



# SUPER-FLASH

## GETTING STARTED

## Registered trademarks copyright

Automa, SUPER-FLASH, SFW, RUNFILE, WRUNFILE, MICROC, SUPER-FLASH OPC CLIENT are registered trademarks of Automa srl

SUPER-FLASH includes portions of the FreeType Project; FreeType Project are Copyright © <2006> by David Turner, Robert Wilhelm e Werber Lemberg

SUPER-FLASH includes portions of Scintilla e Scite; Scintilla e Scite are Copyright © 1998-2003 by Neil Hodgson [neilh@scintilla.org](mailto:neilh@scintilla.org)

Windows is a registered trademark of Microsoft Corporation USA

DOS is a registered trademark of IBM Corporation USA

Other registered trademarks not declared belong to their respective companies.

# INDEX

1 What is SUPER-FLASH? .....	1
1.1 Architecture .....	1
1.2 Development process .....	2
2 Preliminary operations .....	3
2.1 System requirements .....	3
2.2 Setup .....	4
2.3 Launch .....	4
2.4 Exit .....	5
2.5 Uninstall .....	5
3 First approach .....	6
3.1 The project .....	6
3.2 Interface presentation .....	8
3.2.1 Title bar .....	9
3.2.2 Menu bar .....	9
3.2.3 Toolbar .....	10
3.2.4 Status bar .....	10
3.2.5 Work area .....	11
3.2.6 Archives windows .....	11
3.2.7 Graphic editor window .....	12
4 Application wizard .....	13
4.1 Creating a project .....	14
4.2 Developing an application – Phase 1 .....	14
4.2.1 Creating the first variable .....	15
4.2.2 Creating the first graphic page .....	17
4.2.3 Editing the first graphic page .....	18
4.2.4 Creating the first program .....	20
4.2.5 Configuring the communication channel .....	22
4.2.6 Modifying the configuration .....	23
4.3 Test of Phase 1 .....	24
4.4 Developing an application – Phase 2 .....	27
4.4.1 Creating a second variable .....	27
4.4.2 Creating a third variable .....	27
4.4.3 Modifying the first page .....	29
4.4.3.1 Dynamic box .....	29
4.4.3.2 Inserting a button .....	31
4.4.4 Creating a second page .....	33
4.4.5 Editing the second page .....	33
4.4.6 Creating a second program .....	35
4.4.7 Modifying the first program .....	36
4.5 Test of Phase 2 .....	37
4.6 Creating an installation disk for the target machine .....	39
4.7 Installing on the target machine .....	40
4.8 Executing the application on the target machine .....	41
4.9 Connecting a MODBUS peripheral .....	41

4.9.1 Modifying variables .....	41
4.9.2 Modifying the communication channel.....	42
4.10 Solving communication problems .....	44
5 Software potential functionalities .....	46
6 Contacting Automa.....	47

# 1 What is SUPER-FLASH?

SUPER-FLASH is an environment for developing human-machine interfaces (HMI) and SCADA supervision applications. SUPER-FLASH is equipped with a rich set of communication drivers to communicate with field peripherals (PLC, regulators, CNC, etc.) and, moreover, integrate applications with enterprise information systems.

## 1.1 Architecture

SUPER-FLASH is a controlled working environment: development tasks consist mainly in the configuration of the basic elements managed by the system.

The applications developed with SUPER-FLASH rely their operations upon a Runtime Engine which controls adequately the execution safety of the applications developed by end-users.

The reliability level of the final application, obtainable with SUPER-FLASH in the standard way, is therefore intrinsically elevated compared to those developed with traditional programming languages (like Visual C++, Visual Basic, etc.). The causes of this bigger reliability may be understood because of the typical work area used in applications development which may be summarized in:

1. Development phase
2. Compilation phase where metacode, hardware-independent macro language, is produced
3. Execution phase where application functioning reliability is insured by the Runtime Engine through controlled real-time execution of each metacode instruction.

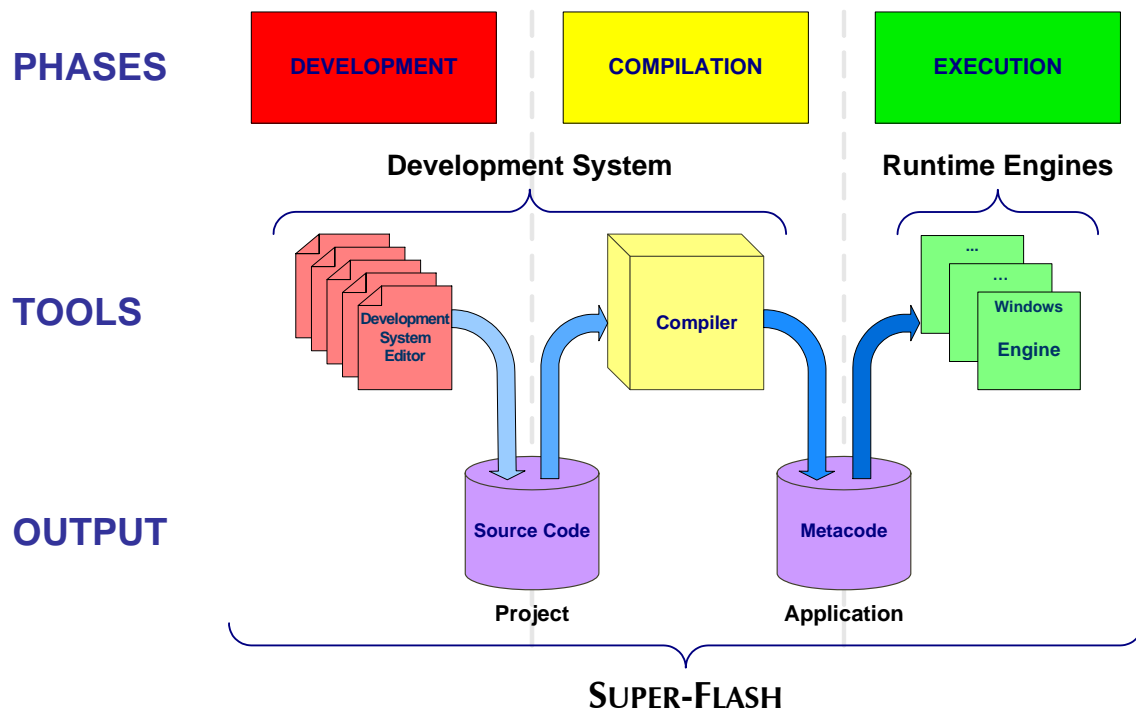
It is important to bring and make clear some concepts that may be taken for granted through the user guide:

- Development system
- Project
- Application
- Runtime Engine

Development System (SFW):	Tool allowing end-users to define in a project all the functionalities to provide an application with.
Project:	Contains information useful for the development system to generate an application.

Application:	Contains coded information useful for Runtime Engines to obtain the functionalities expected by end-users. In this user guide you will find some similar expressions: application, supervision application, application package.
Runtime Engine:	Software skilled to execute the instructions contained in an application on its own operating platform. The Engine now available allows an application to run on Windows, and on Linux in a near future. The Engine that allows an application to run on Windows is called WRUNFILE and the executable program "WRUNFILE.EXE".

## 1.2 Development process



## 2 Preliminary operations

This chapter explains some details related with the installation and launch of SFW development system.

### Note

To describe the operating procedures we have used Windows XP Professional operating system. The name of some windows or commands brought on end may be different in other Windows environments.

### 2.1 System requirements

The SFW development system requires the following:

Component	Minimum requirements
Operating system	Windows XP Professional SP3 with Internet Explorer 7
Processor	Pentium 4 \ Athlon
System memory	512 Mbyte
Available disk space	90 Mbyte
Graphic card resolution	1024 x 768 64K colours
Ports	Parallel or USB
Other peripherals	CD-ROM

The system requirements of the target machine where you want to install the final application may be different from those required for the machine used for the application development. For the application, instead, it is not possible to define the requirements a priori as they can change according to two important factors:

- the operating system present on the target machine (Windows XP-PRO, Windows XP-E, Windows 7, etc.)
- the features of the developed application

The selection of an operating system conditionates strongly the machine requirements. Moreover, some requirements depend strongly on application features. For example: in a monitoring application the data recording operation is important and, therefore, it is necessary to evaluate the available disk space. An on-board application may require touch-screen requirements instead of using a keyboard. A remote application does not need accessing supports such as a floppy disk or a CD writer.

## 2.2 Setup

The SFW development system is distributed in CD-ROM. It is also possible to download it from Automa's website <http://www.automa.it> in the "Free Download" section for RUNTIME versions or in the "Download" section for ROYALTY-FREE versions.

The setup procedure installs the development system, the Runtime Engines, some demo applications and a tool set. To setup do the following:


1. click twice on the "SETUP.EXE" program. The program can be found in the CD-ROM, as well as, in the files downloaded from Automa's website
2. select the language to install
3. the setup is almost automatic if end-users select the complete installation option. If selected the customized installation option, end-users have to specify:
  - the directory where to install SFW: the directory "\Programs\Automa\SFW [version]" is proposed as default
  - the directory where to install the SFW projects: the directory "\My SFW Projects" is proposed as default
  - which components to install

During the installation you will be asked whether to install or not the protection key drivers: the installation is automatic.

## 2.3 Launch

There are two possibilities to launch SFW development system:




- Click on the "Start" button in the Windows application bar to visualize the Start menu and select "Programs" ► "Automa" ► "SFW [version]" ► "SFW [version] RUNTIME" or "SFW [version] ROYALTY-FREE"
- Click twice on the "SFW [version] RUNTIME" or "SFW [version] ROYALTY-FREE"

short-cut  icon present on Windows desktop



## 2.4 Exit

To exit the SFW development system you can use one of the following possibilities:

- select the “Exit” option from the “File” menu (“File” ► “Exit”)
- click on the SFW  button (“Close”) (last right button on the title bar)
- click on the SFW control icon  (title bar left corner) to display the Control menu and select “Close”
- click twice on the SFW  control icon (title bar left corner)
- press down the **ALT + F4** combination

## 2.5 Uninstall

To uninstall the SFW development system you have to proceed in the following way:

1. click on the “Start” option in the Windows application bar to display the Start menu
2. select “Control panel”
3. click twice on the “Add or Remove Programs” icon
4. in the section “Change or Remove program” run through the program list and click on “SFW [version]”
5. click on the “Change/Remove” button to remove the program
6. select “Remove”

The procedure to uninstall the drivers for protection key management is analogous to the one above described. In this case the program to uninstall is “SmartKey Driver”.

## 3 First approach

This chapter describes some important elements regarding SFW development system, such as projects and archives, fundamental to understand how to develop a basic application with SFW.

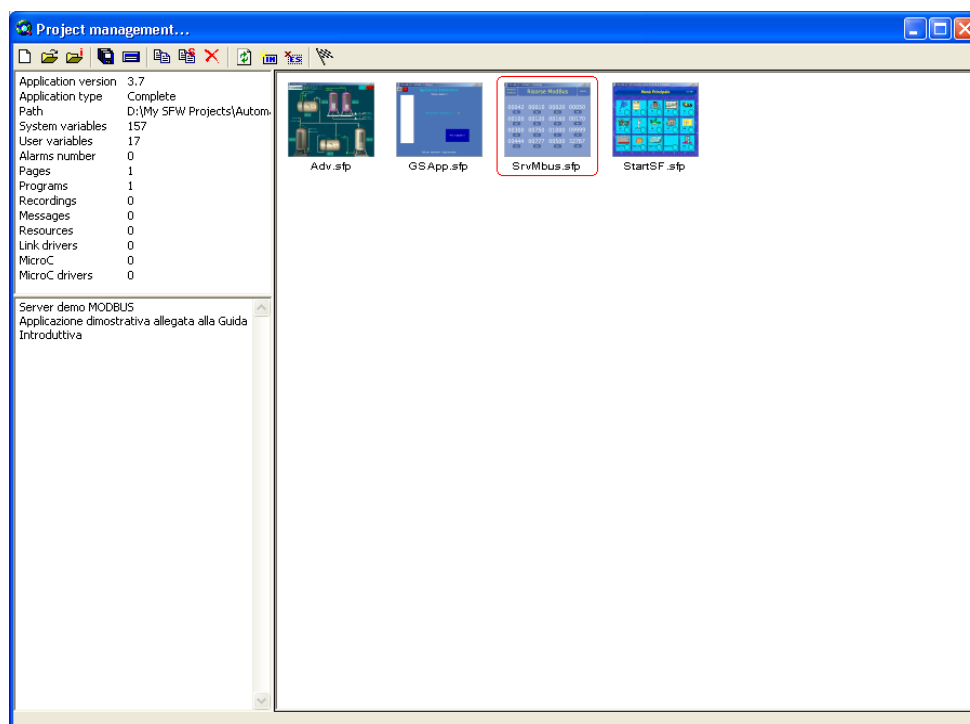
It also illustrates SFW end-user interfaces: describes the main toolbars, the work area and the windows.

### 3.1 The project

Before talking about end-user interfaces it is important to remember the meaning of the following terms: development system, project, application and Runtime Engine as described in the [Architecture](#) paragraph (page 1).

