The information in this manual is subject to change without notice and should not be construed as a commitment by ABB. ABB assumes no responsibility for any errors that may appear in this manual.

Except as may be expressly stated anywhere in this manual, nothing herein shall be construed as any kind of guarantee or warranty by ABB for losses, damages to persons or property, fitness for a specific purpose or the like.

In no event shall ABB be liable for incidental or consequential damages arising from use of this manual and products described herein.

This manual and parts thereof must not be reproduced or copied without ABB's written permission, and contents thereof must not be imparted to a third party nor be used for any unauthorized purpose. Contravention will be prosecuted.

Additional copies of this manual may be obtained from ABB at its then current charge.
## Table of Contents

Manual overview .................................................. 5

### 1 Introduction

1.1 Introduction to ScreenMaker .................................. 7
1.2 Installing ScreenMaker ....................................... 10
1.3 Terminology .................................................. 12
1.4 Development environment ................................... 13

### 2 Managing ScreenMaker projects

2.1 Overview .................................................... 25
2.2 Managing ScreenMaker projects ............................. 26
2.3 Application variables ......................................... 34
2.4 Form designer ............................................... 35
2.5 Data binding ............................................... 38
2.6 Screen navigation ........................................... 43

### 3 Tutorial

3.1 Overview .................................................... 45
3.2 Prerequisites for designing FlexArc Operator Panel .... 46
3.3 Designing the screen ......................................... 48
3.4 Building and deploying the project ......................... 54

### 4 Frequently asked questions

4.1 Frequently asked questions ................................. 55

Index ............................................................... 63
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:

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

References

<table>
<thead>
<tr>
<th>Reference</th>
<th>Document ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>Operating manual - RobotStudio</td>
<td>3HAC032104-001</td>
</tr>
<tr>
<td>Operating manual - IRC5 with FlexPendant</td>
<td>3HAC16590-1</td>
</tr>
</tbody>
</table>

Revisions

<table>
<thead>
<tr>
<th>Revision</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-</td>
<td>First edition. Released with RobotWare 5.12.02.</td>
</tr>
</tbody>
</table>
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

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
1 Introduction

1.1. Introduction to ScreenMaker

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

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
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:

<table>
<thead>
<tr>
<th>Software requirements</th>
<th>• RobotStudio 5.12 or later with Premium license activated</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>• Microsoft .NET Compact Framework 2.0</td>
</tr>
<tr>
<td></td>
<td>• Robot Application Builder (RAB) 5.12 or later</td>
</tr>
</tbody>
</table>

| 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.

<table>
<thead>
<tr>
<th>Before ...</th>
<th>you must ...</th>
</tr>
</thead>
<tbody>
<tr>
<td>installing ScreenMaker</td>
<td>ensure that the following applications are installed:</td>
</tr>
<tr>
<td></td>
<td>• RobotStudio with Premium license</td>
</tr>
</tbody>
</table>
| 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.

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**

![Diagram showing LAN, IRC5, IRC5, IRC5, ScreenMaker connections]

**PC Interface option is not needed**

![Diagram showing IRC5, Service Port, ScreenMaker connections]
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

<table>
<thead>
<tr>
<th>Term</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IRC5</td>
<td>ABB’s robot controller.</td>
</tr>
<tr>
<td>FlexPendant</td>
<td>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.</td>
</tr>
<tr>
<td>Robot Application Builder</td>
<td>ABB software tool, which enables the development of custom operator interfaces for IRC5. Often referred to as RAB.</td>
</tr>
<tr>
<td>Microsoft .NET Compact Framework</td>
<td>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.</td>
</tr>
<tr>
<td>C++,VisualBasic and C#</td>
<td>Programming languages</td>
</tr>
</tbody>
</table>

Acronym

<table>
<thead>
<tr>
<th>Acronym</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GUI</td>
<td>Graphical User Interface</td>
</tr>
<tr>
<td>OS</td>
<td>Operating System</td>
</tr>
<tr>
<td>RAB</td>
<td>Robot Application Builder</td>
</tr>
<tr>
<td>I/Os</td>
<td>Input /Output signals</td>
</tr>
<tr>
<td>WYSIWYG</td>
<td>What You See Is What You Get</td>
</tr>
</tbody>
</table>
### 1.4. Development environment

#### Overview

This section presents an overview of the ScreenMaker development environment.

