these properties.

Dynamic assigning of values to the properties

Some properties can have a variable assigned to them rather than having a constant value. The value of the properties can change in Runtime in line with the changes of the variables assigned to them.
To pass from the assigning-a-constant mode to assigning-a-variable mode, just click on the icon present on the left of the editable field. If in assigning-a-constant mode, the icon will be and pressing on it will take you to assigning-a-variable mode. If in assigning-a-variable mode, the icon will be and pressing on it will take you to assigning-a-constant mode.
The type of variable assigned must, naturally, be compatible with the values requested by the properties; for example:

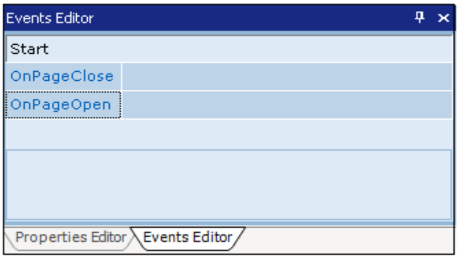
- for the properties True/False, the variable must assume Boolean values
- for the properties DateAndTime, the variable must assume Long values
- for properties defining colors (e.g. BorderColor, AreaColor, etc.), the variable must assume admissible RGB (Long) values as indicated in the following table:

Tabella 1:

Color	RGB	Hexadecimal value
Red	255,0,0	00 00 00 FF
Green	0,255,0	00 00 FF 00
Blue	0,0,255	00 FF 00 00

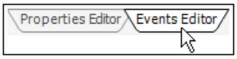
Events Editor

The Events Editor is composed of a list of events that can be assigned to the element in question.

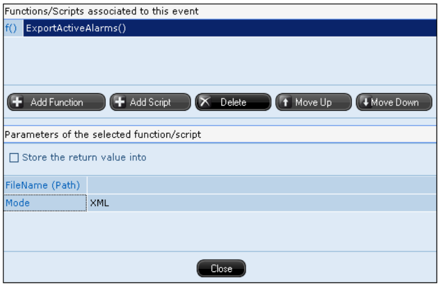


If the Events Editor does not appear on the screen because it has already been closed, it can be recalled to the screen by clicking on the icon on the toolbar or, using the Main menu, by clicking on Display->Show->Events Editor. Like all anchorable windows, the Events Editor too can be moved, reduced to an icon or closed (see chap. 3, "Moving Anchorable windows" page 41).
Over the next paragraphs we will show, in relation to each object, the events to which functions and scripts can be assigned.
For further details relating to the functions and scripts, the reader is advised to consult the appropriate sections of this manual (see chap.‶Appendix B - Predefined functions" page 563 and see chap. 9, "Scripts" page 379).

Note: *When POLYMATH is first started the Events Editor is incorporated as a clickable icon for the Properties Editor. To access it just click in the corresponding area at the bottom of the window:*



Use this window to assign a predefined function or a user script to each event simply by double-clicking on the corresponding row in the table and then on the key.



Functions/Scripts associated to this event

f() ExportActiveAlarms()

Parameters of the selected function/script

☐ Store the return value into

FileName (Path)	
Mode	XML

The resulting mask will allow you to make all the settings necessary; to add a function just click on 'Add Function' and choose the function you want from the list that appears. Similarly, by clicking on 'Add Script' you can choose the Script to be assigned. For objects like touch buttons, Function keys and Switchbuttons up to 2 functions/scripts per corresponding event can be introduced; for the events of other objects generally only one function or script can be assigned. To change the order in which the functions must be executed just move them using the 'Move up' and 'Move down' keys. To eliminate a function you just need to select it and click on the 'Delete' button.

Warning: *Where two functions can be assigned to the same event the user must take care to furnish an order which is logical for consecutive functions: there would be no sense, for example, in having a function referring to an old page follow a function of Change page.*

If you choose to assign a predefined function to an event, the lower part of the window can be used to enter its parameters (e.g. file name, name of objects, etc.).
If you choose to assign a script to an event, you can choose to save the value returned by that script (if the script is set to return a value) in a variable.
For further details relating to the assignable functions and the scripts, the reader is advised to consult the appropriate chapters of this manual (see chap.‟Appendix B - Predefined functions” page 563 e see chap. 9, “Scripts” page 379).

Note: *If you wish to assign more than one function to a key, it is better to use a user script containing those functions.*

We set out below a description of the events that can be assigned to some of the elements already seen in Project Explorer. The list of events in the case of graphic elements will by contrast be dealt with case by case.

Events related to variables

Tabella 2: Events assignable to variables

Event	Description
OnInizialization	Activated immediately after initialization of the variable, that is, at the startup of Runtime
OnRawValueChange	Activated when the peripheral device assigns a new rough value to the variable (therefore also at the startup of the project and when the connection with the device is re-established). The event is always generated before the value itself is transferred
OnValSent	Activated when the rough value has been correctly sent to the field device
OnOnScan	Activated when the update of the variable field is enabled; this happens via a script setting the attribute OffScan at False (see chap. 5, "Device" page 93)
OnOffScan	Activated when the update of the variable field is disabled; this happens via a script setting the attribute OffScan at True (see chap. 5, "Device" page 93)
OnOnLine	Activated when the variable goes On Line, that is, when it becomes available again after a break in communication
OnOffLine	Activated when the variable goes Off Line, that is, when it becomes unavailable following a break in communication
OnValueChange	Activated when a new value is assigned to the variable (thus also at the startup of the project and when the connection with the device is re-established). The event is always generated before the value itself is transferred

Tabella 2: Events assignable to variables

Event	Description
OnThdLevelN	This event is present only if at least one Level threshold is assigned to the variable (see chap. 5, “Thresholds” page 99); for every defined level there will be a single event with N indicating the reference level. This event is activated when the N level threshold is reached
OnThdDevLo	This event is present only if a Low Deviation Threshold (see chap. 5, “Thresholds” page 99) is assigned to the variable and is activated when the low level threshold is reached
OnThdDevHi	This event is present only if a High Deviation Threshold (see chap. 5, “Thresholds” page 99) is assigned to the variable and is activated when the high level threshold is reached
OnThdRateLo	This event is present only if a Low Variation Speed Threshold (see chap. 5, “Thresholds” page 99) is assigned to the variable and is activated when the low level threshold is reached
OnThdRateHi	This event is present only if a High Variation Speed Threshold (see chap. 5, “Thresholds” page 99) is assigned to the variable and is activated when the high level threshold is reached

Events related to alarms

Tabella 3: Events assignable to Alarms

Event	Description
OnAlarmOn	Launched when the alarm enters the active stat; the script or the function are run after the instance of the alarm event in the table of active alarms
OnAlarmAck	Activated when the alarm has been acknowledged

Events related to Recipes

Tabella 4: Events assignable to Recipes

Event	Description
OnRecipeCreate	Activated when the Recipe is created
OnRecipeDelete	Activated when the Recipe is about to be deleted from the archive. The event is generated immediately before the effective deletion of the Recipe
OnRecipeRead	Activated when the Recipe is correctly transferred from the memory of the terminal to the video buffer; this operation is executed using the Load button of the editor table (see chap. 6, "Recipe Editing Table" page 310)
OnDownloadComplete	Activated when the download from the VT to the device is completed
OnDownloadError	Activated when errors occur in the download from the VT to the device
OnUploadComplete	Activated when the upload from the VT to the device is completed
OnUploadError	Activated when errors occur in the upload from the VT to the device

Script Events key

Tabella 5: Events that can be associated to the Scripts

Evento	Descrizione
OnScriptError	Activated when errors appear during execution of the Script

Events related to Pipelines

Tabella 6: Events assignable to Pipelines

Event	Description
OnStart	The event is activated following the startup of the Pipeline; that is, it occurs at the start of Runtime, or after a break in the connection between the variables is restored
OnStop	When a Pipeline stop is requested
OnSourceDown	When anomalies in the source variable stop the Pipeline working correctly (break in the connection with the device, invalid value assignment etc.)
OnDestDown	When anomalies in the destination variable stop the Pipeline working correctly

Events related to Passwords

The events relating to the Passwords can be edited by the Events editor by keeping the User table for Password configuration selected (see chap. 5, “Users” page 82).

Tabella 7: Events assignable to Passwords

Event	Description
OnLogin	Activated following a successful Login operation
OnLogout	Activated following a successful Logout operation
OnLoginError	Activated when wrong Login data is emitted
OnPasswordChanged	Activated following a change in password for a user via the user grid (see chap. 6, “Properties of the Password Grid” page 306)

Events related to Timers

The events relating to the Timers can be edited by the Events editor by keeping the reference Timer in the related list selected.

Tabella 8: Events assignable to Timers

Event	Description
OnTimerFired	Activated following the completion of the Timer count
OnTimerStart	Activated when the Timer count is started
OnSuspend	Activated when the Timer is suspended by means of a stop command
OnTimerStop	Activated following a stop command to the Timer

Events related to Trend Buffers

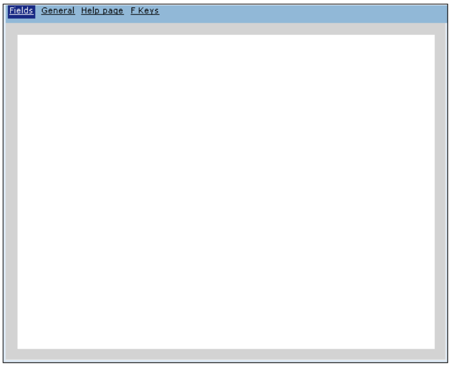
Tabella 9: Events assignable to Trend buffers

Event	Description
OnBufferFull	Activated following the admission of a new sample if, after the reading, the buffer becomes full
OnBufferOverflow	Activated when the buffer is full and a new sample has arrived
OnBufferClear	Activated when the buffer has been emptied
OnWarningLevel	Activated following the admission of a new sample and the filling of the buffer has reached the warning level (see chap. 5, “Buffer” page 143)
OnSample	A new sample has been admitted. Not generated for the trend buffers assigned to a tag array

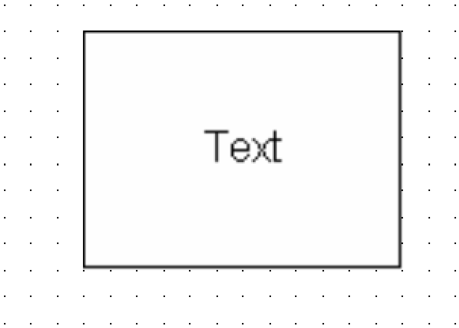
Managing a page

To set graphic and visual characteristics of a project special attention must be paid its the base element, the Page. Each graphic element, navigation or function button, command and

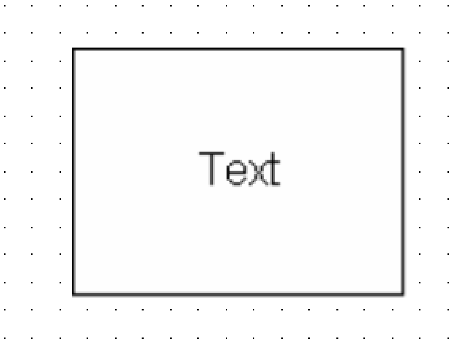
Data viewing/editing field must be positioned in a Page for it to be visible to the operator in Runtime.
To create and manage the pages in a project the reader is advised to consult the preceding chapter (see chap. 5, “Pages” page 103).
When you enter a page’s Fields mask, the work area will show a preview of how the page will be displayed on the VT.



During the editing of a page a series of programming commands are made available. Use buttons ⊕, ⊖ and 100% ▾ of the toolbar (accessible also via Layout menu ->Zoom) to change the display dimensions of a page, defining these with the Zoom (the same operation can be performed by clicking the right-hand mouse key when the pointer is on the page and choosing the required function from the menu that appears).
By clicking on the ▦ icon of the toolbar (Layout -> Show Grid) you can decide whether to show or hide the editing grid in the page preview. The grid is very useful for bringing objects in alignment very quickly when they are being arranged on the page. The grid dimensions can also be edited using Project Explorer in the VT options (see chap. 5, “Main window” page 75).
By clicking on the ▣ icon of the toolbar (Layout -> Align Grid) you can decide whether to align the objects to the grid once they have been introduced or whether to have them introduced freely.
With alignment to the grid is activated, the element can only be introduced within the limits delineated by the grid.



While if the alignment function is deactivated the elements can be freely introduced into the page as shown in the figure below



Note: *You are recommended to activate the alignment to the grid function to be sure to have a well-ordered and coherent arrangement of objects on the page.*

Should you be creating a multilanguage project (see chap. 5, "Languages" page 78), the elements in the page and the related texts can be displayed in a particular project language. To do this just select the required language from the drop-down menu English(United States) containing all the languages added to the project (this command can be accessed also via the Main menu using the sequence Display->Project language). Each time a language is chosen the display of the page changes instantly.

To introduce an object into the page click on the related icon in the toolbar (or use the Main menu) and draw the outline in the position desired on the page preview. Once an element is added it will appear in the page and can be selected simply by clicking on it. For each object selected there will appear in the Properties and Events Editor all the options the user can set, while by clicking with the right key on an object selected you can access a menu with standard functions like Edit, Duplicate, Delete, Cut, Copy and Zoom.

Page properties

Tabella 10: Page properties

Properties	Description
PageBColor	Background page color; editable using RGB code or a palette of colors
BackgroundImage Enabled	Defines whether the page must have a background page
BackgroundImageId	Chooses the background image (from the list of images introduced)
ImageReprMode	Mode of representation of the image: can be Cut, Stretched, Stretched maintaining the proportions and Position
ImageHPosition	Horizontal positioning of the image (Centered, Right or Left)
ImageVPosition	Vertical positioning of the image (Centered, Top or Bottom)
PageFrameEnabled	Defining whether to display the frame of the Page
PageFrameSize	Dimensions of the frame
PageFrameColor	Color of the frame
PageFrameStyle	Style of the frame, Solid or Broken
PageFrame3DEffect	Defines the effects of the frame: Flat, Relief or Sunken

Events related to Pages

Tabella 11: Events related to Page

Event	Description
OnPageOpen	Activated after a Page is shown
OnPageClose	Activated when a Page is about to be closed

Properties of Popup pages

The properties of the Popup page editor are exactly the same as those of the standard pages (see chap. 6, “Page properties” page 170).

Events related to Popup pages

The Editor events that can be assigned to the Popup pages are exactly the same as those of the standard pages (see chap. 6, “Events related to Pages” page 171).

Properties of Frames

Tabella 12: Frame properties

Properties	Meaning
FrameBColor Transparent	Defines whether the background of the Frame must be transparent
FrameBColor	Background color of the frame; editable using RGB code or palette of colors
PageFrameEnabled	Defines whether the page must have a frame
PageFrameSize	Indicates the dimensions of the frame
PageFrameColor	Indicates the color of the frame; editable using RGB code or palette of colors
PageFrameStyle	Permits the style of the frame to be chosen
PageFrame3DEffect	Permits the 3-D effect that can be assigned to the frame

Predefined graphic elements

POLYMATH has a set of predefined graphic elements that can be added to a page. These elements can have simple graphic functions, navigation functions and display and edit data functions. The icons relating to these objects can be found in the toolbar and the Main menu using Fields->Create.
All the graphic elements have been grouped, depending on their function, in four groups:

- Simple Figures
- Value Fields
- Simple Controls
- Complex Controls

The next paragraphs contain a list of all the graphic elements predefined by POLYMATH which can be introduced into a page. For each property we shall indicate in a schematic way the related editable properties and the events that can be assigned to them.

Warning: *When planning your project you need to bear in mind that when two buttons on the Touch Screen panel are pressed at the same time this is interpreted as having pressed halfway between these buttons. So you are advised to avoid settings that involve this situation.*

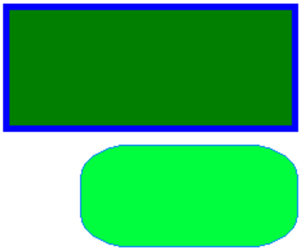
Simple Figures

The first group of graphic elements to be considered is that of the Simple Figures; these can be useful for creating more or less complex drawings or for assigning special effects to the pages.

Rectangle

A rectangle can be introduced into a page by clicking on the icon ▣ or using the Main Menu (Fields->Create->Simple figures->Rectangle) and drawing its dimensions directly in the page. This procedure enables you also to introduce rectangles with rounded outlines (see TypeOfBox properties).

The characteristics of the Rectangle must be set in the Properties Editor as indicated in the following section.



Properties of the Rectangle

Tabella 13: Properties of the Rectangle

Properties	Description
Name	Identifying name of the Rectangle. Must be unique among the graphic elements
Comment	Identifying comment within POLYMATH
Top	Vertical position coordinate
Left	Horizontal position coordinate
Width	Width dimension
Height	Height dimension
TypeOfBox	Determines whether the Rectangle must be normal or rounded
RoundX	Editable if the Rectangle is rounded; corresponds to the horizontal distance between the position of the corner and the point at which the curve joins the horizontal side of the Rectangle
RoundY	Editable if the Rectangle is rounded; corresponds to the vertical distance between the position of the corner and the point at which the curve joins the vertical side of the Rectangle

Tabella 13: Properties of the Rectangle

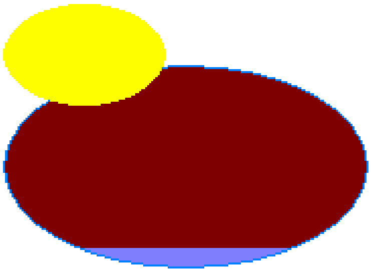
Properties	Description
AreaVisibility	Determines whether the Rectangle should have a background (True) or be transparent (False); a Boolean variable can be assigned to this value or it can be managed with thresholds
AreaColor	Determines the color of the Area that can be selected using the RGB code or color palettes. The value can be assigned to a whole variable or it can be managed with thresholds
BorderVisibility	Determines whether there will be a Border to the Rectangle or not; a Boolean variable can be assigned to this value
BorderSize	Determines the size of the Border, which must be a number to which a whole variable could be assigned if desired or it can be managed with thresholds
BorderColor	Determines the color of the Border, which can be selected using the RGB code or color palette. The value can be assigned to a whole variable or it can be managed with thresholds
BorderStyle	Determines the style of the Border, which can be Solid or Broken. The value can be assigned to a whole variable or it can be managed with thresholds
BorderBlink	Determines the flashing of the Border, which can be No blinking, Slow blinking or Rapid blinking. The value can be assigned to a whole variable or it can be managed with thresholds
Border3D	Determines a 3D effect for the Border, which can be Flat, Relief, Recessed, Bump or Etched. The value can be assigned to a whole variable or it can be managed with thresholds
PartialFillFlag	Determines whether to make a partial color infill. The value can be assigned to a whole variable

Tabella 13: Properties of the Rectangle

Properties	Description
FillColor	Determines the color of the Border infill that can be selected using the RGB code or color palette. The value can be assigned to a whole variable
FillDir	Determines the direction of the Border infill. The infill can happen From Low to High, From High to Low, From Right to Left or From Left to Right. The value can be assigned to a whole variable
FillPercent	Indicates the percentage of the infill. The value can be assigned to a whole variable
Hide	Determines whether the object is initially visible. It is also possible to assign a Boolean variable (for changes in Runtime) or it can be managed with thresholds

Ellipse

An ellipse can be introduced into a page by clicking on the icon ⬭ or using the Main Menu (Fields->Create->Simple figures->Ellipse) and drawing its dimensions directly in the page.
To define the characteristics of the Ellipse, they must be set in the Properties Editor as indicated in the following section.



Properties of the Ellipse

Tabella 14: Properties of the Ellipse

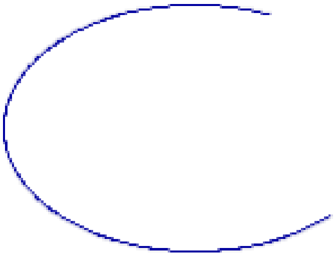
Properties	Description
Name	Identifying name of the Ellipse. Must be unique among the graphic elements
Comment	Identifying comment within POLYMATH
Top	Vertical position coordinate
Left	Horizontal position coordinate
Width	Width dimension
Height	Height dimension
LineColor	Determines the color of the Ellipse outline that can be selected using the RGB code or color palette. The value can be assigned to a whole variable or it can be managed with thresholds
Ellipse3D	Determines the 3D effect of the Ellipse, which can be Flat, Bump or Etched. The value can be associated with Tag or it can be managed with thresholds
AreaVisibility	Determines whether the Sector has a background area (True) or should be transparent (False); a Boolean variable can be assigned to this value or it can be managed with thresholds
AreaColor	Determines the color of the Area that can be selected using the RGB code or color palette. The value can be assigned to a whole variable or it can be managed with thresholds
PartialFillFlag	Determines whether to make a partial color infill. The value can be assigned to a whole variable
FillColor	Determines the color of the Ellipse infill which can be selected using the RGB code or color palette. The value can be assigned to a whole variable

Tabella 14: Properties of the Ellipse

Properties	Description
FillDir	Determines the direction of the Ellipse infill. The infill can happen From Low to High, From High to Low, From Right to Left or From Left to Right. The value can be assigned to a whole variable
FillPercent	Indicates the percentage of the infill. The value can be assigned to a whole variable
Hide	Determines whether the object is initially visible; it is also possible to assign a Boolean variable (dynamic in Runtime) or it can be managed with thresholds

Arc

An Arc can be introduced into a page by clicking on the icon ◝ or using the Main Menu (Fields->Create->Simple figures->Arc) and drawing its dimensions directly in the page.
To define the characteristics of the Arc they must be set in the Properties Editor as indicated in the following section.



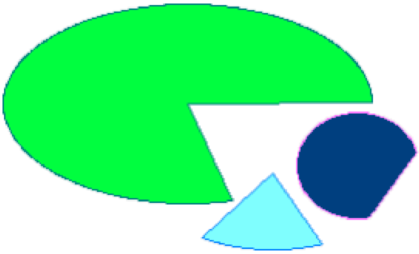
Properties of the Arc

Tabella 15: Properties of the Arc

Properties	Description
Name	Identifying name of the Arc. Must be unique among the graphic elements
Comment	Identifying comment within POLYMATH
Top	Vertical position coordinate
Left	Horizontal position coordinate
Width	Width dimension
Height	Height dimension
LineColor	Determines the color of the Arc outline which can be selected using the RGB code or color palette. The value can be assigned to a whole variable or it can be managed with thresholds
StartAngle	Determines the Arc starting position (given as an angle)
SweepAngle	Determines the angle (in degrees) of the opening of the Arc
Hide	Determines whether the object is initial visible; it is also possible to assign a Boolean variable (dynamic in Runtime) or it can be managed with thresholds

Sector

A Sector can be introduced into a page by clicking on the icon ◔ or using the Main Menu (Fields->Create->Simple figures->Sector) and drawing its dimensions directly in the page.
To define the characteristics of the Sector they must be set in the Properties Editor as indicated in the following section.



Properties of the Sector

Tabella 16: Properties of the Sector

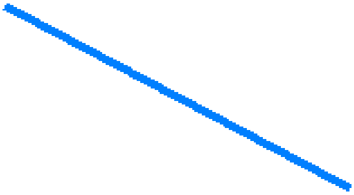
Properties	Description
Name	Identifying name of the Sector. Must be unique among the graphic elements
Comment	Identifying comment within POLYMATH
Top	Vertical position coordinate
Left	Horizontal position coordinate
Width	Width dimension
Height	Height dimension
CircularSectorType	Determines the type of Sector. If True, the line closing the sector does not pass through the center (forming a convex figure); otherwise, if False, the line passes through the center (concave figure)
StartAngle	Determines the Sector starting position (given as an angle)
SweepAngle	Determines the angle (in degrees) internal to the Sector

Tabella 16: Properties of the Sector

Properties	Description
AreaVisibility	Determines whether the Sector has a background area (True) or should be transparent (False); a Boolean variable can be assigned to this value or it can be managed with thresholds
AreaColor	Determines the color of the Area, which can be selected using the RGB code or color palette. The value can be assigned to a whole variable or it can be managed with thresholds
LineColor	Determines the color of the Sector outline, which can be selected using the RGB code or color palette. The value can be assigned to a whole variable or it can be managed with thresholds
PartialFillFlag	Determines whether to make a partial color infill. The value can be assigned to a whole variable
FillColor	Determines the color of the Sector infill, which can be selected using the RGB code or color palette. The value can be assigned to a whole variable
FillDir	Determines the direction of the Sector infill. The infill can happen From Low to High, From High to Low, From Right to Left or From Left to Right. The value can be assigned to a whole variable
FillPercent	Indicates the percentage of the infill. The value can be assigned to a whole variable
Hide	Determines whether the object is initially visible; it is also possible to assign a Boolean variable (dynamic in Runtime) or it can be managed with thresholds

Line

A Line can be introduced into a page by clicking on the icon ╲ or using the Main Menu (Fields->Create->Simple figures->Line) and drawing its dimensions directly in the page.
To define the characteristics of the Line they must be set in the Properties Editor as indicated in the following section.



Properties of the Line

Tabella 17: Properties of the Line

Properties	Description
Name	Identifying name of the Line. Must be unique among the graphic elements
Comment	Identifying comment within POLYMATH
Hide	Determines whether the object is initial visible; it is also possible to assign a Boolean variable (dynamic in Runtime) or it can be managed with thresholds
Top	Vertical position coordinate (calculated by software according to the values of X1,X2,Y1,Y2)
Left	Horizontal position coordinate (calculated by software according to the values of X1,X2,Y1,Y2)
Width	Width dimension (calculated by software according to the values of X1,X2,Y1,Y2)
Height	Height dimension (calculated by software according to the values of X1,X2,Y1,Y2)

Tabella 17: Properties of the Line

Properties	Description
Effect3D	Determines the 3D effect to be applied to the image: Flat, Relief, Recessed, Tube in Relief or Recessed Tube. Can be assigned to a whole variable or it can be managed with thresholds
X1	Horizontal coordinate of starting point
X2	Horizontal coordinate of destination point
Y1	Vertical coordinate of starting point
Y2	Vertical coordinate of destination point
LineSize	Determines the thickness of the line. The value can be assigned to a whole variable or it can be managed with thresholds
LineColor	Determines the color of the infill, which can be selected using the RGB code or color palette. The value can be assigned to a whole variable or it can be managed with thresholds

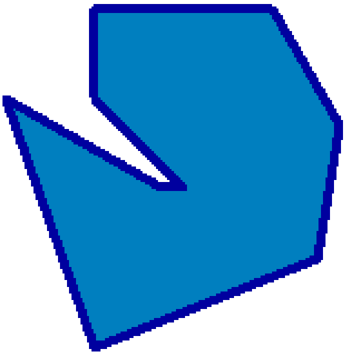
Polygon

A Polygon can be introduced into a page by clicking on the icon ▱ or using the Main Menu (Fields->Create->Simple figures->Polygon). After clicking on the icon, click on the page at the points that you want the vertices of the Polygon to appear in. POLYMATH will show the preview of the Polygon as soon as the mouse is moved. Every click made will produce a new vertex. The introduction of the Polygon is confirmed by just double-clicking it (thereby ending its edit).

Once a Polygon has been introduced, its structure (that is, its vertices) can be edited: after selecting the Polygon and then moving one of its vertices the lines (sides) adjacent to this vertex are automatically removed by POLYMATH.

Using this function, an irregular Polygon can be created, that is one having angles and sides with dimensions chosen at will. Regular polygons can also be introduced using the appropriate POLYMATH tool (see chap. 6, “Regular polygon” page 186).

To define the characteristics of the Polygon they must be set in the Properties Editor as indicated in the following section.



Properties of the Polygon

Tabella 18:

Properties	Description
Name	Identifying name of the Polygon. Must be unique among the graphic elements
Comment	Identifying comment within POLYMATH
Top	Vertical position coordinate
Left	Horizontal position coordinate
Width	Width dimension
Height	Height dimension
AreaVisibility	Determines whether the Sector has a background area (True) or should be transparent (False); a Boolean variable can be assigned to this value or it can be managed with thresholds
AreaColor	Determines the color of the Polygon, which that can be selected using the RGB code or color palette. The value can be assigned to a whole variable or it can be managed with thresholds

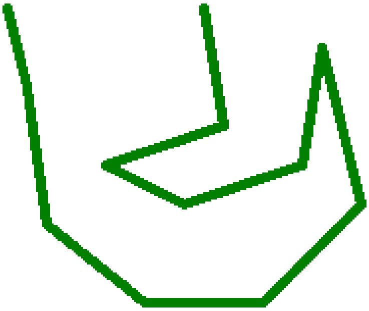
Tabella 18:

Properties	Description
LineSize	Determines the thickness of the outline of the Polygon. The value can be assigned to a whole variable or it can be managed with thresholds
LineColor	Determines the color of the Polygon outline, which can be selected using the RGB code or color palette. The value can be assigned to a whole variable or it can be managed with thresholds
PartialFillFlag	Determines whether to make a partial color infill. The value can be assigned to a whole variable
FillColor	Determines the color of the Polygon infill using the RGB code or the color palette. The value can be assigned to a whole variable
FillDir	Determines the direction of the Polygon infill. The infill can happen From Low to High, From High to Low, From Right to Left or From Left to Right. The value can be assigned to a whole variable
FillPercent	Indicates the percentage of the infill. The value can be assigned to a whole variable
NPoints	Indicates the number of sides assigned to the Polygon in the drawing phase
Hide	Determines whether the object is initially visible; it is also possible to assign a Boolean variable (dynamic in Runtime) or it can be managed with thresholds

Irregular line

A Irregular line can be introduced into a page by clicking on the icon or using the Main Menu (Fields->Create->Simple figures->Irregular line). After clicking on the icon, click on the page at the points that you want the vertices of the figure to appear (in practice, the beginning and the end of the various line sections). POLYMATH will show the preview of the line as soon as the mouse is moved. Every click made will produce a

new line sections. The introduction of the Irregular line is confirmed by just double-clicking it (thereby ending its edit).
Once an Irregular line has been introduced, its structure (that is, its vertices) can be edited: after selecting the Line and then moving one of its lines, those adjacent to this vertex are automatically removed by POLYMATH.
Using this function, an open line can be created, that differs from an irregular Polygon in that it is not necessarily closed to form a closed geometric figure.
To define the characteristics of the Irregular line they must be set in the Properties Editor as indicated in the following section.



Properties of the Irregular line

Tabella 19: Properties of the Irregular line

properties	Description
Name	Identifying name of the Irregular line. Must be unique among the graphic elements
Comment	Identifying comment within POLYMATH
Top	Vertical position coordinate
Left	Horizontal position coordinate
Width	Width dimension
Height	Height dimension

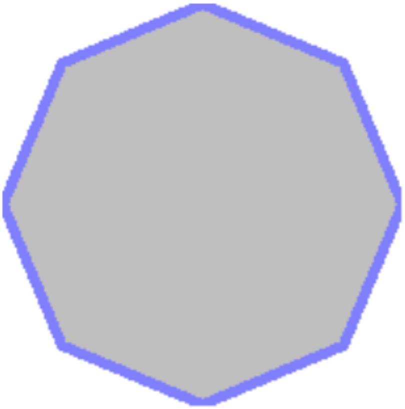
Tabella 19: Properties of the Irregular line

properties	Description
LineSize	Determines the thickness of the outline of the Line. The value can be assigned to a whole variable or it can be managed with thresholds
LineColor	Determines the color of the Line, which can be selected using the RGB code or color palette. The value can be assigned to a whole variable or it can be managed with thresholds
NPoints	Indicates the number of sides assigned to the Irregular Line in the drawing phase
Hide	Determines whether the object is initially visible; it is also possible to assign a Boolean variable (dynamic in Runtime) or it can be managed with thresholds

Regular polygon

A Regular polygon can be introduced into a page by clicking on the icon ⬡ or using the Main Menu (Fields->Create->Simple figures->Regular polygon). After clicking on the icon, use the mouse to indicate the area in the page where POLYMATH should draw the Polygon.
The default setting is that a pentagon (5 sides) is drawn; to change the number of sides (vertices) just edit the properties Number of Points using the Properties Editor (see the following section).
This function allows the creation only of regular polygons, that is, one with all the angles and sides equal. Irregular Polygons can also be introduced by using the appropriate POLYMATH tool (see chap. 6, “Polygon” page 182).

To define the characteristics of the Regular Polygon they must be set in the Properties Editor as indicated in the following section.



Properties of the Regular Polygon

The properties of the Regular polygon are identical to those of the Irregular polygon(see chap. 6, "Properties of the Polygon" page 183).

Label

A Label can be introduced into a page by clicking on the icon A or using the Main Menu (Fields->Create->Simple figures->Label). After clicking on the icon, use the mouse to indicate the area in the page where POLYMATH should draw the Polygon. A Label is a text field (may be multilanguage) into which you can introduce text strings that will not change in Runtime. To define the characteristics of the Label they must be set in the Properties Editor as indicated in the following section.

Properties of the Label

Tabella 20: Properties of the Label

Properties	Description
Name	Identifying name of the Label. Must be unique among the graphic elements
Comment	Identifying comment within POLYMATH
Top	Vertical position coordinate
Left	Horizontal position coordinate
Width	Width dimension
Height	Height dimension
Text	Text shown in the Label; by clicking on ... you can edit multilanguage texts and their related Fonts (see chap. 5, "Languages" page 78)
FontField	Font related to the text shown in the field; by clicking on ... you can edit multilanguage Fonts (see chap. 5, "Languages" page 78)
TextColor	Determines the color of the Label text, which can be selected using the RGB code or color palette. The value can be assigned to a whole variable or it can be managed with thresholds
TextHAlign	Allows you to specify the horizontal centering of the text within the Label. The value can be assigned to a whole variable or it can be managed with thresholds
TextVAlign	Allows you to specify the vertical centering of the text within the Label. The value can be assigned to a whole variable or it can be managed with thresholds
TextAutoAdjust	Determines whether automatically to distribute the text uniformly within the Label; this causes a resizing of the Label in relation to the text contained in Runtime

Tabella 20: Properties of the Label

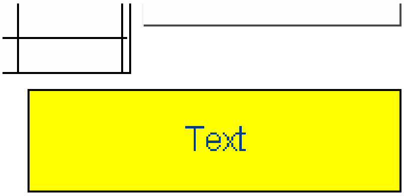
Properties	Description
TextMultiLine	Determines whether the Label text can start a new line
TextBlink	Determines the flashing of the text; the possibilities are No Blinking, Slow blinking or Rapid Blinking. The value can be assigned to a whole variable or it can be managed with thresholds
TextMaxLen	Determines the maximum value in relation to the length of the text string
TextTranslateDisable	Determines whether the translation of the Label text must be disabled
AreaVisibility	Determines whether the Sector has a background area (True) or should be transparent (False); a Boolean variable can be assigned to this value or it can be managed with thresholds
AreaColor	Determines the color of the Label, which can be selected using the RGB code or color palette. The value can be assigned to a whole variable or it can be managed with thresholds
BorderVisibility	Determines whether there will be a Border to the Label or not; a Boolean variable can be assigned to this value
BorderSize	Determines the size of the Border which must be a number to which a whole variable can be assigned, if wished, or it can be managed with thresholds
BorderColor	Determines the color of the Border using the RGB code or color palette. The value can be assigned to a whole variable or it can be managed with thresholds
BorderStyle	Determines the style of the Border, which can be Solid or Broken. The value can be assigned to a whole variable or it can be managed with thresholds

Tabella 20: Properties of the Label

Properties	Description
BorderBlink	Determines the flashing of the Border, which can be No blinking, Slow blinking or Rapid blinking. The value can be assigned to a whole variable or it can be managed with thresholds
Border3D	Determines a 3D effect for the Border, which can be Flat, Relief, Recessed, Bump or Etched. The value can be assigned to a whole variable or it can be managed with thresholds
Hide	Determines whether the object is initially visible. It is also possible to assign a Boolean variable (for changes in Runtime) or it can be managed with thresholds

Complex label

A Complex label can be introduced into a page by clicking on the icon A or using the Main Menu (Fields->Create->Simple figures->Complex label). This icon (or Menu option) is active only if it is within a Complex Control editor (see chap. 6, "Complex Controls" page 268).
After clicking on the icon, use the mouse to indicate the area in the page where POLYMATH should draw the Label. A Label is a text field (may be multilanguage) into which you can introduce text strings that will not change in Runtime.
To define the other characteristics of the Label they must be set in the Properties Editor as indicated in the following section.

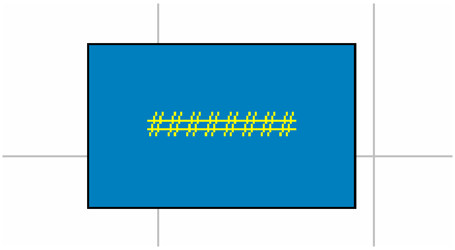


Properties of the Complex Label

The properties of the Complex label are identical to those of the Label. The reader is, therefore, advised to consult the appropriate part of the previous section (see chap. 6, "Properties of the Label" page 188).

Trend Pen

A Trend Pen can be introduced into a page by clicking on the icon abc or using the Main Menu (Fields->Create->Simple figures->Trend pen). This icon (or Menu option) is active only if it is within a Trend editor (see chap. 6, "Editing a TrendView" page 278). In practice, the Trend Pen makes it possible to view the current value next to the Pen selected, in such a way as to couple a numeric indication with the graphic display of the Trend.
After clicking on the icon, use the mouse to indicate the area in the page where POLYMATH should draw the Trend Pen.
For the characteristics of the Trend Pen to be defined, they must be set in the Properties Editor as indicated in the following section.



Properties of the Trend Pen

Tabella 21: Properties of the Trend Pen

Properties	Description
Name	Identifying name of the Trend Pen. Must be unique among the graphic elements
Comment	Identifying comment within POLYMATH
Top	Vertical position coordinate
Left	Horizontal position coordinate

Tabella 21: Properties of the Trend Pen

Properties	Description
Width	Width dimension
Height	Height dimension
PenValue	Allows you to select the type of Pen to which to assign the field
FontField	Font related to the text shown in the field; by clicking on .. you can edit multilanguage Fonts (see chap. 5, “Languages” page 78)
TextColor	Determines the color of the Field text, which can be selected using the RGB code or color palette. The value can be assigned to a whole variable
TextHAlign	Determines the type of horizontal text alignment, which can be Centered, Left or Right
TextVAlign	Determines the type of vertical text alignment, which can be Centered, Top or Bottom
Digits	Defines the maximum number of characters visible in the field representing the value
Representation	Indicates the value representation format, which will be either Decimal with or without a Sign, Hexadecimal, Binary, Floating or Fixed Point
LeadingZeroes	Indicates if zeroes should be set before the significant digits; e.g. if True 000541 will be displayed, otherwise it will simply be 541
TruncationDigits	Indicates the number of digits to be truncated when finally representing the field; the digits are truncated starting from the right (e.g. 1456 when truncation = 2 is 14)
DecimalDigits	Indicates the number of decimal digits to display if the representation format is Fixed Point

Tabella 21: Properties of the Trend Pen

Properties	Description
Picture	Indicates the layout of the representation of the numerical value; for example, if the value is 35403 and if the picture is ##!#:## the field displayed will be 35!4:03
Thousep	Indicates whether to show "thousand separators" or not
LeftMinus	Indicates the leftward positioning of the minus sign for negative values
LeftPlus	Indicates the leftward positioning of the plus sign for positive values
RightMinus	Indicates the rightward positioning of the minus sign for negative values
RightPlus	Indicates the rightward positioning of the plus sign for positive values
AreaVisibility	Determines whether the Sector has a background area (True) or should be transparent (False); a Boolean variable can be assigned to this value
AreaColor	Determines the color of the Trend Pen, which can be selected using the RGB code or color palette. The value can be assigned to a whole variable
BorderVisibility	Determines whether there will be a Border to the Field or not; a Boolean variable can be assigned to this value
BorderSize	Determines the size of the Border which must be a number to which a whole variable can be assigned, if wished
BorderColor	Determines the color of the Border using the RGB code or color palette. The value can be assigned to a whole variable
BorderStyle	Determines the style of the Border, which can be Solid or Broken. The value can be assigned to a whole variable

Tabella 21: Properties of the Trend Pen

Properties	Description
BorderBlink	Determines the flashing of the Border, which can be No blinking, Slow blinking or Rapid blinking. The value can be assigned to a whole variable
Border3D	Determines a 3D effect for the Border, which can be Flat, Relief, Recessed, Bump or Etched. The value can be assigned to a whole variable
PasswordLevel	Determines the authorization level required to access the field (see chap. 5, "Password configuration" page 81)
ReadOnly	Indicates whether the field should be editable in Runtime
Disable	Indicates whether the field should be disabled
Hide	Determines whether the object is initially visible. It is also possible to assign a Boolean variable (for changes in Runtime)
TabIndex	Makes it possible to control the focus movement when using movement keys of the cursor within a page. It also controls the order in which data is introduced in several fields when the automatic setting of the next field of the page is enabled (see chap. 5, "General" page 90)

Image Field

An Image field can be introduced into a page by clicking on the icon 🖼 or using the Main Menu (Fields->Create->Simple figures->Image). After clicking on the icon, use the mouse to indicate the area in the page where POLYMATH should draw the Image. The area created in this way will contain one of the images added to the project (see chap. 5, "Frames" page 128). To define the characteristics of the Image they must be set in the Properties Editor as indicated in the following section.

Note: *An image can also be added to a page by simply dragging it from Project Explorer into the work area to the page position required. With this procedure POLYMATH automatically creates an Image field relating to the dragged image.*

Properties of the Image field

Tabella 22: Properties of the Image Field

Properties	Description
Name	Identifying name of the Image field. Must be unique among the graphic elements
Comment	Identifying comment within POLYMATH
Top	Vertical position coordinate
Left	Horizontal position coordinate
Width	Width dimension
Height	Height dimension
Image	Reference to the image that must be contained within the Field
ImageHAlign	Indicates the type of horizontal alignment of the image within the Field, which can be Central, Leftward or Rightward

Tabella 22: Properties of the Image Field

Properties	Description
ImageVAlign	Indicates the type of vertical alignment of the image within the Field, which can be Central, Top or Bottom
ImageAutoSize	Indicates whether the image should automatically sized to fit the dimensions of the Field
ImageKeepAspect Ratio	Indicates whether the image should maintain the proportions of the source image
ImageTransparent	Indicates whether a transparency filter should be applied to the image
ImageTransColor	Indicates the color, selectable using the RGB code or the color palette, for which the transparency filter should be applied
AreaVisibility	Determines whether the Sector has a background area (True) or should be transparent (False); a Boolean variable can be assigned to this value or it can be managed with thresholds
AreaColor	Determines the color of the Image field, which can be selected using the RGB code or color palette. The value can be assigned to a whole variable or it can be managed with thresholds
BorderVisibility	Determines whether there will be a Border to the Image Field or not; a Boolean variable can be assigned to this value
BorderSize	Determines the size of the Border which must be a number to which a whole variable can be assigned, if wished, or it can be managed with thresholds
BorderColor	Determines the color of the Border using the RGB code or color palette. The value can be assigned to a whole variable or it can be managed with thresholds

Tabella 22: Properties of the Image Field

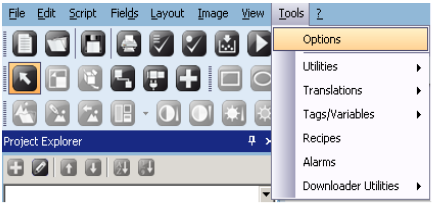
Properties	Description
BorderStyle	Determines the style of the Border, which can be Solid or Broken. The value can be assigned to a whole variable or it can be managed with thresholds
BorderBlink	Determines the flashing of the Border, which can be No blinking, Slow blinking or Rapid blinking. The value can be assigned to a whole variable or it can be managed with thresholds
Border3D	Determines a 3D effect for the Border, which can be Flat, Relief, Recessed, Bump or Etched. The value can be assigned to a whole variable or it can be managed with thresholds
Hide	Determines whether the object is initially visible. It is also possible to assign a Boolean variable (for changes in Runtime) or it can be managed with thresholds

Value fields

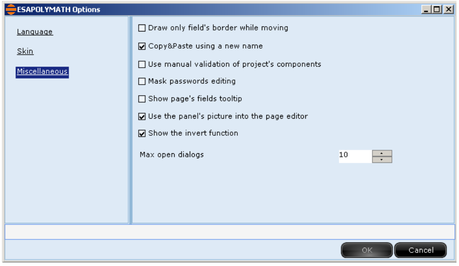
Value fields are objects (graphic) that can be inserted into a page in order to show the operator the value of an item of data (variable) or a representation of it. Some of these fields can also have their value edited by the operator. In this section we will analyze each Value field indicating its functional characteristics, its particular properties (that can be configured by the Properties Editor) and its Events (Events Editor).

"Invert Function" Option

A general property of all graphic objects (buttons, value fields, numerical fields etc.) that can be inserted in a project page is called the "Invert Function" option.
The "Invert Function" option can be associated to variables that have a Boolean behaviour (true/false) and can be used only if activated as shown hereafter.
Select the "Tools" menu and then the "Options" sub-menu :



clicking on "Various" the following image will appear :



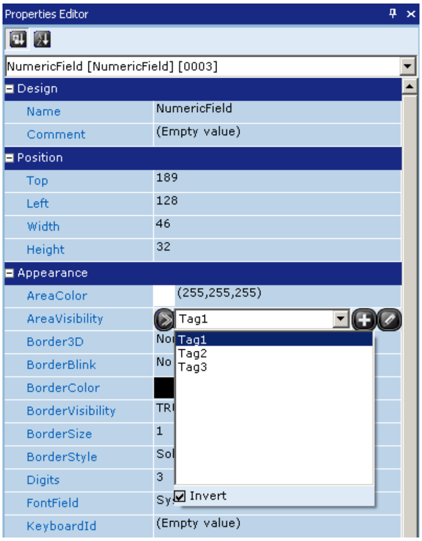
- □ Draw only field's border while moving
- ☑ Copy&Paste using a new name
- □ Use manual validation of project's components
- □ Mask passwords editing
- □ Show page's fields tooltip
- ☑ Use the panel's picture into the page editor
- ☑ Show the invert function

Max open dialogs 10

Selecting the "Show the invert function" box, the option will be activated.

"Invert Function" option operation

Select the "Invert" option in "Editor Properties", using it as an example with a numerical field associated to a "Tag" :



NumericField [NumericField] [0003]

Design	
Name	NumericField
Comment	(Empty value)
Position	
Top	189
Left	128
Width	46
Height	32
Appearance	
AreaColor	(255,255,255)
AreaVisibility	Tag1
Border3D	No
BorderBlink	No
BorderColor	■
BorderVisibility	TR
BorderSize	1
BorderStyle	Sol
Digits	3
FontField	Sy
KeyboardId	(Empty value)

☑ Invert

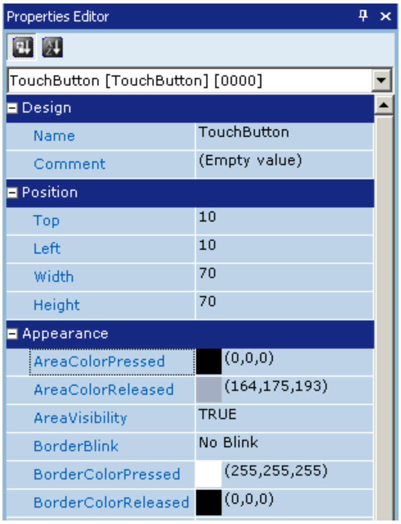
The behaviour associated to the "Tag" is inverted. For example, if a "Tag" enables the display of the background colour of the numerical field when its value is "1", selecting the "Invert" option, the numerical field background will be visible even when the value of the "Tag" is "0" and not "1".

"Gestione a Soglie" (Thresholds Management) Function

A new functionality is present inside POLYMATH (starting from version 1.7).
The new POLYMATH function, called "Soglie" (Thresholds) is present inside the Editor Properties and as been created as an additional option in order to manage the "cambio colore" (colour change), the "lampeggio" (flashing), "nascondi" (hide) and "disabilita" (disable) and other properties of the various objects.

"Soglie" (Thresholds) option functioning

To explain functioning of the "Soglie" (Thresholds) option, the "ColoreAreaPremuta" (AreaColourPressed) property of a "PulsanteSfioramento" (Touch-sensitive Button) will be taken as an example :

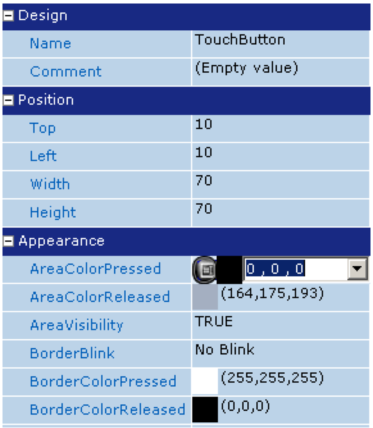


Properties Editor

TouchButton [TouchButton] [0000]

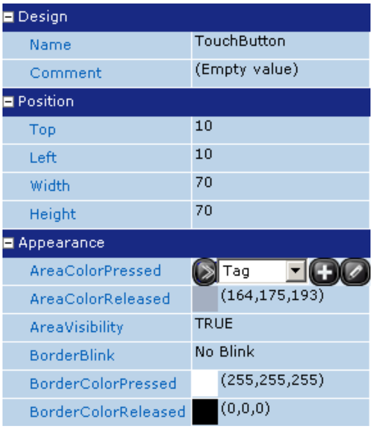
Design	
Name	TouchButton
Comment	(Empty value)
Position	
Top	10
Left	10
Width	70
Height	70
Appearance	
AreaColorPressed	■ (0,0,0)
AreaColorReleased	▨ (164,175,193)
AreaVisibility	TRUE
BorderBlink	No Blink
BorderColorPressed	□ (255,255,255)
BorderColorReleased	■ (0,0,0)

On the first click, access the immediate colour selection :



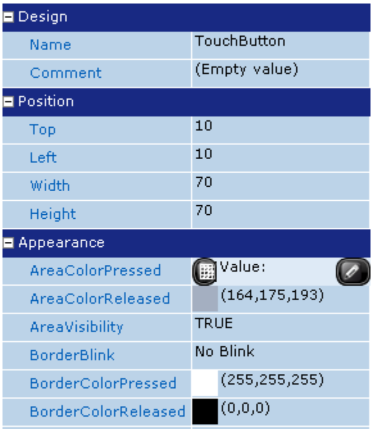
Design	
Name	TouchButton
Comment	(Empty value)
Position	
Top	10
Left	10
Width	70
Height	70
Appearance	
AreaColorPressed	■ 0 , 0 , 0 ▾
AreaColorReleased	▨ (164,175,193)
AreaVisibility	TRUE
BorderBlink	No Blink
BorderColorPressed	□ (255,255,255)
BorderColorReleased	■ (0,0,0)

With the second click, access the direct assignment to Tag :



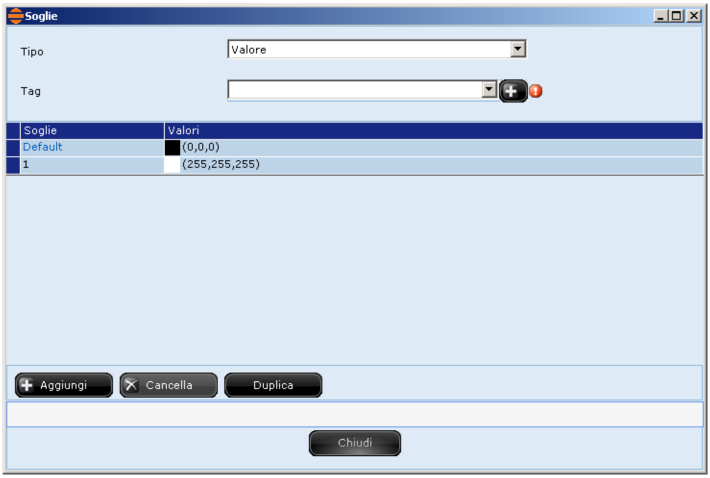
Design	
Name	TouchButton
Comment	(Empty value)
Position	
Top	10
Left	10
Width	70
Height	70
Appearance	
AreaColorPressed	Tag ▾
AreaColorReleased	▨ (164,175,193)
AreaVisibility	TRUE
BorderBlink	No Blink
BorderColorPressed	□ (255,255,255)
BorderColorReleased	■ (0,0,0)

With the third click, access the new "Soglie" (Thresholds) item:



Design	
Name	TouchButton
Comment	(Empty value)
Position	
Top	10
Left	10
Width	70
Height	70
Appearance	
AreaColorPressed	Value: ✎
AreaColorReleased	▨ (164,175,193)
AreaVisibility	TRUE
BorderBlink	No Blink
BorderColorPressed	□ (255,255,255)
BorderColorReleased	■ (0,0,0)

The following editing mask will appear by clicking on the ✎ icon :



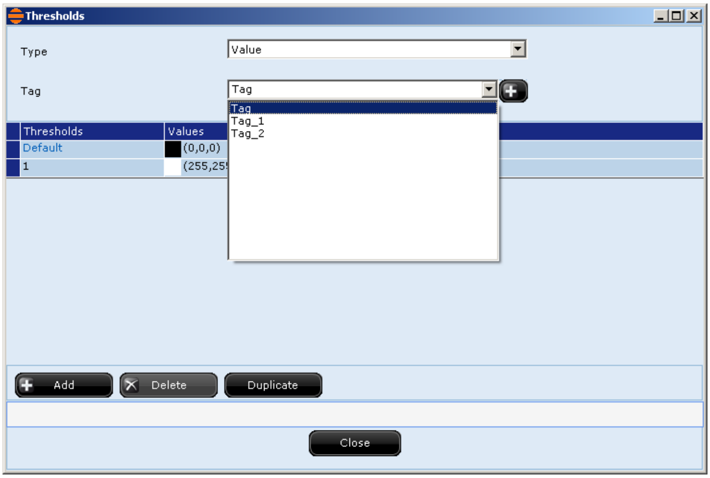
Soglie

Tipo: Valore

Tag:

Soglie	Valori
Default	■ (0,0,0)
1	□ (255,255,255)

From the previous mask, select the type of thresholds management to be performed, whether with "Valore" (Values) or "Bits". Moreover, the "ColoreAreaPremuta" (AreaColourPressed) property of the "PulsanteSfioramento" (Touch-sensitive Button) must be associated to a "Tag". In our example we have selected the "Valore" type of management. In this case the user can add all of the values he wants without any limits :

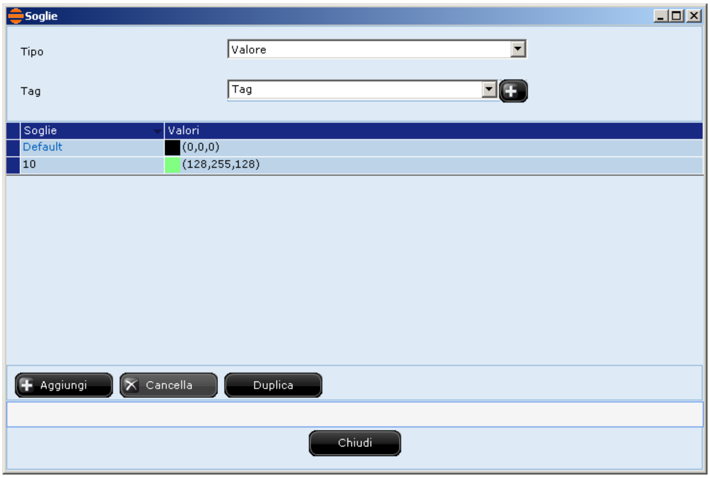


Type: Value

Tag: Tag

Thresholds	Values
Default	■ (0,0,0)
1	□ (255,25

The first threshold is assigned with the value "10", associating it to green :

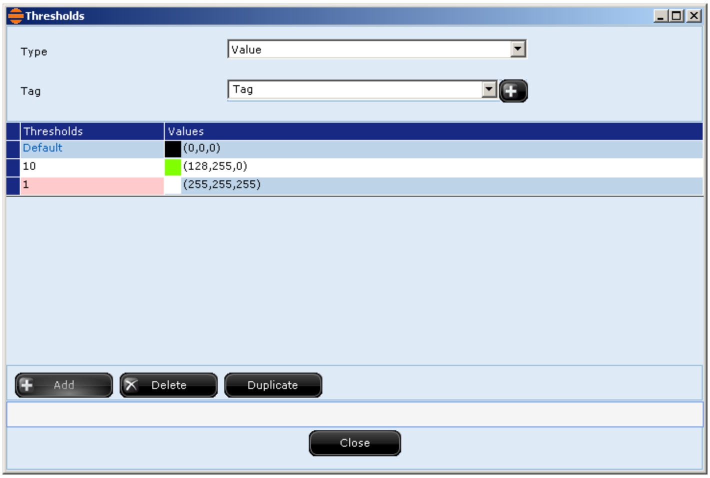


Tipo: Valore

Tag: Tag

Soglie	Valori
Default	■ (0,0,0)
10	🟩 (128,255,128)

A second threshold is now added by clicking on the "Aggiungi" (Add) button :

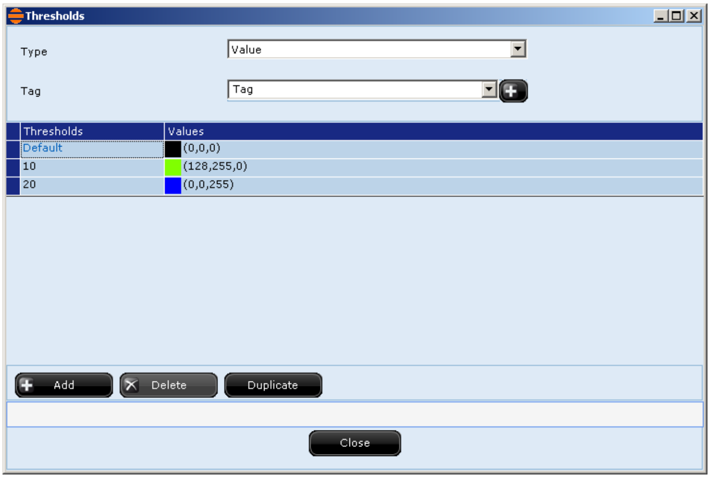


Type: Value

Tag: Tag

Thresholds	Values
Default	⬛ (0,0,0)
10	🟩 (128,255,0)
1	⬜ (255,255,255)

The second threshold is assigned with the value "20", associating it to blue :



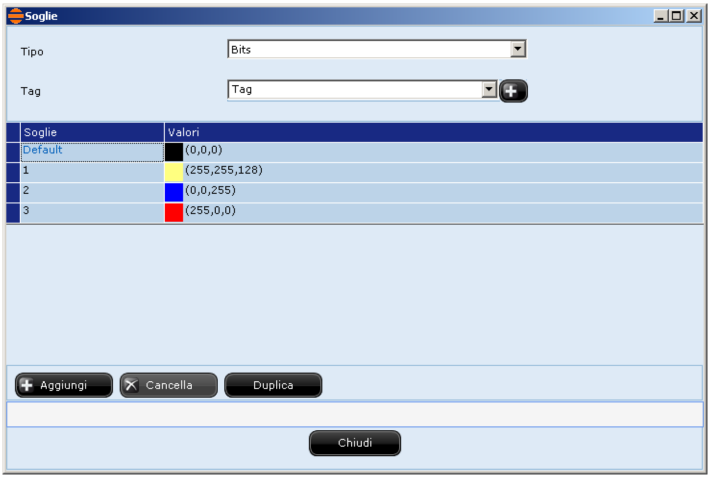
Type: Value

Tag: Tag

Thresholds	Values
Default	⬛ (0,0,0)
10	🟩 (128,255,0)
20	🟦 (0,0,255)

Clicking on the Close button to end editing.

If the user should select the "Bits" type management, the same amount of values must be introduced as there are Bits defined to which the desired settings are to be associated.
Practically, the user can assign a different colour to every Bit, for example when the second Bit is at 1, the object will be yellow. When the third Bit is at 1, the object will be blue, when the fourth Bit is at 1, the object will be red and so on.
If there are more Bits at 1, the lowest one will be considered.



Tipo: Bits

Tag: Tag

Soglie	Valori
Default	⬛ (0,0,0)
1	🟨 (255,255,128)
2	🟦 (0,0,255)
3	🟥 (255,0,0)

The Bits that the user addresses may not be adjoining.
The most insignificant Bit must be Bit "1" while the most significant Bit will depend on the length of the type of Tag associated, e.g. if the Tag is at 16 Bit, the user can insert the Bits from 1 to 16.

Objects to which the "Soglie" (Thresholds) functionality can be applied

The new "Gestione a Soglie" (Thresholds Management) functionality is supported by the following objects, with properties described below :

Rectangle

Area Colour
Area Visibility
3D Border
Border Flashing
Border Colour
Border Thickness
Border Style
Hidden

Ellipse

Area Colour
Area Visibility
Line Colour
Hidden
3D Ellipse

Arc

Line Colour
Hidden

Circular Sector

Area Colour
Area Visibility
Line Colour
Hidden

Line

3D effect
Line Colour
Line Size
Hidden

Polygon

Area Colour
Area Visibility
Line Colour
Line Size
Hidden

Polyline

Line Colour
Line Size
Hidden

Regular Polygon

Area Colour
Area Visibility
Line Colour
Line Size
Hidden

Label

Area Colour
Area Visibility
3D Border
Border Flashing
Border Colour
Border Thickness
Border Style
Text Flashing
Text Colour
Text Horizontal Alignment
Text Vertical Alignment
Hidden

Image

Area Colour
Area Visibility
3D Border
Border Flashing

Border Colour
Border Thickness
Border Style
Hidden

Numerical Field

Area Colour
Area Visibility
3D Border
Border Flashing
Border Colour
Border Thickness
Border Style
Text Flashing
Text Colour
Text Horizontal Alignment
Text Vertical Alignment
Rejected Characters
Disabled
Hidden
Mile Separator

Dynamic Text

Area Colour
Area Visibility
3D Border
Border Flashing
Border Colour
Border Thickness
Border Style
Text Flashing
Text Colour
Text Horizontal Alignment
Text Vertical Alignment
Disabled
Hidden

Ascii Field

Area Colour
Area Visibility
3D Border

Border Flashing
Border Colour
Border Thickness
Border Style
Text Flashing
Text Colour
Text Horizontal Alignment
Text Vertical Alignment
Disabled
Hidden

Symbol Field

Area Colour
Area Visibility
3D Border
Border Flashing
Border Colour
Border Thickness
Border Style
Disabled
Hidden

Date Time Field

Area Colour
Area Visibility
3D Border
Border Flashing
Border Colour
Border Thickness
Border Style
Text Flashing
Text Colour
Text Horizontal Alignment
Text Vertical Alignment
Disabled
Hidden

Bar Field

Area Colour
Area Visibility
3D Border

Border Flashing
Border Colour
Border Thickness
Bar Background Colour
Disabled
Hidden

Indicator

Area Visibility
Border Style
Hidden

Touch-sensitive Button

Area Colour Pressed
Area Colour Released
Area Visibility
Border Flashing
Border Colour Pressed
Border Colour Released
Border Thickness
3D Button
Text Flashing
Text Colour
Text Horizontal Alignment
Text Vertical Alignment
Image Horizontal Alignment
Image Vertical Alignment
Disabled
Hidden

Tactile Area

Disabled

Slide Potentiometer

Area Colour
Area Visibility
3D Border
Border Flashing
Border Colour

Border Thickness
Disabled
Hidden

Slide Selector

Area Colour
Area Visibility
3D Border
Border Flashing
Border Colour
Border Thickness
Disabled

Knob potentiometer

Area Visibility
Border Style
Disabled
Hidden

Knob Selector

Area Visibility
Border Style
Disabled
Hidden

Monostable Button

Area Colour Pressed
Area Visibility Pressed
Area Colour Released
Area Visibility Released
Disabled
Hidden

Bistable Button

Area Colour Key Off
State Area Off Present
Area Colour State On

State Area On Present
Disabled
Hidden

Frame

Background Frame Colour
Transparent Background Frame Colour
Page Edge Colour
Page Border Thickness
Page Border Style

Trend Buffer View

Area Colour
Area Visibility
3D Border
Border Flashing
Border Colour
Border Thickness
Border Style
Hidden

Trend Buffer Graphics

Area Colour
Area Visibility
3D Border
Border Flashing
Border Colour
Border Thickness
Border Style

Trend Buffer X-Y View

Area Colour
Area Visibility
3D Border
Border Flashing
Border Colour
Border Thickness
Border Style
Hidden

Trend X-Y Graphics

Area Colour
Area Visibility
3D Border
Border Flashing
Border Colour
Border Thickness
Border Style

Data Log View

Area Colour
Area Visibility
3D Border
Border Flashing
Border Colour
Border Thickness
Border Style
Hidden

Active Alarms View

Area Colour
Area Visibility
3D Border
Border Flashing
Border Colour
Border Thickness
Border Style

Historic Alarms View

Area Colour
Area Visibility
3D Border
Border Flashing
Border Colour
Border Thickness
Border Style
Hidden

User List

Area Colour
Area Visibility
3D Border
Border Flashing
Border Colour
Border Thickness
Border Style
Hidden

Recipes List

Area Colour
Area Visibility
3D Border
Border Flashing
Border Colour
Border Thickness
Border Style
Hidden

Recipe Editing

Area Colour
Area Visibility
3D Border
Border Flashing
Border Colour
Border Thickness
Border Style
Hidden

Recipe Type Name Text

Area Colour
Area Visibility
3D Border
Border Flashing
Border Colour
Border Thickness
Border Style
Text Flashing
Text Colour

Text Horizontal Alignment
Text Vertical Alignment
Disabled
Hidden

Recipe Comment

Area Colour
Area Visibility
3D Border
Border Flashing
Border Colour
Border Thickness
Border Style
Text Flashing
Text Colour
Text Horizontal Alignment
Text Vertical Alignment
Disabled
Hidden

Chronothermostat View

Area Colour
Area Visibility
3D Border
Border Flashing
Border Colour
Border Thickness
Border Style
Hidden

Chronothermostat Grid

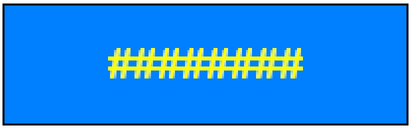
Area Colour
Area Visibility
3D Border
Border Colour
Border Thickness
Border Flashing
Vertical Scale Label Colour

Keyboard Display

Area Colour
Border Colour
Text Colour

Numerical Field

A Numerical Field can be introduced into a page by clicking on the icon 123 or using the Main Menu (Fields->Create->Value Fields->Numerical field). After clicking on the icon, use the mouse to indicate the area in the page where POLYMATH should draw the Field.
The purpose of the Numerical Field is to show the operator the updated value of a particular variable. These fields can be also be edited to become Edit value fields.
The reader is advised to consult the following subsections to learn about the details of the properties and events that can be assigned to a Numerical Field. In POLYMATH, the value of the Numerical Field is represented by a series of hash characters which in Runtime are substituted by the effective value.



__Note:__ an alternative method of creating a Numerical Field is to drag a numerical variable from the Project Explorer directly onto the destination page in the work area.

Properties of the Numerical Field

Properties	Description
Name	Identifying name of the Numerical field. Must be unique among the graphic elements
Comment	Identifying comment within POLYMATH
Top	Vertical position coordinate
Left	Horizontal position coordinate
Width	Width dimension
Height	Height dimension
TagId	Reference variable for the value to be displayed. This is a numerical variable. Using the appropriate keys you can create a new variable or edit an existing one
FontField	Font related to the text shown in the field; by clicking on .. you can edit multilanguage Fonts (see chap. 5, “Languages” page 78)
TextColor	Determines the color of the Field text which can be selected using the RGB code or color palette. The value can be assigned to a whole variable or it can be managed with thresholds
TextHAlign	Determines the type of horizontal alignment of the text; this can be Center, Left or Right
TextVAlign	Determines the type of horizontal alignment of the text; this can be Center, Top or Bottom
TextBlink	Determines the flashing of the text; the possibilities are No Blinking, Slow blinking or Rapid Blinking. The value can be associated with Tag or it can be managed with thresholds
Digits	Defines the maximum number of characters visible in the field representing the value

Properties	Description
Representation	Indicates the value representation format, which will be either Decimal with or without a Sign, Hexadecimal, Binary, Floating or Fixed Point
LeadingZeroes	Indicates if zeroes should be set before the significant digits; e.g. if True 000541 will be displayed, otherwise it will simply be 541
TruncationDigits	Indicates the number of digits to be truncated when finally representing the field; the digits are truncated starting from the right (e.g. 1456 when truncation = 2 is 14). The value can be associated with Tag or it can be managed with thresholds
DecimalDigits	Indicates the number of decimal digits to display if the representation format is Fixed Point
Picture	Indicates the layout of the representation of the numerical value; for example, if the value is 35403 and if the picture is ##!#:## the field displayed will be 35!4:03
Thousep	Indicates whether to show “thousand separators” or not. The value can be associated with Tag or it can be managed with thresholds
LeftMinus	Indicates the leftward positioning of the minus sign for negative values
LeftPlus	Indicates the leftward positioning of the plus sign for positive values
RightMinus	Indicates the rightward positioning of the minus sign for negative values
RightPlus	Indicates the rightward positioning of the plus sign for positive values
AreaVisibility	Determines whether the numerical field must have the background area (True) or if it must be transparent (False). A Boolean variable can be associated to this value or it can be managed with thresholds

Properties	Description
AreaColor	Determines the color of the Numeric Field, which can be selected using the RGB code or color palette. The value can be assigned to a whole variable or it can be managed with thresholds
BorderVisibility	Determines whether the Border of the Numeric Field is present or not; a Boolean variable can be assigned to this value
BorderSize	Determines the size of the Border which must be a number to which a whole variable can be assigned, if wished, or it can be managed with thresholds
BorderColor	Determines the color of the Border using the RGB code or color palette. The value can be assigned to a whole variable or it can be managed with thresholds
BorderStyle	Determines the style of the Border, which can be Solid or Broken. The value can be assigned to a whole variable or it can be managed with thresholds
BorderBlink	Determines the flashing of the Border, which can be No blinking, Slow blinking or Rapid blinking. The value can be assigned to a whole variable or it can be managed with thresholds
Border3D	Determines a 3D effect for the Border, which can be Flat, Relief, Recessed, Bump or Etched. The value can be assigned to a whole variable or it can be managed with thresholds
PasswordLevel	Determines the authorization level required to be able to edit the field (see chap. 5, “Password configuration” page 81). This property is ignored if the field is Read Only
ReadOnly	Indicates whether the field should be editable in Runtime
Disable	Indicates whether the field should be disabled

Properties	Description
Hide	Determines whether the object is initially visible. You can also assign a Boolean variable (for Runtime changes) or it can be managed with thresholds
TabIndex	Makes it possible to control the focus movement when using movement keys of the cursor within a page. It also controls the order in which data is introduced in several fields when the automatic setting of the next field of the page is enabled (see chap. 5, “General” page 90)

Numerical Field events

Tabella 23: Numerical Field events

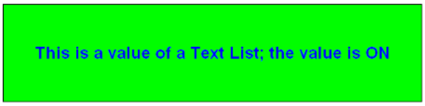
Event	Description
OnBeginInput	Activated when data input using the keyboard starts
OnAbortInput	Activated when data input operation is ended
OnValueChange	Activated when the value of the Field is changed using the keyboard

Dynamic Text

A Dynamic text can be introduced into a page by clicking on the icon A or using the Main Menu (Fields->Create->Value Fields->Dynamic text). After clicking on the icon, use the mouse to indicate the area in the page where POLYMATH should draw the Field.

The purpose of a Dynamic text is to show the operator a given string as the function of the value of a variable. Which string contained in a Text List is displayed depends on the value of the variable, (see chap. 5, “Text list” page 113). For example the words “On” or “Off” can be shown as a function of a Boolean variable.

The reader is advised to consult the following subsections to learn about the details of the properties and events that can be assigned to a Dynamic text. In POLYMATH, the value of a Dynamic text is represented by the first value of the text list which in Runtime is substituted by the correct value.