A project may be created or opened. When a project is opened it may be saved and closed any time and, if wanted, without saving the last modifications.

SFW makes available a tool which allows to manage the development of the various projects present in the same machine in a practical way. In the “**Project management**” window there are available some useful commands as for project backup and restore, which allow to prepare the application installation disk for the target machine.



Once a project is opened the development phase is based mainly on the creation and modification of the elements of the following archives: variables, pages, programs, recordings, messages, resources, configuration, link drivers, MICROC and MICROC drivers. Usually all the mentioned elements have a code and a description.

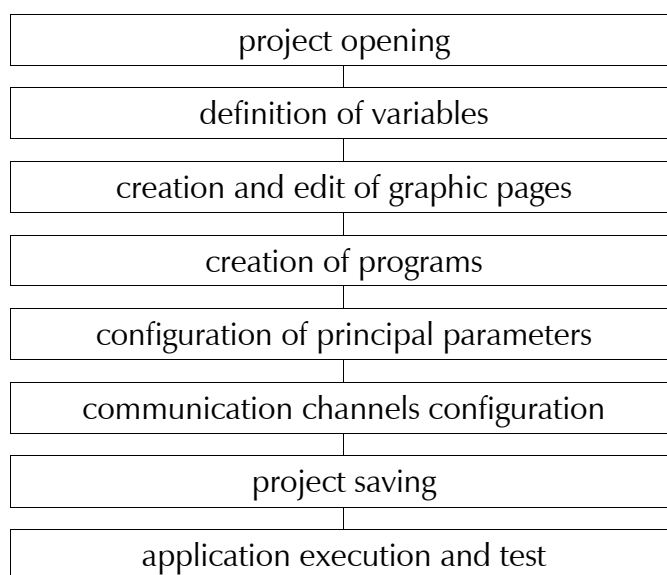
Only detailed information regarding the basic archives useful to develop a sample application are described below:

- variables: archive containing variables connecting the application with peripherals. The archive contains elements predefined as “system variables” created to manage some system functionalities (there are system variables for recipe management, alarm management, etc.)
- pages: archive containing graphic and printing pages, as well as, disk data saving pages. This archive contains single pages along with their main functionalities; for the edit phase it is necessary to use the appropriate editor according to the page type
- programs: archive containing programs to structure the application with its functional logic and the actions to be undertaken when an event is verified (key pressed down, change of variable values, etc.)
- configuration: archive containing the principal parameters for a specified application. The archive is composed of elements already predefined and configured with default values. End-users may modificate single parameters (peripheral parameters, alarm parameters, etc.)

Development tasks may be considered as finished only after the compilation and test phases. In the compilation process the information contained in the archives is codified in metacode to be then executed by Runtime Engines. Therefore, the archives present in a project are in two formats: source code and metacode.

The project compilation is transparent to end-users as it is executed along with the project saving operation.

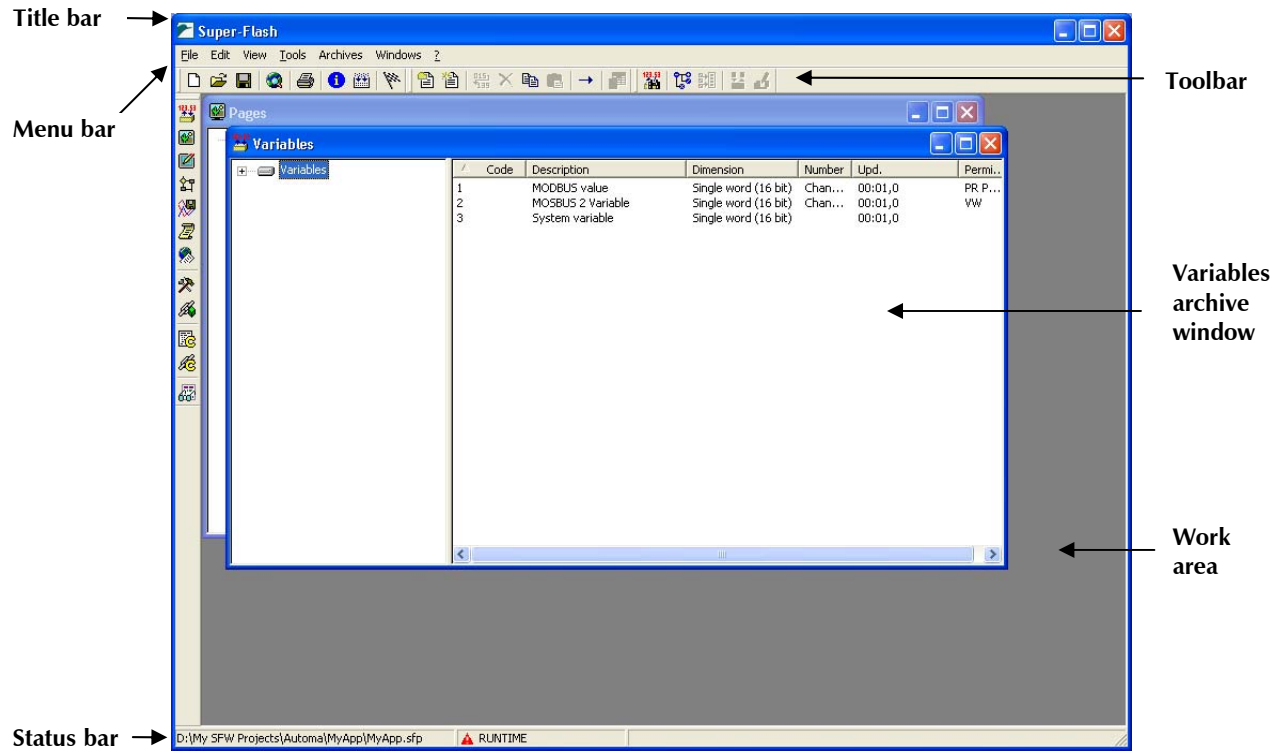
Summarizing, the developer activities will be as follows:



## 3.2 Interface presentation

After launching SFW, a main window is being displayed. It is delimited upwards by the title bar and downwards by the status bar.

The main window is composed of a menu bar, a toolbar and a work area where windows regarding archives management are displayed and, when necessary, dialogue and message windows according to specified situations.



### 3.2.1 Title bar

The title bar appears in the upper part of the main window, as well as, in all the windows present in the application.







By clicking on the control icon end-users may activate the control menu. It contains commands which function on the active window.



By clicking twice on the control icon end-users may close quickly an active window.

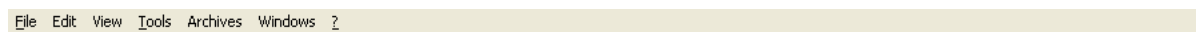
The title bar describes the window content.

The Windows buttons allow end-users to minimize , maximize , restore  and close  the current window.

### 3.2.2 Menu bar

The menu bar can be found below the title bar. The menu bar brings back the heading of each menu and by clicking on a determined one, an associated menu appears. End-users may select a determined command according to what they want to do.

The commands may have an associated icon: the icon is useful for remembering the command correspondence in the toolbar.



### 3.2.3 Toolbar

The toolbar is usually below the menu bar. The toolbar contains the buttons associated to the frequently used commands.

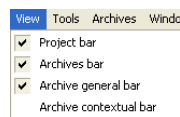
The SFW toolbar is composed of:

- a project bar
- an archives bar
- an archives general bar
- an archives contextual bar



The associated command name can be displayed by stopping over a determined button with the mouse. Click a button to execute the respective command and if a specified button is not active, it means that the command is not available.

Single bars may be displayed or hidden by selecting “View” and checking on/off the respective checkbox in the menu.



A bar may be “docked” or “floating”



“docked”



“floating”

An “docked” bar may be positioned only on the margins of the work area: below the menu bar, above the progress bar, at the left or right side of the work area. A “floating” bar may be positioned in any screen point. To move a bar from a screen point to another: click on the bar with the mouse left button and drag and drop it in any screen point.

It is possible to dock or undock a bar by clicking twice on its background area prive of buttons or on the title bar. When a bar is released it can be resized.

### 3.2.4 Status bar

The status bar can be found in the lower part of the main window. It displays some general information (active project path, protection key absence), as well as, information depending on the operating context.



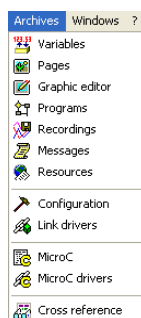
### 3.2.5 Work area

The work area is where an application is developed by operating in the archives windows.

### 3.2.6 Archives windows

The archives windows are opened in the SFW work area. There are two ways of opening a window:

- select the archives group from the “Archives” menu



- click on the respective archives button in the archives bar



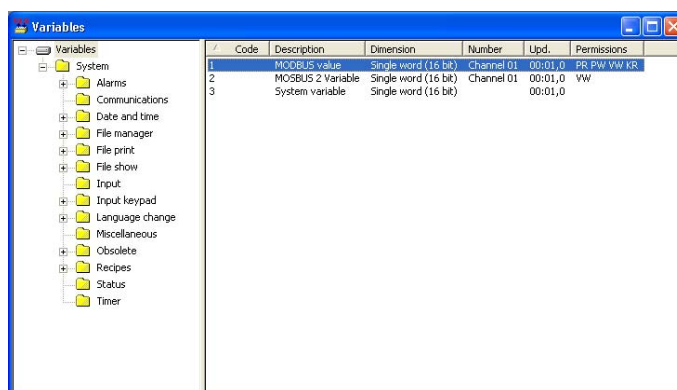
#### Note

“Graphic editor” and “Cross reference” are not archives, they have been placed in this section because both open windows in the work area. In particular:

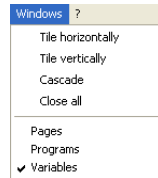
- “Graphic editor” is a development tool which allows the edit of graphic pages
- “Cross reference” is a tool which produces a detailed map of the basic resources used in the application (variables, pages, programs and MICROC programs)

All archives windows are typically composed of two areas:

- on the left there is a group tree representing the elements hierarchical organization.
- on the right there is an element list (variables, pages, etc.)



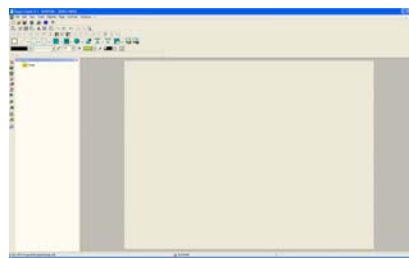
Archives windows, as all Windows operating systems windows, may be moved from one point screen to another, restored, maximized, minimized and closed. A specific section (“**Windows**”) is dedicated to windows management, from where it is possible to organize windows according to established screens, close all of them, see which windows are open and which one is active, as well as, change the active window.



### 3.2.7 Graphic editor window

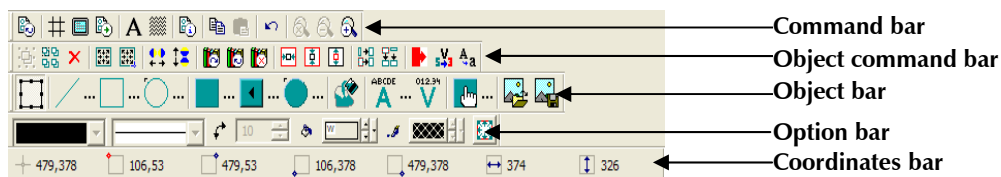
The SFW Graphic editor uses a different window in the work area and is composed of two parts:

- an object tree is displayed on the left
- the page edit area is present on the right



The standard toolbar is enriched with the graphic editor bars:

- command bar
- object command bar
- object bar
- option bar
- coordinates bar





## 4 Application wizard

This chapter illustrates step by step how to develop a small SUPER-FLASH application by using the main basic functionalities:

- variables
- pages
- programs
- some configuration elements

We will call the application 'MyApp' and it will manage:

1. the display and modification of variables associated to a MODBUS peripheral
2. the display of dynamic graphic objects according to variables values
3. graphic objects with input functionalities

The application development is composed of two phases: the first one achieves the point 1 and the second the points 2 and 3 respectively.

At the end of the procedure, **MyApp** should be similar, from a functional point of view, to the **GSApp** demo application (Getting Started Application) supplied with SUPER-FLASH.

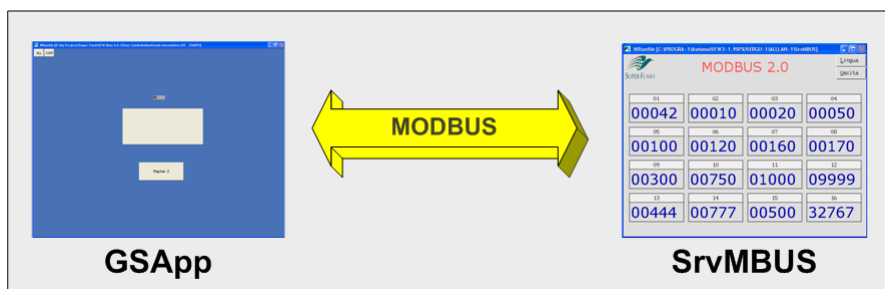
We suggest you to try to execute the wizard procedure to then check it out with the **GSApp** demo application to compare the results obtained.

