Application manual

ScreenMaker

5.12

Document ID: 3HAC035956-001

Revision: -

ABB AB
Robotics Products
721 68 Västerås
Sweden

Manual overview

About this manual

This manual describes how to create FlexPendant Graphical User Interfaces (GUIs) using ScreenMaker.

Usage

ScreenMaker is designed for simplicity of use, and its functionality is basic and intuitive. This manual describes the background of GUI development, followed by descriptions of the menus and commands, and followed by a tutorial.

Who should read this manual?

This manual is intended for ScreenMaker users, for example:

- Robot programmers
- PLC programmers
- Robot System integrators

Prerequisites

The reader should have a basic knowledge of:

- RobotStudio
- RAPID
- Working on Windows platform
- GUI development

Organization of chapters

The manual is organized in the following chapters:

Chapter	Contents
1.	Describes the ScreenMaker development tool, as well as GUI and FlexPendant concepts.
2.	Describes how to manage projects in ScreenMaker and the various menus and commands used in the application.
3.	Serves as an example and takes you through the steps involved in designing the GUI screens.
4.	Contains a list of frequently asked questions.

References

Reference	Document ID
Operating manual - RobotStudio	3HAC032104-001
Operating manual - IRC5 with FlexPendant	3HAC16590-1

Revisions

Revision	Description
-	First edition. Released with RobotWare 5.12.02.

1 Introduction

1.1. Introduction to ScreenMaker

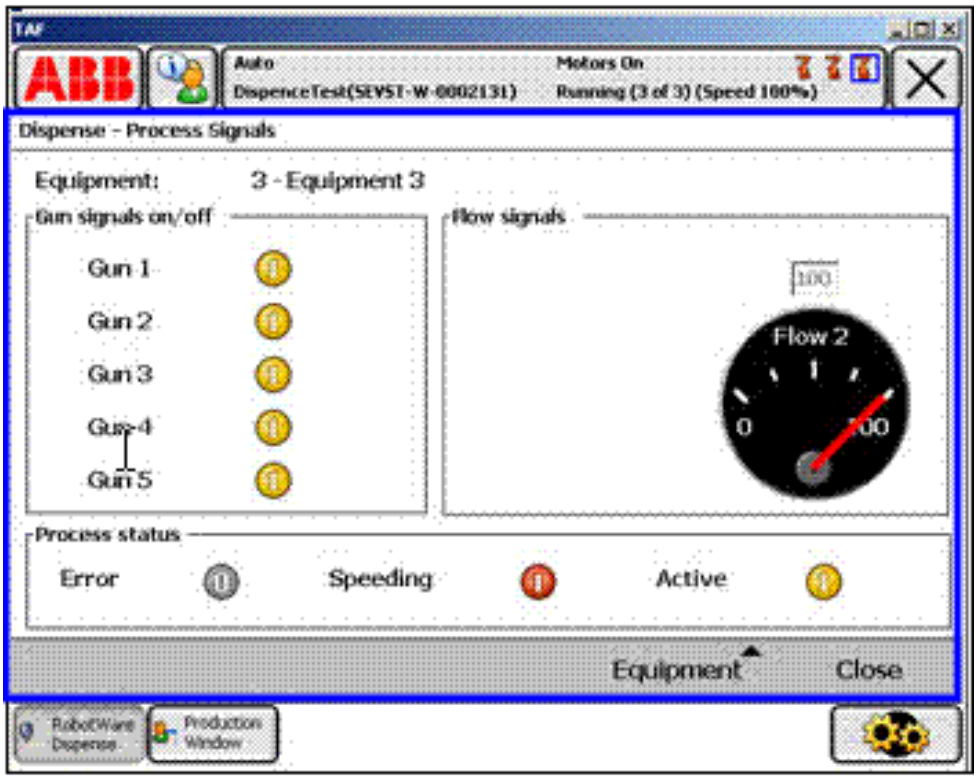
What is ScreenMaker?

ScreenMaker is a tool in RobotStudio for developing custom screens. It is used to create customized FlexPendant GUIs without the need to learn Visual Studio development environment and .NET programming.

Why ScreenMaker?

A customized operator interface on the factory floor is the key to a simple robotic system. A well-designed custom operator interface presents the right amount of information at the right time and in the right format to the user, as such the training time and downtime (due to operating errors) are minimal. However, customized user interfaces are expensive and very time-consuming to develop. Currently, an understanding of some object-oriented programming languages (such as C, C++; VB and C#) and development framework (.NET, Visual Studio) are required to develop screens. Since, this is a requirement for IT professionals and not for the robotics industry whose workforce is generally accustomed to simple programming languages such as BASIC and RAPID; ScreenMaker is used.

GUI concepts



xx080000226

A GUI makes it easier for people to work with industrial robots by presenting a visual front to the internal workings of a robotic system. For FlexPendant GUI applications, the graphical interface consists of a number of screens, each occupying the user window area (the blue box

Continues on next page

Continued

in the figure above) of the FlexPendant touch screen. A FlexPendant screen is then composed of a number of smaller graphical components in a design layout. Typical controls (sometimes referred as widgets or graphic components) include buttons, menus, images, and text fields.

A user interacts with a GUI application by:

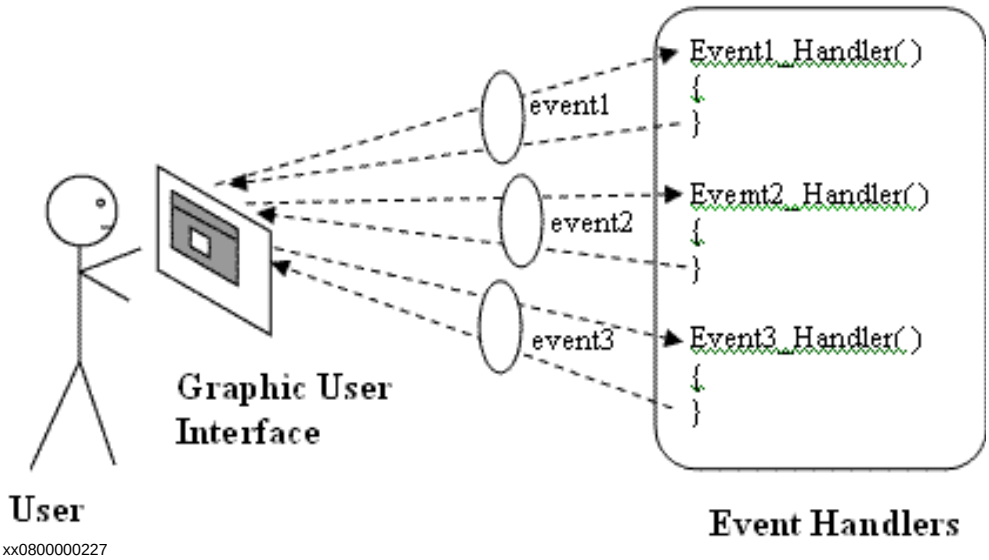
- Clicking a button
- Selecting from a menu
- Typing a text in a text box
- Scrolling

An action such as clicking a button is called an event. Whenever an action is performed, an event is sent to the GUI application. The exact content of an event is solely dependent on the graphic component itself. Different components trigger different types of events. The GUI application responds to the events in the order generated by the user. This is called event-driven programming, since the main flow of a GUI application is dictated by events rather than being sequential from start to finish. Due to the unpredictability of the user's actions, one major task in developing a robust GUI application is to ensure that it works correctly no matter what the user does. Of course, a GUI application can, and actually does, ignore events that are irrelevant.

The event handler holds sets of actions to be executed after an event occurs. Similar to trap routines in the RAPID program, the event handler allows the implementation of application-specific logic, such as running a RAPID program, opening a gripper, processing logic or calculating.

In summary, from a developer's point of view, A GUI consists of at least two parts:

- *the view part*: layout and configuration of controls
- *the process part*: event handlers that respond to events

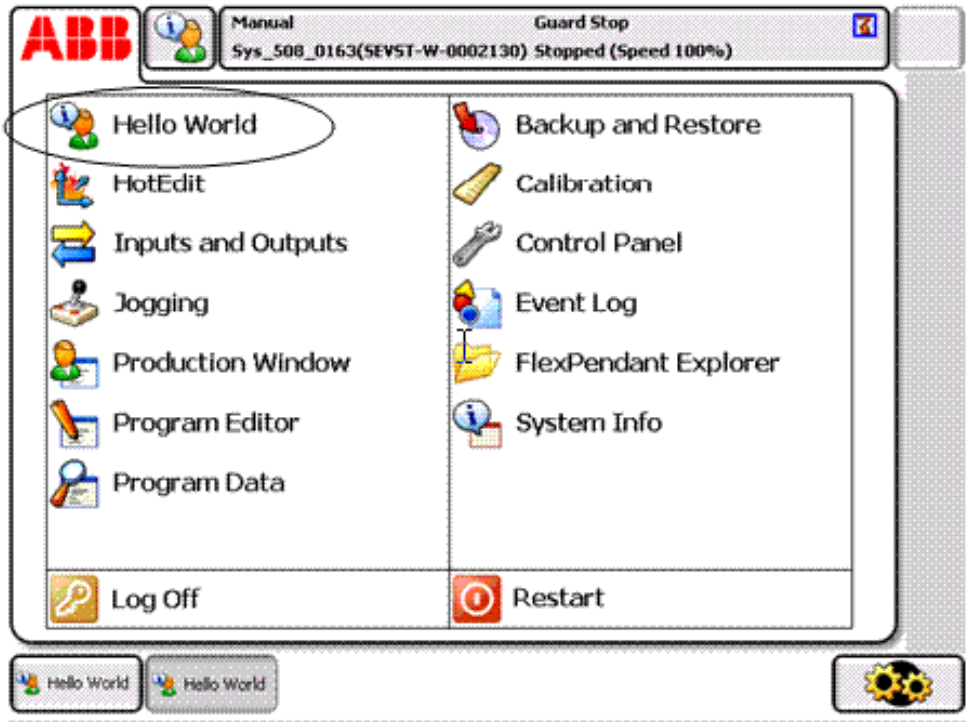


xx0800000227

Modern GUI development environments often provide a form designer, a WYSIWYG tool to allow the user to select, position and configure the widgets. As for event handlers, typically the developer must use a special programming language recommended by the development environment.

Continues on next page

FlexPendant concepts



xx0800000228

Running Windows CE, the ABB FlexPendant has limited CPU power and memory compared to a PC. A custom GUI application must therefore be placed in the designated folders on the controller hard drive before being loaded. Once loaded, it can be found in the ABB menu as seen in the figure above. Clicking the menu item will launch the GUI application.

As the robot controller is the one actually controlling the robot and its peripheral equipment by executing a RAPID program, a GUI application needs to communicate with the RAPID program server in order to read and write RAPID variables and set or reset I/O signals.

It is essential for RAPID programmers to understand that there are two different softwares controlling a work cell: an event-driven GUI application running on the FlexPendant, and a sequential RAPID program running in the controller. These reside on different CPUs and use different operating systems, so communication and coordination are important and must be carefully designed.

1.2. Installing ScreenMaker

Overview

This section describes installing ScreenMaker application on your computer. It is an add-in in RobotStudio and is launched from the RobotStudio application.

System requirements

The following are the system requirements to be met:

Software requirements	• RobotStudio 5.12 or later with Premium license activated • Microsoft .NET Compact Framework 2.0 • Robot Application Builder (RAB) 5.12 or later
Hardware requirements	For more information on the recommended hardware, see *RobotStudio Release Notes*.
Operating system	For more information on the supported Operating Systems, see *RobotStudio Release Notes*.

Prerequisites

To install and use ScreenMaker, the following requirements have to be met.

Before ...	**you must ...**
installing ScreenMaker	ensure that the following applications are installed: • RobotStudio with Premium license
testing on VC/RC	See *Testing on Virtual controller/Real controller on page 11*.

Installing ScreenMaker

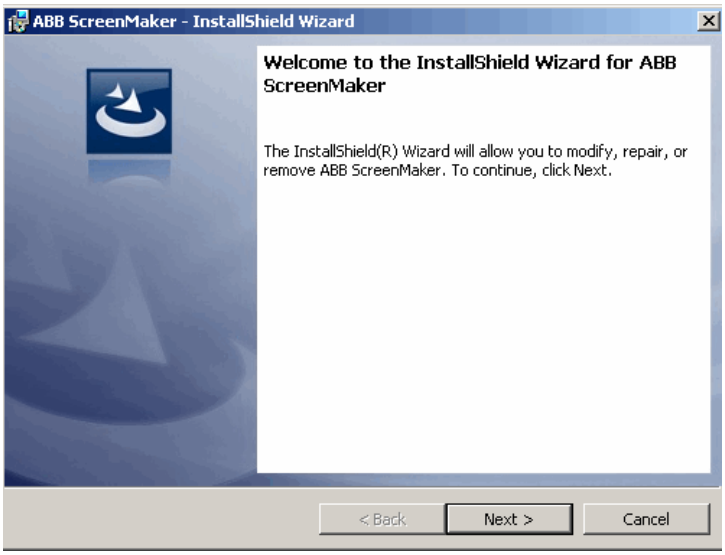
Use this procedure to install ScreenMaker:

NOTE!

You should have administrator privileges on the PC before installing ScreenMaker.