This is a value of a Text List; the value is ON

Properties of the Dynamic Text

Tabella 24: Properties of the Dynamic Text

Properties	Description
Name	Identifying name of the Dynamic text. Must be unique among the graphic elements
Comment	Identifying comment within POLYMATH
Top	Vertical position coordinate
Left	Horizontal position coordinate
Width	Width dimension
Height	Height dimension
FontField	Font related to the text shown in the field; by clicking on … you can edit multilanguage Fonts (see chap. 5, “Password configuration” page 81)
TextColor	Determines the color of the Field text, which can be selected using the RGB code or color palette. The value can be assigned to a whole variable or it can be managed with thresholds
TextHAlign	Determines the type of horizontal alignment of the text; this can be Center, Left or Right. The value can be associated with Tag or it can be managed with thresholds

Tabella 24: Properties of the Dynamic Text

Properties	Description
TextVAlign	Determines the type of horizontal alignment of the text; this can be Center, Top or Bottom. The value can be associated with Tag or it can be managed with thresholds
TextMultiLine	Indicates whether the text can be arranged on more than one line; if set as False, the excess text is cut in Runtime
TextBlink	Indicates the blinking mode of the text displayed; can be No Blinking, Slow Blinking or Rapid Blinking
TextMaxLen	Indicates the maximum length of the text
TextTranslateDisable	Determines whether the translation of the Label text must be disabled
TagId	Reference variable for the value to be displayed. This is a numerical variable. Using the appropriate keys you can create a new variable or edit an existing one
TextListId	Defines the text list from which the string to be displayed will be selected in Runtime. Using the appropriate keys you can create a new list or edit an existing one (see chap. 5, "Text list" page 113)
ControlType	Indicates the type of control to exercise over the control variable; this can be value-oriented, single-bit orientated or bit-group-oriented.
FirstBit	Active if the type of control is single-bit orientated or bit-group-oriented. It indicates the bit reference to apply the control to (or the group initial reference if the control relates to a group)
LastBit	Active if the type of control is single-bit orientated. Indicates the last bit of the group to apply the control to

Tabella 24: Properties of the Dynamic Text

Properties	Description
Value	Active if the type of control is value-orientated; the values on which to apply the list strings must be indicated. By clicking on ... you can access the mask associating values and elements of the text list
AreaVisibility	Determines whether the Sector has a background area (True) or should be transparent (False); a Boolean variable can be assigned to this value . The value can be associated with Tag or it can be managed with thresholds
AreaColor	Determines the color of the Dynamic text, which can be selected using the RGB code or color palette. The value can be assigned to a whole variable . The value can be associated with Tag or it can be managed with thresholds
BorderVisibility	Determines whether there will be a Border to the Dynamic text or not; a Boolean variable can be assigned to this value
BorderSize	Determines the size of the Border which must be a number to which a whole variable can be assigned, if wished, or it can be managed with thresholds
BorderColor	Determines the color of the Border using the RGB code or color palette. The value can be assigned to a whole variable or it can be managed with thresholds
BorderStyle	Determines the style of the Border, which can be Solid or Broken. The value can be assigned to a whole variable or it can be managed with thresholds
BorderBlink	Determines the flashing of the Border, which can be No blinking, Slow blinking or Rapid blinking. The value can be assigned to a whole variable or it can be managed with thresholds

Tabella 24: Properties of the Dynamic Text

Properties	Description
Border3D	Determines a 3D effect for the Border, which can be Flat, Relief, Recessed, Bump or Etched. The value can be assigned to a whole variable or it can be managed with thresholds
PasswordLevel	Determines the authorization level required to be able to edit the field (see chap. 5, "Password configuration" page 81). This property is ignored if the field is Read Only
ReadOnly	Indicates whether the field should be editable in Runtime
Disable	Indicates whether the field should be disabled. The value can be associated with Tag or it can be managed with thresholds
Hide	Determines whether the object is initially visible. You can also assign a Boolean variable (for Runtime changes) or it can be managed with thresholds
TabIndex	Makes it possible to control the focus movement when using movement keys of the cursor within a page. It also controls the order in which data is introduced in several fields when the automatic setting of the next field of the page is enabled (see chap. 5, "General" page 90)

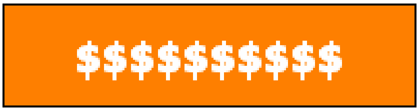
Dynamic Text Events

Tabella 25: Dynamic Text Events

Event	Description
OnBeginInput	Activated when data input using the keyboard starts
OnAbortInput	Activated when data input operation is ended
OnValueChange	Activated when the value of the Field is changed using the keyboard

ASCII Field

An ASCII field can be introduced into a page by clicking on the icon abc or using the Main Menu (Fields->Create->Value Fields->ASCII field). After clicking on the icon, use the mouse to indicate the area in the page where POLYMATH should draw the Field.
ASCII fields tell the operator the updated value of a particular String variable. These fields can also be edited thereby becoming value changing fields.
The reader is advised to consult the following subsections to learn about the details of the properties and events that can be assigned to an ASCII Field. In POLYMATH, the value of an ASCII field is represented by a series of dollar symbols ($) that can be substituted by the effective value in Runtime.



$$$$$$$$$$

Note: *An alternative method of creating an ASCII Field is to drag a string variable from the Project Explorer directly onto the destination page in the work area.*

Properties of the ASCII Field

Tabella 26: Properties of the ASCII Field

Properties	Description
Name	Identifying name of the ASCII field. Must be unique among the graphic elements
Comment	Identifying comment within POLYMATH
Top	Vertical position coordinate
Left	Horizontal position coordinate
Width	Width dimension
Height	Height dimension

Tabella 26: Properties of the ASCII Field

Properties	Description
TagId	Reference variable for the value to be displayed. This is a numerical variable. Using the appropriate keys you can create a new variable or edit an existing one
FontField	Font related to the text shown in the field; by clicking on ... you can edit multilanguage Fonts (see chap. 5, “Languages” page 78)
TextColor	Determines the color of the Field text, which can be selected using the RGB code or color palette. The value can be assigned to a whole variable or it can be managed with thresholds
TextHAlign	Determines the type of horizontal alignment of the text; this can be Center, Left or Right. The value can be associated with Tag or it can be managed with thresholds
TextVAlign	Determines the type of horizontal alignment of the text; this can be Center, Top or Bottom. The value can be associated with Tag or it can be managed with thresholds
TextBlink	Determines the flashing of the text; the possibilities are No Blinking, Slow blinking or Rapid Blinking. The value can be associated with Tag or it can be managed with thresholds
AsciiLen	Determines the maximum length of the string represented in the Field
AreaVisibility	Determines whether the Sector has a background area (True) or should be transparent (False); a Boolean variable can be assigned to this value or it can be managed with thresholds
AreaColor	Determines the color of the ASCII field, which can be selected using the RGB code or color palette. The value can be associated with Tag or it can be managed with thresholds

Tabella 26: Properties of the ASCII Field

Properties	Description
BorderVisibility	Determines whether there will be a Border to the ASCII field or not; a Boolean variable can be assigned to this value
BorderSize	Determines the size of the Border which must be a number to which a whole variable can be assigned, if wished, or it can be managed with thresholds
BorderColor	Determines the color of the Border using the RGB code or color palette. The value can be assigned to a whole variable or it can be managed with thresholds
BorderStyle	Determines the style of the Border, which can be Solid or Broken. The value can be assigned to a whole variable or it can be managed with thresholds
BorderBlink	Determines the flashing of the Border, which can be No blinking, Slow blinking or Rapid blinking. The value can be assigned to a whole variable or it can be managed with thresholds
Border3D	Determines a 3D effect for the Border, which can be Flat, Relief, Recessed, Bump or Etched. The value can be assigned to a whole variable or it can be managed with thresholds
PasswordLevel	Determines the authorization level required to be able to edit the field (see chap. 5, “Password configuration” page 81). This property is ignored if the field is Read Only
ReadOnly	Indicates whether the field should be editable in Runtime
Disable	Indicates whether the field should be disabled. The value can be associated with Tag or it can be managed with thresholds

Tabella 26: Properties of the ASCII Field

Properties	Description
Hide	Determines whether the object is initially visible. You can also assign a Boolean variable (for Runtime changes) or it can be managed with thresholds
TabIndex	Makes it possible to control the focus movement when using movement keys of the cursor within a page. It also controls the order in which data is introduced in several fields when the automatic setting of the next field of the page is enabled (see chap. 5, "General" page 90)

ASCII Field events

Tabella 27: ASCII Field events

Events	Properties
OnBeginInput	Activated when data input using the keyboard starts
OnAbortInput	Activated when data input operation is ended
OnValueChange	Activated when the value of the Field is changed using the keyboard

Symbol Field

A Symbol field can be introduced into a page by clicking on the icon ★ or using the Main Menu (Fields->Create->Value fields->Symbol field). After clicking on the icon, use the mouse to indicate the area in the page where POLYMATH should draw the Field.
The Symbol field serves to indicate to the operator a given image according to the value of a related variable; depending on the value of the variable, an image contained in a list of images is displayed (see chap. 5, "Image list" page 114). For example, the image of a led that may be ON or OFF can be shown, according to the Boolean variable.

The reader is advised to consult the following subsections to learn about the details of the properties and events that can be assigned to a Symbol field. In POLYMATH, the Symbol field value is represented by the first image in the image list will be replaced by the correct image in Runtime.

Properties of the Symbol Field

Tabella 28: Properties of the Symbol Field

Properties	Description
Name	Identifying name of the Symbol field. Must be unique among the graphic elements
Comment	Identifying comment within POLYMATH
Top	Vertical position coordinate
Left	Horizontal position coordinate
Width	Width dimension
Height	Height dimension
ImageHAlign	Indicates the type of horizontal alignment of the image within the Field, which can be Center, Left or Right
ImageVAlign	Indicates the type of vertical alignment of the image within the Field, which can be Center, Top or Bottom
ImageAutoSize	Indicates whether the image should automatically sized to fit the dimensions of the Field

Tabella 28: Properties of the Symbol Field

Properties	Description
ImageKeepAspectRatio	Indicates whether the image should maintain the proportions of the source image
ImageTransparent	Indicates whether a transparency filter should be applied to the image
ImageTransColor	Indicates the color, selectable using the RGB code or the color palette, for which the transparency filter should be applied
TagId	Reference variable for the value to be displayed. This is a numerical variable. Using the appropriate keys you can create a new variable or edit an existing one
ControlType	Indicates the type of control to exercise over the control variable; this can be value-oriented, single-bit orientated or bit-group-oriented.
ImageListId	Indicates the list of images from which a Runtime selection of the image to be displayed is made. Using the appropriate keys a new list can be created or an existing one edited (see chap. 5, “Image list” page 114)
FirstBit	Active if the type of control is single-bit orientated or bit-group-oriented. It indicates the bit reference to apply the control to (or the group initial reference if the control relates to a group)
LastBit	Active if the control is bit-group oriented. Indicates the last bit of the group to which the control is applied
Value	Active if the type of control is value-oriented; it is necessary to indicate the values on which to apply the strings in the list. By clicking on ... you access the mask for associating values with elements in the list of images

Tabella 28: Properties of the Symbol Field

Properties	Description
AreaVisibility	Determines whether the Sector has a background area (True) or should be transparent (False); a Boolean variable can be assigned to this value or it can be managed with thresholds
AreaColor	Determines the color of the Symbol Field, which can be selected using the RGB code or color palette. The value can be assigned to a whole variable or it can be managed with thresholds
BorderVisibility	Determines whether there will be a Border to the Symbol Field or not; a Boolean variable can be assigned to this value
BorderSize	Determines the size of the Border which must be a number to which a whole variable can be assigned, if wished, or it can be managed with thresholds
BorderColor	Determines the color of the Border using the RGB code or color palette. The value can be assigned to a whole variable or it can be managed with thresholds
BorderStyle	Determines the style of the Border, which can be Solid or Broken. The value can be assigned to a whole variable or it can be managed with thresholds
BorderBlink	Determines the flashing of the Border, which can be No blinking, Slow blinking or Rapid blinking. The value can be assigned to a whole variable or it can be managed with thresholds
Border3D	Determines a 3D effect for the Border, which can be Flat, Relief, Recessed, Bump or Etched. The value can be assigned to a whole variable or it can be managed with thresholds

Tabella 28: Properties of the Symbol Field

Properties	Description
PasswordLevel	Determines the authorization level required to be able to edit the field (see chap. 5, "Password configuration" page 81). This property is ignored if the field is Read Only
ReadOnly	Indicates whether the field should be editable in Runtime
Disable	Indicates whether the field should be disabled. The value can be associated with Tag or it can be managed with thresholds
Hide	Determines whether the object is initially visible. You can also assign a Boolean variable (for Runtime changes) or it can be managed with thresholds
TabIndex	Makes it possible to control the focus movement when using movement keys of the cursor within a page. It also controls the order in which data is introduced in several fields when the automatic setting of the next field of the page is enabled (see chap. 5, "General" page 90)

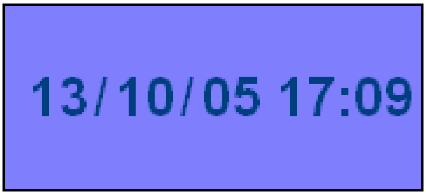
Symbol Field events

Tabella 29: Symbol Field events

Event	Description
OnBeginInput	Activated when data input using the keyboard starts
OnAbortInput	Activated when data input operation is ended
OnValueChange	Activated when the value of the field is changed using the keyboard

DateTime field

A DateTime field can be introduced into a page by clicking on the icon 01:12 or using the Main Menu (Fields->Create->Value fields->DateTime field). After clicking on the icon, use the mouse to indicate the area in the page where POLYMATH should draw the Field.
The DateTime field serves to indicate to the operator the current date and/or time while a project is running. A variable can be associated to the field, like Long or UnsignedLong (e.g. to use the data set in the device) or the system variable SYS_DateAndTime that shows the time of the operating system of the panel (see chap.�due0060Appendix A - System Variables" page 555). To express the day or the month in letters rather than in numbers, the corresponding translations must be given in appropriate text lists (see chap. 5, "Text list" page 113).
The reader is advised to consult the following subsections to learn about the details of the properties and events that can be assigned to a DateTime field. In POLYMATH, the value of the DateTime field is represented by the DateTime of the operating system of the machine the programmer is using.



Properties of the DateTime field

Tabella 30: Properties of the DateTime field

Properties	Description
Name	Identifying name of the Date-Time field. Must be unique among the graphic elements
Comment	Identifying comment within POLYMATH
Top	Vertical position coordinate
Left	Horizontal position coordinate
Width	Width dimension
Height	Height dimension

Tabella 30: Properties of the DateTime field

Properties	Description
TagId	Reference variable for the data value to be displayed. The variable selected can be Long or Unsigned Long. Using the appropriate keys you can create a new variable or edit an existing one
FontField	Font related to the text shown in the field; by clicking on ... you can edit multilanguage Fonts (see chap. 5, “Languages” page 78)
TextColor	Determines the color of the Text field, which can be selected using the RGB code or color palette. The value can be assigned to a whole variable or it can be managed with thresholds
TextHAlign	Determines the type of horizontal alignment of the text; this can be Center, Left or Right. The value can be associated with Tag or it can be managed with thresholds
TextVAlign	Determines the type of horizontal alignment of the text; this can be Center, Top or Bottom. The value can be associated with Tag or it can be managed with thresholds
TextBlink	Determines the flashing of the text; the possibilities are No Blinking, Slow blinking or Rapid Blinking. The value can be associated with Tag or it can be managed with thresholds
DateRepresentation	Indicates the format of the date to be shown; the order and the layout of the day, month and year can be selected. You can choose whether or not to assign a text list to the value of the month so as to display the string related to the current month (see chap. 5, “Text list” page 113)
TimeRepresentation	Indicates the format of the date to be shown; you can indicate whether or not to insert the seconds and whether to create an AM/PM type of display

Tabella 30: Properties of the DateTime field

Properties	Description
TypeOfDayOfWeek	Indicates the display related to the day of the week; it is possible not to display anything, to show an ordinal number (Sunday being 0, etc.) or to assign a text list to the number so as to view (also in multilanguage) the current day (see chap. 5, "Text list" page 113)
DayOfWeek	Active if the TypeOfDayOfWeek is set as a text list. This property indicates the text list assigned to the day of the week; a text list of 7 values must be created (see chap. 5, "Text list" page 113)
Month	Active if the DateRepresentation display is to be the full name of the month. This property indicates the text list assigned to the month; a text list of 12 values must be created (see chap. 5, "Text list" page 113)
DatePosition	Indicates the position of the Date within the field: this may be centered, leftward or rightward
TimePosition	Indicates the position of the Time within the field: this may be centered, leftward or rightward
DayOfWeekPosition	Indicates the position of the text related to the Day (if desired) within the field: this may be centered, leftward or rightward
AreaVisibility	Determines whether the Field has a background area (True) or should be transparent (False); a Boolean variable can be assigned to this value or it can be managed with thresholds
AreaColor	Determines the color of the Date/Time field, which can be selected using the RGB code or color palette. The value can be assigned to a whole variable or it can be managed with thresholds

Tabella 30: Properties of the DateTime field

Properties	Description
BorderVisibility	Determines whether there will be a Border to the Date/Time field or not; a Boolean variable can be assigned to this value
BorderSize	Determines the size of the Border which must be a number to which a whole variable can be assigned, if wished, or it can be managed with thresholds
BorderColor	Determines the color of the Border using the RGB code or color palette. The value can be assigned to a whole variable or it can be managed with thresholds
BorderStyle	Determines the style of the Border, which can be Solid or Broken. The value can be assigned to a whole variable or it can be managed with thresholds
BorderBlink	Determines the flashing of the Border, which can be No blinking, Slow blinking or Rapid blinking. The value can be assigned to a whole variable or it can be managed with thresholds
Border3D	Determines a 3D effect for the Border, which can be Flat, Relief, Recessed, Bump or Etched. The value can be assigned to a whole variable or it can be managed with thresholds
PasswordLevel	Determines the authorization level required to be able to edit the field (see chap. 5, “Password configuration” page 81). This property is ignored if the field is Read Only
ReadOnly	Indicates whether the field should be editable in Runtime
Disable	Indicates whether the field should be disabled. The value can be associated with Tag or it can be managed with thresholds

Tabella 30: Properties of the DateTime field

Properties	Description
Hide	Determines whether the object is initially visible. You can also assign a Boolean variable (for Runtime changes) or it can be managed with thresholds
TabIndex	Makes it possible to control the active focus using cursor keys within a page. It also controls the order in which data is introduced in several fields when the automatic setting of the next field of the page is enabled (see chap. 5, "General" page 90)

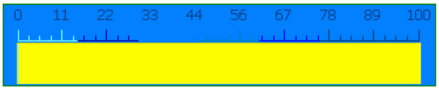
DateTime field events

Tabella 31: DateTime field events

Event	Description
OnBeginInput	Activated when data input using the keyboard starts
OnAbortInput	Activated when data input operation is ended
OnValueChange	Activated when the value of the field is changed using the keyboard

Bar Field

A Bar field can be introduced into a page by clicking on the icon or using the Main Menu (Fields->Create->Value Fields->Bar field). After clicking on the icon, use the mouse to indicate the area in the page where POLYMATH should draw the Field.
The Bar field serves to give a graphic indication of the value of a variable within a Scroll bar guided by a scale of values. If the field is editable, the operator can change the value simply by moving the pointer onto desired the Scale value.
The reader is advised to consult the following subsections to learn about the details of the properties and events that can be assigned to a Bar field.



Properties of the Bar field

Tabella 32: Properties of Bar field

Properties	Description
Name	Identifying name of the Bar field. Must be unique among the graphic elements
Comment	Identifying comment within POLYMATH
Top	Vertical position coordinate
Left	Horizontal position coordinate
Width	Width dimension
Height	Height dimension
AreaColor	Determines the color of the Bar field, which that can be selected using the RGB code or color palette. The value can be assigned to a whole variable or it can be managed with thresholds
AreaVisibility	Determines whether the Sector has a background area (True) or should be transparent (False); a Boolean variable can be assigned to this value or it can be managed with thresholds
Border3D	Determines a 3D effect for the Border, which can be Flat, Relief, Recessed, Bump or Etched. The value can be assigned to a whole variable or it can be managed with thresholds
BorderBlink	Determines the flashing of the Border, which can be No blinking, Slow blinking or Rapid blinking. The value can be assigned to a whole variable or it can be managed with thresholds

Tabella 32: Properties of Bar field

Properties	Description
BorderColor	Determines the color of the Border using the RGB code or color palette. The value can be assigned to a whole variable or it can be managed with thresholds
BorderVisibility	Determines whether there will be a Border to the Bar Field or not; a Boolean variable can be assigned to this value
BorderSize	Determines the size of the Border which must be a number to which a whole variable can be assigned, if wished, or it can be managed with thresholds
FontSize	Indicates the size of the character of the values written above the numerical division lines
ScaleNotches	Indicates the number of subdivision marks appearing between two numerical divisions. These are shorter division lines than the numerical ones, giving greater precision to the representation
ScaleColorRanges	Indicates the color ranges to be assigned to given value intervals within the scale. By clicking on ... you enter an editing window in which the value intervals and their respective colors can be defined; the window also allows you only to specify the limits in relation to the scale to be displayed
ScaleSectors	Indicates the number of notches on the scale of values. You will also see the figure for the value above the notch (calculated based on the number of notches)
TagId	Reference variable corresponding to the position of the indicator. Using the appropriate keys you can create a new variable or editor an existing one

Tabella 32: Properties of Bar field

Properties	Description
AlignBarColorToScale Color	Allows to align, or not, the color of the bar to the color of the scale
BarBackgroundColor	Allows you to assign a color to the bar background. The value can be associated with Tag or it can be managed with thresholds
Direction	Indicates the direction of the Bar: whether vertical or horizontal
Disable	Indicates whether the field should be disabled. The value can be associated with Tag or it can be managed with thresholds
FillBarColorRanges	Determines the color of the filling of the bar through code or RGB color palette. The value can be associated with integer variable.
Hide	Determines whether the object is initially visible. You can also assign a Boolean variable (for Runtime changes) or it can be managed with thresholds
Lock	Determines if the object can move or not
PasswordLevel	Determines the authorization level required to be able to edit the field (see chap. 5, “Password configuration” page 81). This property is ignored if the field is Read Only
TabIndex	Makes it possible to control the focus movement when using movement keys of the cursor within a page. It also controls the order in which data is introduced in several fields when the automatic setting of the next field of the page is enabled (see chap. 5, “General” page 90)
BarOrigin	Indicates from which value the Bar value count should start

Tabella 32: Properties of Bar field

Properties	Description
ScaleValueColor	Color of the values (figures) related to the numerical subdivisions of the scale. This can be selected using the RGB code or the color palette
ScalePosition	Indicates where the scale of values should be positioned in relation to the Bar. If the Bar is vertical, the scale can be positioned to the left or the right; if it is horizontal, the scale can be above or below
TypeOfMovement	Allows you to associate to an object a type of movement

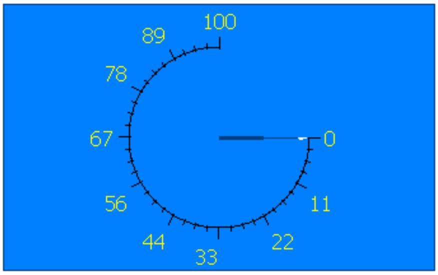
Bar field events

Tabella 33: Bar field events

Event	Description
OnValueChange	Activated when the value of the Field is changed using the touch screen

Indicator

An Indicator can be introduced into a page by clicking on the icon or using the Main Menu (Fields->Create->Value fields->Pointer). After clicking on the icon, use the mouse to define the area in the page where POLYMATH should draw the Indicator.
The Indicator gives a graphic representation of the value of a variable within a given scale of values. Unlike the Bar, the Indicator cannot be edited and has a different graphic form.
The reader is advised to consult the following subsections to learn about the details of the properties and events that can be assigned to an Indicator.



Properties of the Indicator

Tabella 34: Properties of the Indicator

Properties	Description
Name	Identifying name of the Indicator. Must be unique among the graphic elements
Comment	Identifying comment within POLYMATH
Top	Vertical position coordinate
Left	Horizontal position coordinate
Width	Width dimension
Height	Height dimension
TagId	Reference variable corresponding to the position of the indicator. Using the appropriate keys you can create a new variable or edit an existing one
StartAngle	Determines the Indicator start position (given as an angle)
SweepAngle	Determines the angle (in degrees) of the aperture of the Indicator
IndicatorColor	Determines the color of the Indicator (hand) using the RGB code or the color palette
TipColor	Determines the color of the Indicator (hand) using the RGB code or color palette. The value can be assigned to a whole variable

Tabella 34: Properties of the Indicator

Properties	Description
AreaVisibility	Determines whether the Sector has a background area (True) or should be transparent (False); a Boolean variable can be assigned to this value or it can be managed with thresholds
AreaColor	Determines the color of the Indicator area, which can be selected using the RGB code or color palette. The value can be assigned to a whole variable
ScaleValues	Indicates the number of divisions on the scale of values. The number relating to the value above the division is also displayed (calculated according to the number of divisions)
ScaleNotches	Indicates the number of subdivision marks appearing between two numerical divisions. These are shorter division lines than the numerical ones, giving greater precision to the representation
ScaleColorRanges	Indicates the color ranges to be assigned to given value intervals within the scale. By clicking on .. you enter an editing window in which the value intervals and their respective colors can be defined; the window also allows you only to specify the limits in relation to the scale to be displayed
ScaleValueColor	Color of the values (figures) related to the numerical subdivisions of the scale. This can be selected using the RGB code or the color palette
FontSize	Indicates the size of the character of the values written above the numerical division lines
BorderVisibility	Determines whether there will be a Border to the Indicator or not; a Boolean variable can be assigned to this value

Tabella 34: Properties of the Indicator

Properties	Description
BorderSize	Determines the size of the Border which must be a number to which a whole variable can be assigned, if wished
BorderColor	Determines the color of the Border using the RGB code or color palette. The value can be assigned to a whole variable
BorderStyle	Determines the style of the Border, which can be Solid or Broken. The value can be assigned to a whole variable or it can be managed with thresholds
BorderBlink	Determines the flashing of the Border, which can be No blinking, Slow blinking or Rapid blinking. The value can be assigned to a whole variable
Border3D	Determines a 3D effect for the Border, which can be Flat, Relief, Recessed, Bump or Etched. The value can be assigned to a whole variable
Hide	Determines whether the object is initially visible. You can also assign a Boolean variable (for Runtime changes) or it can be managed with thresholds

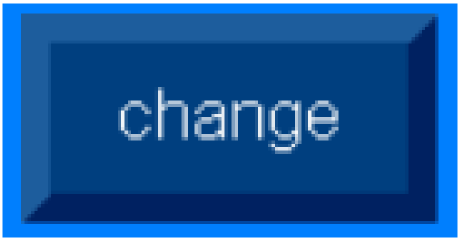
Simple Controls

Simple Controls are objects that can be inserted into a page to show the operator the value of an item of data (variable) and/or edit it. In this section we will analyze each Simple Control, identifying their functional characteristics, their properties (configurable using the Properties Editor) and their associated events (Events Editor).

Touch Button

A Touch Button can be introduced into a page by clicking on the icon ● or using the Main Menu (Fields->Create->Simple controls->TouchButton). After clicking on the icon, use the mouse to indicate the area in the page where POLYMATH should draw the button.
Touch Buttons are useful as they allow the operator to assign a given function or script user with a single click. For further details regarding scripts and predefined functions, the reader

is advised to consult the relevant section of this manual (see chap.``Appendix B - Predefined functions” page 563 and see chap. 9, “Scripts” page 379).
The reader is advised to consult the following subsections to learn about the details of the properties and events that can be assigned to a button.



Properties of the Touch button

Tabella 35: Properties of the Touch button

Properties	Description
Name	Identifying name of the Touch button. Must be unique among the graphic elements
Comment	Identifying comment within POLYMATH
Top	Vertical position coordinate
Left	Horizontal position coordinate
Width	Width dimension
Height	Height dimension
Caption	Indicates the type of inscription to display on the button: None, Label or Text list
Text	Active only if Caption is set for Label; makes it possible to indicate the text to be applied to the button. Inside a multilanguage project, click on ● to edit texts in any language (see chap. 5, ”Languages” page 78)

Tabella 35: Properties of the Touch button

Properties	Description
FontField	Font related to the text shown in the field; by clicking on ... you can edit multilanguage Fonts (see chap. 5, “Languages” page 78)
TextColor	Determines the color of the text of the button, which can be selected using the RGB code or color palette. The value can be assigned to a whole variable or it can be managed with thresholds
TextHAlign	Allows you to specify the horizontal centering of the text within the Label. The value can be assigned to a whole variable or it can be managed with thresholds
TextVAlign	Allows you to specify the vertical centering of the text within the Label. The value can be assigned to a whole variable or it can be managed with thresholds
TextMultiLine	Determines whether the Label text can start a new line
TextBlink	Determines the flashing of the text; the possibilities are No Blinking, Slow blinking or Rapid Blinking. The value can be assigned to a whole variable or it can be managed with thresholds
TextMaxLen	Determines the maximum value in relation to the length of the text string
TextTranslateDisable	Determines whether the translation of the Button text must be disabled
TextTagId	Active only if Caption is a Text List type. Determines the variable to choose from the text list (see chap. 5, “Text list” page 113)
TextListId	Active only if Caption is a Text List type. Determines the reference text list (see chap. 5, “Text list” page 113)

Tabella 35: Properties of the Touch button

Properties	Description
TextListType	Indicates which type of check to perform on the control variable: can be value-oriented, single-bit or bit-group-oriented
TextFirstBit	Active if TextListType is single-bit or bit-group-oriented. Indicates the reference bit to be checked (or the group initial reference if the control relates to a group)
TextLastBit	Active if TextListType is bit-group-oriented, indicates the last bit of the group to which the control is applied.
TextListValue	Active if TextListType is value-oriented; it is necessary to indicate the values on which to apply the list strings. By clicking on ... you can access the mask for associating values and text list elements
Bitmap	Indicates the choice of image to apply to the button: No image, a single image or list of images
Image	Active only if Caption set on Image; with this you can indicate which of the images in the project to apply to the button
ImageHAlign	Allows you to specify the horizontal centering of the image within the button. The value can be assigned to a whole variable or it can be managed with thresholds
ImageVAlign	Allows you to specify the vertical centering of the image within the button. The value can be assigned to a whole variable or it can be managed with thresholds
ImageAutoSize	Indicates whether the image should automatically sized to fit the dimensions of the Field
ImageKeepAspectRatio	Indicates whether the image should maintain the proportions of the source image

Tabella 35: Properties of the Touch button

Properties	Description
ImageTransparent	Indicates whether a transparency filter should be applied to the image
ImageTransColor	Indicates the color, selectable using the RGB code or the color palette, for which the transparency filter should be applied
ImageTagId	Active only if the Bitmap is an Image List. Determines which text list variable to choose (see chap. 5, “Image list” page 114)
ImageListId	Active only if the Bitmap is an Image List. Determines the reference text list (see chap. 5, “Image list” page 114)
ImageListType	Indicates which type of check to perform on the variable: value-oriented, single-bit or bit-group-oriented
ImageListFirstBit	Active if the ImageListType is single-bit orientated or bit-group-oriented. It indicates the bit reference to apply the control to (or the group initial reference if the control relates to a group)
ImageListLastBit	Active if ImageListType is bit-group-oriented. Indicates the last bit of the group to which the check is applied
ImageListValue	Active if ImageListType is value-oriented. Indicate the values corresponding to the Strings in the list. By clicking on ... you access the mask for associating values and text list elements
AreaVisibility	Determines whether the Button has a background area (True) or should be transparent (False); a Boolean variable can be assigned to this value or it can be managed with thresholds
AreaColorPressed	Determines the color of the Area of the button (when depressed), which can be selected using the RGB code or color palette. The value can be assigned to a whole variable or it can be managed with thresholds

Tabella 35: Properties of the Touch button

Properties	Description
AreaColorReleased	Determines the color of the Area of the button (when released), which can be selected using the RGB code or color palette. The value can be assigned to a whole variable or it can be managed with thresholds
BorderVisibility	Determines whether there will be a Border to the button or not; a Boolean variable can be assigned to this value
BorderSize	Determines the size of the Border which must be a number to which a whole variable can be assigned, if wished, or it can be managed with thresholds
BorderColorPressed	Determines the color of the Border (when the button is depressed) using the RGB code or color palette. The value can be assigned to a whole variable or it can be managed with thresholds
BorderColorReleased	Determines the color of the Border (when the button is released) using the RGB code or color palette. The value can be assigned to a whole variable or it can be managed with thresholds
BorderStyle	Determines the style of the Border, which can be Solid or Broken. The value can be assigned to a whole variable
BorderBlink	Determines the flashing of the Border, which can be No blinking, Slow blinking or Rapid blinking. The value can be assigned to a whole variable or it can be managed with thresholds
Button3D	Determines whether the button is in 3D. The value can be associated with Tag or it can be managed with thresholds
PasswordLevel	Determines the authorization level required to access the button functions (see chap. 5, "Password configuration" page 81)

Tabella 35: Properties of the Touch button

Properties	Description
Disable	Indicates whether the field should be disabled. The value can be associated with Tag or it can be managed with thresholds
Hide	Determines whether the object is initially visible. You can also assign a Boolean variable (for Runtime changes) or it can be managed with thresholds
TabIndex	Makes it possible to control the focus movement when using cursor keys within a page. It also controls the order in which data is introduced in several fields when the automatic setting of the next field of the page is enabled (see chap. 5, "General" page 90)

Touch button events

Tabella 36: Touch button events

Event	Description
OnPressed	Activated whenever the button is pressed
OnReleased	Activated whenever the button is released after being pressed

Touch Area

A touch-sensitive area can be introduced into a page by clicking on the icon or using the Main Menu (Fields->Create->Simple controls->Touch area). After clicking on the icon, use the mouse to indicate the area in the page where POLYMATH should draw the item.
The Touch Area is useful when you want to assign an entire screen area to a given function or user script (for example a part of an image to create a 'map'). The area in question can contain other graphic objects or elements.

Note: *If a Touch area is superimposed on other objects, only the function relating to the Touch area is performed in Runtime. In general, the operation relating to the object positioned on the surface is performed while those relating to the objects underneath are ignored.*

For further details regarding the script and predefined functions, the reader is advised to consult the relevant section of this manual (see chap.“Appendix B - Predefined functions” page 563 and see chap. 9, “Scripts” page 379).
The reader is advised to consult the following subsections to learn about the details of the properties and events that can be assigned to a Touch area.

Properties of the Touch Area

Tabella 37: Properties of the Touch Area

Properties	Description
Name	Identifying name of the Touch area. Must be unique among the graphic elements
Comment	Identifying comment within POLYMATH
Top	Vertical position coordinate
Left	Horizontal position coordinate
Width	Width dimension
Height	Height dimension

Tabella 37: Properties of the Touch Area

Properties	Description
PasswordLevel	Determines the authorization level required to access the Area functions (see chap. 5, “Password configuration” page 81)
Disable	Indicates whether the field should be disabled. The value can be associated with Tag or it can be managed with thresholds
TabIndex	Makes it possible to control the focus movement when using movement keys of the cursor within a page. It also controls the order in which data is introduced in several fields when the automatic setting of the next field of the page is enabled (see chap. 5, “General” page 90)

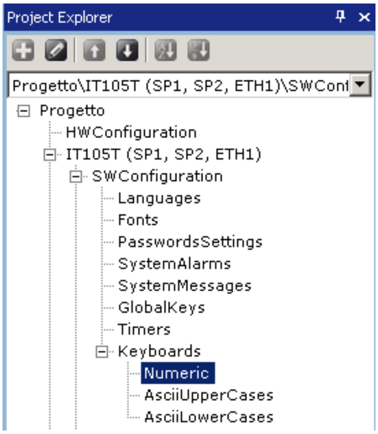
Touch Area events

Tabella 38: Touch Area events

Event	Description
OnPressed	Activated whenever the button is pressed
OnReleased	Activated whenever the button is released after being pressed

Touch Keyboard Button

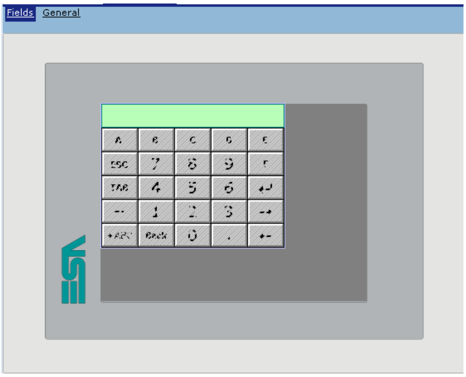
A "Touch Keyboard Button" can be inserted inside the keyboard.
Double-click one of the keyboard types on Polymath default in the "Keyboards" sub-menu of the "SW Configuration" menu of "Explore Project".



Progetto\IT105T (SP1, SP2, ETH1)\SWConf

```
Progetto
    HWConfiguration
    IT105T (SP1, SP2, ETH1)
        SWConfiguration
            Languages
            Fonts
            PasswordsSettings
            SystemAlarms
            SystemMessages
            GlobalKeys
            Timers
            Keyboards
                Numeric
                AsciiUpperCases
                AsciiLowerCases
```

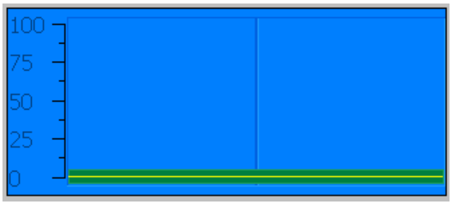
The following image will appear :



Click on the icon or the "Main Menu" (Fields->Create->Simple controls->Touch Keyboard Button.
The "Touch Keyboard Button" allows to insert a touch key for the creation and configuration of a new keyboard.
After having clicked the icon, indicate the area in which POLYMATH must designate the button using the mouse inside of a key.
The main property of the "Touch Keyboard Button" consists in the possibility of associating the ASCII code of the symbol to which the button is placed during the creation of the keyboard.

Slide potentiometer

A Slide Potentiometer can be introduced into a page by clicking on the icon or using the Main Menu (Fields->Create->Simple controls->SlidePotentiometer). After clicking on the icon, use the mouse to indicate the area in the page where POLYMATH should draw the Potentiometer.
A Slide Potentiometer is useful for introducing a direct check on a variable. There is a continuous representation of the value of the reference variable and the operator can attribute any value by just clicking on the indicator (slide control).
The reader is advised to consult the following subsections to learn about the details of the properties and events that can be assigned to a Potentiometer.



Properties of the SlidePotentiometer

Tabella 39: Properties of the Slide Potentiometer

Properties	Description
Name	Identifying name of the Potentiometer. Must be unique among the graphic elements
Comment	Identifying comment within POLYMATH
Top	Vertical position coordinate
Left	Horizontal position coordinate
Width	Width dimension
Height	Height dimension
TagId	Reference variable whose value is checked. Using the appropriate keys you can create a new variable or edit an existing one

Tabella 39: Properties of the Slide Potentiometer

Properties	Description
IndicatorColor	Indicates the color of the precision indicator of the Potentiometer; this is selected using the RGB code or the color palette
CursorColor	Indicates the color of the whole cursor of the Potentiometer: this is selected using the RGB code or the color palette
AreaVisibility	Determines whether the Potentiometer has a background area (True) or should be transparent (False); a Boolean variable can be assigned to this value or it can be managed with thresholds
AreaColor	Determines the color of the Potentiometer, selectable using the RGB code or color palette. The value can be assigned to a whole variable or it can be managed with thresholds
ScalePosition	Indicates where the scale of values should be positioned in relation to the Potentiometer. If the Potentiometer is vertical, the scale can be positioned to the left or the right; if it is horizontal, the scale can be above or below
ScaleValues	Indicates the number of divisions on the scale of values. The number relating to the value above the division is also displayed (calculated according to the number of divisions)
ScaleNotches	Indicates the number of subdivision marks appearing between two numerical divisions. These are shorter division lines than the numerical ones, giving greater precision to the representation

Tabella 39: Properties of the Slide Potentiometer

Properties	Description
ScaleColorRanges	Indicates the color ranges to be assigned to given value intervals within the scale. By clicking on .. you enter an editing window in which the value intervals and their respective colors can be defined; the window also allows you only to specify the limits in relation to the scale to be displayed
ScaleValueColor	Color of the values (figures) related to the numerical subdivisions of the scale. This can be selected using the RGB code or the color palette
FontSize	Indicates the size of the character of the values written above the numerical division lines
Direction	Indicates the direction of the scale: whether vertical or horizontal
BorderVisibility	Determines whether there will be a Border to the Potentiometer or not; a Boolean variable can be assigned to this value
BorderSize	Determines the size of the Border which must be a number to which a whole variable can be assigned, if wished, or it can be managed with thresholds
BorderColor	Determines the color of the Border using the RGB code or color palette. The value can be assigned to a whole variable or it can be managed with thresholds
BorderStyle	Determines the style of the Border, which can be Solid or Broken. The value can be assigned to a whole variable
BorderBlink	Determines the flashing of the Border, which can be No Blinking, Slow blinking or Rapid blinking. The value can be assigned to a whole variable or it can be managed with thresholds

Tabella 39: Properties of the Slide Potentiometer

Properties	Description
Border3D	Determines a 3D effect for the Border, which can be Flat, Relief, Recessed, Bump or Etched. The value can be assigned to a whole variable or it can be managed with thresholds
PasswordLevel	Determines the authorization level required to be able to edit the potentiometer value (see chap. 5, "Password configuration" page 81). This property is ignored if the field is Read Only
ReadOnly	Indicates whether the field should be editable in Runtime
Disable	Indicates whether the field should be disabled. The value can be associated with Tag or it can be managed with thresholds
Hide	Determines whether the object is initially visible. You can also assign a Boolean variable (for Runtime changes) or it can be managed with thresholds
TabIndex	Makes it possible to control the focus movement when using cursor keys within a page. It also controls the order in which data is introduced in various fields when the automatic setting of the next field of the page is enabled (see chap. 5, "General" page 90)

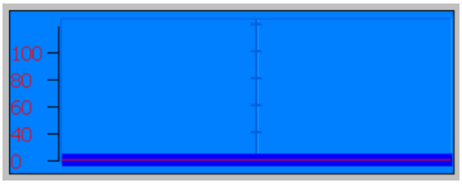
Slide Potentiometer events

Tabella 40: Slide Potentiometer events

Event	Description
OnValueChange	Activated when the value of the Potentiometer is changed using the touch screen

Slide Selector

A Slide Selector can be introduced into a page by clicking on the icon or using the Main Menu (Fields->Create->Simple controls->SlideSelector. After clicking on the icon, use the mouse to indicate the area in the page where POLYMATH should draw the Selector.
Slide Selectors are useful is useful for introducing a direct check on a variable. There is a discrete representation of the value of the reference variable and the operator can attribute one of the available values by just clicking on the indicator (slide control).
The reader is advised to consult the following subsections to learn about the details of the properties and events that can be assigned to a Slide Selector.



Note: *It is advisable to use the SlideSelector (rather than a Potentiometer) if the number of choices that can be executed by the operator is restricted, giving a limited range of options.*

Properties of the SlideSelector

Tabella 41: Properties of the SlideSelector

Properties	Description
Name	Identifying name of the Selector. Must be unique among the graphic elements
Comment	Identifying comment within POLYMATH
Top	Vertical position coordinate
Left	Horizontal position coordinate
Width	Width dimension
Height	Height dimension

Tabella 41: Properties of the SlideSelector

Properties	Description
TagId	Reference variable whose value is checked. Using the appropriate keys you can create a new variable or edit an existing one
IndicatorColor	Indicates the color of the precision Indicator of the Potentiometer. This can be selected using the RGB code or the color palette
CursorColor	Indicates the color of the entire cursor of the Potentiometer. This can be selected using the RGB code or the color palette
AreaVisibility	Determines whether the Potentiometer has a background area (True) or should be transparent (False); a Boolean variable can be assigned to this value or it can be managed with thresholds
AreaColor	Determines the color of the Potentiometer, which can be selected using the RGB code or color palette. The value can be assigned to a whole variable or it can be managed with thresholds
ScalePosition	Indicates where the scale of values should be positioned in relation to the Bar. If the Bar is vertical, the scale can be positioned to the left or the right; if it is horizontal, the scale can be above or below
ScaleValueColor	Determines the color Color relating to the scale of values. This can be selected using the RGB code or the color palette
NumValues	Defines the values to be inserted into the scale. By clicking on ... you enter an editing window in which the value intervals and their respective colors can be defined; the window also allows you only to specify the limits in relation to the scale to be displayed

Tabella 41: Properties of the SlideSelector

Properties	Description
ValueType	Defines the type of display setting for the scale: the display of values can be maintained or a text list can be used corresponding to the values
TextListId	Active if the value type is Text List. Allows you to choose the text list associated with the values in question (see chap. 5“Text list” page 113)
FontField	Active if the value type is Text List. Font related to the text shown in the field; by clicking on .. you can edit multilanguage Fonts (see chap. 5, “Languages” page 78)
FontSize	Establishes the Font size
Direction	Indicates the direction of the scale: whether vertical or horizontal
BorderVisibility	Determines whether there will be a Border to the Selector or not; a Boolean variable can be assigned to this value
BorderSize	Determines the size of the Border which must be a number to which a whole variable can be assigned, if wished, or it can be managed with thresholds
BorderColor	Determines the color of the Border using the RGB code or color palette. The value can be assigned to a whole variable or it can be managed with thresholds
BorderStyle	Determines the style of the Border, which can be Solid or Broken. The value can be assigned to a whole variable
BorderBlink	Determines the flashing of the Border, which can be No blinking, Slow blinking or Rapid blinking. The value can be assigned to a whole variable or it can be managed with thresholds

Tabella 41: Properties of the SlideSelector

Properties	Description
Border3D	Determines a 3D effect for the Border, which can be Flat, Relief, Recessed, Bump or Etched. The value can be assigned to a whole variable or it can be managed with thresholds
PasswordLevel	Determines the authorization level required to edit the potentiometer value (see chap. 5, "Password configuration" page 81). This property is ignored if the field is Read Only
ReadOnly	Indicates whether the field should be editable in Runtime
Disable	Indicates whether the field should be disabled. The value can be associated with Tag or it can be managed with thresholds
Hide	Determines whether the object is initially visible. You can also assign a Boolean variable (for Runtime changes)
TabIndex	Makes it possible to control the focus movement when using cursor keys within a page. It also controls the order in which data is introduced in several fields when the automatic setting of the next field of the page is enabled (see chap. 5, "General" page 90)

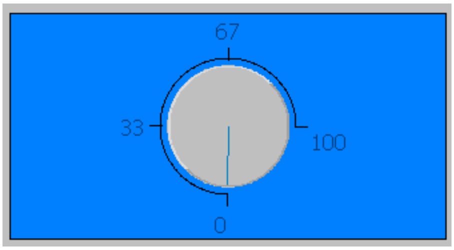
Slide Selector events

Tabella 42: Slide Selector events

Event	Description
OnValueChange	Activated when the value of the Selector is changed using the touch screen

Knob Potentiometer

A Knob Potentiometer can be introduced into a page by clicking on the icon ◉ or using the Main Menu (Fields->Create->Simple controls->KnobPotentiometer). After clicking on the icon, use the mouse to indicate the area in the page where POLYMATH should draw the potentiometer.
Knob potentiometers are useful for introducing a direct control on a variable. A continuous representation of the value of the reference variable is given and the operator can attribute any value simply by clicking on the knob indicator.
The reader is advised to consult the following subsections to learn about the details of the properties and events that can be assigned to a potentiometer.



Properties of the Knob Potentiometer

Tabella 43: Properties of the Knob Potentiometer

Properties	Description
Name	Identifying name of the Potentiometer. Must be unique among the graphic elements
Comment	Identifying comment within POLYMATH
Top	Vertical position coordinate
Left	Horizontal position coordinate
Width	Width dimension
Height	Height dimension
TagId	Reference variable whose value is checked. Using the appropriate keys you can create a new variable or editing an existing one

Tabella 43: Properties of the Knob Potentiometer

Properties	Description
StartAngle	Determines the Knob starting position (given as an angle)
SweepAngle	Determines the angle (in degrees) of the aperture of the Knob
IndicatorColor	Indicates the color of the precision indicator of the Potentiometer. This is selected using the RGB code or the color palette
KnobColor	Determines the color of the Potentiometer knob, selectable using the RGB code or color palette
AreaVisibility	Determines whether the Potentiometer has a background area (True) or should be transparent (False); a Boolean variable can be assigned to this value or it can be managed with thresholds
AreaColor	Determines the color of the Potentiometer, selectable using the RGB code or color palette. The value can be assigned to a whole variable
ScaleEnable	Determines whether the scale of values is to be present or not
ScaleValues	Indicates the number of divisions on the scale of values. The number relating to the value above the division is also displayed (calculated according to the number of divisions)
ScaleNotches	Indicates the number of subdivision marks appearing between two numerical divisions. These are shorter division lines than the numerical ones, giving greater precision to the representation

Tabella 43: Properties of the Knob Potentiometer

Properties	Description
ScaleColorRanges	Indicates the color ranges to be assigned to given value intervals within the scale. By clicking on ... you enter an editing window in which the value intervals and their respective colors can be defined; the window also allows you simply to specify the limits in relation to the scale to be displayed
ScaleValueColor	Determines the color relating to the scale of values. This can be selected using the RGB code or the color palette
FontSize	Establishes the Font size for representing the text of the scale
BorderVisibility	Determines whether there will be a Border to the Potentiometer or not; a Boolean variable can be assigned to this value
BorderSize	Determines the size of the Border which must be a number to which a whole variable can be assigned, if wished
BorderColor	Determines the color of the Border using the RGB code or color palette. The value can be assigned to a whole variable
BorderStyle	Determines the style of the Border, which can be Solid or Broken. The value can be assigned to a whole variable or it can be managed with thresholds
BorderBlink	Determines the flashing of the Border, which can be No blinking, Slow blinking or Rapid blinking. The value can be assigned to a whole variable
Border3D	Determines a 3D effect for the Border, which can be Flat, Relief, Recessed, Bump or Etched. The value can be assigned to a whole variable

Tabella 43: Properties of the Knob Potentiometer

Properties	Description
PasswordLevel	Determines the authorization level required to edit the potentiometer value (see chap. 5, “Password configuration” page 81). This property is ignored if the field is Read Only
ReadOnly	Indicates whether the field should be editable in Runtime
Disable	Indicates whether the field should be disabled. The value can be associated with Tag or it can be managed with thresholds
Hide	Determines whether the object is initially visible. You can also assign a Boolean variable (for Runtime changes) or it can be managed with thresholds
TabIndex	Makes it possible to control the focus movement when using the cursor keys within a page. It also controls the order in which data is introduced in several fields when the automatic setting of the next field of the page is enabled (see chap. 5, “General” page 90)

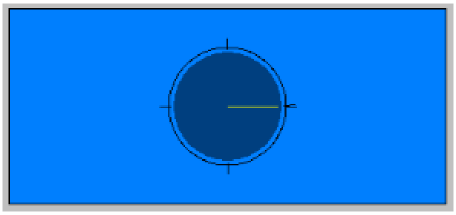
Knob Potentiometer events

Tabella 44: Knob Potentiometer events

Event	Description
OnValueChange	Activated when the Potentiometer value is changed using the touch screen

Knob Selector

A Knob Selector can be introduced into a page by clicking on the icon ◎ or using the Main Menu (Fields->Create->Simple controls->KnobSelector). After clicking on the icon, use the mouse to indicate the area in the page where POLYMATH should draw the selector.
Knob selectors are useful for introducing a direct control on a given variable. A discrete representation of the value of the reference variable is given and the operator can attribute one of the values present simply by clicking on the knob.
The reader is advised to consult the following subsections to learn about the details of the properties and events that can be assigned to a Slide selector.



__Note:__ It is advisable to use the Knob selector (rather than a potentiometer) if the number of choices the operator can make is to be restricted, giving a limited range of options.

Properties of the Knob Selector

Tabella 45: Properties of the Knob Selector

Properties	Description
Name	Identifying name of the Selector. Must be unique among the graphic elements
Comment	Identifying comment within POLYMATH
Top	Vertical position coordinate
Left	Horizontal position coordinate
Width	Width dimension
Height	Height dimension

Tabella 45: Properties of the Knob Selector

Properties	Description
TagId	Reference variable whose value is checked. Using the appropriate keys you can create a new variable or edit an existing one
StartAngle	Determines the Knob starting position (given as an angle)
SweepAngle	Determines the angle (in degrees) of the aperture of the Knob
IndicatorColor	Defines the color of the indicator hand. This is selected using the RGB code or the color palette
KnobColor	Determines the color of the Selector knob, which can be selected using the RGB code or color palette.
ScaleValueColor	Determines the color of the scale of values. This can be selected using the RGB code or the color palette
AreaVisibility	Determines whether the Selector has a background area (True) or should be transparent (False); a Boolean variable can be assigned to this value or it can be managed with thresholds
AreaColor	Determines the color of the Selector, which can be selected using the RGB code or color palette. The value can be assigned to a whole variable
NumValues	Indicates the values to insert into the scale. By clicking on ... you enter an editing window in which the value intervals and their respective colors can be defined; the window also allows you only to specify the limits in relation to the scale to be displayed
ValueType	Indicates the display type for setting the scale: the display of values can be maintained or a text list corresponding to the values can be used

Tabella 45: Properties of the Knob Selector

Properties	Description
TextListId	Active if the value is a Text List value. Allows you to choose the text list associated with the values in question (see chap. 5“Text list” page 113)
FontField	Active if the value is a Text List value. Font for the text shown in the field; by clicking on ... you can edit multilanguage Fonts (see chap. 5, “Languages” page 78)
FontSize	Establishes the Font size
BorderVisibility	Determines whether the Border of the Selector should be present or not; a Boolean variable can be assigned to this value
BorderSize	Determines the size of the Border which must be a number to which a whole variable can be assigned, if so desired
BorderColor	Determines the color of the Border using the RGB code or color palette. The value can be assigned to a whole variable
BorderStyle	Determines the style of the Border, which can be Solid or Broken. The value can be assigned to a whole variable or it can be managed with thresholds
BorderBlink	Determines the flashing of the Border, which can be No blinking, Slow blinking or Rapid blinking. The value can be assigned to a whole variable
Border3D	Determines a 3D effect for the Border, which can be Flat, Relief, Recessed, Bump or Etched. The value can be assigned to a whole variable
PasswordLevel	Determines the authorization level required to be able to edit the selector value (see chap. 5, “Password configuration” page 81). This property is ignored if the field is Read Only

Tabella 45: Properties of the Knob Selector

Properties	Description
ReadOnly	Indicates whether the field should be editable in Runtime
Disable	Indicates whether the field should be disabled. The value can be associated with Tag or it can be managed with thresholds
Hide	Determines whether the object is initially visible. You can also assign a Boolean variable (for Runtime changes) or it can be managed with thresholds
TabIndex	Makes it possible to control the focus movement when using the cursor keys within a page. It also controls the order in which data is introduced in several fields when the automatic setting of the next field of the page is enabled (see chap. 5, "General" page 90)

Knob Selector events

Tabella 46: Knob Selector events

Event	Description
OnValueChange	Activated when the Selector value is changed using the touch screen

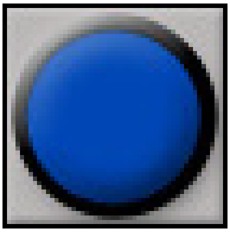
Complex Controls

Complex Controls are objects that can be inserted into a page in order to show the operator the value of one or more data items (or groups of data, like recipes, alarms, trends etc.) and, if required, edit them. In this section we will analyze each Complex Control, setting out its functional characteristics, their respective properties (to be configured by the Properties Editor) and associated events (Events Editor).

Monostable Button

A Monostable Button can be introduced into a page by clicking on the icon or using the Main Menu (Fields->Create->Complex controls->Monostable button). After clicking on the icon, use the mouse to indicate the area in the page where POLYMATH should draw the button. The Monostable Button serves basically to trigger OnPressed and OnReleased events to which the desired functions (or scripts) can be assigned. For further details regarding predefined functions and user scripts that can be assigned to the button, the reader is advised to consult the appropriate section in this manual (see chap.‚‚Appendix B - Predefined functions" page 563 and see chap. 9, "Scripts" page 379).
The reader is advised to consult the following subsections to learn about the details of the meaning of the properties that can be assigned to a Monostable Button and how to edit them.

Note: *No variable has to be assigned to a monostable button. When you want the pressing of the monostable button to have an effect on a variable, just assign the appropriate functions to the button's events.*



Properties of the Monostable button

Tabella 47: Properties of the Monostable button

Properties	Description
Name	Identifying name of the Monostable button. Must be unique among the graphic elements
Comment	Identifying comment within POLYMATH
Top	Vertical position coordinate
Left	Horizontal position coordinate
Width	Width dimension

Tabella 47: Properties of the Monostable button

Properties	Description
Height	Height dimension
PressedAreaVisibility	Determines whether the Button has a background (True) or is transparent (False) when pressed; a Boolean variable can be assigned to this value or it can be managed with thresholds
PressedAreaColor	Determines the color of the area of the button (when pressed), selectable using the RGB code or color palette. The value can be assigned to a whole variable or it can be managed with thresholds
ReleasedArea Visibility	Determines whether the button has a background (True) or is transparent (False) when released; a Boolean variable can be assigned to this value or it can be managed with thresholds
ReleasedAreaColor	Determines the color of the area of the button (when released), which can be selected using the RGB code or color palette. The value can be assigned to a whole variable or it can be managed with thresholds
PasswordLevel	Determines the authorization level required to access the button utilities (see chap. 5, “Password configuration” page 81)
Disable	Indicates whether the field should be disabled. The value can be associated with Tag or it can be managed with thresholds
Hide	Determines whether the object is initially visible. You can also assign a Boolean variable (for Runtime changes) or it can be managed with thresholds

Tabella 47: Properties of the Monostable button

Properties	Description
TabIndex	Makes it possible to control the focus movement when using the cursor keys within a page. It also controls the order in which data is introduced in several fields when the automatic setting of the next field of the page is enabled (see chap. 5, “General” page 90)

Monostable button events

Tabella 48: Monostable button events

Event	Description
OnPressed	Activated whenever the button is pressed
OnReleased	Activated whenever the button is released after being pressed

Monostable button events

Once a monostable button is added to a page its form can be edited but in OnPressed state and in OnReleased state. To edit the button just double-click on it within the page; editing the monostable button comprises three windows: Pressed, Released and General.
In the Pressed and Released masks the graphic appearance of the button in its two states can be defined. Editing these windows works like normal editing for project pages (see chap. 6, “Managing a page” page 167).

Note: *A monostable button differs from a touch button in its graphic form which is completely editable. As described in this chapter, images or geometric forms can be applied to them. The library supplied with POLYMATH contains a set of buttons ready for use in a project (see chap. 7, “Standard library in POLYMATH” page 346).*

The General window can be used to set identifying properties related to the button; the Name is a unique string within a set of graphic objects, while the comment is a recognition text to be used only within POLYMATH. You can also choose to overwrite the global grid dimensions to make positioning on the surface of the button more (or less) precise.

Bistable button

A Bistable button can be introduced into a page by clicking on the icon or using the Main Menu (Fields->Create->Complex controls->Bistable button). After clicking on the icon, use the mouse to indicate the area in the page where POLYMATH should draw the button.
A bistable button is useful when you need to change and memorize the value of a variable by pressing it. Unlike a monostable button, the bistable button must have two values of a variable assigned to it (one for the ON-state and one for the OFF-state) and pressing it changes the value variable.
The library supplied with POLYMATH contains a set of buttons ready for use within the project (see chap. 7, “Standard library in POLYMATH” page 346).
The reader is advised to consult the following subsections to learn about the details of the properties and events that can be assigned to a bistable button and how to edit it.

Properties of a Bistable button

Tabella 49: Properties of a Bistable button

Properties	Description
Name	Identifying name of the Bistable button. Must be unique among the graphic elements
Comment	Identifying comment within POLYMATH
Top	Vertical position coordinate
Left	Horizontal position coordinate
Width	Width dimension
Height	Height dimension
TagId	Reference variable whose value is checked. Using the appropriate keys you can create a new variable or edit an existing one

Tabella 49: Properties of a Bistable button

Properties	Description
OffStateAreaFlag	Determines whether the Button has a background (True) or should transparent (False) when it is OFF; a Boolean variable can be assigned to this value or it can be managed with thresholds
OffStateAreaColor	Determines the color of the area of the Button (in OFF state), which can be selected using the RGB code or color palette. The value can be assigned to a whole variable or it can be managed with thresholds
OnStateAreaFlag	Determines whether the Button has a background (True) or should transparent (False) when it is ON; a Boolean variable can be assigned to this value or it can be managed with thresholds
OnStateAreaColor	Determines the color of the area of the button (in ON state), which can be selected using the RGB code or color palette. The value can be assigned to a whole variable or it can be managed with thresholds
ValueStateOff	Indicates the value that the reference variable must assume for the button to be OFF; if the button goes to OFF by being pressed, the value of the variable is updated to that value
ValueStateOn	Indicates the value that the reference variable must assume for the button to be ON; if the button goes to ON by being pressed, the value of the variable is updated to that value
PasswordLevel	Determines the authorization level required to access the button utilities (see chap. 5, “Password configuration” page 81)
Disable	Indicates whether the button should be disabled. The value can be associated with Tag or it can be managed with thresholds

Tabella 49: Properties of a Bistable button

Properties	Description
Hide	Determines whether the object is initially visible. You can also assign a Boolean variable (for Runtime changes) or it can be managed with thresholds
TabIndex	Makes it possible to control the focus movement when using the cursor keys within a page. It also controls the order in which data is introduced in several fields when the automatic setting of the next field of the page is enabled (see chap. 5, “General” page 90)

Bistable button events

Tabella 50: Bistable button events

Event	Properties
OnSwitchButtonOn	Activated when the Button is pressed in position ON
OnSwitchButtonOff	Activated when the Button is pressed in position OFF

Editing the Bistable button

Once a Bistable button has been added to a page, its form can be edited both for the ON state and the OFF state. To edit the button just double-click on it within the page; editing a Bistable button comprises three windows: OFF, ON and General.
The OFF and ON masks can be used to define the graphic appearance of the button in its two states. Editing these windows works like normal editing for project pages (see chap. 6, “Managing a page” page 167).
The General window is used to set identifying properties relating to the Button: the Name is a unique string within the set of graphic objects while the comment is a recognition text to be used only within POLYMATH. You can also choose to overwrite the global grid dimensions to make positioning within the surface of the button more (or less) precise.

Frame Field

A Frame field can be introduced into a page by clicking on the icon 🟩 or using the Main Menu (Fields->Create->Complex controls->Frame). After clicking on the icon, use the mouse to indicate the area in the page where POLYMATH should add the Frame.
A Frame Field is simply an area for containing an actual Frame. We have already described in the preceding chapter how to create a Frame (see chap. 5, “Frames” page 128). The reader is advised to consult the following subsections to learn about the details of the properties and events that can be assigned to a Frame Field.

__Note:__ A Frame can also be added to a page simply by dragging the frame in question from Project Explorer to the desired page position in the Work area. With this procedure POLYMATH automatically creates a field to contain the dragged frame.

Properties of a Frame Field

Tabella 51: Properties of a Frame Field

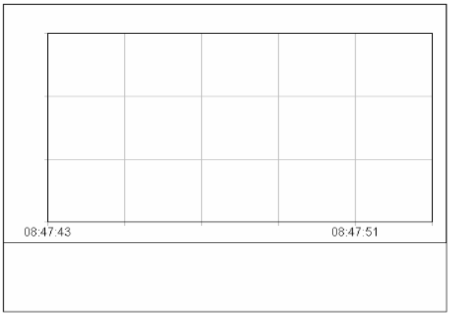
Properties	Description
Name	Identifying name of the Frame Field. Must be unique among the graphic elements
Comment	Identifying comment within POLYMATH
FrameId	Indicates the Frame to be contained by the Frame Field being edited
Top	Vertical position coordinate
Left	Horizontal position coordinate
Width	Dimension of the width; not editable but determined by the width of the Frame contained
Height	Dimension of the height; not editable but determined by the height of the Frame contained

Tabella 51: Properties of a Frame Field

Properties	Description
TabIndex	Makes it possible to control the focus movement using cursor keys within the page. It also controls the order in which data is introduced in several fields when the automatic setting of the next field of the page is enabled (see chap. 5, “General” page 90)

Trend View

A Trend View can be introduced into a page by clicking on the icon 📈 or using the Main Menu (Fields->Create->Complex controls->TrendView). After clicking on the icon, use the mouse to indicate the area in the page where POLYMATH should draw the TrendView.
A Trend View is the field inside which you can see the contents of the Trend Buffer, whose working was described in the preceding section (see chap. 5, “Trend Buffers” page 142).
The reader is advised to consult the following subsections to learn about the details of the properties and events that can be assigned to a TrendView and how it is edit.



Properties of a TrendView

Tabella 52: Properties of a TrendView

Properties	Description
Name	Identifying name of the TrendView. Must be unique among the graphic elements

Tabella 52: Properties of a TrendView

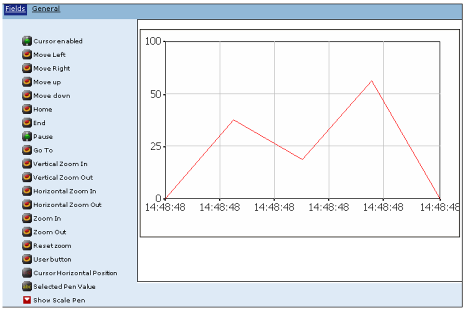
Properties	Description
Comment	Identifying comment within POLYMATH
Top	Vertical position coordinate
Left	Horizontal position coordinate
Width	Width dimension
Height	Height dimension
AreaVisibility	Determines whether the sector has a background area (True) or should be transparent (False); a Boolean variable can be assigned to this value or it can be managed with thresholds
AreaColor	Determines the color of the display, which can be selected using the RGB code or color palette. The value can be assigned to a whole variable or it can be managed with thresholds
BorderVisibility	Determines whether there will be a Border to the display or not; a Boolean variable can be assigned to this value
BorderSize	Determines the size of the Border which must be a number to which a whole variable can be assigned, if wished, or it can be managed with thresholds
BorderColor	Determines the color of the Border using the RGB code or color palette. The value can be assigned to a whole variable or it can be managed with thresholds
BorderStyle	Determines the style of the Border, which can be Solid or Broken. The value can be assigned to a whole variable or it can be managed with thresholds
BorderBlink	Determines the flashing of the Border, which can be No blinking, Slow blinking or Rapid blinking. The value can be assigned to a whole variable or it can be managed with thresholds

Tabella 52: Properties of a TrendView

Properties	Description
Border3D	Determines a 3D effect for the Border, which can be Flat, Relief, Recessed, Bump or Etched. The value can be assigned to a whole variable or it can be managed with thresholds
Hide	Determines whether the object is initially visible. You can also assign a Boolean variable (for Runtime changes) or it can be managed with thresholds
Lock	Determines if the object can move or not

Editing a TrendView

After inserting a TrendView into a page, just double-click on it to edit. The default object is a Trend Graph (see chap. 6, "Properties of a Trend Graph" page 281) that can be accompanied, if the programmer wishes, by a series of control buttons for displaying the Trend. Editing is organized through two masks, Fields and General.



The Fields mask allows you to indicate which buttons have to be present together with the table and position them in the area. Each button has properties which can be edited in the Properties Editor as happens with normal touch buttons (see chap. 6, "Touch Button" page 243). To add or eliminate a button just click on the list of buttons present to the left of the table. If an object is already present in the page it will appear highlighted within the list (it will be visible in the Table Edit Area). To move an element (button or table) just drag it to the desired position. Insertable buttons are different and a non-editable predefined function can be assigned to each of these:

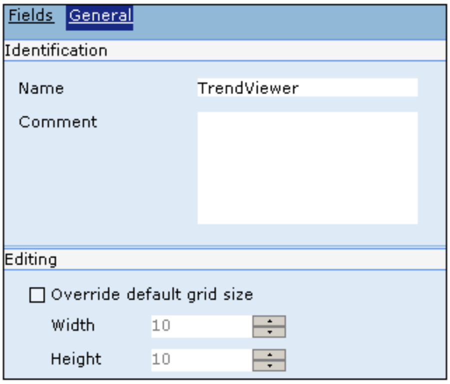
- Move Left: The button has two functions depending on whether the cursor is displayed or not: if the cursor is invisible, pressing the key makes the graph move from right to left. If, however, the cursor is visible, the button moves it to the left and when it reaches the furthest point, the graph moves from right to left by a unit defined by the principal horizontal division of the grid.
- Move Right: The button has two functions depending on whether the cursor is displayed or not: if the cursor is invisible, pressing the key makes the graph move from left to right. If, however, the cursor is visible the button moves it to the right and when it reaches the furthest point, the graph moves from left to right by a unit defined by the principle horizontal division of the grid.
- Move Up: the button makes the graph move upwards by a unit defined by the major vertical division of the grid.
- Move Down: the button makes the graph move downwards by a unit defined by the major vertical division of the grid.
- Principal: the button makes the graph move from right to left until the oldest sample readings are positioned on the left side of the graph.
- End: the button makes the graph move from right to left until the most recent sample readings are positioned on the right side of the graph.
- GoTo: the button makes a dialog window appear to ask the user at what date and time the right side of the graph should be put.
- IncreaseVerticalEnlargement: increases the vertical scale factor
- ReduceVerticalEnlargement: decreases the vertical scale factor
- IncreaseHorizontalEnlargement: increases the horizontal scale factor
- ReduceHorizontalEnlargement: decreases the horizontal scale factor
- "Zoom": Increases the total graphic display
- "Reduction": Decreases the total graphic display
- Reset Enlargement: restores the original scale factors (no zoom)
- "User Button": Button to which the user can assign a function/script.
- "HorizontalCursorPosition": it represents the sample acquisition time (when it identifies at least one sample on the graphics)

- "Selected Pendrive Value": field that indicates the pendrive currently selected
- ShowScalePen: Determines the scale pen to be shown via a pull-down menu

There are also two bistable buttons, a Date-Time Field and a Numerical Field that can be edited as already described in this chapter (see chap. 6, “Bistable button” page 272, see chap. 6, “DateTime field” page 232 and see chap. 6, “Numerical Field” page 215) each having its own function:

- CursorEnabled: allows the graphic cursor to be displayed or not
- Pause: shows whether or not the update of the graph is enabled (does not disable the acquisition of samples).
- HorizontalCursorPosition: represents the time of the acquisition of the sample (when it identifies at least one sample on the graph)
- Pen Selected: field indicating that the Pen is currently selected.

As already described in this chapter, another customized label can be added to the complex field (see chap. 6, “Complex label” page 190) or a dynamic field showing the value of the Trend on the pen (see chap. 6, “Trend Pen” page 191).



Identification

Name TrendViewer

Comment

Editing

□ Override default grid size

Width 10

Height 10

The General mask can be used to insert a name and an identifying comment for the TrendView being edited. In addition, you can overwrite the default dimensions of the editing grid of the page (see chap. 5, “Main window” page 75) introducing new measures in pixels valid only for editing the current field.

Properties of a Trend Graph

Tabella 53: Properties of a Trend Graph

Properties	Description
Top	Vertical position coordinate
Left	Horizontal position coordinate
Width	Dimension of the width
Height	Dimension of the height
AreaColor	Determines the color of the Area, which can be selected using the RGB code or color palette. The value can be assigned to a whole variable or it can be managed with thresholds
AreaVisibility	Determines whether the Trend Graph has a background area (True) or should be transparent (False); a Boolean variable can be assigned to this value or it can be managed with thresholds
BorderVisibility	Determines whether there will be a Border to the Table or not; a Boolean variable can be assigned to this value
BorderSize	Determines the size of the Border which must be a number to which a whole variable can be assigned or it can be managed with thresholds
BorderColor	Determines the color of the Border using the RGB code or color palette. The value can be assigned to a whole variable or it can be managed with thresholds
BorderStyle	Determines the style of the Border, which can be Solid or Broken. The value can be assigned to a whole variable or it can be managed with thresholds
Border3D	Determines a 3D effect for the Border, which can be Flat, Relief, Recessed, Bump or Etched. The value can be assigned to a whole variable or it can be managed with thresholds

Tabella 53: Properties of a Trend Graph

Properties	Description
BorderBlink	Determines the flashing of the Border, which can be No blinking, Slow blinking or Rapid blinking. The value can be assigned to a whole variable or it can be managed with thresholds
UpdateMode	Indicates the way the Trend display is updated: automatically, with a change of value or on command
RefreshTime	Indicates the refresh period of the trend expressed in milliseconds
HorScaleVisible	Indicates whether there needs to be a horizontal scale
HorScaleMode	Indicates the way the scale should be displayed. The Date alone, the Time alone, both or tenths of seconds can be represented
HorScaleDateFormat	Active if the type of scale envisages the Date and permits its format to be specified
HorScaleTimeFormat	Active if the type of scale envisages the Time and permits its format to be specified
HorScaleLabelFont	Indicates the Font for the label texts of the horizontal scale
HorScaleLabelColor	Indicates the color for the label texts of the horizontal scale; these can be selected using the RGB code or color palette
HorScaleLabelSkip	Indicates the frequency with which the horizontal scale labels should be inserted
VerScaleVisible	Indicates whether there should be a vertical scale
VerScaleVisible Number	Number of digits to show on the vertical scale
VerScaleLabel Decimal	Number of decimal digits to show on the vertical scale

Tabella 53: Properties of a Trend Graph

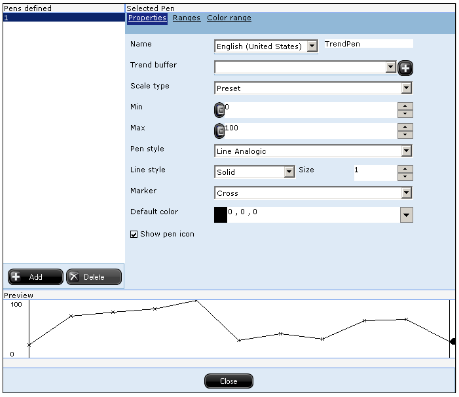
Properties	Description
ScrollType	Indicates how the scroll movement of the table should operate: may be continuous, half screen or full screen
TimeSpan	Length of time periods expressed in thousandths of a second. If the value 10000 is entered, for example, at any point the trend table will display the values gathered in 10 seconds
GridHorVisible	Indicates whether there should be a horizontal grid
GridHorDivision Number	Indicates the number of horizontal divisions in the grid
GridHorMinDivision Number	Indicates the number of horizontal subdivisions in the grid, that is, the number of horizontal lines between any two divisions
GridHorDivisionStyle	Indicates the style of the divisions of the horizontal grid: may be Solid or Broken line
GridHorMinDivision Style	Indicates the style of the subdivisions of the horizontal grid: may be Solid or Broken line
GridHorDivisionColor	Indicates the color of the divisions of the horizontal grid, can be done using the RGB code or the color palette
GridHorMinDivision Color	Determines the color of the horizontal grid subdivisions, which can be selected using the RGB code or color palette.
GridVerVisible	Indicates whether there needs to be a vertical grid
GridVerDivision Number	Indicates the number of vertical divisions in the grid
GridVerMinDivision Number	Indicates the number of vertical subdivisions in the grid, that is, the number of horizontal lines between any two divisions

Tabella 53: Properties of a Trend Graph

Properties	Description
GridVerDivisionStyle	Indicates the style of the divisions in the vertical grid: may be Solid or Broken line
GridVerMinDivision Style	Indicates the style of the subdivisions in the vertical grid: may be Solid or Broken line
GridVerDivisionColor	Determines the color of the vertical grid divisions, which can be selected using the RGB code or color palette.
GridVerMinDivision Color	Determines the color of the vertical subdivisions of the grid, which can be selected using the RGB code or color palette.
Pens	Indicates the pens to use in representing the trend. By clicking on the ... key you can edit the types of pen (as shown in the next subsection)

Editing Trend Pens

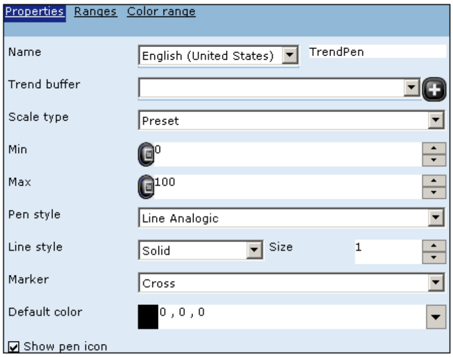
To be able to edit the Pens for writing Trends, you have to enter the complex field edit function, TrendView, (double-click on it in the page). After selecting the Trend Graph, use the appropriate Properties Editor to click on the icon ... in the Pens option.