SUPER-FLASH supplies a set of demo applications. The application references for this workshop are:

- **SrvMBUS** : MODBUS server application
- **GSApp** : demo client application

These applications will be installed by SFW when selecting the complete installation or by selecting the "Getting Started" component in case you have chosen the customized installation.

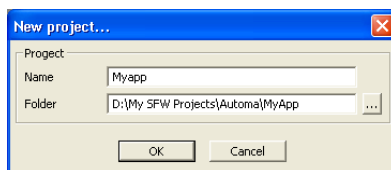
They are present in Super-Flash installation directory (if the default directories have been maintained they should be present in "C:\Programs\Automa\SFW x.y") respectively in "User Guides\AllLanguage\SrvMBUS" e "User Guides\English\GSApp" directories.



SUPER-FLASH includes a considerable number of communication drivers. The MODBUS (MODBUS RTU) protocol, one of the well-known protocols, was selected for the application development. A second application called **SrvMBUS** which, as the name suggests, has the role of a small MODBUS peripheral (SUPER-FLASH may be a MODBUS server!), is supplied to facilitate the **MyApp** test phase. In case end-users have a peripheral with MODBUS protocol, they can do a real-time connection test by modifying appropriately the configuration parameters of the **MyApp** application. For more information read the [Connecting a MODBUS peripheral](#) paragraph (page 41).

## 4.1 Creating a project

1. Launch SFW
2. Select from the “File” menu ► “New project”: the window for the creation of a new project will be displayed
3. Set the following parameters:
  - “Name” → “MyApp”
  - Change, if necessary, the “Folder” where to save the project created by browsing the directory list which may be activated by clicking the ... button



4. “OK” to continue

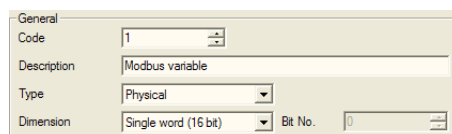
## 4.2 Developing an application – Phase 1

In the first phase of **MyApp** application development end-users will be driven through the following operations:

1. [creating the first variable with a MODBUS peripheral](#) (page 15)
2. [creating the first graphic page](#) (page 17)
3. [editing the first graphic page](#) (where the first variable will be displayed) (page 18)
4. [creating the first program](#) (which will control the first page display) (page 20)
5. [configuring the communication channel](#) (page 22)
6. [modifying some configuration elements](#) (page 23)
7. [testing the application](#) (page 24)

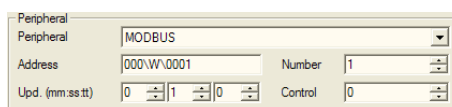
## 4.2.1 Creating the first variable

1. Select from the “Archives” menu ► “Variables”
2. Select from the “Edit” menu ► “Insert new object”: the variable properties window will be displayed
3. In the “General” group set the following parameters:
  - “Code” → “1”
  - “Description” → “MODBUS value” (or another description)
  - “Type” → “Physical”
  - “Dimension” → “Single word (16-bit)”



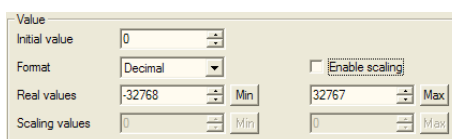
The screenshot shows the 'General' tab of a variable properties window. The 'Code' field contains '1'. The 'Description' field contains 'Modbus variable'. The 'Type' dropdown is set to 'Physical'. The 'Dimension' dropdown is set to 'Single word (16 bit)'. The 'Bit No.' field contains '0'.

4. In the “Peripheral” group set the following parameters:
  - “Peripheral” → “MODBUS”
  - “Address” → “000W\0001” (Unit id 0, Word resource at 16-bit, address 1)
  - “Number” → “1” (see Note 1)
  - “Control” → “0”
  - “Upd (mm:ss:tt)” → “0”, “1”, “0” (see Note 2)



The screenshot shows the 'Peripheral' tab of a variable properties window. The 'Peripheral' dropdown is set to 'MODBUS'. The 'Address' field contains '000W\0001'. The 'Number' field contains '1'. The 'Upd. (mm:ss:tt)' field contains '0'. The 'Control' field contains '0'.

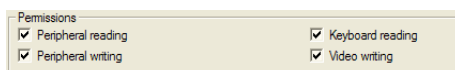
5. In the “Value” group set the following parameters:
  - “Initial value” → “0”
  - “Format” → “Decimal”
  - “Enable scaling” → “☐” (disabled)
  - “Real values - Min” → “-32768”
  - “Real values - Max” → “32767”



The screenshot shows the 'Value' tab of a variable properties window. The 'Initial value' field contains '0'. The 'Format' dropdown is set to 'Decimal'. The 'Enable scaling' checkbox is disabled. The 'Real values - Min' field contains '-32768' and the 'Real values - Max' field contains '32767'. The 'Scaling values - Min' field contains '0' and the 'Scaling values - Max' field contains '0'.

6. In the “Permissions” group set the following parameters:

- “Peripheral reading” → “☒” (enabled)
- “Video writing” → “☒” (enabled)
- “Keyboard reading” → “☒” (enabled)
- “Peripheral writing” → “☒” (enabled)



7. “OK” to continue

### Notes

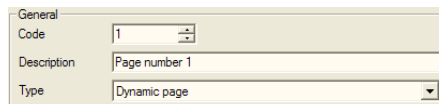
1) The field “Number” indicates the ‘logic channel’ associated to a certain peripheral. In a specific section regarding application configuration, there will be defined the associations between ‘logic channels’ and their physical connections. It will allow end-users, if modifying the communication port (for example: if changing from COM1 to COM2), to take part only in the logic channel configuration, instead of modifying each variable linked to the peripheral.

The configuration of channel 1, useful for the **MyApp** application, is described in the [Configuring the communication channel](#) paragraph (page 22).

2) The “Upd (mm:ss:tt)” field (updating time) determinates the minimal time interval between a peripheral reading and the successive one. The value is expressed in minutes:seconds:tenths. In the example was indicated a time of 1 second.

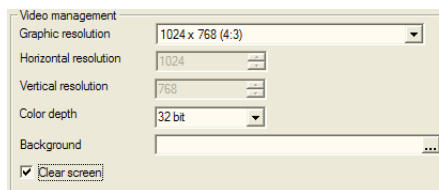
## 4.2.2 Creating the first graphic page

1. Select from the “Archives” menu ► “Pages”
2. Select from the “Edit” menu ► “Insert new object”: the page properties window will be displayed
3. In the “General” group set the following parameters:
  - “Code” → “1”
  - “Description” → “Page number 1” (or another description)
  - “Type” → “Dynamic page”



General	
Code	1
Description	Page number 1
Type	Dynamic page

4. In the “Video management” group set the following parameters:
  - “Graphic resolution” → “1024 x 768 (4:3)”
  - “Color depth” → “32 bit”
  - “Background” → “” (empty)
  - “Clear screen” → ☒ (enabled)

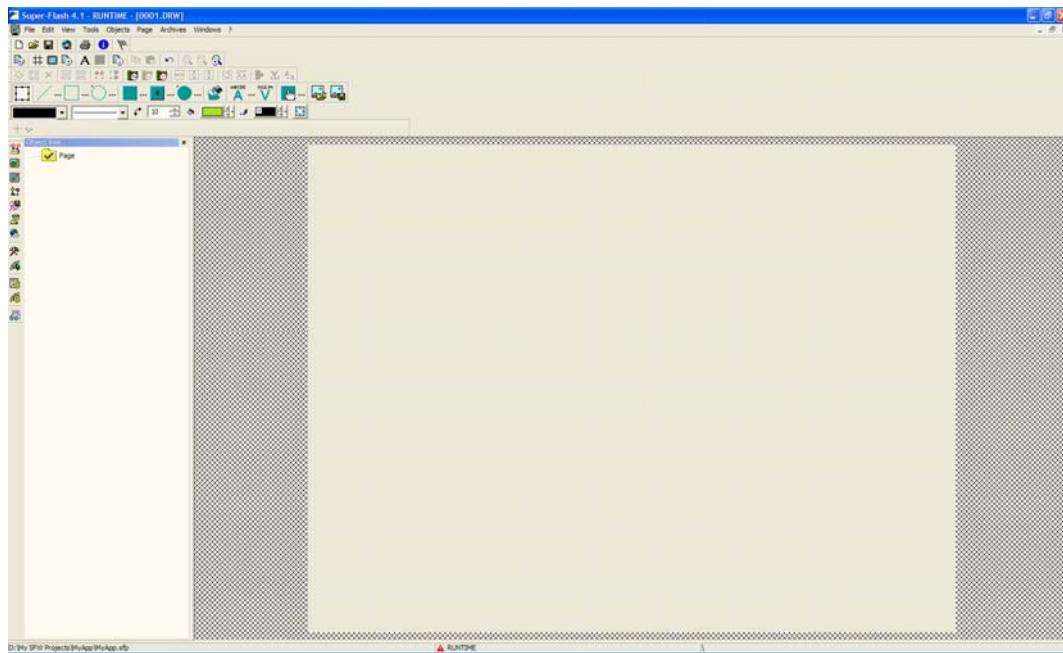


Video management	
Graphic resolution	1024 x 768 (4:3)
Horizontal resolution	1024
Vertical resolution	768
Color depth	32 bit
Background	
<input checked="" type="checkbox"/> Clear screen	

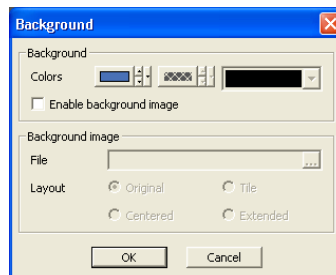
5. “OK” to continue


## 4.2.3 Editing the first graphic page

1. Activate the page editor by clicking twice on page 1





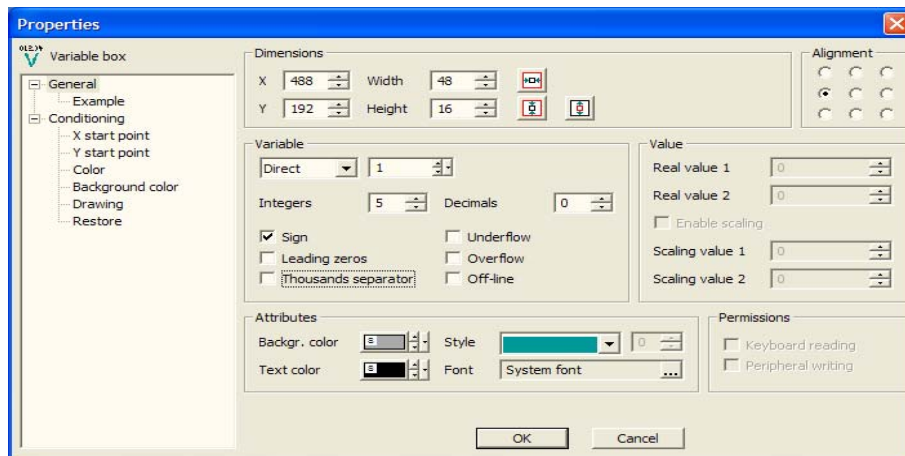
2. Select from the “Page” menu ► “Background”: the page background window will be displayed
3. Select the page background color by selecting one of the predefined colors through the use of two small buttons with arrows, or by defining a customized color in the color selection interface which may be activated through the single button with an arrow on it.



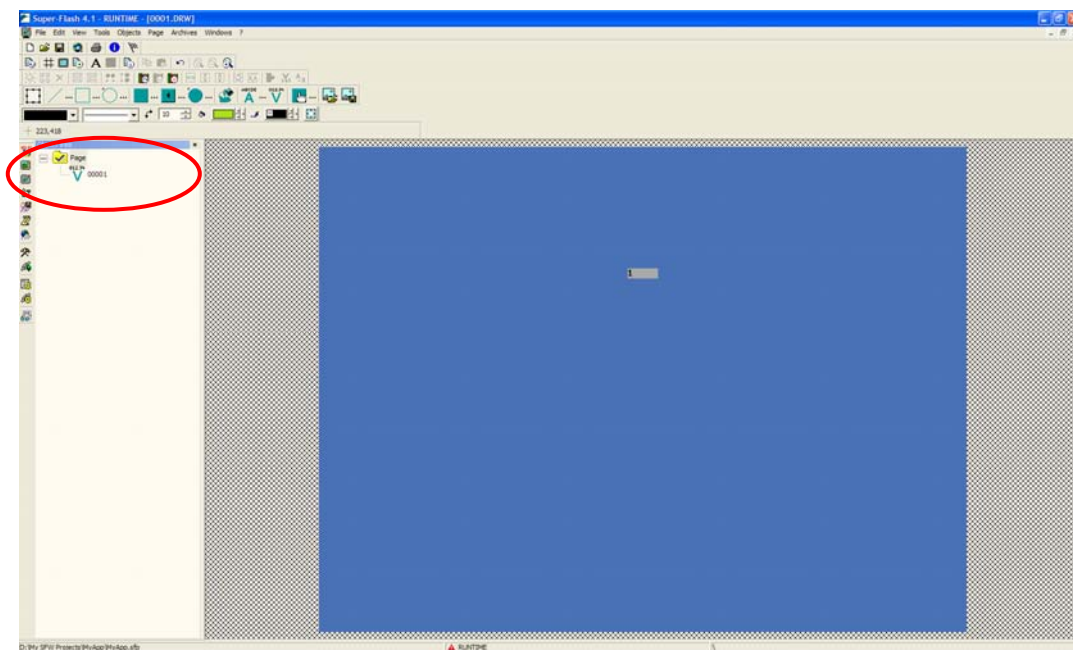
4. “OK” to continue
5. Design a variable object by selecting the  (“Box Variable”) option in the object bar and by clicking on the page edit area: the variable object properties window will be displayed.