1. Double-click **SetupScreenMaker.exe**.

   The InstallShield Wizard window appears.



en0900000453

2. Click **Next** and follow the instructions to complete the installation.

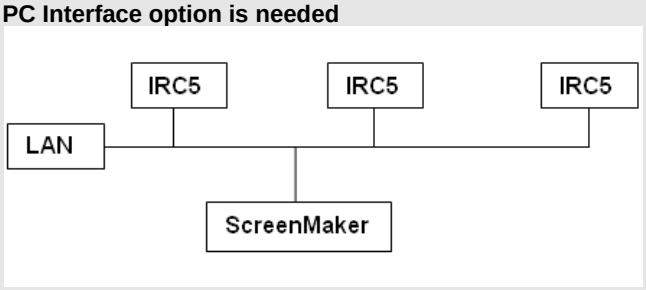
Continues on next page

Testing on Virtual controller/Real controller

RobotWare FlexPendant Interface option is required for ScreenMaker applications.

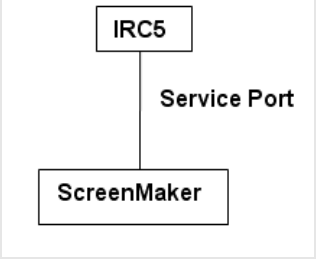
NOTE: RobotWare PC Interface option is required only when using ScreenMaker for Robots on a LAN (to get the data from the controller, bind, and deploy). If there is no PC Interface option, service port can be used to design and deploy screens.

PC Interface option is needed



en0900000723

PC Interface option is not needed



en0900000724

1.3. Terminology

About terms and acronyms

Some terms used in this manual are product specific and crucial for understanding. Moreover, acronyms, words formed from initial letters, are sometimes used instead of long terms. To avoid confusion, important terminology are clarified below.

Definitions

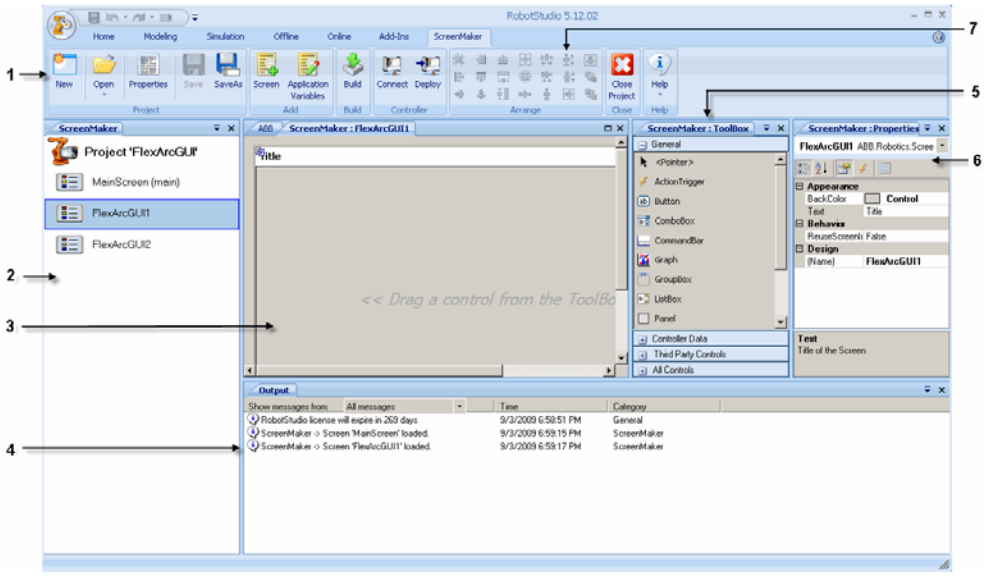
Term	Description
IRC5	ABB's robot controller.
FlexPendant	ABB's hand held device, used with the IRC5 robot controller. It is developed with Microsoft's technology for embedded systems, Windows CE and .NET Compact Framework.
Robot Application Builder	ABB software tool, which enables the development of custom operator interfaces for IRC5. Often referred to as RAB.
Microsoft .NET Compact Framework	Version of Microsoft's .NET framework providing the run-time environment for applications running on embedded devices, such as the FlexPendant. It includes a class library, which is almost a subset of the rich .NET framework for the desktop.
C++,VisualBasic and C#	Programming languages

Acronym	Description
GUI	Graphical User Interface
OS	Operating System
RAB	Robot Application Builder
I/Os	Input /Output signals
WYSIWYG	What You See Is What You Get

1.4. Development environment

Overview

This section presents an overview of the ScreenMaker development environment.



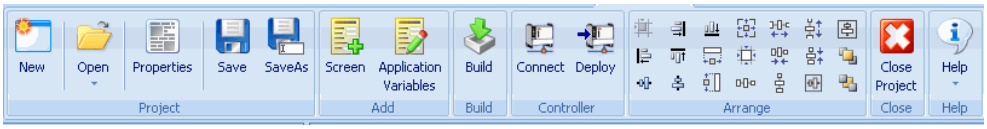
en0900000584

	Parts	Description
1	Ribbon	Displays group of icons organized in a logical sequence of functions. See *Ribbon on page 14*.
2	Project explorer	Shows the active screen project and lists the screens that are defined in the project. For more information, see *Managing ScreenMaker projects on page 26*.
3	Design surface	Layout to design the screen with the available controls. For more information, see *Form designer on page 35*.
4	Output window	Displays information about the events that occur during ScreenMaker development.
5	ToolBox	Displays a list of available controls. For more information. see *ToolBox on page 15*.
6	Properties window	Contains the available properties and events of the selected control(s). The value of the properties can either be a fixed value or a link to an IRC5 data or an Application Variable. For more information, see *Properties window on page 17*.
7	Arrange	Displays icons for resizing and positioning controls on the design surface. See *Arrange on page 14*.

Continues on next page

Ribbon

The ScreenMaker ribbon tab contains a group of icons organized in a logical sequence of functions that facilitates the user in managing SscreenMaker projects.



en0900000452

The Ribbon is categorized into the following groups:

Group	Description
Project	Facilitates the user to manage ScreenMaker project. See *Managing ScreenMaker projects on page 26*.
Add	Facilitates the user to add screen and application variables. See *Managing screens on page 29* and *Managing application variables on page 34*.
Build	Facilitates the user to build a project. See *Building a project on page 33*.
Controller	Facilitates the user to connect and deploy to the controller. See *Connecting to controller on page 32* and *Deploying to controller on page 33*.
Arrange	Facilitates the user to resize and position the controls on the design surface. See *Arrange on page 14*.
Close	Facilitates the user to close a project.
Help	Facilitates the user to open the ScreenMaker help.

Arrange

This toolbar displays icons for resizing and positioning controls on the design surface.

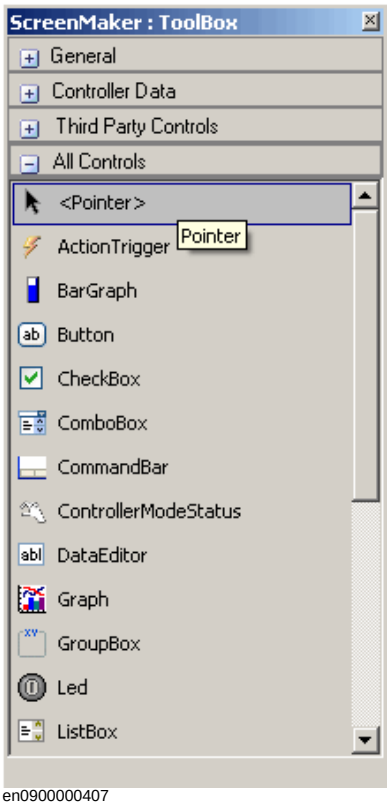
NOTE: The icons are enabled once you select a control or group of controls on the design surface.

en0900000592

Continues on next page

ToolBox

ToolBox acts a container for holding all the available controls that can be placed on a screen.



en0900000407

The following table displays the GUI controls that can be dragged on to the design surface.

Control	Description
ActionTrigger	Allows to run a list of actions when either a signal or rapid data changes
BarGraph	Represents an analog value in a bar
Button	Represents a control that can be clicked. Provides a simple way to trigger an event, and is commonly used to execute commands. It is labeled either with text or an image.
CheckBox	Allows multiple selections from a number of options. They are displayed as a square box with white space (for unselected) or as a tick mark (for selected).
ComboBox	Represents a control that enables to select items from a list Combination of a drop-down list and a textbox. It allows you to either type a value directly into the control or choose from the list of existing options.
CommandBar	Provides a menu system for a ScreenForm
ControllerModeStatus	Displays the mode of the Controller (Auto - Manual)
DataEditor	Represents a text box control that can be used to edit the data.
Graph	Represents a control that plots data with lines or bars.
GroupBox	Represents a Windows control that displays a frame around a group of controls with an optional caption. Is a container used to group a set of graphic components. It usually has a title at the top.

Continues on next page

Continued

Control	Description
LED	Displays a two states value, like a Digital Signal.
ListBox	Represents a control to display a list of items. Allows the user to select one or more items from a list contained within a static, multiple line text box.
NumEditor	Represents a text box control that can be used to edit a number. When the user clicks it, a Numpad is opened.
NumericUpDown	Represents a spin box that displays numeric values.
Panel	Used to group collection of controls.
PictureBox	Represents a picture box control that displays images.
RadioButton	Allows to select only one of a predefined set of options.
RapidExecutionStatus	Displays the execution status of the Controller Rapid Domain (Running - Auto)
RunRoutineButton	Represents a Windows button control that calls a RapidRoutine when clicked
Switch	Displays and lets change a two states value, like a Digital Output Signal.
TabControl	Manages a set of tab pages.
TpsLabel	Very commonly used widget that displays text, a label is usually static, that is, it has nointeractivity. A label generally identifies a nearby text box or other graphic component.

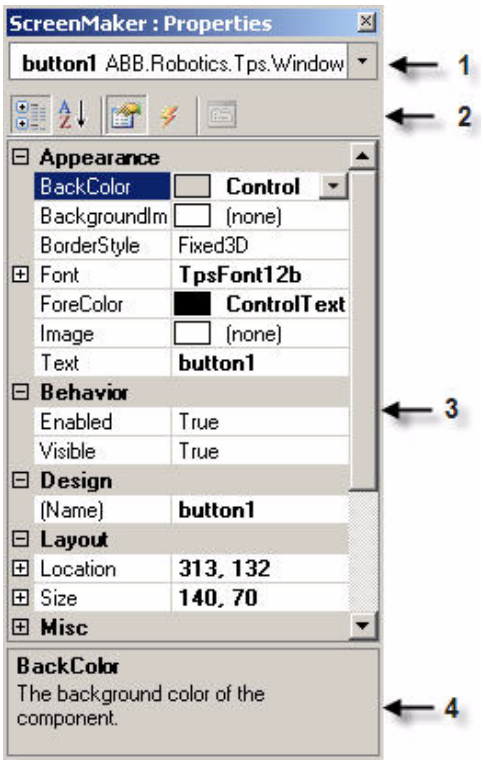
NOTE!

For more information on using these controls and their properties, see the section *Building the user interface on page 18* and the chapter *Using the FlexPendant SDK* of the *Application manual - Robot Application Builder*.

Continues on next page

Properties window

A control is characterized by its properties and events. Properties describe the appearance and behavior of the component, while events describe the ways in which a control notifies its internal state change to others. By changing the value of a property, the controls have a different look and feel, or exhibit different behavior.



en0900000408

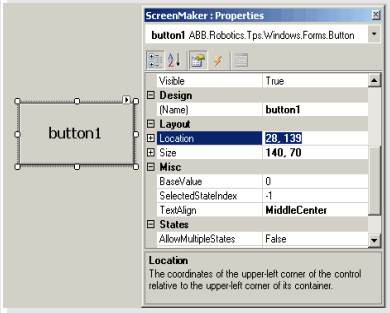
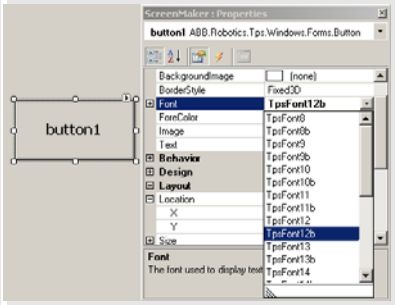
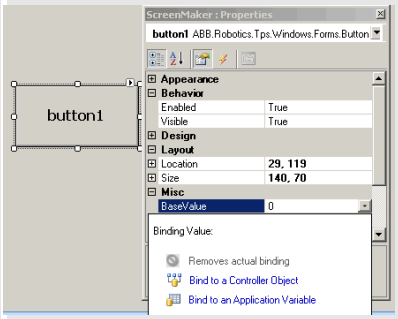
	Element	Description
1	Graphical component name panel	Displays the selected component, and lists the available components of the on the active design screen.
2	Properties window toolbar	 en0900000409 1. Organizes table panel in categories 2. Organizes table panel alphabetically 3. Displays Properties in table panel 4. Displays Events in table panel
3	Table panel	Displays all the properties or events in two-columns. The first column shows the property or event name, the second shows the value of the property or name of the event handler.
4	Information panel	Display information about a property or event.

Continues on next page

Continued

Editing the property value

You can edit the property value of a control from the **Properties window** in three ways:

1	By typing the numerics, strings and text. For example, Location, Size, Name etc.	 en0900000631
2	By selecting the predefined values from the list. For example, BackColor, Font etc.	 en0900000633
3	By entering the values in the dialog box. For example, Enabled, States, BaseValue etc.	 en0900000632

Building the user interface

This section describes building the GUIs using the following controls from the ToolBox.

ActionTrigger

An action trigger initiates an event, such as making a hidden object visible when an action is performed using a control. It allows to run a list of actions when the property value changes. The property value can be bound to a signal, rapid data, or application variable.

ActionTrigger control can also be used to invoke the application from RAPID.