Name: English (United States) TrendPen
Trend buffer:
Scale type: Preset
Min: 0
Max: 100
Pen style: Line Analogic
Line style: Solid Size: 1
Marker: Cross
Default color: 0 , 0 , 0
☑ Show pen icon

The Pen edit window that is displayed is composed of three sections. The left part has a list of pens created by the user from which it is possible to create and eliminate elements. The bottom part contains a preview of the pen currently being edited, while the middle part of the window contains the real editing area for the pen selected. This window is organized into property masks, Intervals and Interval Colors that are dealt with in the next subsections.

Properties



Name: English (United States) TrendPen
Trend buffer:
Scale type: Preset
Min: 0
Max: 100
Pen style: Line Analogic
Line style: Solid Size 1
Marker: Cross
Default color: 0 , 0 , 0
☑ Show pen icon

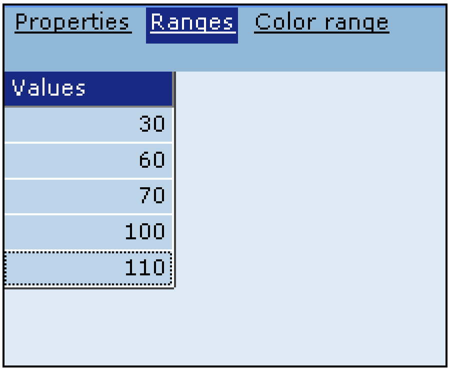
First of all, it is possible to assign a Name to the Trend Pen and assign a Trend Buffer, for which the Pen must be used.
A type must be indicated for the scale of values: this may be:

- Programmed - it is necessary to indicate the maximum and minimum values which can also be assigned to variables
- Automatic - calculated in Runtime on the basis of the values contained in the Buffer (but limits can also be inserted)
- Tag Limit related - the Buffer has to refer to a limited variable (see chap. 5, “Limits” page 94)
- Client - maximum and minimum values must be defined

You can also choose the appearance of the penline, which can be of the following: samples only, analog (continuous, with oblique connections between the values) or digital (scaled, with digital steps). Also the dimensions of the line, its color

and style (solid, broken or dotted) can be edited to suit the user’s taste.
The pen marker can assume various different geometric forms (pixel, circle, cross etc.) and you can choose not to show the icon relating to the pen.
Each variation updates the preview at the bottom of the mask.

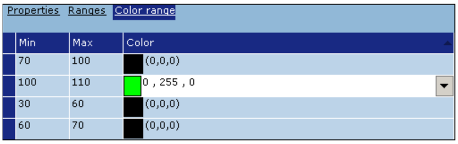
Intervals



Values
30
60
70
100
110

This mask is used to insert the values relating to the intervals to which different representation colors can be attributed. The scale of intervals must present values in increasing order.

Interval colors

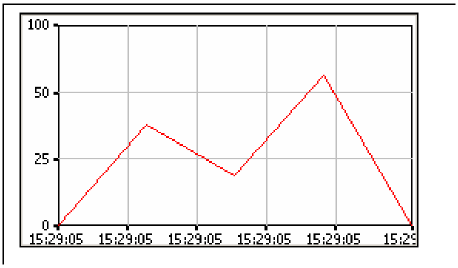


Min	Max	Color
70	100	■ (0,0,0)
100	110	🟩 0 , 255 , 0
30	60	■ (0,0,0)
60	70	■ (0,0,0)

This mask lets you assign a color to each of the value intervals set out in the Interval mask. A color is applied when its value read by the Buffer memory is within the corresponding interval.

TrendXY

A "TrendXY" can be inserted inside of the page clicking on the icon or from the "Main Menu" (Fields->Create->Controls Complexes->TrendXY). After having clicked the icon, indicate the area in which POLYMATH must designate the "TrendXY" using the mouse inside of the page.
The "TrendXY" is the field inside of which the content of the TrendBufferXY is displayed, the functioning of which was described in the previous paragraph.
Consult the next sub-paragraphs to know the details of the properties that can be associated to a VistaTrendXY and its editing modes.



Properties of the TrendXY

Tabella 54: Properties of a TrendViewXY

Properties	Description
Comment	Identifying comment within POLYMATH
Name	Identifying name of the TrendView. Must be unique among the graphic elements
Height	Height dimension
Left	Horizontal position coordinate
Top	Vertical position coordinate
Width	Width dimension

Tabella 54: Properties of a TrendViewXY

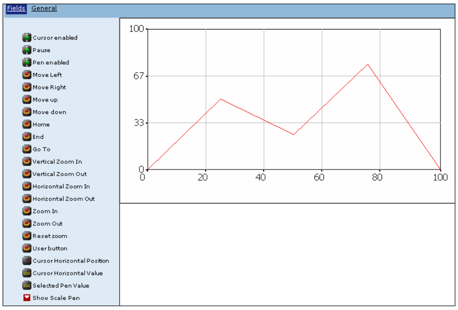
Properties	Description
AreaColor	Determines the color of the display, which can be selected using the RGB code or color palette. The value can be assigned to a whole variable or it can be managed with thresholds
AreaVisibility	Determines whether the Trend View has a background area (True) or should be transparent (False); a Boolean variable can be assigned to this value or it can be managed with thresholds
Border3D	Determines a 3D effect for the Border, which can be Flat, Relief, Recessed, Bump or Etched. The value can be assigned to a whole variable or it can be managed with thresholds
BorderBlink	Determines the flashing of the Border, which can be No blinking, Slow blinking or Rapid blinking. The value can be assigned to a whole variable or it can be managed with thresholds
BorderColor	Determines the color of the Border using the RGB code or color palette. The value can be assigned to a whole variable or it can be managed with thresholds
BorderVisibility	Determines whether there will be a Border to the display or not; a Boolean variable can be assigned to this value
BorderSize	Determines the size of the Border which must be a number to which a whole variable can be assigned, if wished, or it can be managed with thresholds
BorderStyle	Determines the style of the Border, which can be Solid or Broken. The value can be assigned to a whole variable or it can be managed with thresholds

Tabella 54: Properties of a TrendViewXY

Properties	Description
Hide	Determines whether the object is initially visible. You can also assign a Boolean variable (for Runtime changes) or it can be managed with thresholds
Lock	Determines if the object can move or not

Edit the TrendXY

After having inserted a "TrendXY" on a page, double-click it to start its editing. A "TrendXY Graphic" will be present by default which, upon the programmer's choice, can be accompanied by a range of control buttons to view the "TrendXY". The editing is organised on two masks: "Fields" and "General".



From the "Fields" mask, the buttons which should be present together with the table can be indicated and positioned inside of the area. Each button has its relative properties which can be edited in "Editor Properties" as for normal touch buttons (see chapter 6). To insert or remove a button, click the list of buttons at the left of the table. If an object is already present on the page, it will be highlighted inside of the list (and it will be visible inside of the drawing area). To move an element (button or table) drag it to the desired position. There are several buttons that can be inserted and each one has a pre-determined function associated to it (unchangeable) :

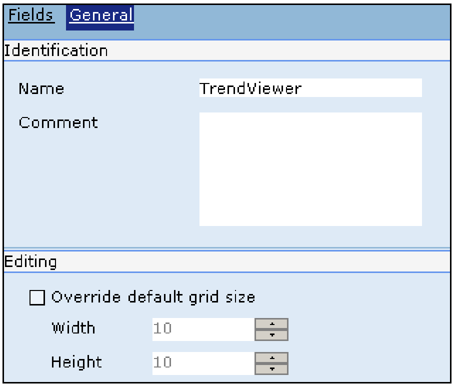
- “"Move to the Left": the button has two functions depending on whether or not the cursor is viewed: if the

- cursor is invisible, the key pressure scrolls the graphic from right to left. If the cursor is
- visible, move the cursor to the left. When it reaches the left edge, the graphic scrolls from the right
- to the left of a unit specified by the greater horizontal division of the grid.
- "Move to the Right": the button has two functions depending on whether or not the cursor is viewed: if the
- cursor is invisible, the key pressure scrolls the graphic from left to right. If the cursor is
- visible, move the cursor to the left. When it reaches the right edge, the graphic scrolls from the left
- to the right of a unit specified by the greater horizontal division of the grid.
- "Move up": the button scrolls the graphic from the bottom to the top of a unit specified by the
- greater vertical division of the grid.
- "Move down": the button scrolls the graphic from the top to the bottom of a unit specified by the
- greater vertical division of the grid.
- "Initial page": the button scrolls the graphic from right to left until the oldest samples are positioned
- on the left of the graphic.
- "End": the button scrolls the graphic from left to right until the newest samples are positioned
- on the right of the graphic.
- "Go to": the button displays a dialogue box to ask the user which time and date must be
- placed at the right of the graphic.
- "Vertical Size Increase": increases the vertical scale factor
- "Vertical Size Decrease": decreases the vertical scale factor
- "Horizontal Size Increase": increases the horizontal scale factor
- "Horizontal Size Decrease": decreases the horizontal scale factor
- "Zoom": Increases the total graphic display
- "Reduction": Decreases the total graphic display
- "Reset zoom": restores the original scale factors (no zoom).
- "User Button": Button to which the user can assign a function/script.
- "Show Pen Scale": determines the Scale pen to be shown by means of the pull-down menu

Also present: three "Bistable Buttons", a "Time Date field" and two "Numerical Fields" which can be edited as described in this chapter (see chap. 6, “Bistable button” page 272, see chap. 6, “DateTime field” page 232 e see chap. 6, “Numerical Field” page 215) Each one of them has a particular function :

- "Cursor Enabled": allows the graphic cursor to be displayed or not.
- "Break": represents if the graphic update is enabled or not (it does not disable
- sample acquisition).
- "Pen Enabled": Displays or hides the selected pen.
- "Time Date Field": Displays the sample instant.
- "Numerical Field" 1: "X" value.
- "Numerical Field" 2: "Y" value.

As already described in this chapter, a further customised label can be inserted on the complex field (see chap. 6, “Complex label” page 190) or go and insert a dynamic field which shows the Trend value on the pen (see chap. 6, “Trend Pen” page 191).



Name: TrendViewer
Comment:

☐ Override default grid size
Width: 10
Height: 10

An identification name and comment for the "Trend" that is being edited can be inserted on the "General" mask. The page editing grid default dimensions can be overwritten as well (see chap. 5, “Main window” page 75) introducing new measurements in pixel, valid only for the current field editing.

Properties of a Trend Graph XY

Tabella 55: Properties of a Trend Graph XY

Properties	Description
Top	Vertical position coordinate
Left	Horizontal position coordinate
Width	Dimension of the width
Height	Dimension of the height
AreaColor	Determines the color of the Area, which can be selected using the RGB code or color palette. The value can be assigned to a whole variable or it can be managed with thresholds
AreaVisibility	Determines whether the Graph has a background area (True) or should be transparent (False); a Boolean variable can be assigned to this value or it can be managed with thresholds
BorderVisibility	Determines whether there will be a Border to the Table or not; a Boolean variable can be assigned to this value
BorderSize	Determines the size of the Border which must be a number to which a whole variable can be assigned, if so desired or it can be managed with thresholds
BorderColor	Determines the color of the Border using the RGB code or color palette. The value can be assigned to a whole variable or it can be managed with thresholds
BorderStyle	Determines the style of the Border, which can be Solid or Broken. The value can be assigned to a whole variable or it can be managed with thresholds
Border3D	Determines a 3D effect for the Border, which can be Flat, Relief, Recessed, Bump or Etched. The value can be assigned to a whole variable or it can be managed with thresholds

Tabella 55: Properties of a Trend Graph XY

Properties	Description
BorderBlink	Determines the flashing of the Border, which can be No blinking, Slow blinking or Rapid blinking. The value can be assigned to a whole variable or it can be managed with thresholds
UpdateMode	Indicates the way the Trend display is updated: automatically, with a change of value or on command
RefreshTime	Indicates the refresh period of the trend expressed in milliseconds
HorScaleVisible	Indicates whether there needs to be a horizontal scale
HorScaleMode	Indicates the way the scale should be displayed. The Date alone, the Time alone, both or tenths of seconds can be represented
HorScaleDateFormat	Active if the type of scale envisages the Date and permits its format to be specified
HorScaleTimeFormat	Active if the type of scale envisages the Time and permits its format to be specified
HorScaleLabelFont	Indicates the Font for the label texts of the horizontal scale
HorScaleLabelColor	Indicates the color for the label texts of the horizontal scale; these can be selected using the RGB code or color palette
HorScaleLabelSkip	Indicates the frequency with which the horizontal scale labels should be inserted
VerScaleVisible	Indicates whether there should be a vertical scale
VerScaleVisible Number	Number of digits to show on the vertical scale
VerScaleLabel Decimal	Number of decimal digits to show on the vertical scale

Tabella 55: Properties of a Trend Graph XY

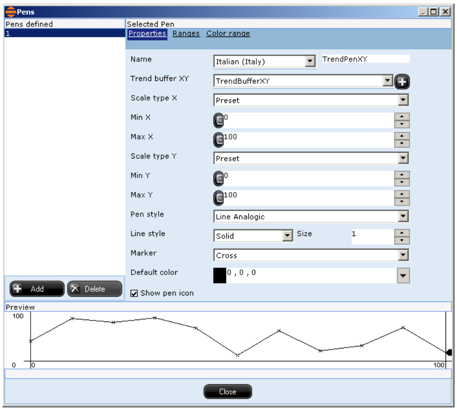
Properties	Description
ScrollType	Indicates how the scroll movement of the table should operate: may be continuous, half screen or full screen
TimeSpan	Length of time periods expressed in thousandths of a second. If the value 10000 is entered, for example, at any point the trend table will display the values gathered in 10 seconds
GridHorVisible	Indicates whether there should be a horizontal grid
GridHorDivision Number	Indicates the number of horizontal divisions in the grid
GridHorMinDivision Number	Indicates the number of horizontal subdivisions in the grid, that is, the number of horizontal lines between any two divisions
GridHorDivisionStyle	Indicates the style of the divisions of the horizontal grid: may be Solid or Broken line
GridHorMinDivision Style	Indicates the style of the subdivisions of the horizontal grid: may be Solid or Broken line
GridHorDivisionColor	Indicates the color of the divisions of the horizontal grid, can be done using the RGB code or the color palette
GridHorMinDivision Color	Determines the color of the horizontal grid subdivisions, which can be selected using the RGB code or color palette.
GridVerVisible	Indicates whether there needs to be a vertical grid
GridVerDivision Number	Indicates the number of vertical divisions in the grid
GridVerMinDivision Number	Indicates the number of vertical subdivisions in the grid, that is, the number of horizontal lines between any two divisions

Tabella 55: Properties of a Trend Graph XY

Properties	Description
GridVerDivisionStyle	Indicates the style of the divisions in the vertical grid: may be Solid or Broken line
GridVerMinDivision Style	Indicates the style of the subdivisions in the vertical grid: may be Solid or Broken line
GridVerDivisionColor	Determines the color of the vertical grid divisions, which can be selected using the RGB code or color palette.
GridVerMinDivision Color	Determines the color of the vertical subdivisions of the grid, which can be selected using the RGB code or color palette.
Pens	Indicates the pens to use in representing the trend. By clicking on the … key you can edit the types of pen (as shown in the next subsection)

Editing of the Trend Pens

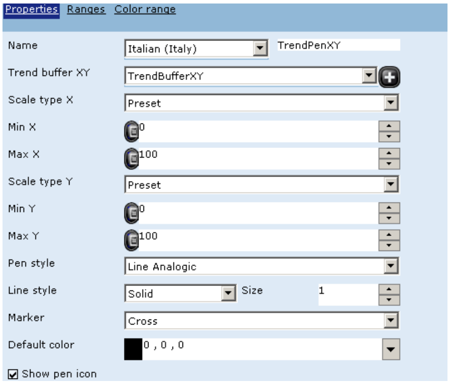
To access the "TrendXY" writing pens editing, enter the TrendXY" complex field editing (double-click it on the page). After having selected the "TrendXY Graphic", click on the icon … in the "Pens" voice in the relative "Editor Properties".



Name: Italian (Italy) TrendPenXY
Trend buffer XY: TrendBufferXY
Scale type X: Preset
Min X: 0
Max X: 100
Scale type Y: Preset
Min Y: 0
Max Y: 100
Pen style: Line Analogic
Line style: Solid Size 1
Marker: Cross
Default color: ■ 0 , 0 , 0
☑ Show pen icon

The displayed pens editing window is made up of three sections. On the left, there is a list of pens created by the user from which to create and eliminate elements. At the bottom there is a preview of the pen currently being edited. At the centre of the window there is the actual editing area for the selected pen. This window is divided in "Properties", "Intervals" and "Interval Colours" masks. They will be described in the following sub-paragraphs.

Properties



Name	Italian (Italy)	TrendPenXY	
Trend buffer XY	TrendBufferXY		
Scale type X	Preset		
Min X	0		
Max X	100		
Scale type Y	Preset		
Min Y	0		
Max Y	100		
Pen style	Line Analogic		
Line style	Solid	Size	1
Marker	Cross		
Default color	0 , 0 , 0		

☑ Show pen icon

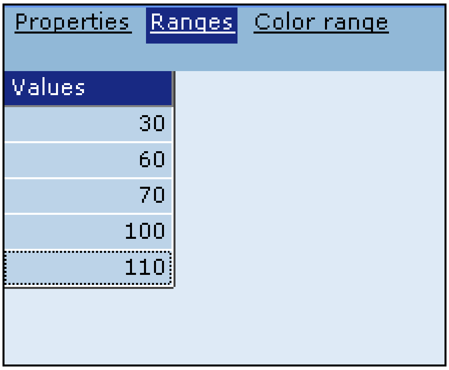
First of all, a "Name" and a "Trend bufferXY", for which the "Pen" must be used, can be associated to the Trend Pen. Indicate a value "Scale type" that can be :

- "Programmed": the Max and Min values that can also be associated to variables must be indicated
- "Automatic": calculated by Runtime, based on the values contained in the Buffer (but the limits can also be inserted)
- "By the tag limits": the Buffer must refer to a limited variable (see chap. 5, “Limiti” page 95)
- "Client": the Max and Min values must be indicated

From a graphical point of view, the aspect of the "Pen" can be established. It can be "Signal only", have an "Analogue" dash

(continuous, with oblique connections between the values) or "Digital" (scaled, with digital steps). Even the dimensions of the line, the colour and the style ("Solid", "Dash", "Dot", "Dash Dot", "Dash Dot Dot") can be edited at will.
The pen marker can assume different geometrical shapes ("Pixel", "Cross", "Plus, "Cross and plus" and "circle") and one can choose to not show the icon relative to the pen.
Every variation will update the preview at the back of the mask.

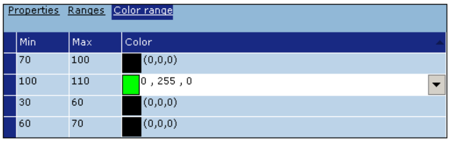
Intervals



Values
30
60
70
100
110

From this mask, insert the values relative to the intervals to which different representation colours can be attributed. The interval scale must present values in increasing order.

Interval Colours

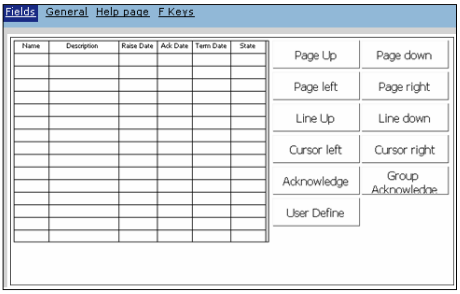


Min	Max	Color
70	100	■ (0,0,0)
100	110	🟩 0 , 255 , 0
30	60	■ (0,0,0)
60	70	■ (0,0,0)

In this mask, a colour can be associated to each of the value intervals described in the Interval mask. The colour is applied when the value read by Buffer is contained in the relative interval.

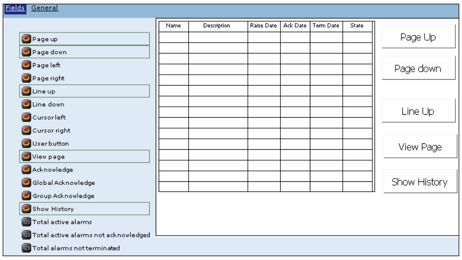
Active Alarm View

Active Alarm View is a predefined element in POLYMATH, one that can be inserted into the project pages. It allows the operator to access the alarm list and perform the principal operations with a simple click. To insert an Active Alarm table into a page, click on the icon ⚠ or, alternatively, use the main menu: Fields->Create->Complex Controls->ActiveAlarm-View. After clicking draw just its outline in the page and the table appears automatically.



Name	Description	Raise Date	Ack Date	Term Date	State

Once the table has been inserted into the page and been selected, a series of properties contained in the Properties Editor can be attributed to it; the meanings of these properties are identical to those of TrendView properties (see chap. 6, "Properties of a TrendView" page 276).
By double-clicking on the table, you access its editing page which comprises two masks: Fields and General.



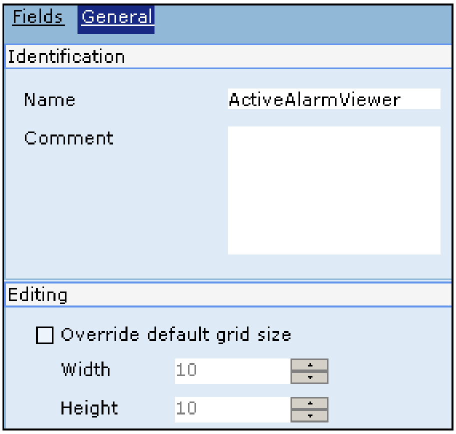
Name	Description	Raise Date	Ack Date	Term Date	State

The default contents of the Fields mask include the Alarm Grid table, whose properties will be dealt with in the next subsections (see chap. 6, “Properties of the Active Alarm Grid” page 300). Using this mask you can proceed to indicate which buttons should be present with the table and position them within the area. To insert or remove a button just click on the list of buttons to the left of the table; if an object is already present in the page, it will appear highlighted within the list (and will be visible in the Table Edit Area). To move an element (button or table) just drag it to the desired position. The buttons that can be inserted are different and each has a predefined (non editable) function assigned to it:

- Page Up: allows the operator to go up the pages of the table
- Page Down: allows the operator to go down the pages of the table
- Page Left: allows the operator to move left within the page
- Page Right: allows the operator to move right within the page
- Line Up: select the line above the current one
- Line Down: select the line below the current one
- Cursor Left: move the table cursor leftwards
- Cursor Right: move the table cursor rightwards
- User button: this button can have a user-chosen function or a script assigned to it (see chap.‘‘Appendix B - Predefined functions” page 563 and see chap. 9, “Scripts” page 379)
- Show Page: displays the page assigned to the alarm (see chap. 5, “Properties” page 122)
- Acknowledgement: acquires the selected alarm
- Global Acknowledgement: allows the operator to perform a global (cumulative) acquisition of all the alarms present in the table, if this option has been enabled for the alarm in question (see chap. 5, “Properties” page 122)
- Group Acknowledgement: allows the operator to perform a global (cumulative) acquisition of all the alarms in the table that belong to the same group as the one selected, if this option has been enabled for the alarm in question (see chap. 5, “Properties” page 122)
- Show History: shows the page containing the Alarm History. Enter the appropriate Events Editor and indicate the name of the page to go to after pressing this key

In addition, Dynamic fields can be assigned to the system variables related to the alarms, each of which has properties that can be edited using the Properties Editor (see chap. 6, “Label” page 187). These fields are:

- Total of active alarms: shows the total number of active alarms (not yet acknowledged or not yet terminated)
- Total of active alarms not acknowledged: shows the total number of alarms not acknowledged
- Total of alarms not returned: shows the total number of alarms not terminated (still present in the device)



Name: ActiveAlarmViewer

Comment:

□ Override default grid size

Width: 10

Height: 10

The General mask can be used to insert a name and an identifying comment for the Alarm table being edited. In addition, you can overwrite the default dimensions of the editing grid of the page (see chap. 5, “Main window” page 75) introducing new measures in pixels valid only for editing the current field The graphic properties (fonts and colors) of the Active Alarm View grid can be configured by using together the Fields and the Priorities mask of the Alarms (see chap. 5, “Fields” page 104 and see chap. 5, “Priorities” page 119).

Properties of the Active Alarm Grid

Tabella 56: Properties of the Active Alarm Grid

Properties	Description
Top	Vertical position coordinate
Left	Horizontal position coordinate
Width	Dimension of the width
Height	Dimension of the height

Tabella 56: Properties of the Active Alarm Grid

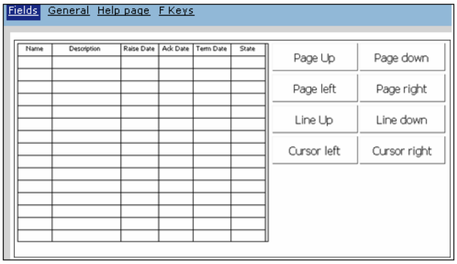
Properties	Description
RowHeight	Determines height in pixels of each row
TimeStampOrder	Indicates the chronological order in which to arrange the alarms in the grid; may choose to show the most recent ones first, or the oldest ones
RibbonFlag	Allows the operator to decide whether or not to display the index numbers of the alarm (in the columns to the left of the table)
AutoScrollEnabled	Indicates whether table scrolling should be enabled automatically.
AutoScrollInterval	Active if autoscroll is enabled. Sets the number of lines for the autoscroll interval.
LetterHeadFlag	Indicates whether the tables should have titles.
Filters	This field allows the operator to insert filtering parameters for the alarms to be displayed within the table. To apply these filters click on the icon ... . In Runtime only the alarm instances respecting the conditions indicated in the Filters window will be shown. If more than one filter is set, only the alarm instances respecting the limits (AND conditions) will be shown in runtime.
Columns	This field allows the operator to determine which columns to put in the table and define their respective properties. To edit the columns click on the icon ... . In the window which appears enter the details relating to their width, to the font and to the dimension and format of the titles of each column in the table.
Lock	Determines whether the object can move or not
TabIndex	Determines the index that the object will occupy in the table order

Tabella 56: Properties of the Active Alarm Grid

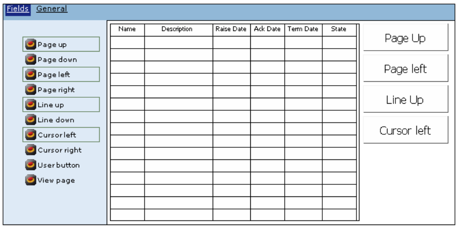
Properties	Description
HScrollBarVisible	Indicates whether the horizontal scroll bar should be visible in Runtime.
VScrollBarVisible	Indicates whether the vertical scroll bar should be visible in Runtime.

Alarm History View

The Alarm History table is a predefined element in POLYMATH, one that can be inserted into the project pages. It allows the operator to access the active alarm list and perform the principal operations with a simple click. To insert an Alarm History View table into a page, click on the icon or, alternatively, use the main menu: Fields->Create->Complex Controls->Alarm History. This table contains only those alarms whose configuration explicitly says that they are to be saved in the terminal's Alarm History (see chap. 5, "Properties" page 122). After clicking on the table draw just its outline in the page and the table will appear automatically.

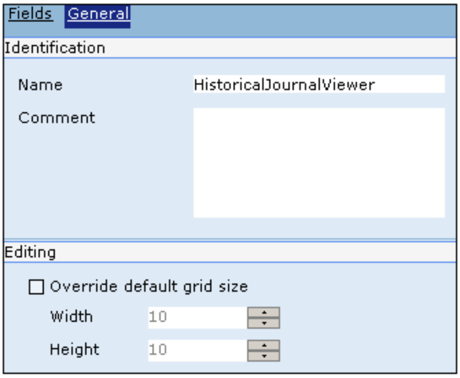


Once the table has been inserted into the page and been selected, a series of properties can be attributed using the Properties Editor; the meanings of these properties are identical to those of the properties in TrendView (see chap. 6, "Properties of a TrendView" page 276).
By double-clicking on the table itself you access its editing page which comprises two masks: Fields and General.



The default contents of the Fields mask include the Alarm Grid table, whose properties will be dealt with in the next subsections (see chap. 6, “Properties of the Active Alarm Grid” page 300). Using this mask you can proceed to indicate which buttons should be present with the table and position them within the area. To insert or remove a button just click on the list of buttons to the left of the table; if an object is already present in the page, it will appear highlighted within the list (and will be visible in the Table Edit Area). To move an element (button or table) just drag it to the desired position. The buttons that can be inserted are different and each has a predefined (non editable) function assigned to it:

- Page Up: allows the operator to go up the pages of the table
- Page Down: allows the operator to go down the pages of the table
- Page Left: allows the operator to move left within the page
- Page Right: allows the operator to move right within the page
- Line Up: select the line above the current one
- Line Down: select the line below the current one
- Cursor Left: move the table cursor leftwards
- Cursor Right: move the table cursor rightwards
- User button: this button can have a user-chosen function or a script assigned to it (see chap.‘‘Appendix B - Predefined functions” page 563 and see chap. 9, “Scripts” page 379)
- Show Page: displays the page assigned to the alarm (see chap. 5, “Properties” page 122)



Fields General

Identification

Name HistoricalJournalViewer

Comment

Editing

□ Override default grid size

Width 10

Height 10

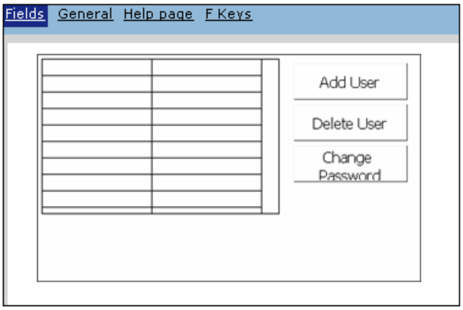
The General mask can be used to insert a name and an identifying comment for the Alarm History table being edited. In addition, you can overwrite the default dimensions of the editing grid of the page (see chap. 5, “Main window” page 75) introducing new measures in pixels valid only for editing the current field.
The graphic properties (fonts and colors) of the Active Alarm View grid can be configured using together the Fields and Priorities masks of the Alarm (see chap. 5, “Fields” page 104 and see chap. 5, “Priorities” page 119).

Properties of the Alarm History Grid

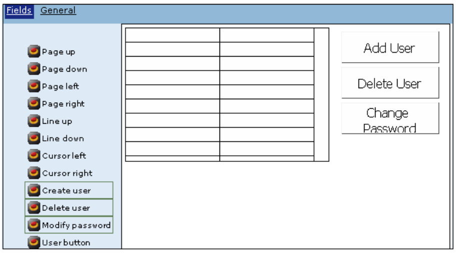
The properties of the Alarm History Grid coincide with those of the Active Alarm Grid (see chap. 6, “Properties of the Active Alarm Grid” page 300).

User List Table

The User List table (see chap. 5, “Password configuration” page 81) is a predefined element in POLYMATH, one that can be inserted into the project pages. It allows the operator to access the user list (respecting the limits of its level of protection) and perform the principal operations with a simple click.
To insert a User List table into a page, click on the icon ▦ or, alternatively, use the main menu: Fields->Create->Complex Controls->User list.
After clicking draw just the outline of the table and it will appear automatically.



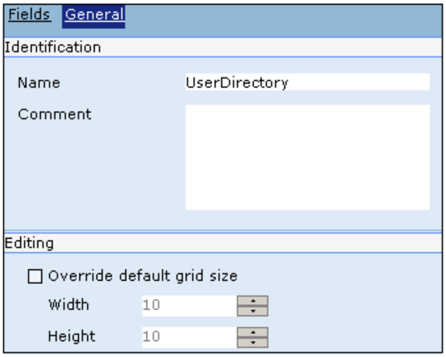
Once the table has been inserted into the page and been selected, a series of properties contained in the Properties Editor can be attributed to it; the meanings of these properties are identical to those of the properties in TrendView (see chap. 6, “Properties of a TrendView” page 276).
By double-clicking on the table itself you access its editing page which comprises two masks: Fields and General.



The default contents of the Fields mask include the Alarm Grid table, whose properties will be dealt with in the next subsections (see chap. 6, “Properties of the Active Alarm Grid” page 300). Using this mask you can proceed to indicate which buttons should be present with the table and position them within the area. To insert or remove a button just click on the list of buttons to the left of the table; if an object is already present in the page, it will appear highlighted within the list (and will be visible in the Table Edit Area). To move an element (button or table) just drag it to the desired position. The buttons that can be inserted are different and each has a predefined (non editable) function assigned to it:

- Add User: allows the operator to add a new user
- Delete User: allows the operator to remove the user selected

- Change Password: allows the operator to change the password relating to the user selected



Identification

Name: UserDirectory

Comment:

Editing

□ Override default grid size

Width: 10

Height: 10

The General mask can be used to insert a name and an identifying comment for the User table being edited. In addition, you can overwrite the default dimensions of the editing grid of the page (see chap. 5, “Main window” page 75) introducing new measures in pixels valid only for editing the current field The graphic properties (fonts and colors) of the User List can be configured using together the Fields mask and Password element (see chap. 5, “Fields grid” page 83).

Note: *After inserting a new user in Runtime, you will have to change his/her password by selecting the corresponding row in the table and then by clicking on ‘Change Password’. Just insert the new password in the ensuing mask, leaving blank the field relating to the old password (since the new user does not possess any assigned password).*

Properties of the Password Grid

Tabella 57: Properties of the Password Grid

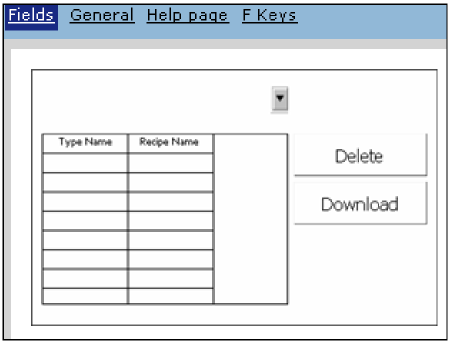
Properties	Description
Top	Vertical position coordinate
Left	Horizontal position coordinate
Width	Dimension of the width
Height	Dimension of height

Tabella 57: Properties of the Password Grid

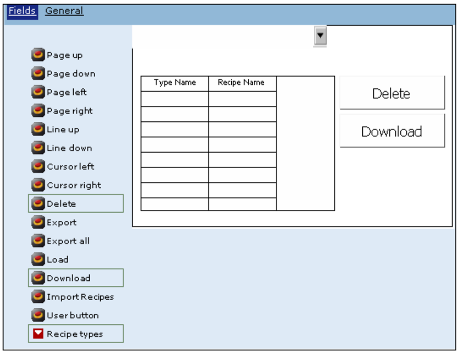
Properties	Description
RowHeight	Determines height in pixels of each row
HScrollBarVisible	Indicates whether the horizontal scroll bar should be visible in Runtime.
VScrollBarVisible	Indicates whether the vertical scroll bar should be visible in Runtime.
Columns	This field allows the operator to edit the appearance of the table. To edit the columns click on the icon .... In the window which appears enter the details relating to their width, to the font and to the dimension and format of the titles of each column in the table
Lock	Determines whether the object can move or not
TabIndex	Determines the index that the object will occupy in the table order

Recipe List Table

The Recipe List table is a predefined element in POLYMATH, one that can be inserted into the project pages. It allows the operator to access the Recipe list in the terminal (see chap. 5, “Recipes” page 124). To insert a Recipe List table into a page, click on the icon or, alternatively, use the main menu: Fields->Create->Complex Controls->Recipe list.
After clicking draw just the outline of the table and it will appear automatically.



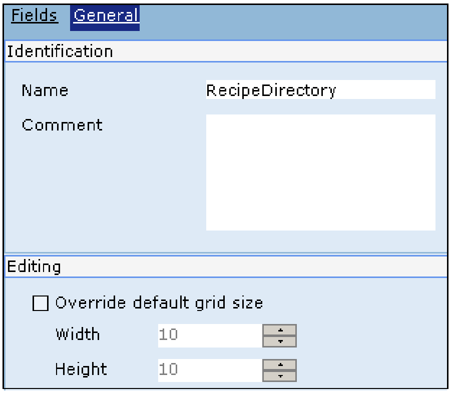
Once the table has been inserted into the page and been selected, a series of properties contained in the Properties Editor can be attributed to it; the meanings of these properties are identical to those occurring in TrendView (see chap. 6, “Properties of a TrendView” page 276).
By double-clicking on the table, you access its editing page which comprises two masks: Fields and General.



The default contents of the Fields mask include the Alarm Grid table, whose properties will be dealt with in the next subsections (see chap. 6, “Properties of the Active Alarm Grid” page 300). Using this mask you can proceed to indicate which buttons should be present with the table and position them within the area. To insert or remove a button just click on the list of buttons to the left of the table; if an object is already present in the page, it will appear highlighted within the list (and will be visible in the Table Edit Area). To move an element (button or table) just drag it to the desired position. The buttons that can be inserted are different and each has a predefined (non editable) function assigned to it:

- Delete: deletes the Recipe selected
- Export: exports the Recipe selected into a .csv file
- Export all Recipes: exports all the Recipes in the table into a .csv or .xml file
- Transfer (download): downloads the Recipe selected onto a device
- Import Recipes: imports the Recipes from a .csv file

In addition, a Dynamic field can be inserted, which contains the Recipe type list in a pull-down menu which allows the operator to filter the display for a specific type of Recipe. The properties relating to position and dimension can be inserted into the Properties Editor of this field, and it is also possible to indicate which type of Recipe to display as default when the page opens.



Identification

Name: RecipeDirectory

Comment:

Editing

□ Override default grid size

Width: 10

Height: 10

The General mask can be used to insert a name and an identifying comment for the Recipe table being edited. In addition, you can overwrite the default dimensions of the editing grid of the page (see chap. 5, “Main window” page 75) introducing new measures in pixels valid only for editing the current field. The graphic properties (fonts and colors) of the Recipe list can be configured using the Fields mask of the Recipes element (see chap. 5, “Fields” page 104).

Properties of the RecipeGrid

Tabella 58: Properties of the Recipe Grid

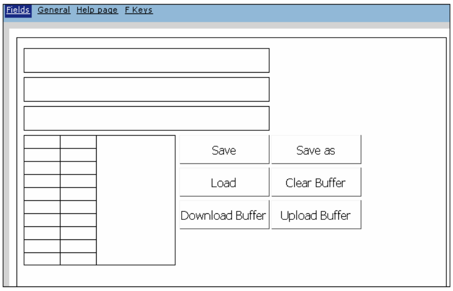
Properties	Description
Top	Vertical position coordinate
Left	Horizontal position coordinate
Width	Dimension of the width
Height	Dimension of the height
RowHeight	Determines height in pixels of each row
LetterHeadFlag	Indicates whether the tables should have titles

Tabella 58: Properties of the Recipe Grid

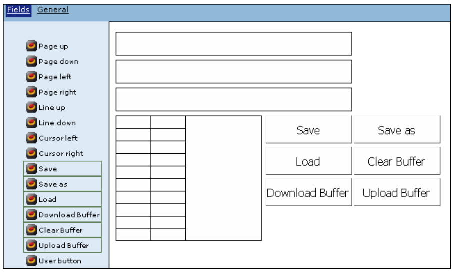
Properties	Description
HScrollBarVisible	Indicates whether the horizontal scroll bar should be present in Runtime when the dimensions of the instances allow for it
VScrollBarVisible	Indicates whether the vertical scroll bar should be present in Runtime when the number of instances allow for it
OrderMode	Indicates the way the instances should be ordered within the table; the order can be alphabetical, chronological order of editing and Recipe ID order
Columns	This field allows the operator to determine which columns to put in the table and define their respective properties. To edit the columns click on the icon …. In the window which appears enter the details relating to their width, to the font and to the dimension and format of the titles of each column in the table
Lock	Determines whether the object can move or not
TabIndex	Determines the index that the object will occupy in the table order

Recipe Editing Table

The Recipe Editing table is a predefined element in POLYMATH, one that can be inserted into the project pages. It allows the operator to access the Recipe editor in the terminal (see chap. 5, “Recipes” page 124). To insert a Recipe Editing table into a page, click on the icon ✎ or, alternatively, use the main menu: Fields->Create->Complex Controls->Recipe-Editing.
After clicking draw just the outline of the table in the page and it will appear automatically.



Once the table has been inserted into the page and been selected, a series of properties contained in the Properties Editor can be attributed to it; the meanings of these properties are identical to those of TrendView (see chap. 6, “Properties of a TrendView” page 276).
By double-clicking on the table, you access its editing page which comprises two masks: Fields and General.

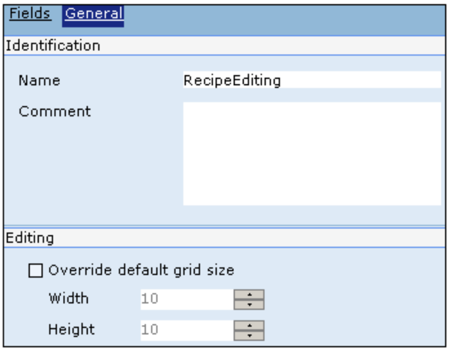


The default contents of the Fields mask include the Alarm Grid table, whose properties will be dealt with in the next subsections (see chap. 6, “Properties of the Active Alarm Grid” page 300). Using this mask you can proceed to indicate which buttons should be present with the table and position them within the area. To insert or remove a button just click on the list of buttons to the left of the table; if an object is already present in the page, it will appear highlighted within the list (and will be visible in the Table Edit Area). To move an element (button or table) just drag it to the desired position. The buttons that can be inserted are different and each has a predefined (non editable) function assigned to it:

- Save: saves the Recipe in the terminal memory (if necessary, overwriting the one being edited)
- Save as: saves the Recipe open in any case the insertion window of the name
- Load: loads the Recipe selected into the video buffer

- Delete Buffer: empty the buffer
- Transfer to Buffer (upload): transfers the Recipe from the device to the video buffer
- Transfer from Buffer (download): transfers the data of the Recipe present from the video buffer to the device

In addition, dynamic fields can be inserted which contain a Recipe list and the name of the uploaded Recipe which has same properties as the Label objects (see chap. 6, "Label" page 187) that can be edited using the Properties Editor.



Identification

Name RecipeEditing

Comment

Editing

□ Override default grid size

Width 10

Height 10

The General mask can be used to insert a name and an identifying comment for the Recipe table being edited. In addition, you can overwrite the default dimensions of the editing grid of the page (see chap. 5, "Main window" page 75) introducing new measures in pixels valid only for editing the current field. The graphic properties (fonts and colors) of the Recipe list can be configured using the Fields mask of the Recipes element (see chap. 5, "Fields" page 104).

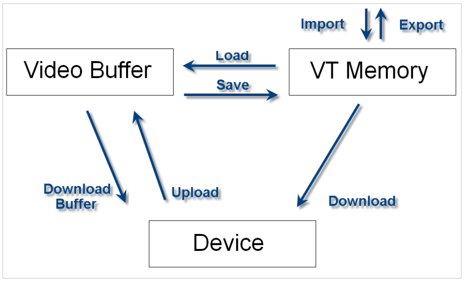
Properties of the RecipeGrid

Tabella 59: Properties of the RecipeGrid

Properties	Description
Top	Vertical position coordinate
Left	Horizontal position coordinate
Width	Dimension of the width
Height	Dimension of the height
RowHeight	Determines height in pixels of each row
HScrollBarVisible	Indicates whether the horizontal scroll bar should be present in Runtime when the dimensions of the instances allow for it
VScrollBarVisible	Indicates whether the vertical scroll bar should be present in Runtime when the number of instances allow for it
Columns	This field allows the operator to determine which columns to put in the table and define their respective properties. To edit the columns click on the icon … . In the window which appears enter the details relating to their width, to the font and to the dimension and format of the titles of each column in the table
ShowRecipeType	Allows the operator to specify the Recipe type whose instances are displayed within the table
Lock	Determines whether the object can move or not
TabIndex	Determines the index that the object will occupy in the table order
Disable	Indicates whether the field should be disabled
OrderMode	Indicates the way the instances should be ordered within the table; the order can be alphabetical, chronological order of editing and Recipe ID order

Operations for transferring Recipes

The following summary gives an overview of all the operations that can be performed on transfers of Recipes using VTs and devices. It is important to note that transfer operations see the interaction of 3 elements: the physical memory of the VT (where the Recipes are saved), the VT video buffer (containing the data of just one Recipe, the one being displayed on the panel) and the device (in whose memory the Recipe data really resides).



When you decide to manage the transfer of Recipes in synchronized mode constitutes a special case.
In this case, before transferring the data the terminal asks for the status of the device, waiting for an authorization. The synchronization procedure happens by means of the write/read of certain exchange areas (see chap.‵‵Appendix C - Status area” page 575 and see chap.‵‵Appendix D - Command area” page 579).
A synchronized transfer is defined at the moment the function is attributed (or the script instruction, see chap. 9, “Scripts” page 379)

Let us give a practical example. Supposing we are performing a synchronized download and are using a Recipe type in noncompatible mode (see chap. 5, “Modes of compatibility” page 125), then, at the request for a transfer of data, the devices will behave as follows:

- the VT will send the data transfer request to the PLC (WORD0.BIT1 of the Status Area)
- the PLC responds, enabling the transfer using bit 4 of WORD0 of the Command Area

- At this point, the data transfer will begin (WORD0.BIT0 of the Status Area)
- At the end of the transfer, the VT will signal to the PLC (WORD0.BIT3 of the Status Area) that the download has terminated
- the PLC will respond confirming the reception (WORD0.BIT0 of the Command Area)

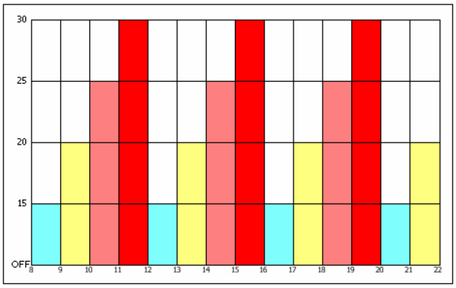
If during the data transfer the handshake times are not respected, the VT puts at 1 in the Status Area the Error In Transferring bit (bit 14 = download, bit 15 = upload).

Chronothermostat

A "Cronotermostato" (Chronothermostat) can be inserted inside of the page by clicking on the 🌡 icon or from the "Main Menu" (Fields->Create->Complex Controls->Chronothermostat). After having clicked the icon, indicate the area in which POLYMATH must designate the "Chronothermostat" using the mouse inside of the page.

A "Cronotermostato" (Chronothermostat) represents a Polymath object that allows to detect a temperature and edit the behaviour of the system in "Manuale" (Manual) or "Automatico" (Automatic) mode at will. Scheduling allows to program the temperature trend weekly.

The "Cronotermostato" (Chronothermostat) is the field inside of which the content of the TaskSettimanali (WeeklyTasks) is displayed, the functioning of which was described in the last previous paragraph (see chapter 7 "TaskSettimanali" on page Consult the next sub-paragraphs to know the details of the properties that can be associated to a "Cronotermostato" (Chronothermostat) and its editing modes.



Properties of the Chronothermostat Grid

Tabella 60: Properties of the Chronothermostat Grid

Properties	Description
Comment	Identification comment inside of POLYMATH
Name	Identification name of the Chronothermostat View. It must be unique among the graphic elements
Height	Height dimension
Left	Horizontal position coordinate
Top	Vertical position coordinate
Width	Width dimension
AreaColor	Determines the color of the Area, which can be selected using the RGB code or color palette. The value can be assigned to a whole variable or it can be managed with thresholds
AreaVisibility	Determines whether the Chronothermostat has the background area (True) or if it must be transparent (False). A Boolean variable can be associated to this value or it can be managed with thresholds
Border3D	Determines the 3D effect of the Border. It can be Flat, Relief, Rec-essed, Bump or Etched. The value can be associated to a whole variable or it can be managed with thresholds
BorderBlink	Determines the flashing of the Border. It can be No Flash, Slow Flash or Fast Flash. The value can be associated to a whole variable or it can be managed with thresholds
BorderColor	Determines the Border colour by means of the RGB code or the colour palette. The value can be associated to a whole variable or it can be managed with thresholds

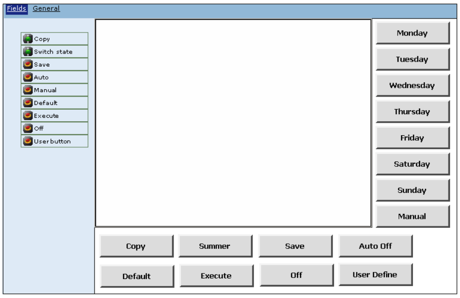
Tabella 60: Properties of the Chronothermostat Grid

Properties	Description
BorderVisibility	Determines whether the Viewer border is present or not. A Boolean variable can be associated to this value
BorderSize	Determines the dimension of the Border. It must be a number to which a whole variable can be associated optionally, or it can be managed with thresholds
BorderStyle	Determines the style of the Border. It can be Solid or Dashed. The value can be associated to a whole variable or it can be managed with thresholds
Hide	Determines whether the object is visible initially. It can also be associated to a Boolean variable (for Runtime modifications) or it can be managed with thresholds
Lock	Determines whether the object can move or not

Edit the Chronothermostat

After having inserted a "Cronotermostato" (Chronothermostat) on a page, double-click it to start its editing. A "graphic" will be present by default which, upon the programmer's choice, can be accompanied by a range of control buttons to view the "Cronotermostato" (Chronothermostat) The editing is organised on two masks: "Campi" e "Generale" (Fields and General)

Fields

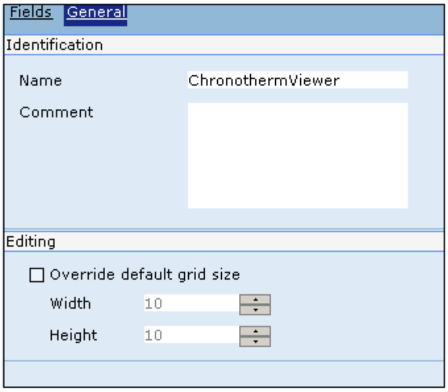


From the "Fields" mask, the buttons which should be present together with the table can be indicated and positioned inside of the area. Each button has its relative properties which can be edited in "Editor Proprietà" (Properties Editor) as for normal touch-sensitive buttons. To insert or remove a button, click the list of buttons at the left of the table. If an object is already present on the page it will be highlighted inside of the list (and it will be visible inside of the drawing area). To move an element (button or table) drag it to the desired position. Several buttons can be inserted and each one has a pre-determined function associated to it (unchangeable) :

- "Copy": the button copies the daily trend of the temperature in order to overwrite it on another day at will. This is useful when it is necessary to have the same daily schedule on different days. Functions by pressing the button and then selecting the days where the modification is to be applied. To conclude, re-select the "Copia" (Copy) key
- "Change state": the button allows to pass from heating mode to cooling mode and vice versa. This function must be activated in the "Script" della "WeeklyTask" "Script" screen used. It is only active if the selection of "Tipo di Cronotermostato" (Type of Chronothermostat) corresponds to "Riscaldamento e Raffreddamento" (Heating and Cooling)
- "Save": the button memorises and saves the modifications made to this function
- "Automatic": allows to start "Cronotermostato" (Chronothermostat) scheduling. The system reads the previously-set cycle values and behaves according to the same
- "Manual": the system leaves "Automatico" (Automatic) mode and follows a variable temperature value at will at any time from the panel

- "Default": pass to a pre-established temperature value (Default) in the design phase. The "Automatico" (Automatic) function is deactivated
- "Perform": activates "Cronotermostato" (Chronothermostat) functioning
- "Off": exists the "Cronotermostato" (Chronothermostat) functioning mode
- "User Button": identifies a new button that can be customised freely by the user

General



Fields General

Identification

Name: ChronothermViewer

Comment:

Editing

☐ Override default grid size

Width: 10

Height: 10

An identification name and comment for the "Cronotermostato" (Chronothermostat) that is being edited can be inserted on the "General" mask. Moreover, it is possible to overwrite the default dimensions of the page editing grid introducing new measurements in pixel valid only for editing the current field.

Properties of the Chronothermostat Grid

Tabella 61: Properties of the Chronothermostat Grid

Properties	Description
Comment	Identification comment inside of POLYMATH
Name	Identification name of the Chronothermostat View. It must be unique among the graphic elements
Height	Height dimension
Left	Horizontal position coordinate
Top	Vertical position coordinate
Width	Width dimension
AreaColor	Determines the color of the Area, which can be selected using the RGB code or color palette. The value can be assigned to a whole variable or it can be managed with thresholds
AreaVisibility	Determines whether the Chronothermostat has the background area (True) or if it must be transparent (False). A Boolean variable can be associated to this value or it can be managed with thresholds
Border3D	Determines the 3D effect of the Border. It can be Flat, Relief, Rec-essed, Bump or Etched. The value can be associated to a whole variable or it can be managed with thresholds
BorderBlink	Determines the flashing of the Border. It can be No Flash, Slow Flash or Fast Flash. The value can be associated to a whole variable or it can be managed with thresholds
BorderColor	Determines the Border colour by means of the RGB code or the colour palette. The value can be associated to a whole variable or it can be managed with thresholds

Tabella 61: Properties of the Chronothermostat Grid

Properties	Description
BorderVisibility	Determines whether the Viewer border is present or not. A Boolean variable can be associated to this value
BorderSize	Determines the dimension of the Border. It must be a number to which a whole variable can be associated optionally, or it can be managed with thresholds
AreaColor	Determines the colour of the area inside the graphics by means of the RGB code or the colour palette. The value can be associated to a whole variable
WeeklyType	Determines the type of use of the Chronothermostat. It can be None, Days or Week
GridLineVisible	Determines if the grid has reference lines (True) or must be transparent (False)
GridLineColor	Determines the Visible Grid Line colour by means of the RGB code or the colour palette. The value can be associated to a whole variable
GridUnusedCellColor	It determines the colour of the cells not used in the Grid by means of the RGB code or the colour palette. The value can be associated to a whole variable
OffLabel	Determines the text to display in the grid in the intersection point of the graphical lines
WeekTask	Determines the selection between TaskSettimanali present for programming
GrdiColors	Allows to use the ... button to access the configuration window of the colours relative to the graphic temperature intervals
Lock	Determines whether the object can move or not

Tabella 61: Properties of the Chronothermostat Grid

Properties	Description
TabIndex	Determines the index that the object will occupy in the table order
BackgrImageEnabled	Determines the presence (True) or not (False) of a background image in the grid
BackgroundImageID	Determines the background image used in the program
HorScaleVisible	Determines the presence (True) or not (False) of the horizontal scale in the graphics
HorScaleLabelFont	Determines the Font to use for the labels of the horizontal scale
HorScaleLabelColor	It determines the colour of the labels in the horizontal scale by means of the RGB code or the colour palette
VerScaleVisible	Determines the presence (True) or not (False) of the vertical scale in the graphics
VerScaleLabelFont	Determines the Font to use for the labels of the vertical scale
VerScaleLabelColor	It determines the colour of the labels in the vertical scale by means of the RGB code or the colour palette The value can be associated with Tag or it can be managed with thresholds

Movement properties of the objects

A movement property can be associated to every object present in the creation of the POLYMATH project.
This is used to determine the behaviour of this object associated to a "Tag".
The commands useful for introduction of a movement are present in the object "Editor Proprietà" (Properties Editor) :

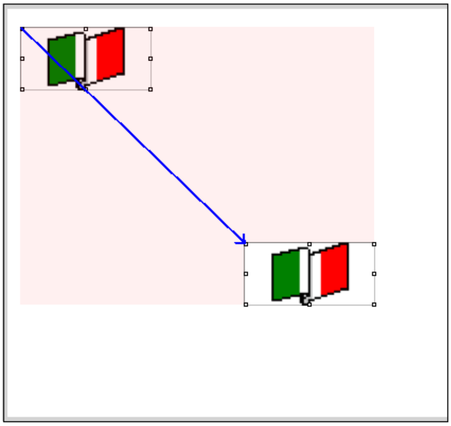
Tabella 62: Movement properties of the objects

Properties	Description
TypeOfMovement	Identifies the movement associated to the object: None, Direct, Horizontal, Vertical and Horizontal And Vertical
TagDirectMovement	Associates a Tag to the Direct movement
TagX	Horizontal movement tag
TagY	Vertical movement tag
Steps	Movement intervals
FinalX	Horizontal co-ordinate
FinalY	Vertical co-ordinate

Once an object has been selected, by clicking the button, the movement associated to it can be displayed graphically.

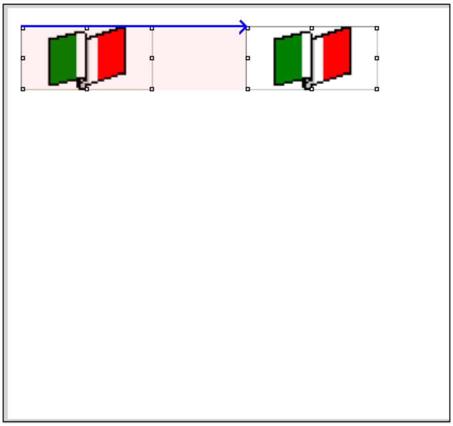
Shift the object to the desired position.

Direct Movement



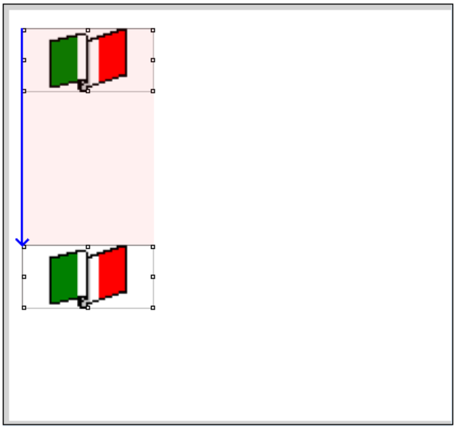
The image shifts to the pre-established point.

Horizontal Movement



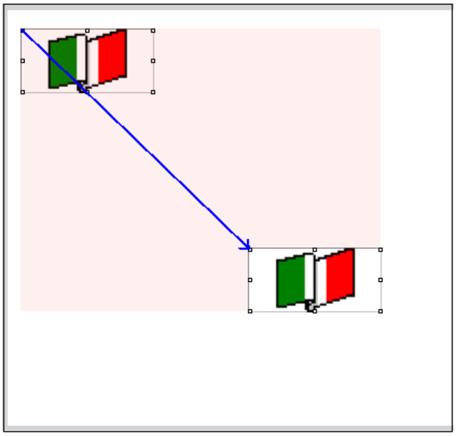
The image follows the established horizontal movement (Tag X).

Vertical Movement



The image follows the established vertical movement (Tag Y).

Horizontal and Vertical Movement



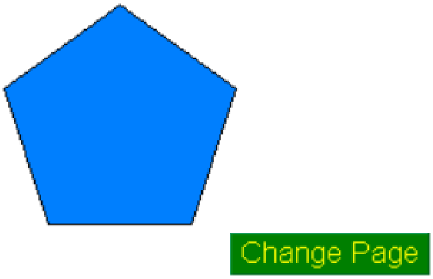
The image follows the established horizontal and vertical movements (X and Y Tags).

Operations on graphic elements

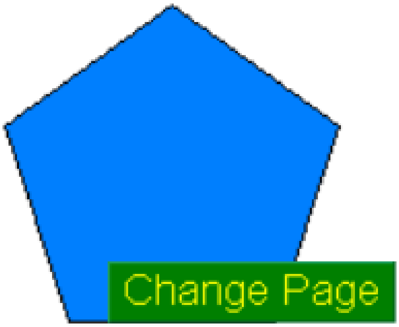
For all the graphic elements described in this chapter it is possible to perform a series of useful operations aimed at further improving the graphic presentation of the project.
In this section we will give a complete description of practical examples relating to standard operations, like grouping, alignment and distribution.

Grouping of two or more graphic elements

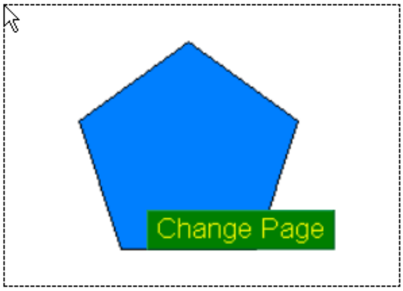
The grouping function is useful whenever you want to deal with a group of graphic elements as a single block so as to be able to perform cumulative operations on all the elements. To group two or more elements select them simultaneously (using the mouse to construct an area to enclose them) and click on the icon ⧉ of the toolbar (or Layout->Group using the Main menu).
For example, let us insert into a page a Regular 5-sided polygon and then a touch button.



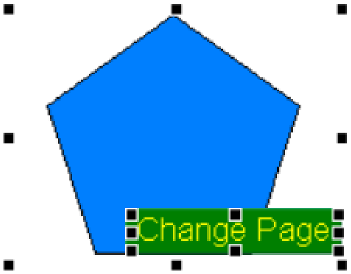
Let us suppose that after individually defining the properties of the single objects (as shown in the previous sections), we want the button to be over the Polygon and want this structure to be a single structure, so that we can move, resize or duplicate them together as a group. First of all we will move the button (selecting it and dragging it to the desired position).



Now we have to select the two objects collectively. We just click on the icon ↖ and draw a selection area inside the page big enough to contain the outlines of both objects as shown in the following figure:



at this point release the button and both elements will be selected,

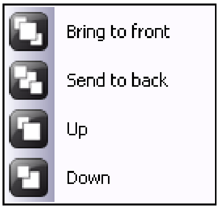


and the Group key on the toolbar will become activated ▣ (Layout->Group). By clicking on this icon the elements become grouped and the Polygon-button ensemble becomes usable as a single element. The type of object created is Group Field and, when selected, the Properties Editor can be used to attribute the properties relating to dimensions and position besides the name and identifying comment in the context of POLYMATH. Once a Group Field is created it is possible to edit the position of the objects within it (or delete and add objects) after double-clicking on the group itself. In this way you access the Group editor in which it is also possible to set the Group’s general properties and dimensions.
This operation can of course be performed on all the types of graphic element described in this chapter. Moreover, the elements of a group can be disaggregated later by clicking on the ▣ key (Layout->Separate), active only when a group of ob-

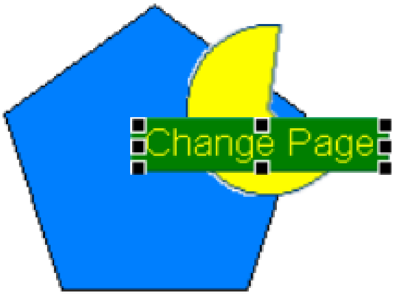
jects is selected. After their division the elements return to being separately editable.

Depth order of objects

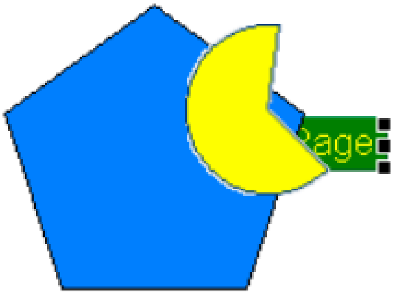
When there is an overlapping of more than one object in a page, the operator can establish display priority policy for the overlapping objects. By selecting one of the objects it is possible to determine at what depth level to position it by pressing one of the four keys also to be found in the Main menu Layout->Level.



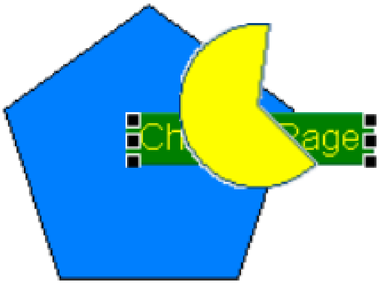
To understand the way the four options work let us take the example given in the previous section, but adding to the polygon and the touch button a third element, a Sector.
We will apply the Level commands to the touch button, then select it (click on ↖ and then on the button). Now by clicking on ⧉ (Layout->Level->Move to First Level) the button is brought to the top level above all the other objects.



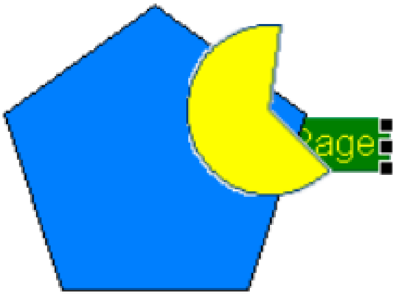
When, instead, we click on the icon ⧉ (Layout->Level->Move to Lowest Level), the button selected is taken to the bottom level, that is, below all the other objects. See figure:



Now, if we click on the icon ⧉ (Layout->Level->Up), the button is moved one level towards the top, that is, it rises only above the object that was immediately above it at that moment (in our example, the polygon).

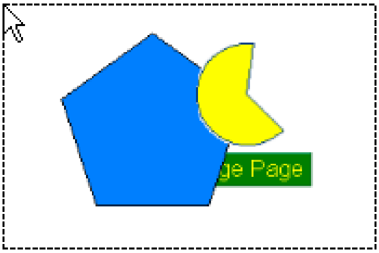


Naturally, with each click of this icon, the object selected appears at a different level.
Similarly, by clicking on the icon ⧉ (Layout->Level->Down) the button is moved down by one level, that is, it drops only below the object that was immediately below it at that moment (in our example, the polygon).

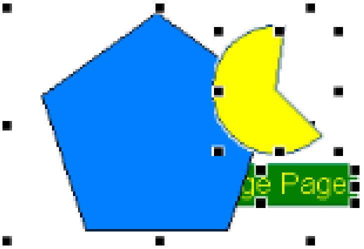


Alignment of objects

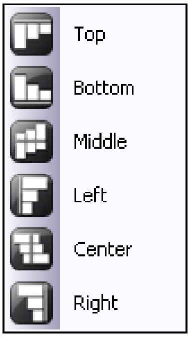
When there are two or more objects in a page the operator can use the tools supplied by POLYMATH to obtain their automatic alignment; these tools can be accessed directly via the Main menu (Layout->Align) or via the respective icons of the toolbar, as described below.
To describe the various behaviors of the Alignment function, we will use the same example we utilized in the last subsection: three elements in a page, namely a touch button, a polygon and a Sector.
First of all we enable the icon relating to alignment by clicking on the icon ↖, then we draw in the page a selection area big enough to contain the outlines of both objects. See figure:



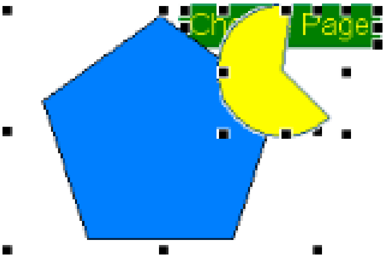
Once the mouse key is released the objects become selected and the alignment icons become clickable.



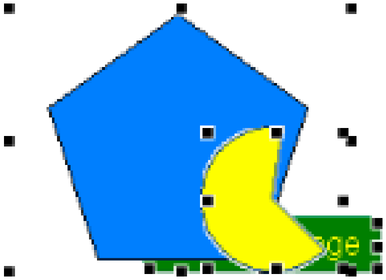
There are six icons in the Alignment Menu, each of which behaves differently, as shown below:



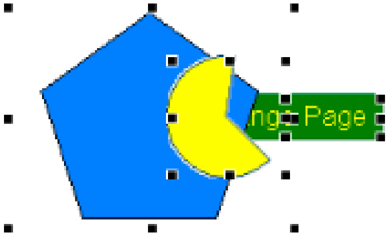
By clicking on (Layout->Align->Top), the top edges of all the figures selected are aligned with one another at the level of the top edge of the highest positioned object (in our example, the button). See figure:



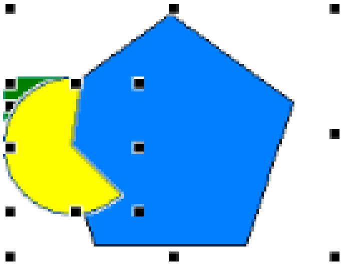
By clicking on (Layout->Align->Bottom) the lowest edges of all the figures selected are aligned with one another at the level of the bottom edge of the lowest positioned object (in our example, the button). See figure:



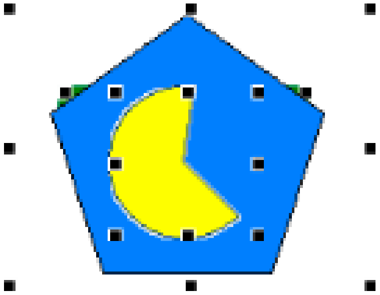
By clicking on (Layout->Align->Mid-point) the (vertical) mid-points of all the figures selected are aligned with one another at the level of the (vertical) mid-point of the lowest positioned object (in our example, the button). See figure:



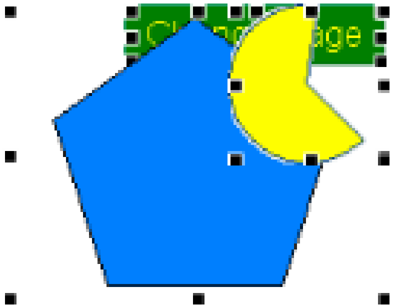
By clicking on (Layout->Align->Left) the left edges all the figures selected are aligned with one another at the level of the left edge of the leftmost object (in our example, the button). See figure:



By clicking on (Layout->Align->Center) the (horizontal) mid-points of all the figures selected are aligned with one another at the level of the (horizontal) mid-point of the lowest positioned object (in our example, the button). See figure:



By clicking on (Layout->Align->Right) the right edges all the figures selected are aligned with one another at the level of the right edge of the rightmost object (in our example, the button). See figure:

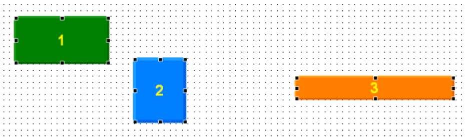


Arrangement of objects

When there are at least three objects in a page the operator can use the tools supplied by POLYMATH to obtain their automatic arrangement; these tools can be accessed directly via the Main menu (Layout->Arrange) or via the respective icons of the toolbar, as described below.
Objects are arranged within a page by taking as a point of reference the distance between the first two objects in the page. (For vertical arrangements, the reference is the distance between the first two objects encountered scrolling the page from top to bottom; for horizontal arrangements, the reference is the distance between the first two objects encountered scrolling the page from left to right).
The following subsections offer simple examples which take into consideration only three touch buttons of different dimensions but more complex configurations are dealt with in the same way.

Horizontal arrangement

Using our example, let us add three different colored buttons to the page. See below:



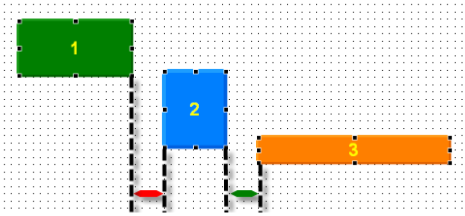
After drawing the three buttons, let us click on the ↖ icon of the toolbar and draw a selection area that includes all the objects. This activates the arrangement options of the toolbar or Main menu: Layout->Arrange.

In the examples in this subsection, our starting point to illustrate how the arrangement operation works will always be this same initial situation. For horizontal arrangements, POLYMATH takes as its reference the distance between buttons 1 and 2 (being the first two from the left).

To operate a simple horizontal arrangement, just click on the icon of the toolbar or Main menu (Layout->Arrange->Horizontally).

POLYMATH will arrange all the objects selected such that the distance between the left side of one object and the right side of the object preceding it is always equal to the distance between the left side of the second object and the right side of the first object (reference objects calculated according to their order when scrolling the page from the left). If the reference distance is less than zero, POLYMATH takes it automatically to 0.