6. In the “General” group set the following parameters:

- “Variable” → “Direct” “1”
- “Integers” → “5”
- “Sign” → “☒” (enabled)
- set colors as you prefer
- Act on buttons  e  to automatically calculate object dimensions



7. “OK” to continue



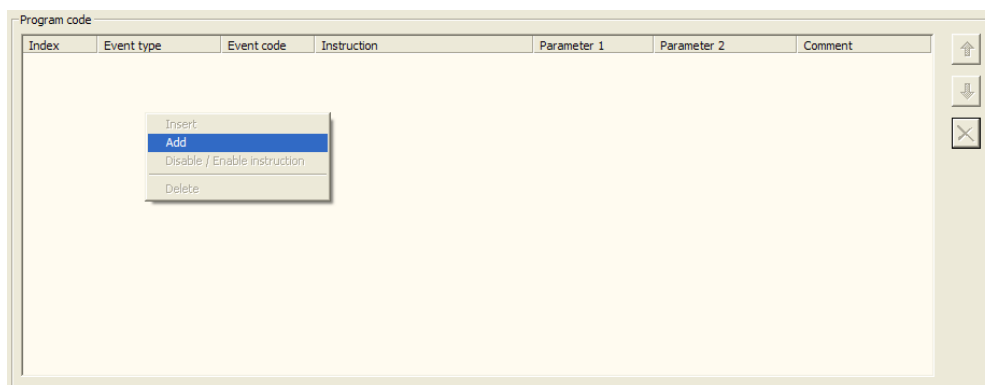
Note that a variable object appears in the object tree.

The object tree is very important because it allows end-users to modificate the objects order through simple drag&drop operations. The drag&drop object order determinates the object to be displayed in the foreground, as well as, those to be displayed in the background.

8. Select from the “Page” menu ► “Save”

## 4.2.4 Creating the first program

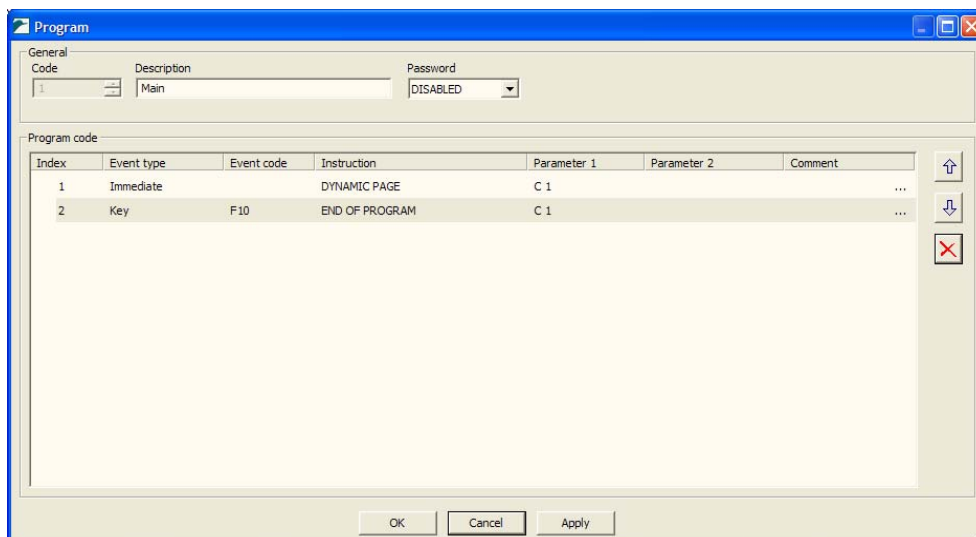
1. Select from the “Archives” menu ► “Programs”
2. Select from the “Edit” menu ► “Insert new object”: the program properties window will be displayed
3. Set the following program general data:
  - “Code” → “1” (see Note 1)
  - “Description” → “Main” (or another description)
  - “Password” → “DISABLED”
4. Add an instruction by selecting the “Add” option from the popup menu available in “Program code” area (through right-click of mouse)



5. By clicking directly on each instruction fields set the following parameters:
  - “Event type” → “IMMEDIATE”
  - “Instruction” → “Display” → “DYNAMIC PAGE”
  - “Parameter 1” → “CONSTANT”, “1”



6. Add the END OF PROGRAM instruction by inserting the following parameters into the respective fields:
- "Event type" → "Key"
  - "Event code" → "F10" (click on the input area and press down on **F10** function key)
  - "Instruction" → "Prog. Structure" → "END OF PROGRAM"
  - "Parameter 1" → "CONSTANT", "1" (see Note 2)



7. "OK" to continue

In case end-users want to modify the instruction, they should select the instruction by clicking on it, modify the field parameters and confirm the new modifications through the "Modify" button.

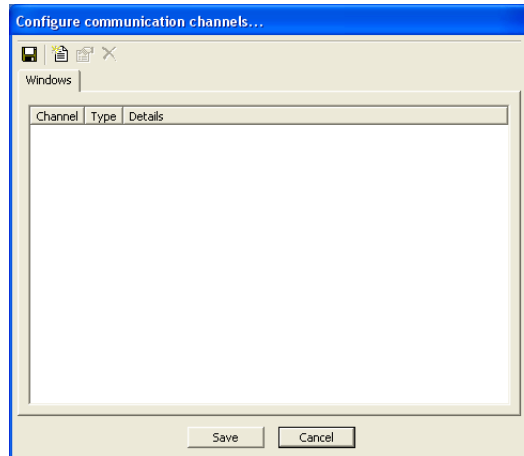
### Notes


1) A program with code 1 is the SUPER-FLASH application entry-point and it must be always defined. Avoiding this may cause errors when trying to launch the application. The code 0, instead, is reserved for the background program (not used in the MyApp application).

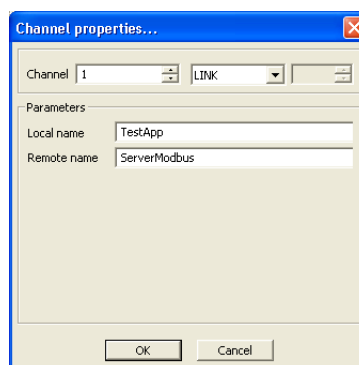
2) The "END OF PROGRAM" instruction may be interpreted as the end of the current program and the return to the calling program. In the parameter there is specified the call level number (in this case it coincides with the exit of the MyApp application).

## 4.2.5 Configuring the communication channel

1. Select from the “Tools” menu ► “Configure communication channels”: the communication channels configuration window will be displayed
2. Go to the “Windows” group



3. Create a new channel through the  icon (“New”) and set the following parameters:
  - “Channel” → “1”, “LINK” (see Note 1)
  - “Local name” → “TestApp”
  - “Remote name” → “ServerModbus”



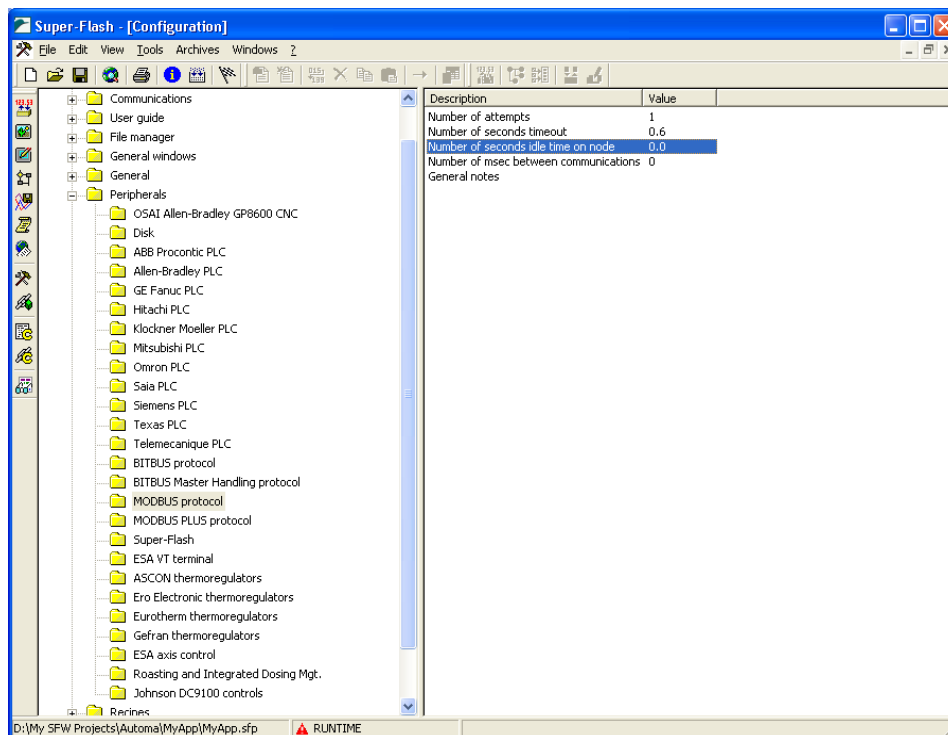
4. “OK” and “Save” to continue

### Note

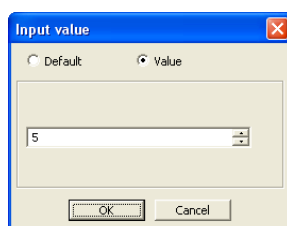
1) The “LINK” channel type may be used exclusively to communicate two SUPER-FLASH applications running on the same machine. It was created in the **MyApp** application just because the first communication tests will be executed by using the **SrvMBUS** application. Selecting other communication types (like COM or TCP/NETBIOS) should have involved a longer configuration phase (network configuration analysis, serial communication ports availability and potentially the use of two PCs).

## 4.2.6 Modifying the configuration

1. Select from the “Archives” menu ► “Configuration”
2. Expand the tree present on the left of the interface, follow the “Configuration \ Peripherals” path and select the “MODBUS protocol” element



3. Insert the parameter value “5.0” (seconds.tenths) into the field “Number of seconds idle time on node” (see **Note 1**) displayed on the right by clicking twice on the parameter: an input value window will be displayed as shown below



4. “OK” to save the new parameter

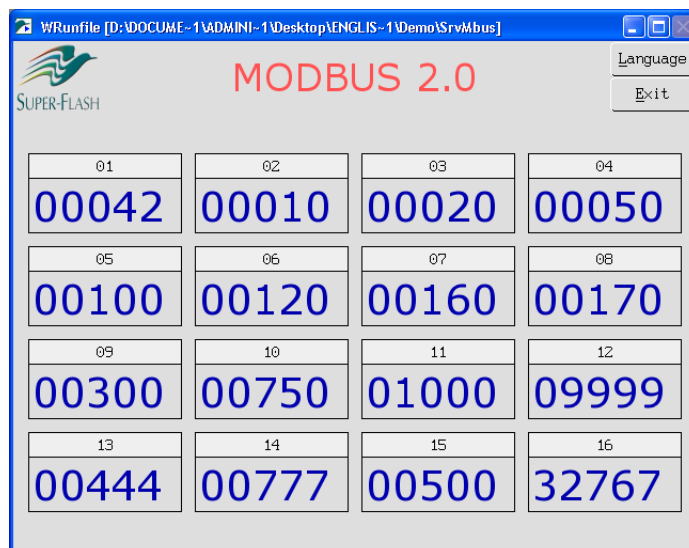
### Note

1) In case of communication failure, SUPER-FLASH attempts to connect continuously to the peripheral slowing down the end-user interface management. By inserting a parameter value in this section end-users are specifying the inactivity time between each connection attempt.

## 4.3 Test of Phase 1

Now we can test the **MyApp** application.

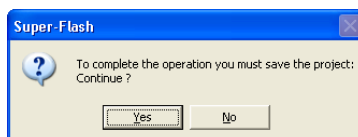
1. Launch the MODBUS server application through “Start” ► “Programs” ► “Automa” ► “SFW [version]” ► “User guides” ► “Getting Started (SrvMBUS)”



The application emulates a MODBUS peripheral with the following attributes:

- peripheral address : 0 (unit identifier)
- 16-bit word managed : addresses from 1 to 16

2. Launch the recently developed **MyApp** application by selecting the “Tools” menu option ► “Run WRunfile”, a message asking end-users to save the project will be displayed

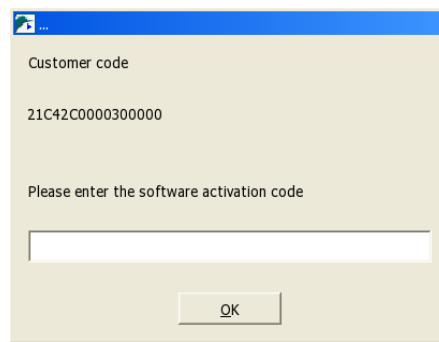


The message appears because the project has not been saved during the development phase. When saving a project, the compilation is executed automatically; the compiled files are those required by the Runtime Engine to run the application.