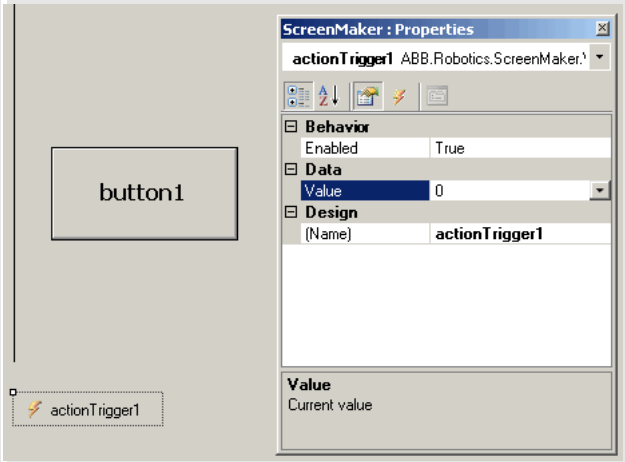
Use this procedure to add an ActionTrigger control::

	Action
1	Drag an **ActionTrigger** control from the **ToolBox** on to the design surface.

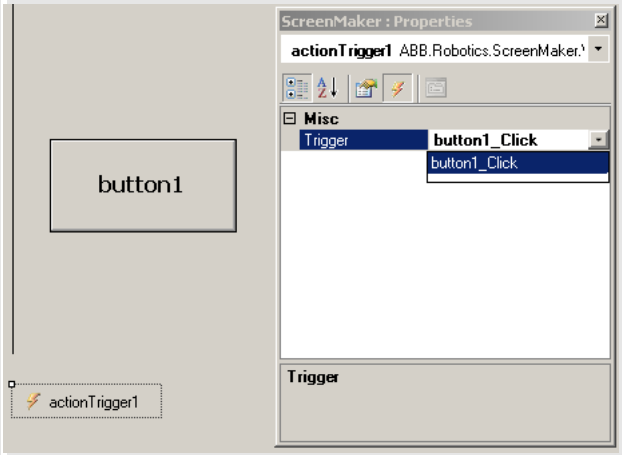
Continues on next page

Continued

	Action
2	You can modify the name, set the default value and configure data binding value for a ActionTrigger control. • To set the values of a property, see *Properties window on page 17*. • You can set the trigger event for an ActionTrigger to any of the event handler created either from a control or from an Events Manager option. To set up the events, see *Setup Events on page 35*. • To configure the data binding values, see *Configuring data binding on page 38*. • To set the application variables, see *Managing application variables on page 34*.



en0900000629



en0900000630

NOTE: An action is not triggered when the screen is launched for the first time, but is triggered when there is a difference in the binded value at any point of time. This functionality is supported only in RobotWare 5.12.02.

Example: Consider a signal being binded to the value property. The value of the signal changes at runtime on performing a specific action. The event handler configured for ActionTrigger control gets triggered based on this value change.

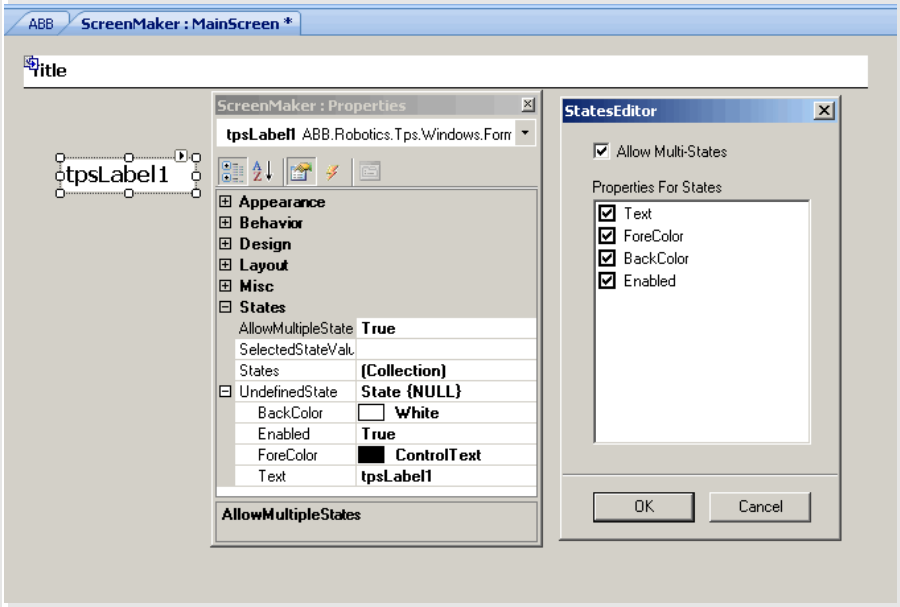
Continues on next page

Continued

TpsLabel

TpsLabel is a standard Windows label that displays a descriptive text.

Use this procedure to add a TpsLabel control:

Step	Action
1	Drag a **TpsLabel** control from the **ToolBox** on to the design surface.
2	You can set the values, setup events, configure data binding values and set the application values for a TpsLabel control. • To set the values of a property, see *Properties window on page 17*. • To set up the events, see *Setup Events on page 35*. • To configure the data binding values, see *Configuring data binding on page 38*. • To set the application variables, see *Managing application variables on page 34*.
3	You can set the option **Allow Multiple States** to true and change the property. 1. Click **AllowMultipleStates**. The Status Editor dialog box appears. 2. Click the check-box **Allow Multi-States**, select the properties to change from Properties For States and click **OK**.  en0900000499

NOTE: Button, PictureBox, and TpsLabel controls also support **AllowMultipleStates** option.

Panel

Panel is used to group a collection of controls.

Use this procedure to add a Panel control:

Step	Action
1	Drag a **Panel** control from the **ToolBox** on to the design surface.
2	You can add a group of controls to a panel.

Continues on next page

Continued

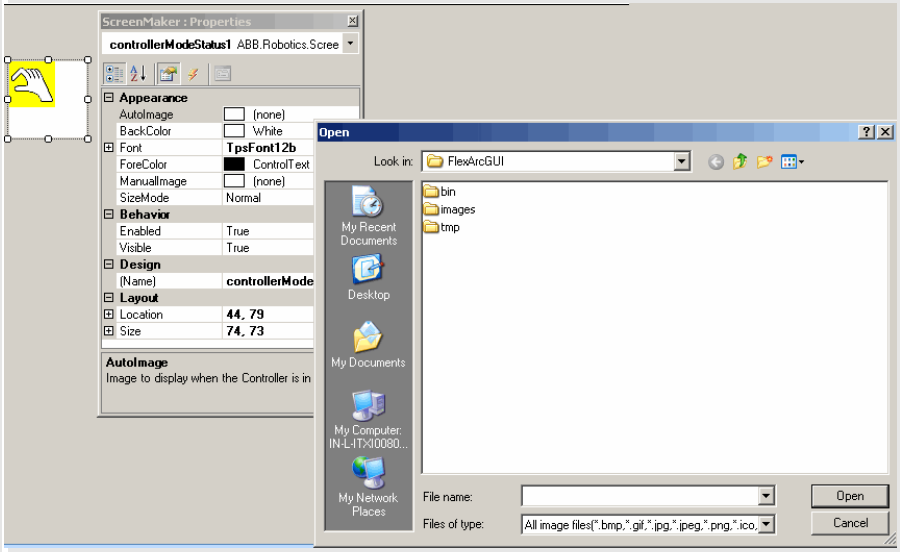
Step	Action
3	You can modify the name, set the default value and binding value for a Panel control. • To set the values of a property, see *Properties window on page 17*. • To set up the events, see *Setup Events on page 35*. • To configure the data binding values, see *Configuring data binding on page 38*. • To set the application variables, see *Managing application variables on page 34*.

NOTE: Currently only EventHandler, CancelEventHandlers, and MouseEventArgs are supported.

ControllerModeStatus

ControllerModeStatus displays the mode of the controller (Auto - Manual).

Use this procedure to add a ControllerModeStatus control:

Step	Action
1	Drag a **ControllerModeStatus** control from the **ToolBox** on to the design surface.
2	You can set the values, setup events, configure data binding values, and set the application variables for a ControllerModeStatus control. • To set the values of a property, see *Properties window on page 17*. • To set up the events, see *Setup Events on page 35*. • To configure the data binding values, see *Configuring data binding on page 38*. • To set the application variables, see *Managing application variables on page 34*.
3	You can select the image to be displayed when the controller is in Auto mode and in Manual mode. • Click **AutoImage** in the Properties window and browse to select the image to be displayed in Auto mode. • Click **ManualImage** in the Properties window and browse to select the image to be displayed in Manual mode.  en0900000536

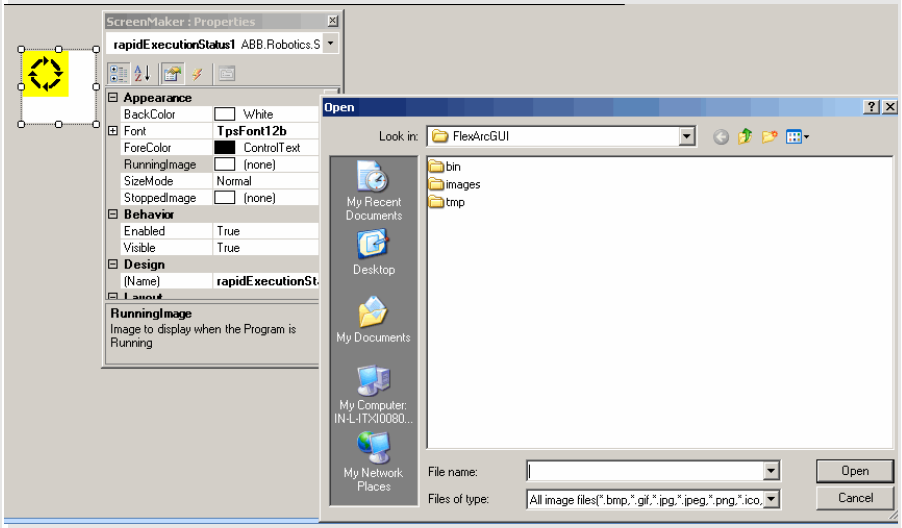
Continues on next page

Continued

RapidExecutionStatus

RapidExecutionStatus displays the execution status of the Controller Rapid Domain (Running - Auto). This control is used

Use this procedure to add a RapidExecutionStatus control:

Step	Action
1	Drag a **RapidExecutionStatus** control from the **ToolBox** on to the design surface.
2	You can set the values, setup events, configure data binding values, and set the application values for a RapidExecutionStatus control. • To set the values of a property, see *Properties window on page 17*. • To set up the events, see *Setup Events on page 35*. • To configure the data binding values, see *Configuring data binding on page 38*. • To set the application variables, see *Managing application variables on page 34*.
3	You can select the image to be displayed when the Program is running and is stopped. • Click **RunningImage** in the Properties window and browse to select the image to be displayed when the Program is running. • Click **StoppedImage** in the Properties window and browse to select the image to be displayed when the Program is stopped.  en0900000537

RunRoutineButton

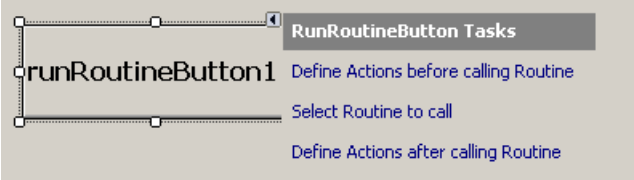
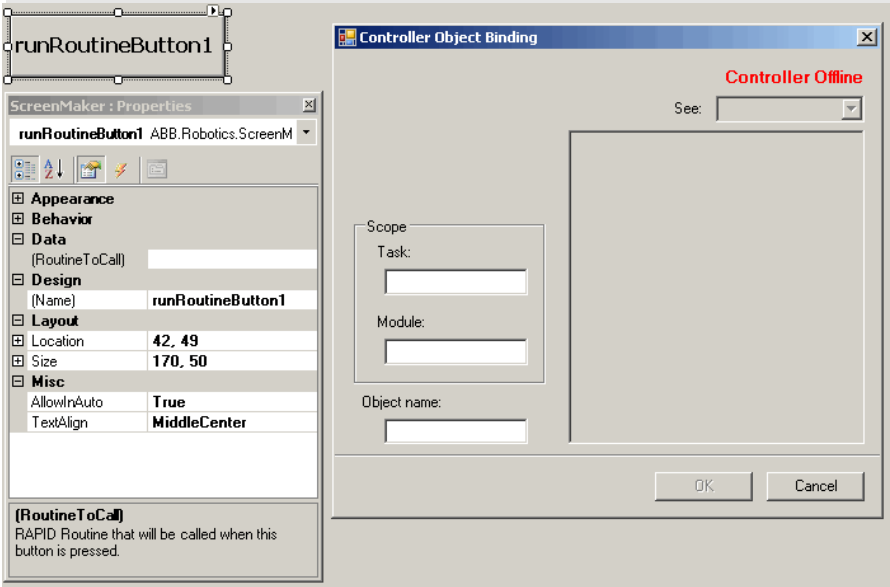
RunRoutineButton represents a Windows button that calls a RapidRoutine when clicked.

Use this procedure to add a RunRoutineButton control:

Step	Action
1	Drag a **RunRoutineButton** control from the **ToolBox** on to the design surface.