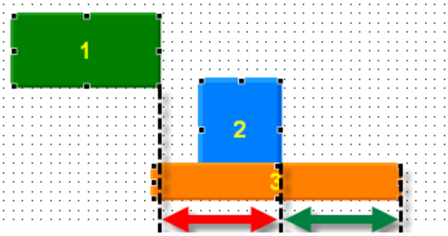
In our example, the result obtained will be the one represented in the next figure:



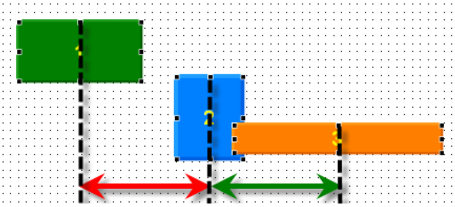
To operate a rightward arrangement, just click on icon of the toolbar or Main menu (Layout->Arrange->Right).

POLYMATH will arrange all the objects selected such that the distance between the right sides of consecutive objects is always equal to the distance between the right sides of the first two objects (reference objects calculated according to their order when scrolling the page from the left). If the reference

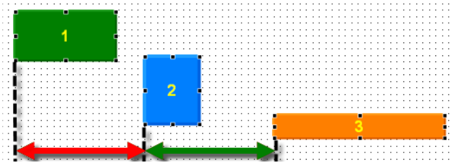
distance is less than zero, POLYMATH takes it automatically to 0, thereby aligning to the right.
In our example, the result obtained will be the one represented in the next figure:



To arrange to the center, just click on icon of the toolbar or the Main menu (Layout->Arrange->Center).
POLYMATH will arrange all the objects selected such that the distance between the central vertical axes of consecutive objects is always equal to the distance between the central vertical axes of the first two objects (reference objects calculated according to the order when scrolling the page from the left). If the reference distance is less than zero, POLYMATH takes it automatically to 0, thereby aligning to the center.
In our example, the result obtained will be the one represented in the next figure:

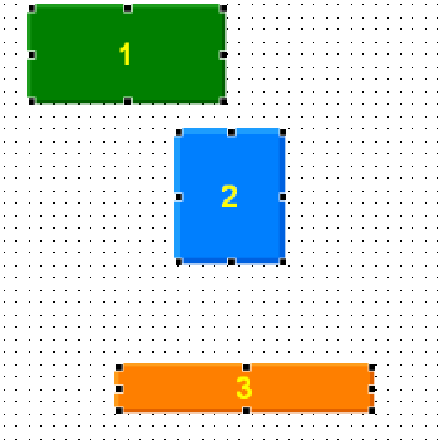


To operate a leftward arrangement, just click on the icon of the toolbar or Main menu (Layout->Arrange->Left).
POLYMATH will arrange all the objects selected such that the distance between the left sides of consecutive objects is always equal to the distance between the left sides of the first two objects (reference objects calculated according to their order when scrolling the page from the left). If the reference distance is less than zero, POLYMATH takes it automatically to 0, thereby aligning to the right.
In our example, the result obtained will be the one represented in the next figure:



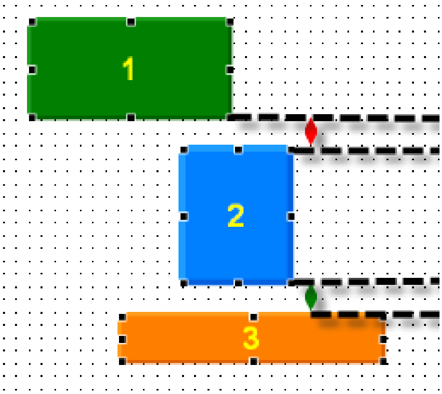
Vertical arrangement

Using our example, let us add three different colored buttons to the page as indicated in the next figure:

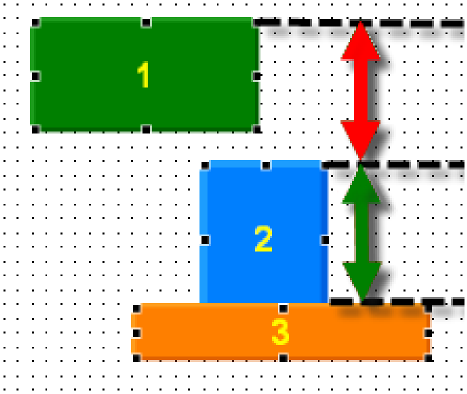


After drawing the three buttons, let us click on icon of the toolbar and draw a selection area that includes all the objects. This activates the arrangement options of the toolbar or Main menu: Layout->Arrange.
In the examples in this subsection, our starting point to illustrate how the arrangement operation works will always be this same initial situation. For vertical arrangements, POLYMATH takes as its reference the distance between buttons 1 and 2 (being the first two from the top).
To operate a simple vertical arrangement, just click on the icon of the toolbar or Main menu (Layout->Arrange->Vertically).
POLYMATH will arrange all the objects selected such that the distance between the top side of one object and the bottom side of the object preceding it is always equal to the distance between the bottom side of the first object and the top side of the second object (reference objects calculated according to

their order when scrolling the page from the top). If the reference distance is less than zero, POLYMATH takes it automatically to 0.
In our example, the result obtained will be the one represented in the next figure:

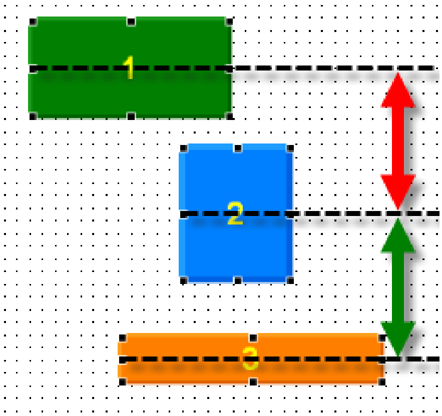


For a top-line arrangement, just click on icon ☰ of the toolbar or the Main menu (Layout->Arrange->Top).
POLYMATH will arrange all the objects selected such that the distance between the top sides of consecutive objects is always equal to the distance between the top sides of the first two objects (reference objects calculated according to their order when scrolling the page from the top).
In our example, the result obtained will be the one represented in the next figure:

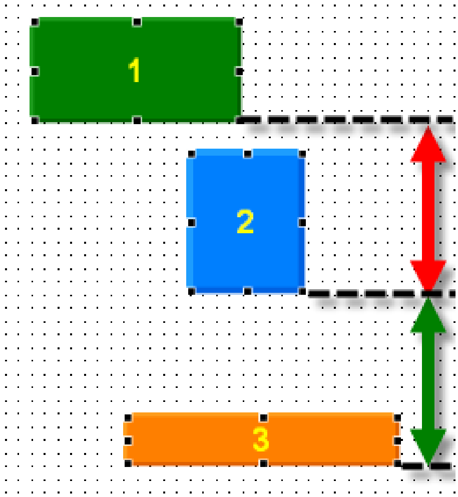


For a mid-point arrangement, just click on the icon ☰ of the toolbar or the Main menu (Layout->Arrange->Mid-point).

POLYMATH will arrange all the objects selected such that the distance between the central horizontal axes of consecutive objects is always equal to the distance between the central horizontal axes of consecutive objects (reference objects calculated according to their order when scrolling the page from the top). If the reference distance is less than zero, POLYMATH takes it automatically to 0 (thus making the alignment in the center).
In our example, the result obtained will be the one represented in the next figure:



For a bottom-line arrangement, just click on the icon of the toolbar or the Main menu (Layout->Arrange->Bottom).
POLYMATH will arrange all the objects selected such that the distance between the lowest sides of consecutive objects is always equal to the distance between the lowest sides of the first two objects (reference objects calculated according to their order when scrolling the page from the top). If the reference distance is less than zero, POLYMATH takes it automatically to 0 (thus making the alignment at the bottom).
In our example, the result obtained will be the one represented in the next figure:


1
2
3

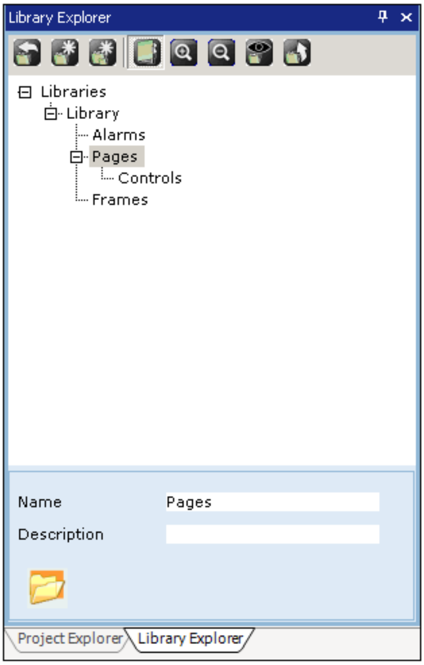
7. Other anchorable windows

In the last chapters we dealt with the workings of the three main anchorable windows: Project Explorer, Properties Editor and Events Editor.
But POLYMATH contains other anchorable windows, each of which has its particular purposes and functions as we shall now see: Library Explorer, Error Viewer and Complirer Output.

POLYMATH Libraries

POLYMATH has a structure saving tool that also functions outside the context of the project being edited: the Library.
This too is useful as it makes it possible to store, save and re-use portions of a project; each individual element or set of elements - indeed, even a whole project - can be put in a library to be easily re-usable in new projects. A classic example of the use of the Library is when you want to maintain a uniform style in different projects without having to redefine them each time. For example, you need simply create a frame with the colors, the size and the style required and save it in a Library to conserve it and make it available to be inserted in all the other projects.
With POLYMATH an unlimited number of libraries can be managed, each of which is then saved in a .vtl file, which is nothing other than a container of POLYMATH objects. The libraries are managed using the Explorer Library window which is an anchorable window and thus can be customized at will (see chap.3, “Anchorable windows” page 40). This chapter will describe how to interact with Library Explorer and the functions offered by the object, Library, and the two standard libraries supplied with POLYMATH.

Library Explorer tools



Name Pages

Description

Library Explorer is the window that shows the contents of the libraries being worked on and allows them to be managed. The upper part contains the function keys as described below:

- by clicking on the icon you access a window for opening files in which a .vtl file must be selected to download the relevant library;
- by clicking on the icon a new library is created in which the elements must be inserted manually;
- by clicking on the icon a new folder of objects is created in the area currently selected;
- in the Mode area the display mode for the chosen elements is selected: the options are “tree-form” (like Project Editor) or as folders of the contents of a library (Folder List). If the former is selected, a reduced-size preview of the object selected is shown at the bottom of the page, while in the latter case the objects will already be shown by their preview;
- if the display mode type is Folder List, you can specify the size of the folders by using buttons and .
- if the display mode type is Folder List, you can click on to determine whether or not to show the identifying

property of the object selected (name and description) at the bottom of the page;

- if the display mode type is Folder List, by clicking on the icon you can go on to display the directory above the current one.

A series of simple operations can be performed regarding the elements present in Library Explorer just by clicking on them with the right-hand key: Copy, Cut, Edit and Delete. By clicking on a Library folder, however, you can also operate Save and download library file (.vtl) commands.

Moving objects between Project and Library

An element can be moved from the project to the destination library by dragging the object from Project Explorer to within the Library. To move elements inside a page (that is, not part of Project Explorer like buttons, fields and controls) the drag must be performed after pressing the keys CTRL+SHIFT of the keyboard.

Every time an object is dragged to a Library, POLYMATH automatically also adds to the Library all the elements needed to save the element correctly for later use in new projects (images, fields, text labels, etc.).

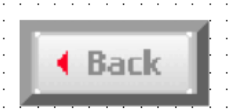
Warning: *If an element is dragged from the project to a library and this object refers to a variable (e.g. Numeric field), POLYMATH does not automatically import the variable into the Library, this operation must be done manually. When a variable is imported into a Library, its reference to the memory address is lost.*

Dragging is also the way an object moves from the library to a project: just take an element from the Library Explorer to the Project Explorer so that it is included in the project.

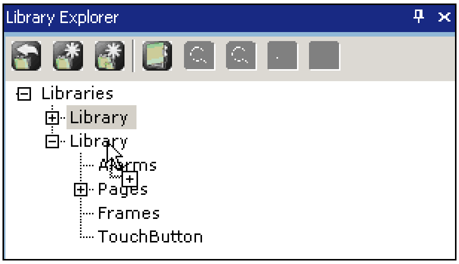
Warning: *When inserting an element from the Library to the Project pay special attention to the nomenclature of the objects. When the object inserted has the same name as an element already present in the project, POLYMATH will replace the element with the one present in the library*

Example of using the Library

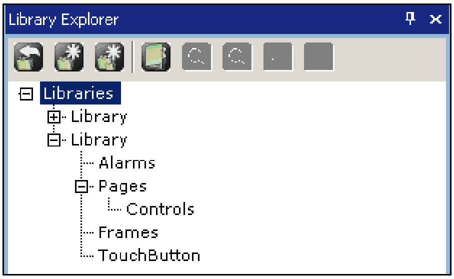
In this subsection we give a complete example to illustrate how convenient it is to use ESA-POLYMATH libraries.
We start with the assumption that we need to create various projects to be downloaded onto different terminals but want to keep the same overall look and functional approach throughout. We begin by defining a touch button (see chap.6, “Touch Button” page 243) with a customized style to suit our needs.



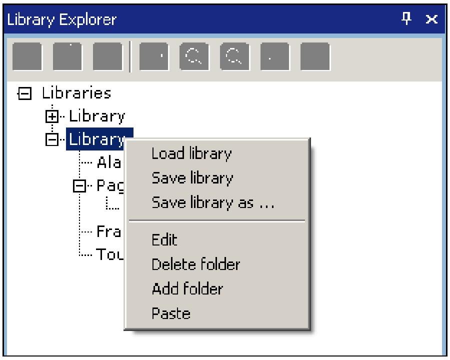
We have given the button a border with the value 7, dark grey as the color and applied a background image to it; in addition, we assign to the event of pressing the button the function of displaying the previous page. To avid having to redefine these operations every time this button appears in the following projects we now put this button into our library.
We move to Library Explorer and create a new library by using the icon ; then we drag the button to within the library. First we press CTRL+SHIFT on the keyboard and, using the mouse, select the button (as it is an element within the page not present in Project Explorer) to make it dragable to the Library as shown in the figure :



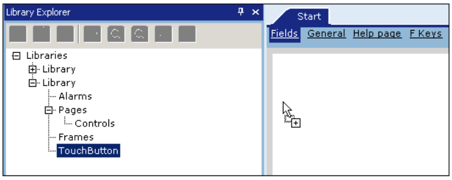
At this point, when we release the mouse key, the touch button we have created becomes part of the Library list as an offspring of the element selected (or enters the current folder if our display mode is List).



As we can see from the previous image, we have added both the button and, as offspring, the image we have applied to the Library, so the button is already available for use in new projects.
It is advisable to save the library to preserve our work so we click with the right-hand key on the Library element and choose Save As (then choosing a name and a path for the library).



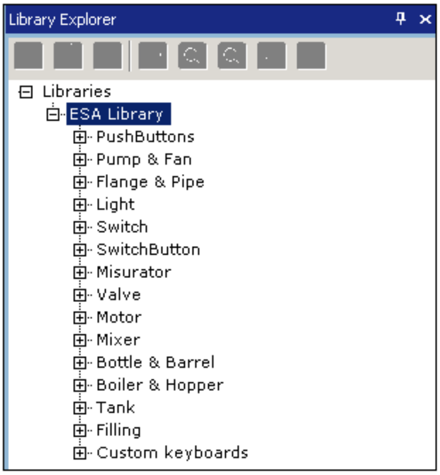
We now create a project from scratch and perform the same operation backwards, that is, we apply the button created in the previous project inside the new one.



In the Library we select the button from the list and drag it into a new page: the button has the same graphic characteristics as the one created before and preserves the function assigned to the keystroke event. This operation can be performed for all the projects for which we want to maintain this style and this type of button.
As in the example just given, we can save more complex work relating to a frame, a page or highly complex controls, too.

Standard library in POLYMATH

When POLYMATH is installed on a PC, a library of standard element to be inserted into projects as desired is also installed. POLYMATH's predefined library (Esa Library.vtl file) is found in the main directory in which the program was installed, generally (if the standard path has not been modified) C:\Program Files\Esa Elettronica\ESAPOLYMATH\Esa Library.vtl


Library Explorer

- Libraries
  - ESA Library
    - PushButtons
    - Pump & Fan
    - Flange & Pipe
    - Light
    - Switch
    - SwitchButton
    - Misurator
    - Valve
    - Motor
    - Mixer
    - Bottle & Barrel
    - Boiler & Hopper
    - Tank
    - Filling
    - Custom keyboards

The library is composed of the following categories of elements:

- Push Buttons
- Pump & Fan
- Flange & Pipe
- Light
- Switch
- Switch Button
- Measurer
- Valve

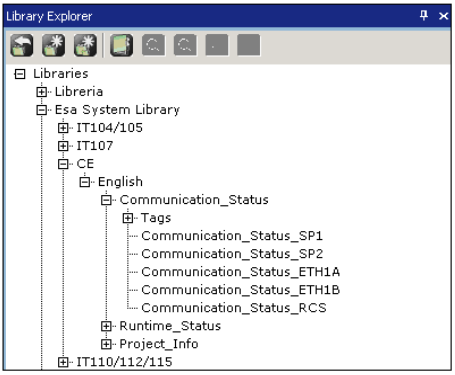
- Motor
- Mixer
- Bottle & Barrel
- Boiler & Hopper
- Tank
- Filling
- “Custom keyboards”

Each category is a folder containing a certain quantity of objects that can be inserted into a project. To insert one of these objects just drag it into the destination page. Once it has been added to the project, each object has its own Properties Editor containing a series of properties corresponding to the function the object will perform. In general these properties and the method of editing such objects in POLYMATH are the same as those for monostable and bistable touch buttons.

Warning: *When an object from the ESA library is added to the project, POLYMATH automatically introduces the images needed to make the objection clearly displayed in runtime. If the object is eliminated from the project, it is important to bear in mind that the images will remain and it will be necessary to eliminate them manually when they are no longer used (see chap.5, “Frames” page 128).*

System Library present in POLYMATH

When POLYMATH is installed, a system library is also installed. This system library (Esa System Library file .vtl) may be found in the main directory where the program is installed, generally (if the standard path has not been modified) C:\Program Files\Esa Elettronica\ESAPOLYMATH\Esa System Library.vtl.

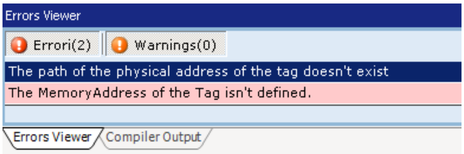


This library writes precompiled pop-up pages containing the system variables in each of the principal languages (English, Italian, Portuguese, Spanish, German, French, Chinese and Japanese) (see chap.13, “Appendix A - System Variables” page 555). Each of these pages just needs to be dragged into a project for them to be available in it. The pages that are available differ in five ways:

- Project information
- Status of runtime
- Status of the ASP communication port
- Status of the CAN communication port
- Status of the DP communication port

Errors Viewer

"Error Viewer" is an anchorable window (see , " Le Finestre ancorabili" (Anchorable Windows on page 47" chapter) that supplies information regarding the errors present in the project.



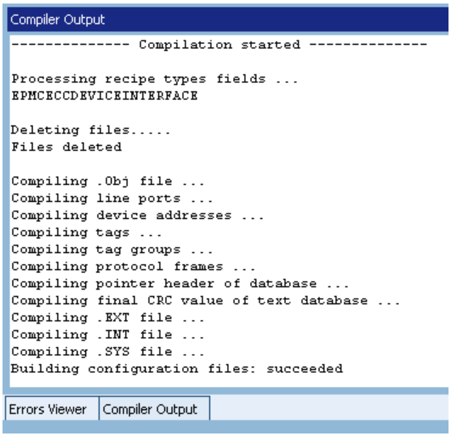
The path of the physical address of the tag doesn't exist

The MemoryAddress of the Tag isn't defined.

The "Error Viewer" mask has a real time report of the errors and warnings relative to the project that is being edited validated. The errors appear in red and the warnings in orange. By double clicking on the description of the problem, POLYMATH will take the focus of the application onto the origin of the error, i.e. onto the ("Editor Proprietà" (Properties Editor), mask in the work area, etc.) mask, from where the correction can be made. The errors disappear as soon as their correction has been supplied in the relative area.

Compiler Output

"Compiler Output" is an anchorable window where the log information relative to the last compilation (if already carried out during the actual session) of the project currently being edited is reported. During the compilation phase, the log in this window is updated in real time, showing the object and the compiled file on which it is working. The errors and warnings are signalled as in the "Error Viewer" mask and a double click on them causes movement of the focus onto the area of application from where it is possible to solve the problem.



```
-------------- Compilation started --------------

Processing recipe types fields ...
EPMCECCDEVICEINTERFACE

Deleting files.....
Files deleted

Compiling .Obj file ...
Compiling line ports ...
Compiling device addresses ...
Compiling tags ...
Compiling tag groups ...
Compiling protocol frames ...
Compiling pointer header of database ...
Compiling final CRC value of text database ...
Compiling .EXT file ...
Compiling .INT file ...
Compiling .SYS file ...
Building configuration files: succeeded
```

8. Compiling, Downloading and Runtime

The preceding sections have provided all the necessary concepts for creating and editing a project by describing all the utilities offered by POLYMATH. Once the editing phase is over, the work done needs to be downloaded onto the ESA panel.

First of all it is necessary to check that there are no problems in the project that might prevent it behaving properly in runtime. To detect any errors there needs to be a validation operation which analyzes all the objects created and checks that the properties are complete and coherent without, however, creating any transfer files.

Transfer files are created, though, when compiling, which, therefore, is the more complex operation. Once the compiled files have been created, they can be downloaded onto the panel using the appropriate POLYMATH function.

This chapter will supply the details of the operations of validation, compilation and download, illustrating at the end another very useful function of POLYMATH, the download of the image of the operating system onto a Windows® CE terminal.

Validation

Validation is the operation that checks the coherence of the objects added to the project. Any errors or warnings are shown in the Errors Viewer window (see chap. 7, "Errors Viewer" page 348).

POLYMATH offers wither a global validation of the project or the validation of only the object currently being edited: a project can be globally validated by clicking on the ✓ icon of the Toolbar (or main menu, using File->Validates Project) while partial validation requires a click on ✓ (File->Validates Current).

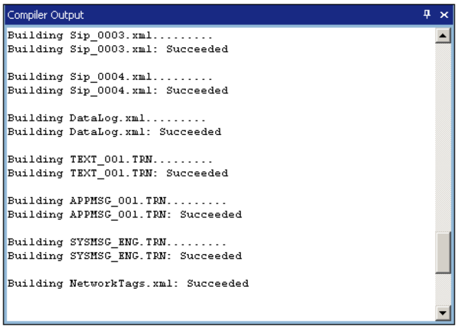
There is also a choice as to whether to let POLYMATH perform a validation in real-time (signals problems as they are edited) or whether validation should be carried out only when requested by the appropriate commands. This function can be configured using the main menu by clicking on Tools->Options (see chap. 3, "Menu: Tools" page 31).

The Errors mask contains a report in real time of the errors and warnings relating to the project being validated. The errors appear in red while the warnings appear in orange. If we double-click on the description of the problem, POLYMATH will focus the application on the origin of the error, that is, on the mask (Properties Editor, work area mask, etc.) where a correction can be made. As soon as they are corrected in the appropriate area, the errors disappear.

Compilation

Compilation is the operation whereby a project created with POLYMATH is transformed into files ready to be sent to the panel to be then interpreted by the VT's firmware.
To start off the compilation of a project click on the icon in the Toolbar (or use the main menu File->Compile). Any errors or warnings detected in the process of compilation are signalled in the Errors Viewer window (see chap. 7, "Errors Viewer" page 348). Errors appear in red while Warnings appear in orange.

Warning: *It is always advisable to correct errors (in red) signalled by the compiler before downloading the project onto the panel, as failure to do so could cause runtime malfunctioning. By contrast, the warnings relate to incomplete parts of the project that it would be advisable to correct although their runtime impact is less grave.*



```
Building Sip_0003.xml.........
Building Sip_0003.xml: Succeeded

Building Sip_0004.xml.........
Building Sip_0004.xml: Succeeded

Building DataLog.xml.........
Building DataLog.xml: Succeeded

Building TEXT_001.TRN.........
Building TEXT_001.TRN: Succeeded

Building APPMSG_001.TRN.........
Building APPMSG_001.TRN: Succeeded

Building SYSMSG_ENG.TRN.........
Building SYSMSG_ENG.TRN: Succeeded

Building NetworkTags.xml: Succeeded
```

When the compilation has finished the project is ready to be downloaded onto the panel. When there is an attempt to download a project that has not been compiled (or that contains changes compared with the last compilation), POLYMATH will advise the user and ask whether to go ahead with the compilation again before beginning to transfer.

Project simulation

Run time simulator

The project can be simulated directly on the PC without being transferred to the terminal; all device variables and the project's accurate execution can be verified without the device actually being connected.

List of menu items

File

Menu Path	Function
File -> Import	Imports .csv files of following elements: • Watch list list of variables selected by check mark • Tag values value of variables • Simulations type of simulation associated with single variables
File -> Export	Exports .csv files of following elements: • Watch list list of variables selected by check mark • Tag values value of variables • Simulations type of simulation associated with single
File -> Print	Prints

Tags

Menu path	Function description
Tags -> Edit value	Edits the value of the selected variable
Tags -> Reset all values	Resets the values of all variables
Tags -> Add to watch list	Adds the selected variable to the "Watch List"
Tags -> Remove from watch list	Removes the selected variable from the "Watch List"
Tags -> Reset watch list	Removes all variables from the "Watch List"
Tags ->Show watch list	Shows list of variables included in the "Watch list"
Tags -> Show complete list	Shows list of all variables included in the project

Simulation

Simulation -> Play	Carries out all simulations
Simulation -> Pause	Pauses the simulations
Simulation -> Stop	Stops the simulations
Simulation -> Add new simulation	Adds a new simulation
Simulation -> Edit simulation	Edits the selected simulation
Simulation -> Remove simulation	Removes the selected simulation
Simulation -> Enable simulation	Enables the selected simulation

Simulation -> Disable simulation	Disables the selected simulation
Simulation -> Remove all simulation	Removes all simulations
Simulation -> Enable all simulation	Enables all simulations
Simulation -> Disable all simulation	Disables all simulations

List of buttons in the tags

"Device simulation" tag
List of configured simulations

- Add: adds a simulation to a selected variable
- Edit: edits the selected simulation
- Remove: removes the selected simulation

"Project Tags" Tag
List of all the variables in the project

- Edit Value: Edits the value of the selected variable.
- Watch List: shows only the variables in the Watch List, selected by check mark.
- Reset List: Reset List: unchecks all variables, removing them from the watch list.

!!! WARNING !!! When quitting the simulator, all the values of the variables, the watch list and the simulations are lost. To save them for further use, use the file menu to export Watch lists, Tag values and Simulations.

Downloading a project

Preparing Windows® CE panels for the first download

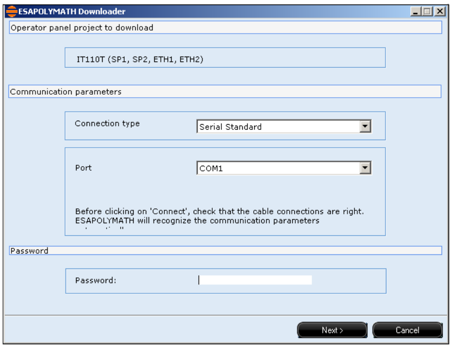
To ensure that projects created with POLYMATH are correctly run on Windows® CE panels, 2 files to be found on the installation CD need to be copied onto the memory (Hard Disk) of the panel:

- Esa.cfg - this file is different for every model of panel in that it contains hardware information and goes under the root directory, Hard Disk\ .
- Startup.esa - this file must be copied into the directory Hard Disk\Esa\Startup. It allows the ESA Downloader application to start and therefore enables communication to take place in the project download phase. In addition, the addition of this file leads to the project being started automatically when the panel is switched on.

After copying these two files the terminal’s operating system starts; after successive start-ups the ESA Downloader application will start automatically without any additional operation being necessary.

Downloading a project

When a project is compiled it is ready to be transferred to the terminal by invoking the Download function (if Esa Downloader has been properly configured in accordance with the indications in the preceding subsection). To start the transfer just click on the icon of the toolbar (or use main menu File->Download). If there are files compiled for the version of the project currently being edited, POLYMATH will show the window relating to the hardware configuration of the PC-terminal connection. If, on the other hand, no files have been compiled for the project yet, POLYMATH will ask the programmer whether it should start compiling.



Operator panel project to download

IT110T (SP1, SP2, ETH1, ETH2)

Communication parameters

Connection type: Serial Standard

Port: COM1

Before clicking on 'Connect', check that the cable connections are right.
ESAPOLYMATH will recognize the communication parameters

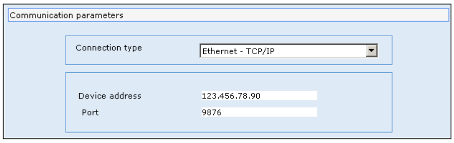
Password

Password:

The download window allows the operator first to select the terminal to which the project file is sent and the parameters for the type of connection to be used. The types of connection catered for are :

- "Standard serial"
- "Ethernet - TCP/IP"
- "Local"
- "USB"
- "http"

The serial connection is the most common type and it is achieved by connecting the ports of the PC on which POLYMATH has been installed and those of the terminal with the appropriate cable (see chap. 5, "Communication ports" page 73). The type of port to be used for the PC-terminal connection must be specified.



Communication parameters

Connection type: Ethernet - TCP/IP

Device address: 123.456.78.90

Port: 9876

If an Ethernet TCP/IP connection is required, it is necessary to specify the parameters for making the connection: the IP address and communication port (see chap. 8, "Establishing an Ethernet connection" page 368). A remote connection (out of those set on the PC being used) can be used when the project download starts; in this case user credentials for authenticating access rights (username and password) also need to be given.

If the connection is Local (that is, the files are sent to a server present on the same PC), just define the port through which POLYMATH and the application will communicate.

Once the type of connection has been selected, just click on 'Connect' to activate the connection (which can be aborted by clicking on 'Cancel'.

Preparation of IT panels before download

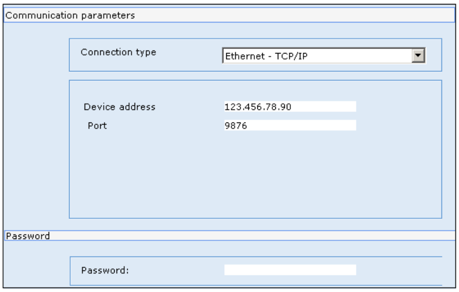
The IT panels do not require any particular preparation if choosing a USB connection. If an ethernet connection is chosen, the panel must be connected to the network and the network parameters must be configured as indicated in the hardware manual “Video terminal ITxxx/Control Panel/ Network”. Finally, configure the connection gate on the "Service page/configuration download" page of the terminal (See hardware manual “Video terminal ITxxx/Downloader Configuration”)

Perform the project Download on the IT terminal

When a project is compiled it can be transferred on to the terminal by means of the Download function; to start transfer, click on the icon of the instruments bar (or on the main menu File->Download). If there are compiled files for the current project version edited, POLYMATH will display a window relating to the configuration of the hardware connection machine-terminal. If however, no files are compiled with respect the project, POLYMATH will ask the programmer to perform compilation.
In the window relating to download, it is possible to select the terminal to send the project files and the relative parameters to the type of connection to use; types foreseen for the IT terminal are:

- “Standard Serial”
- “Ethernet - TCP/IP”
- “Local”
- “USB”
- “http”

The USB connection is carried out by connecting, using a suitable cable, the gates of the machine with POLYMATH installed and those of the terminal (see chap. 5, “Communication ports” page 73).



Communication parameters

Connection type: Ethernet - TCP/IP

Device address: 123.456.78.90
Port: 9876

Password

Password:

If an Ethernet TCP/IP connection is chosen, specify the parameters in order to carry out the connection as IP address and communication gate that must be configured before the terminal (see chap. 8, “Preparation of IT panels before download” page 357).

Once the connection has been selected, click on 'Connect' to start the connection whilst clicking on 'Cancel', cancels the operation.

Preparation of the PCs or PC terminals based on the first download

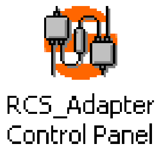
If an ethernet connection is chosen, connect the PC to the network and configure the network parameters (refer to the windows guide or consult the network administrator).

PCUSB ADAPTER

If a SERIAL CAN or PROFIBUS connection is chosen, the ESA "PCUSBxxxxxxx" product must be used.
The "PCUSBxxxxxxx" product is essential to establish communication between PC/XS and the PLC provided with a CAN/DP or SERIAL port.
ESA puts the following order codes at disposal :

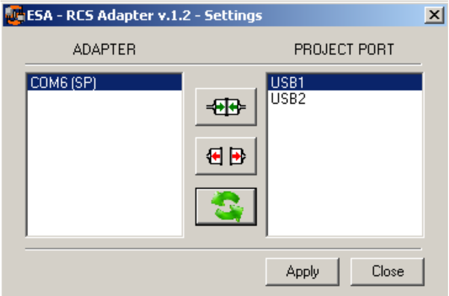
PCUSBADP0SP2	(RS232/485 serial communication board)
PCUSBADP0CAN	(CAN-BUS communication board)
PCUSBADP0DP	(PROFIBUS-DP communication board)

In order to perform a project transfer, it is necessary to install on the PC where the runtime is located, a program called ESAPOLYMATH Downloader available on installation CD of the POLYMATH. To install the downloader, follow the simple instructions provided by the installation guide. At the end of installation, the program asks if it must always be started with the start of Windows. If the answer is “no” it must be manually activated each time a transfer is to be carried out or runtime launched.
After having installed the program, the icon is added automatically inside "Control panel"
“RCS_ADAPTER CONTROL PANEL” :



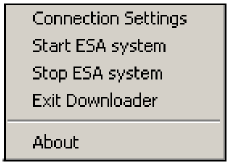
During installation of the "ESAPOLYMATH Downloader" application, the drivers necessary for connecting the "PCUSB" are installed as well.
The drivers are requested the first time the "PCUSBxxxxxxx" is connected to the PC/XS and are found skimming through the path :
C\PROGRAM FILES\ESAELETTRONICA SPA\ESAPOLYMATH DOWNLOADER\DRIVER RCS_ADAPTER.

Once the driver path requested by the application is inserted, we connect the USB port used in our project (for example USB1) with the COM to which the "PCUSB" is associated.
This connection is carried out double-clicking the "RCS_ADAPTER CONTROL PANEL" icon previously described.
The following image will appear :



At this point, click "APPLY" to save the settings.
If there are problems during the connection, temporarily deactivate firewall and the antivirus installed on the PC/XS, if there is one.

Clicking the icon near the system clock in the traybar with the right key of the mouse, one accesses the program functions :



- “Connection Setting”: allows you to configure the connection parameters by selecting between the serial or Ethernet port (port 4096), If the serial transfer is chosen, a CVCOM41102 cable must be used. If the direct ETHERNET transfer is chosen, a CVNET11002 cable (crossed type) must be used. If passing through a HUB or a SWITCH, a standard network cable must be used

- Start ESA system: start runtime (the project starts after it has been transferred onto XS/PC)
- Stop ESA system: stop runtime
- Exit Downloader: close downloader
- About: shows information of the downloader versions

PCMACHINEBASE

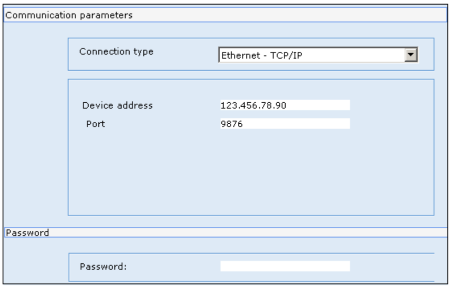
The whole system regarding Runtime on the PC/XS working with "ESAPOLYMATH DOWNLOADER" application described until now, will automatically close after 20 minutes if the USB Hardware key has not been inserted.
Inserting the "PCMACHINEBASE" key in the XS/PC terminal, the system could request to insert the drivers to acknowledge the key. These drivers are found skimming through the following path :
C\PROGRAM FILES\EUTRONSEC\SMARTKEY DRIVERS

Carry out project Download on the PC or PC terminal based

When a project is compiled it can be transferred on to the terminal by means of the Download function; to start transfer, click on the icon 📥 of the instruments bar (or on the main menu File->Download). If there are compiled files for the current project version edited, POLYMATH will display a window relating to the configuration of the hardware connection machine-terminal. If however, no files are compiled with respect the project, POLYMATH will ask the programmer to perform compilation.
In the window relating to download, it is possible to select the relative parameter connection types to be used; the envisioned connection types for the PC platforms are:

- “Standard Serial”
- “Ethernet - TCP/IP”
- “Local”
- “USB”
- “http”

The serial connection is carried out by connecting, using a suitable cable, the gates of the machine with POLYMATH installed and those of the terminal/PC (see chap. 5, “Communication ports” page 73).



Communication parameters

Connection type: Ethernet - TCP/IP

Device address: 123.456.78.90

Port: 9876

Password

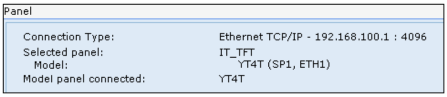
Password:

If an Ethernet TCP/IP connection is chosen, specify the parameters in order to carry out the connection as IP address and communication gate that must be configured before the PC (see chap. 8, “Preparation of the PCs or PC terminals based on the first download” page 359).

Once the connection has been selected, click on 'Connect' to start the connection whilst clicking on 'Cancel', cancels the operation.

Transferring data

After setting all the connection variables and activated the connection, POLYMATH checks the status of the terminal. In particular, there is a check of the space available in the terminal’s memory (relative to the needs of the current project) and status of the project’s components.



Panel

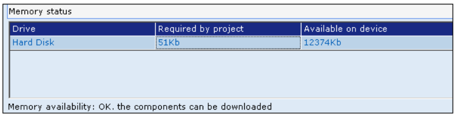
Connection Type: Ethernet TCP/IP - 192.168.100.1 : 4096

Selected panel: IT_TFT

Model: YT4T (SP1, ETH1)

Model panel connected: YT4T

The upper part of the download mask is used to indicate whether the project being sent is consistent with the type of terminal being used to receive the transfer. If it is not, the mask shows an error message.

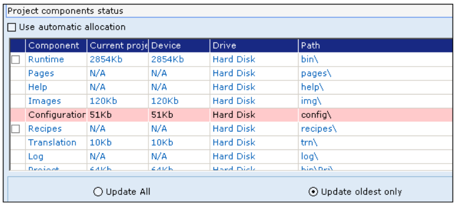


Memory status

Drive	Required by project	Available on device
Hard Disk	51Kb	12374Kb

Memory availability: OK. the components can be downloaded

Following this the details regarding the memory required for the project and that available on the terminal's supports is shown: the operator can see if there is enough space on the panel to hold the project files and, if there are problems, an error message is shown.

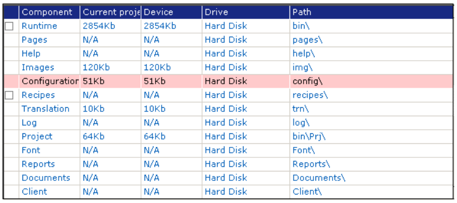


Project components status

□ Use automatic allocation

	Component	Current proje	Device	Drive	Path
□	Runtime	2854Kb	2854Kb	Hard Disk	bin\
	Pages	N/A	N/A	Hard Disk	pages\
	Help	N/A	N/A	Hard Disk	help\
	Images	120Kb	120Kb	Hard Disk	img\
	Configuratior	51Kb	51Kb	Hard Disk	config\
□	Recipes	N/A	N/A	Hard Disk	recipes\
	Translation	10Kb	10Kb	Hard Disk	trn\
	Log	N/A	N/A	Hard Disk	log\
	Project	64Kb	64Kb	Hard Disk	bin\Prj\

Finally, in the lower part there is a list of project components, the more recent of which (compared with those residing in the terminal) are highlighted in pink. The support and the path used for saving the files of the related section can be shown or the operator can decide to let POLYMATH automatically allocate the component on the physical supports available on the panel.
At this point, all the elements on the panel (firmware, project and all the other components) can be updated or, to save time, only those elements requiring updating because the currently used version is more recent than that of the elements in the terminal.

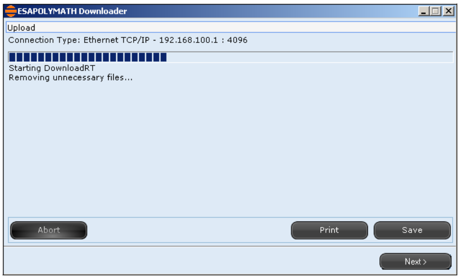


	Component	Current proje	Device	Drive	Path
□	Runtime	2854Kb	2854Kb	Hard Disk	bin\
	Pages	N/A	N/A	Hard Disk	pages\
	Help	N/A	N/A	Hard Disk	help\
	Images	120Kb	120Kb	Hard Disk	img\
	Configuratior	51Kb	51Kb	Hard Disk	config\
□	Recipes	N/A	N/A	Hard Disk	recipes\
	Translation	10Kb	10Kb	Hard Disk	trn\
	Log	N/A	N/A	Hard Disk	log\
	Project	64Kb	64Kb	Hard Disk	bin\Prj\
	Font	N/A	N/A	Hard Disk	Font\
	Reports	N/A	N/A	Hard Disk	Reports\
	Documents	N/A	N/A	Hard Disk	Documents\
	Client	N/A	N/A	Hard Disk	Client\

- Runtime: transfer of firmware files in the currently used version of POLYMATH.
- Pages: files containing information about the pages created in the project
- Help: files containing information about the help pages created in the project
- Images: the project images are simply copied into this folder
- Configuration: files containing information useful for running the project properly (.xml component files).

The Scripts added by the user and the password files can be found here, too.

- Recipes: files (.rec) containing information on the recipes saved in the memory of the VT
- Translation: files containing translations of multilingual project texts and system messages
- Log: log files used by the application; this folder, for example, contains the log files of the login/logout operations, the alarm history and trend buffer logs.
- Font: files containing relative information to fonts used in the project
- Report: files containing the relative information to the project reports
- Documents: empty directory ready to accommodate the reports in pdf
- Project: OPC files useful for managing the project’s communications
- VTWinPro: contains general project information (.xml files)
- Font: font files installed and used by the project



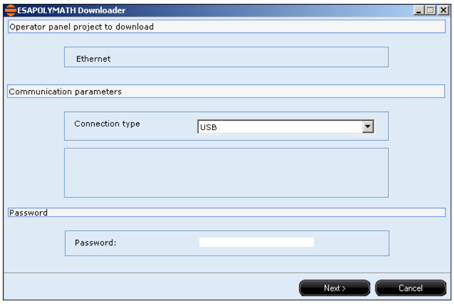
Connection Type: Ethernet TCP/IP - 192.168.100.1 : 4096

```
Starting DownloadRT
Removing unnecessary files...
```

After starting the download of the files (whether this be a partial or total update) a window for logging the transfer operations that POLYMATH is running will appear: what you see is the names of the files currently being transferred and those already transferred. During the transfer to the panel a message appears indicating the status of the download; as soon as this is finished the project starts being executed.

Change Password

Polymath provides the possibility to set a password on the panel, necessary then (if configured), to transfer the project. In order to configure the Main Menu, click on Instruments->Utility downloader->Change Password Downloader CE.

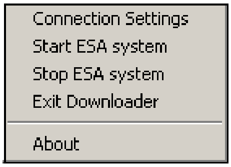


Connection type USB

At this point, the type of connection with which to communicate to the panel will be requested and once chosen the password can be changed.
To remove the password carry out again the procedure listed above leaving the fields empty.

Execution runtime on XP

Once download is complete, runtime is automatically carried out.
By means of the "ESAPOLYMATH Downloader" functions, runtime can be stopped and restarted:



With regards runtime of Polymath on windows XP, a USB pen drive is required that acts as a license without which runtime will not execute. In order to create a project for the PC or any other platform terminal PC based, no hardware drive is required. The advanced polymath version with license is sufficient. The same project can be used on a limited number of PCs as long as each PC has a hardware drive.

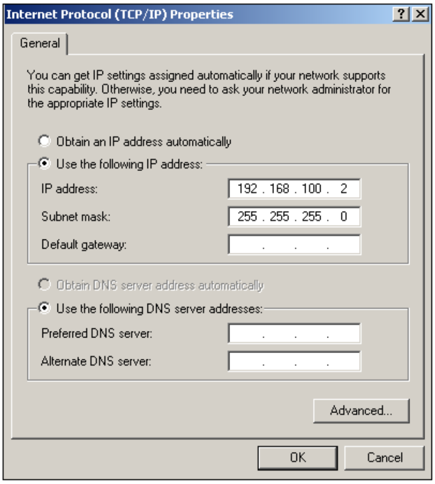
__Attention:__ if the drive is withdrawn from the USB gate in which it is inserted, a message will appear on the terminal: " The USB must be connected while Esaruntime is running retry key check or close runtime". At this point, re insert the drive in the gate and click on the retry runtime to execute or cancel to close.

Download the IT OPERATING SYSTEM image

In POLYMATH, it is possible to transfer the whole image of the Windows Operating System® CE on the terminal. This operation is reachable from the main menu clicking on Instruments->Utility downloader->Update Boot Windows CE for IT.
It is necessary to set the connection mode as in the case of project download (see chap. 8, "Transferring data" page 362). The loaded images on the panel will overwrite the existing one for which backup should be carried out before performing this operation.

Set up an Ethernet connection

To update the Boot between the PC and the IT panel, a crossed ethernet cable is necessary (only possible method), successively, it is necessary to set on the PC a local network with the IT panel, setting as IP address : 192.168.100.2 and as Subnet Mask : 255.255.255.0



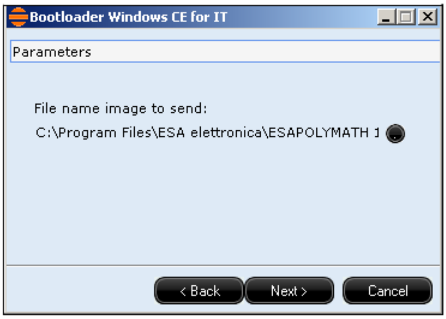
You can get IP settings assigned automatically if your network supports this capability. Otherwise, you need to ask your network administrator for the appropriate IP settings.

○ Obtain an IP address automatically
● Use the following IP address:
IP address: 192 . 168 . 100 . 2
Subnet mask: 255 . 255 . 255 . 0
Default gateway: . . .

○ Obtain DNS server address automatically
● Use the following DNS server addresses:
Preferred DNS server: . . .
Alternate DNS server: . . .

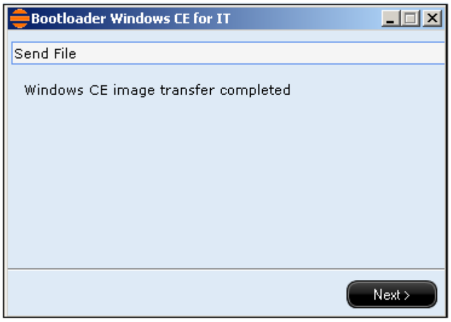
From the Main Menu, click on Instruments->Utility downloader->Update Boot Windows CE for IT to start the procedure and follow the instructions.

POLYMATH will ask for the previously explained IP address to be set and to select the file source from which the program will read the Boot to be transferred.
During the installation of POLYMATH, a file is created on the PC where the images for the operating systems of various ESA panel models are copied ready to be downloaded on to the terminal; generally the image files are found in the main directory of the POLYMATH in the path \xml\OSImages\IT1xx (for example for a IT105T, if the installation path has not be modified, the image is in C:\Program Files\ESA Elettronica\ESAPOLYMATH\xml\OSImages\IT105 TFT\NK.bin).


Parameters

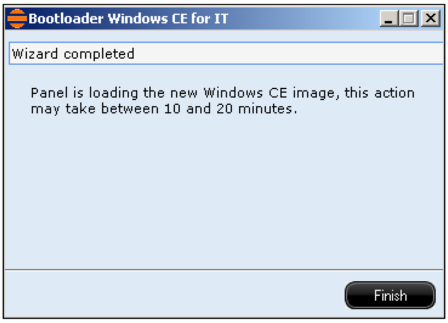
File name image to send:
C:\Program Files\ESA elettronica\ESAPOLYMATH 1

After selecting the source from the Boot File to transfer, press the NEXT button and switch on the IT panel so that transfer can begin.


Send File

Windows CE image transfer completed

Once transfer of the file is terminated, the IT panel should be left on until the initial page is displayed.



Bootloader Windows CE for IT

Wizard completed

Panel is loading the new Windows CE image, this action may take between 10 and 20 minutes.

At this point, if the image is different to that already installed, connect a Mause USB because the panel will lose calibration. Two error Pop ups will display, click ok.
Entering in CONTROL PANEL->STYLUS, it will be necessary to calibrate the touch screen again.

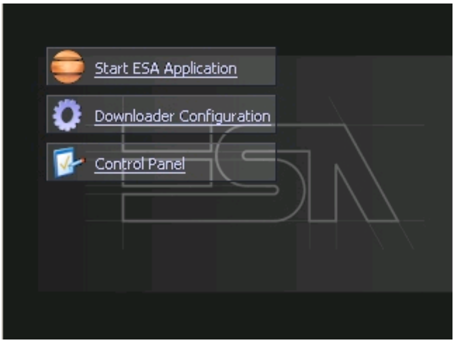
Downloading the image of the Operating System for VT CE

POLYMATH offers the possibility of transferring the entire Windows® CE Operating System image to the terminal. This can be done using the main menu by clicking on Tools->Update OS Image OS in panel.
The type of connection has to be defined as in the case of the project download (see chap. 8, “Transferring data” page 362).
The image loaded onto the panel overwrites the existing one (which should be backed up before running this operation).
While installing POLYMATH, the images of the operating systems for the various models of ESA panels are copied onto the PC ready to be downloaded onto the terminal. Generally the image files are in the main directory of POLYMATH in the path \xml\OSImages\VTxxx. For example, if the installation path has not been changed, the image for VT595 will be in C:\Program Files\ESA Elettronica\ESAPOLYMATH\xml\OSImages\VT595\NK800.bin.

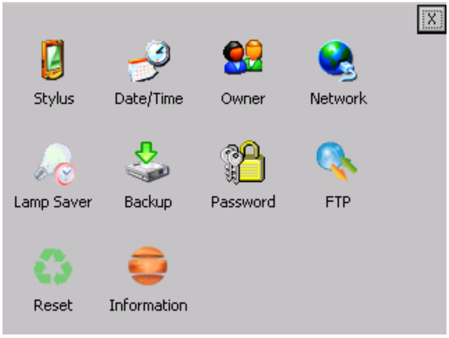
Establishing an Ethernet connection

ESA panels with Windows® CE operating system allow a connection to be made to an Ethernet network by means of just a few simple steps. After connecting the terminal to the network using the appropriate network cable, just define the connection as set out below:

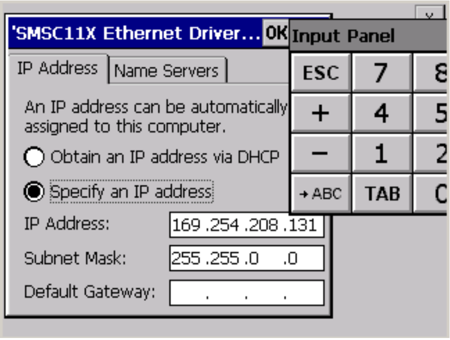
- From the initial page of the terminal, click on "Control Panel" :



- From the "Control Panel, click on the "Network" icon :



- Use the next window to insert the details by which the panel is to be recognised within the network.
- Select the "Specify an IP address" option and insert an IP address and a Subnet Mask address. These parameters must be used in order to interact with the terminal inside of the network (for example, Downloading a project via Ethernet, see chapter, "Download the project on the VTCE panels" on page 290") :


'SMSC11X Ethernet Driver... OK

IP Address | Name Servers

An IP address can be automatically assigned to this computer.

○ Obtain an IP address via DHCP

● Specify an IP address

IP Address: 169 .254 .208 .131

Subnet Mask: 255 .255 .0 .0

Default Gateway: . . .

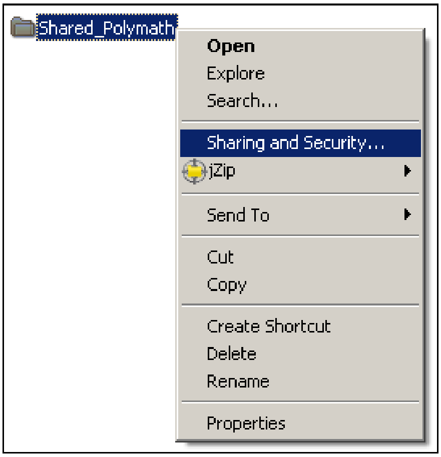
Note: *To check that the panel has been correctly set in the Ethernet network, the operator is advised to perform a "Ping" operation using a different terminal in the network. For example, using a Windows PC, click on Start->Run->and write 'ping ***.***.***.***' replacing the asterisks with the IP address assigned to the panel. A command window for checking the actual connection and its speed will appear.*

Naturally, to have a PC interact with the panel via Ethernet the PC also needs to be configured to access the network with its own IP address (the configuration is identical to that seen for the terminal).

Sharing folders between panel and PC

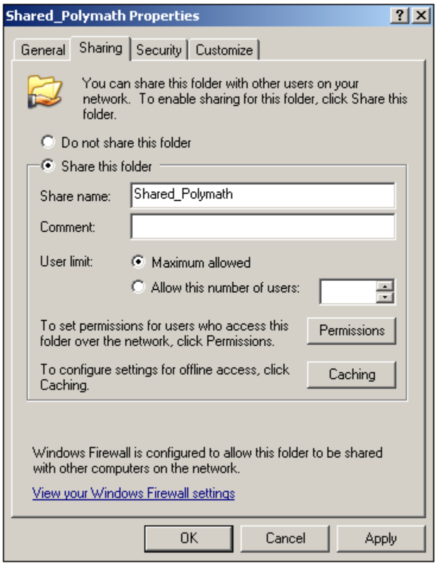
It can sometimes prove useful to share folders in a network to make them accessible to a Windows® CE panel in the same network (after having first carried out the configuration indicated in the preceding section).
In this section we shall give an example of how to access a PC folder with Windows® XP using a POLYMATH project.

- First of all create a new folder on your PC's hard disk (e.g. C:\); we will rename this folder "Shared_Polymath" and then select it by clicking with the right-hand mouse key as indicated below:



- Now select the option „Sharing and Protection" from the resulting menu. This takes you to the window for the sharing settings.

- Now move to the "Sharing" tab as shown in the figure below:



Shared_Polymath Properties

You can share this folder with other users on your network. To enable sharing for this folder, click Share this folder.

○ Do not share this folder

● Share this folder

Share name: Shared_Polymath

Comment:

User limit: ● Maximum allowed
○ Allow this number of users:

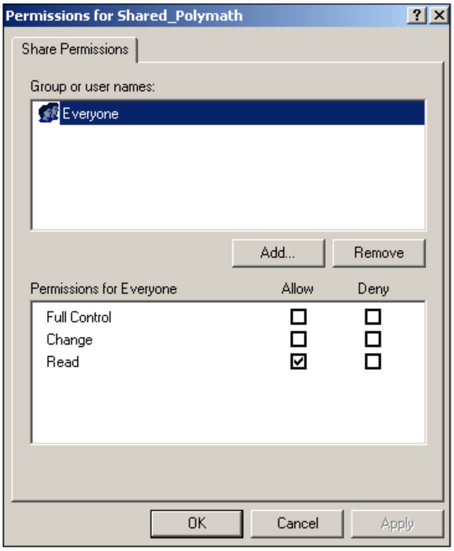
To set permissions for users who access this folder over the network, click Permissions.

To configure settings for offline access, click Caching.

Windows Firewall is configured to allow this folder to be shared with other computers on the network.

View your Windows Firewall settings

- We select the option "Share this folder"; we then leave the sharing name unchanged (leaving the default one corresponding to the name we chose for the folder, in our case "Shared_Polymath").
- Finally we click on the "Authorization" button to define which users can have access to the folder (for details regarding network users, consult your network administrator) and which actions can be performed:



Group or user names:
Everyone

Permissions for Everyone	Allow	Deny
Full Control	□	□
Change	□	□
Read	☑	□

- Using the lower part of the window, we select all 3 options available. In this way outside users can read and write the files contained in this folder.
- At this point we click on 'Apply' and 'Ok' in this window and then on 'Apply' and 'Ok' in the window for assigning the properties of the folder.

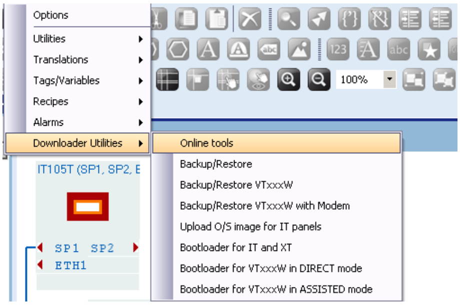
After making these settings, the folder C:\Shared_Polymath will be accessible from any Windows® CE panel connected to the same network. In particular, the folder can be reached by the panel by digit ting the following path :
\\NOMEPC\c$\Shared_Polymath
where NOMEPC indicates the ID name of one's personal computer within the network (this name is given in the System Properties of the PC under the option "Name of Computer" or it must be requested from the network administrator). The code c$ indicates the drive on which the shared folder can be found.
A typical example of this function is when exporting recipes, alarms or trend buffers directly to a PC so that they can be dealt with more easily. To do this just carry out the export by indicating the path \\NOMEPC\c$\Shared_Polymath\file.xml in the Scripts or when configuring the function predefined in POLYMATH.

Exporting files to various supports

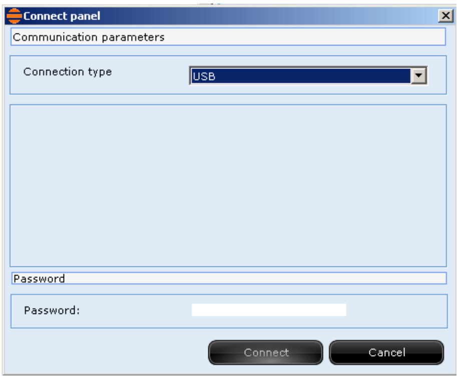
After configuring the panel for Ethernet access or for sharing folders, this connection can be used for exporting data from POLYMATH projects. And therefore it is possible to use the function for exporting and importing recipes, alarms or trend buffers to various devices.
For example, to export data to a physical support other than the main disk (like a mass storage card or USB key), just specify the name of the file including the complete path (e.g. ‘Hard Disk2\fileexportato.xml’) in the destination file path.

Source project File and Backup transfer

It is possible to transfer files between the panel and the PC and vice versa by means of the Transfer File for Windows CE function.
This function is reachable from the main menu clicking on Tools->Utility downloader->Online Tools :



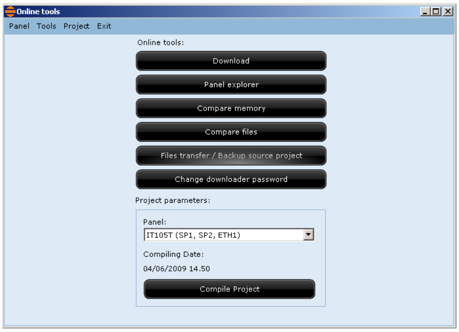
the following image will appear :



Connection type USB

Password:

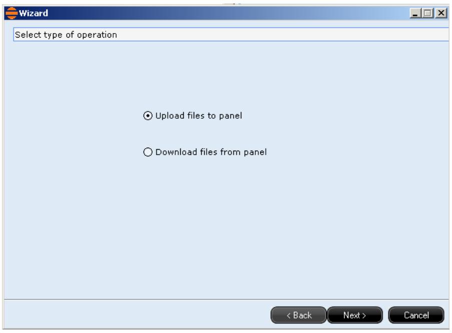
After having chosen the type of connection (USB in our case), click "Connect" to establish the connection between the PC and the Terminal. The following image will appear :



Panel: IT105T (SP1, SP2, ETH1)

Compiling Date: 04/06/2009 14.50

Click "File Transfer/Project source backup". The following image will appear :

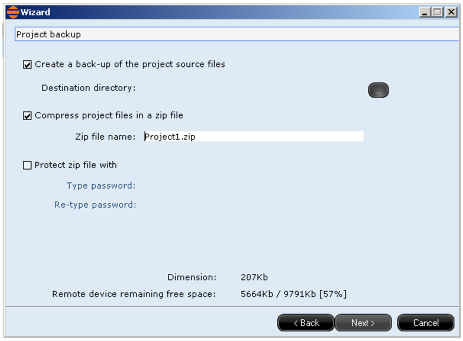


Select type of operation

● Upload files to panel

○ Download files from panel

At this point, the user must choose whether to transfer files from the PC to the panel or vice versa. (In our example, we have chosen to send files to the terminal). If a project is open in Polymath, the following screen will appear :



Project backup

☑ Create a back-up of the project source files

Destination directory:

☑ Compress project files in a zip file

Zip file name: Project1.zip

☐ Protect zip file with

Type password:

Re-type password:

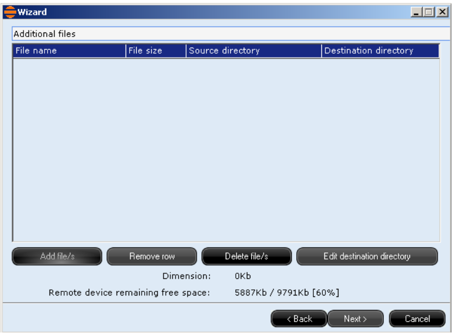
Dimension: 207Kb

Remote device remaining free space: 5664Kb / 9791Kb [57%]

On the screen displayed above, the following Flags appear :

- CREATE PROJECT BACKUP ->with this option, one may decide whether to create a project backup file on the panel (pressing to choose the destination).
- COMPRESS THE PROJECT INTO A ZIP FILE -> to choose whether this backup file is wanted compress into .Zip format.
- PROTECT ZIP FILE WITH A PASSWORD -> allows to decide whether the compressed file should be protected by a password.

Clicking "Forward", the following image appears :



Additional files

File name	File size	Source directory	Destination directory

Dimension: 0Kb

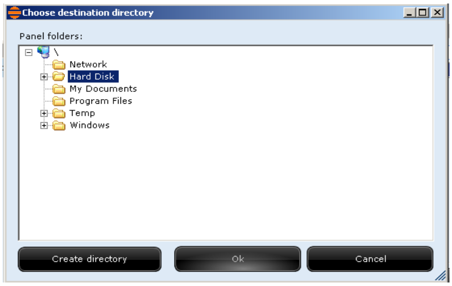
Remote device remaining free space: 5887Kb / 9791Kb [60%]

Clicking "Add Files", the file to be sent to the panel can be chosen.
On the screen displayed above we find the following buttons :

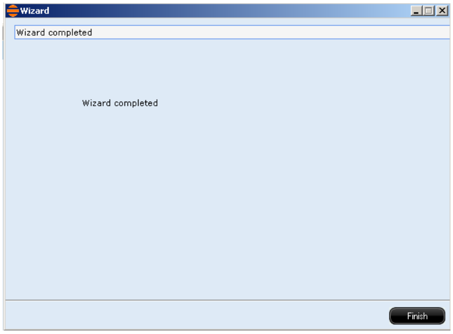
- “Add file/s” -> the procedure for new files can be repeated.

- "Remove row" -> once the row is selected, it can be removed
- "Cancel file/s" -> used to remove files or folders inside of the Panel. This can be useful when there is not enough space available in the panel.
- "Modify destination folder" -> the file destination path can be changed.

Once the file to be sent has been chosen, the user must choose the destination folder. In the example, the "Hard Disk" folder was chosen. Clicking "OK", the transfer will start :



Click "End" when the transfer is complete :



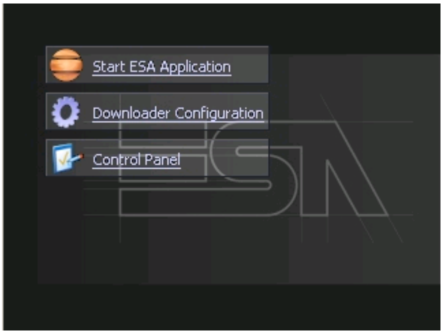
Wizard completed

Wizard completed

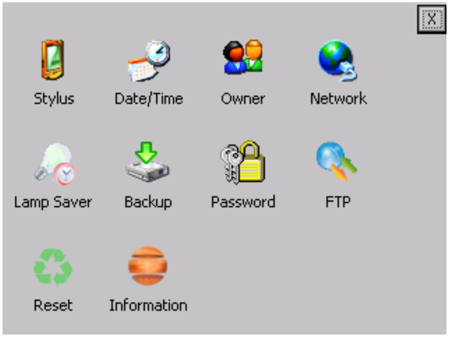
Panel Reset

"Panel Reset" is an application of the terminal control panel which allows to cancel all that has been transferred onto the Hard Disk.

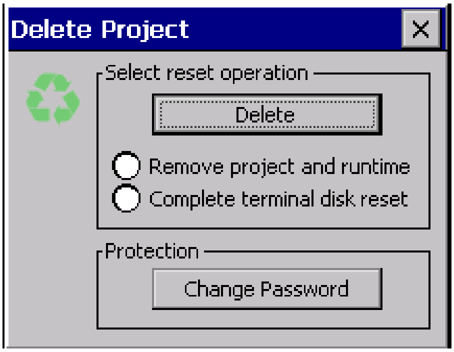
From the initial page of the terminal, click on "Control Panel" :



From the "Control Panel, click on the "Reset" icon :



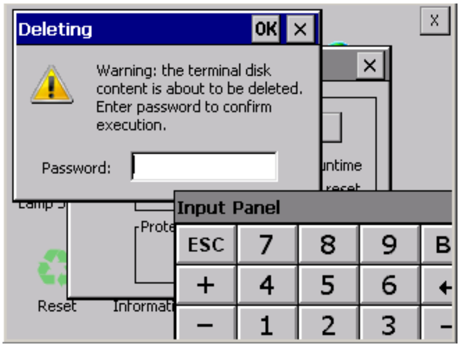
The following image will appear :



From the image above, the user can choose between 2 options:

- “Remove project and runtime” -> choosing this option, both the project and the runtime that have been transferred from Polymath onto the terminal will be cancelled.
- “Complete terminal disk reset” -> choosing this option, the whole content of the "Hard Disk" folder will be cancelled, with the exception of the files that are essential for operating the terminal.

Choosing one of the two options described above and clicking "Delete", the safety password will be requested, since important information contained in the terminal is about to be cancelled :



Deleting

Warning: the terminal disk content is about to be deleted. Enter password to confirm execution.

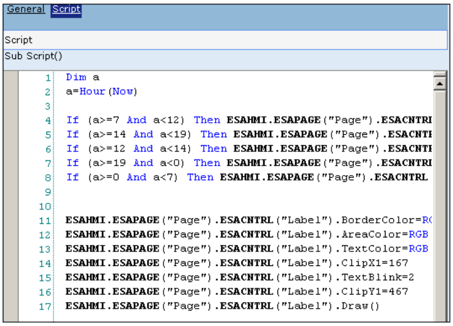
Password:

The default password is "1234". If wanted, it can be changed. After having typed in the password, confirm pressing "Send" on the "Input Panel" keyboard.

9. Scripts

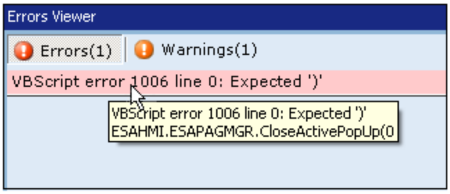
POLYMATH allows the programmer to add to the project whole programmes or functions for managing and editing all the application’s components (graphic objects, variables, recipes etc.) in runtime. Thanks to this, users can complement the set of predefined functions supplied by POLYMATH with those they have created according to their needs. User scripts can be called up in a project when a button is pressed, when an event is triggered or in response to being called by other scripts.
Scripts can be inserted into a project using Project Explorer (see chap. 5, “Scripts” page 140) and their code can be written using simple programming/scripting languages like VBScript.
For details concerning programming techniques (variable declarations, operators, conditional structures and predefined functions) the user is advised to consult specialist manuals relating to the language to be used.
In this chapter we will give information relating to the properties and methods that can be used in POLYMATH scripts with relevant examples.

Editing codes



```
Dim a
a=Hour(Now)

If (a>=7 And a<12) Then ESAHMI.ESAPAGE("Page").ESACNTRI
If (a>=14 And a<19) Then ESAHMI.ESAPAGE("Page").ESACNTI
If (a>=12 And a<14) Then ESAHMI.ESAPAGE("Page").ESACNTI
If (a>=19 And a<0) Then ESAHMI.ESAPAGE("Page").ESACNTRI
If (a>=0 And a<7) Then ESAHMI.ESAPAGE("Page").ESACNTRL


ESAHMI.ESAPAGE("Page").ESACNTRL("Label").BorderColor=R
ESAHMI.ESAPAGE("Page").ESACNTRL("Label").AreaColor=RGB
ESAHMI.ESAPAGE("Page").ESACNTRL("Label").TextColor=RGB
ESAHMI.ESAPAGE("Page").ESACNTRL("Label").ClipX1=167
ESAHMI.ESAPAGE("Page").ESACNTRL("Label").TextBlink=2
ESAHMI.ESAPAGE("Page").ESACNTRL("Label").ClipY1=467
ESAHMI.ESAPAGE("Page").ESACNTRL("Label").Draw()
```

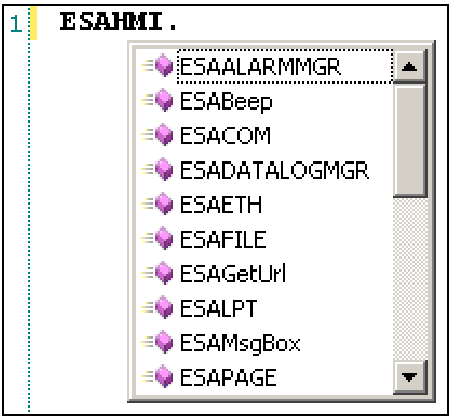
In POLYMATH once a script has been inserted using Project Explorer (see chap. 5, “Scripts” page 140) the editor page for writing the code can be used.
The editor runs a real-time check of the syntax of the code, immediately posting an on-screen warning should it detect any imprecision in the formulation of the instructions.



VBScript error 1006 line 0: Expected ')'

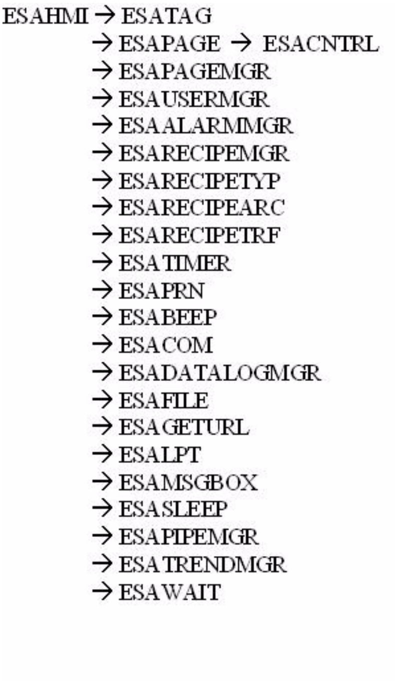
VBScript error 1006 line 0: Expected ')'
ESAHMI.ESAPAGMGR.CloseActivePopUp(0

As indicated in the figure above, a red circle is shown to indicate the existence of an error. When the mouse cursor is placed on it, a complete description of the problem is put on screen. The errors and their related descriptions are also listed the moment the project is validated and compiled.



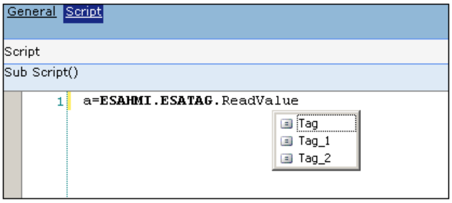
```
ESAHMI.
```

The editor facilities the drafting of the code, showing too the list of objects and properties available for the object that has been inserted (Intellisense mechanism). This list appears whenever the user presses the separation point between the objects or between an object and the method (or property) to be called. When the code is edited, the objects are, in fact, separated from their respective children or methods by the insertion of a point '.' (dot). There follows a chart showing the hierarchy of objects accessible by script.



```
ESAHMI → ESATAG
       → ESAPAGE → ESACNTRL
       → ESAPAGEMGR
       → ESAUSERMGR
       → ESAALARMMGR
       → ESARECIPEMGR
       → ESARECIPETYP
       → ESARECIPEARC
       → ESARECIPETRF
       → ESATIMER
       → ESAPRN
       → ESABEEP
       → ESACOM
       → ESADATALOGMGR
       → ESAFILE
       → ESAGETURL
       → ESALPT
       → ESAMSGBOX
       → ESASLEEP
       → ESAPIPEMGR
       → ESATRENDMGR
       → ESAWAIT
```

Therefore, to indicate an element of the page, we use an instruction of the type:
ESAHMI.**ESAPAGE**("Page").**ESACNTRL**("Label").ControlWidth=67



Script

Sub Script()

```
a=ESAHMI.ESATAG.ReadValue
```

In the case of those objects that require a passage of the name of the reference object (for example, ESAPAGE, ESACNTRL, etc.), after the opening of the brackets just press the ‘?’ key of the keyboard to obtain the list of objects that can be inserted.
The following sections of this chapter will deal with the various objects accessible by Script and set out their properties and functions, giving where necessary practical examples of their use.