End-users may save the project anytime by selecting “Save” from the “File” menu.

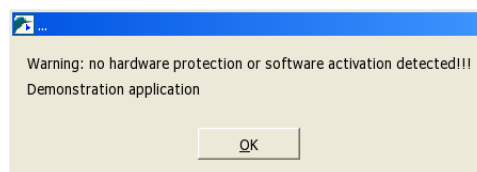
3. Press “Yes” to continue

Because no hardware smart key was found, a second message requesting a software activation code will appear.



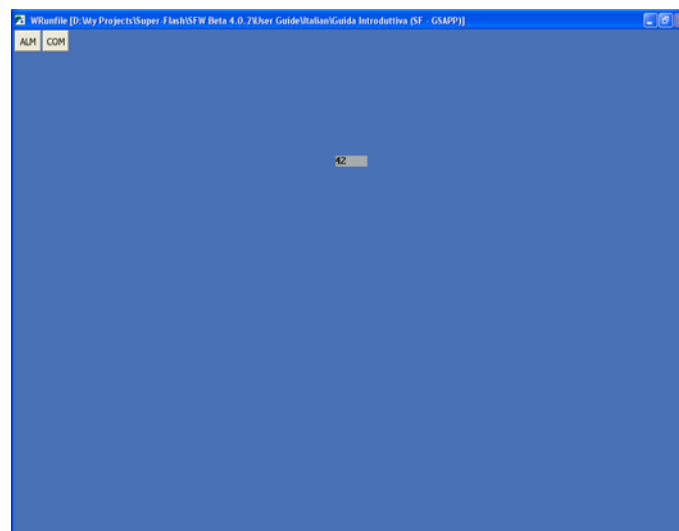
4. Press "OK" to continue

A third message indicating the application launch as demo will appear. Applications launched as demo ends automatically after 15 minutes.



5. Press "OK" to continue

Once launched the **MyApp** application, a page with a variable value of 42 (default value contained in the address 1 of the **SrvMBUS** application) is displayed: it means that the procedure has been correctly executed.

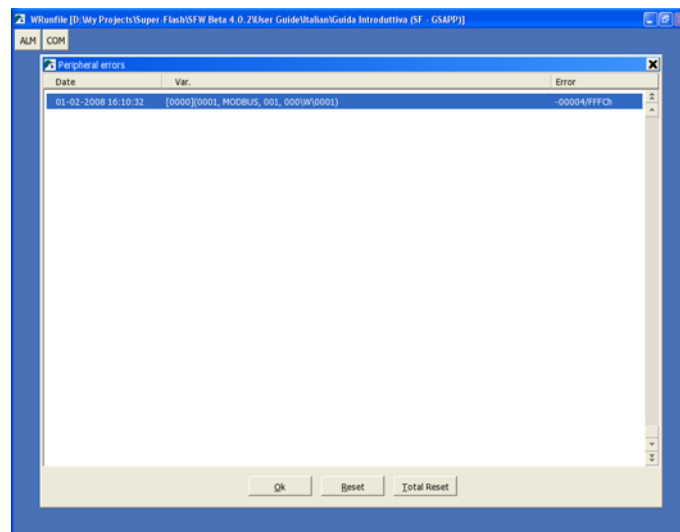


It is possible to modify the value variable with just a click on the variable area. To cancel the modifications done press down the **Esc** key, to confirm them press down the **Return** key: the value is sent to the peripheral and the **SrvMBUS** application will show the new value (address 1).

Vice versa, if modifying the variable value on the **SrvMBUS** application, the page will be updated.

Besides the variable designed by end-users, there are also two distinguished buttons: “ALM” and “COM”. Both of them indicate their respective status allowing end-users to access to alarm management and communication error windows respectively. By default the buttons are present in all application pages and end-users may modify their attributes by accessing the “Alarms \ Alarm block” and “Communication \ Communication block” configuration sections respectively.

In case of communication problems, the variable shows a value of 0 and the “COM” button flashes from grey to red. An error log window is then displayed when clicking the “COM” button.



For detailed information regarding the error log description as shown above, read the [Solving communication problems](#) paragraph (page 44).

When having communication problems end-users must terminate the application and verify the following in this order:

- that the server application is active
- the variable parameters
- the communication channel parameters

6. Press down **F10** function key to terminate the **MyApp** application

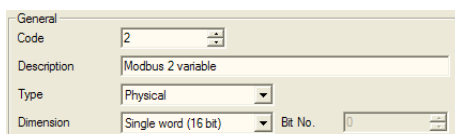
## 4.4 Developing an application – Phase 2

Now we can pass to a second phase for the following operations:

1. [creating a second variable with a MODBUS peripheral](#) (page 27)
2. [creating a third variable with a System peripheral](#) (page 27)
3. [modifying the first page to insert](#) (page 29):
  - 3.1. [a dynamic object](#) (page 29)
  - 3.2. [a button for executing a command](#) (page 31)
4. [creating a second page](#) (page 33)
5. [editing the second page](#) (page 33)
6. [creating a second program](#) (pag. 35)
7. [modifying the first program to hook up to the second](#) (page 36)
8. [testing the phase 2](#) (page 37)

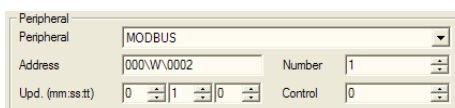
### 4.4.1 Creating a second variable

1. Select from the “Archives” menu ► “Variables”
2. Select the variable 1
3. Select from the “Edit” menu ► “Insert new object”: the variable properties window will be displayed. The second variable inherits by default the properties of the first one (it is created by duplicating the one selected in the list). The new variable is still a MODBUS one, therefore it is enough to change just few parameters.
4. In the “General” group set the following parameters:
  - “Code” → “2”
  - “Description” → “MODBUS 2 Variable” (or another description)



General	
Code	2
Description	Modbus 2 variable
Type	Physical
Dimension	Single word (16 bit)
Bit No.	0

5. In the “Peripheral” group set the following parameters:
  - “Address” → “000\W\0002”



Peripheral	
Peripheral	MODBUS
Address	000\W\0002
Number	1
Upd. (mm:ss.tt)	0 1 0
Control	0

6. “OK” to continue

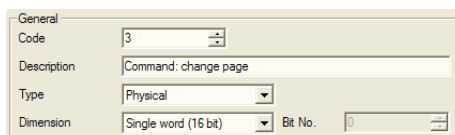
### 4.4.2 Creating a third variable

1. Select from the “Archives” menu ► “Variables”

2. Select from the “Edit” menu ► “Insert new object”: the variable properties window will be displayed

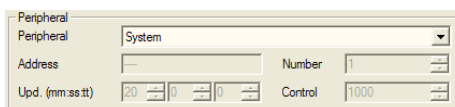
3. In the “General” group set the following parameters:

- “Code” → “3”
- “Description” → “Command: change page” (or another description)
- “Type” → “Physical”
- “Dimension” → “Single word (16 bit)”
- “Format” → “Decimal”



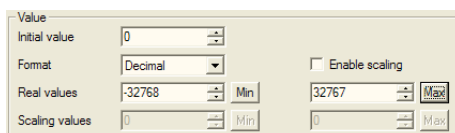
4. In the “Peripheral” group set the following parameters:

- “Peripheral” → “System” (see Note 1)



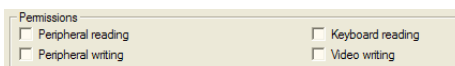
5. In the “Value” group set the following parameters:

- “Initial value” → “0”
- “Enable scaling” → “☐” (disabled)



6. In the “Permissions” group set the following parameters:

- “Peripheral reading” → “☐” (disabled)
- “Video writing” → “☐” (disabled)
- “Keyboard reading” → “☐” (disabled)
- “Peripheral writing” → “☐” (disabled)



7. “OK” to continue

### Note



1) The “System” peripheral is generally used for variables destined to contain application internal data (constant values, temporary variables, events managed by programs, etc.). If the peripheral writing and reading options are enabled, the variables with System peripherals store their values on the disk, avoiding in this way their loss in case of exiting the application.

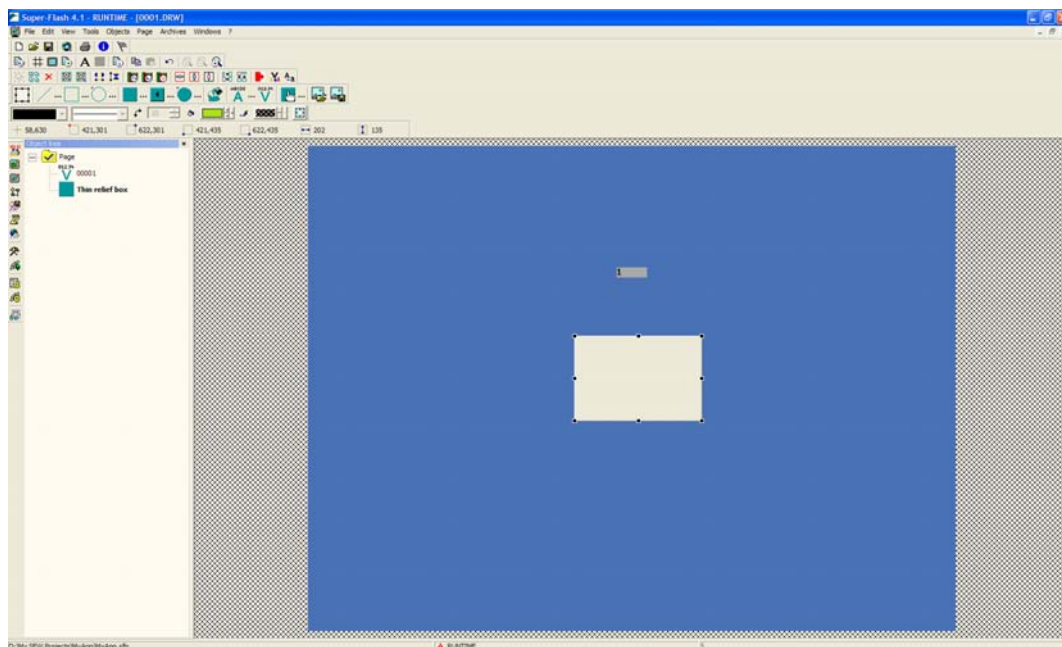


### 4.4.3 Modifying the first page

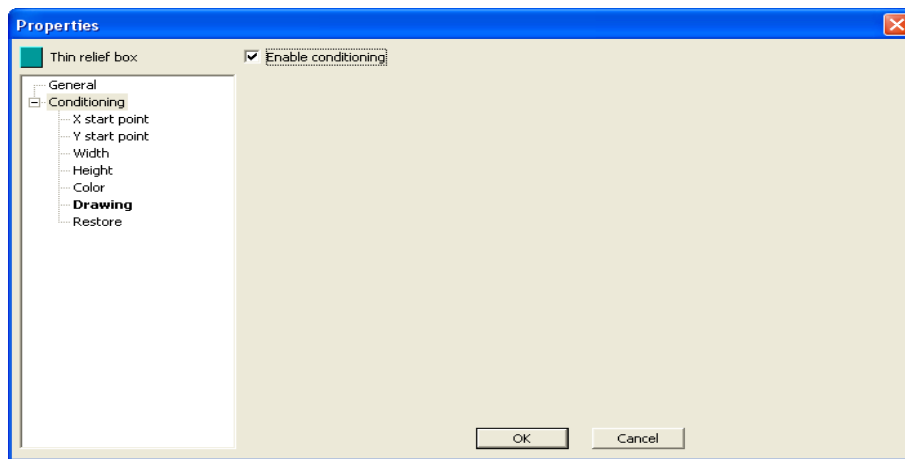
1. Select from the “Archives” menu ► “Pages”
2. Open the page editor by clicking twice on page 1

#### 4.4.3.1 Dynamic box

1. Draw a box object by selecting the  (“Thin relief box”) button in the object bar and clicking on the edit area. Then “stretch” the mouse while holding down the mouse left button to then drop it once satisfied with the expected object dimensions.  
This procedure is valid to design any graphic object except texts, resources and variables where just a click is needed.
2. Choose the  (“Select”) option, click on the object and try to move and change the object dimensions through the appropriate handles.