Continues on next page

Continued

Step	Action
2	You can set the values, setup events, configure data binding values, and set the application values for a RapidExecutionStatus control. • To set the values of a property, see *Properties window on page 17*. • To set up the events, see *Setup Events on page 35*. • To configure the data binding values, see *Configuring data binding on page 38*. • To set the application variables, see *Managing application variables on page 34*.
3	You can perform the following **RunRoutineButton** tasks from the SmartTag: • Define Actions before calling Routine • Select Routine to call • Define Actions after calling Routine  en0900000538
4	You can perform the following **RunRoutineButton** tasks from the Properties window. • **RoutineToCall** - RAPID Routine will be called • **AllowInAuto** - Indicates if the routine could be called in the Auto mode • **TextAlign** - Indicates the alignment of text **NOTE:** • You cannot bind **RunRoutineButton** to built-in Service routines as only the user defined procedures without arguments are bindable. • Set the PP to task before performing action through **RunRoutineButton**.  en0900000545

1.4. Development environment

2 Managing ScreenMaker projects

2.1. Overview

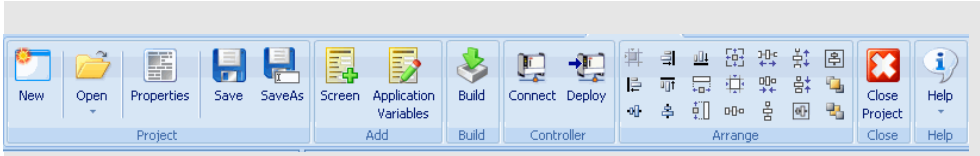
Overview

This chapter describes how to manage projects in ScreenMaker. A complete cycle includes creating, saving, building, connecting, and deploying a ScreenMaker project.

2.2. Managing ScreenMaker projects

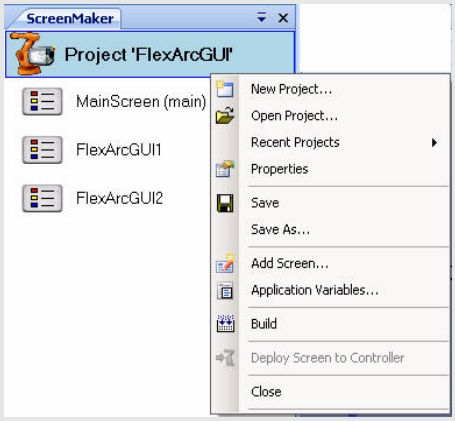
Overview

You can manage a project (create, delete, load, or save) either from ScreenMaker ribbon or Context menu.



en0900000452

ScreenMaker Ribbon



en0900000395

Context menu

Continues on next page

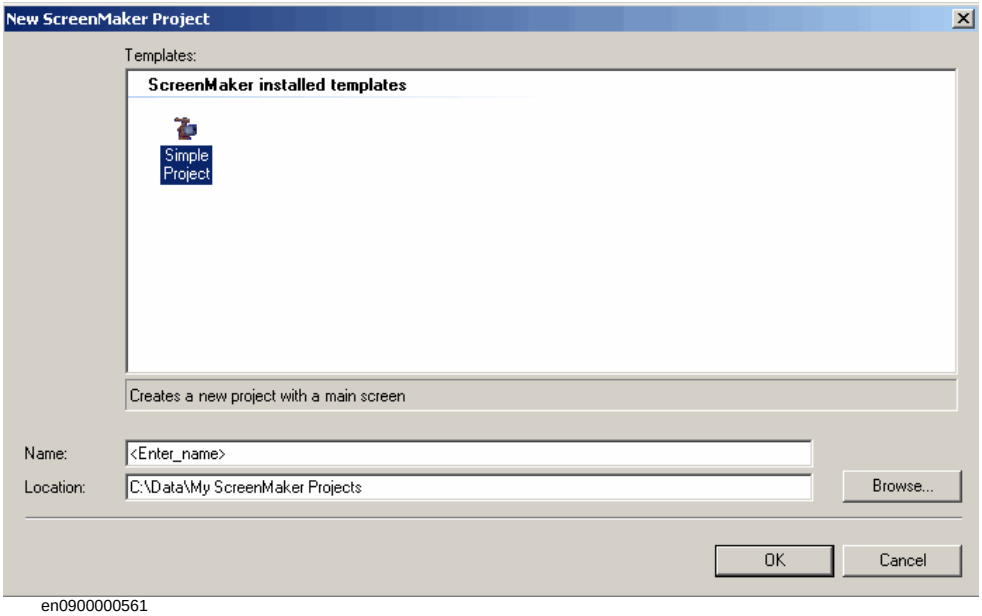
Continued

Creating a new project

Use this procedure to create a new project:

1. Click **New** from the ScreenMaker ribbon or right-click **Project** context menu and select **New Project.**

   The **New ScreenMaker Project** dialog box appears.



en0900000561

2. Enter a new project name and specify a location for the new project.

   A default screen *MainScreen (main)* is added in the tree view.

   By default, the new project is saved on *C:\My Documents\My ScreenMaker Projects.*

3. Click **OK**.

Continues on next page

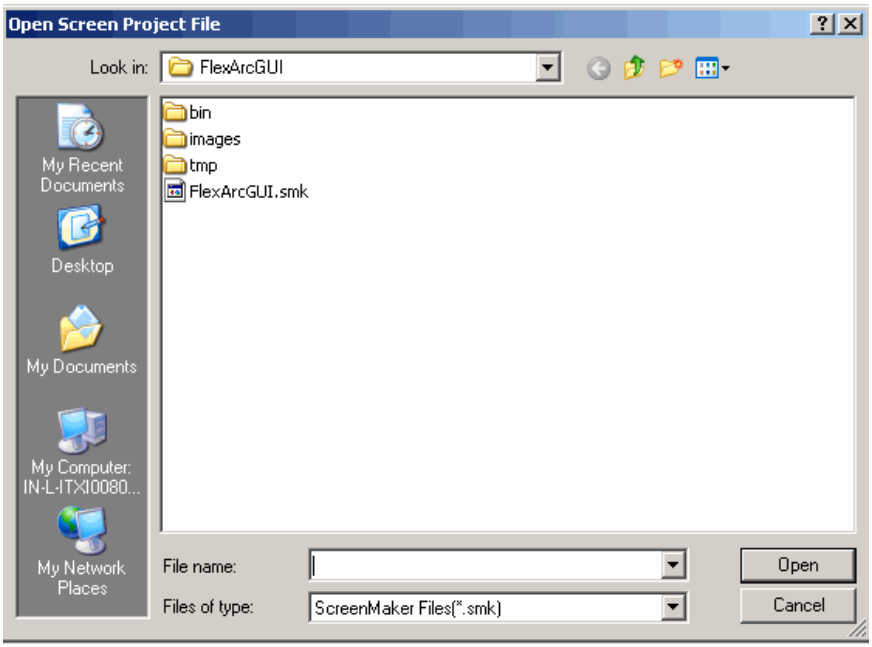
Continued

Loading a project

Use this procedure to load an existing project:

1. Click **Open** from the ScreenMaker ribbon or right-click **Project** context menu and select **Open Project**.

   The **Open Screen Project file** dialog box appears.



en0900000562

2. Browse to the location of the project file to be loaded and click **Open.**

NOTE!

You can also load an existing project using a quick access method.

1. Click **Recent** from the ScreenMaker ribbon or right-click **Project** context menu and select **Recent Projects**.
2. Select the project file from the list of most recently opened projects.

Continues on next page

Continued

Saving a project

To save a project, follow this step:

- Click **Save** from the ScreenMaker ribbon or right-click **Project** context menu and select **Save**.

To save the existing project with a new name, follow this step:

- Click **SaveAs** from the ScreenMaker ribbon or right-click **Project** context menu and select **Save As**.

NOTE!

Project files are saved with the extension *.smk*.

Closing a project

To close a project, follow this step:

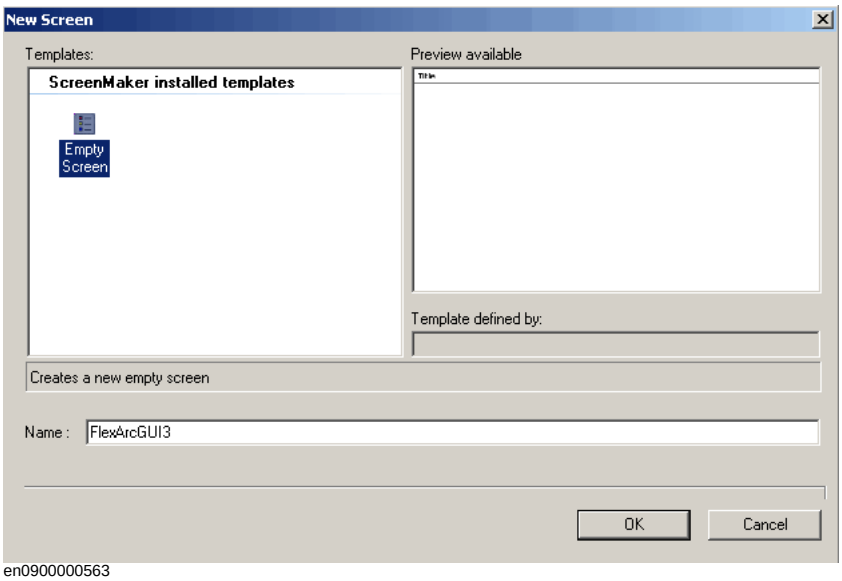
- Click **Close Project** from the ScreenMaker ribbon or right-click **Project** context menu and select **Close**.

Managing screens

This section describes adding, renaming, deleting, and editing a screen.

Creating a screen

1. Click **Screen** from the ScreenMaker ribbon or right-click Project context menu and select **Add Screen**.The New Screen dialog box appears.
2. Enter the name of the new screen in **Name** text box.
3. Click **OK.**



en0900000563

Deleting a screen

1. From the Project tree view, select the screen to be deleted.
2. Right-click and select **Delete.**

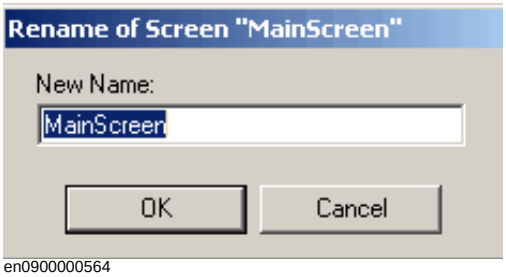
Continues on next page

2.2. Managing ScreenMaker projects

Continued

Renaming a screen

1. From the Project tree view, select the screen to be renamed.
2. Right-click and select **Rename**. The Rename of Screen dialog box appears.
3. Enter the new name in the text box and click **OK**.



en0900000564

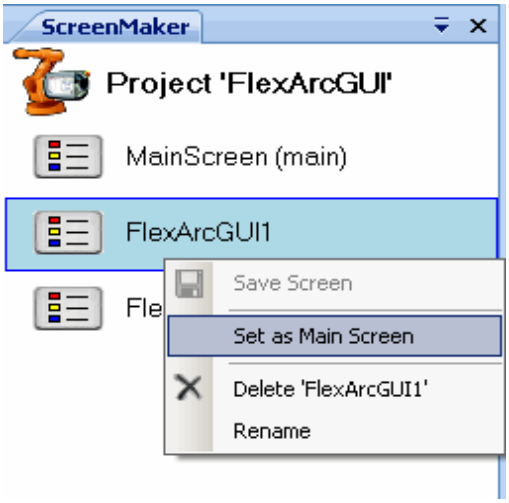
Editing a screen

For information on editing a screen, see *Editing a screen on page 35*.

Changing the Main screen

You have the option to change the main screen in the project to active.

1. From the Project tree view, select the screen to be changed.
2. Right-click and select **Set as Main Screen**.



en0900000546

Continues on next page

Continued

Modifying Project properties

Project properties define the properties of the ScreenMaker project, including how the GUI is loaded and displayed in the FlexPendant.

Use this procedure to modify the project properties:

1. Right-click **Project** context menu and select **Properties**.

   The **Project Properties** dialog box appears.

en0900000394

2. In the **Display** tab under **Texts**, enter the text in **Caption of the Application** to edit the caption.

   The updated caption appears in the ABB menu on the right side.

3. In the **Display** tab under **Images**, browse and select the images of ABB Menu and Taskbar in the respective boxes.

NOTE!

By default, **User default image** and **User Menu image** checkboxes are enabled and the default image *tpu-Operator32.gif* is selected.

Continues on next page

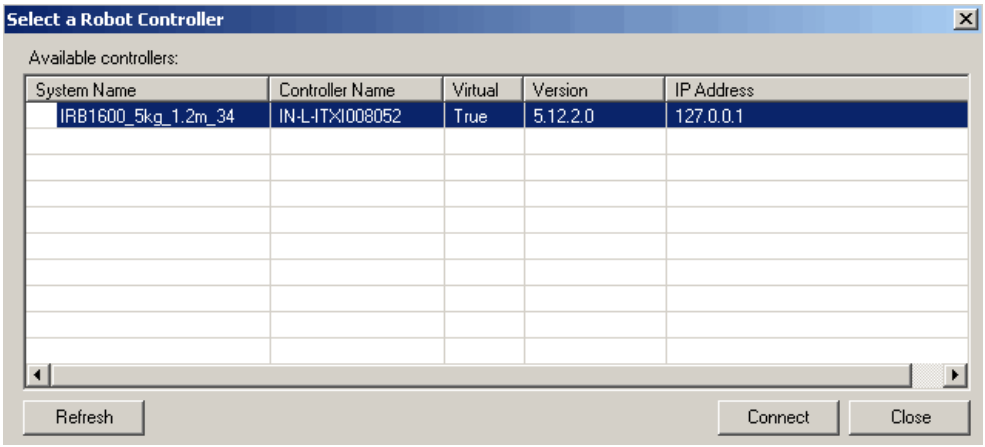
Continued

Connecting to controller

Use this procedure to establish the connection with a controller:

1. From the ScreenMaker ribbon, click **Connect.**

   The **Select a Robot Controller** dialog box appears.



en0900000581

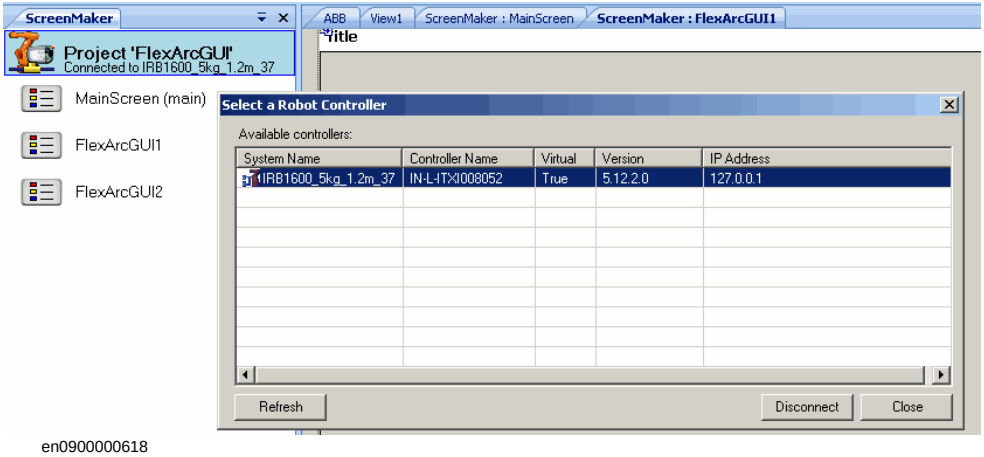
2. Click **Refresh** to find a list of all the available controllers.