__Note:__ Some properties mentioned in the following paragraphs are described as being in read-only mode when using Scripts; for many of these properties, however, there is no physical protection, so there is the possibility that the script will overwrite their value. This overwrite operation is, in any case, not advised. It is thus the programmer's responsibility to avoid the properties indicated as being read-only (R) being edited by the scripts.

Key to types of variable and syntactical premises

The following sections will refer to properties and methods characteristic of objects. The table below gives a rapid key to the abbreviations that will be used.

Table 1: Key to Abbreviations

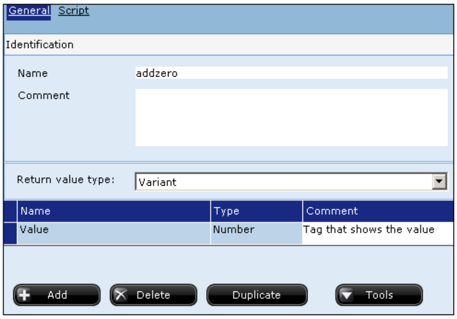
Variable	Abbreviation
Whole	Int
String	Str
Boolean	Bool
Long	Long
Double	Dbl
RGB (color, returned by the RGB function)	RGB
Variant	Var
R	Read, read-only
RW	Read&Write, read and write

If a subroutine (method returning no value) requires an input parameter, the passage can be achieved by using brackets or by leaving them out:
ESAHMI.ESAMSGBOX "Text"
ESAHMI.ESAMSGBOX("Text")
When a subroutine requires more than one input parameter, these parameters must be written consecutively and

separated by a comma (without brackets) as shown in the following example:
ESAHMI.**ESAPAGEMGR**.ShowPageByNumber 32,0
If a function (a method returning a value) requires one of more input parameters, a passage must be made using brackets, as follows:
a=**ESAHMI**.**ESATAG**("Tag_Array").GetTagBitValue(1)
a=**ESAHMI**.**ESAPAGEMGR**.GetTAGBuffer
(“RecipeType‟,”RecipeName”)

Use of functions and subroutines

It is possible to insert into one’s projects functions and subroutines (the former return a value while the latter do not) that can be called by a script at any moment. The definition of these functions happens as with normal scripts, only that it is necessary to specify the type of input and output parameters.



Identification

Name addzero
Comment

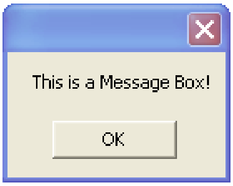
Return value type: Variant

Name	Type	Comment
Value	Number	Tag that shows the value

After creating the script using Project Explorer, enter in the General mask the type of output value (‘None’ if it is a subroutine) and enter the type in question in the list of input parameters.

ESAMSGBOX Method

The ESAMSGBOX method serves to make a message window appear on the terminal with the value provided.


This is a Message Box!

This method is useful for debugging the script as it allows the user to view, for example, the value of a variable at a given moment of its execution.
According to the simplest syntax for invoking that method, the passage of a value (whether constant or variable) is as set out in the following example:
ESAHMI.**ESAMSGBOX**("Message Text")
which in Runtime makes the window containing the message "Message Text" appear. If, on the other hand, we use the following syntax:
ESAHMI.**ESAMSGBOX**(VariableName)
in Runtime a window containing the current value of the variable 'VariableName') appears.
Alternatively the method can be invoked with the passage of two values; in this case the first parameter indicates the string to be displayed while the second parameter must be a string that then appears in the title bar of the window. See example below:
ESAHMI.**ESAMSGBOX** "Message Text","Message Title"
will make a window containing the message 'Message Text' appear, while the title appearing in the bar above this window will be 'Message Title'.

Warning: *The ESAMSGBOX method is advised only for debugging the script, or rather during its testing. For the final project, when messages are to be presented on screen for the operator, we strongly advise the use of pop-up pages (whose appearance can be controlled using Scripts).*

Warning: *To execute scripts relating to a variable the continuous update option must be enabled during the setting of the variable in POLYMATH (see chap. 5, "Device" page 93). In addition, tags can only use the methods and properties relating to thresholds if these have already been assigned.*

Methods of the ESATAG objects accessible from ScriptESAUSERMGR

Table 2: Methods of the ESATAG objects accessible from Script

Method	Description	OUT	IN
GetId	Returns the IC variable code inserted under string form	Long	String
GetDataType	Gets the code type of the value corresponding to the type of tag	Integ er	String
GetRawDataT ype	Gets the code type of the "rough" value (value inside the device) from the variable	Integ er	String
IsInvalid	Checks if the tag value is not valid	Bool ean	String
IsOffline	Checks if the tag is correctly Off Line	Bool ena	String
IsOffscan	Checks if the tag is correctly Off Scan	Bool ean	String
IsForced	Checks if the tag value had been forced whilst in Off Scan	Bool ean	String
IsInhibit	Checks if the threshold of the tag is forbidden	Bool ean	String

SetOffscan	Set the tag to the Off Scan status	-	String(tag name)
			Boolean(O ffscan)

SetInhibit	Set the threshold of the tag to the forbidden status	-	String(Tag name)
			Boolean(I nhibit)

GetStringLen	Gets the length configured of the tag's string	Long	String
GetFillingType	Gets the filling assigned to the code of the tag's string	Integ er	String

Table 2: Methods of the ESATAG objects accessible from Script

Method	Description	OUT	IN
GetFillingChar	Gets the fillings character of the tag's string	Integer	String

SetFillingChar	Changes the fillings character of the tag's string	-	String(Tag name)
			Integer(Fillchar)

GetInputValueLowerLimitGetTagThrsDevReference	Gets the lower limit of the operations in entry of a numerical tag	Double	String
GetInputValueUpperLimit	Gets the upper limit of the operations in entry of a numerical tag	Dbl	String
GetInputRawValueLowerLimit	Gets the lower limit of the operations in exit of a numerical tag	Dbl	String
GetInputRawValueUpperLimit	Gets the upper limit of the operations in exit of a numerical tag	Dbl	String
GetConversionType	Gets the conversion type code of a numerical tag	Integer	String
GetConversionX1Par	Gets the parameter of the mathematical conversion of the value of a numerical tag	Dbl	String
GetConversionY1Par	Gets the parameter of the mathematical conversion of the value of a numerical tag	Dbl	String
GetConversionX2Par	Gets the parameter of the mathematical conversion of the value of a numerical tag	Dbl	String
GetConversionY2Par	Gets the parameter of the mathematical conversion of the value of a numerical tag	Dbl	String

Table 2: Methods of the ESATAG objects accessible from Script

Method	Description	OUT	IN

SetInputValueLowerLimit	Changes the lower limit of the operations in entry of a numerical tag	-	String(Tag name)
			Dbl(Limit)

SetInputValueUpperLimit	Changes the upper limit of the operations in entry of a numerical tag	-	String(Tag name)
			Double(Limit)

SetInputRawValueLowerLimit	Changes the lower limit of the operations in exit of a numerical tag	-	String(Tag name)
			Double(Limit)

SetInputRawValueUpperLimit	Changes the upper limit of the operations in exit of a numerical tag	-	String(tag name)
			Double(Limit)

SetConversionX1Par	Changes the parameter of the mathematical conversion of the value of a numerical tag	-	String(Tag name)
			Double(ConvPatameter)

SetConversionY1Par	Changes the parameter of the mathematical conversion of the value of a numerical tag	-	String(Tag name)
			Double(ConvPatameter)

SetConversionX2Par	Changes the parameter of the mathematical conversion of the value of a numerical tag	-	String(Tag name)
			Double(ConvPatameter)

SetConversionX2Par	Changes the parameter of the mathematical conversion of the value of a numerical tag	-	String(Tag name)
			Double(ConvPatameter)

GetCurrentValue	Reads the current value saved in the tag	Variant	String

Table 2: Methods of the ESATAG objects accessible from Script

Method	Description	OUT	IN
GetCurrentRawValue	Read the current "rough" value (value inside the device) saved in the tag	Variant	String
ReadValue	Read the value of the tag from the device	Variant	String

WriteValue	Writes a new value of the tag in the device	-	String(Tag name)
			Variant(Value)

ReadElement	Reads from the device the single element value of the tag's array	Variant	String(Tag name)
			Integer(Index)

WriteElement	Writes from the device the single element value of the tag's array	-	String(tag Name)
			Integer(Index)
			Variant(Value)

ReadBit	Reads from the device the single bit value from an array or the numerical variable	Boolean	String(Tag name)
			Long(Index)

WriteBit	Writes on the device the single bit value from an array or the numerical variable	-	String(Tag name)
			Long(Index)
			Boolean(Value)

GetThresholdType	Gets the code type of the configured threshold	Integer	String
GetThreshold State	Gets the current state of the tag's threshold	Integer	String

GetThresholdLevelState	Gets the current specific level state of the threshold	Integer	String(Tag name)
			Integer(Level)

The object ESAUSERMGR

This object offers functions relating to the user currently logged onto the terminal. The following table describes the methods that can be used with this object using a syntax of **ESAHMI**.**ESAUSERMGR**.GetCurrentUser()

ESAUSERMGR methods accessible with Scripts

Table 3: ESAUSERMGR methods accessible with Scripts

Method	Description	OUT	IN
GetCurrentUserName	Returns the name of the user currently logged in	Str	-
GetCurrentUserLevel	Returns the level of the user currently logged in	Int	-

The object ESAALARMMGR

This object offers functions relating to the management of the alarms in the project. The following table describes the methods that can be used with this object using a syntax of **ESAHMI**.**ESAALARMMGR**.ClearAlarm(“Alarm_1“)

ESAALARMMGR methods accessible with Scripts

Table 4: ESAALARMMGR methods accessible with Scripts

Method	Description	OUT	IN
AlarmOn	Raises named alarm with lag set using POLYMATH. Needs as an input parameter the name of the alarm to be acquired. Bear in mind that you cannot activate in Runtime the event ON for an alarm whose status is already ON (the status must first be changed to OFF)	-	AlarmName (Str)
RaiseAlarm	Raises named alarm without lag set using POLYMATH. Needs as an input parameter the name of the alarm to be acquired. Bear in mind that you cannot activate in Runtime the event ON for an alarm whose status is already ON (the status must first be changed to OFF)	-	AlarmName (Str)

Table 4: ESAALARMMGR methods accessible with Scripts

Method	Description	OUT	IN
ClearAlarm	Forces the named alarm setting its status as 'terminated' (OFF). Needs as an input parameter the name of the alarm to be terminated	-	AlarmName (Str)
AckInstances	Acknowledges all the instances of the named alarm (whether of AlarmISA or OnlyAck type), that is, if allowed by the settings of the alarm relating to the acknowledgement of multiple instances. Needs as an input parameter the name of the alarm, of the operator and of the station from which the request is made (valid parameter in the case of a network)	-	AlarmName (Str) Operator (Str) Station (Str)
AckGroup	Acknowledges all the instances of the alarm (whether of AlarmISA or OnlyAck type) of the named group, that is, if allowed by the settings of the alarm relating to global acknowledgement . Needs as an input parameter the name of the alarm, of the operator and of the station from which the request is made (valid parameter in the case of a network)	-	GroupName (Str) Operator (Str) Station (Str)
AckGlobal	Acknowledges all the alarms (whether of AlarmISA or OnlyAck type), that is, if allowed by the settings of the alarm relating to global acknowledgement. Needs as an input parameter the name of the alarm, of the operator and of the station from which the request is made (valid parameter in the case of a network)	-	Operator (Str) Station (Str)

Table 4: ESAALARMMGR methods accessible with Scripts

Method	Description	OUT	IN
AckAlarm	Acknowledges the alarm specified by the first input parameter (whether it is AlarmISA type or OnlyAck). Needs as an input parameter the number with which the alarm has been registered (ID), the operator's name and that of the station from which the request is made (valid parameter in the case of a network)	-	RegistrationID (Long) Operator (Str) Station (Str)
AlarmsExport	Exports active alarms to the file in main directory of the terminal. Needs two input parameters, one relating to the name to give to the file and one (whole) relating to its type; the possible file extensions are: (FileType=1) XML (FileType=2) CSV	-	FileName (Str) FileType (Int)
HistoryExport	Exports history alarms to the file in main directory of the terminal. Needs two input parameters, one relating to the name to give to the file and one (whole) relating to its type; the possible file extensions are: (FileType=1) XML (FileType=2) CSV	-	FileName (Str) FileType (Int)
HistoryDelete	Cancels the buffer of alarm history and needs no input parameter	-	-

The object ESARECIPEMGR

This object offers functions relating to the management of the recipes in the project. The following table describes the methods that can be used with this object using a syntax of **ESAHMI**.**ESARECIPEMGR**.GetTAGBuffer "RecipeType","variable1"

ESARECIPEMGR methods accessible with Scripts

Table 5: ESARECIPEMGR methods accessible with Scripts

Method	Description	OUT	IN
GetTAGBuffer	Returns the name of the tag buffer related to a field of the recipe; needs as input parameters the name of the type of recipe and the field buffer	Str	StructureName (Str) FieldName (Str)
ClearTAGBuffer	Clears the buffer of all recipe variables (including ID, name and comment): the numerical variables are set at 0, the strings at ""; it needs as an input parameter the type of recipe	-	StructureName (Str)

RecipeCompare	Returns a Boolean value indicating whether the two recipes indicated are the similar (1) or different (0); the test is carried out on the versions that have been saved. Requires as an input parameter the name of the recipe type and of two recipes to compare	Bool	StructureName (Str)
			RecipeName1(Str)
			RecipeName2(Str)

IsActive	Indicates if it is in course and therefore a transfer is active	Integer	StructureName (Str)

The object ESARECIPETYP

This object offers functions relating to the management of the recipe types in the project. The following table describes the methods that can be used with this object using a syntax of **ESAHMI**.**ESARECIPETYP**.GetFirstRecipeTypeName()

ESARECIPETYP methods accessible with Scripts

Table 6: ESARECIPETYP methods accessible with Scripts

Method	Description	OUT	IN
GetFirstRecipeTypeName	Returns the name of the first recipe type (in ascending order of the IDs set using POLYMATH)	Str	-
GetNexttRecipeTypeName	Returns the name of the recipe type after the one just displayed (in ascending order of the IDs set using POLYMATH). Requires the method GetFirstRecipeTypeName to have been called at least once	Str	-

The object ESARECIPEARC

This object offers functions relating to the management of the filing of recipes in the project. The following table describes the methods that can be used with this object using a syntax of
ESAHMI.**ESARECIPEARC**.RecipeImport(“filename.xml‟)

ESARECIPEARC methods accessible with Scripts

Table 7: ESARECIPEARC methods accessible with Scripts

Method	Description	OUT	IN
GetFirstRecipeName	Returns the name of the first recipe in the terminal belonging to the type specified. Needs as an input parameter the type of recipe whose list is to be examined (in chronological order of the insertion of the recipes).	Str	StructurName (Str)

Table 7: ESARECIPEARC methods accessible with Scripts

Method	Description	OUT	IN
GetNextRecipeName	Returns the name of the next recipe in the terminal belonging to the type specified. Needs as an input parameter the type of recipe whose list is to be examined (in chronological order of the insertion of the recipes). The method GetFirstRecipeName has to have been called at least once	Str	StructurName (Str)
DeleteRecipe	Deletes the recipe specified by the parameters as an input parameter. Requires: the recipe type name, the recipe name and a Boolean variable indicating whether the user must confirm the operation (1) or whether deletion is automatic (0)	-	StructurName (Str) RecipeName (Str) UserFlag (Bool)
RecipeExists	Returns a Boolean value indicating whether the recipe referred to exists (1) or not (0). The test is carried out on thee recipes saved. Requires as an input parameter the type of recipe and the name of the recipe to be checked	Bool	StructurName (Str) RecipeName (Str)

Table 7: ESARECIPEARC methods accessible with Scripts

Method	Description	OUT	IN
RecipeExport	Exports the recipes referred to in the input parameters. Necessary specifications: the name of the destination file (.xml or .csv. If an empty string is provided, the name can be assigned in Runtime), the recipe type name and the list of recipes to be inserted (names separated by the TAB character on the keyboard). If one of the two parameters (or both) is an empty string, all recipes are exported without consideration to their type or name	Int	Filename (Str) StructurName (Str) RecipeList (Str)
RecipeImport	Imports the recipes contained in the file (.xml or .csv) indicated by the input string. If the input string is empty, when this method is called in Runtime the window for exporting files is shown to allow a search for the file from which to import the recipe	Int	Filename (Str)

The object ESARECIPETRF

This object offers functions relating to the transfer of recipes the project. The following table describes the methods that can be used with this object using a syntax of **ESAHMI**.**ESARECIPETRF**.RecipeBufferUpload "RecipeType",”1”

ESARECIPETRF methods accessible with Scripts

Table 8: ESARECIPETRF methods accessible with Scripts

Method	Description	OUT	IN
LoadRecipe	Loads the recipe specified by the input parameter into the video buffer. It is necessary to provide: the type of recipe, the name of the recipe and a Boolean variable indicating whether the user must confirm the operation (1) or whether the loading is automatic (0)	-	StructureName (Str) RecipeName (Str) UserFlag (Bool)
SaveRecipe	Saves the data in the video buffer into the recipe specified by the input parameter. It is necessary to provide: the type of recipe, the name of the recipe and a Boolean variable indicating whether the user must confirm the operation (1) or whether the loading is automatic (0)	-	StructureName (Str) RecipeName (Str) UserFlag (Bool)
RecipeDownload	Downloads onto a device the recipe specified by the input parameter. It is necessary to provide: the type of recipe, the name of the recipe and a Boolean variable indicating whether the download must follow synchronization (1) or not (0)	-	StructureName (Str) RecipeName (Str) SyncFlag (Bool)

Table 8: ESARECIPETRF methods accessible with Scripts

Method	Description	OUT	IN
RecipeBufferDownload	Downloads the video buffer onto the device corresponding to the recipe specified by the input parameter. It is necessary to provide: the type of recipe and a Boolean variable indicating whether the download must follow synchronization (1) or not (0)	-	StructureName (Str) SyncFlag (Bool)
RecipeBufferUpload	Uploads the video buffer from the device corresponding to the recipe indicated by the input parameters. It is necessary to provide: the type of recipe and a Boolean variable indicating whether the download must follow synchronization (1) or not (0)	-	StructureName (Str) SyncFlag (Bool)

The object ESAPIPEMGR

This object offers functions relating to the Pipelines in the project. The following table describes the methods that can be used with this object using a syntax of **ESAHMI**.**ESAPIPEMGR**.StartPipelineByNumber(2)

ESAPIPEMGR methods accessible with Scripts

Table 9: ESAPIPEMGR methods accessible with Scripts

Method	Description	OUT	IN
StartPipelineByName	Starts the Pipeline indicated by the input parameter; needs the string containing the name of the Pipeline to be provided	-	PipelineName (Str)

Table 9: ESAPIPEMGR methods accessible with Scripts

Method	Description	OUT	IN
StartPipelineByNumber	Starts the Pipeline indicated by the input parameter; needs the number relating to the Pipeline ID	-	PipelineID (Int)
StopPipelineByName	Stops the Pipeline indicated by the input parameter; needs the string containing the name of the Pipeline to be provided	-	PipelineName (Str)
StopPipelineByNumber	Stops the Pipeline indicated by the input parameter; needs the number relating to the Pipeline ID	-	PipelineID (Int)
WritePipelineByName	Forces the writing of the Pipeline indicated by the input parameter (also if the Pipeline has been stopped); the string containing the name of the Pipeline must be provided	-	PipelineName (Str)
WritePipelineByNumber	Forces the writing of the Pipeline indicated by the input parameter (also if the Pipeline has been stopped); needs the number relating to the Pipeline ID to be provided	-	PipelineID (Int)
GetPipelineStatusByName	Returns an indication of the status of the pipeline referred to in the input parameter; the string containing the name of the Pipeline must be provided. The complete returned data will have one of the following values and meanings§: 1 - Inactive Pipeline 2 - Active Pipeline 3 - Disconnected Pipeline (no communication)	Int	PipelineName (Str)

Table 9: ESAPIPEMGR methods accessible with Scripts

Method	Description	OUT	IN
GetPipelineStatusByNumber	Returns an indication of the status of the pipeline referred to in the input parameter; the number relating to the ID of the Pipeline must be transferred. The complete returned data will have one of the following values and meanings§: 1 - Inactive Pipeline 2 - Active Pipeline 3 - Disconnected Pipeline (no communication)	Int	PipelineID (Int)

The object ESATIMER

This object offers functions relating to the timers in the project. The following table describes the methods that can be used with this object using a syntax of the type var=**ESAHMI**.**ESATIMER**(‶nomeTimer‶).State **ESAHMI**.**ESATIMER**(‶nomeTimer‶).Stop()

ESATIMER properties accessible with Scripts

Table 10: ESATIMER properties accessible with Scripts

Properties	Description	Type	RW
DirectionCount	Defines Timer counting mode; possible values of this property are: 1 - Ascending (0 to Duration) 2 - Descending (Duration to 0)	Long	R
Duration	Defines the duration of the Timer. The meaning of this value depends on the type of Timer: - if it is Once Only or Normal, the unit of duration is 1/10 second - if it is Single alarm, the duration is a Date and Time expressed as the number of seconds from 1/1/1970 - if it is AlarmTime mode, the duration is the current time of day expressed a number of seconds past midnight	Long	R

Table 10: ESATIMER properties accessible with Scripts

Properties	Description	Typ e	RW
Mode	Defines the Timer mode; possible values of this property are: 1 - Once onlt 2 - Normal 3 - Single alarm 4 - AlarmTime	Long	R
State	Defines the current state of the Timer; possible values of this property are: 0 - Not Active 1 - Counting 2 - Terminated 3 - Suspended	Long	R
Count	Indicates the current position of the counter	Long	R

ESATIMER methods accessible with Scripts

Table 11: ESATIMER methods accessible with Scripts

Method	Description	OUT	IN
SetTimerValue	Sets the duration value to correspond with the input value. Returns duration set for timer	Long	Duration (Long)
Start	Starts the timer; returns the value of the start-timer activity	Long	-
Stop	Stops the timer; returns the value of the stop-timer activity	Long	-
Suspend	Suspends the timer; returns the value of the suspend-timer activity	Long	-

The object ESATRENDMGR

ESATRENDMGR gives access to certain properties and methods that are useful for managing Trend Buffers.

ESATRENDMGR properties accessible with Scripts

It is important to emphasize that the properties offered by ESATRENDMGR are only available as Read-only and the Trend buffer ID that the following code lines refer to must be defined at the beginning of the Script (or at least before using the properties).
To exemplify this, we will analyze the following code lines:

```
ESAHMI.ESATRENDMGR.TrendId=5
a=ESAHMI.ESATRENDMGR.Name
ESAHMI.ESATRENDMGR.TrendId=1
b=ESAHMI.ESATRENDMGR.Name
```

After performing the 4 instructions listed above, variable 'a' will contain the name of the Trend buffer with ID=5, while variable 'b' will contain the name of the Trend buffer with ID=1. (POLYMATH assigns the IDs during the editing of the Trend buffer).

Table 12: ESATRENDMGR properties accessible with Scripts

Properties	Description	Type	RW
TrendId	This is a unique code identifying the trend selected	Long	R
Name	This is the name of the Trend buffer selected. It is unique in that two Trend buffers with the same name cannot exist	Str	R
Type	Defines the type of variable assigned to the Trend buffer selected. The possible values of this property are: 0 - Single value; 1 - Array	Long	R
SourceTag	This is the name of the variable assigned to the Trend buffer selected	Str	R
StrobeType	This is a code identifying the event to start the acquisition of new sample readings for the Buffer selected. Possible codes are: 0 ONTIMER 1 ONSTROBERISE 2 ONSTROBEFALL 3 ONCOMMAND 4 ONTAG	Long	R

Table 12: ESATRENDMGR properties accessible with Scripts

Properties	Description	Type	RW
StrobeTag	Used only if StrobeType has a value of 1 or 2; indicates the name of the variable that triggers the acquisition	Str	R
StrobeTimer	Used only if StrobeType has a value of 0. This is the Timer identity code used by the Trend	Long	R
BufferSize	Represents the maximum number of sample readings that can be saved in the Buffer selected (value set in POLYMATH). With Array-type trends, this value is an exact multiple of the array dimension	Long	R
SamplesNum	Represents the number of sample readings currently in the Buffer selected	Long	R
WarningLevel	Defines the percentage threshold of the number of samples for which the OnWarningLevel event is generated for the buffer selected (value set in POLYMATH).	Long	R
Enabled	This is a Boolean flag indicating the active state of the buffer selected. If at 0 the trend activities are ignored, while if at 1, the trend functions regularly	Bool	R
StatusBit	The number of trend status area bits assigned to the trend. If the buffer is full, the bit assumes a value of 1; if it is not full, 0; if the buffer is not assigned to external bits, -1	Long	R

ESATRENDMGR methods accessible with Scripts

Table 13: ESATRENDMGR methods accessible with Scripts

Method	Description	OUT	IN
GetFirstSample	Returns attributes of the first (least recent) sample of the trend buffer specified by the input parameter. Apart from the Trend ID, requires as input parameters pointers linked to Variant, String and Boolean type variables to which the values are returned. Returns TRUE if the operation is successful, while Quality indicates whether the value of 'Value' exists or not (if Quality=FALSE, the buffer is empty)	Bool	TrendId (Long) Value (Var) Time (Str) Quality (Bool)
GetNextSample	Returns attributes of the next sample of the trend buffer specified by the input parameter (next in chronological order relative to the last sample read by the GetFirstSample methods or by GetNextSamples itself). Apart from the Trend ID, requires as input parameters pointers linked to Variant, String and Boolean type variables to which the values are returned. Returns TRUE if the operation is successful, while Quality indicates whether the value of 'Value' exists or not (if Quality=FALSE, the buffer has no successive elements)	Bool	TrendId (Long) Value (Var) Time (Str) Quality (Bool)
IsEmpty	Checks whether the Trend specified by the input ID is empty (returns 1) or not (returns 0)	Bool	TrendId (Long)

Table 13: ESATRENDMGR methods accessible with Scripts

Method	Description	OUT	IN
PutValue	Adds to the trend indicated by the input ID a new sample with the attributes provided as input parameters. The sample time is the current one.	-	TrendId (Long) Value (Var) Quality (Bool)
PutValueAt	Adds to the trend indicated by the input ID a new sample with the attributes passed as input parameters. Time must be expressed as “DD/MM/YYYY hh:mm:ss,mmm”	-	TrendId (Long) Value (Var) Time (Str) Quality (Bool)
ResetSamples	Removes all samples from the specified trend and triggers the event OnBufferClear	-	TrendId (Long)
AcquireSample	Acquires a new sample for the trend indicated by the input parameter. This method functions independently of the type of trend acquisition and of the value of its attribute Enabled	-	TrendId (Long)
ExportPartTrendBuffer	Exports part of the buffer of the trend indicated by the input parameter. Requires in addition the passage of the destination file name, the type of file (1 - xml, 2 - csv) and the times of the first and last samples to be exported	-	TrendId (Long) FileName (Str) Type (Int) TimeStart (Str) TimeEnd (Str)
ExportFullTrendBuffer	Exports all the samples in the buffer of the trend indicated by the input parameter. Requires in addition the passage of the destination file name, the type of file (1 - xml, 2 - csv)	-	TrendId (Long) FileName (Str) Type (Int)

Table 13: ESATRENDMGR methods accessible with Scripts

Method	Description	OUT	IN
ChangeScaleLimit	Changes the limits of the vertical scale assigned to the penline. All tracks of the trend specified are updated. In addition needs as input parameter the new upper and lower coordinates of the track scale	-	TrendId (Long) MinLimit (Dbl) MaxLimit (Dbl)
GetTrendId	Returns the identifying number (ID) of a trend whose name is known (provided as input parameter for the method)	Long	TrendName (Str)
Enable	Used to enable or disable the trend (in practice operates on attribute Enabled). Requires as input parameter the trend to edit and the value to be attributed (1 enabled; 0 disabled)	-	TrendId (Long) Enabled (Boolean)

The object ESAPAGEMGR

The object ESAPAGEMGR offers functions and methods for the global management of pages within the project. The following table describes the methods that can be used with this object using a syntax similar to:
ESAHMI.**ESAPAGEMGR**.ShowNextPage()

Properties of the object ESAPAGEMGR accessible from the script

Table 14: Properties of the object ESAPAGEMGR accessible from the script

Properties	Description	OUT	IN
CurrentLanguage	Indicates the identity code of the current language in use	Integer	RW

ESAPAGEMGR methods accessible with Scripts

Table 15: ESAPAGEMGR methods accessible with Scripts

Method	Description	OUT	IN
ShowNext Page	Shows the next page (following order of page ID number)	-	-
ShowPage ByName	Shows the page identifying it by the input parameter; needs a string containing the name of the page to be passed	-	PageName (Str)
ShowPage ByNumber	Shows the page identifying it by the input parameter; needs an integer containing the page ID to be passed	-	PageID (Int)
Show Previous Page	Shows the preceding page (following order of page ID number)	-	-
ShowHelp Page	Makes it possible to show the Help defined in POLYMATH relating to the page (full or popup) currently being displayed (see chap. 5, "Help pages" page 106)	-	-
ClosePopUp PageBy Name	Closes the popup page indicated in the input parameter; needs the passage of a string relating to the name of the popup page		PageName (Str)
ClosePopUp PageBy Number	Closes the popup page indicated in the input parameter; needs the passage of an integer relating to the identifying number of the popup page		PageID (Int)

Table 15: ESAPAGEMGR methods accessible with Scripts

Method	Description	OUT	IN
CloseActive PopUp	Closes the currently active popup page (the one in focus); does not work if no popup is in focus at the moment the method is invoked	-	-
CloseAllPop Up	Closes all currently open popup pages	-	-
GetNum PopupOpen	Returns the number (counter) of the currently open popup pages	Int	-
GetPopup Open	Returns the identifying number of the popup page corresponding to the index number provided as an input parameter. The index number provided as an input parameter marks the order of the opening of the pages; for example, if 2 pages are opened, index 0 identifies the first page opened while index 1 identifies the second page opened. It is advisable to use this command when in the programming phase the number of popup opened at the moment the method is invoked can be foreseen	Int	Index (Int)
GetPage Name	Returns a string of the page name corresponding to the input parameter; needs the identifying number of the page whose name is required to be passed	Str	PageID (Int)
GetPage Number	Returns the identifying number of the page corresponding to the input parameter; needs the name of the page whose ID is required to be passed	Int	PageName (Str)

Table 15: ESAPAGEMGR methods accessible with Scripts

Method	Description	OUT	IN
IsPageName Open	Returns a Boolean value (0 if False, 1 if True) indicating whether the page relating to the input parameter is open or not; needs the passage of the name of the page that is to be checked	Bool	PageName (Str)
IsPageNum Open	Returns a Boolean value (0 if False, 1 if True) indicating whether the page relating to the input parameter is open or not; needs the passage of the ID number of the page that is to be checked	Bool	PageID (Int)
ActivePage	Activates a specific window	-	PageID (Int)
ShowSequencePageBy Name	Displays the page where the name of the page and the name of the sequence are specified	-	Sequence Name(Str) PageName (Str)

ShowSequencePageBy Number	Displays the page where the Id of the page and the name of the sequence are specified	-	SequenceId(Int)
			PageId(Int)

ShowPreviousSequence Page	Displays the previous page inside the current sequence	-	-
ShowNext Sequence Page	Displays the following page inside the current sequence	-	-
Get Sequence Name	Gets the name of the page sequence	String	SequenceId(Int)
Get Sequence Page	Gets the name of the page sequence	Integer	Sequence Name(Str)
LightUp	Varies the light on the display by increasing it	-	-

Table 15: ESAPAGEMGR methods accessible with Scripts

Method	Description	OUT	IN
LightDown	Varies the light on the display by decreasing it	-	-
LightSet	Varies the light on the display by setting the specific value	-	LightLevel (Int)

The object ESAPAGE

The object ESAPAGE allows some properties of an individual page to be managed as set out in the following table. The string relating to the name of the references pages must be passed to it. This object does not have usable methods but in the following section we will analyze the object ESACNTRL (child of the object ESAPAGE) which enables the user to act on the individual objects contained in a page.
The correct syntax for using the object ESAPAGE is as follows:
ESAHMI.**ESAPAGE**(“NamePage“).AreaColor=RGB(23,24,23)

Warning: *The methods and Properties of ESAPAGE (and thus of its children's objects) are only applicable to the currently open page. If a Script tries to edit elements in a page not currently open in runtime, an error signal will appear.*

ESAPAGE properties accessible with Scripts

Table 16: ESAPAGE properties accessible with Scripts

Properties	Description	Type	RW
Name	The name attributed to the page by POLYMATH in the project editing phase	Str	R
Number	The number attributed to the page by POLYMATH in the project editing phase	Int	R
Width	The value of the width of the page	Int	R
Height	The value of the height of the page	Int	R
AreaColor	The background color of the page. This can also be changed by inserting in the input phase an RGB value (Long) returned, for example, by the RGB function (e.g. AreaColor=RGB(24,255,0).	RGB	RW

The object ESACNTRL

ESACNTRL (contained within ESAPAGE) puts at the operator's disposal a series of methods and properties relating to the individual objects present in a page. The following sections will analyze the properties and methods accessible using Scripts in relation to each graphic object that can be added to a page (maintaining the order followed in the chapter on the Properties Editor). All the graphic elements use the same Draw () method for redrawing the element in question.
The correct syntax to access the properties of the object ESACNTRL is:
ESAHMI.**ESAPAGE**("PageName").**ESACNTRL**("ObjectName").BorderColor=RGB(32,255,0)

Warning: *When a Script modifies the graphic properties of an object, these are displayed only when the object is redrawn using the appropriate Draw () method. This method redraws the object simultaneously applying all the changes made to the attributes up to the moment the display is invoked. Dynamic fields that show a value also have the RefreshControl () method capable of updating only the value of the field while ignoring the graphic properties that have been changed.*

Properties of ESACNTRL - Rectangle

Table 17: Properties of ESACNTRL - Rectangle

Properties	Description	Type	RW
ControlLeft	Defines the position (in pixels) counting from the left where the object has currently been drawn. If this value is changed, the object is moved horizontally (when redrawn with the Draw method).	Int	RW
ControlTop	Defines the position (in pixels) counting from the top where the object has currently been drawn. If this value is changed, the object is moved horizontally (when redrawn with the Draw method).	Int	RW
ControlWidth	Defines the width of the object	Int	R
ControlHeight	Defines the height of the object	Int	R
ControlHide	Defines whether the object should be visible (0) or invisible (1). Modifying this command provokes an immediate redrawing without needing to invoke the Draw method.	Bool	RW
BorderColor	Defines the color of the border of the object currently being redrawn. Can be changed by attributing an RGB (Long) value returned, for example, by the RGB function (e.g. BorderColor=RGB(24,255,0). The change will appear in runtime after invoking the Draw method.	RGB	RW

Table 17: Properties of ESACNTRL - Rectangle

Properties	Description	Typ e	RW
BorderBlink	Defines whether the edge of the object should blink or not. Possible values of this property are: 0 - No blinking 1 - Slow blinking 2 - Rapid blinking If a different value from the preceding ones is attributed, the property is forced to 0. Modifying this command provokes an immediate redrawing without needing to invoke the Draw method.	Int	RW
AreaColor	Defines the color of the internal area of the object currently being redrawn. Can be changed by attributing an RGB (Long) value returned, for example, by the RGB function (e.g. BorderColor=RGB(24,255,0). The change will appear in runtime after invoking the Draw method.	RGB	RW
FillDir	Defines infill direction of the object currently being redrawn; the values may be as follows: 0 - From bottom to top 1 - From top to bottom 2 - From left to right 3 - From right to left If a different value from the preceding ones is attributed, the property is forced to 0. The change is shown in runtime after the Draw method is invoked.	Int	RW
FillPercent	Defines the percentage infill of the object currently being redrawn. The change is shown in runtime after the Draw method is invoked.	Int	RW

Table 17: Properties of ESACNTRL - Rectangle

Properties	Description	Typ e	RW
FillColor	Defines the infill color of the object currently being redrawn. Can be changed by attributing an RGB (Long) value returned, for example, by the RGB function (e.g. BorderColor= RGB (24,255,0). The change is shown in runtime after the Draw method is invoked.	RGB	RW

Methods of ESACNTRL - Rectangle

Table 18: Methods of ESACNTRL - Rectangle

Method	Description	OUT	IN
Draw	Redraws the whole object from the beginning, updating all the graphic properties that were changed.	-	-

Properties of ESACNTRL - Ellipse

Table 19: Properties of ESACNTRL - Ellipse

Properties	Description	Type	RW
ControlLeft	Defines the position (in pixels) counting from the left where the object has currently been drawn. If this value is changed, the object is moved horizontally (when redrawn with the Draw method).	Int	RW
ControlTop	Defines the position (in pixels) counting from the top where the object has currently been drawn. If this value is changed, the object is moved horizontally (when redrawn with the Draw method).	Int	RW
ControlWidth	Defines the width of the object	Int	R

Table 19: Properties of ESACNTRL - Ellipse

Properties	Description	Type	RW
ControlHeight	Defines the height of the object	Int	R
ControlHide	Defines whether the object should be visible (0) or invisible (1). Modifying this command provokes an immediate redrawing without needing to invoke the Draw method.	Bool	RW
BorderColor	Defines the color of the border of the object currently being redrawn. Can be changed by attributing an RGB (Long) value returned, for example, by the RGB function (e.g. BorderColor=RGB(24,255,0). The change will appear in runtime after invoking the Draw method.	RGB	RW
BorderBlink	Defines whether the border of the object should blink or not. Possible values of this property are: 0 - No blinking 1 - Slow blinking 2 - Rapid blinking If a different value from the preceding ones is attributed, the property is forced at 0. Modifying this command provokes an immediate redrawing without needing to invoke the Draw method.	Int	RW
AreaColor	Defines the color of the internal area of the object currently being redrawn. Can be changed by attributing an RGB (Long) value returned, for example, by the RGB function (e.g. BorderColor=RGB(24,255,0). The change will appear in runtime after invoking the Draw method.	RGB	RW

Table 19: Properties of ESACNTRL - Ellipse

Properties	Description	Type	RW
FillDir	Defines infill direction of the object currently being redrawn; the values may be as follows: 0 - From bottom to top 1 - From top to bottom 2 - From left to right 3 - From right to left If a different value from the preceding ones is attributed, the property is forced to 0. The change is shown in runtime after the Draw method is invoked.	Int	RW
FillPercent	Defines the percentage infill of the object currently being redrawn. The change is shown in runtime after the Draw method is invoked.	Int	RW
FillColor	Defines the infill color of the object currently being redrawn. Can be changed by attributing an RGB (Long) value returned, for example, by the RGB function (e.g. BorderColor= RGB (24,255,0). The change is shown in runtime after the Draw method is invoked.	RGB	RW

Methods of ESACNTRL - Ellipse

Table 20: Methods of ESACNTRL - Ellipse

Method	Description	OUT	IN
Draw	Redraws the whole object from the beginning, updating all the graphic properties that were changed.	-	-

Properties of ESACNTRL - Arc

Table 21: Properties of ESACNTRL - Arc

Properties	Description	Typ e	RW
ControlLeft	Defines the position (in pixels) counting from the left where the object has currently been drawn. If this value is changed, the object is moved horizontally (when redrawn with the Draw method).	Int	RW
ControlTop	Defines the position (in pixels) counting from the top where the object has currently been drawn. If this value is changed, the object is moved vertically (when redrawn with the Draw method).	Int	RW
ControlWidth	Defines the width of the object	Int	R
ControlHeight	Defines the height of the object	Int	R
ControlHide	Defines whether the object should be visible (0) or invisible (1). Modifying this command provokes an immediate redrawing without needing to invoke the Draw method.	Bool	RW
ArcColor	Defines the color of the border of the arc currently being redrawn. Can be changed by attributing an RGB (Long) value returned, for example, by the RGB function (e.g. BorderColor= RGB (24,255,0). The change will appear in runtime after invoking the Draw method.	RGB	RW

Table 21: Properties of ESACNTRL - Arc

Properties	Description	Typ e	RW
ArcBlink	Defines whether the arc should blink or not. Possible values of this property are: 0 - No blinking 1 - Slow blinking 2 - Rapid blinking If a different value from the preceding ones is attributed, the property is forced to 0. Modifying this command provokes an immediate redrawing without needing to invoke the Draw method.	Int	RW

Methods of ESACNTRL - Arc

Table 22: Methods of ESACNTRL - Arc

method	Description	OUT	IN
Draw	Redraws the whole object from the beginning, updating all the graphic properties that were changed.	-	-

Properties of ESACNTRL - Circular sector

Table 23: Properties of ESACNTRL - Circular sector

Properties	Description	Typ e	RW
ControlLeft	Defines the position (in pixels) counting from the left where the object has currently been drawn. If this value is changed, the object is moved horizontally (when redrawn with the Draw method).	Int	RW

Table 23: Properties of ESACNTRL - Circular sector

Properties	Description	Typ e	RW
ControlTop	Defines the position (in pixels) counting from the top where the object has currently been drawn. If this value is changed, the object is moved horizontally (when redrawn with the Draw method).	Int	RW
ControlWidth	Defines the width of the object	Int	R
ControlHeight	Defines the height of the object	Int	R
ControlHide	Defines whether the object should be visible (0) or invisible (1). Modifying this command provokes an immediate redrawing without needing to invoke the Draw method.	Bool	RW
ArcColor	Defines the color of the arc of the circular section currently being redrawn. Can be changed by attributing an RGB (Long) value returned, for example, by the RGB function (e.g. BorderColor= RGB (24,255,0). The change will appear in runtime after invoking the Draw method.	RGB	RW
ArcBlink	Defines whether the arc of the circular section should blink or not. Possible values of this property are: 0 - No blinking 1 - Slow blinking 2 - Rapid blinking If a different value from the preceding ones is attributed, the property is forced to 0. Modifying this command provokes an immediate redrawing without needing to invoke the Draw method.	Int	RW

Table 23: Properties of ESACNTRL - Circular sector

Properties	Description	Typ e	RW
AreaColor	Defines the color of the internal area of the object currently being redrawn. Can be changed by attributing an RGB (Long) value returned, for example, by the RGB function (e.g. BorderColor=RGB(24,255,0). The change will appear in runtime after invoking the Draw method.	RGB	RW
FillDir	Defines infill direction of the object currently being redrawn; the values may be as follows: 0 - From bottom to top 1 - From top to bottom 2 - From left to right 3 - From right to left If a different value from the preceding ones is attributed, the property is forced to 0. The change is shown in runtime after the Draw method is invoked.	Int	RW
FillPercent	Defines the percentage infill of the object currently being redrawn. The change is shown in runtime after the Draw method is invoked.	Int	RW
FillColor	Defines the infill color of the object currently being redrawn. Can be changed by attributing an RGB (Long) value returned, for example, by the RGB function (e.g. FillColor= RGB (24,255,0). The change is shown in runtime after the Draw method is invoked.	RGB	RW

Methods of ESACNTRL - Circular sector

Table 24: Methods of ESACNTRL - Circular sector

Method	Description	OUT	IN
Draw	Redraws the whole object from the beginning, updating all the graphic properties that were changed.	-	-

Properties of ESACNTRL - Line

Table 25: Properties of ESACNTRL - Line

Properties	Description	Type	RW
ControlLeft	Defines the position (in pixels) counting from the left where the object has currently been drawn. If this value is changed, the object is moved horizontally (when redrawn with the Draw method).	Int	RW
ControlTop	Defines the position (in pixels) counting from the top where the object has currently been drawn. If this value is changed, the object is moved vertically (when redrawn with the Draw method).	Int	RW
X1	Horizontal coordinate of the starting point. Changing this value means moving the starting point horizontally (when redrawing using the Draw method). This value, if read with a Script, assumes a value of X1-Left (values set with POLYMATH). Similarly, the point is drawn on the pixel with the value X1+Left (Script values).	Int	R

Table 25: Properties of ESACNTRL - Line

Properties	Description	Typ e	RW
X2	Horizontal coordinate of the arrival point. Changing this value means moving the arrival point horizontally (when redrawing using the Draw method). This value, if read with a Script, assumes a value of X2-Left (values set with POLYMATH). Similarly, the point is drawn on the pixel with the value X2+Left (Script values).	Int	R
Y1	Vertical coordinate of the starting point. Changing this value means moving the starting point vertically (when redrawing using the Draw method). This value, if read with a Script, assumes a value of Y1-Top (values set with POLYMATH). Similarly, the point is drawn on the pixel with the value Y1+Top (Script values).	Int	R
Y2	Vertical coordinate of the arrival point. Changing this value means moving the arrival point vertically (when redrawing using the Draw method). This value, if read with a Script, assumes a value of Y2-Top (values set with POLYMATH). Similarly, the point is drawn on the pixel with the value Y2+Top (Script values).	Int	R
ControlHide	Defines whether the object should be visible (0) or invisible (1). Modifying this command provokes an immediate redrawing without needing to invoke the Draw method.	Bool	RW

Table 25: Properties of ESACNTRL - Line

Properties	Description	Typ e	RW
LineColor	Defines the color of the line currently being redrawn. Can be changed by attributing an RGB (Long) value returned, for example, by the RGB function (e.g. BorderColor= RGB (24,255,0). The change will appear in runtime after invoking the Draw method.	RGB	RW
LineBlink	Defines whether the line should blink or not. Possible values of this property are: 0 - No blinking 1 - Slow blinking 2 - Rapid blinking If a different value from the preceding ones is attributed, the property is forced to 0. Modifying this command provokes an immediate redrawing without needing to invoke the Draw method.	Int	RW

Methods of ESACNTRL - Line

Table 26: Methods of ESACNTRL - Line

Method	Description	OUT	IN
Draw	Redraws the whole object from the beginning, updating all the graphic properties that were changed.	-	-

Properties of ESACNTRL - Polygon

Table 27: Properties of ESACNTRL - Polygon

Properties	Description	Typ e	RW
ControlLeft	Defines the position (in pixels) counting from the left where the object has currently been drawn (that is, the rectangle containing it). If this value is changed, the object is moved horizontally (when redrawn with the Draw method).	Int	RW
ControlTop	Defines the position (in pixels) counting from the top where the object (or the rectangle containing it) has currently been drawn. If this value is changed, the object is moved vertically (when redrawn with the Draw method).	Int	RW
ControlHide	Defines whether the object should be visible (0) or invisible (1). Modifying this command provokes an immediate redrawing without needing to invoke the Draw method.	Bool	RW
LineColor	Defines the color of the outline of the polygon currently being redrawn. Can be changed by attributing an RGB (Long) value returned, for example, by the RGB function (e.g. BorderColor= RGB (24,255,0). The change will appear in runtime after invoking the Draw method.	RGB	RW
AreaColor	Defines the color of the internal area of the object currently being redrawn. Can be changed by attributing an RGB (Long) value returned, for example, by the RGB function (e.g. BorderColor=RGB(24,255,0). The change will appear in runtime after invoking the Draw method.	RGB	RW

Table 27: Properties of ESACNTRL - Polygon

Properties	Description	Typ e	RW
FillDir	Defines infill direction of the object currently being redrawn; the values may be as follows: 0 - From bottom to top 1 - From top to bottom 2 - From left to right 3 - From right to left If a different value from the preceding ones is attributed, the property is forced to 0. The change is shown in runtime after the Draw method is invoked.	Int	RW
FillPercent	Defines the percentage infill of the object currently being redrawn. The change is shown in runtime after the Draw method is invoked.	Int	RW
FillColor	Defines the infill color of the object currently being redrawn. Can be changed by attributing an RGB (Long) value returned, for example, by the RGB function (e.g. FillColor= RGB (24,255,0). The change is shown in runtime after the Draw method is invoked.	RGB	RW

Methods of ESACNTRL - Polygon

Table 28: Methods of ESACNTRL - Polygon

Method	Description	OUT	IN
Draw	Redraws the whole object from the beginning, updating all the graphic properties that were changed.	-	-

Properties of ESACNTRL - Irregular line

Table 29: Properties of ESACNTRL - Irregular line

Properties	Description	Typ e	RW
ControlLeft	Defines the position (in pixels) counting from the left where the object has currently been drawn (or the rectangle containing it). If this value is changed, the object is moved horizontally (when redrawn with the Draw method).	Int	RW
ControlTop	Defines the position (in pixels) counting from the top where the object (or the rectangle containing it) has currently been drawn. If this value is changed, the object is moved vertically (when redrawn with the Draw method).	Int	RW
ControlHide	Defines whether the object should be visible (0) or invisible (1). Modifying this command provokes an immediate redrawing without needing to invoke the Draw method.	Bool	RW
LineColor	Defines the color of the outline of the polygon currently being redrawn. Can be changed by attributing an RGB (Long) value returned, for example, by the RGB function (e.g. BorderColor= RGB (24,255,00). The change will appear in runtime after invoking the Draw method.	RGB	RW

Methods of ESACNTRL - Broken line

Table 30: Methods of ESACNTRL - Broken line

Method	Description	OUT	IN
Draw	Redraws the whole object from the beginning, updating all the graphic properties that were changed.	-	-

Properties of ESACNTRL - Regular polygon

The properties and methods of the regular polygon coincide with those of the polygon drawn by the user as already described (see chap. 9, “Properties of ESACNTRL - Polygon” page 423 and see chap. 9, “Methods of ESACNTRL - Polygon” page 424).

Properties of ESACNTRL - Label

Table 31: Properties of ESACNTRL - Label

Properties	Description	Type	RW
ControlLeft	Defines the position (in pixels) counting from the left where the rectangle of the label has currently been drawn. If this value is changed, the object is moved horizontally (when redrawn with the Draw method).	Int	RW
ControlTop	Defines the position (in pixels) counting from the top where the object (or the rectangle of the label) has currently been drawn. If this value is changed, the object is moved vertically (when redrawn with the Draw method).	Int	RW
ControlWidth	Defines the width of the object	Int	R
ControlHeight	Defines the height of the object	Int	R
ControlHide	Defines whether the object should be visible (0) or invisible (1). Modifying this command provokes an immediate redrawing without needing to invoke the Draw method.	Bool	RW

Table 31: Properties of ESACNTRL - Label

Properties	Description	Typ e	RW
BorderColor	Defines the color of the border of the rectangle of the label currently being redrawn. Can be changed by attributing an RGB (Long) value returned, for example, by the RGB function (e.g. BorderColor=RGB(24,255,0). The change will appear in runtime after invoking the Draw method.	RGB	RW
BorderBlink	Defines whether the border of the rectangle of the label should blink or not. Possible values of this property are: 0 - No blinking 1 - Slow blinking 2 - Rapid blinking If a different value from the preceding ones is attributed, the property is forced to 0. Modifying this command provokes an immediate redrawing without needing to invoke the Draw method.	Int	RW
AreaColor	Defines the color of the internal area of the label currently being redrawn. Can be changed by attributing an RGB (Long) value returned, for example, by the RGB function (e.g. BorderColor=RGB(24,255,0). The change will appear in runtime after invoking the Draw method.	RGB	RW
TextValue	Defines the value of the text currently written on the label. Can be varied by providing a new string and the on screen update happens after the Draw method has been invoked.	Str	RW

Table 31: Properties of ESACNTRL - Label

Properties	Description	Typ e	RW
TextColor	Defines the color of the text currently being written on the label. Can be changed by attributing an RGB (Long) value returned, for example, by the RGB function (e.g. BorderColor=RGB(24,255,0). The change will appear in runtime after invoking the Draw method.	RGB	RW
TextBlink	Defines whether the text of the label should blink or not. Possible values of this property are: 0 - No blinking 1 - Slow blinking 2 - Rapid blinking If a different value from the preceding ones is attributed, the property is forced to 0. Modifying this command provokes an immediate redrawing without needing to invoke the Draw method.	Int	RW
FontFaceNa me	Defines the font to use for writing the text. Can be edited by inserting the string relating to the name of the Font (one of those included in the project). The change will appear in runtime after Draw method is invoked.	Str	RW
FontSize	Defines the size of the label text. Can be changed by attributing the required value. The change will appear in runtime after Draw method is invoked.	Int	RW
FontItalic	Defines whether the label text is shown in Italics (1) or in Roman (0). If modified using a Script, the variation will appear in runtime after Draw method is invoked.	Bool	RW

Table 31: Properties of ESACNTRL - Label

Properties	Description	Type	RW
FontBold	Defines whether the label text is shown in Bold (1) or in Roman (0). If modified using a Script, the variation will appear in runtime after Draw method is invoked.	Bool	RW
FontUnderline	Defines whether the label text is shown underlined (1) or normal (0). If modified using a Script, the variation will appear in runtime after Draw method is invoked.	Bool	RW
FontStrikeOut	Defines whether the label text is shown barred (1) or normal (0). If modified using a Script, the variation will appear in runtime after Draw method is invoked.	Bool	RW

Methods of ESACNTRL - Label

Table 32: Methods of ESACNTRL - Label

Method	Description	OUT	IN
Draw	Redraws the whole object from the beginning, updating all the graphic properties that were changed.	-	-
GetTextLen	Returns the length of the string currently written in the label.	Int	-

Properties of ESACNTRL - Image field

Table 33: Properties of ESACNTRL - Image field

Properties	Description	Typ e	RW
ControlLeft	Defines the position (in pixels) counting from the left where the rectangle of the field has currently been drawn. If this value is changed, the object is moved horizontally (when redrawn with the Draw method).	Int	RW
ControlTop	Defines the position (in pixels) counting from the top where the object (or the rectangle of the field) has currently been drawn. If this value is changed, the object is moved vertically (when redrawn with the Draw method).	Int	RW
ControlWidth	Defines the width of the object	Int	R
ControlHeight	Defines the height of the object	Int	R
ControlHide	Defines whether the object should be visible (0) or invisible (1). Modifying this command provokes an immediate redrawing without needing to invoke the Draw method.	Bool	RW
BorderColor	Defines the color of the border of the rectangle of the field currently being redrawn. Can be changed by attributing an RGB (Long) value returned, for example, by the RGB function (e.g. BorderColor=RGB(24,255,0). The change will appear in runtime after invoking the Draw method.	RGB	RW

Table 33: Properties of ESACNTRL - Image field

Properties	Description	Typ e	RW
BorderBlink	Defines whether the border of the rectangle of the field should blink or not. Possible values of this property are: 0 - No blinking 1 - Slow blinking 2 - Rapid blinking If a different value from the preceding ones is attributed, the property is forced to 0. Modifying this command provokes an immediate redrawing without needing to invoke the Draw method.	Int	RW
AreaColor	Defines the color of the internal area of the field being drawn. Can be changed by attributing an RGB (Long) value returned, for example, by the RGB function (e.g. BorderColor=RGB(24,255,0). The change will appear in runtime after invoking the Draw method.	RGB	RW

Methods of ESACNTRL - Image field

Table 34: Methods of ESACNTRL - Image field

Method	Description	OUT	IN
Draw	Redraws the whole object from the beginning, updating all the graphic properties that were changed.	-	-
GetHorDim	Returns the original value of the horizontal dimension of the image currently displayed within the symbol field.	Int	-

Table 34: Methods of ESACNTRL - Image field

Method	Description	OUT	IN
GetVertDIm	Returns the original value of the vertical dimension of the image currently displayed within the symbol field.	Int	-

Properties of ESACNTRL - Numerical field

Table 35: Properties of ESACNTRL - Numerical field

Properties	Description	Typ e	RW
ControlLeft	Defines the position (in pixels) counting from the left where the rectangle of the field has currently been drawn. If this value is changed, the object is moved horizontally (when redrawn with the Draw method).	Int	RW
ControlTop	Defines the position (in pixels) counting from the top where the object (or the rectangle of the field) has currently been drawn. If this value is changed, the object is moved vertically (when redrawn with the Draw method).	Int	RW
ControlWidt h	Defines the width of the object	Int	R
ControlHeig ht	Defines the height of the object	Int	R
ControlHide	Defines whether the object should be visible (0) or invisible (1). Modifying this command provokes an immediate redrawing without needing to invoke the Draw method.	Bool	RW

Table 35: Properties of ESACNTRL - Numerical field

Properties	Description	Type	RW
BorderColor	Defines the color of the border of the rectangle of the field currently being redrawn. Can be changed by attributing an RGB (Long) value returned, for example, by the RGB function (e.g. BorderColor=RGB(24,255,0). The change will appear in runtime after invoking the Draw method.	RGB	RW
BorderBlink	Defines whether the border of the rectangle of the field should blink or not. Possible values of this property are: 0 - No blinking 1 - Slow blinking 2 - Rapid blinking If a different value from the preceding ones is attributed, the property is forced to 0. Modifying this command provokes an immediate redrawing without needing to invoke the Draw method.	Int	RW
AreaColor	Defines the color of the internal area of the field currently being drawn. Can be changed by attributing an RGB (Long) value returned, for example, by the RGB function (e.g. BorderColor=RGB(24,255,0). The change will appear in runtime after invoking the Draw method.	RGB	RW
FontFaceName	Defines the font to use for writing the text. Can be edited by inserting the string relating to the name of the Font (one of those included in the project). The change will appear in runtime after Draw method is invoked.	Str	RW
FontSize	Defines the size of the field text. Can be changed by attributing the required value. The change will appear in runtime after Draw method is invoked.	Int	RW

Table 35: Properties of ESACNTRL - Numerical field

Properties	Description	Type	RW
FontItalic	Defines whether the field text is shown in Italics (1) or in Roman (0). If modified using a Script, the variation will appear in runtime after Draw method is invoked.	Bool	RW
FontBold	Defines whether the field text is shown in Bold (1) or in Roman (0). If modified using a Script, the variation will appear in runtime after Draw method is invoked.	Bool	RW
FontUnderline	Defines whether the field text is shown underlined (1) or in normal (0). If modified using a Script, the variation will appear in runtime after Draw method is invoked.	Bool	RW
FontStrikeOut	Defines whether the field text is shown barred (1) or in normal (0). If modified using a Script, the variation will appear in runtime after Draw method is invoked.	Bool	RW
Disable	Defines whether the field is enabled (0) or disabled (1), that is, whether the user can edit it. Editing this property provokes immediate redrawing without needing to invoke the Draw method.	Bool	RW
ValueColor	Defines the color of the current text contained in the field. Can be changed by attributing an RGB (Long) value returned, for example, by the RGB function (e.g. BorderColor=RGB(24,255,0). The change will appear in runtime after invoking the Draw method.	RGB	RW

Table 35: Properties of ESACNTRL - Numerical field

Properties	Description	Typ e	RW
ValueBlink	Defines whether the current field text should blink or not. Possible values of this property are: 0 - No blinking 1 - Slow blinking 2 - Rapid blinking If a different value from the preceding ones is attributed, the property is forced to 0. Modifying this command provokes an immediate redrawing without needing to invoke the Draw method.	Int	RW
Value	Defines the value of the text currently written onto the field. Can be varied by providing a new string and the on screen update happens after the Draw method or Refresh Control is invoked.	Var	RW

Methods of ESACNTRL - Numerical field

Table 36: Methods of ESACNTRL - Numerical field

Method	Description	OUT	IN
Draw	Redraws the whole object from the beginning, updating all the graphic properties that were changed.	-	-
RefreshCont rol	Redraws only the part of the field relating to the value shown, leaving out the graphic aspects of the field. This function is preferable to Draw when all that is needed is a refresh of the value (it is quicker).	-	-

Properties of ESACNTRL - Dynamic text

Table 37: Properties of ESACNTRL - Dynamic text

Properties	Description	Typ e	RW
ControlLeft	Defines the position (in pixels) counting from the left where the rectangle of the field has currently been drawn. If this value is changed, the object is moved horizontally (when redrawn with the Draw method).	Int	RW
ControlTop	Defines the position (in pixels) counting from the top where the object (or the rectangle of the field) has currently been drawn. If this value is changed, the object is moved vertically (when redrawn with the Draw method).	Int	RW
ControlWidt h	Defines the width of the object	Int	R
ControlHeig ht	Defines the height of the object	Int	R
ControlHide	Defines whether the object should be visible (0) or invisible (1). Modifying this command provokes an immediate redrawing without needing to invoke the Draw method.	Bool	RW
BorderColor	Defines the color of the border of the rectangle of the field currently being redrawn. Can be changed by attributing an RGB (Long) value returned, for example, by the RGB function (e.g. BorderColor=RGB(24,255,0). The change will appear in runtime after invoking the Draw method.	RGB	RW

Table 37: Properties of ESACNTRL - Dynamic text

Properties	Description	Typ e	RW
BorderBlink	Defines whether the border of the rectangle of the field should blink or not. Possible values of this property are: 0 - No blinking 1 - Slow blinking 2 - Rapid blinking If a different value from the preceding ones is attributed, the property is forced to 0. Modifying this command provokes an immediate redrawing without needing to invoke the Draw method.	Int	RW
AreaColor	Defines the color of the internal area of the field currently being drawn. Can be changed by attributing an RGB (Long) value returned, for example, by the RGB function (e.g. BorderColor=RGB(24,255,0). The change will appear in runtime after invoking the Draw method.	RGB	RW
TextColor	Defines the color of the text currently contained in the field. Can be changed by attributing an RGB (Long) value returned, for example, by the RGB function (e.g. BorderColor= RGB (24,255,0). The change will appear in runtime after invoking the Draw method.	RGB	RW

Table 37: Properties of ESACNTRL - Dynamic text

Properties	Description	Typ e	RW
TextBlink	Defines whether the current field text should blink or not. Possible values of this property are: 0 - No blinking 1 - Slow blinking 2 - Rapid blinking If a different value from the preceding ones is attributed, the property is forced to 0. Modifying this command provokes an immediate redrawing without needing to invoke the Draw method.	Int	RW
FontFaceNa me	Defines the font to use for writing the text. Can be edited by inserting the string relating to the name of the Font (one of those included in the project). The change will appear in runtime after Draw method is invoked.	Str	RW
FontSize	Defines the size of the field text. Can be changed by attributing the required value. The change will appear in runtime after Draw method is invoked.	Int	RW
FontItalic	Defines whether the field text is shown in Italics (1) or in Roman (0). If modified using a Script, the variation will appear in runtime after Draw method is invoked.	Bool	RW
FontBold	Defines whether the field text is shown in Bold (1) or in Roman (0). If modified using a Script, the variation will appear in runtime after Draw method is invoked.	Bool	RW
FontUnderli ne	Defines whether the field text is shown underlined (1) or in normal (0). If modified using a Script, the variation will appear in runtime after Draw method is invoked.	Bool	RW

Table 37: Properties of ESACNTRL - Dynamic text

Properties	Description	Type	RW
FontStrikeOut	Defines whether the field text is shown barred (1) or in normal (0). If modified using a Script, the variation will appear in runtime after Draw method is invoked.	Bool	RW
Disable	Defines whether the field is enabled (0) or disabled (1), that is, whether the user can edit it. Editing this property provokes immediate redrawing without needing to invoke the Draw method.	Bool	RW
Value	Defines the value of the text currently written onto the field. Can be varied by providing a new string and the on screen update happens after the Draw method or Refresh Control is invoked.	Var	RW

Methods of ESACNTRL - Dynamic text

Table 38: Methods of ESACNTRL - Dynamic text

Method	Description	OUT	IN
Draw	Redraws the whole object from the beginning, updating all the graphic properties that were changed.	-	-
RefreshControl	Redraws only the part of the field relating to the value shown, leaving out the graphic aspects of the field. This function is preferable to Draw when all that is needed is a refresh of the value (it is quicker).	-	-
GetTextLen	Returns the length of the string currently written into the field	Int	-

Properties of ESACNTRL - ASCII field

The properties of the ASCII field accessible using Scripts coincide with those of the Numerical field (see chap. 9, “Properties of ESACNTRL - Numerical field” page 432).

Methods of ESACNTRL - ASCII field

Table 39: Methods of ESACNTRL - ASCII field

Method	Description	OUT	IN
Draw	Redraws the whole object from the beginning, updating all the graphic properties that were changed.	-	-
RefreshControl	Redraws only the part of the field relating to the value shown, leaving out the graphic aspects of the field. This function is preferable to Draw when all that is needed is a refresh of the value (it is quicker).	-	-
GetTextLen	Returns the length of the string currently written into the field	Int	-

Properties of ESACNTRL - Symbol field

Table 40: Properties of ESACNTRL - Symbol field

Properties	Description	Type	RW
ControlLeft	Defines the position (in pixels) counting from the left where the rectangle of the field has currently been drawn. If this value is changed, the object is moved horizontally (when redrawn with the Draw method).	Int	RW

Table 40: Properties of ESACNTRL - Symbol field

Properties	Description	Typ e	RW
ControlTop	Defines the position (in pixels) counting from the top where the object (or the rectangle of the field) has currently been drawn. If this value is changed, the object is moved vertically (when redrawn with the Draw method).	Int	RW
ControlWidt h	Defines the width of the rectangle containing the object	Int	R
ControlHeig ht	Defines the height of the rectangle containing the object	Int	R
ControlHide	Defines whether the object should be visible (0) or invisible (1). Modifying this command provokes an immediate redrawing without needing to invoke the Draw method.	Bool	RW
BorderColor	Defines the color of the border of the rectangle of the field currently being redrawn. Can be changed by attributing an RGB (Long) value returned, for example, by the RGB function (e.g. BorderColor=RGB(24,255,0). The change will appear in runtime after invoking the Draw method.	RGB	RW
BorderBlink	Defines whether the border of the rectangle of the field should blink or not. Possible values of this property are: 0 - No blinking 1 - Slow blinking 2 - Rapid blinking If a different value from the preceding ones is attributed, the property is forced to 0. Modifying this command provokes an immediate redrawing without needing to invoke the Draw method.	Int	RW

Table 40: Properties of ESACNTRL - Symbol field

Properties	Description	Type	RW
AreaColor	Defines the color of the internal area of the field currently being drawn. Can be changed by attributing an RGB (Long) value returned, for example, by the RGB function (e.g. BorderColor=RGB(24,255,0). The change will appear in runtime after invoking the Draw method.	RGB	RW
Disable	Defines whether the field is enabled (0) or disabled (1), that is, whether the user can edit it. Editing this property provokes immediate redrawing without needing to invoke the Draw method.	Bool	RW
Value	Defines the value the symbol field refers to. Can be varied by providing a new string and the on screen update happens after the Draw method or Refresh Control is invoked.	Var	RW

Methods of ESACNTRL - Symbol field

Table 41: Methods of ESACNTRL - Symbol field

Method	Description	OUT	IN
Draw	Redraws the whole object from the beginning, updating all the graphic properties that were changed.	-	-
RefreshControl	Redraws only the part of the field relating to the value shown, leaving out the graphic aspects of the field. This function is preferable to Draw when all that is needed is a refresh of the value (it is quicker).	-	-

Table 41: Methods of ESACNTRL - Symbol field

Method	Description	OUT	IN
GetHorDim	Returns the original value of the horizontal dimension of the image currently displayed inside the Symbol field.	Int	-
GetVertDIm	Returns the original value of the vertical dimension of the image currently displayed inside the Symbol field.	Int	-

Properties of ESACNTRL - DateTime field

The properties of the DateTime field accessible using Scripts coincide with those of the Numerical field (see chap. 9, “Properties of ESACNTRL - Numerical field” page 432).

Methods of ESACNTRL - DateTime field

Table 42: Methods of ESACNTRL - DateTime

Method	Description	OUT	IN
Draw	Redraws the whole object from the beginning, updating all the graphic properties that were changed.	-	-
RefreshControl	Redraws only the part of the field relating to the value shown, leaving out the graphic aspects of the field. This function is preferable to Draw when all that is needed is a refresh of the value (it is quicker).	-	-
GetTextLen	Returns the length of the string currently written into the field.	Int	-

Properties of ESACNTRL - Bar

Table 43: Properties of ESACNTRL - Bar

Properties	Description	Type	RW
ControlLeft	Defines the position (in pixels) counting from the left where the rectangle containing the bar has currently been drawn. If this value is changed, the object is moved horizontally (when redrawn with the Draw method).	Int	RW
ControlTop	Defines the position (in pixels) counting from the top where the object (or the rectangle containing the bar) has currently been drawn. If this value is changed, the object is moved vertically (when redrawn with the Draw method).	Int	RW
ControlWidth	Defines the width of the rectangle containing the object	Int	R
ControlHeight	Defines the height of the rectangle containing the object	Int	R
ControlHide	Defines whether the object should be visible (0) or invisible (1). Modifying this command provokes an immediate redrawing without needing to invoke the Draw method.	Bool	RW
BorderColor	Defines the color of the border of the rectangle containing the bar currently being redrawn. Can be changed by attributing an RGB (Long) value returned, for example, by the RGB function (e.g. BorderColor=RGB(24,255,0). The change will appear in runtime after invoking the Draw method.	RGB	RW

Table 43: Properties of ESACNTRL - Bar

Properties	Description	Typ e	RW
BorderBlink	Defines whether the border of the rectangle containing the bar should blink or not. Possible values of this property are: 0 - No blinking 1 - Slow blinking 2 - Rapid blinking If a different value from the preceding ones is attributed, the property is forced to 0. Modifying this command provokes an immediate redrawing without needing to invoke the Draw method.	Int	RW
AreaColor	Defines the color of the internal area of the bar currently being drawn. Can be changed by attributing an RGB (Long) value returned, for example, by the RGB function (e.g. BorderColor=RGB(24,255,0). The change will appear in runtime after invoking the Draw method.	RGB	RW
IndicatorColor	Defines the color of the indicator used in the bar currently being redrawn. Can be changed by attributing an RGB (Long) value returned, for example, by the RGB function (e.g. AreaColor= RGB (24,255,0). The change will appear in runtime after invoking the Draw method.	RGB	RW
Value	Defines the value the bar refers to. Editing this property provokes immediate redrawing without needing to invoke the Draw method.	Var	RW
Disable	Defines whether the field is enabled (0) or disabled (1), that is, whether the user can edit it. Editing this property provokes immediate redrawing without needing to invoke the Draw method.	Bool	RW

Methods of ESACNTRL - Bar

Table 44: Methods of ESACNTRL - Bar

Method	Description	OUT	IN
Draw	Redraws the whole object from the beginning, updating all the graphic properties that were changed.	-	-
RefreshControl	Redraws only the part of the field relating to the value shown, leaving out the graphic aspects of the field. This function is preferable to Draw when all that is needed is a refresh of the value (it is quicker).	-	-

Properties of ESACNTRL - Indicator

Table 45: Properties of ESACNTRL - Indicator

Properties	Description	Type	RW
ControlLeft	Defines the position (in pixels) counting from the left where the rectangle containing the indicator has currently been drawn. If this value is changed, the object is moved horizontally (when redrawn with the Draw method).	Int	RW
ControlTop	Defines the position (in pixels) counting from the top where the object (or the rectangle containing the indicator) has currently been drawn. If this value is changed, the object is moved vertically (when redrawn with the Draw method).	Int	RW
ControlWidth	Defines the width of the rectangle containing the object	Int	R
ControlHeight	Defines height of the rectangle containing the object	Int	R

Table 45: Properties of ESACNTRL - Indicator

Properties	Description	Typ e	RW
ControlHide	Defines whether the object should be visible (0) or invisible (1). Modifying this command provokes an immediate redrawing without needing to invoke the Draw method.	Bool	RW
BorderColor	Defines the color of the border of the rectangle containing the indicator currently being redrawn. Can be changed by attributing an RGB (Long) value returned, for example, by the RGB function (e.g. BorderColor= RGB (24,255,0). The change will appear in runtime after invoking the Draw method.	RGB	RW
BorderBlink	Defines whether the border of the rectangle containing the indicator should blink or not. Possible values of this property are: 0 - No blinking 1 - Slow blinking 2 - Rapid blinking If a different value from the preceding ones is attributed, the property is forced to 0. Modifying this command provokes an immediate redrawing without needing to invoke the Draw method.	Int	RW
AreaColor	Defines the color of the internal area of the bar currently being drawn. Can be changed by attributing an RGB (Long) value returned, for example, by the RGB function (e.g. BorderColor=RGB(24,255,0). The change will appear in runtime after invoking the Draw method.	RGB	RW
Value	Defines the value the indicator refers to. Editing this property provokes immediate redrawing without needing to invoke the Draw method.	Var	RW

Methods of ESACNTRL - Indicator

Table 46: Methods of ESACNTRL - Indicator

Method	Description	OUT	IN
Draw	Redraws the whole object from the beginning, updating all the graphic properties that were changed.	-	-
RefreshControl	Redraws only the part of the field relating to the value shown, leaving out the graphic aspects of the field. This function is preferable to Draw when all that is needed is a refresh of the value (it is quicker).	-	-

Properties of ESACNTRL - Touch button

Table 47: Properties of ESACNTRL - Touch button

Properties	Description	Type	RW
ControlLeft	Defines the position (in pixels) counting from the left where the button has currently been drawn. If this value is changed, the object is moved horizontally (when redrawn with the Draw method).	Int	RW
ControlTop	Defines the position (in pixels) counting from the left where the button has currently been drawn. If this value is changed, the object is moved vertically (when redrawn with the Draw method).	Int	RW
ControlWidth	Defines the width of the button.	Int	R
ControlHeight	Defines the height of the button.	Int	R

Table 47: Properties of ESACNTRL - Touch button

Properties	Description	Typ e	RW
ControlHide	Defines whether the object should be visible (0) or invisible (1). Modifying this command provokes an immediate redrawing without needing to invoke the Draw method.	Bool	RW
BorderColor	Defines the color of the border of the button currently being redrawn. Can be changed by attributing an RGB (Long) value returned, for example, by the RGB function (e.g. BorderColor=RGB(24,255,0). The change will appear in runtime after invoking the Draw method.	RGB	RW
BorderBlink	Defines whether the border of the button should blink or not. Possible values of this property are: 0 - No blinking 1 - Slow blinking 2 - Rapid blinking If a different value from the preceding ones is attributed, the property is forced to 0. Modifying this command provokes an immediate redrawing without needing to invoke the Draw method.	Int	RW
AreaColor	Defines the color of the internal area of the button currently being drawn. Can be changed by attributing an RGB (Long) value returned, for example, by the RGB function (e.g. BorderColor=RGB(24,255,0). The change will appear in runtime after invoking the Draw method.	RGB	RW

Table 47: Properties of ESACNTRL - Touch button

Properties	Description	Typ e	RW
Disable	Defines whether the button is enabled (0) or disabled (1), that is, whether the pressing it has an effect or not (for example, the function is executed or the Script corresponding to it). Editing this property provokes immediate redrawing without needing to invoke the Draw method.	Bool	RW
TextLabel	Provides the text of the label	Strin g	RW
TextColor	Provides the colour of the label displayed	Strin g	RW

Methods of ESACNTRL - Touch button

Table 48: Methods of ESACNTRL - Touch button

Method	Description	OUT	IN
Draw	Redraws the whole object from the beginning, updating all the graphic properties that were changed.	-	-

Properties of ESACNTRL - Touch Area

Table 49: Properties of ESACNTRL - Touch area

Properties	Description	Typ e	RW
ControlLeft	Defines the position (in pixels) counting from the left where the area has currently been drawn. If this value is changed, the object is moved horizontally (when redrawn with the Draw method).	Int	RW

Table 49: Properties of ESACNTRL - Touch area

Properties	Description	Type	RW
ControlTop	Defines the position (in pixels) counting from the top where the object (that is, the rectangle containing the bar) has currently been drawn. If this value is changed, the object is moved vertically (when redrawn with the Draw method).	Int	RW
ControlWidth	Defines the width of the currently drawn area	Int	R
ControlHeight	Defines the height of the currently drawn area.	Int	R

Methods of ESACNTRL - Touch area

Table 50: Methods of ESACNTRL - Touch area

Method	Description	OUT	IN
Draw	Redraws the whole object from the beginning, updating all the graphic properties that were changed.	-	-

Properties of ESACNTRL - Slide-Potentiometer

Table 51: Properties of ESACNTRL - Slide-potentiometer

Properties	Description	Type	RW
ControlLeft	Defines the position (in pixels) counting from the left where the button has currently been drawn. If this value is changed, the object is moved horizontally (when redrawn with the Draw method).	Int	RW

Table 51: Properties of ESACNTRL - Slide-potentiometer

Properties	Description	Typ e	RW
ControlTop	Defines the position (in pixels) counting from the top where the potentiometer has currently been drawn. If this value is changed, the object is moved vertically (when redrawn with the Draw method).	Int	RW
ControlWidt h	Defines the width of the object	Int	R
ControlHeig ht	Defines the height of the object	Int	R
ControlHide	Defines whether the object should be visible (0) or invisible (1). Modifying this command provokes an immediate redrawing without needing to invoke the Draw method.	Bool	RW
BorderColor	Defines the color of the border of the rectangle containing the potentiometer currently being redrawn. Can be changed by attributing an RGB (Long) value returned, for example, by the RGB function (e.g. BorderColor=RGB(24,255,0). The change will appear in runtime after invoking the Draw method.	RGB	RW
BorderBlink	Defines whether the border of the rectangle containing the potentiometer should blink or not. Possible values of this property are: 0 - No blinking 1 - Slow blinking 2 - Rapid blinking If a different value from the preceding ones is attributed, the property is forced to 0. Modifying this command provokes an immediate redrawing without needing to invoke the Draw method.	Int	RW

Table 51: Properties of ESACNTRL - Slide-potentiometer

Properties	Description	Typ e	RW
AreaColor	Defines the color of the internal area of the potentiometer currently being drawn. Can be changed by attributing an RGB (Long) value returned, for example, by the RGB function (e.g. BorderColor=RGB(24,255,0). The change will appear in runtime after invoking the Draw method.	RGB	RW
Value	Defines the value represented by the potentiometer. Can be varied by providing a new string and the on screen update happens after the Draw method is invoked.	Var	RW
Disable	Defines whether the field is enabled (0) or disabled (1), that is, whether the user can edit its value. Editing this property provokes immediate redrawing without needing to invoke the Draw method.	Bool	RW

Methods of ESACNTRL - Slide-Potentiometer

Table 52: Methods of ESACNTRL - Slide-potentiometer

Method	Description	OUT	IN
Draw	Redraws the whole object from the beginning, updating all the graphic properties that were changed.	-	-
RefreshCont rol	Redraws only the part of the field relating to the value shown, leaving out the graphic aspects of the field. This function is preferable to Draw when all that is needed is a refresh of the value (it is quicker).	-	-

Properties and Methods of ESACNTRL - Slide-Selector

The properties and methods of the Slide-Selector that can be accessed using Scripts coincide with those of the Slide-Potentiometer (already described, see chap. 9, “Properties of ESACNTRL - Slide-Potentiometer” page 451 and see chap. 9, “Methods of ESACNTRL - Slide-Potentiometer” page 453).