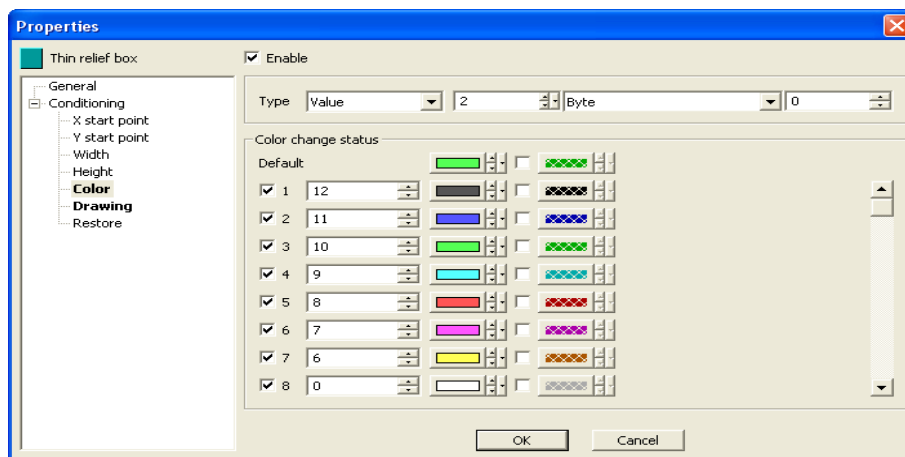


3. Visualize the object layout popup menu (the object must be selected) by clicking with the right mouse button on the object and selecting “Object Properties”: the object properties window will be displayed
4. In the “Conditioning” group set the following parameters:
  - “Enable conditioning” → ☒ (enabled)



5. In the “Color” group set the following parameters:

- “Enable” → “☒” (enabled)
- “Type” → “Value”, “2”, “Byte”, “0”
- “Color change status” → activate all first-column check-boxes (“☒”)  
→ insert values “12”, “11”, “10”, “9”, “8”, “7”, “6”, “0”






The box color will be conditioned with the parameters recently settled (‘conditioned’ is a term used in SUPER-FLASH environments to indicate a dynamic object, it means that some of its attributes change according to variable runtime values).

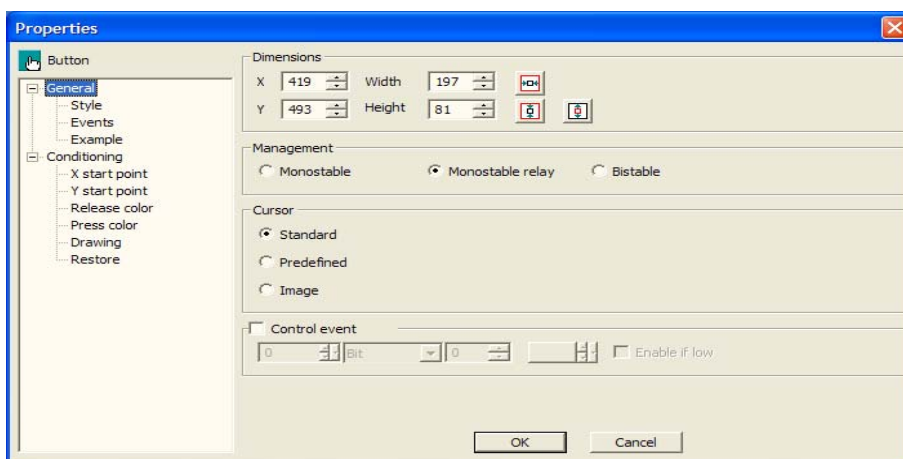
The parameters settled to conditionate the “Value” type used in the example is:

- the object color is bound to the byte 0 (LSB) of the variable 2
- when the byte value is 12, the box color will be bright black (dark grey)
- when the byte value is 0, the box color will be white
- when the byte value is 8, the box color will be bright red
- when the byte value is different to those specified, the box color will be as indicated as default
- .....

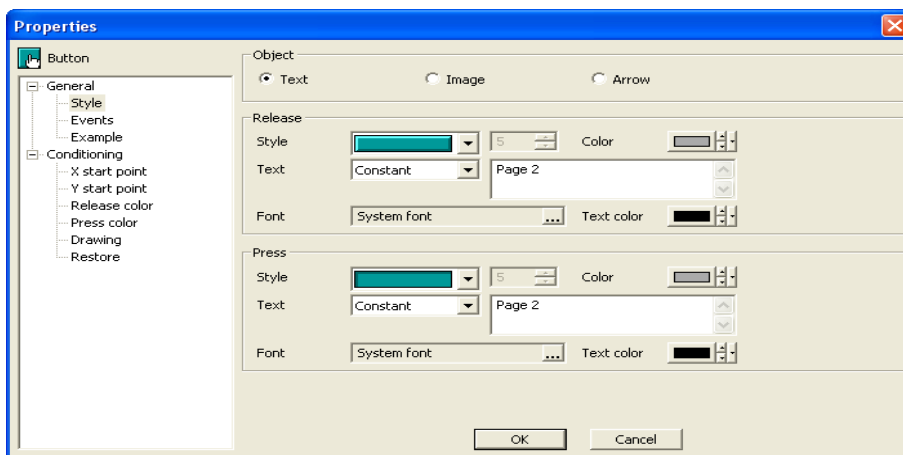
6. “OK” to continue

### 4.4.3.2 Inserting a button

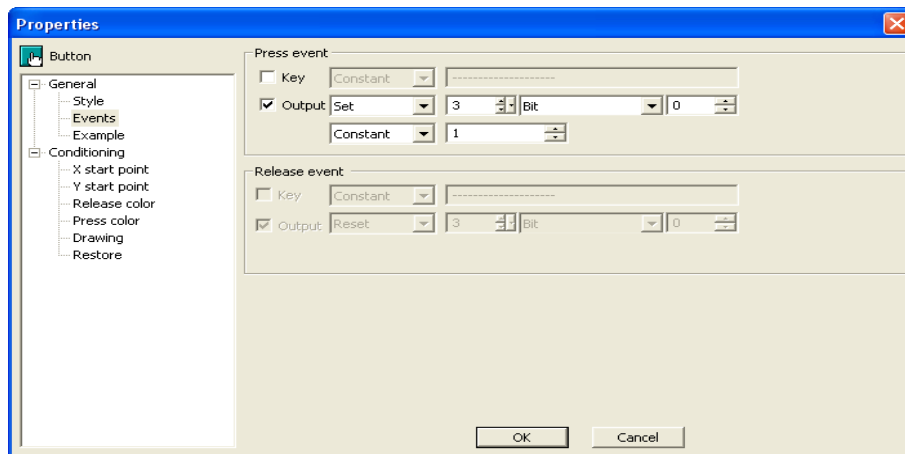
1. Draw a button object by selecting the  ("Button") option in the object bar. Click on the edit area to then "stretch" the mouse while holding down the mouse left button. Drop it once satisfied with the expected object dimensions.
2. In the "General" group set the following parameters:
  - "Management" → "Monostable relay"
  - "Cursor" → "Standard"
  - "Control event" → "□" (disabled)
  - Act on buttons  e  to automatically calculate object dimensions



3. In the "Style" group set the following parameters:
  - "Release - Text" → "Constant", "Page 2"
  - "Press - Text" → "Constant", "Page 2"
  - set colors as you prefer



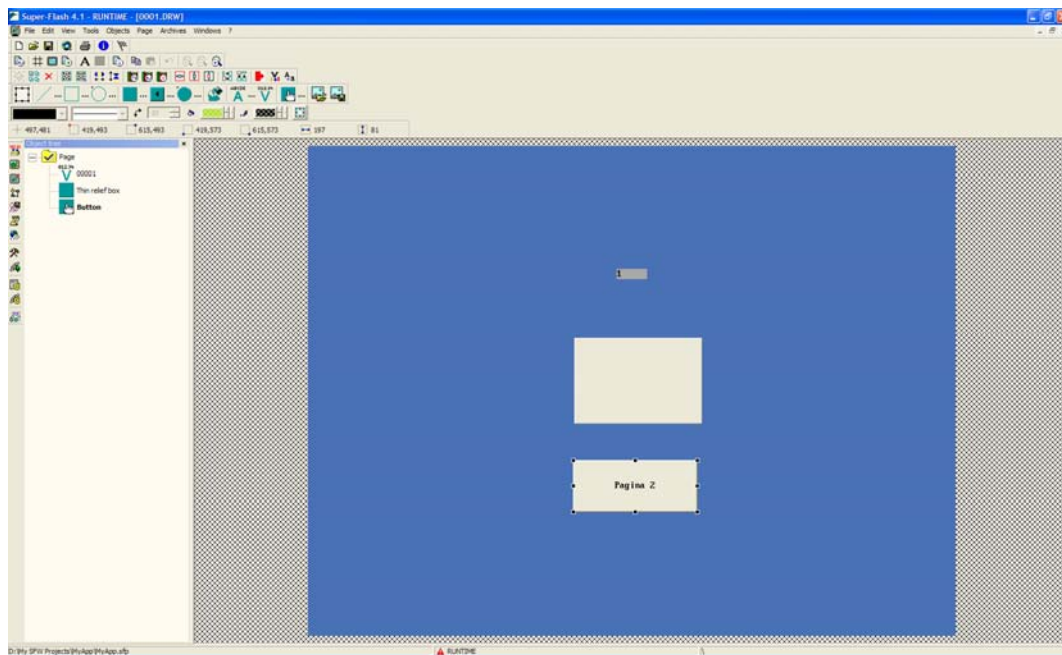
4. In the “Events” group set the following parameters:
- “Press event - Output” → “☒” (enabled), “Set”, “3”, “Bit”, “0”, “Constant”, “1”



Setting the above mentioned parameters means that:

- the button is bound to the bit 0 of the variable 3
- when clicking the button the bit will be set (Set) to 1

5. “OK” to continue






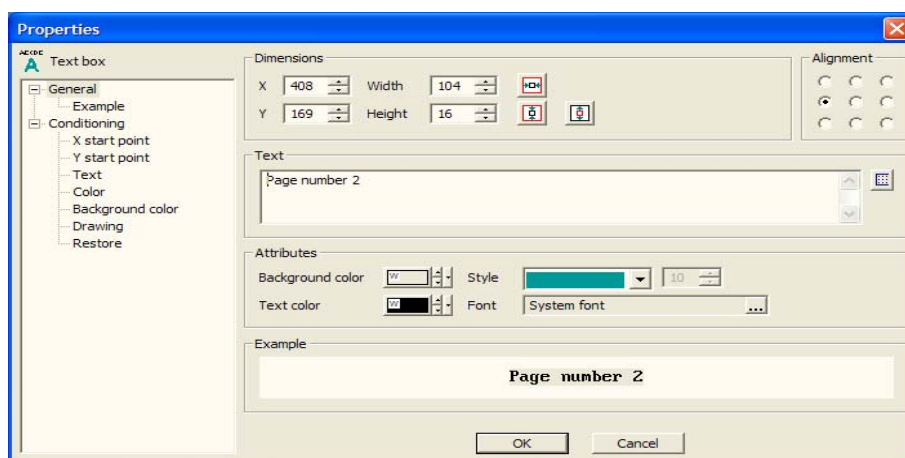
6. Select from the “Page” menu ► “Save”


#### 4.4.4 Creating a second page



1. Select from the “Archives” menu ► “Pages”
2. Select the page 1 in the list
3. Select from the “Edit” menu ► “Insert new object”: the page properties window will be displayed. The second page inherits the properties of the first one by default (it is created by duplicating the one selected in the list), therefore it is enough to change just few parameters.
4. In the “General” group set the following parameters:
  - “Code” → “2”
  - “Description” → “Page number 2” (or another description)
5. “OK” to continue

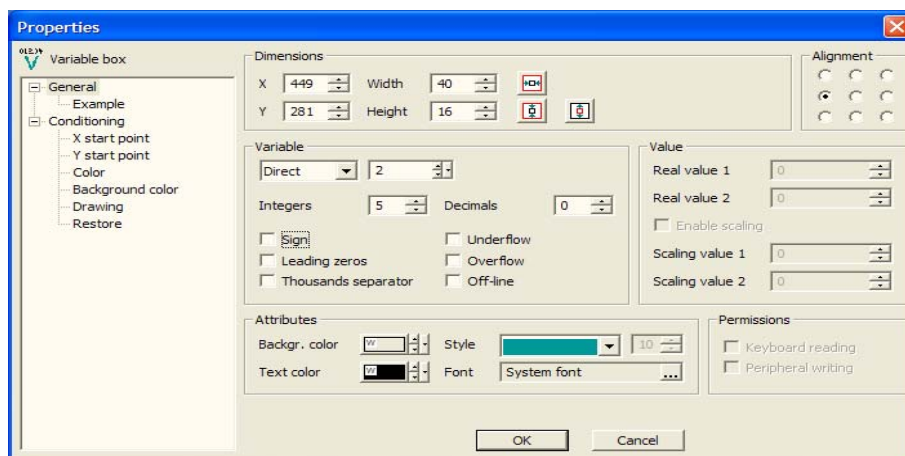
#### 4.4.5 Editing the second page

1. Activate the page editor by clicking twice on page 2
2. Design a text object by selecting the  (“Text”) option in the object bar and by clicking then on the page edit area. The text properties window will be displayed.
3. In the “General” group set the following parameters:
  - “Text” → “Page number 2” (or another text)
  - set colors as you prefer
  - Act on buttons  e  to automatically calculate object dimensions






4. “OK” to continue
5. Design a variable object by selecting the  (“Variable”) option in the object bar and by clicking then on the page edit area. The variable properties window will be displayed.