NOTE!

- By default, the currently connected controller is highlighted and has a small icon before the row as an indicator.

3. Select the controller to be connected from the list and click **Connect**.

   The connection status is displayed in the Project tree view.



en0900000618

Continues on next page

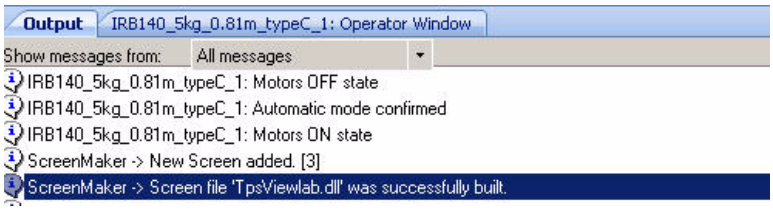
Building a project

The result from building the ScreenMaker project is a set of files including DLL file and images. The ScreenMaker project can be compiled into binary format (.dll) that can be deployed on a FlexPendant.

Use this procedure to build a project:

1. From the ScreenMaker ribbon, click **Build** and select **Build.**

   The result is displayed in the output window.



en0900000404

Deploying to controller

Use this procedure to deploy a ScreenMaker project on a real controller or virtual controller:

1. Connect to the controller you want to deploy. See *Connecting to controller on page 32*.
2. From the ScreenMaker ribbon, click **Deploy**.

   The Download dialog box appears displaying the progress of download. It disappears once the download is successful.

   The **TpsViewxxxxxx.dll** file is downloaded.
3. Restart the controller.

NOTE!

- If a real controller is used, you can reboot the FlexPendant by moving its joystick three times to the right, once to the left, and once towards you.
- If a virtual controller is used, you can reboot the FlexPendant by closing the virtual FlexPendant window.

2.3. Application variables

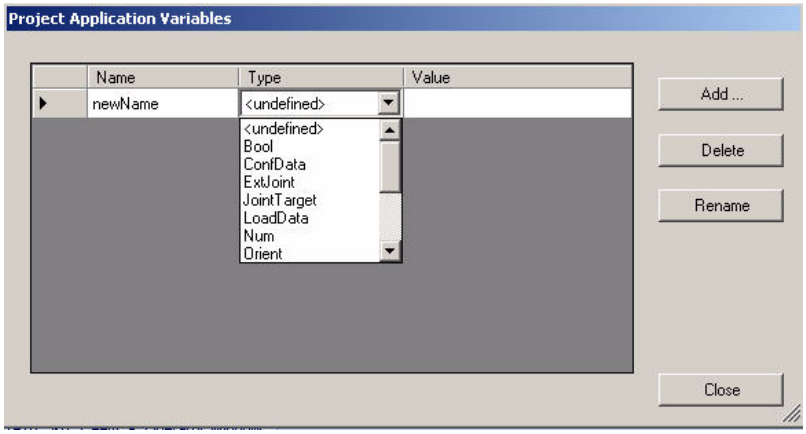
Overview

Application variables, also called temporary variables are created for a ScreenMaker project. During runtime, they reside in the FlexPendant memory. It is used for data sharing and performance improvement. It is similar to a RAPID variable and has a RAPID data type.

Managing application variables

To create, delete, and rename an application variable, follow these steps:

1. In the **Project context** menu, right-click and select **Application Variables**. The **Project Application Variables** dialog box appears.
2. Click **Add** and define the name, type and value of the new variable.
3. Select the variable, click **Delete** to delete a variable.
4. Select the variable, click **Rename**, enter the new name and click **OK** to rename a variable.
5. Click **Close**.



en0900000402

NOTE!

For information on application variable data binding, see *Application variable data binding on page 42*.

2.4. Form designer

Overview

The Form designer is a tool to edit or design a screen. It allows you to design the screen with the required controls and the design surface resembles a FlexPendant screen.

Editing a screen

To edit a screen, follow these steps:

1. Drag a control from the toolbox and drop it on the design surface.

   The Properties window displays all the properties of the control.
2. Select the control and resize or reposition for configuration.

   NOTE!

   You can either select a single control or multiple controls:

   - Single control : Left-click the control on the design surface or select the control from the list in the Properties window.
   - Multiple controls: Left-click on the design surface, drag the mouse and create a window selecting all the controls.
3. Click the smart tag on the upper right corner of the control to perform the basic tasks of configuration. See *Configuring data binding on page 38*.

   NOTE!

   You can perform additional configuration by editing the attributes in the of the Properties window. See *Properties window on page 17*.

Setup Events

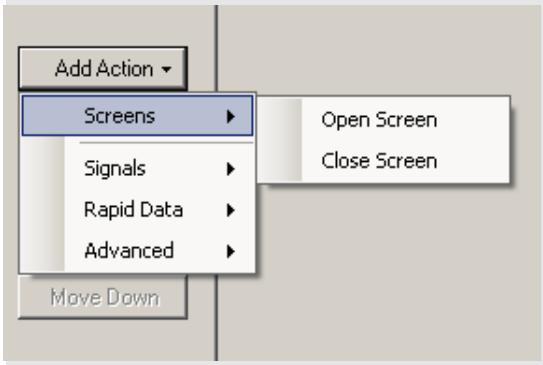
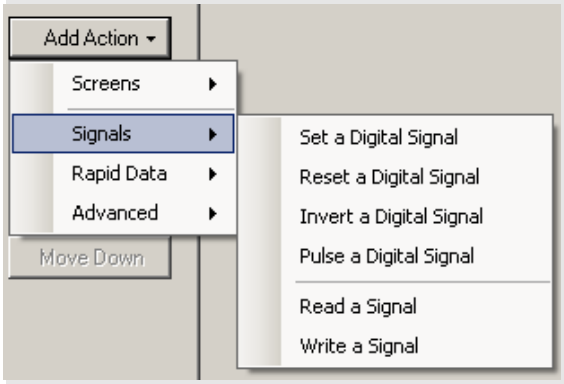
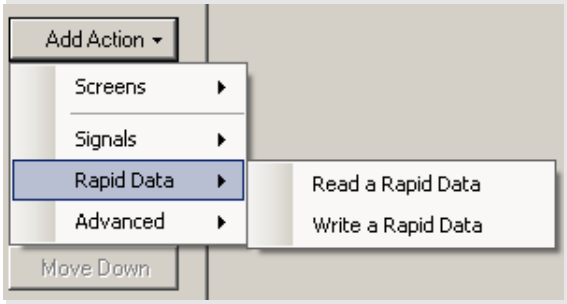
Event handler is a set of actions to be executed after an event occurs. To set up an event, follow these steps:

1. Select the control for which the event handler is to be defined.
2. Open the **Events Panel** dialog box in any one of the following ways:
   - Double-click the control.
   - Right-click the control, select **Events Manager**, click **Add,** enter the name, and click **OK** and close.
   - Click smart tag and select the task from the list.
   - In the Properties window, click **Events** icon and select the desired event from the list.
3. Click **Add Action** to add an action from a predefined list of actions.

Continues on next page

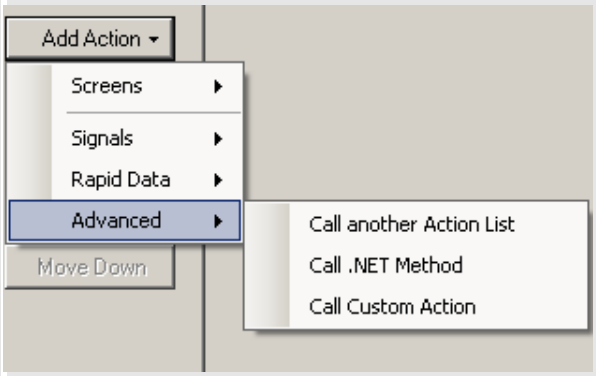
Continued

The following table lists the set of predefined actions:

Screen	• Open Screen • Close Screen  en0900000554
Signals	• Set a Digital Signal • Invert a Digital Signal • Pulse a Digital Signal • Read a Signal • Write a Signal • Reset a Digital Signal  en0900000555
RapidData	• Read a Rapid Data • Write a Rapid Data  en0900000553

Continues on next page

Continued

Advanced	• Call another Action list • Call .NET method • Call Cuctom Action  en0900000552

4. Select the action from the left window and perform the following:
   - Click **Delete** to delete the action.
   - Click **Move Up** or **Move Down** to change the order of execution of actions.
5. Click **OK**.

2.5. Data binding

Overview

Data binding is the mechanism that links a GUI property with an external data source such that whenever the data source is updated the GUI property will be updated automatically and vice verse. Databinding has the following three aspects:

- A *unidirectional* connection means that an update of the data source is reflected by the GUI, or vice versa; a *bidirectional* connection means that updates to either are reflected by the other.
- A *temporal* connection can be suspended and resumed at any time.
- A c*onvertable* connection negotiates between the different data types or formats between the data source and the GUI property.

A screen has to be linked with data to be useful. There are two ways of linking the data with the GUI properties:

- Controller object data binding
- Application variable data binding

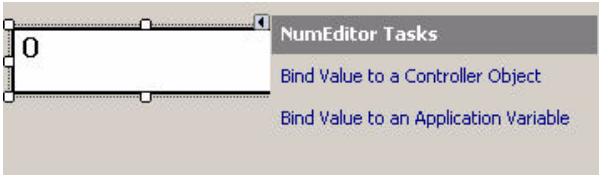
Configuring data binding

Data binding can be configured in the following two ways:

Using smart tag

Smart tags perform basic configuration tasks like binding default GUI property with controller data. Controls either display or edit information have a value to represent the information. Smart tag binds the value to the controller object.

- On the design surface, select the control and click the smart tag. The tasks menu appears.



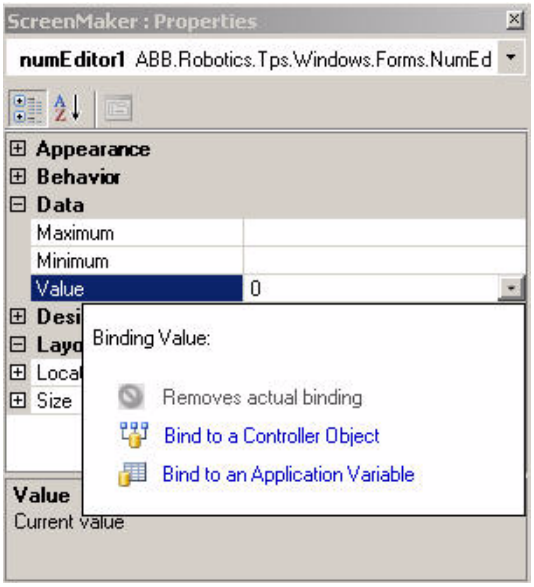
en0900000398

Using Binding menu

1. On the design surface, select the control.
2. In the Properties window, locate the row from the table for binding the value.
3. Select the attribute and click the list to display the Binding menu.

Continues on next page

Continued



en0900000399

Menu	Description
Remove actual binding	Removes the existing data binding
Bind to a Controller object	Select available data in the controller for binding.
Bind to an Application variable	Select available data in project temporary data store for binding.