Properties and Methods of ESACNTRL - Knob-Potentiometer

The properties and methods of the Knob-Potentiometer that can be accessed using Scripts coincide with those of the Slide-Potentiometer (already described, see chap. 9, “Properties of ESACNTRL - Slide-Potentiometer” page 451 e see chap. 9, “Methods of ESACNTRL - Slide-Potentiometer” page 453).

Properties and Methods of ESACNTRL - Knob-selector

The properties and methods of the Knob-Selector that can be accessed using Scripts coincide with those of the Slide-Potentiometer (already described, see chap. 9, “Properties of ESACNTRL - Slide-Potentiometer” page 451 e see chap. 9, “Methods of ESACNTRL - Slide-Potentiometer” page 453).

Properties of ESACNTRL - Complex Control Grid

Table 53: Properties of ESACNTRL - Complex Control Grid

Properties	Description	Type	RW
ControlLeft	Defines the position (in pixels) counting from the left where the Grid has currently been drawn. If this value is changed, the object is moved horizontally (when redrawn with the Draw method).	Int	RW
ControlTop	Defines the position (in pixels) counting from the top where the Grid has currently been drawn. If this value is changed, the object is moved vertically (when redrawn with the Draw method).	Int	RW
ControlWidth	Defines the width of the object	Int	R
ControlHeight	Defines the height of the object	Int	R

Table 53: Properties of ESACNTRL - Complex Control Grid

Properties	Description	Typ e	RW
ControlHide	Defines whether the object should be visible (0) or invisible (1). Modifying this command provokes an immediate redrawing without needing to invoke the Draw method.	Bool	RW
BorderColor	Defines the color of the border of the rectangle containing the Grid currently being redrawn. Can be changed by attributing an RGB (Long) value returned, for example, by the RGB function (e.g. BorderColor=RGB(24,255,0). The change will appear in runtime after invoking the Draw method.	RGB	RW
AreaColor	Defines the color of the internal area of the rectangle containing the Grid. Can be changed by attributing an RGB (Long) value returned, for example, by the RGB function (e.g. BorderColor=RGB(24,255,0). The change will appear in runtime after invoking the Draw method.	RGB	RW
Disable	Defines whether the field is enabled (0) or disabled (1), that is, whether the user can edit its values. Editing this property provokes immediate redrawing without needing to invoke the Draw method.	Bool	RW
GridColor	Defines the color of the Grid. Can be changed by attributing an RGB (Long) value returned, for example, by the RGB function (e.g. BorderColor= RGB (24,255,0). The change will appear in runtime after invoking the Draw method.	RGB	RW

Table 53: Properties of ESACNTRL - Complex Control Grid

Properties	Description	Type	RW
RibbonBackColor	Defines the color of the Grid ribbon. Can be changed by attributing an RGB (Long) value returned, for example, by the RGB function (e.g. BorderColor= RGB (24,255,0). The change will appear in runtime after invoking the Draw method.	RGB	RW
RibbonForeColor	Defines the color of the text of the Grid ribbon. Can be changed by attributing an RGB (Long) value returned, for example, by the RGB function (e.g. AreaColor= RGB (24,255,0). The change will appear in runtime after invoking the Draw method.	RGB	RW
SelBackColor	Defines the color of the cell/row selected in the Grid. Can be changed by attributing an RGB (Long) value returned, for example, by the RGB function (e.g. AreaColor= RGB (24,255,0). The change will appear in runtime after invoking the Draw method.	RGB	RW
SelForeColor	Defines the color of the text of the cell/row selected in the Grid. Can be changed by attributing an RGB (Long) value returned, for example, by the RGB function (e.g. BorderColor= RGB (24,255,0). The change will appear in runtime after invoking the Draw method.	RGB	RW
SortCol	Defines the number of columns used to create the order.	Long	RW
SortMode	Defines how to create the order; admissable values are: 0 - ascending 1 - descending	Long	RW

Methods of ESACNTRL - Complex Control Grid

Table 54: Methods of ESACNTRL - Complex Control Grid

Method	Description	OUT	IN
CountColumn	Returns the number of columns in the Grid	Long	-
CountRow	Returns the number of rows in the Grid	Long	-

Properties of ESACNTRL - Trend Graph

Table 55: Properties of ESACNTRL - Trend Graph

Properties	Description	Type	RW
ControlLeft	Defines the position (in pixels) counting from the left where the Grid has currently been drawn. If this value is changed, the object is moved horizontally (when redrawn with the Draw method).	Long	RW
ControlTop	Defines the position (in pixels) counting from the top where the Grid has currently been drawn. If this value is changed, the object is moved vertically (when redrawn with the Draw method).	Long	RW
ControlWidth	Defines the width of the object	Long	R
ControlHeight	Defines the height of the object	Long	R
ControlHide	Defines whether the object should be visible (0) or invisible (1). Modifying this command provokes an immediate redrawing without needing to invoke the Draw method.	Bool	RW

Table 55: Properties of ESACNTRL - Trend Graph

Properties	Description	Type	RW
BorderColor	Defines the color of the border of the rectangle containing the Grid. Can be changed by attributing an RGB (Long) value returned, for example, by the RGB function (e.g. BorderColor= RGB (24,255,0). The change will appear in runtime after invoking the Draw method.	RGB	RW
AreaColor	Defines the color of the internal area of the rectangle containing the Grid. Can be changed by attributing an RGB (Long) value returned, for example, by the RGB function (e.g. BorderColor=RGB(24,255,0). The change will appear in runtime after invoking the Draw method.	RGB	RW
ChartAreaColor	Defines the color of the area of the internal table. Can be changed by attributing an RGB (Long) value returned, for example, by the RGB function (e.g. AreaColor= RGB (24,255,0). The change will appear in runtime after invoking the Draw method.	RGB	RW
ChartBorder Color	Defines the color of the border of the internal table. Can be changed by attributing an RGB (Long) value returned, for example, by the RGB function (e.g. BorderColor= RGB (24,255,0). The change will appear in runtime after invoking the Draw method.	RGB	RW
ChartTimeLeft	Specifies the exact position of the left margin of the table, that is, the start time of the table displayed in the format “DD/MM/ YYYY HH:MM:SS,mmm”	Str	RW

Table 55: Properties of ESACNTRL - Trend Graph

Properties	Description	Typ e	RW
GridHorLine Color	Defines the color of the horizontal Grid. Can be changed by attributing an RGB (Long) value returned, for example, by the RGB function (e.g. AreaColor= RGB (24,255,0). The change will appear in runtime after invoking the Draw method.	RGB	RW
GridHorMinL ineColor	Defines the color of the minimum horizontal Grid. Can be changed by attributing an RGB (Long) value returned, for example, by the RGB function (e.g. AreaColor= RGB (24,255,0). The change will appear in runtime after invoking the Draw method.	RGB	RW
GridVertLine Color	Defines the color of the vertical Grid. Can be changed by attributing an RGB (Long) value returned, for example, by the RGB function (e.g. AreaColor= RGB (24,255,0). The change will appear in runtime after invoking the Draw method.	RGB	RW
GridVertMin LineColor	Defines the color of the minimum vertical Grid. Can be changed by attributing an RGB (Long) value returned, for example, by the RGB function (e.g. AreaColor= RGB (24,255,0). The change will appear in runtime after invoking the Draw method.	RGB	RW
ScaleHorLab elColor	Defines the color of the horizontal scale label. Can be changed by attributing an RGB (Long) value returned, for example, by the RGB function (e.g. AreaColor= RGB (24,255,0). The change will appear in runtime after invoking the Draw method.	RGB	RW
CursorFlag	Defines whether the cursor is visible (1) in the table or not (0).	Bool	RW

Table 55: Properties of ESACNTRL - Trend Graph

Properties	Description	Typ e	RW
CursorColor	Defines the color of the cursor. Can be changed by attributing an RGB (Long) value returned, for example, by the RGB function (e.g. AreaColor= RGB (24,255,0). The change will appear in runtime after invoking the Draw method.	RGB	RW
NumTracks	Defines the number of tracks currently in the table. Available as read-only.	Long	R
ActiveTrack	Specifies the identifying code of the track that is currently active	Long	RW
TrackId	Specifies the identifier of a track. Gives access to the attributes of a specific track.	Long	RW
TrackNumR anges	The number of intervals into which the track values are divided.	Long	RW
TrackRange 1	Defines the limits of the values of interval 1 into which the domain of the track values has been divided. Used only if the attribute TraclNumRanges specifies a sufficient number of intervals.	Dbl	RW
TrackRange 2	Defines the limits of the values of interval 2 into which the domain of the track values has been divided. Used only if the attribute TraclNumRanges specifies a sufficient number of intervals.	Dbl	RW
TrackRange 3	Defines the limits of the values of interval 3 into which the domain of the track values has been divided. Used only if the attribute TraclNumRanges specifies a sufficient number of intervals.	Dbl	RW

Table 55: Properties of ESACNTRL - Trend Graph

Properties	Description	Type	RW
TrackRange 4	Defines the limits of the values of interval 4 into which the domain of the track values has been divided. Used only if the attribute TraclNumRanges specifies a sufficient number of intervals.	Dbl	RW
TrackRange 5	Defines the limits of the values of interval 5 into which the domain of the track values has been divided. Used only if the attribute TraclNumRanges specifies a sufficient number of intervals.	Dbl	RW
TrackColor1	Defines the color of the sample readings and the track lines in relation to the intervals they belong to. This is the standard color used for the track icons and the labels on the vertical scale.	RGB	RW
TrackColor2	Defines the color of the sample readings and the track lines relating to the interval of track number 2.	RGB	RW
TrackColor3	Defines the color of the sample readings and the track lines relating to the interval of track number 3.	RGB	RW
TrackColor4	Defines the color of the sample readings and the track lines relating to the interval of track number 4.	RGB	RW
TrackColor5	Defines the color of the sample readings and the track lines relating to the interval of track number 5.	RGB	RW
TrackColor6	Defines the color of the sample readings and the track lines relating to the interval of track number 6.	RGB	RW
TrackValueLow	Specifies the exact coordinate of the bottom margin of the table.	Long	RW

Table 55: Properties of ESACNTRL - Trend Graph

Properties	Description	Typ e	RW
TrackMaxSamples	The maximum number of tracks that can be inserted in a table buffer. This is a read-only value.	Long	R
TrackNumSamples	Indicates the number of samples currently in the table buffer. This is a read-only value.	Long	R

Methods of ESACNTRL - Trend Graph

Table 56: Methods of ESACNTRL - Trend Graph

Method	Description	OUT	IN
AddTrack	Adds a new track to the table. Requires the passage of the track identifier and the maximum number of samples the buffer will hold.	Long	TrackId (Long) NumSamples (Int)
AddSample	Adds a new sample to the track indicated by the input parameter. Also needs the passage of the value of the sample, the acquisition time and a flag indicating whether the value is valid (1) or not (0). Invalid values (flag=0) are used to specify acquisition errors.	Long	TrackId (Long) Value (Var) Time (Str) Quality (Bool)
RemoveTrack	Removes the track indicated by the identifier passed as an input parameter from the table.	Long	TrackId (Long)
RemoveSamples	Removes all the samples related to the track indicated by the identifier passed as an input parameter.	Long	TrackId (Long)

Table 56: Methods of ESACNTRL - Trend Graph

Method	Description	OUT	IN
GetCursorTrackValue	Returns the value of the track at the position indicated by the cursor. Needs as an input parameter the ID of the track and the value and value-type pointers. The value type is numerical, with the following meanings: 0 - intersection value; 1-sample value; 2 - valid value, but cursor is in cut-off area; -1 non-valid value, the cursor is out of range; -2 non-valid value, the cursor is in a track gap; -3 non-valid value, the cursor is hidden.	Long	TrackId (Long) Value (Var) Result (Long)
GetCursorPosition	Returns the time coordinates of the cursor; functions only if the cursor is active	Long	Time (Str)
SetCursorPosition	Changes cursor time coordinates; functions only if the cursor is active	Long	Time (Str)
MoveUp	Moves display of the table up.	Long	Step (Long)
MoveDown	Moves display of the table down.	Long	Step (Long)
MoveLeft	Moves display of the table leftwards.	Long	Step (Long)
MoveRight	Moves display of the table rightwards.	Long	Step (Long)
Goto	Moves the coordinates of the table to the position indicated by the input parameter.	Long	Time (Str)
Draw	Redraws the table completely.	Long	-

Table 56: Methods of ESACNTRL - Trend Graph

Method	Description	OUT	IN
ChartAlignment	Aligns the contents of the table to the right.	Long	-
RelativeToAbsoluteTime	Changes the display times from Relative to Absolute. Needs as an input parameter the times at which to run the display.	Long	RelTime (Str) AbsTime (Str)
AbsoluteToRelativeTime	Changes the display times from Absolute to Relative. Needs as an input parameter the times at which to run the display.	Long	RelTime (Str) AbsTime (Str)

object ESAPRN

ESAPRN puts at the user's disposal simple functions for printing strings on printers connected to the panel. A print session can be managed by inserting and positioning a variety of texts in the page. The page is printed and released only after the method End has been invoked. This type of printing is, therefore, useful when you need to print data destined to change over time on the same page. In fact, the method Start opens a buffer of elements to be printed that closes only when the method End is invoked.

For a concrete example of the use of the print functions, the reader is advised to consult Example 6 of this chapter (see chap. 9, "Example 6: Creates printout of list of recipes" page 474).

ESAPRN properties accessible with Scripts

Table 57: ESAPRN properties accessible with Scripts

Properties	Description	Type	RW
LastError	Indicates the code of the last error	Int	RW

Table 57: ESAPRN properties accessible with Scripts

Properties	Description	Type	RW
FontSize	Defines (in points) the size of the font in which the strings inserted during the print session will be written. Can be called more than once within the same print session. Moreover, when a value is assigned to this property, the properties PageRows and PageColumns are updated.	Int	RW
PageWidth	Defines the page width in pixels.	Int	R
PageHeight	Defines page height in pixels.	Int	R
MarginHor	Indicates the horizontal margin of the page in pixel	Int	RW
MarginVert	Indicates the vertical margin of the page in pixel	Int	RW
PageRows	Defines the number of printable rows in the page. This property is updated whenever the value of the property FontSize changes.	Int	R
PageColumns	Defines the number of columns that can be printed in the page. This property is updated whenever the value of the property FontSize changes.	Int	R

ESAPRN methods accessible with Scripts

Table 58: ESAPRN methods accessible with Scripts

Method	Description	OUT	IN
Start	Starts the print procedure and leaves the panel waiting for other print inputs. This function needs as an input parameter a value indicating whether the print setup window should be shown. If the value True is passed, the Options window is shown as soon as this instruction is executed. If the value False is passed, the print command is sent to the last printer used in the current session or to the default printer, if no printing has been performed in the current session. The actual printing starts when the method PRNEnd is invoked. The method returns 1 if the user has clicked on Ok (or if simply it has been decided not to show the DialogBox), 0 if the user has cancelled the operation or there is a negative integer indicating an error code. It is important to deal with cases in which a value other than 1 is returned so that no further print operations are run.	Int	DialogBox (Bool)
End	Concludes the print setup phase and sends the data to the printer.	-	-
Abort	Interrupts and aborts the print procedure being run.	-	-

Table 58: ESAPRN methods accessible with Scripts

Method	Description	OUT	IN
NewPage	With this a new print page can be created. After this function has been invoked, the next texts are printed on a new page.	-	-
WriteLN	Writes the text contained in the input string in a single row (going to the next line when the row has been printed).	-	Text (Str)

WriteXY	Writes the text contained in the input string into the position indicated, in pixels, by the two parameters PosX and PosY	-	PosX(Int)
			PosY(Int)
			Text (Str)

WriteRC	Writes the text contained in the input string into the position indicated, in terms of row and column positions, by the two parameters Col and Row.	-	Row(Int)
			Col(Int)
			Text (Str)

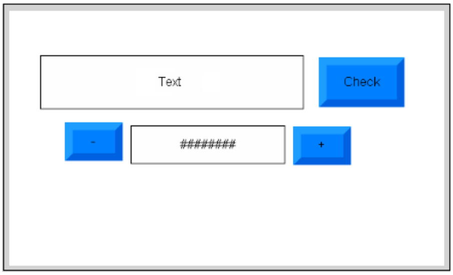
Examples of Script use

This paragraph deals practically with writing the scripting code. We offer examples relating the use of all the accessible objects described so far.

Example 1 - Analysis of variables and launching events

In this example we will suppose we have a project in which we configure a page, a variable, an alarm and the controls assigned to the page.
Using POLYMATH we set the objects we need while running the Script. We set a variable, calling it 'Tag' (the names of the objects assigned using POLYMATH are important as this is the key to accessing them using Scripts) of the Integer type assigning an initial value of 0. In addition, we set a generic alarm ('Alarm') that will be set off when the variable 'Tag' assumes the value 10. We remember to set in Alarms, in the User Signals mask (see chap. 5, "Usersignals" page 120), the display of one of the user signals present.
We set a page called 'Page' in which we insert a label (called 'Label') and a touch button ('Touch Button') to which we assign the Script ('Script) corresponding to the event 'onReleased'. Using Project Explorer, we drag the variable to

the work area to create a dynamic field showing its value in runtime (useful for constantly monitoring its value). We add two buttons to which we assign the predefined functions of increase-decrease value acting on the variable 'Tag' so as to be able to change the value in runtime. The page created will look like this:



Our Script must be able to get the value of the variable 'Tag', check that the value is less than 5 and, should this not be the case, launch an alarm, edit the layout of the label and the page and take the variable to a low value.
To get the value of the variable, we use ESATAG and save it into variable 'a' with the following instruction:

```
a=ESAHMI.ESATAG.ReadValue (“Tag”)
```

Now let us analyze the received value: if the value is greater than or equal to 5, an alarm is raised. Using POLYMATH we set an alarm to be activated when the value 10 was reached, so we are certain that it is the Script activating it now. The control and activation code uses the object ESAALARMMGR as indicated by the following rows:

```
If a>4 Then
ESAHMI.ESAALARMMGR.RaiseAlarm("Alarm")
End If
```

We can also run other instructions within the same condition such that when we change the value of the variable and launch the Script other changes will be applied, too. For example, we change the text, the color and the blinking of the label (object ESACNTRL, remembering to invoke the Draw method related to the label) and the background of the page (object ESAPAGE) as set out below:

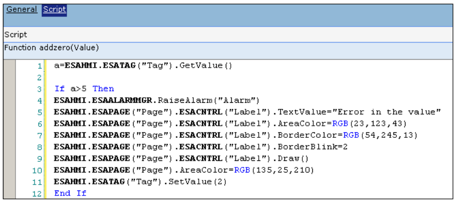
```
If a>4 Then
ESAHMI.ESAALARMMGR.RaiseAlarm("Alarm")
ESAHMI.ESAPAGE("Page").ESACNTRL("Label").TextValue
="ValueError"
ESAHMI.ESAPAGE("Page").ESACNTRL("Label").AreaColor
=RGB (23,123,43)
```

```
ESAHMI.ESAPAGE("Page").ESACNTRL("Label").BorderColor=RGB (54,245,13)
ESAHMI.ESAPAGE("Page").ESACNTRL("Label").BorderBlink=2
ESAHMI.ESAPAGE("Page").ESACNTRL("Label").Draw()
ESAHMI.ESAPAGE("Page").AreaColor=RGB(25,25,25)
End If
```

Finally, we re-establish an admissible value for the variable with the following instruction:

```
ESAHMI.ESATAG.WriteValue ‘‘Tag’’,2
```

The final code inserted in the POLYMATH editor is the following:



Script
Function addzero(Value)

```
a=ESAHMI.ESATAG("Tag").GetValue()

If a>5 Then
ESAHMI.ESAALARMMGR.RaiseAlarm("Alarm")
ESAHMI.ESAPAGE("Page").ESACNTRL("Label").TextValue="Error in the value"
ESAHMI.ESAPAGE("Page").ESACNTRL("Label").AreaColor=RGB(23,123,43)
ESAHMI.ESAPAGE("Page").ESACNTRL("Label").BorderColor=RGB(54,245,13)
ESAHMI.ESAPAGE("Page").ESACNTRL("Label").BorderBlink=2
ESAHMI.ESAPAGE("Page").ESACNTRL("Label").Draw()
ESAHMI.ESAPAGE("Page").AreaColor=RGB(135,25,210)
ESAHMI.ESATAG("Tag").SetValue(2)
End If
```

Example 2 - Page access according to user level

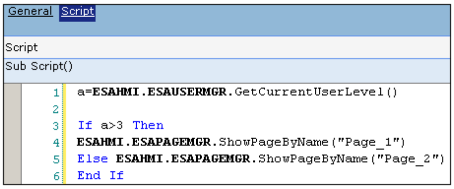
Another example of using Scripts is the way access to project pages is managed according to the level of the user currently logged onto the terminal.
Using POLYMATH we can set the objects we need while the Script is run. We set two levels of use (see chap. 5, “Password configuration” page 81), assigning a password for levels 3 and 8, for example. Remember that when the project starts the predefined level is 10, that is, the lowest.
We add 3 buttons to the default page (‘Page’): one recalling the Script, the other two the log-in and log-out functions respectively. Finally we set two new pages (‘Page_1’ and ‘Page_2’) that will be recalled by the Script depending on the user level.
Let us look now at the implementation of the code: first of all we must use the object USERMGR to get the level of the user currently logged in:

```
a=ESAHMI.ESAUSERMGR.GetCurrentUserLevel()
```

Now we need merely create a check condition for this level (the function returns an integer). The credentials of the user will determine which page is displayed.

```
If a>3 Then
ESAHMI.ESAPAGEMGR.ShowPageByName("Page_1")
Else ESAHMI.ESAPAGEMGR.ShowPageByName("Page_2")
End If
```

The complete Script code is ass follows:

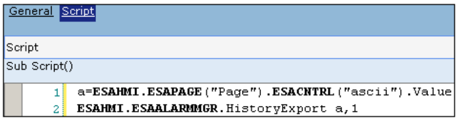


Script

```
Sub Script()
a=ESAHMI.ESAUSERMGR.GetCurrentUserLevel()

If a>3 Then
ESAHMI.ESAPAGEMGR.ShowPageByName("Page_1")
Else ESAHMI.ESAPAGEMGR.ShowPageByName("Page_2")
End If
```

Example 3 - Exporting alarms to a file chosen by the user

Another example of how POLYMATH Scripts can be used is provided by the use of value fields to receive data to be used to invoke dynamic functions. We insert a complex field into a page and this displays the Alarm history, an ASCII field (‘ASCII’, assigned to a string-variable) and a button to which we assign a Script (event onReleased).
The Script reads the value of the ASCII field and saves it in variable ‘a’ using the following instruction:
a=**ESAHMI**.**ESAPAGE**("Page").**ESACNTRL**("ASCII").Value
Then we invoke the Export alarms function to which we pass the string that has just been read:
ESAHMI.ESAALARMMGR.HistoryExport a,1



Script

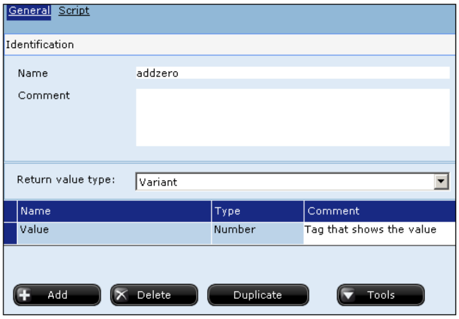
```
Sub Script()
a=ESAHMI.ESAPAGE("Page").ESACNTRL("ascii").Value
ESAHMI.ESAALARMMGR.HistoryExport a,1
```

Naturally this is only a simple example useful for illustrating the ease of programming via scripting that makes creating the project extremely dynamic.

Example 4 - Saving a Recipe into a memory

In this subsection we will show how it is possible, for example, using a Script to force the loading, the saving and exporting of certain recipes when a bit in the device is raised. To do this, we assign this Script to the event OnRawValueChange of the control bit (in our case, the variable ‘Control’). The PLC raises the status of the bit every X minutes allowing the Script to operate.
The Script operates the saving of the export file using a format of the type ric_DATA_ORA.xml.
In the project we create a type of recipe called ‘Proportions’ and define it as we wish; this will be used in the Script.

In this example we also introduce the use of a function that checks a variable and returns a value; namely, the values relating to the days, months, hours, minutes and seconds returned by the functions VBSCRIPT can be values of 1 digit. So that all files saved have the same format and the same length, we write a function of a few rows that adds a 0 in front of a digit if it is less than 10.



Identification

Name: addzero

Comment:

Return value type: Variant

Name	Type	Comment
Value	Number	Tag that shows the value

Using POLYMATH we create a Script in the usual way, but in the general page we assign a name (‘addzero’), a type of returned value (Variant) and an input value (‘value’, numeric). We have created the structure of our function: now we write its code:

```
If value<10 Then
value="0" & value
End If
addzero=value
```

If the input value (‘value’) is less than ten (that is, consists of only one digit), add the string “0” to the variable and finally it returns the value of ‘Value’ (if the cycle is not accessed the function simply returns the value received as an input parameter). The following is an example of applying this function: addzero(5) is invoked by giving the value 5, and returns the value ‘05’.

Now let us analyze the code of our main Script:

```
a=ESAHMI.ESATAG("Check").GetRawValue()
```

First of all we read the raw value of the Check variable and if its value is 1 we run our operations (this way we avoid executing them when the bit passes from 1 to 0). The If cycle is as follows:

```
If (a=1) Then
ESAHMI.ESARECIPETRF.RecipeBufferUpload
"Proportions",0
ESAHMI.ESARECIPETRF.SaveRecipe
"Proportions","Recipe",0
```

```
End If
```

We execute the upload of the recipe loaded onto the PLC (type is ‘Proportions’) in the first rows of the cycle, while in the second row we save that recipe (using the name ‘Recipe’) onto the terminal. Now all we need is the save phase that is run using the following instruction:

```
ESAHMI.ESARECIPEARC.RecipeExport
dest,"Proportions",""
ESAHMI.ESATAG.WriteValue ‘‘Check’’,0
```

All the recipes are exported (the third parameter is an empty string) and they are saved in the file indicated in the string variable ‘dest’ which we shall now go on to construct. After the save operation the check bit returns to 0.

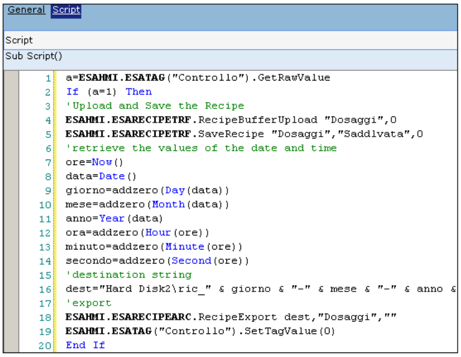
The string ‘dest’ is constructed by adding the details relating to the date and time of the execution of the operation. This information can be obtained using the functions put at the user’s disposal by the programming language, VBScript:

```
time=Now()
date=Date()
day=addzero(Day(date)
month=addzero(Month(date)
year=Year(dat)
hour=addzero(Hour(time)
minute=addzero(Minute(time)
second=addzero(Second(time)
dest="Hard Disk2\ric_"  & day & "-" & month & "-" & year &
"_h" & hour & "." & minute & "." & second & ".xml"
```

As we can see, the variables day, month, hour, minute and second are passed to the addzero function defined by us in which the zeroes for one-digit values are added.

The final instruction leads to constructing the string ‘dest’ indicating the path and name of the file to which the recipes are exported. In our case, we will save onto the support called ‘Hard Disk2’ (which, for example, could be a USB key) with a name of the type ‘ric_02-12-2005_h12.13.08.xml’. In this way we will be certain to have a series of distinct exportations in a file with unique names in terms of the support.

What follows is an overall view of the Script that has just been configured



Script
Sub Script()

```
a=ESAHMI.ESATAG("Controllo").GetRawValue
If (a=1) Then
'Upload and Save the Recipe
ESAHMI.ESARECIPETRF.RecipeBufferUpload "Dosaggi",0
ESAHMI.ESARECIPETRF.SaveRecipe "Dosaggi","Saddlvata",0
'retrieve the values of the date and time
ore=Now()
data=Date()
giorno=addzero(Day(data))
mese=addzero(Month(data))
anno=Year(data)
ora=addzero(Hour(ore))
minuto=addzero(Minute(ore))
secondo=addzero(Second(ore))
'destination string
dest="Hard Disk2\ric_" & giorno & "-" & mese & "-" & anno &
'export
ESAHMI.ESARECIPEARC.RecipeExport dest,"Dosaggi",""
ESAHMI.ESATAG("Controllo").SetTagValue(0)
End If
```

Example 5 - Canceling all the recipes in the VT

Putting together the methods described in this section you can construct customized functions according to your own project needs. In this example we shall see how to create a function of just a few rows that will cancel all the recipes saved in the VT. This is useful for avoiding cancelling each individual recipe manually and substituting it with a cumulative cancellation.
We also introduce a few rows of code allowing us to 'time' the execution of the entire script (giving us an identifying value of the time taken for it).
Let us now analyze the code:

```
t=Timer()
R_Type="Dieci_Var"
```

In the first line we ask for the instant the Script starts (the Timer function returns the number of seconds elapsed since 12:00 AM) and we save this in variable (t). In the second line we define the recipe type whose instances we want to cancel completely (alternatively we could pass this string value as a parameter for the function, as seen in example 4 for the 'addzero' function).
Next we go and get the name of the first recipe and save it in a variable (a):

```
a=ESAHMI.ESARECIPEARC.GetFirstRecipeName(R_Type)
```

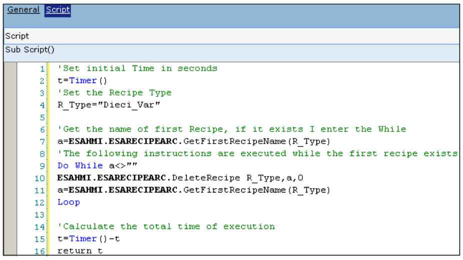
if there are no recipes for the type indicated (R_Type), the function returns an empty string (""). Thus cancellation should only occur if the string returned is different from "". We, therefore, use a 'Do While' cycle to make operation:

```
Do While a<>""
ESAHMI.ESARECIPEARC.DeleteRecipe R_Type,a,0
a=ESAHMI.ESARECIPEARC.GetFirstRecipeName(R_Type)
Loop
```

As we can see, the ‘While’ cycle remains open until such time as the value of ‘a’ is different from the empty string (that is, until recipes have been saved).
Cancellation occurs in accordance with the type indicated at the beginning of the Script and the current value of a (recipe name). In addition the value 0 is passed to avoid confirmation being asked of the operator. Within the cycle we also update the value of a by getting the new first recipe (we again use GetFirst rather than GetNext because the delete operation has changed the order of the recipes).
Exiting from the While cycle, all the recipes have been eliminated, so all we can do is get the time taken by the Script:

```
t=Timer()-t
return t
```

Using this instruction, the value of t is updated by removing from the current value of Timer() the value obtained at the beginning of the Script (saved in t). Thus, at the end of this instruction t will contain the number of seconds elapsed between the beginning and the end of the cancel operation.
Below is the complete code of our Script:



Script

Sub Script()

```
'Set initial Time in seconds
t=Timer()
'Set the Recipe Type
R_Type="Dieci_Var"

'Get the name of first Recipe, if it exists I enter the While
a=ESAHMI.ESARECIPEARC.GetFirstRecipeName(R_Type)
'The following instructions are executed while the first recipe exists
Do While a<>""
ESAHMI.ESARECIPEARC.DeleteRecipe R_Type,a,0
a=ESAHMI.ESARECIPEARC.GetFirstRecipeName(R_Type)
Loop

'Calculate the total time of execution
t=Timer()-t
return t
```

Example 6: Creates printout of list of recipes

What follows is an example illustrating the use of the print functions. Supposing we want a paper printout of the list of recipes present in the memory of the terminal. The logic behind searching for recipes is similar to that used in the previous example.
Let us first of all initialize the print session using the Start method: by providing parameter 1 the Print options window is shown in runtime before the print session starts. To abort the print operation the user can click on the X of the window. This control is carried out with an If that checks and, where appropriate, stops the running of all the other code rows:

```
if (ESAHMI.ESAPRN.Start(1)=1) Then
```

Now we create a page heading of a title and two blank rows to separate the title from the contents. To leave blank rows we use the method WriteLN, passing an empty string. Before writing the title, we set the font at a higher value which we then reduce to a smaller font for the rest of the page.

```
ESAHMI.ESAPRN.FontSize=16
ESAHMI.ESAPRN.WriteLN(“Recipe Lists in the VT“)
ESAHMI.ESAPRN.WriteLN(““)
ESAHMI.ESAPRN.WriteLN(““)
ESAHMI.ESAPRN.FontSize=12
```

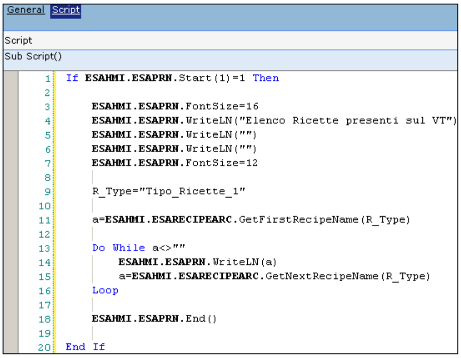
At this point we instance the read-cycle of the recipes saved in the VT using the methods GetFirstRecipeName and GetNextRecipeName. Within the cycle we use the method PrintLN to have the name of a recipe in each line.

```
R_Type= “Tipo_Recipes_1”
a=ESAHMI.ESARECIPEARC.GetFirstRecipeName(R_Type)
Do While a<>""
ESAHMI.ESAPRN.WriteLN(a)
a=ESAHMI.ESARECIPEARC.GetNextRecipeName(R_Type)
Loop
```

Up to this point we have prepared the contents of the page, now we launch the command that actually starts the printing:

```
ESAHMI.ESAPRN.End()
End If
```

With the execution of this method the print process begins.
Below we show the complete text of the Script:



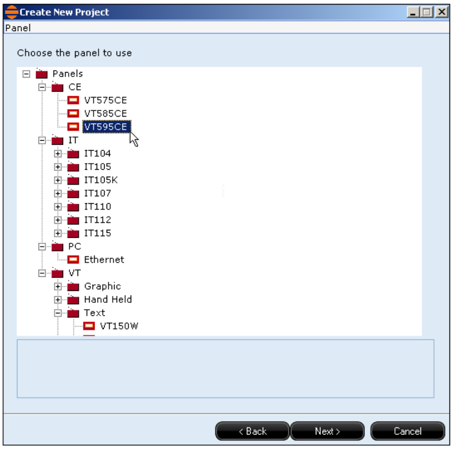
```
If ESAHMI.ESAPRN.Start(1)=1 Then

    ESAHMI.ESAPRN.FontSize=16
    ESAHMI.ESAPRN.WriteLN("Elenco Ricette presenti sul VT")
    ESAHMI.ESAPRN.WriteLN("")
    ESAHMI.ESAPRN.WriteLN("")
    ESAHMI.ESAPRN.FontSize=12

    R_Type="Tipo_Ricette_1"

    a=ESAHMI.ESARECIPEARC.GetFirstRecipeName(R_Type)

    Do While a<>""
        ESAHMI.ESAPRN.WriteLN(a)
        a=ESAHMI.ESARECIPEARC.GetNextRecipeName(R_Type)
    Loop

    ESAHMI.ESAPRN.End()

End If
```

10. Tutorial

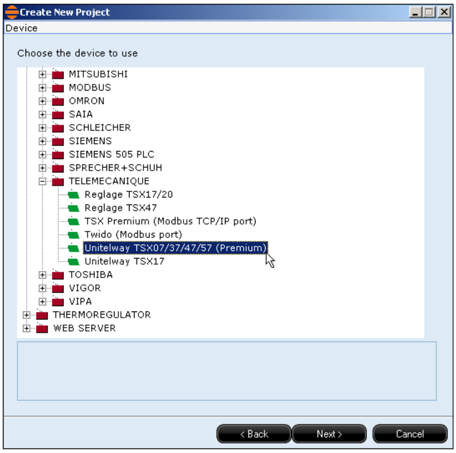
The purpose of this chapter is to give practical examples of how POLYMATH can be used to create complete projects. We shall try to include all the functions offered by the application together with simple but exhaustive descriptions. For our project we will be using an ESA VT595 terminal with Windows® CE operating system and a Telemecanique TSX 37 Micro PLC.
The following sections deal with the editing for every aspect of the programming phase beginning from Hardware and Software configuration to navigation procedures, access and management of alarms and recipes in the project. In this Tutorial project we do not intend to explain how Scripts are composed or how to use the Library, as these topics are already fully dealt with elsewhere in this manual (see chap. 9, “Scripts” page 379 and see chap. 7, “POLYMATH Libraries” page 341).

Phase 1 - The Project and Hardware Configuration

The first operation that must be performed with POLYMATH is the creation of a new project, defining its operating procedures. The quickest and most natural way to do this is to create a project using a Wizard. To do this go to the main menu and select File->New. In the work area there will be a series of windows for determining general preferences and those relating to the Hardware aspect of the project.



First of all we are asked to identify the model of ESA terminal that we are using and that we intend to use for our project. In our case, we will select VT595CE from among the CE panels in the list and then click on 'Continue'.

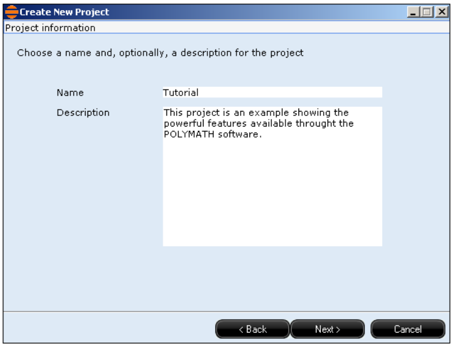


Device

Choose the device to use

- MITSUBISHI
- MODBUS
- OMRON
- SAIA
- SCHLEICHER
- SIEMENS
- SIEMENS 505 PLC
- SPRECHER+SCHUH
- TELEMECANIQUE
  - Reglage TSX17/20
  - Reglage TSX47
  - TSX Premium (Modbus TCP/IP port)
  - Twido (Modbus port)
  - Unitelway TSX07/37/47/57 (Premium)
  - Unitelway TSX17
- TOSHIBA
- VIGOR
- VIPA
- THERMOREGULATOR
- WEB SERVER

In the next page we select the device the ESA panel should interface with (in our example we will use a Telemecanique PLC. Using the list of devices (in the form of a tree-diagram), we search the category of PLC to find the makers of our chosen device. We select the model we are using, one which uses a compatible protocol (in our case, Unitelway TSX07/37/47/57 Premium) and then click on 'Continue' to proceed with the configuration.



Project information

Choose a name and, optionally, a description for the project

Name: Tutorial

Description: This project is an example showing the powerful features available throught the POLYMATH software.

The last operation is to give a name to our project and a description allowing us to identify it. The names and the descriptions given in this phase have no functional value in the project but serve only to make identifying it easier. At this point we click on 'Continue' and then 'End' to conclude the project set-up operations.

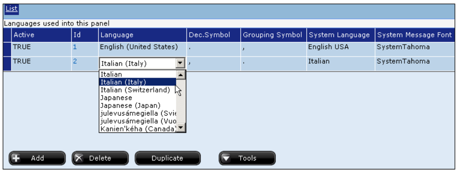
POLYMATH now creates all the connections we need to interact with the panel-device (communication ports, addresses etc.). The project's Hardware settings can, of course, be changed at any time simply by clicking on Hardware Configuration in Project Explorer.

Phase 2 - Software configuration

When configuring the Software we can define the global preferences relating to our project. In our example we customize the project by stipulating two languages, three user levels, a personalized font (not one of the default ones) and a Timer. We shall, however, leave the options relating to the translations of messages and systems alarms unchanged. (The programmer can, of course, decide to customize every single message.)

Setting project languages

We wish to produce a multi-language project, in which it will be up to the end-user (operator) to decide the language with which to interact with the panel in runtime (in practice, this means choosing the language in which the messages, the errors and the texts that appear in the pages will appear). In our example we shall set two languages: Italian and English. To carry out this operation, we go to the appropriate page by clicking on the Project Explorer option Languages in the VT595CE Configuration Software.

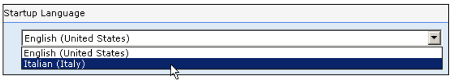


List

Languages used into this panel

Active	Id	Language	Dec.Symbol	Grouping Symbol	System Language	System Message Font
TRUE	1	English (United States)	.	,	English USA	SystemTahoma
TRUE	2	Italian (Italy)	,	.	Italian	SystemTahoma

By default POLYMATH inserts English in the project, while by clicking on 'Add' a new language can be introduced. In our case Italian is introduced as a second default language (we can always change the project languages by selecting them from the pull-down menu).

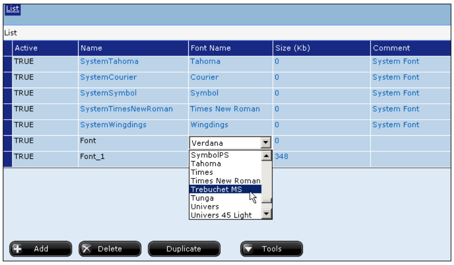


Startup Language: English (United States)

Using the same page, we set Italian as the display language at the opening the project in runtime. In any case, the operator can change the current display language by using the commands that give us access to it.

Inserting a new font

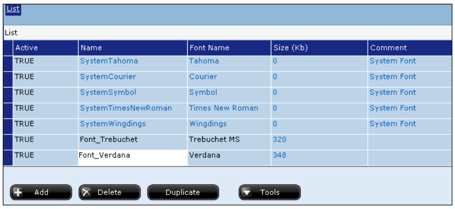
Now let us add two more fonts to display the project texts with.



List

Active	Name	Font Name	Size (Kb)	Comment
TRUE	SystemTahoma	Tahoma	0	System Font
TRUE	SystemCourier	Courier	0	System Font
TRUE	SystemSymbol	Symbol	0	System Font
TRUE	SystemTimesNewRoman	Times New Roman	0	System Font
TRUE	SystemWingdings	Wingdings	0	System Font
TRUE	Font	Verdana	0	
TRUE	Font_1		348	

For example, we select the fonts Trebuchet MS and Verdana from the list containing the fonts in the PC where POLYMATH has been installed.



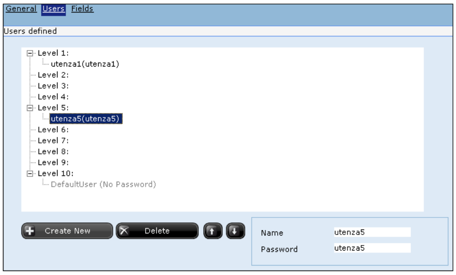
List

Active	Name	Font Name	Size (Kb)	Comment
TRUE	SystemTahoma	Tahoma	0	System Font
TRUE	SystemCourier	Courier	0	System Font
TRUE	SystemSymbol	Symbol	0	System Font
TRUE	SystemTimesNewRoman	Times New Roman	0	System Font
TRUE	SystemWingdings	Wingdings	0	System Font
TRUE	Font_Trebuchet	Trebuchet MS	320	
TRUE	Font_Verdana	Verdana	348	

We must also assign IDs to the fonts we have just added so that they can easily be identified when we want to use them in the project: Font_Trebuchet and Font_Verdana.

Settings identifying users

The next step is to define the users of our project. It is necessary to know from the outset who will be interfacing with our project and what their respective rights will be.

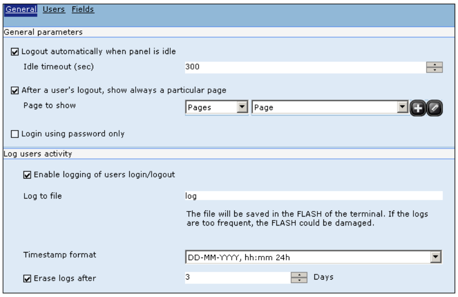


Users defined

- Level 1:
  - utenza1(utenza1)
- Level 2:
- Level 3:
- Level 4:
- Level 5:
  - utenza5(utenza5)
- Level 6:
- Level 7:
- Level 8:
- Level 9:
- Level 10:
  - DefaultUser (No Password)

Name utenza5
Password utenza5

In our project we will stipulate 2 users: one at level 1 (maximum access rights) and one at level 5 (lower credentials). As for user names and passwords, we will call them ‘user1’ and ‘user5’ respectively (same value for user name and password, in general it is advisable to insert different strings).



General parameters

☑ Logout automatically when panel is idle
Idle timeout (sec) 300

☑ After a user's logout, show always a particular page
Page to show Pages Page

☐ Login using password only

Log users activity

☑ Enable logging of users login/logout

Log to file log

The file will be saved in the FLASH of the terminal. If the logs are too frequent, the FLASH could be damaged.

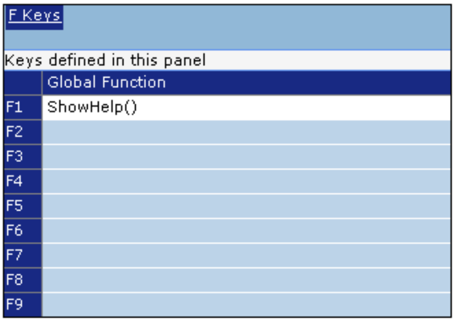
Timestamp format DD-MM-YYYY, hh:mm 24h

☑ Erase logs after 3 Days

The same window now allows us to access the General mask to set general values relating to managing users in runtime. For example, we set the automatic logout after a period of inactivity at 5 minutes (300 seconds) and force the view of the first page (‘Page’) when the logout is executed. In conclusion we set a name (e.g. ‘log’) for the file to register the users’ activity.

Setting global keys

We now set the global keys of the application. Their functions come into effect whenever the button corresponding to them is pressed (using either the physical or the virtual keyboard) irrespective of the context.



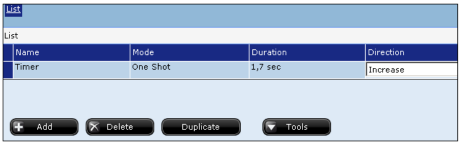
Keys defined in this panel

	Global Function
F1	ShowHelp()
F2	
F3	
F4	
F5	
F6	
F7	
F8	
F9	

In our project F1 will be assigned the function displaying the page-linked Help; in this way, whenever the operator presses button F1 the Help relating to the page being displayed will appear.

Setting timers

The last operation of the software configuration is setting timer to be used to manage a trend (see below).



List

Name	Mode	Duration	Direction
Timer	One Shot	1,7 sec	Increase

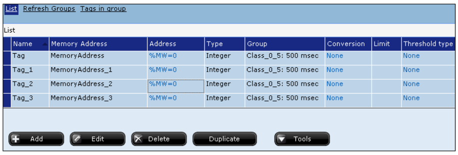
We set a One-shot Timer called ‘Timer’ to last 1.7 seconds with an ascending direction.

Phase 3 - Configuration of variables and Memory areas

Without doubt, data management is the most important aspect in creating a project for the work of the terminal. It is essential to have a clear idea of the structure the data needs to respect in runtime and how the operator can access them. With POLYMATH internal variables as well as device and system variables can be managed (see chap. 5, “Variables” page 88). It is also possible to manage memory areas dedicated, for example, to status indication commands. Naturally, the list of variables can be accessed at any point during the editing of the project. In our example, we do this to define some examples of each type of variable.

Defining device variables

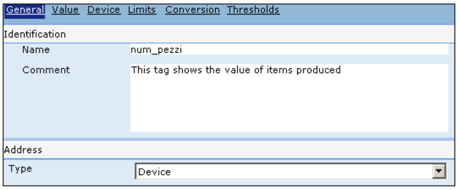
If we click twice on the ‘Tags’ option of the Project Explorer, we access the list of variables defined in the project.
Now click on ‘Add’ to insert the new variables.



List

Name	Memory Address	Address	Type	Group	Conversion	Limit	Threshold type
Tag	MemoryAddress	%MW=0	Integer	Class_0_5: 500 msec	None		None
Tag_1	MemoryAddress_1	%MW=0	Integer	Class_0_5: 500 msec	None		None
Tag_2	MemoryAddress_2	%MW=0	Integer	Class_0_5: 500 msec	None		None
Tag_3	MemoryAddress_3	%MW=0	Integer	Class_0_5: 500 msec	None		None

For the moment we will introduce 4 variables which we will then edit individually.
We shall now describe in detail how the first variable is edited, the procedure being identical for the following ones.
We start with the General mask where we digit the name and the comment of the variable :



Identification

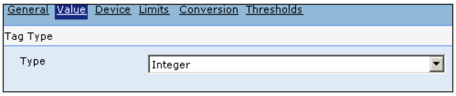
Name: num_pezzi

Comment: This tag shows the value of items produced

Address

Type: Device

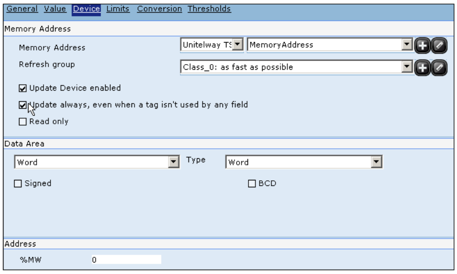
Let us call the variable ‘num_pezzi’ (and add a brief description which may be useful for identifying what the variable is for in the future), defining its location as ‘Device’.



Tag Type

Type: Integer

In the ‘Value’ mask we specify the type of value as ‘Integer’.



Memory Address

Memory Address: Unitelway TS | MemoryAddress

Refresh group: Class_0: as fast as possible

☑ Update Device enabled

☑ Update always, even when a tag isn't used by any field

□ Read only

Data Area

Word | Type: Word

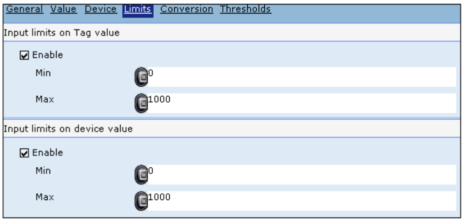
□ Signed □ BCD

Address

%MW 0

We will leave the default settings in the Device mask (memory address = Memory Address) and proceed to enable the option 'Update continually, even when no tag is used by a field' so that this variable can be controlled by a Script. The value of the word containing the variable must also be assigned, specifying memory addresses different for all tags so that there will not be wrong references in runtime (unless there is a definite intention for them to coincide): we give the first variable the memory address 0.

Note: *The address of the variables can be edited directly through this mask when individual variable areas being edited or, alternatively, using the mask for managing addresses in the device memory (see chap. 5, "MemoryAddresses" page 155). In either case, the changes made to a 'MemoryAddress' will influence all the variables referring to it.*



Input limits on Tag value

☑ Enable

Min 0

Max 1000

Input limits on device value

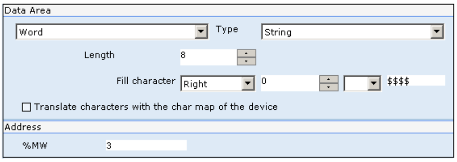
☑ Enable

Min 0

Max 1000

The next step is to set the limits (on both the panel and the device values) for this variable. Let us suppose that this is always a value between 0 and 1000, then if you try to go outside these limits in runtime, the value will automatically be put at the nearest limit to the value requested. Our example will not use conversions and thresholds: we leave the possibility of assigning these in the way described in the related part of this manual (see chap. 5, "Conversion" page 98 and see chap. 5, "Thresholds" page 99) to the user.
Thus we have finished configuring the first variable, 'num_pezzi'; the other 3 variables are configured in exactly the same way and thus we can edit them to give:
- an integer variable called 'int_var' with an address of word 1
- a real variable called 'real_var' with an address of word 2

- a string variable called ‘str_var’ with an address starting at word 3 and having a length of 8 characters (so it also occupies word 4, 5 and 6) as shown in the figure below :



Data Area

Word ▾ Type String ▾

Length 8

Fill character Right ▾ 0 ▾ $$$$

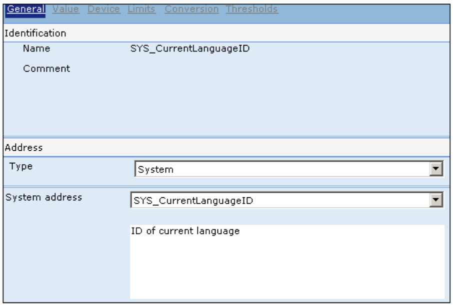
□ Translate characters with the char map of the device

Address

%MW 3

Defining system variables

We will now define a further two project variables indicating the current status of the process in runtime; the variables that allow us to do this are system variables (see “Appendix A - System Variables” page 555).



Identification

Name SYS_CurrentLanguageID

Comment

Address

Type System ▾

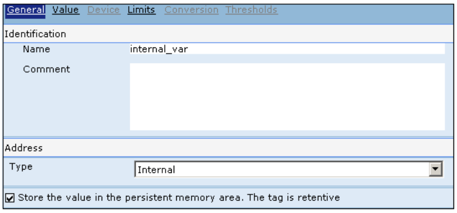
System address SYS_CurrentLanguageID ▾

ID of current language

For example we will define one variable indicating the ID number of the language currently in use in the project (SYS_CurrentLanguageID) and one indicating the date and time of the panel (SYS_DateAndTime).

Defining internal variables

We will now also use an internal variable that does not relate to the device in any way. It is a variable that works on the terminal irrespective of the status of the PLC. This type of variable is defined differently from a device variable, in that it is not possible to define memory, conversion and threshold values.



Identification

Name: internal_var

Comment:

Address

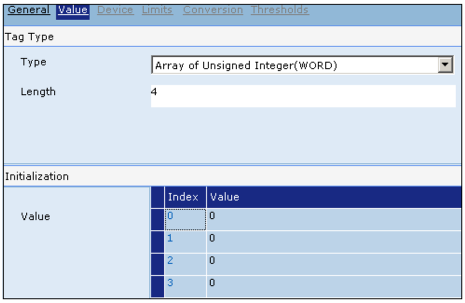
Type: Internal

☑ Store the value in the persistent memory area. The tag is retentive

We use the General mask to specify the name (like 'internal_var') and, of course, we set the type of address as Internal. In addition, we enable the option allowing the variable to be made retentive, that is, to maintain its value even after the terminal has been switched off. In the Value mask we set the type of variable as Integer.

Defining memory areas

Besides individual variables, with POLYMATH it is possible to define consecutive memory areas (value arrays or indexed variables) that can be used, for example, to define Exchange Areas (see chap. 5, "Exchange areas" page 76).
To insert these memory areas in a project use the same procedure as for normal variables :



Tag Type

Type: Array of Unsigned Integer(WORD)

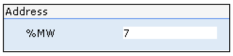
Length: 4

Initialization

Value

Index	Value
0	0
1	0
2	0
3	0

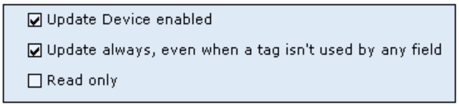
After assigning a name to the variable, 'array_var_4', we use the Value mask to specify the type (like ArrayOfUnsignedInteger) and the dimensions of the area (equivalent of 4 elements).



Address

%MW: 7

We now use the Device mask to set as starting memory address word 7 (namely the first to remain free - as a result, words 8-9 and 10 will also be occupied) and enable the continuous update option as illustrated below :



☑ Update Device enabled

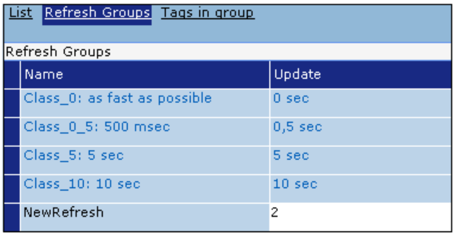
☑ Update always, even when a tag isn't used by any field

□ Read only

In conclusion, we use the same procedure to define another two memory areas, calling them ‘array_var_6’ and ‘array_var_2’ respectively and defining their dimensions as equal to 6 and 2. We give the variable ‘array_var_6’ the addresses from word 11 to 16 inclusive and give the variable ‘array_var_2’ the addresses word 17 and 18.

Setting refresh times

Now that we have defined the list of variables, we can go on to attribute particular refresh policies to some of them by accessing the list of variables (double-clicking on ‘Variables’ in Project Explorer) and opening the Refresh Groups mask.

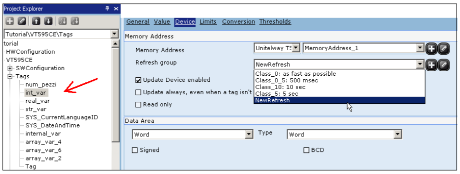


Refresh Groups

Name	Update
Class_0: as fast as possible	0 sec
Class_0_5: 500 msec	0,5 sec
Class_5: 5 sec	5 sec
Class_10: 10 sec	10 sec
NewRefresh	2

We add a customized group to the list of groups by clicking on ‘Add’. We will call the new type NewRefresh and set the update time as 2 seconds.

Finally we attribute the group we have just defined to the variable ‘int_var’ by acting on the tag "Device" window :



Memory Address

Memory Address: Unitelway TS MemoryAddress_1

Refresh group: NewRefresh

☑ Update Device enabled

□ Update always, even when a tag isn't

□ Read only

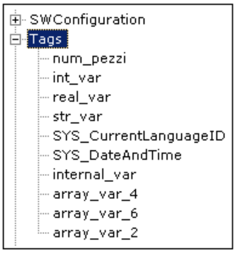
Data Area

Word Type Word

□ Signed □ BCD

For the other variables we will leave the refresh group as predefined "Class_0 : as fast as possible".

Summary of variables and memory area



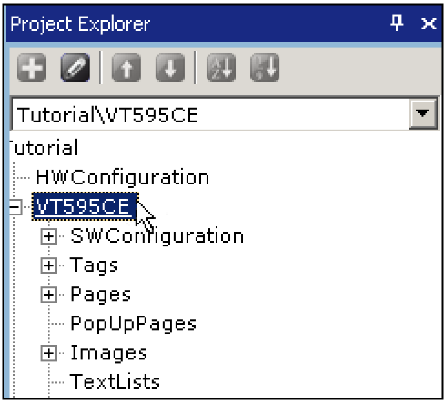
We have thus defined 9 variables of different types that we can use as we wish within our project. The next table offers a summary also of the use of the memory addresses as specified in our work up to this point.

Tabella 1: Organization of Memory area

Address	Memory name	Variable
W 0	MemoryAddress	num_pezzi
W 1	MemoryAddress_1	int_var
W 2	MemoryAddress_2	real_var
W 3-4-5-6	MemoryAddress_3	str_var
W 7-8-9-10	MemoryAddress_4	array_var_4
W 11-12-13-14-15-16	MemoryAddress_5	array_var_6
W 17-18	MemoryAddress_6	array_var_2

Phase 4 - General configuration of the VT

Having defined the Hardware and Software structure as well as the data areas of the job (variables), we now also provide the general work settings for the terminal.



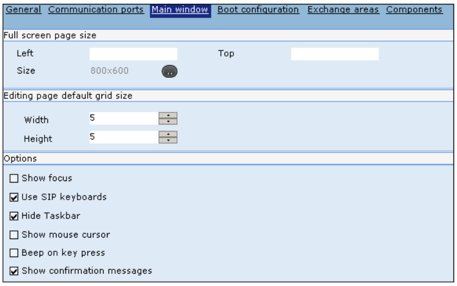
Tutorial\VT595CE

```
utorial
  HWConfiguration
- VT595CE
    + SWConfiguration
    + Tags
    + Pages
      PopUpPages
    + Images
      TextLists
```

For this we double-click on the name of the terminal in Project Explorer (in our case this is *VT595CE*).

Setting the main window

Here, in the VT section, the window is again arranged in masks. We go to the 'Main Window' mask, where we can set our general preferences regarding the appearance of the project in runtime.



Full screen page size

Left Top

Size 800x600

Editing page default grid size

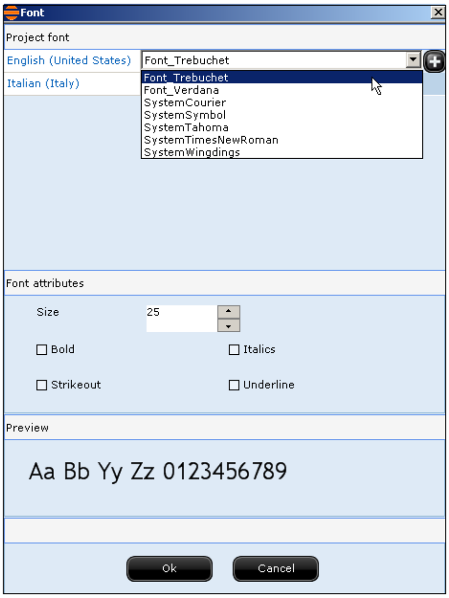
Width 5

Height 5

Options

- ☐ Show focus
- ☑ Use SIP keyboards
- ☑ Hide Taskbar
- ☐ Show mouse cursor
- ☐ Beep on key press
- ☑ Show confirmation messages

In our example we have decided to specify the dimensions of the grid as 5-5 (to make the editing more precise) and to leave the default settings for the page display (focus, reduce button, etc.).
Using the lower part of the mask, we set at 5 the consent level allowing a user to display system pages (then access is given when you log on). We then edit the Help-pages font by clicking on ●



Font

Project font

English (United States) Font_Trebuchet

Italian (Italy)

Font attributes

Size 25

□ Bold □ Italics

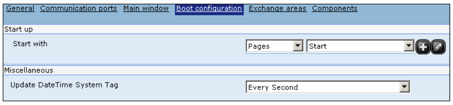
□ Strikeout □ Underline

Preview

Aa Bb Yy Zz 0123456789

A different font can be specified for each of the project languages. In our example we set Font_Trebuchet which we previously inserted for both languages and set the font size at 25. These changes will be valid for all the Help-pages we set.

Configuring the Boot



Start up

Start with Pages Start

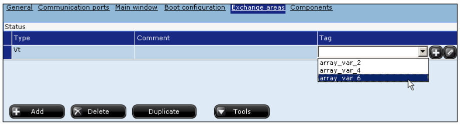
Miscellaneous

Update DateTime System Tag Every Second

In this mask we set the start-up options of the project. We indicate the language of the Operating System (English) and leave the page displayed on start-up as the default option (“Page”).

Setting Exchange areas

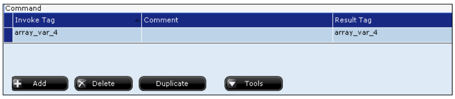
The last terminal configuration operation for our project is the definition of the exchange areas. After moving to the appropriate mask, we specify which of the variables (areas) created are to be dedicate to these checks.



Status

Type	Comment	Tag
Vt		array_var_2 array_var_4 array_var_6

We add a status area (see “Appendix C - Status area” page 575) by clicking on ‘Add’ and in the variables menu we assign the variable with the dimension of 6 (‘array_var_6’). The are type we will leave as the type of VT. The size of the status areas is always 6 words.



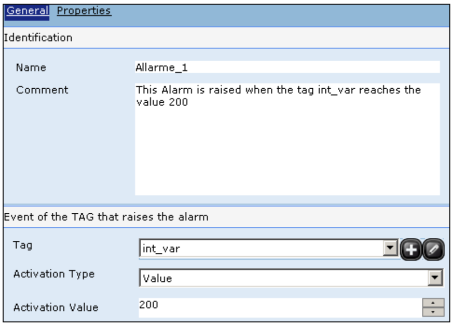
Command

Invoke Tag	Comment	Result Tag
array_var_4		array_var_4

Using the same procedure, we now also add a command area to which we assign the variable ‘array_var_4’ for the invoke command and the variable ‘array_var_2’ for the command to see the results of the operation. Thus in runtime word 7 will be checked for commands as indicated in the appropriate appendix (see “Appendix D - Command area” page 579).

Phase 5 - Defining the alarms

At this point in the programming we proceed to define the alarms to be taken into consideration using runtime. In our project we will define an alarm assigned to the variable num_pezzi: first of all, using Project Explorer, we click on Alarms and then on ‘Add’ to create a new alarm. By clicking twice on the alarm we have just created, we can start editing it :



Identification

Name: Allarme_1

Comment: This Alarm is raised when the tag int_var reaches the value 200

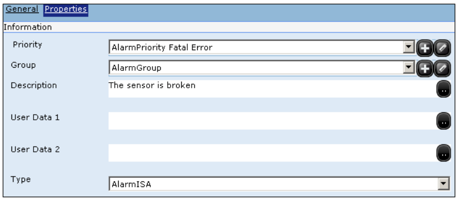
Event of the TAG that raises the alarm

Tag: int_var

Activation Type: Value

Activation Value: 200

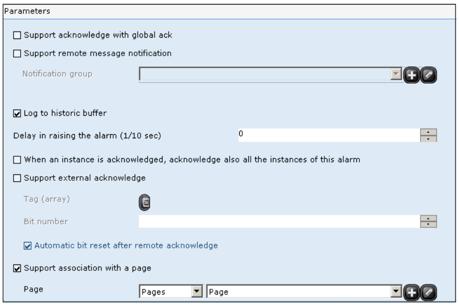
We set the name ‘Allarme_1’ and include a brief description. The lower part of the mask is used to define the more important properties of the alarm, namely the reference variable and the type of activation. Our alarm refers to the variable ‘int_var’ and is activated (by value) when it assumes the value 200.


General Properties

Information

Priority	AlarmPriority Fatal Error
Group	AlarmGroup
Description	The sensor is broken
User Data 1	
User Data 2	
Type	AlarmISA

We go to the properties mask simply to define the priority, here maximum (‘FatalError’), the group of alarms (managing the groups is only useful for cataloguing alarms in the project when many of them are configured) and a description of the alarm (the description is displayed on the panel when there is an error in runtime). As alarm type we will leave the default setting (ISA).


Parameters

□ Support acknowledge with global ack

□ Support remote message notification

Notification group

☑ Log to historic buffer

Delay in raising the alarm (1/10 sec) 0

□ When an instance is acknowledged, acknowledge also all the instances of this alarm

□ Support external acknowledge

Tag (array)

Bit number

☑ Automatic bit reset after remote acknowledge

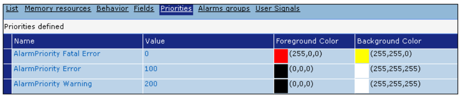
☑ Support association with a page

Page Pages Page

We now use the lower part to enable the log in the history buffer (in practical terms, with this option the instances of this alarm are listed in the history table we will set in due course). We also enable the association with a page to be displayed when requested by the user after the alarm has been raised (for now we will assign the first page).

General alarm settings

A series of general options relating to managing the alarms can be accessed by clicking twice on Alarms in Project Explorer. While we will leave the masks relating to the resources and the behavior of the alarm buffer memory unchanged, we will make changes to the Priorities mask.

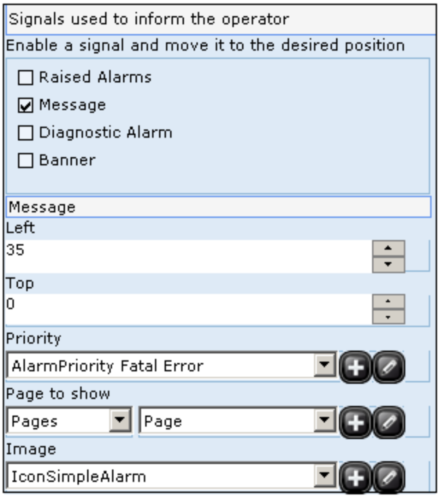


List | Memory resources | Behavior | Fields | Priorities | Alarms groups | User Signals

Priorities defined

Name	Value	Foreground Color	Background Color
AlarmPriority Fatal Error	0	🟥 (255,0,0)	🟨 (255,255,0)
AlarmPriority Error	100	⬛ (0,0,0)	⬜ (255,255,255)
AlarmPriority Warning	200	⬛ (0,0,0)	⬜ (255,255,255)

In this mask we will indicate the colors to be used to represent (in the pages showing the alarms with complex controls) the instances of the alarms that have ‘FatalError’ priority, like the one defined in the project.