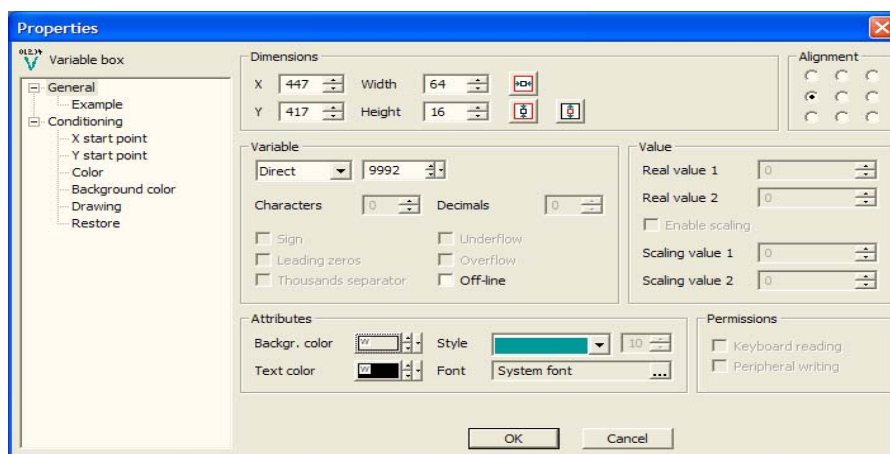
6. In the “General” group set the following parameters:
  - “Variable” → “Direct” “2”
  - “Integers” → “5”
  - set colors as you prefer
  - Act on buttons  e  to automatically calculate object dimensions



7. “OK” to continue

8. Design another variable object by selecting the  (“Variable”) option in the object bar and by clicking then on the page edit area. The variable properties window will be displayed.

9. In the “General” group set the following parameters:
  - “Variable” → “Direct” “9992” (see Note 1)
  - set colors as you prefer
  - Act on buttons  e  to automatically calculate object dimensions

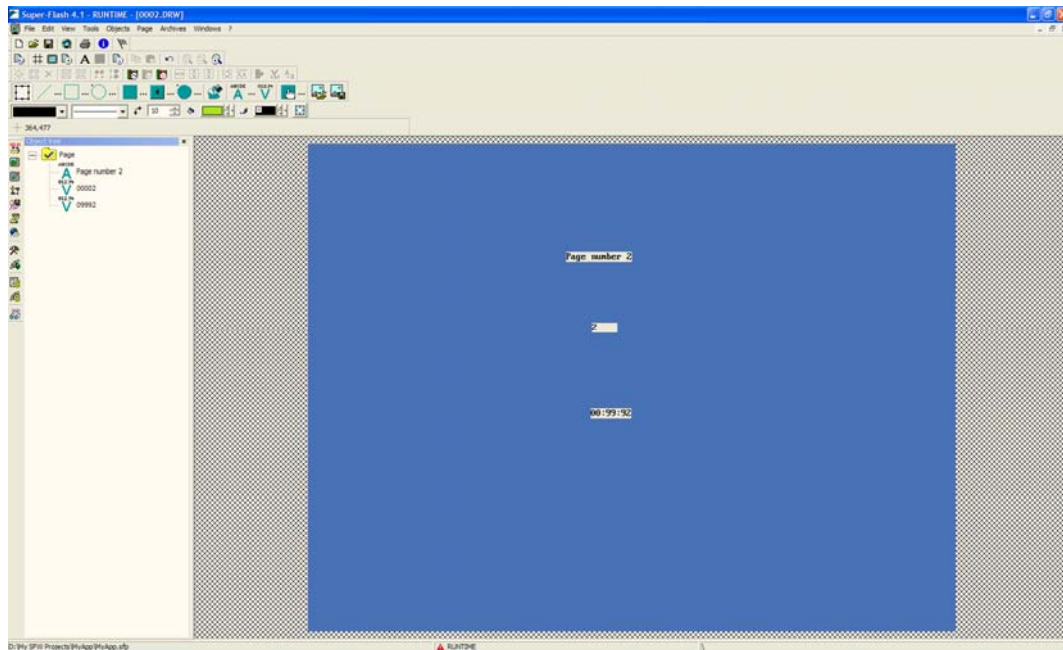


10. “OK” to continue

### Note

- 1) The variable 9992 is the system variable supplying the current time.

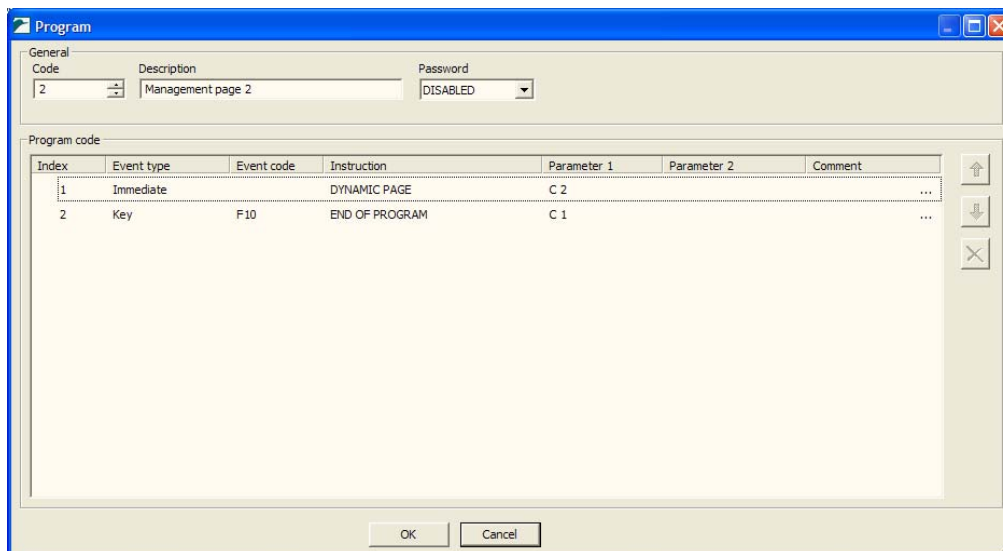




11. Select from the “Page” menu ► “Save”

#### 4.4.6 Creating a second program

1. Select from the “Archives” menu ► “Programs”
2. Select from the “Edit” menu ► “Insert new object”; the program properties window will be displayed
3. Set the following program general information:
  - “Code” → “2”
  - “Description” → “Management page 2” (or another description)
  - “Password” → “DISABLED”
4. Insert an instruction for a page display by setting the following parameters:
  - “Event type” → “IMMEDIATE”
  - “Instruction” → “Display” → “DYNAMIC PAGE”
  - “Parameter 1” → “CONSTANT”, “2”
5. Insert an instruction to end the program by setting the following parameters:
  - “Event type” → “Key”
  - “Event code” → “F10” (click on the input area and press down the **F10** function key)
  - “Instruction” → “Prog. Structure” → “END OF PROGRAM”
  - “Parameter 1” → “CONSTANT”, “1”



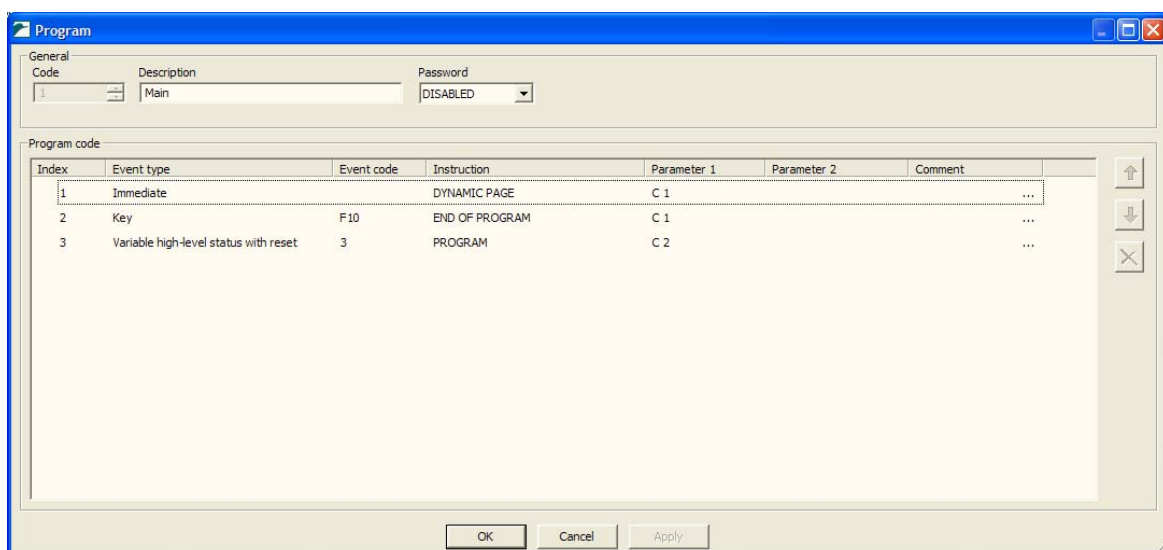
6. "OK" to continue

#### 4.4.7 Modifying the first program

1. Open the program editor by clicking twice on program 1
2. Insert an instruction to call the program 2 by setting the following parameters:
  - "Event type" → "Variable high-level status with reset"
  - "Event Code" → "3"
  - "Instruction" → "Prog. Structure" → "PROGRAM"
  - "Parameter 1" → "CONSTANT", "2"

This instruction controls the variable 3 status.

It is used as an output variable for the button inserted in the first page and its status specifies the access to the second program which controls the display of the second page.



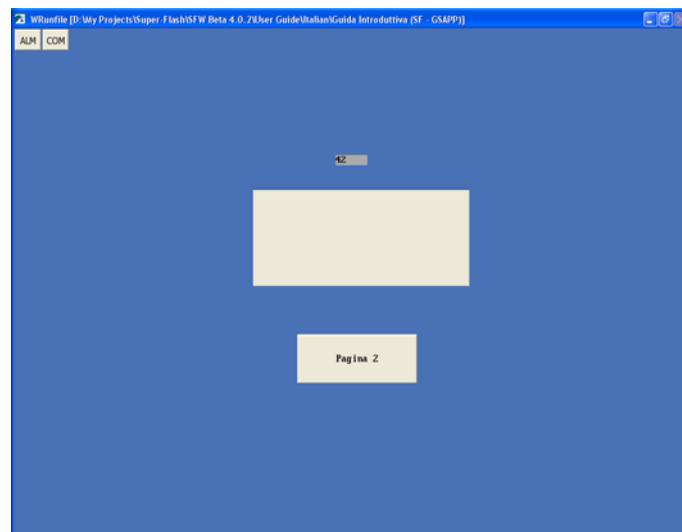
3. "OK" to continue



## 4.5 Test of Phase 2

Now we will check the functionalities added to **MyApp** application: the dynamic box and the use of a button to pass to another page.

1. Launch the MODBUS server application “**Getting Started (SrvMBUS)**”
2. Launch the **MyApp** application by selecting the “**Tools**” menu option ► “**Run Wrunfile**”
3. Confirm the message requesting the software activation code
4. Confirm the message indicating the application launch as demo



5. Modify the value of the address number 2 in the **SrvMBUS** application (by specifying one of the values configured in the box color conditioning section: 6, 7, 8 etc.), the box color will change.
6. Click on the “**Page 2**” button to display the second page



The workflow will be as specified below:

- by clicking on the button in the first page, a value of 1 will be set to the third variable
- a high-level status (value distinct of 0) will be pointed out in the first program for the third variable
- the variable will be immediately set to zero (the event will be 'high-level status with reset')
- a call to the second program will be executed
- the second page will be displayed

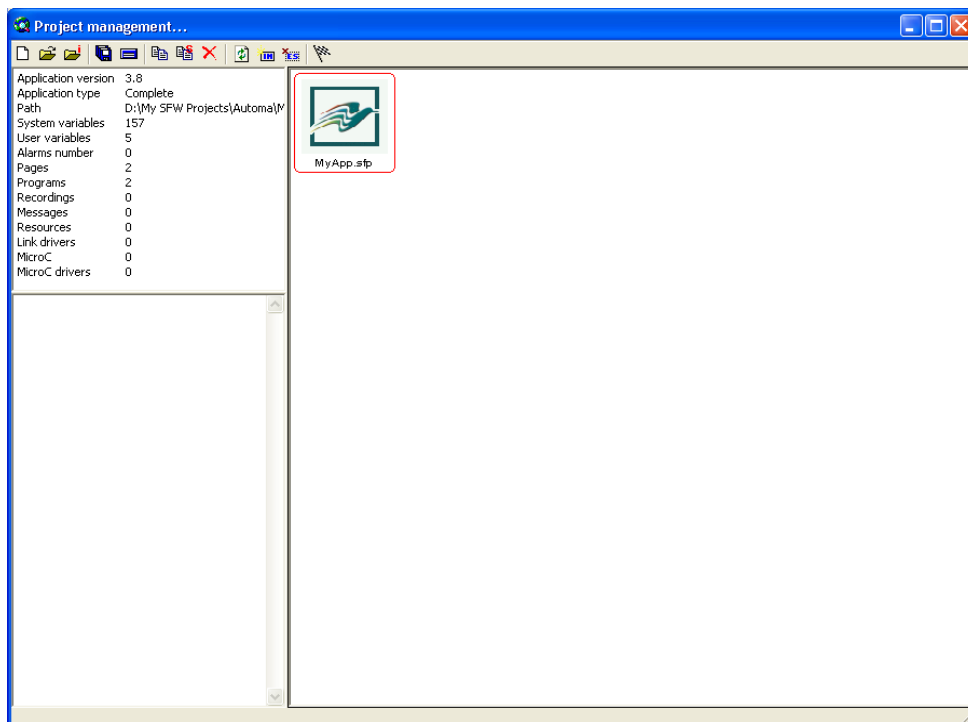
7. Press down the **F10** function key to return to the first program (first page)

8. Press down the **F10** function key to terminate the **MyApp** application

## 4.6 Creating an installation disk for the target machine

This paragraph illustrates the functionalities supplied by the development system for the creation of an installation disk for the target machine.