Configuring data binding for different controls

Almost all the controls defined in the toolbox (except **ComboBox** and **ListBox**) have the following two options for binding values:

- Bind to a Controller Object
- Bind to an Application Variable

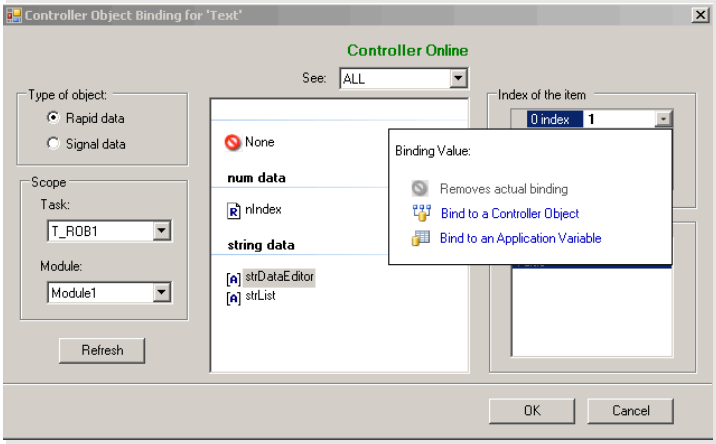
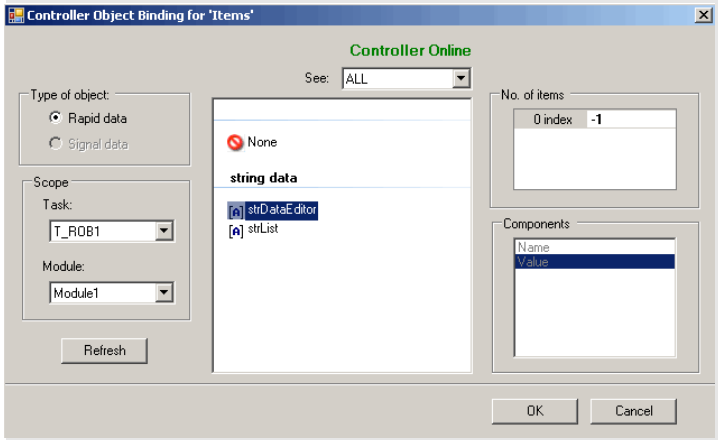
Binding to an array can be done to the following controls:

- DataEditor
- ComboBox
- ListBox

Continues on next page

2.5. Data binding

Continued

Control	Description
DataEditor	The default index value is 1. DataEditor is designed in such a way that the default value of the Rapid array starts with 1 and not 0.  en0900000641
ComboBox and ListBox	The default index value is -1. You can enter the appropriate index value but cannot bind to a controller object or an application variable.  en0900000642

Continues on next page

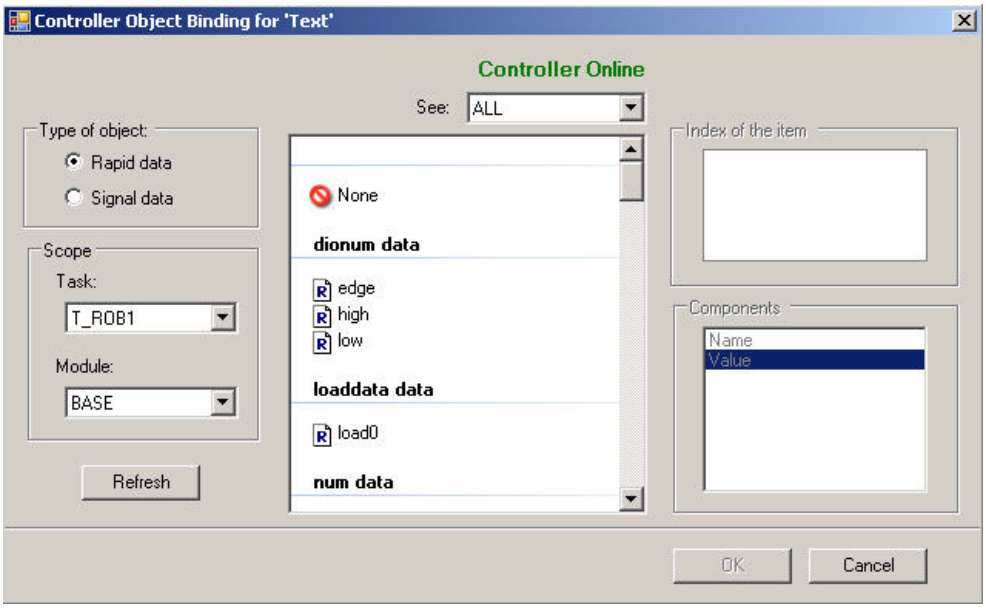
Controller object data binding

Controller object data binding lets you to select the data in the controller for binding.

Use this procedure to set up a binding with controller object:

1. Select **Bind to a Controller Object** either using smart tag or binding menu.

   The **Controller Object Binding** dialog box appears.



en0900000400

2. In the **Type of object** group, select either **Rapid data** or **Signal data**.
3. If you select **Rapid data**, from the **Scope** group, select a task and module from the list.
4. If you select **Signal data**, the **Scope** group is disabled.
5. In the **See** list, select the desired data.

Continues on next page

Continued

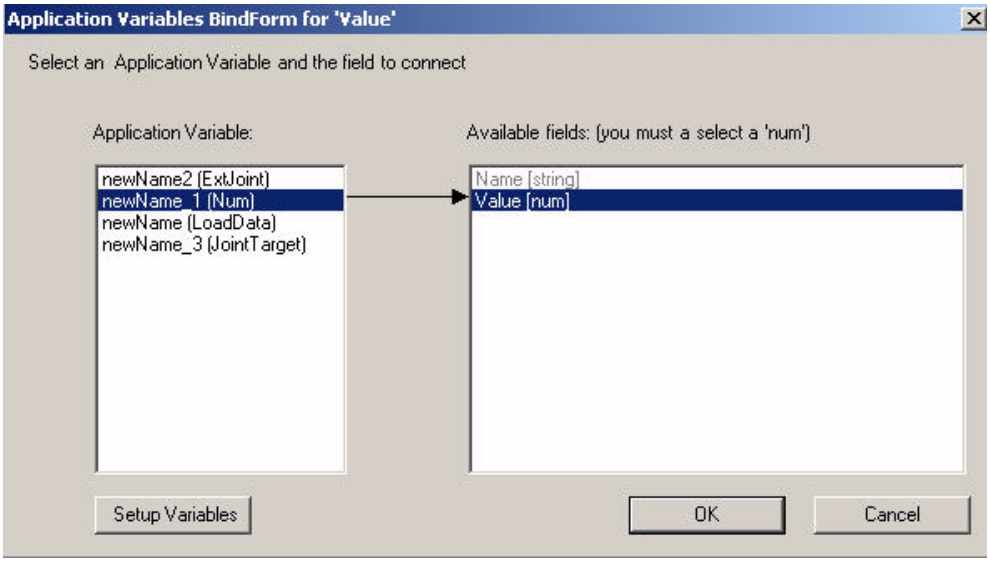
Application variable data binding

Application variables are used for data binding in the same way as controller data. See *Controller object data binding on page 41*.

Use this procedure to set up a binding with application variables:

1. Select **Bind to an Application Variable** either using smart tag or binding menu.

   The **Application Variables Bind Form** dialog box appears.



en0900000403

2. Select an application variable and the field to connect.
3. Click **Setup Variables** to manage the variables.

   The **Project Application Variables** dialog box appears. See *Managing application variables on page 34*.
4. Click **OK.**

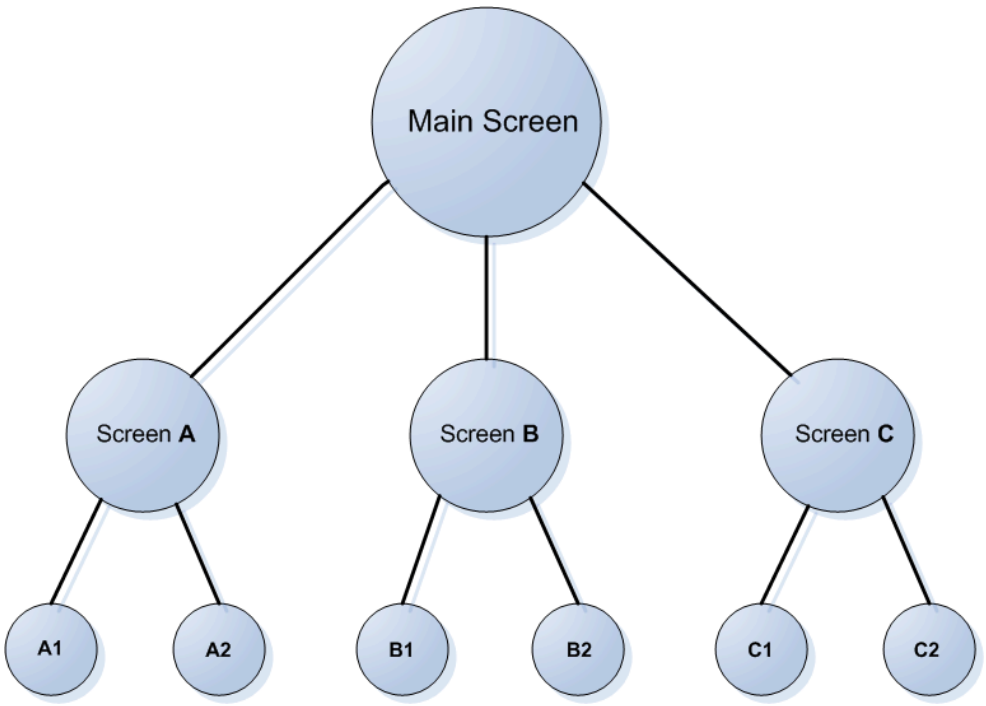
2.6. Screen navigation

Overview

Screen navigation in ScreenMaker follows a tree structure.

Consider the following example,

- To open screen **A1,** you first have to open **Screen A**
- To navigate from screen **A1** to screen **B1**, you first have to close screen **A1** and then **Screen A** and navigate from **Main Screen** through **Screen B** to screen **B1**.
- Similarly, to navigate from screen **B1** to screen **C1**, you first have to close screen **B1** and **Screen B** and then navigate from **Main Screen** through **Screen C** to screen **C1**.



en0900000645

2.6. Screen navigation

3 Tutorial

3.1. Overview

About this chapter

This chapter is designed as a tutorial to take you through the steps involved in designing a FlexArc Operator Panel, a simple arc-welding cell, whose robots perform three different jobs.

The operator panel will display the following graphic elements:

- Controller Status, including the controller mode and the RAPID execution status
- Part Status, including the number of produced parts, the average cycle time per part, and a reset button
- Robot jobs and locations, including Produce (for welding parts), Bull's Eye (for calibration) and Service (for servicing the welding gun)
- Start and Stop buttons

Workflow

1. Create a station with a system, define the project and create application variables. For more information, see *Prerequistes for designing FlexArc Operator Panel on page 46*.
2. Define the controls/graphic elements on the screen. For more information, see *Designing the screen on page 48*.
3. Build and Deploy the project. For more information, see *Building and deploying the project on page 54*.

The following table shows the workflow involved in designing a FlexArc Operator panel:

Prerequistes for designing FlexArc Operator Panel on page 46	For creating a station with a system, defining your project, and creating your application variables.
Designing the screen on page 48	For defining the graphics of your screen.
Building and deploying the project on page 54	For building and deploying the GUI.

3.2. Prerequistes for designing FlexArc Operator Panel

Procedure

Use this procedure to create a station with a system, define the project, and create the application variables involved in designing a FlexArc Operator Panel:

1. Create a system for the FlexArc Operator Panel.

   For more information about creating a system, see *Creating a system from layout*, in *Operating manual - RobotStudio*.

2. Create a station in RobotStudio with the system created in the previous step.

   For more information about creating a station, see *New Station*, in *Operating manual - RobotStudio*.

3. In RobotStudio, click **ScreenMaker** to launch the ScreenMaker application.

   For information about ScreenMaker GUI, see *Development environment on page 13*.

4. From the **ScreenMaker** ribbon, create a new project and name it as *FlexArcGUI*, and save it in the default location.

   The new tab **MainScreen** is added to the Design Surface.

   For more information about creating and managing projects, see *Managing ScreenMaker projects on page 26*.

5. In the context menu, build and deploy the project by connecting to the controller.

   The result is displayed in the output window.

   For more information about building and deploying projects, see *Managing screens on page 29*.

6. In the context menu, create application variables and configure them with the data in the following table:

Name	Type	Value
MyResetValue	num	0
JobProduce	num	1
JobIdle	num	0
JobBulls	num	2
JobService	num	3

   For more information about application variables, see *Managing application variables on page 34*.

7. Create **MainModule.mod** for the task T_ROB1.

8. Create RAPID variables with the following data:

Name	Type		Value
partsReady	num	PERS	16
cycleTime	num	PERS	5.01645
JobIdle	num	CONST	0
JobProduce	num	CONST	1
JobBulls	num	CONST	2

Continues on next page

Continued

Name	Type		Value
JobService	num	CONST	3

3.3. Designing the screen

Overview

A major effort in the GUI project development is designing screens. The Form designer in the ScreenMaker allows you to drag controls from the toolbox to the design surface. Using the Properties window, you can resize, position, label, color, and configure the controls.

For information on the form designer, see *Form designer on page 35*.

For information on the toolbox, see *ToolBox on page 15*.

For information on the property window, see *Properties window on page 17*.

Designing FlexArc Operator Panel screen

The FlexArc Operator Panel screen is designed using the following controls in the ToolBox:

- **GroupBox**. For more information, see *GroupBox on page 48*.
- **Status Icons**. For more information, see *Status icon on page 49*.
- **Label** and **Number editor.** For more information, see *Label and Number Editor on page 49*.
- **Picture box**. For more information, see *PictureBox on page 50*.
- **Button**. For more information, see *Button on page 51*.

GroupBox

1. Drag a **GroupBox** control from the **General** category on to the design surface and set the following values in the **Properties** window.

Property	Value
Location	14,45
Size	150,100
Title	Controller Status
BackColor	LightGray

2. Drag another **GroupBox** control from the **General** category on to the design surface and set the following values in the **Properties** window.