Signals used to inform the operator

Enable a signal and move it to the desired position

☐ Raised Alarms
☑ Message
☐ Diagnostic Alarm
☐ Banner

Message

Left: 35

Top: 0

Priority: AlarmPriority Fatal Error

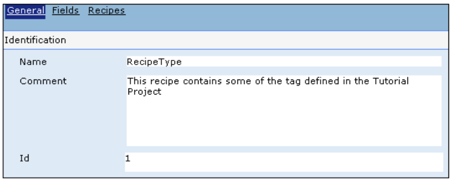
Page to show: Pages
Page

Image: IconSimpleAlarm

in the last mask we select the option ‘Message’ and set the same priority FatalError that we set for the alarm we defined). In this section we can establish how the operator is to be advised of the alarm being set off; we have chosen message, hat is, a little icon will appear on the screen (irrespective of the page the project is in when the alarm is raised). By clicking on this alarm icon in runtime the operator is taken to the page identified in the second field of this mask. Furthermore, we will leave the default image as the image associated with the alarm icon and at the bottom we will keep the enablement for the warning noise.

Phase 6 - Defining recipe types

We will now insert the definition of a type of recipe in our project: in POLYMATH we define only the structure of the recipe, the instances themselves of recipes must be defined (and appropriately saved or exported to a file) in runtime by the operator.



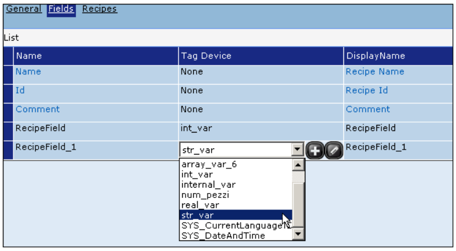
Identification

Name: RecipeType

Comment: This recipe contains some of the tag defined in the Tutorial Project

Id: 1

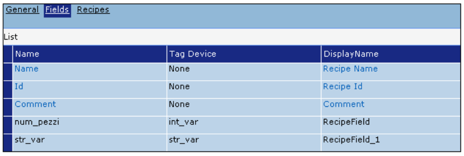
We use the General mask to set only the name 'RecipeType' and a short comment, leaving the default ID set by POLYMATH.



List

Name	Tag Device	DisplayName
Name	None	Recipe Name
Id	None	Recipe Id
Comment	None	Comment
RecipeField	int_var	RecipeField
RecipeField_1	str_var	RecipeField_1

Now we use the Fields mask to insert the variables that really make up the recipe. In our example we click on 'Add' and add two fields as shown above. We introduce two fields containing the variables 'int_var' and 'str_var" obtaining the following list:

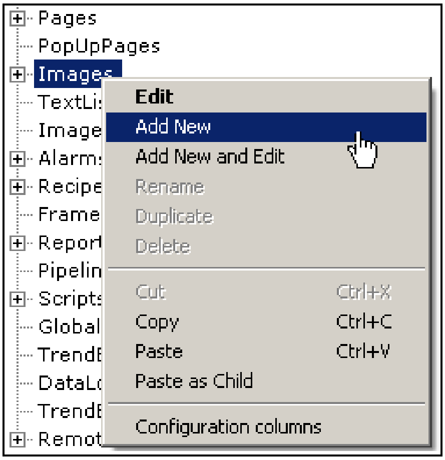


List

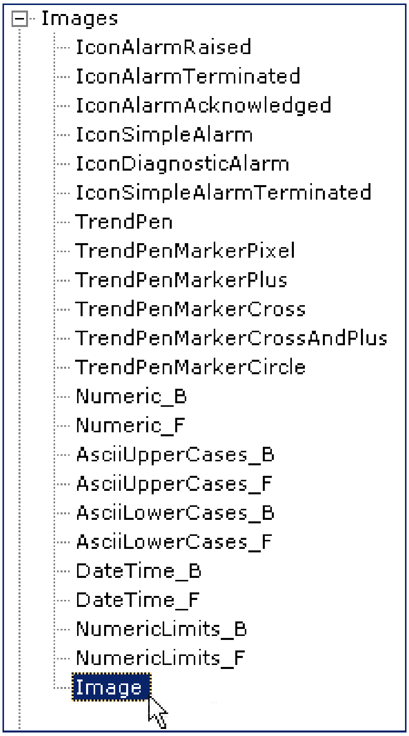
Name	Tag Device	DisplayName
Name	None	Recipe Name
Id	None	Recipe Id
Comment	None	Comment
num_pezzi	int_var	RecipeField
str_var	str_var	RecipeField_1

Phase 7 - Loading Images

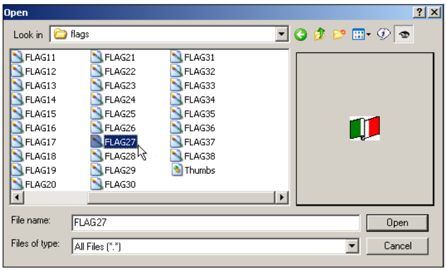
At this point we can try to load some images in the project that we can go on to use in our pages to make the functions more comprehensible or simply to give the project a more pleasing appearance. For example we introduce two images for the project languages, namely English and Italian, provided, of course, that these two images are on our hard disk.



First we load the image of the Italian flag, clicking on the 'Images' option in Project Explorer (right key) and then on 'Add New'.



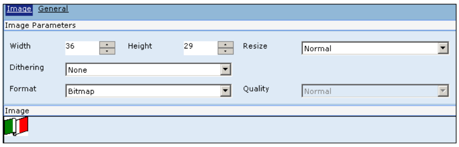
The list of images will now show the new image 'Image'; by clicking twice on this we can edit it.



File name: FLAG27

Files of type: All Files (*.*)

Now just click on the empty area in the work area to browse the contents of the Hard Disk and choose the file to insert.



Width 36

Height 29

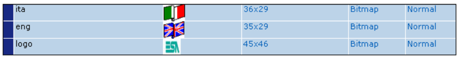
Resize Normal

Dithering None

Format Bitmap

Quality Normal

You are now directed to the editing mask of the images where you can set dimensions and compression characteristics. Color variants of the image can also be defined at this point (see chap. 5, "Operations performable on an image" page 111). For our images we will leave all these properties unchanged. We use the same procedure to introduce the images of the flag denoting the English language and the logo to use in our project ;



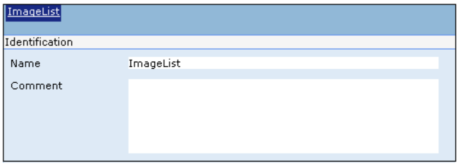
ita		36x29	Bitmap	Normal
eng		35x29	Bitmap	Normal
logo		45x46	Bitmap	Normal

The image objects of POLYMATH take the name of the source file but can still be edited: we have introduced 3 images with the names 'ita', 'eng' and 'logo'.

Phase 8 - Defining text and image lists

When defining a project it may be useful to display images or dynamic texts to the operator in relation to the value of a variable. For this reason, POLYMATH allows you to define object lists that can then be invoked in the project in association with a variable (depending on the value of the variable assigned, one or another item in the list will be shown).

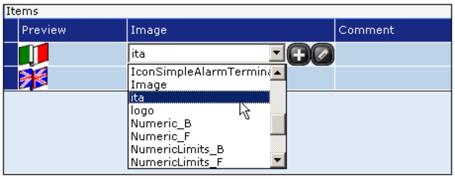
Let us now create a simple list of images using the two images of the flags introduced in the previous section. The procedure for creating the list is the usual one: double-click in Project Explorer on the option 'ImageList'; click on 'Add' and then in Project Explorer click twice on the list to be able to edit.



Identification

Name	ImageList
Comment	

In the edit mask we leave the name of the list as per default, 'ImageList', and click on 'Add' to introduce images to the list :

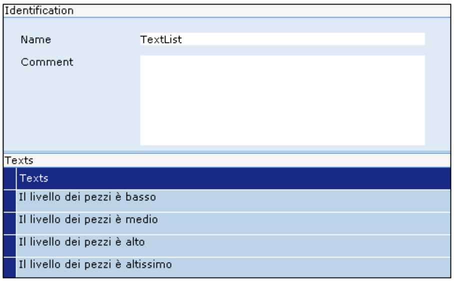


Items

Preview	Image	Comment
	ita	

In the Images column of the table we select the image to be introduced from the pull-down menu (choosing it from those included in the project). We add both the images related to the languages ('ita' and 'eng').

The procedure for creating text lists is exactly the same as for image lists: after creating one (with the default name of 'TextList') the following situation obtains :



Identification

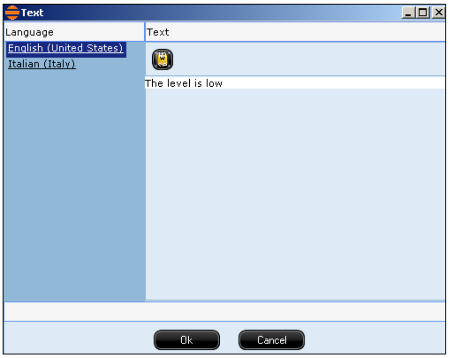
Name	TextList
Comment	

Texts

Texts
Il livello dei pezzi è basso
Il livello dei pezzi è medio
Il livello dei pezzi è alto
Il livello dei pezzi è altissimo

We now add 4 texts relating to the value of a variable; for example, we insert the following strings "The level of pieces is low", "The level of pieces is normal", "The level of pieces is high" and "The level of pieces is very high".

Each string inserted needs a translation in all the languages of the project, thus in our case we have to provide a translation in English. To insert the translation we click on the icon adjacent to each string, thereby opening the corresponding translation window :

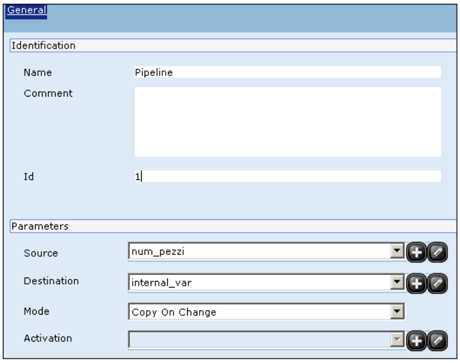


Language	Text
English (United States)	The level is low
Italian (Italy)	

Having provided the translation for all 4 strings, we have finished editing our list and can use it for constructing our pages.

Phase 9 - Setting Pipelines

Suppose we want to link the values of two variables by defining a mechanism whereby the value of one tag is continuously copied onto the other. In POLYMATH this can be done by defining a Pipeline.



Identification

Name	Pipeline
Comment	
Id	1

Parameters

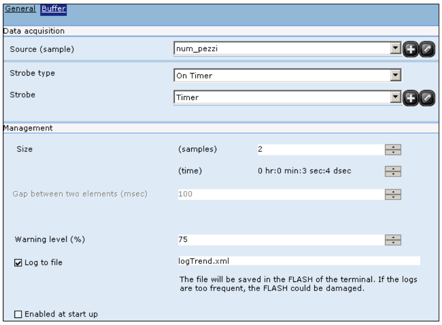
Source	num_pezzi
Destination	internal_var
Mode	Copy On Change
Activation	

After double-clicking on the ‘Pipeline’ option (double-click in Project Explorer), we click on ’Add’ to edit the pipeline created. We shall leave the name (‘Pipeline’) and the default ID (1) created by the application.
We use the lower part of the mask now to set the specific behavior of the Pipeline: the source variable for the value is

‘num_pezzi’, while that value is to be copied onto ‘internal_val’ (destination variable). In the third field we select the copy mode (CopyonChange, that is, the value is copied every time the value of the source variable changes).

Phase 10 - Defining a Trend Buffer

If we want to constantly monitor the progress of a variable, we can do this using Trend. This is a graphic object displaying the data relating to the sampling of the values assumed by a particular variable. The sample readings are saved in a memory called TrendBuffer.



Data acquisition

Source (sample): num_pezzi

Strobe type: On Timer

Strobe: Timer

Management

Size (samples): 2

(time): 0 hr:0 min:3 sec:4 dsec

Gap between two elements (msec): 100

Warning level (%): 75

☑ Log to file: logTrend.xml

The file will be saved in the FLASH of the terminal. If the logs are too frequent, the FLASH could be damaged.

☐ Enabled at start up

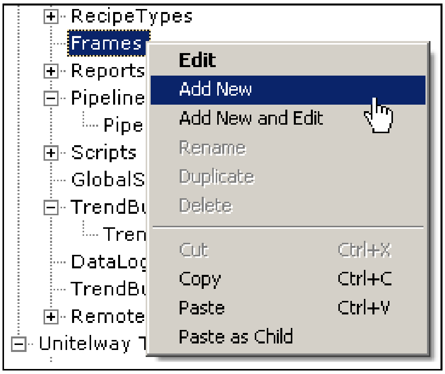
After double-clicking on the item ‘TrendBuffers’ (double-click in Project Explorer), we click on ‘Add’ to be able to edit the buffer created. We will leave the name (‘TrendBuffer’) and the default ID (1) created by the application in the General mask. In the Buffer mask, however, we set the options for how the buffer in question will operate. First of all we set as the source the variable to be monitored (‘num_pezzi’), while we select the acquisition mode OnTimer and assign to it the Timer (‘Timer’) we created in Phase 2. We must remember to change the Timer so that its event, OnTimerFired, has assigned to it the function AcquireSample for this TrendBuffer (the timer must also be made to start in runtime to enable the count. A good solution here is to assign a function or a Script to the opening of the initial page).
We will leave the general settings in the lower part unchanged but activate the log onto file option identifying the file ‘LogTrend.xml’ as the export file. In addition, we disable the automatic start up of the Trend at the beginning of the runtime (it must be enabled when necessary using a function or Script).
We have now defined the buffer of the trend we shall now insert into a page (by introducing a display field referring to this buffer).

Phase 11 - Graphicsetting, drawing a Frame

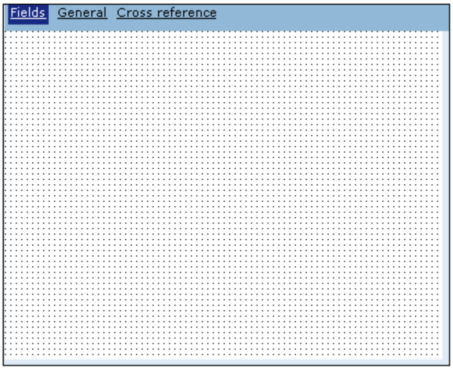
At this point of the work the functional structure of the project has been almost completely set. We now need only define the graphic presentation of the project in runtime. POLYMATH puts at our disposal essentially three presentation elements: classic full-screen pages, pop-up pages (pages that open on request overlapping full-screen pages) and frames (portions of a page common to a group of pages).
The interrelated use of these three elements allows complex and flexible configurations to be used that can meet every operational requirement.

Defining a frame

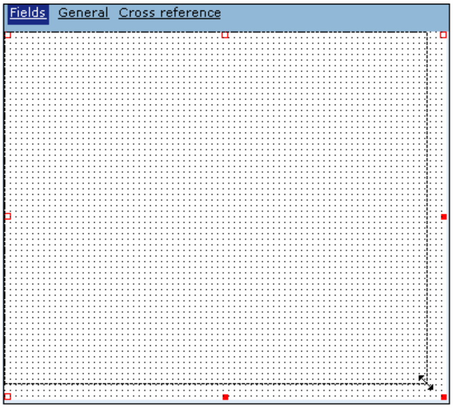
In our tutorial example we start with the definition of a frame that we then introduce into all the pages of our project. Basically, this frame will contain the buttons for navigating between the pages, a Quit project button and information regarding the current time.



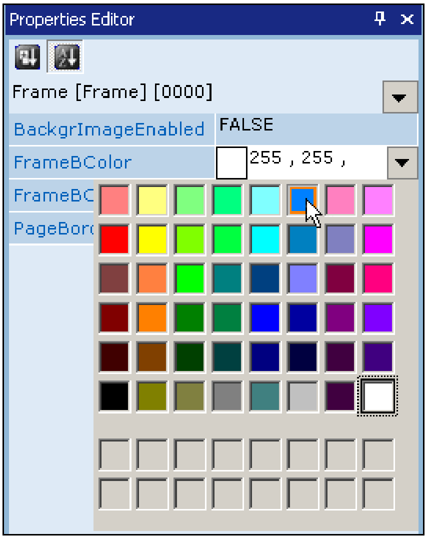
We will create our frame using Project Explorer as indicated in the above figure. By double-clicking on the frame created, the gridded editing page will open in which we can place the objects we want.



First we reduce our frame so that it becomes a horizontal bar. This we do by clicking on ▣ in the toolbar to select the frame and then by going to the red points and dragging the frame as shown below :



This way the frame assumes the size indicated. We can now define a background color by opening the Properties Editor. Within the Background option we select the color blue while leaving all the other options unchanged :



Properties Editor

Frame [Frame] [0000]

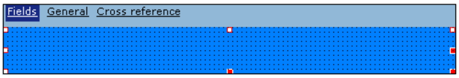
BackgrImageEnabled FALSE

FrameBColor 255 , 255 ,

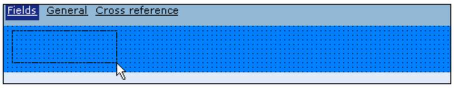
FrameBC

PageBor

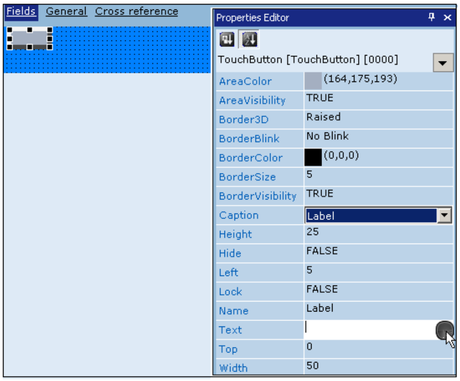
Our frame will thus appear as in the figure below and will be ready to accept objects placed within it :



We shall begin by introducing a button for navigating between pages, to be more precise, for displaying the previous page. To introduce a touch button, we click on in the applications bar and draw the outline inside the frame :



After selecting the new button, we go to Properties Editor and change certain graphic attributes relating to the button; first we set the size: the width at 50 pixels, the height at 25, horizontal position at 5 and vertical at 3.
Now we add the text to be seen on the button :

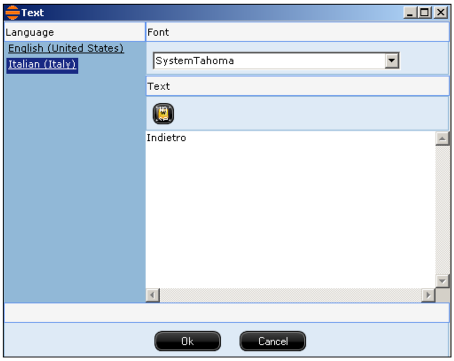


Properties Editor

TouchButton [TouchButton] [0000]

AreaColor	(164,175,193)
AreaVisibility	TRUE
Border3D	Raised
BorderBlink	No Blink
BorderColor	(0,0,0)
BorderSize	5
BorderVisibility	TRUE
Caption	Label
Height	25
Hide	FALSE
Left	5
Lock	FALSE
Name	Label
Text	
Top	0
Width	50

We choose label mode and click on .. to edit the translations of the text: for English we insert ‘Back’, for Italian ‘Indietro’ :



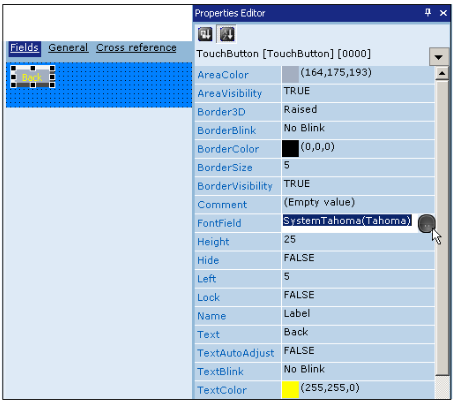
Text

Language: English (United States), Italian (Italy)

Font: SystemTahoma

Text: Indietro

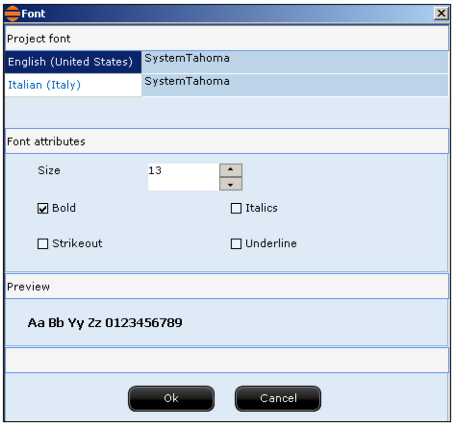
Once the texts have been defined, we edit their size and color. Continuing to work in Properties Editor, we set as text color Yellow and click on .. in the Font option to edit the character size :



TouchButton [TouchButton] [0000]

AreaColor	(164,175,193)
AreaVisibility	TRUE
Border3D	Raised
BorderBlink	No Blink
BorderColor	(0,0,0)
BorderSize	5
BorderVisibility	TRUE
Comment	(Empty value)
FontField	SystemTahoma(Tahoma)
Height	25
Hide	FALSE
Left	5
Lock	FALSE
Name	Label
Text	Back
TextAutoAdjust	FALSE
TextBlink	No Blink
TextColor	(255,255,0)

Using the editing window that now opens we choose the size of 13 and choose Bold from among the properties :



Project font

English (United States)	SystemTahoma
Italian (Italy)	SystemTahoma

Font attributes

Size 13

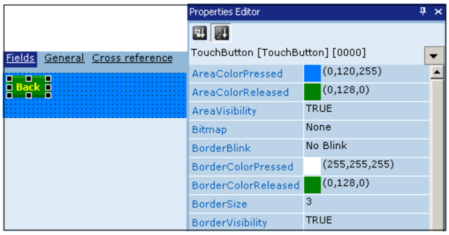
☑ Bold ☐ Italics

☐ Strikeout ☐ Underline

Preview

Aa Bb Yy Zz 0123456789

At this point the only thing left to do is to set the colors of the button: we select green as the background and border color when the button is released and blue with a white border when it is pressed. We also set the border width as 3 pixels.

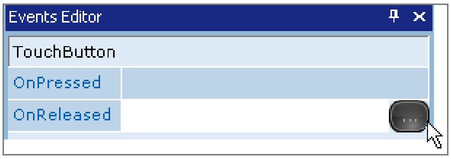


Properties Editor

TouchButton [TouchButton] [0000]

AreaColorPressed	🟦 (0,120,255)
AreaColorReleased	🟩 (0,128,0)
AreaVisibility	TRUE
Bitmap	None
BorderBlink	No Blink
BorderColorPressed	⬜ (255,255,255)
BorderColorReleased	🟩 (0,128,0)
BorderSize	3
BorderVisibility	TRUE

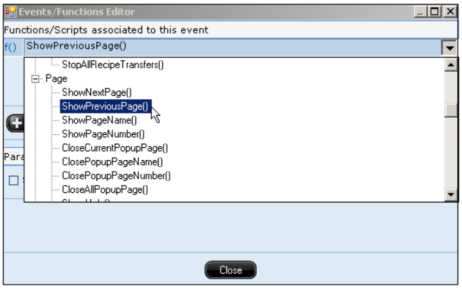
The last operation to be performed on the button is to define its function. For this we open the Events Editor and, while keeping the button selected, assign a function to the event OnReleased by clicking on ... ;



Events Editor

TouchButton
OnPressed
OnReleased ...

This opens the list of predefined functions (see "Appendix B - Predefined functions" page 563) from which we select the function Show previous page (after clicking on 'Add Function') as shown below :



Events/Functions Editor

Functions/Scripts associated to this event

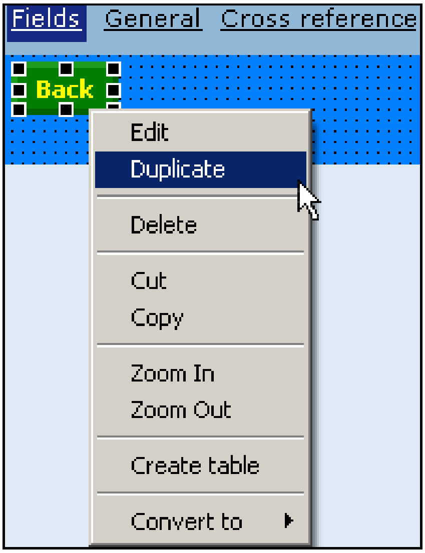
f() ShowPreviousPage()

StopAllRecipeTransfers()
Page
ShowNextPage()
ShowPreviousPage()
ShowPageName()
ShowPageNumber()
CloseCurrentPopupPage()
ClosePopupPageName()
ClosePopupPageNumber()
CloseAllPopupPage()

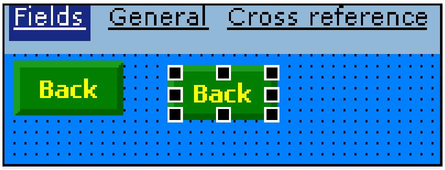
Once we have clicked on 'Close' in the window for assigning functions we have finished editing the button. Following the same procedure explained in this example we can edit all the buttons in our pages. We shall now see how to create other buttons using the work just done.

Duplicating buttons

So far we have only inserted one button into our frame, one with the function of invoking the previous page (the order followed is that of the IDs set for the pages). Now we can create a similar button but one with the opposite function, that is, show the next page. We need just duplicate the button already created to avoid edit again from square one; to do this we select the button with the right-hand key :

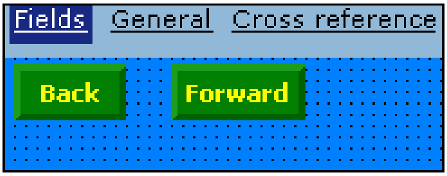


and Duplicate in the menu that appears: we now have two completely identical buttons.

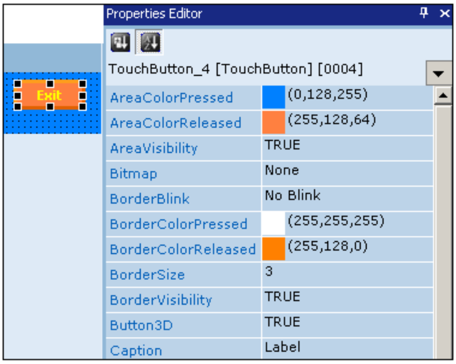


The new button is to differ from the first only in three aspects: the text of the label ("Forward" and "Continue" instead of "Back" and "Reverse"), the horizontal position (which can also be set by dragging the button to the right) and the reference function (using Events Editor "ShowNextPage" rather than

“ShowPreviousPage” can be set). In just a few steps we have inserted a second button :



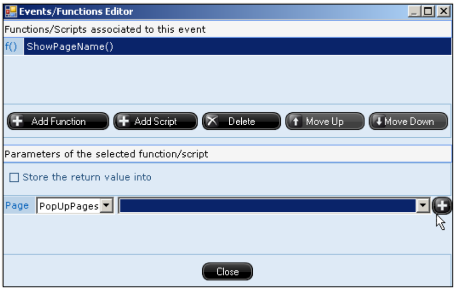
Using the same procedure, that is, duplicating and changing the function in the Events Editor and the translations of the label in the Properties Editor, we insert another 3 buttons: one for the login (“UserLogin” function), one for the logout (“User-Logout” function) and a last one for quitting runtime and set as indicated below.
After duplicating one of the already defined buttons, we use the Properties Editor to change the label text (we insert “Quit” and “Uscita” respectively for the two languages) and the color of the area when the button is released (we insert orange) as indicated below :



TouchButton_4 [TouchButton] [0004]

AreaColorPressed	🟦 (0,128,255)
AreaColorReleased	🟧 (255,128,64)
AreaVisibility	TRUE
Bitmap	None
BorderBlink	No Blink
BorderColorPressed	⬜ (255,255,255)
BorderColorReleased	🟧 (255,128,0)
BorderSize	3
BorderVisibility	TRUE
Button3D	TRUE
Caption	Label

We now move to the Events Editor and change the function assigned to the button as seen above: we choose the function ShowPageByName as shown in the figure below :



Events/Functions Editor

Functions/Scripts associated to this event

f() ShowPageName()

Parameters of the selected function/script

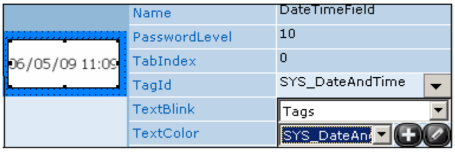
☐ Store the return value into

Page PopUpPages

The lower part of the mask contains a request to specify the page to be seen. In this phase we can also indicate a page that does not yet exist in the project but that we will edit later on. So we now select the option pop-up pages in the first pull-down menu, the second remains empty as no Pop-up pages exist in our job so far. At this point we click on ➕ to create one.

The page "PopUpPage" that was created with this operation will be edited in the next section when we deal with the pop-up pages.

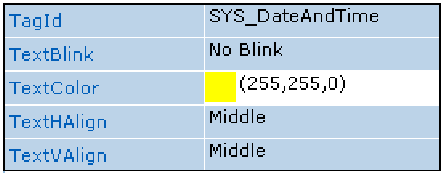
To finish editing our Frame, we insert a DateTime field within it so as to let the operator see the date and time at any point (assuming the frame will be added to every page of the project). We click on the 01:12 button of the toolbar and draw the outline of the field in the frame. Once the field has been inserted, we can begin editing its properties using the Properties Editor.



Name	DateTimeField
PasswordLevel	10
TabIndex	0
TagId	SYS_DateAndTime
TextBlink	Tags
TextColor	SYS_DateAn

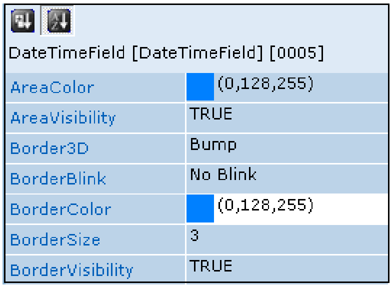
First of all we change the height of the field setting it at 25 pixels and assign the system variable SYS_DateAndTime (set in phase 3) to the attribute ID Variable by selecting it from the pull-down menu.

We can now edit the graphic aspect of the field, like text color: we select Yellow and attribute to the font a size of 15 pixels in Bold with the same procedure that we saw for the touch buttons.



TagId	SYS_DateAndTime
TextBlink	No Blink
TextColor	🟨 (255,255,0)
TextHAlign	Middle
TextVAlign	Middle

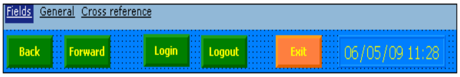
Finally we choose the color blue for the AreaColor and for the BorderColor; for the border we choose as a width dimension 3 pixels and as a style 3D “bump”.



DateTimeField [DateTimeField] [0005]

AreaColor	🟦 (0,128,255)
AreaVisibility	TRUE
Border3D	Bump
BorderBlink	No Blink
BorderColor	🟦 (0,128,255)
BorderSize	3
BorderVisibility	TRUE

The complete frame will look like this :

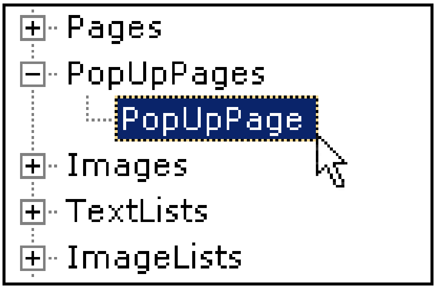


The advantages that derive from using frames are numerous, in particular:
- this portion of the page only needs to be edited once rather than having to re-edit every time its elements are to be present in a new page;
- editing one frame you can make changes to all the pages containing that frame (for example, if later you want to insert a new button in the frame, this will be present in all the pages containing the frame with just one operation).

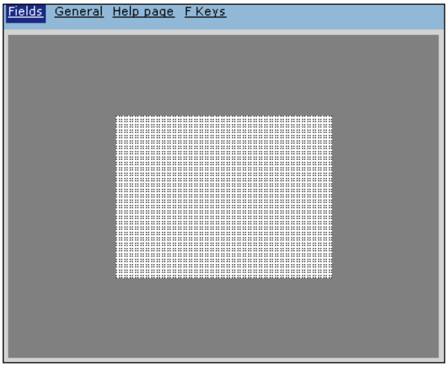
Phase 12 - Creating pop-up pages

Pop-up pages are pages overlapping with already opened (not yet closed) full screen pages. They are generally smaller than the complete page and are invoked by particular events (Scripts, pressing buttons, events assigned to variables, etc.). It is a good idea for the pop-up page to include the function relating to its closure (to avoid leaving Pop-ups open that might create confusion inside the project).

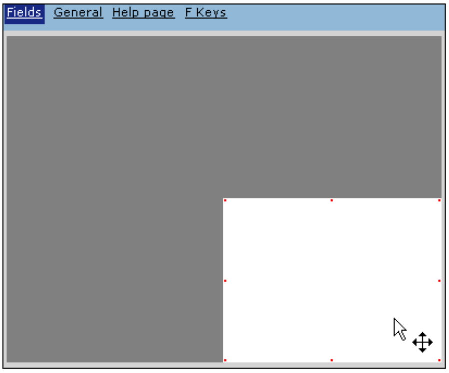
In our example we have created a Pop-up page (‘PopUpPage’) in the foregoing section; we associated its appearance with pressing the Quit button (the one identified by the color orange). Let us suppose that when this key is pressed, a mask for confirming the Quit operation appears (that is, a Pop-up). In effect, two items will appear: a label asking in the two languages of the project whether you wish to quit or not as well as two buttons, one for negating (associated with the function of closing the current pop-up) and one for confirming the Quit operation (associated with the predefined function QuitRuntime).
To be able to edit the pop-up created in the previous section, we scroll the Project Explorer list and find it under the option Pop-up pages.



Before beginning to edit the graphics of the pop-up, we access its general settings present in the General mask. The only change to be made in this mask is to disable the option ‘Show the title bar’ thus there will be no blue bar over the pop-up in runtime. The other options will be left unchanged.
If we return to the Fields mask, we will find a preview of what the pop-up will look like :

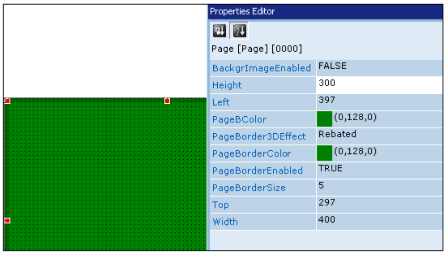


Unlike Frames, Pop-ups can be freely moved around the screen as well as resized. To move a pop-up select it by clicking on ◰ in the toolbar and drag it to the required area. In our example we will move the pop-up to the bottom right-hand corner of the page as shown below :



To make the editing of the graphics easier, we enlarge the pop-up preview by clicking on the zoom icons (100% ▾ or 🔍)in the toolbar.

We are now ready to edit the graphics of the pop-up. Repeating the procedure employed in the previous section for the button, we open the Properties Editor and change some of the options there. We enable the outline of the pop-up and set a size of 5 pixels; for the 3D effect we select Recessed and for the background color and the frame of the pop-up we select green :

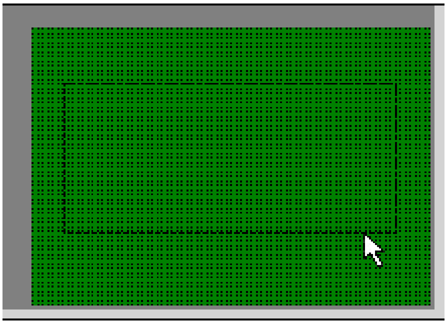


Properties Editor

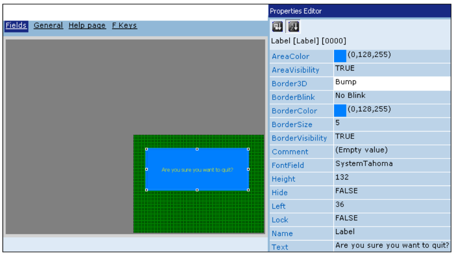
Page [Page] [0000]

BackgrImageEnabled	FALSE
Height	300
Left	397
PageBColor	(0,128,0)
PageBorder3DEffect	Rebated
PageBorderColor	(0,128,0)
PageBorderEnabled	TRUE
PageBorderSize	5
Top	297
Width	400

Our pop-up is now ready to accept objects placed in it. As already mentioned, we will start by inserting a label by clicking on A in the toolbar and drawing its outline inside the pop-up.



Using the Properties Editor we now assign the multilanguage text of the label ("Are you sure you want to quit?" and "Confirm exit from project?"), the font (30, yellow) and a color for the background and border of the label (both blue) as well as the border dimension of 5 pixels and the Bump 3D effect). Now the label looks like this :

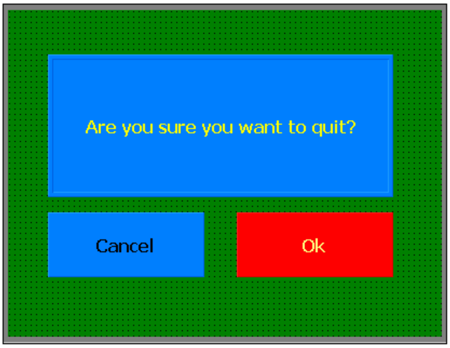


Properties Editor

Label [Label] [0000]

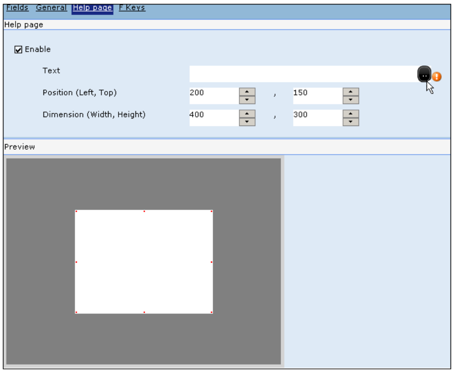
AreaColor	(0,128,255)
AreaVisibility	TRUE
Border3D	Bump
BorderBlink	No Blink
BorderColor	(0,128,255)
BorderSize	5
BorderVisibility	TRUE
Comment	(Empty value)
FontField	SystemTahoma
Height	132
Hide	FALSE
Left	36
Lock	FALSE
Name	Label
Text	Are you sure you want to quit?

The only things missing now are the buttons confirming or canceling the Quit command. We create these as set out in detail for the last parameter, remembering to assign the close current pop-up function to the Cancel key and the exit from runtime function to the Confirm key. In our example we create a blue key for cancelling (with a label saying “Cancel” and “Annulla”) and a red one for the confirmation. The resulting pop-up will be as follows (with the label saying “OK” and “Conferma”) :


Are you sure you want to quit?

Defining a Help-page for the Pop-up

In phase 2 we configured the project to manage the global keys, specifically, we stipulated that when the F1 key was pressed in any context of the project the Help relating to the page being displayed at that point would be shown.
We will therefore define a text to be displayed when the operator presses F1 with the current pop-up open. The Help-pages are edited while the page they refer to is being created: just move to the Help-pages mask to start editing.



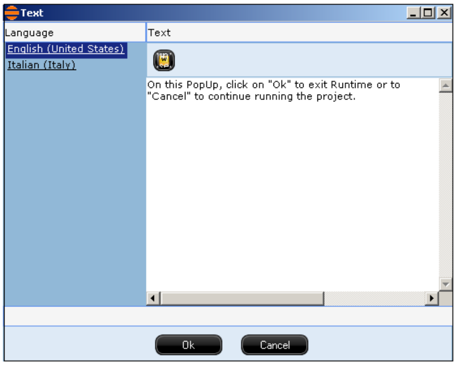
☑ Enable

Text

Position (Left, Top) 200 , 150

Dimension (Width, Height) 400 , 300

In this phase the dimensions and position of the Help-page can be defined: we shall leave the default values but change the text the operator will see. We click on … to start editing the text :



On this PopUp, click on "Ok" to exit Runtime or to "Cancel" to continue running the project.

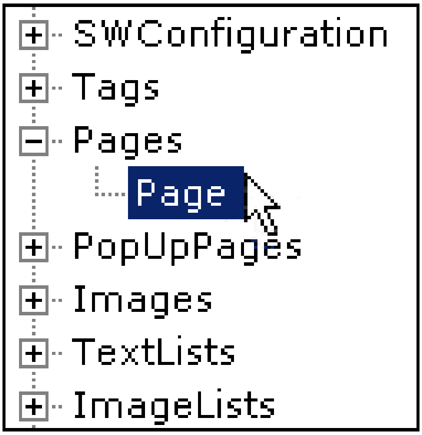
We insert the texts of advisory messages to be displayed in the page, providing, of course, translations in both the project’s languages. After clicking on OK, our Help-page is complete and with that our page pop-up, too.

Phase 13 - Drawing Full Screen pages

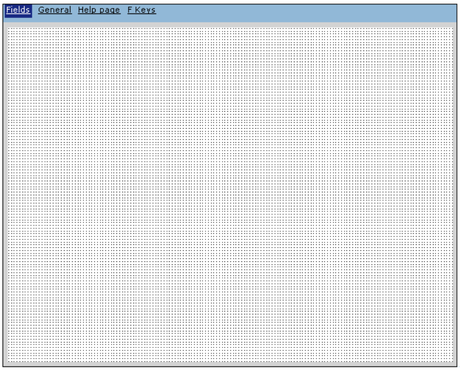
At this point in the project, the only thing left to do is define the number of pages and the way data can be accessed from them. In our example we first define the default page created in POLYMATH that we defined as the Start page of the project (see chap. 10, “Configuring the Boot” page 490) and then we go on to create pages that use complex controls.

Editing the Start page

When a new project is created, POLYMATH defines a default page in it. This is initially empty but can, naturally, be edited by the programmer who can also add an unlimited number of pages to the project. The ways of navigating within these pages are defined by the programmer using the many made available by the application (buttons with predefined functions, Scripts, user checks etc.).
To edit the default page, we double-click on it in the Project Explorer :



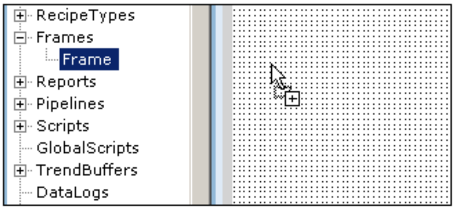
At this point the editing masks for the page appear in the work area. We move to the Fields mask that shows a preview of how the page will appear in runtime. Now just drag the object into the page required (the positions can be modified with greater precision by operating the Properties Editor for each individual object). Naturally, the first time the page preview is accessed it will appear completely empty :



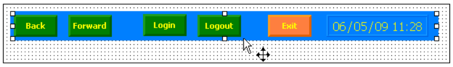
You can now decide whether to display or remove the grid inside the page by clicking on the ▦ icon in the toolbar.

Introducing a frame

We begin editing the page by introducing the frame we created in Phase 11: we select the frame using Project Explorer and drag it into the page as indicated in the figure below :



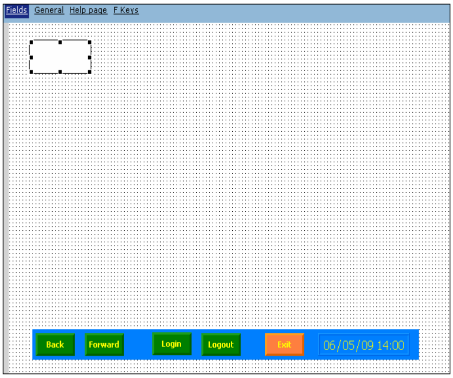
We now position the frame in the lower part of the page :



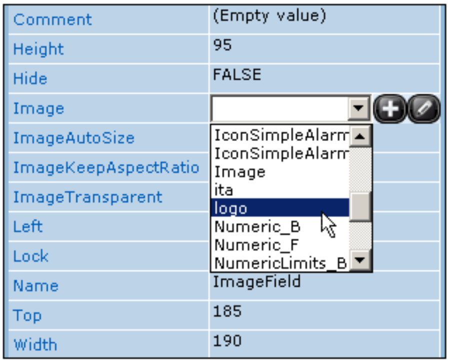
As we can see, all the buttons and graphic properties specified in the edit phase of the frame have been imported.

Introducing an image

Now we can introduce a second element into of our first page: we apply one of the images inserted in Phase 7, that is, the image saved in the project as 'logo'. We can do this in the same way that we inserted the frame, in other words by dragging (the quickest method) or we can use another procedure as set out below. In the toolbar we click on the image icon and trace the outline of the area that will take our image on the page :



We have now defined where the image is to be placed, so we can define which image to introduce. While keeping the area just defined selected, we move to the Properties Editor. Next to the Images option there is a pull-down menu containing all the images introduced into the project: we choose the image ‘logo’ :

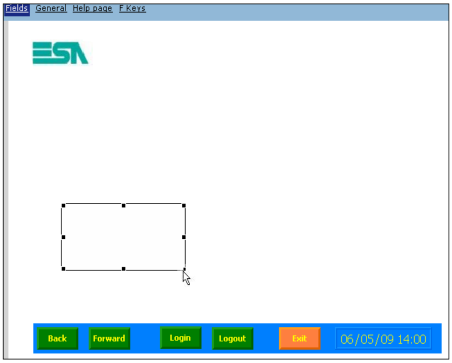


Comment	(Empty value)
Height	95
Hide	FALSE
Image	
ImageAutoSize	
ImageKeepAspectRatio	
ImageTransparent	
Left	
Lock	
Name	ImageField
Top	185
Width	190

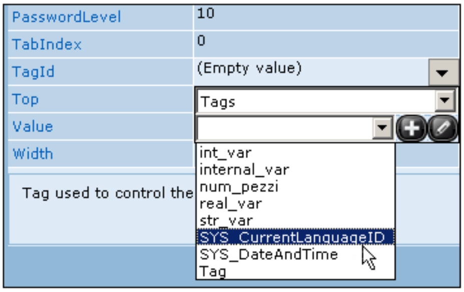
We can also change other properties of the image area: for example, we will set the border color as white so as not to see the edges of the image and have a more pleasing effect.

Inserting a symbol field

Now let us imagine we want to insert a symbol field (relating to a list of images) which will indicate in this page the language currently selected by the operator. We click on the ★ icon in the toolbar and draw the area that will take the field :



Once the perimeter of the area has been defined, we move to the Properties Editor in the usual way, indicating first of all the variable the field refers to (that is, the one whose value will be checked and in relation to which the image to be displayed will be chosen). We select the system variable relating to the ID of the current language (SYS_CurrentLanguageID, defined in Phase 3) as illustrated below :

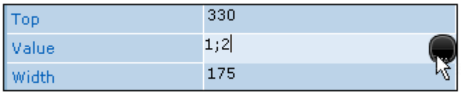


PasswordLevel	10
TabIndex	0
TagId	(Empty value)
Top	Tags
Value	
Width	

Tag used to control the

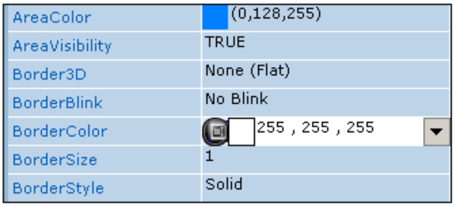
- int_var
- internal_var
- num_pezzi
- real_var
- str_var
- SYS_CurrentLanguageID
- SYS_DateAndTime
- Tag

We also assign as an image list the one created in the course of Phase 8 ('ImageList') and attribute the values relating to the images: we click on the ● key next to the Value option.



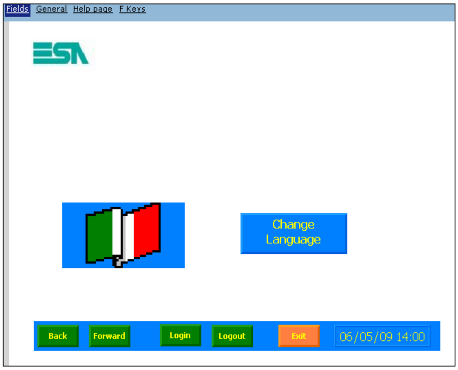
Top	330
Value	1;2
Width	175

In the window that opens we assign the value 1 for the image 'eng' and 2 for the image 'ita'. The IDs of the project languages are shown as these latter are created (Phase 2); take care that they correspond when assigned to items in an image list (or text list when necessary).
Finally we also change some graphic details like the color of the area (blue) and the border (white).



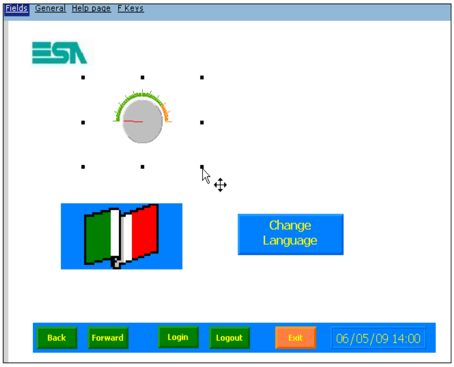
AreaColor	(0,128,255)
AreaVisibility	TRUE
Border3D	None (Flat)
BorderBlink	No Blink
BorderColor	255 , 255 , 255
BorderSize	1
BorderStyle	Solid

At the side of this symbol field we also put a button permitting us to change the display language of the project (by assigning the function ChangeNextLanguage). We dealt with how to edit a button in Phase 11, so now we follow the same steps to create the button for changing languages.

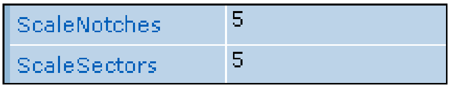


Introducing value indicators

We now give the page an object for indicating value for our variable 'num_pezzi'. As illustrated elsewhere in this manual, there is a set different objects for displaying/editing values. In our project we will insert a knob-potentiometer that allows us also to set the value of the variable to suit our requirements. We click on the icon relating to the knob-potentiometer in the toolbar and draw the destination area in the page. POLYMATH will draw the object in the page as can be seen from the following figure :

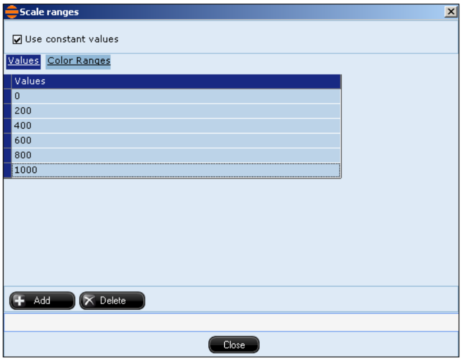


While keeping the potentiometer selected, we move to the Properties Editor and modify some attributes. First of all, we assign the variable 'num_pezzi' as reference variable (the one whose value will be displayed/edited); then we assign the number of values to be displayed on the bar (5) and the number of notches to display between the values (5).



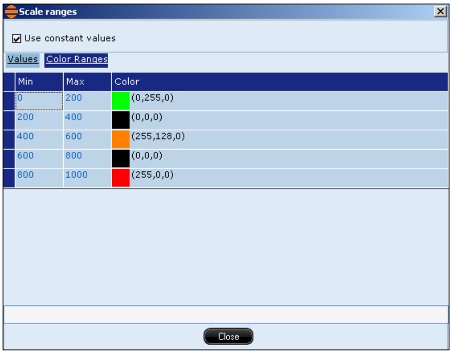
ScaleNotches	5
ScaleSectors	5

In addition we define the intervals of the scale and the related values by clicking on the … icon adjacent to the Color Intervals option. In the window that now appears, we click on ’Add’ to add new intervals and create 6 intervals with gaps of 200 per interval as in the figure below.



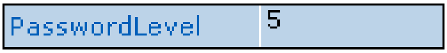
Values
0
200
400
600
800
1000

We then move to the Colors mask and set the color green for the low level, orange for the middle level and red for the highest level. Finally we click on ‘Close’ to confirm the changes made.



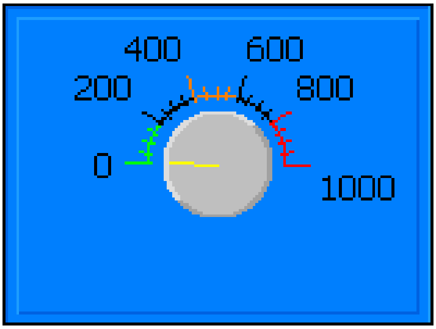
Min	Max	Color
0	200	🟩 (0,255,0)
200	400	⬛ (0,0,0)
400	600	🟧 (255,128,0)
600	800	⬛ (0,0,0)
800	1000	🟥 (255,0,0)

We must now perform an important operation, namely the setting of restrictions on the use of this potentiometer. Still positioned in the Properties Editor, we go to the Password Level option and enter the value 5 (for the user set in Phase 2).



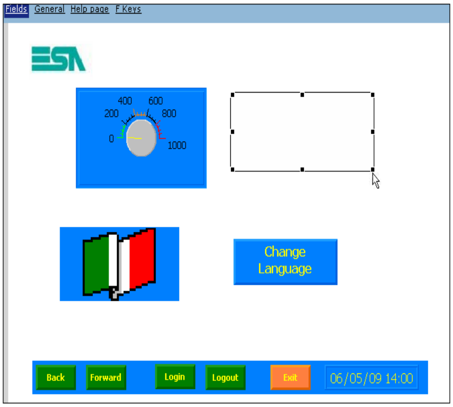
PasswordLevel 5

This setting means that the value can be edited by means of the potentiometer only by users who have logged in and whose user level has a value lower than or equal to 5 (thus, level 1 users can also edit). When the project starts, for example, the system gives the user level 10 until the log-in has been performed. This means that if a user who has not logged in (or with level greater than 5) tries to access the potentiometer (that is, tries to change the value of ‘num_pezzi’) the log-in window will automatically be displayed to make it possible to perform the operation.
We need now only define the graphic details (as we have just seen in the case of other objects) relating to the color of the area, of the border and of the indicator (needle) of the value. For example, in our project we set the color blue for the border and the internal area, 5 for the border size with 3D Etched as the style. The last step is to edit the indicator so that it will be displayed in yellow. The preview of how the potentiometer will appear is as below :

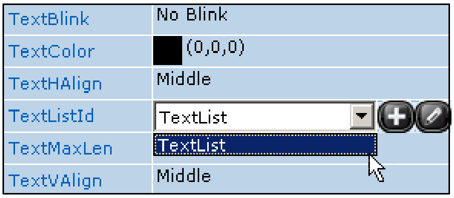


Setting a dynamic text

Suppose we want to relate a text to the value of the variable monitored by the potentiometer. To help us do this, POLYMATH allows us to insert dynamic texts in the project pages: the text displayed is chosen from a text list in runtime and it depends on the current value of reference variable.
To add a dynamic text to the page, we click on the A icon in the toolbar and draw the outline of the field in the page :

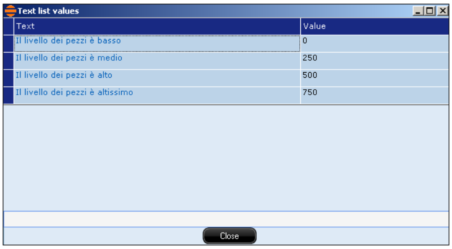


Selecting the field just created, we move to the Properties Editor and specify the check variable (‘num_pezzi’, the same one that the potentiometer is monitoring) and the appropriate text list (‘TextList’) :



TextBlink	No Blink
TextColor	■ (0,0,0)
TextHAlign	Middle
TextListId	TextList
TextMaxLen	TextList
TextVAlign	Middle

We now move to the Value option and click on … to start editing the values: we must specify a reference value for each option in the text list. The corresponding string will be displayed whenever the value of ‘num_pezzi’ reaches the exact value specified in this window :



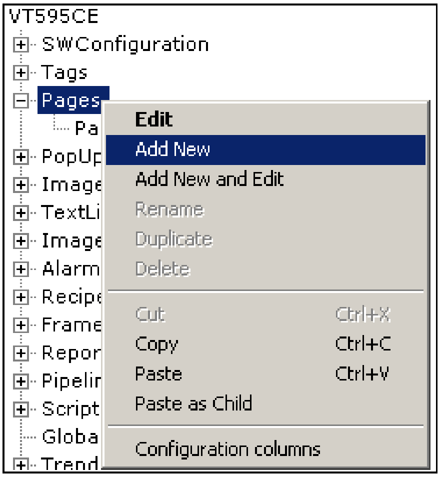
Text list values

Text	Value
Il livello dei pezzi è basso	0
Il livello dei pezzi è medio	250
Il livello dei pezzi è alto	500
Il livello dei pezzi è altissimo	750

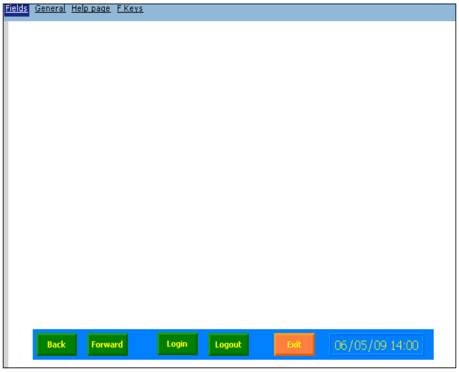
We also have to specify the graphic properties of the label relating to the dynamic text using the same methods as already seen for all the other objects we have added up to now (using the Properties Editor).

Phase 14 - Using complex controls

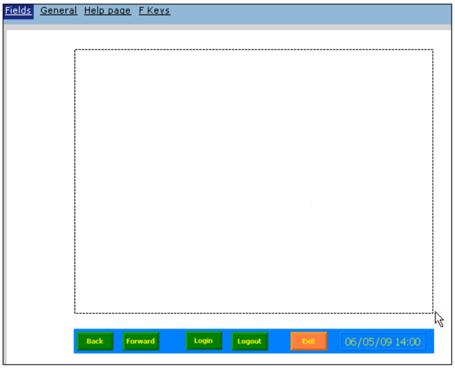
POLYMATH allows you to insert complex controls for managing elements like alarms, recipes, users and trends.
All these elements are edited in the same way, thus in this illustrative project we shall insert only one complex control: a recipe editor.
First of all we create a new page (click with right key on 'Pages' in Project Explorer, then on 'Add') and drag the frame created in Phase 11 into it.



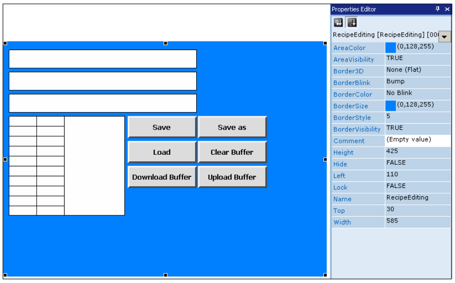
In this way all the tools defined in the frame are again made available to us in this page :



Starting from this empty page we can begin to insert our check window: to insert a recipe editor, we click on and draw the outline of the check window on the page :



POLYMATH will draw the recipe editor viewer with a standard layout. By selecting it we open the Properties Editor and set certain general graphic attributes: for example, we select a color, blue, for the area and border, the width of the latter being set at 5 and the type as Bump :



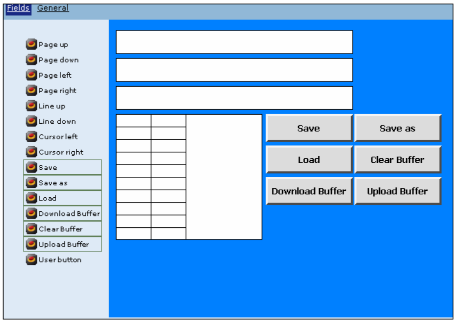
Properties Editor

RecipeEditing [RecipeEditing] [00(

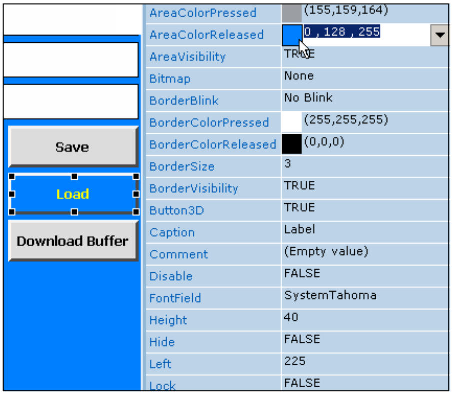
AreaColor	🟦 (0,128,255)
AreaVisibility	TRUE
Border3D	None (Flat)
BorderBlink	Bump
BorderColor	No Blink
BorderSize	🟦 (0,128,255)
BorderStyle	5
BorderVisibility	TRUE
Comment	(Empty value)
Height	425
Hide	FALSE
Left	110
Lock	FALSE
Name	RecipeEditing
Top	30
Width	585

Internal editing of complex controls

After introducing the recipe editor and defining its general graphic properties, we can start editing the internal components of the check window.
To perform this operation, we just double-click on the check window that has just been added to the page. The work area will show the Fields mask that is used to change the buttons and the fields and their characteristics (see below) :



As we can see, the elements making up this control area are: touch buttons, labels indicating the recipe type and a grid in which the instances of recipes are inserted in runtime. In the left section of the mask there is a list of buttons and labels that can be inserted into the viewer: all those already present are default elements so no change is made.
By selecting each internal object (button or label) in the Properties Editor we can change their respective properties, such as graphic attributes or access procedures. For example, we could enable the download button only for level 5 users or those with a lower level (higher priority) :

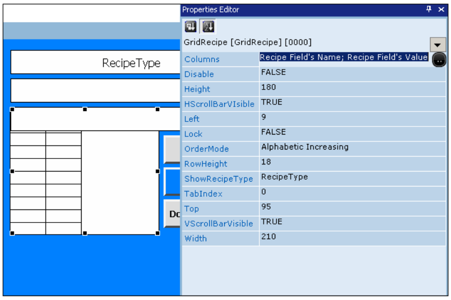


AreaColorPressed	(155,159,164)
AreaColorReleased	0 , 128 , 255
AreaVisibility	TRUE
Bitmap	None
BorderBlink	No Blink
BorderColorPressed	(255,255,255)
BorderColorReleased	(0,0,0)
BorderSize	3
BorderVisibility	TRUE
Button3D	TRUE
Caption	Label
Comment	(Empty value)
Disable	FALSE
FontField	SystemTahoma
Height	40
Hide	FALSE
Left	225
Lock	FALSE

While the check buttons are being edited, the only difference from standard buttons is that the events cannot be changed. With POLYMATH each already has assigned to it a predefined function).
The next section will describe how to edit the basic checking element, the table grid.

Editing the grid

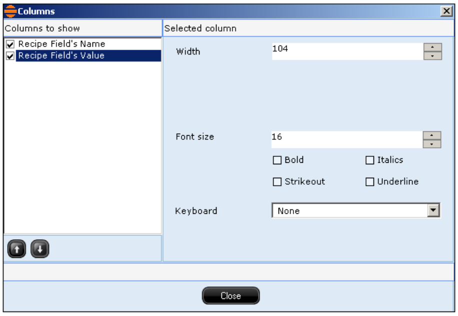
Remaining in the edit mask contained within the control area, select the grid and open the Properties Editor



GridRecipe [GridRecipe] [0000]

Columns	Recipe Field's Name; Recipe Field's Value
Disable	FALSE
Height	180
HScrollBarVIsible	TRUE
Left	9
Lock	FALSE
OrderMode	Alphabetic Increasing
RowHeight	18
ShowRecipeType	RecipeType
TabIndex	0
Top	95
VScrollBarVisible	TRUE
Width	210

We can now change properties like the height of the columns (we will set this at 25), the display of the vertical and horizontal scroll bars (we will leave both at TRUE) and the columns to be displayed in the table. We click on the … icon and a new editing window opens :



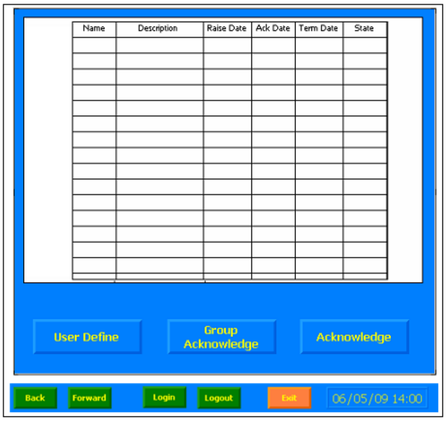
Columns to show
☑ Recipe Field's Name
☑ Recipe Field's Value

Selected column
Width 104
Font size 16
□ Bold □ Italics
□ Strikeout □ Underline
Keyboard None

The left-hand menu contains the list of columns to be seen in the table in runtime. In the case of the recipe editor, this list contains only two non-removable items (in general, the lists for other controls can be customized). Furthermore, if we select each column in this list, we can use the right-hand section of the mask to edit the values of column width and font properties for the table headings (titles). In our example, we set for both columns a width of 104 pixels and heading font size of 16 points. Finally we click on 'Close' to confirm the changes made.
Remember that the colors of the fields selected in the table in runtime can be changed in carrying out the general editing of the recipes (see chap. 5, "Fields" page 104).
In conclusion, using the last option in the Properties Editor (ShowRecipeType), we select the type of recipe the check refers to ('RecipeType').

Introducing other complex controls

Using the same procedure employed in introducing the recipe editor, we can introduce other complex controls in new pages (or even more than one in the same page). For example, in our project, we create a new page in which we insert an active alarms display table. Applying the same procedures as in the previous subsections we obtain the following result :

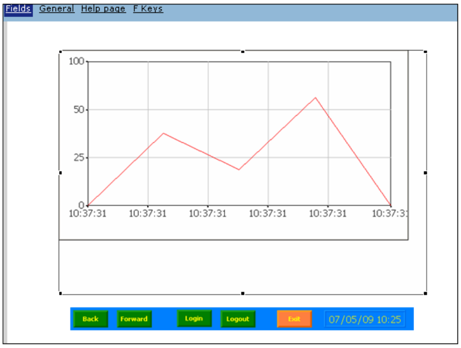


Name	Description	Raise Date	Ack Date	Term Date	State

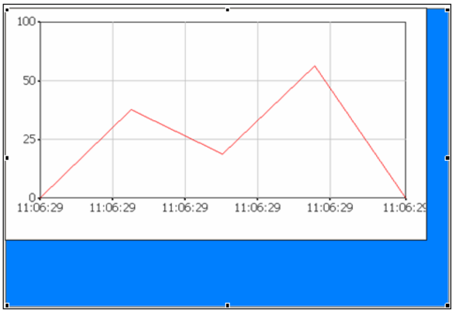
This we obtained by inserting the complex control, resizing it and eliminating certain default buttons. Furthermore, we have configured the control graphically so as to have a uniform setting within the project as a whole. The user can now practice inserting also instances of other complex controls (recipe list, alarm buffer and user table) all of which are edited in exactly the same way as we have just seen.

Phase 15 - Defining the Trend graph

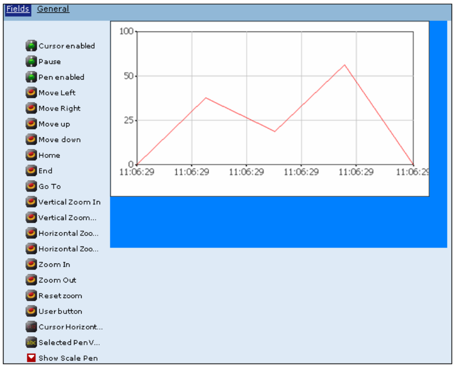
After inserting the complex controls we need into our project, we can add a page containing a trend, a chart displaying the behavior of a variable. Suppose that in our project we want to constantly monitor the situation of the variable ‘num_pezzi’ graphically. In Phase 10 we defined a Trend buffer for acquiring and memorizing value samples acquired by the variable. First of all we create a new page as in the previous phases and drag the navigation frame inside it.
Now we can really start inserting the graph: in the toolbar we click on and draw the space where our trend should appear in the page; when the mouse-key is released POLYMATH will draw the basic structure for us :



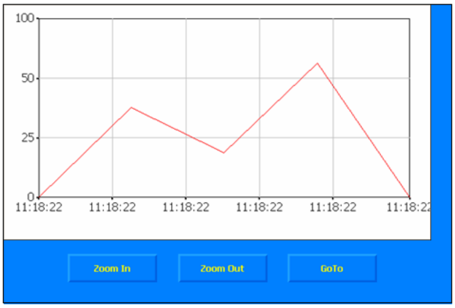
While keeping the graph we have just inserted selected, we move to the Properties Editor to edit the graphic properties of the trend area (employing the usual methods). To maintain a graphic unity with the rest of the project so far edited, we set blue as the color for the area and the border, which again is 5 pixels wide and in Bump style.



Internal editing takes place as with the other complex controls: just double-click on the area of the trend to be able to edit :

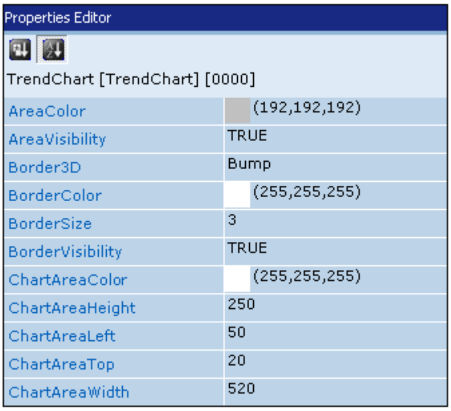


In this case, too, it will be possible to choose from the left-hand list the buttons and fields to be displayed on the Trend simply by clicking on them (if the button chosen has already been inserted, it will be removed).
For example, we insert the Zoom and GoTo buttons that allow us to select the position of the graph. For each of the buttons inserted we can edit their respective graphic properties, trying always to maintain a homogeneous style within the project.



Editing the Trend chart

We can now go on to edit the basic element of the trend viewer, the chart containing the graph. We select the chart and move to its Properties Editor. Using this window we can set all the characteristics relating to graphic representation, like colors and scale values, lines and subdivisions. As area color we choose gray and select a white, 3 pixel wide Bump border.

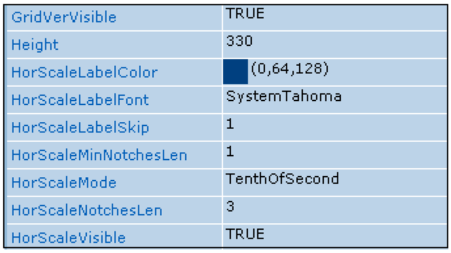


Properties Editor

TrendChart [TrendChart] [0000]

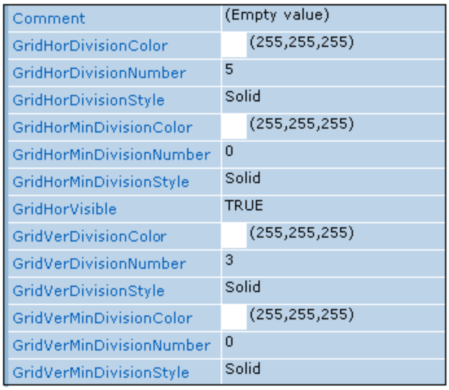
AreaColor	(192,192,192)
AreaVisibility	TRUE
Border3D	Bump
BorderColor	(255,255,255)
BorderSize	3
BorderVisibility	TRUE
ChartAreaColor	(255,255,255)
ChartAreaHeight	250
ChartAreaLeft	50
ChartAreaTop	20
ChartAreaWidth	520

We now change the representational properties of the horizontal value scale, choosing tenths of a second; we choose dark blue as the color and, finally, a value interval of 1 :



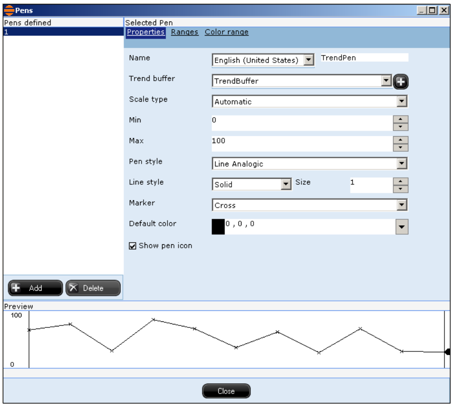
GridVerVisible	TRUE
Height	330
HorScaleLabelColor	(0,64,128)
HorScaleLabelFont	SystemTahoma
HorScaleLabelSkip	1
HorScaleMinNotchesLen	1
HorScaleMode	TenthOfSecond
HorScaleNotchesLen	3
HorScaleVisible	TRUE

We leave all the other values unchanged except for the colors of the dividing lines that we set as white :



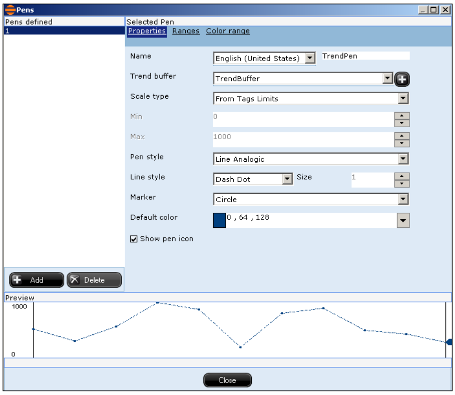
Comment	(Empty value)
GridHorDivisionColor	(255,255,255)
GridHorDivisionNumber	5
GridHorDivisionStyle	Solid
GridHorMinDivisionColor	(255,255,255)
GridHorMinDivisionNumber	0
GridHorMinDivisionStyle	Solid
GridHorVisible	TRUE
GridVerDivisionColor	(255,255,255)
GridVerDivisionNumber	3
GridVerDivisionStyle	Solid
GridVerMinDivisionColor	(255,255,255)
GridVerMinDivisionNumber	0
GridVerMinDivisionStyle	Solid

The final option requires great care. In the Pens field we click on the button and a new configuration window appears :



Name: English (United States) TrendPen
Trend buffer: TrendBuffer
Scale type: Automatic
Min: 0
Max: 100
Pen style: Line Analogic
Line style: Solid Size 1
Marker: Cross
Default color: 0 , 0 , 0
☑ Show pen icon

This new window contains the operating methods of the Trend pens, that is, the different ways the graphs and trend buffers can be drawn. The left-hand section of the mask is used to define any number of pens; in our project we shall edit simply the one created by default.
We can freely modify some of these properties: as type of scale we assign the limits of the variable (already defined in Phase 3 for the variable of the buffer, 'num_pezzi'). We also set the line-style as dash-dot, the marker as a little circle and its color to be dark blue :

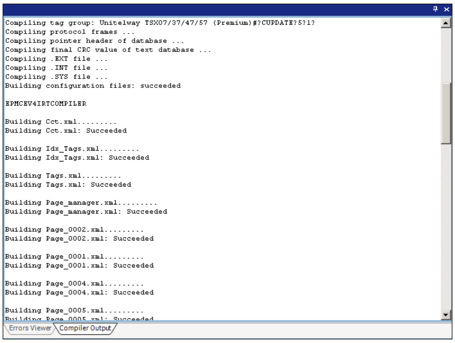


Name: English (United States) TrendPen
Trend buffer: TrendBuffer
Scale type: From Tags Limits
Min: 0
Max: 1000
Pen style: Line Analogic
Line style: Dash Dot Size 1
Marker: Circle
Default color: 0 , 64 , 128
☑ Show pen icon

At the bottom of the page we see an example of a preview of how the graph will appear. Finally we click on 'Close' to confirm our changes and thereby conclude the editing of the trend viewer.