1. Select from the “File” menu ► “Close” to close the project
2. Select from the “File” menu ► “Project management”; the project management window will be displayed



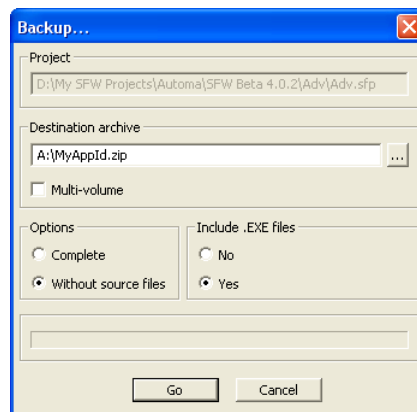
3. Select the **MyApp** application (**MyApp.sfp**) present on the right side of the window
4. Activate the “Project backup” option

This option has a double function as it allows end-users to do a complete project backup and create an application installation disk for the target machine.

The complete project backup must contain the source files required for successive modifications.

The setup, instead, requires the inclusion of the .EXE files in order to execute properly the application on the target machine (the Runtime Engine). Including them and, therefore, the source files on the target machine are according to end-users preferences.

5. Set the following options for the installation disk:
- "Destination archive" → "A:\MYAPPID.ZIP" (complete path filename)
  - "Options" → "Without source files"
  - "Include .EXE files" → "Yes"



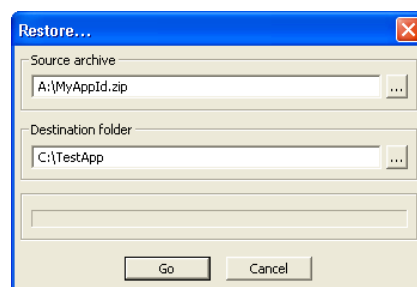
6. "Go" to make the installation disk

In the selected path, besides the zip file containing the application, two new files will be created:

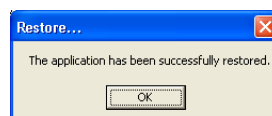
- "SFRESTOR.EXE" : tool for application restore management
- "DUNZIP32.DLL" : DLL useful for "SFRESTOR.EXE"

## 4.7 Installing on the target machine

1. Launch the "SFRESTOR.EXE" file available on the installation disk



2. Select the "Source archive" (.ZIP file) containing the setup application
3. Select the "Destination folder" where to install the application
4. "Go" to install the application; the following message will appear when finished:



5. "OK" to finish

## 4.8 Executing the application on the target machine

1. Use “Browse” to access the folder where the application was installed (in our example “C:\TESTAPP”)
2. Launch the application by clicking twice on the “WRUNFILE.EXE” file

## 4.9 Connecting a MODBUS peripheral

The objective of this paragraph is about supplying end-users with useful information for a serial connection test with an external peripheral device in case of having one supporting the MODBUS protocol.

To achieve this, it is necessary to modify in **MyApp**:

- variables
- communication channel

### 4.9.1 Modifying variables

In the **MyApp** application MODBUS variables are defined with the following addresses:

- Variable 1: “000\W\00001”
- Variable 2: “000\W\00002”

If these addresses are not valid for the peripheral device or if end-users are willing to access other resources, it is necessary to set new parameters in the “Address” field in the variables properties.

The MODBUS address format is: **nnn\t\aaaaa**

Data	Description	Range
nnn	Unit identifier	0 ÷ 255
t	Resource type	W, R, A
aaaaa	Address	0 ÷ 65535

Details regarding resource types:

Resource	Dimension	Access	MODBUS data access functions
W	16-bit word	R/W	function 3 for reading function 16 for writing
R	16-bit word	R/W	function 3 for reading function 6 for writing
A	16-bit word	R	function 4 for reading

Each MODBUS peripheral user guide should clearly provide information about how accessing resources.. Substantially it should indicate:

- the unit identifier
- the MODBUS function number for data access (number used to extract the resource type to indicate in the SUPER-FLASH variable)

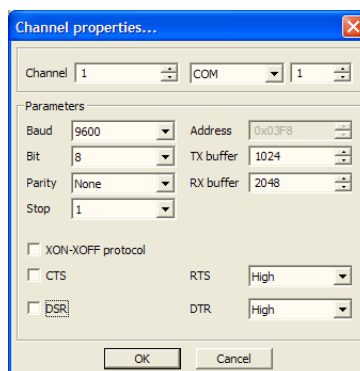
Sometimes this information is coded in just one value as for example: 30000, 40000, etc. In this case the value can be interpreted as follows:

Address	Meaning	Address for SUPER-FLASH
30000	address 0, through function 3	W\0000
41234	address 1234, through function 4	A\1234

Note that the SUPER-FLASH variable needs the unit identifier expressed in decimal; sometimes in the peripheral user guides they are expressed in hexadecimal.

## 4.9.2 Modifying the communication channel

1. Select from the menu “Tools” ► “Configure communication channels”, the communication channels configuration window will be shown
2. Go to the “Windows” group
3. Check the channel properties by clicking twice on channel 1



4. Modify the channel by selecting the “COM” type (serial) and the number at your choice (“1-32”)

Be sure that the selected COM is present in the system “Properties system \ Peripherals management \ COM Ports”.

5. Set the communication parameters “Baud”, “Bit”, “Parity” e “Stop” as required by the peripheral

The necessity of handshake hardware signals ("CTS", "DSR", "RTS" and "DTR") depends on connection types and/or peripherals specifications. A typical case where they are required is in RS485 connections (2 wires).

In this case it is necessary to check out the manual for converters regarding if there is an automatic line switch or if a RTS (Request to Send) signal is required and also if the CTS (Clear to Send) signal should be controlled.

Some typical examples are:

- Line switch through RTS/CTS signals

- "RTS"	→ "Send"
- "CTS"	→ "☑" (enabled)

The output RTS signal is enabled to control the transmission line switch before sending the request message to the peripheral.

The input CTS signal is considered as approval to send characters.

- Line switch through RTS signal

- "RTS"	→ "Send"
- "CTS"	→ "☐" (disabled)

The output RTS signal is enabled to control the transmission line switch before sending the request message to the peripheral.

The input TS signal is not controlled.

- Automatic line switch; no handshake configuration required

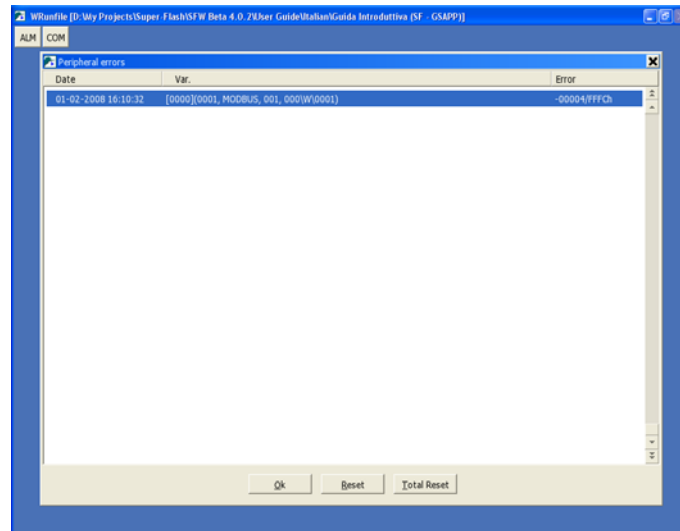
- "RTS"	→ "High"
- "CTS"	→ "☐" (disabled)

Most of the multiseriial cards managing RS485 connections use this type of control.

## 6. "OK" and "Save" to continue

## 4.10 Solving communication problems

In case of having communication errors it is possible to visualize a detailed window by clicking on the “COM” button, as specified in the [Test of Phase 1](#) paragraph (page 24).



The window will show the following information:

- error date and time
- information regarding the variable: [control] code, peripheral, channel and address
- error code: shown between brackets in both decimal / hexadecimal formats

Errors may be generated by the communication driver or by the peripheral.

Driver errors:

Code	Description
-2	sending of request message failed  - handshake problems (for example: CTS low signal) - incorrect connection
-4	no message received, timeout  - incorrect unit identifier - incorrect communication parameters - incorrect connection - timeout very low - invalid resource type or address specified in the variable
-5	message received without terminators  - timeout very low - RS485 line switch problem



Code	Description
-6	incomplete message received - timeout very low
-7	wrong message BCC - incorrect communication parameters - disturbance problems - RS485 converter meets interference characters when commuting lines
-8	message received contains BCC only
-9	answering id identifier not expected
-10	message received contains different code according to message sent - invalid resource type or the address specified in the variable
-11	different data number read according to what expected
-13	communication channel not open - communication channel not configured
-14	characters sending not allowed - handshake problems (for example: low CTS signal) - incorrect connection

Peripheral errors:

Code	Description
1	the request message contains an invalid function number - invalid resource type specified in the variable
2	the request message contains an invalid address - invalid address specified in the variable
3	the request message contains an invalid field
4	the request has been approved but not executed due to peripheral errors
5	the request has been approved but is on hold as it requires long execution times
6	the request has not been approved because the peripheral is busy executing other procedures
7	the request has not been approved because there are control problems
8	the request has been approved but not executed due to a parity error in peripheral memory

## 5 Software potential functionalities

SUPER-FLASH main potentialities are described in the table below.

Communication	<ul style="list-style-type: none"><li>○ SUPER-FLASH offers numerous drivers for the most popular PLCs and for field buses connected through RS232, RS422 and RS485 serial communication networks, as well as, through Ethernet. End-users may develop their own communication drivers with MICROC or by developing DLLs to integrate with those supplied by peripheral manufacturers.</li></ul>
Design	<ul style="list-style-type: none"><li>○ The design of application graphic interfaces is completely at end-users preferences: SUPER-FLASH integrates an advanced graphic editor that permits the edit of special high-level objects (variables, trends, buttons, bar-graphs, etc.) along with the basic ones (text, squares, etc.).</li></ul>
Variables	<ul style="list-style-type: none"><li>○ The variables archive puts information controlled by applications all together. The variables archive keeps field data orderly, as well as, applications generated data. Variables can be of different types: some special types allow the manipulation of complex information.</li></ul>
Trends	<ul style="list-style-type: none"><li>○ Trends allow easier recording and monitoring of real-time analogical variables.</li></ul>
Alarms	<ul style="list-style-type: none"><li>○ The "Alarms management" functionality allows to control easily the most common problematics connected to alarm detection and warning.</li></ul>
Recipes	<ul style="list-style-type: none"><li>○ This functionality allows to identificate a significant values group for reading or writing on PLCs, as well as, to save it on hard disks.</li></ul>
Event management	<ul style="list-style-type: none"><li>○ Event management provides the secure recording of all process events onto disk. It facilitates the synchronisation between SUPER-FLASH applications (HMI) and those of MES level (EPS) in charge of data collection and production management.</li></ul>
Multiplatform	<ul style="list-style-type: none"><li>○ An application developed with SUPER-FLASH may run without significant alterations on PCs equipped with Windows (XP-PRO, XP-E, Windows 7 32/64 bit) and Linux with WINE.</li></ul>
Multilingual	<ul style="list-style-type: none"><li>○ SUPER-FLASH supplies tools for the development of multilingual applications. If adequately configured, applications can be translated into various languages also after applications development phases.</li></ul>
Integration with MicroC	<ul style="list-style-type: none"><li>○ MICROC is a .C compiler that builds programs to be executed by SUPER-FLASH applications. With MICROC, end-users may extend enormously the development system functionalities thanks to the availability of more than 600 functions.</li></ul>
Integration	<ul style="list-style-type: none"><li>○ The ability to integrate with the exterior is one of SUPER-FLASH fundamental features and it offers various opportunities to those wanting to integrate a supervision application with their own process control or business information systems.</li></ul>

Further information regarding SUPER-FLASH functionalities can be found in SUPER-FLASH brochure.

## 6 Contacting Automa

Automa remains at your disposal for further technical and commercial information about its products and services, as well as, satisfy customer needs.



Company information	
Company	Automa srl
Address	Via Don Mazzucotelli, 6 - 24020 Gorle (BG)
Telephone number	(+39) 035.32.33.911 (15 lines in a.r.)
Fax number	(+39) 035.32.33.999
Website	<a href="http://www.automa.it">http://www.automa.it</a>
E-mail address	<a href="mailto:automa@automa.it">automa@automa.it</a>
	<a href="mailto:staffcom@automa.it">staffcom@automa.it</a>



In order to efficiently answer each doubt and avoid difficulties, Automa offers an efficient Free Technical Support service: Tuesdays and Thursdays from 9:30am to 12:00pm.

Free Technical Support	
By telephone	(+39) 035.32.33.911 (15 lines in a.r.)
By e-mail	<a href="mailto:assist@automa.it">assist@automa.it</a>