Property	Value
Location	14,170
Size	150,204
Title	Part Status

Continues on next page

Status icon

1. Drag a **ControllerModeStatus** control from the **Controller Data** category on to the *Controller Status* groupbox and set the following values in the **Properties** window:

Property	Value
Location	19,40
Size	44,44

2. Drag a **RapidExecutionStatus** control from the **ControllerData** category on to the *Controller Status* groupbox and set the following values in the **Properties** window:

Property	Value
Location	80,40
Size	44,44

Label and Number Editor

1. Drag a **Label** control from the **General** category on to the *Part Status* groupbox and set the following values in the **Properties** window:

Property	Value
Location	16,30
Size	131,20
Title	Parts Produced
BackColor	LightGray
Font	TpsFont10

2. Drag a **NumEditor** from the **ControllerData** category on to the *Parts Status*: groupbox and set the following values in the **Properties** window:

Property	Value
Location	16,56
Size	116,23
Value	Bind to a Controller object For example, RAPID **num** datatype.

Continues on next page

Continued

3. Drag another **Label** control from the **General** category on to the *Part Status* groupbox and set the following values in the **Properties** window:

Property	Value
Location	16,89
Size	131,20
Text	Cycle time/part
BackColor	LightGray
Font	TpsFont10

4. Drag another **NumEditor** control from the **General** category on to the *Part Status* groupbox and set the following values in the **Properties** window:

Property	Value
Location	16,115
Size	116,23
Value	Bind to a Controller object For example, RAPID **num** datatype.

PictureBox

NOTE!

It is not mandatory to use only the images (.gif) mentioned in the tabel below (for example, RobotAtBullseye.GIF, FlexArcCell.GIF...). You can use any image (.gif) of your choice. The images (.gif) mentioned below are only examples.

1. Drag a **PictureBox** control from the **General** category on to the design surface and set the following values in the **Properties** window:

Property	Value
Location	177,28
Size	284,359
SizeMode	StretchImage
Image	FlexArcCell.GIF

2. Drag a second **PictureBox** control from the **General** category on to the design surface and set the following values in the **Properties** window.

Property	Value
Location	369,31
Size	48,48
SizeMode	StretchImage
Image	RobotAtBullseye.GIF
Visible	Link to DI_RobotAtBullseye

Continues on next page

Continued

3. Drag a third **PictureBox** control from the **General** category on to the design surface and set the following values in the **Properties** window:

Property	Value
Location	237,31
Size	48,48
SizeMode	StretchImage
Image	RobotAtHome.GIF
Visible	Link to DI_RobotAtHome

4. Drag a fourth **PictureBox** control from the **General** category on to the design surface and set the following values in the **Properties** window:

Property	Value
Location	369,31
Size	48,48
SizeMode	StretchImage
Image	RobotAtService.GIF
Visible	Link to DI_RobotAtService

Button

1. Drag a **Button** control from the **General** category on to the *Part Status* group box. In the properties window, set the following values:

Property	Value
Location	33,154
Size	85,34
Text	Reset

Define the following for the **Reset** button in the *Part Status* group:

1. Double-click the button or click the *Smart tag* and select **Define Actions when clicked** to open the **Events Panel** dialog box.
2. Click **Add Action** and point to **Rapid Data**, and then select **Write Rapid Data.** The Action Parameters dialog box appears.
3. In the Action Parameters dialog box, define the following values and click **OK**.

Rapid Data to Write	Value to Write
T_ROB1.MainModule.partsReady	MyResetValue.Value
T_ROB1.MainModule.cycleTime	MyResetValue.Value

2. Drag a second **Button** control from the **General** category on to the design surface and set the following values in **Properties** window:

Property	Value
Location	486,66
Size	116,105

Continues on next page

Continued

Property	Value
Text	Start
Font	TpsFont20b
BackColor	LimeGreen
Enabled	Link to DI_RobotAtHome

Perform the following actions on the **Start** button:

1. Double-click the button or click the *Smart tag* and select **Define Actions when clicked** to open the **Events Panel** dialog box.
2. Click **Add Action** and point to **Rapid Data**, and then select **Write Rapid Data.** The Action Parameters dialog box appears.
3. In the Action Parameters dialog box, define the following values and click **OK**.

Rapid Data to Write	Value to Write
T_ROB1.MainModule.JobProduce	JobProduce

3. Drag a third **Button** control from the **General** category on to the design surface and set the following values in the **Properties** window:

Property	Value
Location	486,226
Size	116,105
Text	Stop
Font	TpsFont20b
BackColor	LimeGreen
Enabled	Link to DI_PRODUCE

Perform the following actions on the **Stop** button:

1. Double-click the button or click the *Smart tag* and select **Define Actions when clicked** to open the **Events Panel** dialog box.
2. Click **Add Action** and point to **Rapid Data**, and then select **Write Rapid Data.** The Action Parameters dialog box appears.
3. In the Action Parameters dialog box, define the following values and click **OK**.

Rapid Data to Write	Value to Write
T_ROB1.MainModule.JobIdle	JobIdle

4. Drag a a fourth **Button** control from the **General** category on to the design surface and set the following values in the **Properties** window:

Property	Value
Location	274,246
Size	111,47
Text	BullsEye
Font	TpsFont14b
BackColor	LimeGreen

Continues on next page

Property	Value
Enabled	Link to DI_RobotAtHome

Perform the following actions on the **BullsEye** button:

1. Double-click the button or click the *Smart tag* and select **Define Actions when clicked** to open the **Events Panel** dialog box.
2. Click **Add Action** and point to **Rapid Data**, and then select **Write Rapid Data.** The Action Parameters dialog box appears.
3. In the Action Parameters dialog box, define the following values and click **OK**.

Rapid Data to Write	Value to Write
T_ROB1.MainModule.JobBulls	JobBulls

5. Drag a fifth **Button** control from the **General** category on to the design surface and set the following values in the **Properties** window:

Property	Value
Location	274,324
Size	111,47
Text	Service
Font	TpsFont14b
BackColor	LimeGreen
Enabled	Link to DI_RobotAtHome

Perform the following actions on the **Service** button:

1. Double-click the button or click the *Smart tag* and select **Define Actions when clicked** to open the **Events Panel** dialog box.
2. Click **Add Action** and point to **Rapid Data**, and then select **Write Rapid Data.** The Action Parameters dialog box appears.
3. In the Action Parameters dialog box, define the following values and click **OK**.

Rapid Data to Write	Value to Write
T_ROB1.MainModule.JobService	JobService

3.4. Building and deploying the project

Procedure

1. From the **ScreenMaker** ribbon, click **Build**.

   For more information on building the project, see *Building a project on page 33*.
2. From the **ScreenMaker** ribbon, click **Deploy**.

   For more information on deploying the project, see *Deploying to controller on page 33*.
3. In RobotStudio, press **Ctrl+F5** to launch the Virtual Flexpendant and click FlexArc Operator Panel to open the GUI.

   NOTE!

   Ensure that you start the RAPID execution and switch the controller into Auto mode.

4 Frequently asked questions

4.1. Frequently asked questions

How to deploy manually to a Virtual Controller

If for any reason you wish to manually by-pass the **Deploy** button in RobotStudio and the virtual controller, the following information describes what files are to be moved.

Actions

Location of output files

The files that contain the FlexPendant application from ScreenMaker are found (for example) in the **bin** directory under the **My ScreenMaker Projects** located in the **My documents** directory of the user.

For example, **My Documents\My ScreenMaker Projects\SCM_Example\bin** where SCM_Example is the example ScreenMaker project.

The files in the **bin** directory are to be copied to a location where the Virtual FlexPendant can read them during the start of the FlexPendant.

Location where the Virtual FlexPendant reads the files

The recommended location for manually copying the ScreenMaker output files is the location of the virtual controller system.

If the system is created manually from **System Builder**, it is located in the **My Documents** directory.

For example, **My Documents\IRB4400_60_SCM_Example\HOME** where IRB4400_60_SCM_Example is the example controller system.

If the system is created by a Pack-and-Go and then restored, it is located in the **RobotStudio\Systems** folder.

For example,
MyDocuments\RobotStudio\Systems\IRB4400_60_SCM_Example\HOME where IRB4400_60_SCM_Example is the example controller system.

Copy files

Copy the files from the ScreenMaker output to the **Home** directory of the virtual controller system.

Restart the Virtual FlexPendant and the new application will be loaded.

Continues on next page

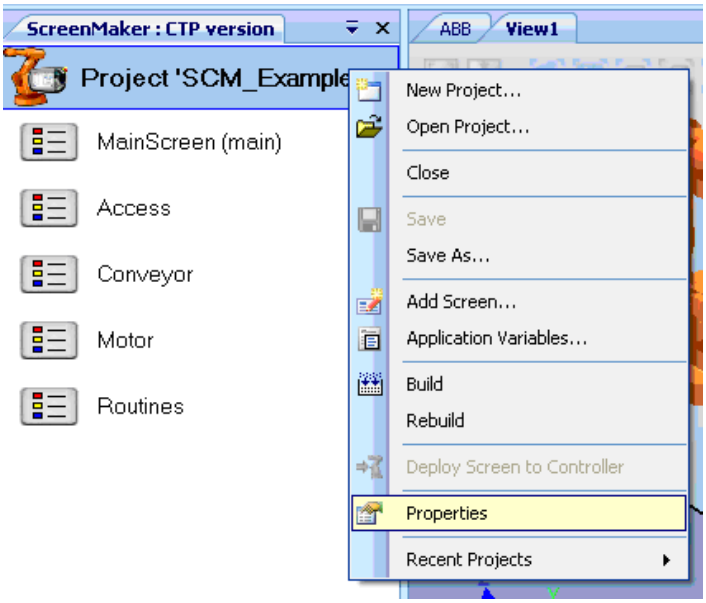
Continued

How to change the application name and image on the FlexPendant

When a ScreenMaker project is created the first time, the caption on the FlexPendant and image is set by default. You will want to change this.

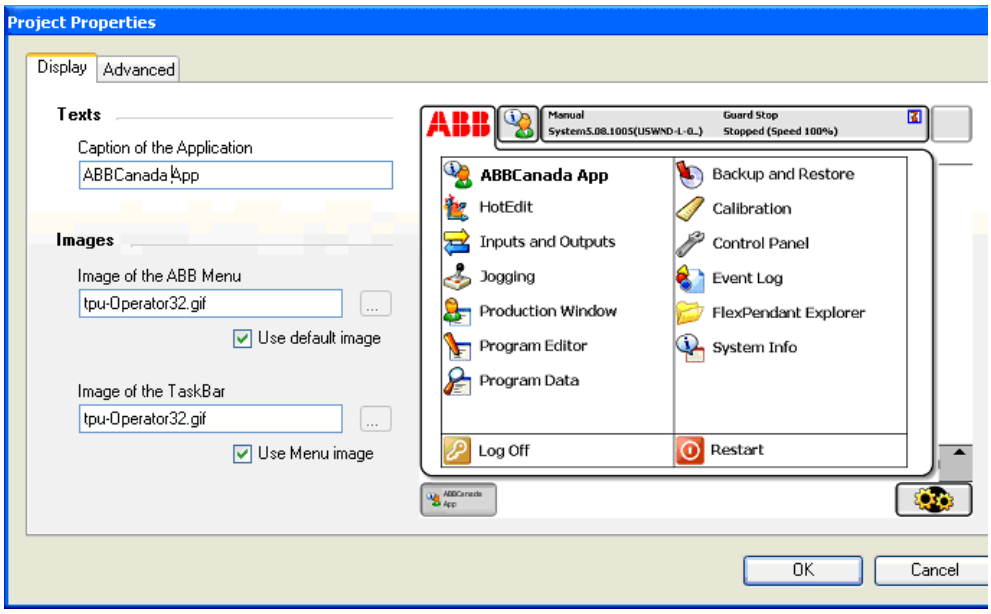
Actions

Right-click **Project** and select **Properties**.



en0900000666

The **Project Properties** dialog box appears showing how the project appears on the FlexPendant.



en0900000667

Continues on next page

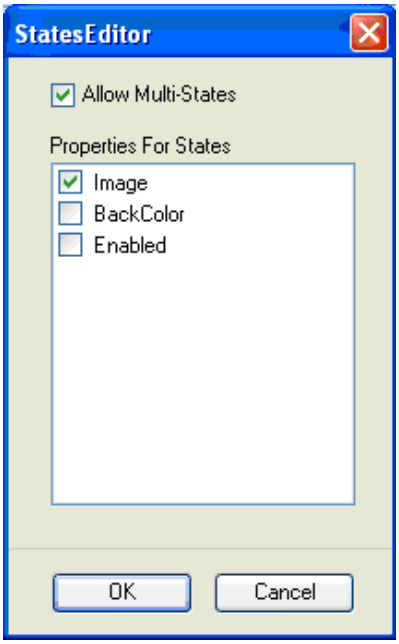
Picture object and changing images due to I/O

The typical user objective is to have an image that changes when an I/O signal changes, this is common for a digital input to affect the state on the FlexPendant.

Actions

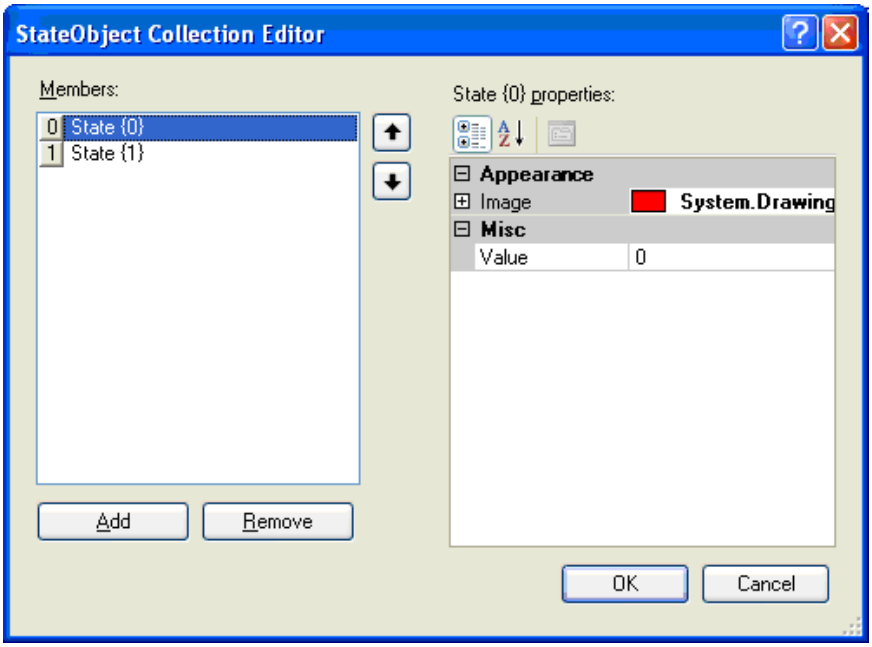
This is accomplished by adding an image and allowing the image to have multiple states.