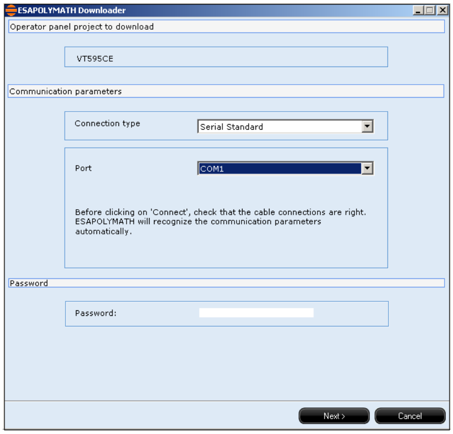
Phase 16 - Compilation and Download

We have now finished editing a simple project that uses all the basic functions offered by POLYMATH. At the end of this tutorial, the reader will be able to be more familiar with the application and ready to create projects with the sure knowledge of how to take full advantage of the numerous functions available.
Once the edit phase is over, before seeing the results of our work, we must compile the relevant files and download them onto the panel.
To start the compilation, in the toolbar click on the icon.
The compilation starts straight away and the messages relating to its status will appear in the Log View under the Compilation mask :



```
Compiling tag group: Unitelway TSX07/37/47/57 (Premium)#?CUPDATE?5?1?
Compiling protocol frames ...
Compiling pointer header of database ...
Compiling final CRC value of text database ...
Compiling .EXT file ...
Compiling .INT file ...
Compiling .SYS file ...
Building configuration files: succeeded

EPMCEV4IRTCOMPILER

Building Cct.xml.........
Building Cct.xml: Succeeded

Building Idx_Tags.xml.........
Building Idx_Tags.xml: Succeeded

Building Tags.xml.........
Building Tags.xml: Succeeded

Building Page_manager.xml.........
Building Page_manager.xml: Succeeded

Building Page_0002.xml.........
Building Page_0002.xml: Succeeded

Building Page_0001.xml.........
Building Page_0001.xml: Succeeded

Building Page_0004.xml.........
Building Page_0004.xml: Succeeded

Building Page_0005.xml.........
Building Page_0005.xml: Succeeded
```

As the instructions supplied in each phase have been correctly followed, no error nor warning message will appear (provided that we have also remembered to assign a Help-page for each page created).
With no error signaled in the compilation phase, we are ready to download onto the panel: we click on and POLYMATH proceeds by asking us for information regarding the connection between the PC and the terminal.



Operator panel project to download

VT595CE

Communication parameters

Connection type: Serial Standard

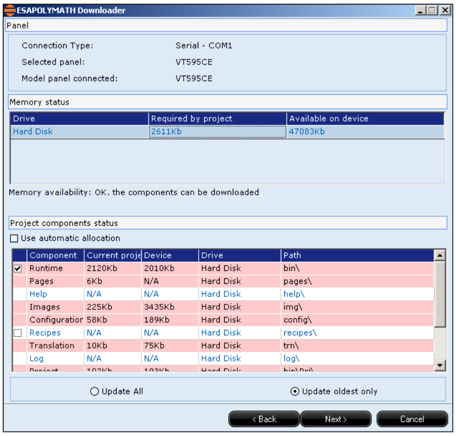
Port: COM1

Before clicking on 'Connect', check that the cable connections are right. ESAPOLYMATH will recognize the communication parameters automatically.

Password

Password:

In our example we have connected the terminal using the standard COM1 serial port, so we do not need to edit what is in the mask: VT model and connection mode. We click on ‘Connect’ after checking that the connection cable has been properly attached to the terminal and PC.
In this phase POLYMATH compares the versions of the project element on the panel and those to be downloaded. The next window shows us a summary of this comparison :



Panel

Connection Type: Serial - COM1
Selected panel: VT595CE
Model panel connected: VT595CE

Memory status

Drive	Required by project	Available on device
Hard Disk	2611Kb	47083Kb

Memory availability: OK. the components can be downloaded

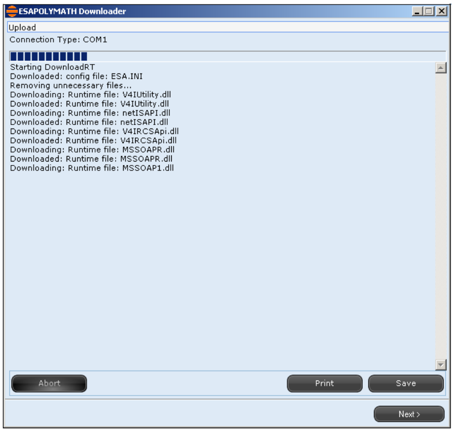
Project components status

☐ Use automatic allocation

	Component	Current proje	Device	Drive	Path
☑	Runtime	2120Kb	2010Kb	Hard Disk	bin\
	Pages	6Kb	N/A	Hard Disk	pages\
	Help	N/A	N/A	Hard Disk	help\
	Images	225Kb	3435Kb	Hard Disk	img\
	Configuratior	58Kb	189Kb	Hard Disk	config\
☐	Recipes	N/A	N/A	Hard Disk	recipes\
	Translation	10Kb	75Kb	Hard Disk	trn\
	Log	N/A	N/A	Hard Disk	log\

○ Update All ● Update oldest only

The parts needing updating are highlighted in pink. In addition, the support and the destination path of the files in the terminal can be changed. In our case we will leave everything unchanged and click on ‘Update only oldest’ to only update the project. If you also want to download the firmware for the first time, you are advised to click on ‘Update all’.



Connection Type: COM1

```
Starting DownloadRT
Downloaded: config file: ESA.INI
Removing unnecessary files...
Downloading: Runtime file: V4IUtility.dll
Downloaded: Runtime file: V4IUtility.dll
Downloading: Runtime file: netISAPI.dll
Downloaded: Runtime file: netISAPI.dll
Downloading: Runtime file: V4IRCSApi.dll
Downloaded: Runtime file: V4IRCSApi.dll
Downloading: Runtime file: MSSOAPR.dll
Downloaded: Runtime file: MSSOAPR.dll
Downloading: Runtime file: MSSOAP1.dll
```

A window reporting the download will then appear, which shows the status of the file transfer. Once this phase is over, the download is ended and the project we have just finished editing will start on the panel automatically.

11. Available functions for Remote connection from the PC

Remote Desktop

ESA puts at the user's disposal an application that can be bought separately (ESA order code: "PCREMOTEACCESS") or that can be installed with the POLYMATH 1.60 version.
The application which we will call "Remote Desktop" allows to have on the PC an identical vision of the terminal display where the project is found.

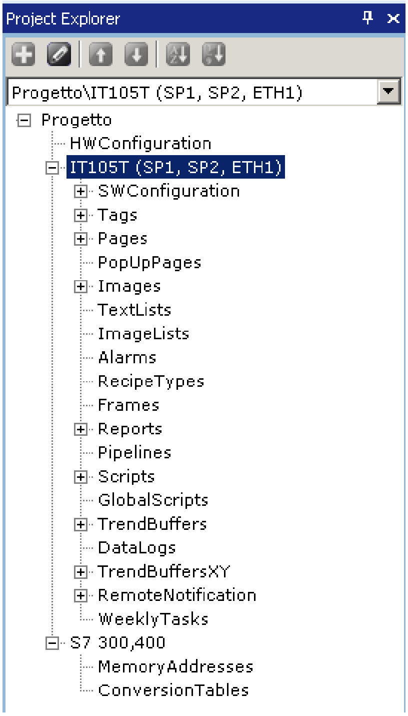
Installation and registration

To use the "Remote Desktop" function, the software called "ESAremote.exe" must be installed on the PC.
To install the application, follow the simple instructions which the guided installation proposes.
At the end of installation, a license code for carrying out the registration will be requested.
Registration is not obligatory, but if it is not carried out, product registration will be requested every time the application is used and connection to the terminal is carried out.

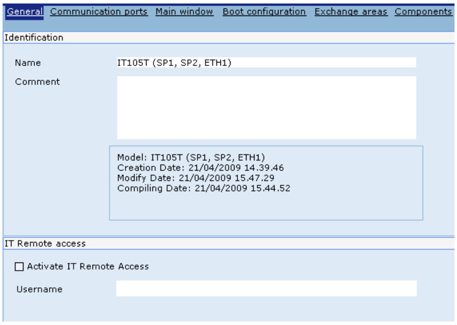
Available functions for Remote connection from the PC

"Remote Desktop" use

To use "Remote Desktop", do as follows:
From the "Explore Project" menu, double-click the product (in this case IT105T) :



the following image will appear :



Identification

Name IT105T (SP1, SP2, ETH1)

Comment

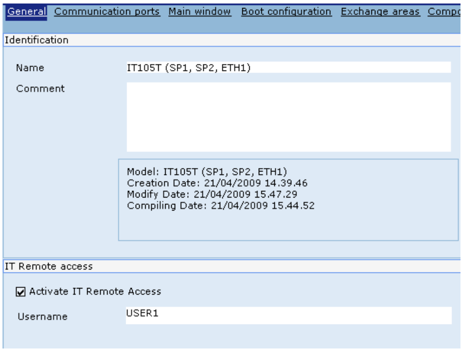
Model: IT105T (SP1, SP2, ETH1)
Creation Date: 21/04/2009 14.39.46
Modify Date: 21/04/2009 15.47.29
Compiling Date: 21/04/2009 15.44.52

IT Remote access

□ Activate IT Remote Access

Username

AEnable the "Activate Remote Access" function and assign a "User Name" (in our example we have inserted "USER1" in the "IT Panel Remote Access" window of the "General" mask) :



Identification

Name IT105T (SP1, SP2, ETH1)

Comment

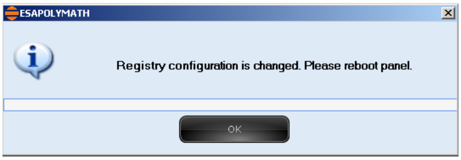
Model: IT105T (SP1, SP2, ETH1)
Creation Date: 21/04/2009 14.39.46
Modify Date: 21/04/2009 15.47.29
Compiling Date: 21/04/2009 15.44.52

IT Remote access

☑ Activate IT Remote Access

Username USER1

The first time the project is transferred with the "Remote Desktop" function enabled, the user will be requested to re-start the terminal :

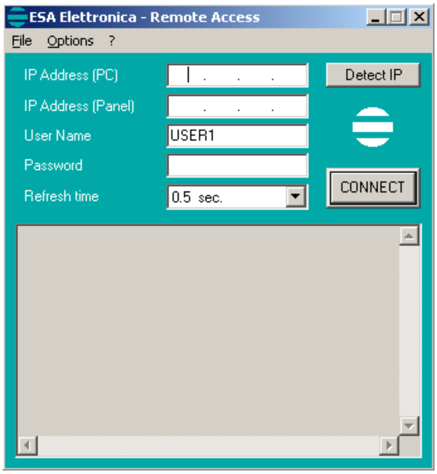


Registry configuration is changed. Please reboot panel.

- Connect a standard Ethernet cable (or the ESA CVNET11002 cable) between the terminal and the PC

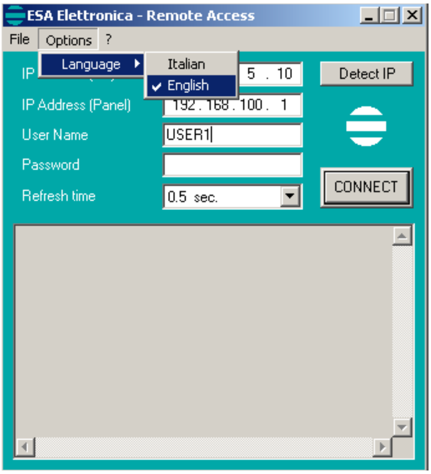
Available functions for Remote connection from the PC

- Launch the "ESA REMOTE.exe" application. The following image will appear :



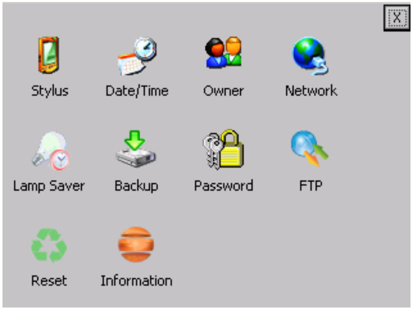
IP Address (PC): . . .
IP Address (Panel): . . .
User Name: USER1
Password:
Refresh time: 0.5 sec.

- Clicking on the "Options" key, the language with which the "Remote Desktop" software mask is displayed (English / Italian) can be chosen :

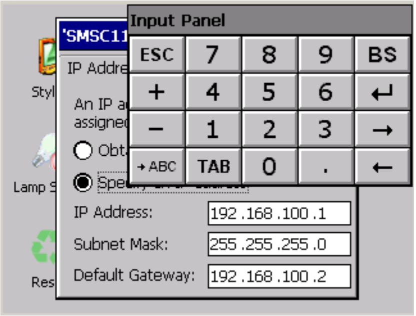


Language: ✓ English
IP Address (PC): 5 . 10
IP Address (Panel): 192 . 168 . 100 . 1
User Name: USER1
Password:
Refresh time: 0.5 sec.

- Carry out the registration on-line (if not already carried out) choosing the "Registration" option from the "?" menu.
- Set the '"PC IP address" clicking on the "Detect IP" key.
- From the IT control panel, clicK the ”Network” icon :



The following image will appear, from this image you can obtain the IT IP address (in our example the IP address is 192.168.100.1) :



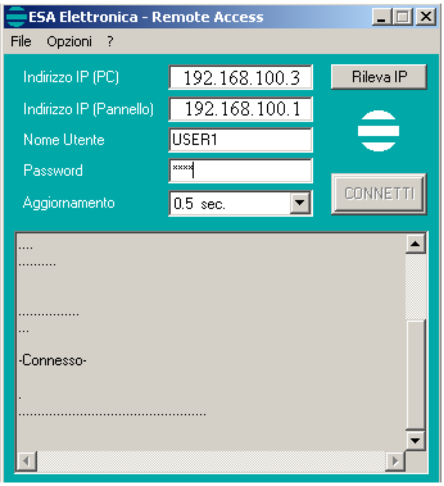
IP Address: 192 .168 .100 .1

Subnet Mask: 255 .255 .255 .0

Default Gateway: 192 .168 .100 .2

To establish the connection between PC and IT, the IP address of the terminal must be compatible with the IP address of the PC.

Available functions for Remote connection from the PC



Indirizzo IP (PC): 192.168.100.3

Indirizzo IP (Pannello): 192.168.100.1

Nome Utente: USER1

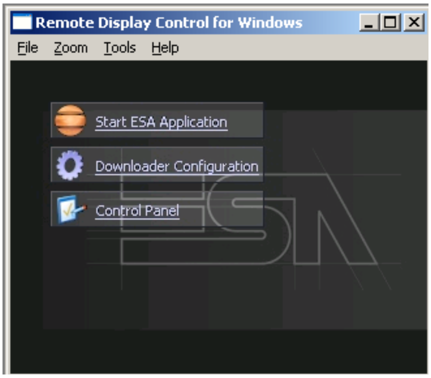
Password: ****

Aggiornamento: 0.5 sec.

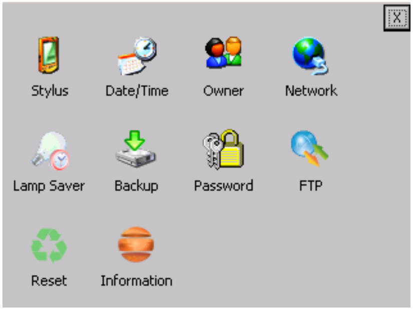
```
....
..........


...............
...

-Connesso-

.
..............................................
```

- Insert the same "User Name" which we assigned previously (USER1).
- The "PASSWORD" field can be left empty. Otherwise, to attribute a password,
- it must be assigned from the terminal :

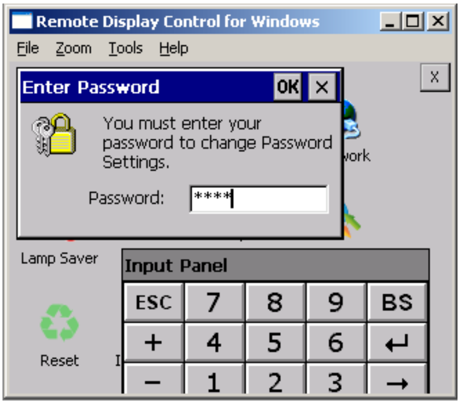
1 - Click "Control panel" :



2 - Click the "Password" icon :



3 - Set the password (for example 1234), then press "OK" :

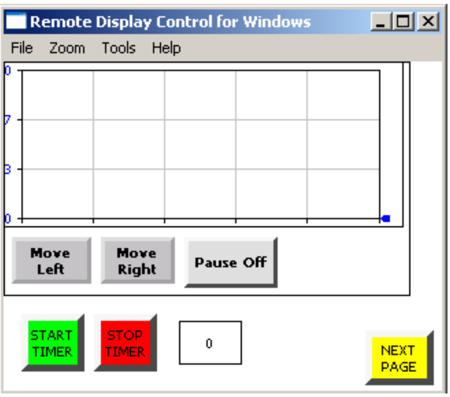


You must enter your password to change Password Settings.

Password: ****

- At this point, the "Password" field (on the PC) can be filled out inserting the same one set on the terminal (1234).
- Assign the "Refresh Time" (it establishes in how much time the image that appears on the PC video will be updated. Normally 0.5 sec. is a good solution).
- Click "Connect". The image displayed on the terminal will appear on the PC :

Available functions for Remote connection from the PC



Every operation carried out on the terminal is displayed on the PC and vice-versa.

Enable and disable "FTP"

FTP Server

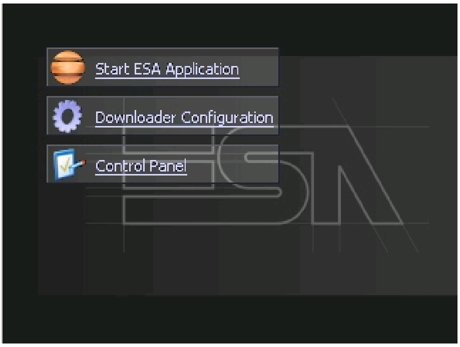
Another important function that ESA puts at the user's disposal (starting with the POLYMATH 1.60 version), is the "FTP Server".
The "FTP" acronym means "Files Transfer Protocol". It gives the user the possibility to enable and disable the "FTP Server" service of the panel from any other device (PC,XS,IT) connected to the network.
This function is very useful when it is necessary to write, cancel or modify data on the terminal easily from a remote access.

The remote access disks are the following :

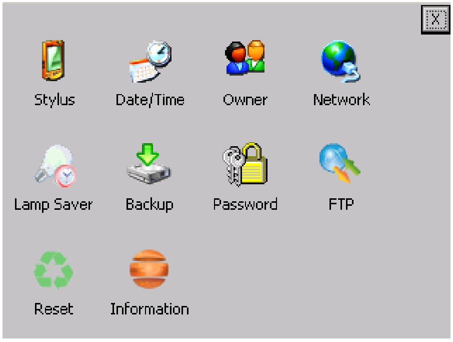
- My Device\Hard Disk\FTP (default folder)
- My Device\Hard Disk2 (if a "USB pen" is used as well)
- My Device\Storage Card (if a "Secure Digital" is used as well)

"FTP Server" features

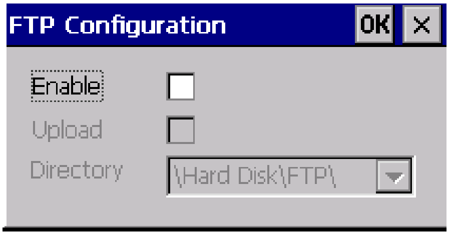
From the ESA terminal, click on "Control panel" :



The following image will appear :



Click on the "FTP" icon. The following image will appear :

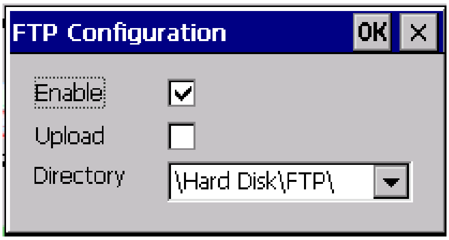

FTP Configuration OK ×

Enable □

Upload □

Directory \Hard Disk\FTP\

Selecting the "Enable" option, the "FTP" folder sharing service in the "Hard Disk" directory is enabled

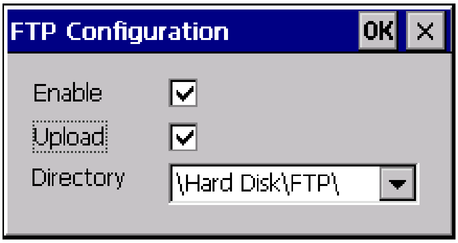

FTP Configuration OK ×

Enable ☑

Upload □

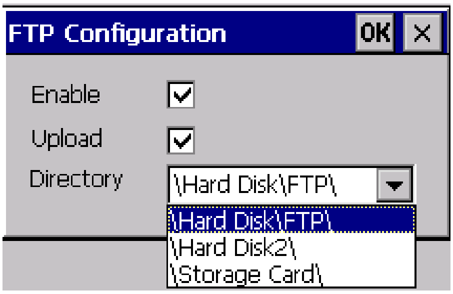
Directory \Hard Disk\FTP\

Selecting the "Upload" option, the "writing / modify" mode of the shared folders is enabled :



Enable ☑
Upload ☑
Directory \Hard Disk\FTP\

The 3 folders that can be used simultaneously are shown in the following image :



Enable ☑
Upload ☑
Directory \Hard Disk\FTP\
\Hard Disk\FTP\
\Hard Disk2\
\Storage Card\

In this way, besides the folder reserved by default (My Device\Hard Disk\FTP), a further memory space than can be used remotely can also be accessed.

At the end of the configurations just described, click "OK" to make them effective.

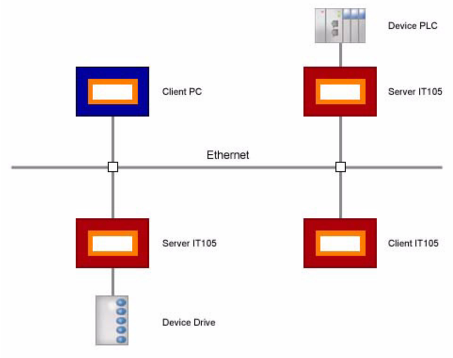
12. Panels network

In order to create a panel network, first create a number n of single products corresponding to the panels to be connected between them. Remember to make available the variables on the network then create a network project that incorporates each single project. Download must be carried out only from the network project until the projects are linked between them. If not, they will remain single and independent.

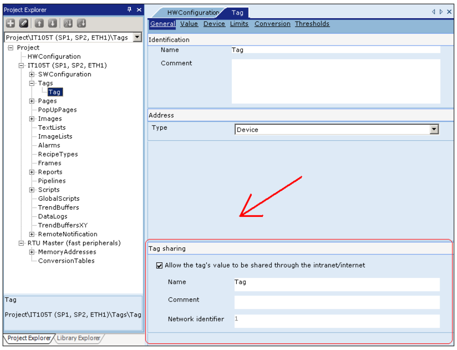
A detailed description of the procedure in order to create a panel's network follows.

Example creation of panel's network

Example of network layout between 2 server panels (those that make the variables available to various clients of the network) and 2 client panels.



Create a new project for the first server panel, connecting the desired device and develop it as per a normal project taking care to check the box "allow the tag value to be visible on the network".



Identification
Name: Tag
Comment:

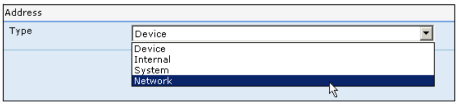
Address
Type: Device

Tag sharing
☑ Allow the tag's value to be shared through the intranet/internet
Name: Tag
Comment:
Network identifier: 1

Name: name of the variable visible on the network
Comment: a text can be inserted that comments the variable
Network identifier: non changeable progressive number that identifies the variable

When it has been created, save the project and using the same method, create the one for the second server.

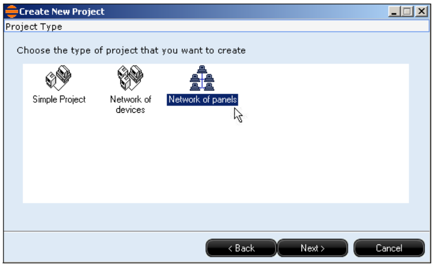
Now create the client project connecting the Tcp/IP http device that is found in the device list below: "Others - Esa Electronic" and develop it as a normal project. In the definition of the variables below "Address -Type" select the Network.


Address
Type: Device

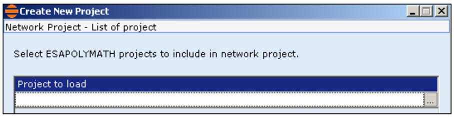
When creation is terminated, save the project and create in the same way, the one for the second client.

Create now a new project.

Select as project type: "Panels network".



Click on ... the three points at the end of the white box to add, by means of the normal window of Windows, the first project. Following click on Add and insert the other three projects developed previously.


Create New Project
Network Project - List of project
Select ESAPOLYMATH projects to include in network project.
Project to load
...

Complete the guided procedure as a normal project.

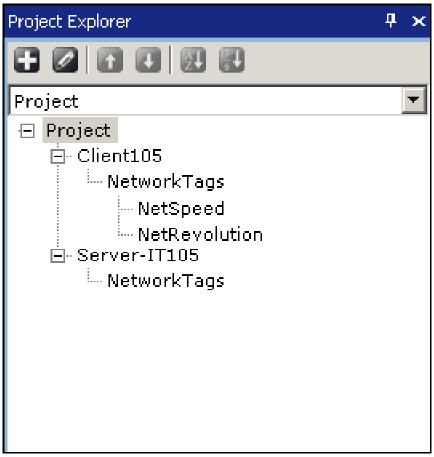
Double click on the name of the first panel (in this case "Client105") and by means of the section "Network" configure the network parameters. The network parameters configured in this section will over write those already configured in the panel. Therefore, they will be those to be used on the system. During “bench” test phase, use IP static addresses.t

The proxy supported by the network projects is HTTP type.

Note: *If the information is not recognised, contact the system administrator.*

Make sure that in the section "share tag" the check "Enable share tag device" is enabled
Shared tags password: a password can be determined to protect the shared tags
Shared tags gate: the gate for the shared tags can be determined

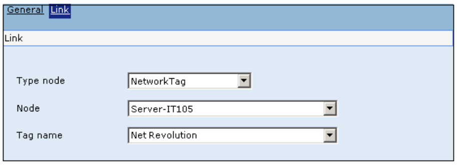
At this point, double click on the first client tag and using the section link associate the client tag with that of the server.



Project Explorer

Project

```
Project
  Client105
    NetworkTags
      NetSpeed
      NetRevolution
  Server-IT105
    NetworkTags
```



General Link

Link

Type node NetworkTag

Node Server-IT105

Tag name Net Revolution

Node type the type of node belonging to the server can be selected
Node: the server from which the tag will be chosen can be selected
Tag name: the tag to be associated can be selected.

When the above phases are terminated, save and compile the project after which proceed to download.

Download the network project

Start the compilation of the project by clicking on the icon of the Tool bar or from File->Compile on the main menu (see chap. capitolo 8, "Compiling, Downloading and Runtime" page 351).

Start transfer by clicking on the icon of the Tool bar or from the File->Download on the main menu. The network panel/ panels window will display towards which download will be carried out.

At the end of compilation, POLYMATH will display the window relative to the hardware configuration of the machine-terminal connection; select therefore, the type of connection between Ethernet -TCP/IP or USB.

Ethernet - TCP/IP connection

In the field "device address" and "Gate" insert the IP address and the panel gate whose name is indicated in the box above. Insert the password if it has been configured (see chap. capitolo 8, "Compiling, Downloading and Runtime" page 351). Click on forward and if the parameters have been configured correctly, a download window will be displayed. Proceed as for a single project(see chap. capitolo 8, "Compiling, Downloading and Runtime" page 351). When the first download is terminated, POLYMATH will return to the connection choice conditions (if in the project choice window there are two or more selected projects) . Insert the IP panel address whose name is indicated in the box above and download. Repeat operation until the last project

USB connection

If a USB connection is selected, connect the USB gate of the PC to that of the panel whose name is indicated in the box above. Insert the password if it has been configured (see chap. capitolo 8, "Compiling, Downloading and Runtime" page 351).
Click on forward and if the parameters and the connections have been configured correctly, a download window will be displayed. Proceed as for a single project (see chap. capitolo 8, "Compiling, Downloading and Runtime" page 351). When the first download is terminated, POLYMATH will return to the connection choice conditions (if in the project choice window there are two or more selected projects). Disconnect the panel where download has been completed and connect the second panel whose name is indicated in the box above. Switch off and on again the panel so that POLYMATH recognises the

second panel and download. Repeat operation until the last project.

13. Appendix A - System Variables

In this section we analyse the meaning of one particular type of variable, the system variable inside the terminal, which in RUNTIME is a read-only variable.
In general, these represent the operating status of the terminal and the project currently being executed.
System variables can be created in the project the same way as other variables and be managed and used in the same way (see chap. 5, “General”, pag. 90). The name of the default system variables begins with the prefix SYS_ followed by a string identifying its function.
To represent the system variables in the project, POLYMATH makes a system library available containing predefined pages for displaying this type of variable (see chap. 7, “System Library present in POLYMATH”, pag. 347).

Table 1: Meaning of System Variables

Variable	Description	Type
SYS_Machine Name	Name of terminal; for TCP/IP network terminals this always coincides with the network name of the terminal (e.g.: \\TermCE)	String
SYS_IPAddress	IP address of terminal	String
SYS_OSName	Operating system (e.g.: “CE4.2.”)	String
SYS_Screen_Hor_ Dim	Horizontal dimension of screen (pixels)	Int
SYS_Screen_Vert _Dim	Vertical dimension of screen (pixels)	Int
SYS_Project Version	Version of project; the string (never an empty string) has the following structure: “Vvv.rr dd-mm-yyyy” where: vv: version (from ‘01’) rr: release (from ‘00’) dd-mm-yyyy: release date (see chap. 4, “User Information”, pag. 53	String

Table 1: Meaning of System Variables

Variable	Description	Type
SYS_Author	Name of project author (see chap. 4, "User Information", pag. 53)	String
SYS_Author_Org	Name of organization to which the project author belongs (see chap. 4, "User Information", pag. 53)	String
SYS_Project_ Name	Name of the project (see chap. 4, "User Information", pag. 53)	String
SYS_AlarmPath	File path for alarm history	String
SYS_RecipePath	File path for the recipes	String
SYS_TrendPath	File path for trends	String
SYS_UsrLog	Log file path and name for user access	String
SYS_PageNum	Number of non POP UP project pages	Int
SYS_UserNum	Number of users configured	Int
SYS_TimerNun	Number of timers	Int
SYS_Pipelines Num	Number of pipelines in project	Int
SYS_PWDDefault	Default protection level (that is, with no user logged on)	Int
SYS_Font	Name of font (face_name) used as system font	String
SYS_Language Num	Number of languages configured	Int
SYS_LanguageX	With X being a value between 1 and 8 (inclusive); name of the Xth language configured	Int
SYS_CurrentPage	Name of current non pop-up page	String
SYS_Page	Name of focus page (including popup)	String
SYS_ShowFocus	TRUE if focus display is enabled, otherwise FALSE	Boolea n

Table 1: Meaning of System Variables

Variable	Description	Type
SYS_Script	Name of Script currently being executed (empty if none)	String
SYS_DateAnd Time	Date and time of system (format t_time Windows); VT settings using POLYMATH also make it possible to define the refresh frequency for this variable (see chap. 5, “Configuring the Boot”, pag. 76)	Long Int
SYS_AlarmNotOff	Number of active alarms not terminated in the system	Int
SYS_AlarmNot Ack	Number of active alarms not acquired in the system	Int
SYS_History Warning	TRUE if alarm history has reached the limit set in POLYMATH (see chap. 5, “Behaviour”, pag. 116)	Boolea n
SYS_HistoryFull	TRUE if alarm history has reached the maximum limit set in POLYMATH (see chap. 5, “Memory resources”, pag. 116)	Boolea n
SYS_BufferFull	TRUE if active alarm buffer has reached the maximum limit set in POLYMATH (see chap. 5, “Memory resources”, pag. 116)	Boolea n
SYS_AlarmNum	Total number of non-acknowledged active alarms in the system	Int
SYS_RecipeNum	Total number of recipes currently saved in the VT memory irrespective of their type	Int
SYS_RecipeXNum	With X being from 1 to the number of recipe types in the project; indicates the number of recipes of type X currently saved in the VT memory. There is one of these TAGs for each type defined in the project	Int
SYS_CurrentUser	Name of present user	String
SYS_CurrentLevel	Current level of protection (password)	Int

Table 1: Meaning of System Variables

Variable	Description	Type
SYS_Current Language	Name of current language	String
SYS_Current LanguageID	ID of current language	Int
SYS_TimerXYZ	Con XYZ being the name of the timer. Becomes TRUE when the timer XYZ is set off. There is a variable of this type for every Timer configured in the system	Boolea n
SYS_ContTimer XYZ	Con XYZ being the name of the timer. Indicates the current value of the XYZ. There is a variable of this type for every Timer configured in the system	Int
SYS_LastError Severity	Level of gravity of last error (0..2)	Int
SYS_LastError Module	Software module that generated last error (1..35>)	Int
SYS_LastError Message	Numerical ID of last error message	Int
SYS_LastError Text	Multilanguage string identifying last error message	String
SYS_ReportPage	Report page number	Int
SYS_ReportPages	Total number of report pages	Int
SYS_ReportName	Name of last current report	String
SYS_ReportPath	Directory of report destination	String
SYS_DM_Name	Name of the Device Manager to which the TAGs are connected	String
SYS_DM_Active	True if DM is active	Boolea na
SYS_DM_Error	Last error verified by the Device Manager	String
SYS_DM_DBName	Name of the DM configuration file (DEF/EXT)	String

Table 1: Meaning of System Variables

Variable	Description	Type
SYS_DM_Groups Num	Number of groups determined in the project	Long
SYS_DM_Items Num	Number of items determined in the project	Long
SYS_RCS_DB Name	Name of the configuration system's configuration file	String
SYS_RCS_Status	Operating status of the first communication card	Int
SYS_RCS_FW Name	Name of the first card's firmware file	String
SYS_RCS_FW Version	Version FW of the first communication card	String
SYS_RCS_Hw Version	Version HW of the first communication card	String
SYS_RCS_BT Version	Version BT of the first communication card	String
SYS_RCS2_Status	Operating status of the second communication card	Int
SYS_RCS2_Fw Name	Name of the second card's firmware file	String
SYS_RCS2_FW Version	Version FW of the second communication card	String
SYS_RCS2_HW Version	Version HW of the second communication card	String
SYS_RCS2_BT Version	Version BT of the second communication card	String
SYS_NATE_ Status	Operating status of the native ethernet gate	Int
SYS_NATE_FW Name	Name of the native ethernet gate's firmware file	String
SYS_NATE_FW Version	Version FW of the native ethernet gate	String
SYS_NATE_HW Version	Version HW of the native ethernet gate	String

Table 1: Meaning of System Variables

Variable	Description	Type
SYS_NATE_BT Version	Version BT of the native ethernet gate	String
SYS_COM1_ DriverName	Driver name on the first gate	String
SYS_COM1_ DriverStatus	Driver status on the first gate	String
SYS_COM1_ DriverPresent	TRUE if communication with the field on the first gate is active	Booleana
SYS_COM1_ DriverVersion	Driver version on the first gate	String
SYS_COM1_ DriverAddress	Terminal address on the first gate	String
SYS_COM2_ DriverName	Driver name on the second gate	String
SYS_COM2_ DriverStatus	Driver status on the second gate	String
SYS_COM2_ DriverPresent	TRUE if communication with the field on the second gate is active	Booleana
SYS_COM2_ DriverVersion	Driver version on the second gate	String
SYS_COM2_ DriverAddress	Terminal address on the second gate	String
SYS_ETH_Driver Name	Driver name on the ethernet gate (gate1 / logic1)	String
SYS_ETH_Driver Status	Driver status on the ethernet gate (gate1 / logic1)	String
SYS_ETH_Driver Present	True if communication with the field on the ethernet gate is active (gate1 / Logic1)	Booleana
SYS_ETH_Driver Version	Driver version on the ethernet gate (gate1 / logic1)	String
SYS_ETH_Driver Address	Terminal address on the ethernet gate (gate1 / logic1)	String
SYS_ETH2_Driver Name	Driver name on the ethernet gate (gate1 / logic 2)	String

Table 1: Meaning of System Variables

Variable	Description	Type
SYS_ETH2_Driver Status	Driver status on the ethernet gate (gate1 / logic2)	String
SYS_ETH2_Driver Present	True if communication with the field on the ethernet gate is active (gate1 / Logic2	Boolea na
SYS_ETH2_Driver Version	Driver version on the ethernet gate (gate1 / logic2)	String
SYS_ETH2_Driver Address	Terminal address on the ethernet gate (gate1 / logic2)	String
SYS_ETH3_Driver Name	Driver name on the ethernet gate (gate2 / logic 1)	String
SYS_ETH3_Driver Status	Driver status on the ethernet gate (gate2 / logic1)	String
SYS_ETH3_Driver Present	True if communication with the field on the ethernet gate is active (gate2 / Logic1	Boolea na
SYS_ETH3_Driver Version	Driver version on the ethernet gate (gate2 / logic1)	String
SYS_ETH3_Driver Address	Terminal address on the ethernet gate (gate2 / logic1)	String
SYS_ETH4_Driver Name	Driver name on the ethernet gate (gate2 / logic 2)	String
SYS_ETH4_Driver Status	Driver status on the ethernet gate (gate2/ logic2)	String
SYS_ETH4_Driver Present	True if communication with the field on the ethernet gate is active (gate2 / Logic2	Boolea na
SYS_ETH4_Driver Version	Driver version on the ethernet gate (gate2 / logic2)	String
SYS_ETH4_Driver Address	Terminal address on the ethernet gate (gate2 / logic2)	String

14. Appendix B - Predefined functions

This section is dedicated to the meanings of the predefined functions in POLYMATH that will prove useful during the development of a project. In general, they can be assigned to the events of the various POLYMATH objects (see chap. 6, "Events Editor" page 161) and can be selected from the relevant pull-down menu. For certain types of function it is also necessary to specify the variables or the objects that that function should effect and indicate the values with which it should operate.
A typical example of the use of predefined functions in POLYMATH is when they are assigned to touch buttons and touch areas when changing values of value fields or when opening and closing pages and pop-ups.

Functions relating to alarms

Tabella 1: Functions relating to alarms

Function	Description
ClearAlarmHistory	Cancels the buffer containing the alarm history; may be useful to insert a button with this function near an alarm history table (see chap. 6, "Alarm History View" page 302)
ExportAlarmHistory	Exports all alarms in the history to a file. The name of the destination file and its format (XML or CSV) need to be specified.
ExportActiveAlarms	Exports all active alarms in RUNTIME to a file. The name of the destination file and its format (XML or CSV) need to be specified.

Functions relating to users

Tabella 2: Functions relating to users

Function	Description
UserLogin	Makes it possible to invoke the user log-in operation (see chap. 5, "Password configuration" page 81). Makes the window for inserting the user name and password appear in RUNTIME.
UserLogout	Makes it possible to invoke the log-out operation. Makes a message of confirmation appear in RUNTIME. If confirmed, this operation takes the session-user to default status (can also be sent to a certain page each time the log-out operation is executed, see chap. 5, "Password configuration" page 81).
ChangeUserPassword	Changes the password of the user currently logged-in; has no effect if no user is logged on when pressed.

Functions relating to recipes

To have an overview of the way the operations performed by the following functions work, the reader is advised to consult the section of the manual dealing with the transfer of recipes between the terminal and the device (see chap. 6, "" page 314).

Tabella 3: Functions relating to recipes

Function	Description
LoadRecipe	Loads a recipe of a particular type. POLYMATH requires that the type of recipe (see chap. 5, "Creating and changing a Recipe type" page 126) to which this command refers be specified. The user is offered a choice between the list of available recipes in runtime.

Tabella 3: Functions relating to recipes

Function	Description
DownloadRecipe Buffer	Downloads the recipe buffer to the terminal. POLYMATH requires the type of recipe (see chap. 5, “Creating and changing a Recipe type” page 126) to which this command refers to be specified and whether the transfer should occur after synchronization or not. By pressing this key in runtime the buffer is downloaded to the terminal.
DownloadRecipe	Downloads one or more recipes to the terminal. POLYMATH requires the type of recipe (see chap. 5, “Creating and changing a Recipe type” page 126) to which this command refers to be specified and whether the transfer should occur after synchronization or not. By pressing in runtime the key associated with this function, the list of the recipes of the type defined is provided and the operator can choose which recipe to download.
SaveRecipeBuffer	Saves the recipe buffer; POLYMATH requires the type of recipe (see chap. 5, “Creating and changing a Recipe type” page 126) to which this command refers to be specified.
ClearRecipeBuffer	Cancels the buffer containing the recipes; may be useful to insert a button with this function near an alarm history table (see chap. 5, “Creating and changing a Recipe type” page 126).
DeleteRecipe	Cancels one or more recipes; POLYMATH requires the type of recipe (see chap. 5, “Creating and changing a Recipe type” page 126) to which this command refers to be specified. By pressing in runtime the key associated with this function, the list of the recipes of the type defined is provided and the operator can choose which recipe to download.

Tabella 3: Functions relating to recipes

Function	Description
DownloadRecipe Buffer	Loads the recipe buffer into the terminal; POLYMATH requires the type of recipe (see chap. 5, "Creating and changing a Recipe type" page 126) to which this command refers to be specified and whether the transfer should occur after synchronization or not.
ExportRecipe	Exports a recipe to a CSV or XML file on the terminal; POLYMATH requires the type of recipe (see chap. 5, "Creating and changing a Recipe type" page 126) to which this command refers to be specified. By pressing in runtime the key associated with this function, the list of the recipes of the type defined is provided and the operator can choose which recipe to export and (once this is selected) the name and path of the destination file.
ImportRecipes	Imports the recipes contained in a CSV or XML file on the terminal
ExportRecipeType	Makes it possible to export to a CSV or XML file all the recipes of a certain type; POLYMATH requires the type of recipe (see chap. 5, "Creating and changing a Recipe type" page 126) to which this command refers to be specified. The operator can indicate the name and path of the destination file in runtime.
ExportRecipeAllTypes	Makes it possible to export to a CSV or XML file all the recipes whatever their type. The operator can indicate the name and path of the destination file in runtime.
StopRecipeTransfer	Ends recipe transfer; POLYMATH requires the type of recipe (see chap. 5, "Creating and changing a Recipe type" page 126) to which this command refers to be specified

Tabella 3: Functions relating to recipes

Function	Description
StopAllRecipe Transfers	Interrupts all current recipe transfers.

Functions relating to pages

Tabella 4: Functions relating to pages

Function	Description
ShowNextPage	Shows next page (follows order of page ID numbers). If this command is on a Pop-up page, the next Pop-up page is shown
ShowPreviousPage	Shows previous page (follows order of page ID numbers). If this command is on a Pop-up page, the previous Pop-up page is shown.
ShowPageName	Displays page defined. the name of the page to be shown needs to be specified in POLYMATH.
ShowPageNumber	Displays page defined. the number of the page to be shown needs to be specified in POLYMATH.
ShowPageList	Shows a system page containing the list of project pages.
CloseCurrentPopup Page	Closes only the current pop-up page (with the command); must be assigned to an element or event of a pop-up page.
ClosePopupPage Name	Closes the pop-up page defined; the name of the page to be closed must be defined.
CloseAllPopupPage	Closes all the pop-up pages currently open in RUNTIME.

Tabella 4: Functions relating to pages

Function	Description
ClosePopupPage Number	Closes the pop-up page defined; the number of the page to be closed must be defined.
ShowHelp	Shows POLYMATH-defined Help relating to the page (full or pop-up) currently being displayed (see chap. 5, “Help pages” page 106 and see chap. 5, “Help pages” page 106)
ShowFocus	Shows the focus of the application (this function makes it possible to change the general settings relating to the focus in RUNTIME, see chap. 5, “Main window” page 75)
HideFocus	Hides the focus of the application (this function makes it possible to change the general settings relating to the focus in RUNTIME, see chap. 5, “Main window” page 75)

Functions relating to the project

Tabella 5: Functions relating to the project

Function	Description
ChangeNextLanguage	Changes the project language currently being used to the next one in the list defined in POLYMATH (see chap. 5, “Languages” page 78); all the elements subject to translation are displayed in the new language.
ChangeLanguage	Changes the project language currently being used to the defined one; all the elements subject to translation are displayed in the new language.
ExitRuntime	In RUNTIME this function exits from the project completely, returning the operator to the environment of the panel’s operating system.

Tabella 5: Functions relating to the project

Function	Description
Minimize	Reduces the project to an icon; the corresponding icon can be found in the applications bar.
Flush Persistent Data	Used to force the writing of the actual persistent-type internal Tag values
RunExcel	Used to launch the "Excel" ® application. The document to be opened can be indicated with the application (name and pathway)
RunInternetExplorer	Used to launch the "Internet Explorer" ® application
RunMediaPlayer	Used to launch the "Media Player" ® application. The document to be opened can be indicated with the application (name and pathway)
RunPDF	Used to launch the "Acrobat Reader" ® application. The document to be opened can be indicated with the application (name and pathway)
RunWord	Used to launch the "Word" ® application. The document to be opened can be indicated with the application (name and pathway)

Note: *The "Run" functions allow to launch applications such as "Excel" ®, "Internet Explorer" ®, "Media Player" ®, "Acrobat Reader" ® and "Word" ®. These functions are only available on ESA terminals with "Windows CE Professional Plus"® license (for example in code IT110T1112 the "Professional Plus" license can be identified by "1" in position "7" of the code).*

Functions relating to trends

Tabella 6: Functions relating to Trends

Function	Description
TrendAcquireSample	Performs a trend sample reading; the trend buffer the command works on must be given as an input parameter (see chap. 5, “Trend Buffers” page 142
TrendExport	Exports the trend indicated to a file; the relevant Trend Buffer and the name and type of destination file need to be defined,
TrendEnable	Enables acquisition of the trend indicator; the trend buffer the command relates to must be defined as an input parameter (see chap. 5, “Trend Buffers” page 142)
TrendDisable	Disables acquisition of the trend indicated; the trend buffer that the command relates to must be defined as an input parameter (see chap. 5, “Trend Buffers” page 142)
TrendReset	Clears the buffer of the trend indicated; the trend buffer that the command relates to must be defined as an input parameter (see chap. 5, “Trend Buffers” page 142)

Functions relating to direct commands

Tabella 7: Functions relating to direct commands

Function	Description
SetBit	Forces the value of a bit of a variable to a defined value; in POLYMATH the variable and the position of the bit to be forced need to be specified.
ResetBit	Allows the value of a bit to be reset; the variable to be reset and the position of the bit to be reset need to be defined.

Tabella 7: Functions relating to direct commands

Function	Description
ToggleBit	Inverts the value of a bit of a variable to a defined value; in POLYMATH the variable and the position of the bit to be inverted need to be specified.
SetValue	Forces the value of a variable to a defined value; in POLYMATH the variable and to be forced and the corresponding value need to be specified.
Add	Used to increase a variable by one value; must indicate the variable to which the command and the increase value should be applied.
Subtract	Used to decrease a variable by one value; must indicate the variable to which the command and the decrease value should be applied.
And	This executes a logical AND-operation on the binary representation of the values; must specify the variable on which to perform the operation and the value with which to execute the AND. The result of the operation will substitute the original value of the variable.
Or	This executes a logical OR-operation on the binary representation of the values; must specify the variable on which to perform the operation and the value with which to execute the OR. The result of the operation will substitute the original value of the variable.
Xor	This executes a logical XOR-operation on the binary representation of the values; must specify the variable on which to perform the operation and the value with which to execute the XOR. The result of the operation will substitute the original value of the variable.

Functions relating to pipelines

Tabella 8: Functions relating to pipelines

Function	Description
StartPipeline	Starts the pipeline defined according to the settings set out in the editor; in POLYMATH the name of the Pipeline to be started must be specified.
StopPipeline	Stops the pipeline defined from working; in POLYMATH the name of the Pipeline to be stopped must be specified.
WritePipeline	When this function is invoked the writing of a pipeline defined independently of its settings (the writing occurs even if the pipeline has been stopped) takes place.

Functions relating to timers

Tabella 9: Functions relating to timers

Function	Description
StartTimer	Starts the count of the selected Timers; need to specify the name of the timer to which the command refers (see chap. 5, “Timers” page 86).
StopTimer	Starts the count of the selected Timers; need to specify the name of the timer to which the command refers (see chap. 5, “Timers” page 86).
SuspendTimer	Momentarily suspends the count of the selected Timers; the count of the selected Timers; need to specify the name of the timer to which the command refers (see chap. 5, “Timers” page 86).
SetTimerValue	Set the value of the selected Timers; the count of the selected Timers; need to specify the name of the timer to which the command refers (see chap. 5, “Timers” page 86).

Functions relating to printing

Tabella 10: Functions relating to printing

Function	Description
PrinterSetup	This command brings up the print preferences window: to choose printer, format, etc.) in runtime.
HardCopy	This function makes it possible to print the contents of the current page (see chap. 5, “Points relating to print formats: XML and Hardcopy” page 137); need to specify if the print preference window, the print mode (1=page hardcopy, 2=full screen hardcopy) and the page orientation (horizontal or vertical) should be shown.
ReportPrint	Prints one of the Reports defined in the project; need to specify if the print preference window and the name of the Report to be printed should be shown.
ReportPrintSave	This function, apart from executing the print, saves the contents in an XML file; need to specify whether to show the print preference window, the name of the Report to be printed and the name and path of the XML file in which the contents of the Report will be saved (if the file already exists, the contents will be overwritten).
ReportSave	This function only saves the Report contents into an XML file; specify the name of the Report to be printed and the path and name of the XML file in which the contents of the Report will be saved (if the file already exists, the contents will be overwritten).

15. Appendix C - Status area

The terminal can be set to write information regarding its status and functioning onto defined memory areas. This information can be used by the device while it is carrying out its work. Unlike in the case of command areas, here the panel supplies information to the device. There are four types of status information that the terminal can write to these memory areas:

- status of VT: informs the device of the display and operating status of the terminal.
- status of keyboard
- status of recipes (new style)
- status of recipes (old style - VTWIN-compatible mode)

The memory area reserved for the status area will depend on the type of information to be supplied by the terminal: the VT status requires 6 Words, the Keyboard status 2 Words and the Recipes status areas are 2 Words and 1 Word respectively.
In POLYMATH, the status areas can be defined in the course of the general configuration of the panel (see chap. 5, “Exchange areas” a pag. 76).

VT Status area

The status area relating to the panel is composed of 6 words, each of which assumes a meaning in line with what is set out in the table below.

Tabella 1: Structure of VT status area

Word	Description
0	VT_STATUS: contains bit-coded status information (see chap. 15, “VT_STATUS values” a pag. 576)
1	SEQUENCE_ID: contains the numeric ID of the active sequence (including pop-ups) in focus. If no sequence is active the value is 0
2	PAGE_ID: contains the numeric ID of the page (including pop-ups) in focus (can never be 0 when the Runtime is active)

Tabella 1: Structure of VT status area

Word	Description
3	CONTEXT_VALUE: the value depends on the page/control in focus (see chap. 15, "CONTEXT_VALUE values" a pag. 577)
4	MAIN_SEQUENCE_ID: contains the numeric ID of the active non pop-up (or 'base') sequence. If no sequence is active the value is 0
5	MAIN_PAGE_ID: contains the numeric ID of the 'base' page currently being displayed (can never be 0 when the Runtime is active).

VT_STATUS values

Tabella 2: Meaning of VT_STATUS bit values

Bit	Description
0	WATCHDOG: in the course of normal working the VT sets the bit at 1. If from time to time the device sets it at 0, you can check the Runtime is active (in which case the VT will set it at 1 with a refresh period corresponding to the TAG-AREA)
1	EDITING_MODE: set at 1 when any active 'base' page field is in editing mode
2	ALARM_PRESENT: set at 1 when at least one alarm is active (whether recognised or not)
3	ALARM_PENDING: set at 1 when at least one alarm has not been acknowledged
4	COMMAND_NACK: set at 1 when a command from the device has not been accepted by the VT
5	ALARM_BUFFER_WLEVEL: set at 1 if the alarm history has reached its threshold (percentage determined by the maximum capacity available)

Tabella 2: Meaning of VT_STATUS bit values

Bit	Description
6	ALARM_BUFFER_FULL: set at 1 if the alarm buffer is full
7	N.U.: not used

CONTEXT_VALUE values

Tabella 3: Meaning of CONTEXT_VALUE bit values

Bit	Description
0	Default value
1	Focus is checking sequence directory or project pages
2	Displayed (and is focus) service/driver status page
3	Focus is a HELP page message
4	Focus is an alarm check
6	Focus is a recipe list check
8	Focus is a check of alarm history list

Keyboard status area

The status area relating to the keyboard is composed of 2 Words, making a total of 32 bits. Each bit corresponds to an F key on the keyboard where bit 0 is assigned to F1, bit 1 to F2 and so on for all the successive keys. The bits are set at 1 when the key is held down, 0 when released. The value of the bit simply reflects the status held-released (irrespective of any script or function assigned to the key) and if the keyboard is disconnected the value is at 0.

Status area of recipes - new style (non-compatible mode)

The status area for the recipes in non-compatible mode (see chap. 5, "Modes of compatibility" a pag. 125) is composed of 2 Words, each of which having a specific meaning:

- Word 0: status word containing the bits indicating the status of the transfer
- Word 1: contains the ID of the recipe to be transferred

The meanings of the bits of Word 0 are listed in the following table:

Recipe status word values

Tabella 4: Meaning of recipe status word values

Bit	Description
0	high bit (1) if transfer is underway
1	high bit (1) if transfer from panel to device has been requested
3	high bit (1) if transfer from panel to device has been completed
4	high bit (1) if transfer from device to panel has been requested
6	high bit (1) if transfer from device to panel has been completed
14	high bit (1) if there is an error in the transfer from panel to device
15	high bit (1) if there is an error in the transfer from device to panel

Status area of recipes - old style (compatible mode)

The status area for recipes in compatible mode (see chap. 5, "Modes of compatibility" a pag. 125) is composed of Word whose bits take on the following meanings:

Recipe status word values

Tabella 5: Meaning of recipe status word values s

Bit	Description
13	high bit (1) if there is an error in the transfer
14	high bit (1) if the transfer is underway
15	high bit (1) if there has been a transfer request

16. Appendix D - Command area

It is often necessary for the VTs in a plant to interact not only with the operators (by means of the appropriate peripheral devices like touch-screens and keyboards) but also with field devices, so that commands can be received and status information transmitted. This information exchange is carried out using special memory areas in the devices called Exchange areas.

These Exchange areas are, therefore, structures containing various types of information (whose meaning and format is set by the VT) which are regularly exchanged with the device. An exchange area is a tag-area (see chap. 5, “Value” a pag. 91) of one or more words residing in a field device. Command response areas (variables) can also be used by the VT to respond to a command sent by the device using the Command area.

To help set command areas POLYMATH has a dedicated section that can be reached using Project Explorer (see chap. 5, “Exchange areas” a pag. 76).

In this appendix we list the Command Areas that can be used by the device to change the operating status of the VT (that is, send commands).

The Command tag function and the Response tag have the same layout and are generally made up of four words:

Tabella 1: Command Tag Structure and Response Tag

Word	Description
0	COMMAND_ID: contains the code of the command requested/executed
1	PARAMETER_1: first parameter
2	PARAMETER_2: second parameter
3	PARAMETER_3: third parameter

The panel will execute the requested operation relative to the value of the Word corresponding to the COMMAND_ID and where necessary use the parameters indicated in the remaining 3 Words. The COMMAND_ID of the function, command area, is set at 0 by the VT when it is able to process a command (free area).

To send a command the device must:

- check that the COMMAND_ID is at 0
- compile the parameters
- set COMMAND_ID of the response tag at 0
- set the command in the COMMAND_ID.

The VT executes the command and when it has finished puts any parameters into the response area and then puts the command code executed into the COMMAND_ID of the response tag. In addition, it frees the command tag by putting 0 into its COMMAND_ID.

If the command cannot be executed or there are errors in any parameters, in the response tag the VT will put the value 0xFFFF (all 16 bits at 1) into the COMMAND_ID and puts the non executed command code into PARAMETER_1. It frees, however, the command tag by putting 0 into the COMMAND_ID.

A command response tag should be assigned to each device equipped with a command area.

The VT polls the command tags residing in the different devices, but always runs one command at a time, interrupting the polling while the command itself is run.

The table below shows the codes relating to the various commands that can be used (COMMAND_ID) and the respective parameters required for the execution.

Tabella 2: Command codes and parameters

ID	Description	Parameters
1	Forces sequence (non POP-UP); if page ID is 0 it starts from the first page. Not on Touch Screen panels	PARAMETER_1:sequence ID PARAMETER_2:page ID PARAMETER_3:
2	Forces page (non POP-UP), if a sequence is active, it is disabled	PARAMETER_1:page ID PARAMETER_2: PARAMETER_3:
3	Forces the cursor onto the current (non POP-UP) page in the field whose index tab is specified	PARAMETER_1:index tab PARAMETER_2: PARAMETER_3:
7	Sets the language indicated in PARAMETER_1	PARAMETER_1:language ID PARAMETER_2: PARAMETER_3:

Tabella 2: Command codes and parameters

ID	Description	Parameters
14	Asks for the current time (writes parameters onto the response tag, see next table)	PARAMETER_1: PARAMETER_2: PARAMETER_3:
15	Asks for the current date (writes parameters onto the response tag, see next table)	PARAMETER_1: PARAMETER_2: PARAMETER_3:
16	Sets time specified in parameters; parameters contain time in BCD with the format HHmmss00	PARAMETER_1: HHmm PARAMETER_2: ss00 PARAMETER_3:
17	Sets date specified in parameters; parameters contain date in BCD with the format DDMMYYYY	PARAMETER_1: DDMM PARAMETER_2: YYYY PARAMETER_3:
18	Reads sample (block) of trend buffer specified by the parameter	PARAMETER_1: trend ID PARAMETER_2: PARAMETER_3:
19	Clears (empties) alarm history	PARAMETER_1: PARAMETER_2: PARAMETER_3:
20	Recipe synchronization: syncro_cmd is bit-structured: bit 15: confirms transfer from VT to PLC bit 14: confirms end of transfer from VT to PLC bit 13: transfer time-out elapsed	PARAMETER_1: syncro_cmd PARAMETER_2: PARAMETER_3:
21	Recipe transfer request from VT to PLC. The first two parameters contain the name of the recipe (4 alphanumeric ASCII characters), parameter 3 is the identifier of the type of recipe. The command can only be used for compatible recipes (see chap. 5, “Modes of compatibility” a pag. 125)	PARAMETER_1: name (2 char) PARAMETER_2: name (2 char) PARAMETER_3: type_id

Tabella 2: Command codes and parameters

ID	Description	Parameters
22	Recipe sent from PLC to VT without overwriting. The first two parameters contain the name of the recipe (4 alphanumeric ASCII characters), parameter 3 is the identifier of the type of recipe. The command can only be used for compatible recipes (see chap. 5, “Modes of compatibility” a pag. 125)	PARAMETER_1: name (2 char) PARAMETER_2: name (2 char) PARAMETER_3: type_id
23	Recipe sent from PLC to VT with overwriting. The first two parameters contain the name of the recipe (4 alphanumeric ASCII characters), parameter 3 is the identifier of the type of recipe. The command can only be used for compatible recipes (see chap. 5, “Modes of compatibility” a pag. 125)	PARAMETER_1: name (2 char) PARAMETER_2: name (2 char) PARAMETER_3: type_id
26	Reads and writes the pipeline specified	PARAMETER_1: pipeline_id PARAMETER_2: PARAMETER_3:
27	Empties the trend buffer specified	PARAMETER_1: trend_id PARAMETER_2: PARAMETER_3:
28	Commands single sample of trend buffer specified	PARAMETER_1: trend ID PARAMETER_2: PARAMETER_3:
29	Stops sampling trend buffer specified	PARAMETER_1: trend ID PARAMETER_2: PARAMETER_3:
30	Starts trend buffer specified	PARAMETER_1: trend ID PARAMETER_2: PARAMETER_3:
35	Commands printing of report specified	PARAMETER_1: report ID PARAMETER_2: PARAMETER_3:

Tabella 2: Command codes and parameters

ID	Description	Parameters
36	Requests printing of alarm history	PARAMETER_1: PARAMETER_2: PARAMETER_3:
37	Requests (HARDCOPY) printing of the screen; if text mode flag is at 1 printing will be in text mode, otherwise in graphic mode	PARAMETER_1: text mode flag PARAMETER_2: PARAMETER_3:
38	Forces printer Form Feed	PARAMETER_1: PARAMETER_2: PARAMETER_3:
39	Resets numbering of print pages	PARAMETER_1: PARAMETER_2: PARAMETER_3:
43	Global alarm acknowledgement	PARAMETER_1: PARAMETER_2: PARAMETER_3:
46	Requests disabling (if flag is at zero) or enabling (if flag is at 1) of the touch screen: if disabled, il terminal does not respond to the 'touch'	PARAMETER_1: flag PARAMETER_2: PARAMETER_3:
50	Requests transfer of recipe from VT to PLC. Parameter 1 contains the ID of the recipe to be transferred while parameter 2 has the identifier of the recipe type	PARAMETER_1: recipe_id PARAMETER_2: type_id PARAMETER_3:
51	Sending recipe from PLC to VT with overwriting. Parameter 1 contains the ID of the recipe to be transferred while parameter 2 has the identifier of the recipe type	PARAMETER_1: recipe_id PARAMETER_2: type_id PARAMETER_3:

Commands number 14 and 15 require data being written onto the response tag as indicated in the next table:

Tabella 3: Response Tag codes and parameters

ID	Description	Parameters
14	Current time: the parameters contain time in BCD with the format HHmmss00	PARAMETER_1: HHmm PARAMETER_2: ss00 PARAMETER_3:
15	Current date: the parameters contain date in BCD with the format DDMMYYYY	PARAMETER_1: DDMM PARAMETER_2: YYYY PARAMETER_3:

Command area for New Style (non compatible) recipes

In the case of non compatible recipes (see chap. 5, “Modes of compatibility” a pag. 125) a 2-Word command area is used in which the first Word indicates the command that the terminal must execute while the second Word indicates the ID of the recipe that has to be transferred.
The meanings of the commands of Word 0 are listed in the table below:

Tabella 4: Meanings of Word 0 bits of the Command area for non-compatible recipes

Bit	Description
0	If the bit is high (1) it indicates confirmation for transfer from panel to device
1	If the bit is high (1) it indicates confirmation for transfer from device to panel
3	If the bit is high (1) indicates request for non-synchronized transfer from panel to device
4	If the bit is high (1) indicates request for synchronized transfer from panel to device

Command area for Old style (compatible) recipes

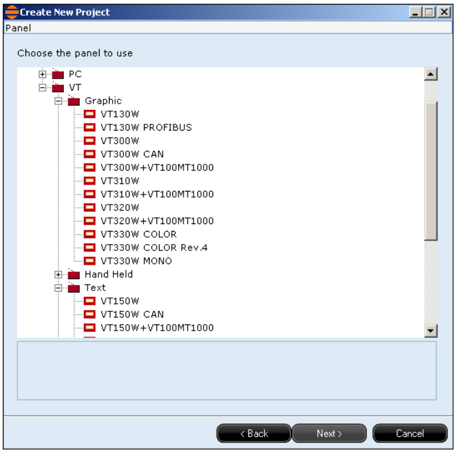
In the case of recipes configured as compatible with the old style it is not necessary to define a dedicated Command area, rather the Command area defined for the project will be used with particular reference to commands 20,21,22 and 23 already described in the first part of this chapter.

17. Appendix E - VTxxxW Panels Management

In this manual reference has been made to programming the terminals of the VTxxxCE range and the IT range. POLYMATH however offers the possibility also to create and manage projects relative to the products in the VTxxxW range. It is possible to create new projects, again or open projects directly realised with ESA VTWIN application and with .vts extension.

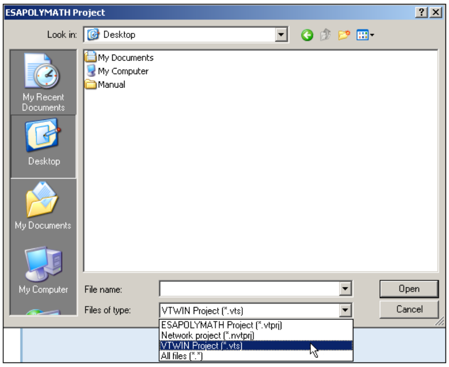
Create new projects for VTxxxW products

The creation procedure of a new project for VTxxxW in POLYMATH is identical to that already described for the other terminals (see chapter , " " on page).
The only difference is in the terminal selection phase, where a panel from the "VT" family must be selected.



Open projects created in VTWIN

POLYMATH allows to open and edit files created with ESA VTWIN software directly. To open a project, the procedure is the same used for any POLYMATH project (see, "   " chapter on page   ); in the file selection window, look through the files with. vts extension.



Look in: Desktop

Files of type: VTWIN Project (*.vts)

Editing differences for different families of panels

By editing a project for a terminal in the VTxxxW family, it is possible to make use of all POLYMATH utilities already described in this manual: copy/paste, library, zoom, graphic functionalities, etc...
The structure of the software (anchorable windows, tools bar and menu) and the functioning mode are those already illustrated during this manual (see chapter, "   " on page   ).
The main difference between editing of CE panels and those of the Windows family is in the contents of the "Esplora Progetto" (Project Explore). Only the functions supported by the operator panel selected in the project creation phase will be present. Moreover, the windows and the options available that will be shown, vary in relation to the terminal model contained in the project. The compilation and download windows are structured following the structure of the relative windows in VTWIN.

The next paragraph analyses the components of the Project Explore in the editing phase of a VTxxxW project.

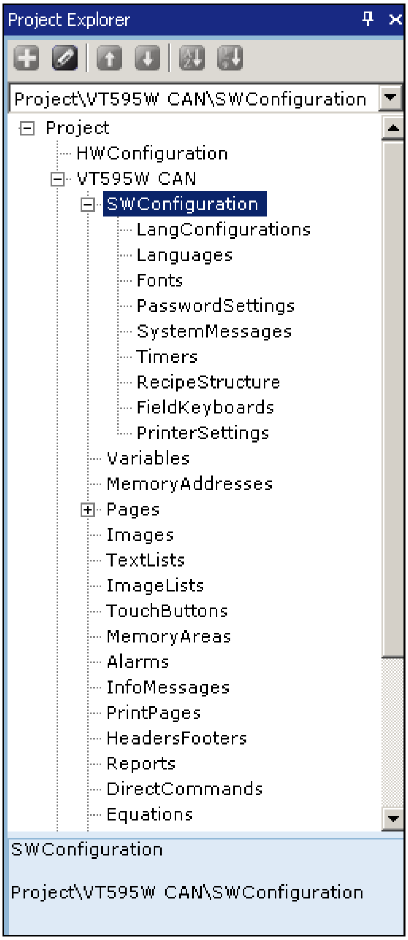
Note: *when editing a project for VTxxxW terminals with POLYMATH, a more modern and simpler interface is offered, which allows to make use of useful tools in the editing phase. However, new functionalities at Runtime level are not introduced.*

Esplora Progetto

Project Explore

The "Esplora Progetto" (Project Explore) contains all of the data relative to the project being edited. Its functioning has already been specified in the relative section of this manual (see chapter , " " on page).

In general, the editing windows of each element of the "Esplora Progetto" (Project Explore) will have the same options contained in the VTWIN application windows.



Project Explorer

Project\VT595W CAN\SWConfiguration

```
⊟ Project
    HWConfiguration
  ⊟ VT595W CAN
    ⊟ SWConfiguration
        LangConfigurations
        Languages
        Fonts
        PasswordSettings
        SystemMessages
        Timers
        RecipeStructure
        FieldKeyboards
        PrinterSettings
      Variables
      MemoryAddresses
    ⊞ Pages
      Images
      TextLists
      ImageLists
      TouchButtons
      MemoryAreas
      Alarms
      InfoMessages
      PrintPages
      HeadersFooters
      Reports
      DirectCommands
      Equations
```

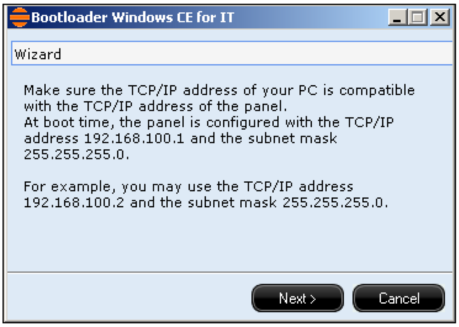
SWConfiguration

Project\VT595W CAN\SWConfiguration

F. Appendix F - Update Operating System

In this chapter is reported operations sequence to follow with Polymath to update the Operating System.

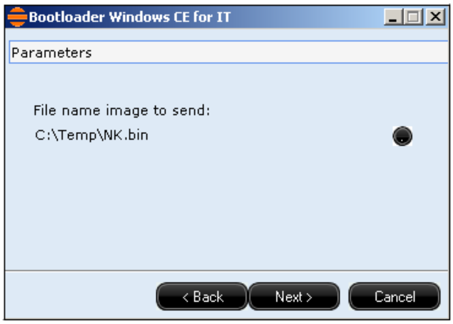
Before start the procedure be sure the terminal is turned off and connect with ethernet cable to the PC.
Be sure subnet mask configurated on PC as follow 255.255.255.0 and IP address is within 192.168.100.2 and 192.168.100.255



Make sure the TCP/IP address of your PC is compatible with the TCP/IP address of the panel.
At boot time, the panel is configured with the TCP/IP address 192.168.100.1 and the subnet mask 255.255.255.0.

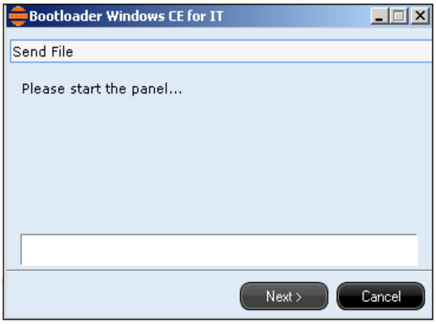
For example, you may use the TCP/IP address 192.168.100.2 and the subnet mask 255.255.255.0.

Use the ● icon to change path where is placed the new OS image to download to the terminal.



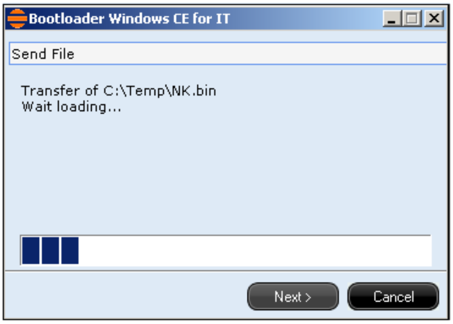
File name image to send:
C:\Temp\NK.bin

Turn on the panel



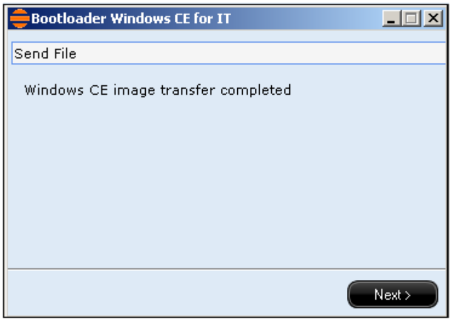
Please start the panel...

The image downloading is in progress.



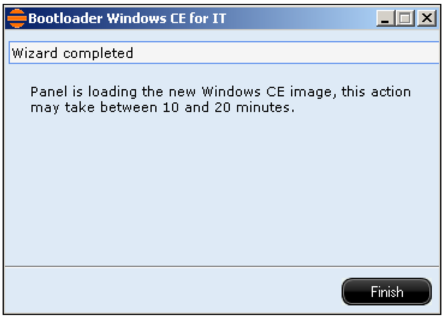
Transfer of C:\Temp\NK.bin
Wait loading...

Waiting the end of the image download.



Windows CE image transfer completed

Waiting for the panel reboot.



Wizard completed

Panel is loading the new Windows CE image, this action may take between 10 and 20 minutes.