Set **AllowMultipleStat**e to **TRUE** and set the **Image** state.



en0900000668

Create two states and add images for each state:



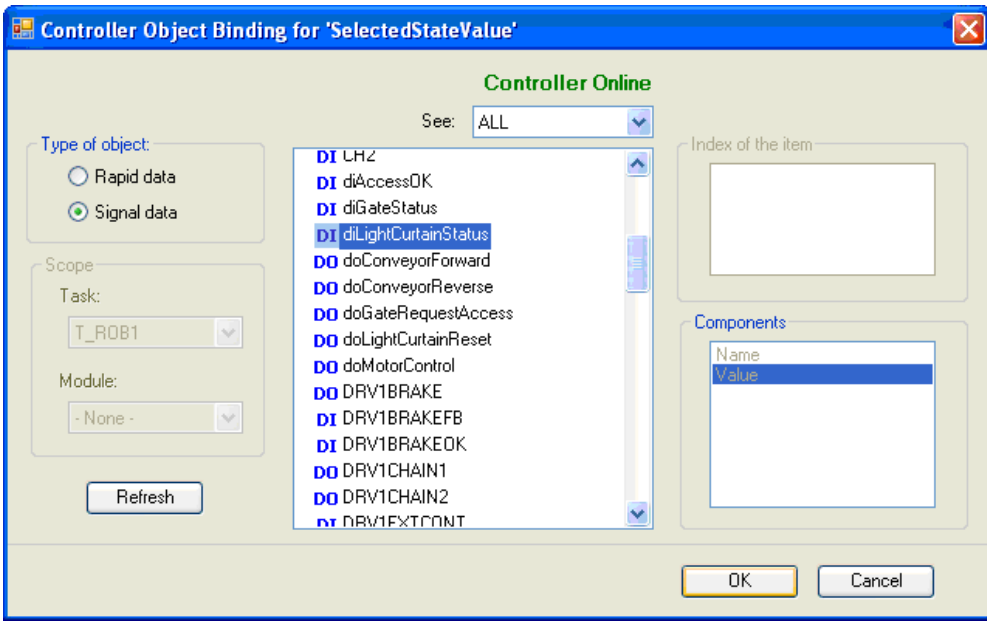
en0900000669

Continues on next page

Continued

The Value property is extremely important. If binding to a digital input then there are two states for the input, 0 and 1. Set the **Value** property to the value of the bound variable. 0 and 1 for digital input. It is also possible to bind to RAPID variables and have multiple states and values for the values in the RAPID variable.

Set the **SelectedStateValue** property to bind to a controller object:



en0900000670

Using CommandBar and the menu items

Using CommandBars allow buttons to appear at the bottom of the screen in a controlled and organized order.

en0900000671

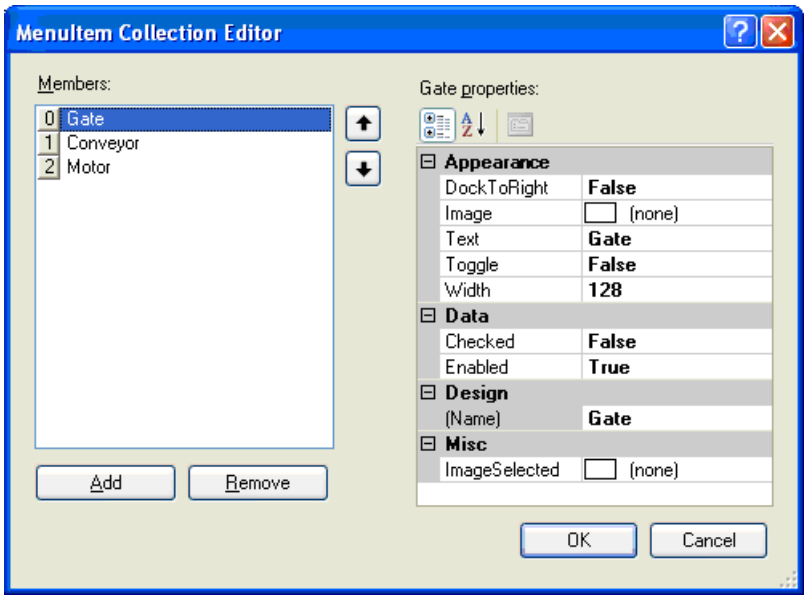
In the preceding graphic, the CommandBar has three menu items **Gate**, **Conveyor**, and **Motor**. The objective is to have events trigger when these are clicked.

Continues on next page

Continued

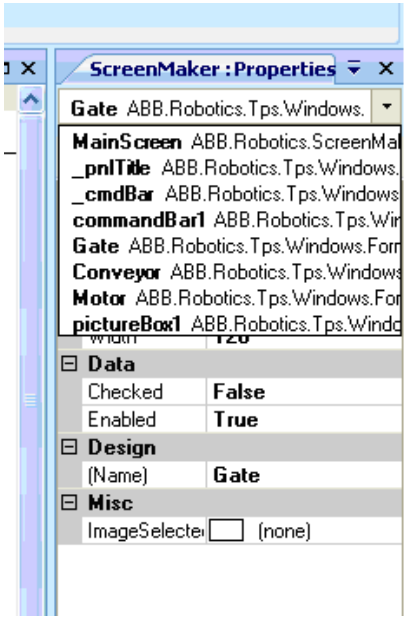
Actions

First create the CommandBar and add the menu items (either by editing the menu items property or by clicking the small arrow on the upper right of the active CommandBar) and add the menu items.



en0900000672

To add events to the menu items, go to the **Properties** dialog and select the menu item from the drop down list.

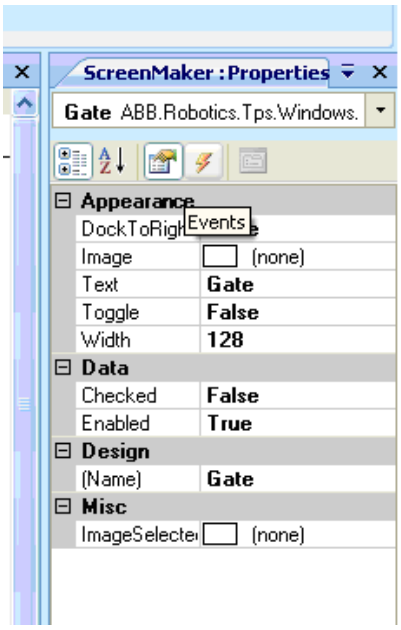


en0900000673

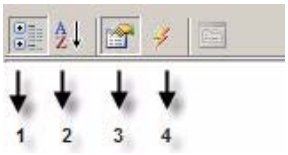
Continues on next page

Continued

Select **Gate** menu item. The following dialog appears.



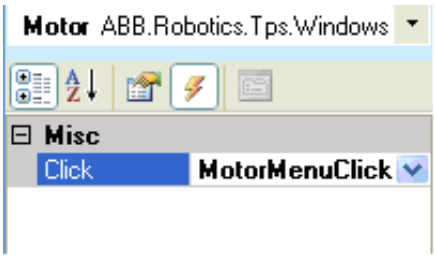
en0900000674



en0900000409

1. In the preceding graphic, select **4** to add an event when the menu item is clicked.
2. Click the drop-down list to select the event. The events are added automatically

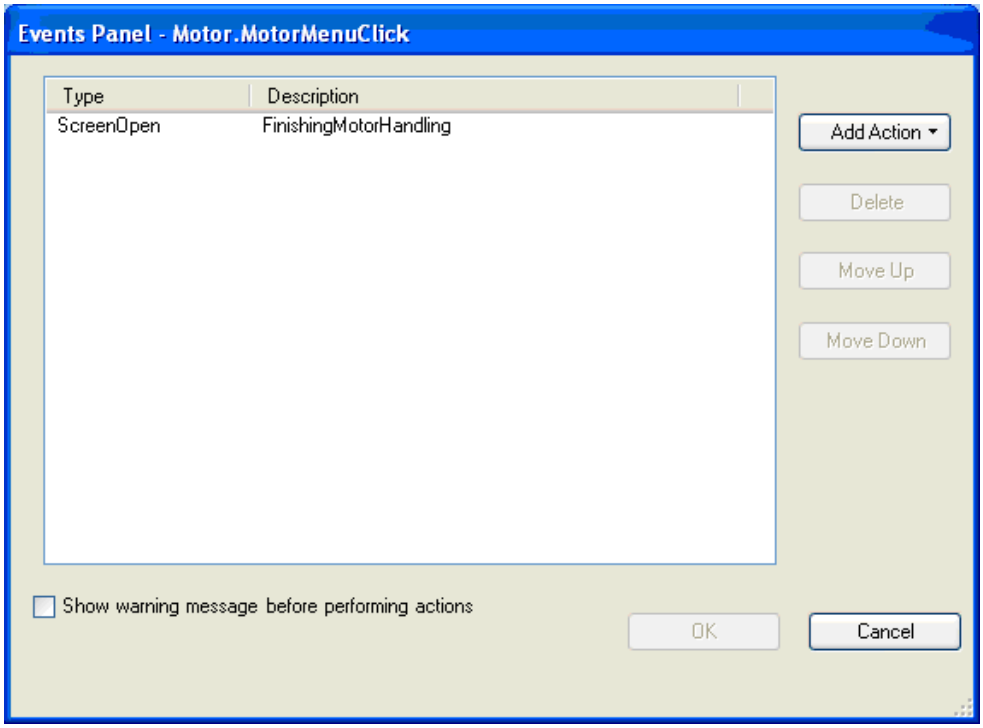
For more information on various ways of opening the **Events Panel** dialog box, see *Setup Events on page 35*.



en0900000675

Continues on next page

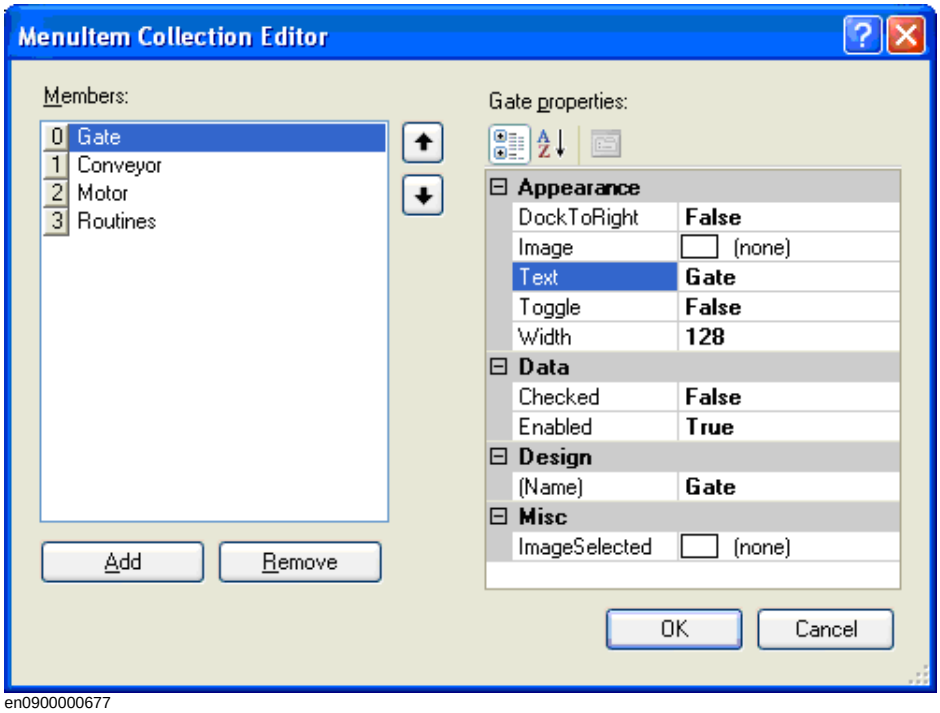
Continued



en0900000676

CommandBar menu items do not appear

When adding CommandBar menu items, ensure that the **Text** property is filled, if not nothing will appear on the CommandBar.



en0900000677

Continues on next page

4.1. Frequently asked questions

Continued

How to get radio buttons to show state when entering

The objective is to have two radio buttons that controls one digital output. When the screen is loaded, the buttons should show the current state of the output.

Actions

Create a group or a panel and place the two radio buttons on the group or panel.

For button1, set the property default value to **True** and bind the property to the value of the controller digital output signal.

For button1, do not do any changes.

When the screen is loaded, the state of the two radio buttons is established correctly.

Binding to arrays and using the index correctly

While a RAPID index starts with 1 (1 selects the first element), the ComboBox index starts with 0 (0 selects the first index). You should be aware of this when using ComboBoxes.

How to reboot the FlexPendant after re-deploying

If a real controller is used, you can reboot your FlexPendant by holding the FlexPendant joystick and performing the following sequence:

Move the joystick three times to the right, once to the left and once down.