<table>
<thead>
<tr>
<th>Parts</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 Ribbon</td>
<td>Displays group of icons organized in a logical sequence of functions. See <a href="#">Ribbon on page 14</a>.</td>
</tr>
<tr>
<td>2 Project explorer</td>
<td>Shows the active screen project and lists the screens that are defined in the project. For more information, see <a href="#">Managing ScreenMaker projects on page 20</a>.</td>
</tr>
<tr>
<td>3 Design surface</td>
<td>Layout to design the screen with the available controls. For more information, see <a href="#">Form designer on page 35</a>.</td>
</tr>
<tr>
<td>4 Output window</td>
<td>Displays information about the events that occur during ScreenMaker development.</td>
</tr>
<tr>
<td>5 ToolBox</td>
<td>Displays a list of available controls. For more information, see <a href="#">ToolBox on page 15</a>.</td>
</tr>
<tr>
<td>6 Properties window</td>
<td>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 <a href="#">Properties window on page 17</a>.</td>
</tr>
<tr>
<td>7 Arrange</td>
<td>Displays icons for resizing and positioning controls on the design surface. See <a href="#">Arrange on page 14</a>.</td>
</tr>
</tbody>
</table>
Ribbon

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

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

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

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

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

*Continues on next page*
### 1 Introduction

#### 1.4. Development environment

*Continued*

<table>
<thead>
<tr>
<th>Control</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>LED</td>
<td>Displays a two states value, like a Digital Signal.</td>
</tr>
<tr>
<td>ListBox</td>
<td>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.</td>
</tr>
<tr>
<td>NumEditor</td>
<td>Represents a text box control that can be used to edit a number. When the user clicks it, a Numpad is opened.</td>
</tr>
<tr>
<td>NumericUpDown</td>
<td>Represents a spin box that displays numeric values.</td>
</tr>
<tr>
<td>Panel</td>
<td>Used to group collection of controls.</td>
</tr>
<tr>
<td>PictureBox</td>
<td>Represents a picture box control that displays images.</td>
</tr>
<tr>
<td>RadioButton</td>
<td>Allows to select only one of a predefined set of options.</td>
</tr>
<tr>
<td>RapidExecutionStatus</td>
<td>Displays the execution status of the Controller Rapid Domain (Running - Auto)</td>
</tr>
<tr>
<td>RunRoutineButton</td>
<td>Represents a Windows button control that calls a RapidRoutine when clicked</td>
</tr>
<tr>
<td>Switch</td>
<td>Displays and lets change a two states value, like a Digital Output Signal.</td>
</tr>
<tr>
<td>TabControl</td>
<td>Manages a set of tab pages.</td>
</tr>
<tr>
<td>TpsLabel</td>
<td>Very commonly used widget that displays text, a label is usually static, that is, it has no interactivity. A label generally identifies a nearby text box or other graphic component.</td>
</tr>
</tbody>
</table>

**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.
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.

<table>
<thead>
<tr>
<th>Element</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Graphical component name panel</td>
</tr>
<tr>
<td>2</td>
<td>Properties window toolbar</td>
</tr>
<tr>
<td></td>
<td><img src="en0900000408" alt="Properties window toolbar" /></td>
</tr>
<tr>
<td></td>
<td>1. Organizes table panel in categories</td>
</tr>
<tr>
<td></td>
<td>2. Organizes table panel alphabetically</td>
</tr>
<tr>
<td></td>
<td>3. Displays Properties in table panel</td>
</tr>
<tr>
<td></td>
<td>4. Displays Events in table panel</td>
</tr>
<tr>
<td>3</td>
<td>Table panel</td>
</tr>
<tr>
<td>4</td>
<td>Information panel</td>
</tr>
</tbody>
</table>
1 Introduction

1.4. Development environment

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.

2. By selecting the predefined values from the list. For example, BackColor, Font etc.

3. By entering the values in the dialog box. For example, Enabled, States, BaseValue etc.

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:

<table>
<thead>
<tr>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 Drag an <strong>ActionTrigger</strong> control from the <strong>ToolBox</strong> on to the design surface.</td>
</tr>
</tbody>
</table>

Continues on next page
### 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*.

#### 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.

---

**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.
1 Introduction

1.4. Development environment

Continued

TpsLabel

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

Use this procedure to add a TpsLabel control:

<table>
<thead>
<tr>
<th>Step</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Drag a TpsLabel control from the Toolbox on to the design surface.</td>
</tr>
</tbody>
</table>
| 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. |

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:

<table>
<thead>
<tr>
<th>Step</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Drag a Panel control from the Toolbox on to the design surface.</td>
</tr>
<tr>
<td>2</td>
<td>You can add a group of controls to a panel.</td>
</tr>
</tbody>
</table>

Continues on next page
1.4. Development environment

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:

<table>
<thead>
<tr>
<th>Step</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Drag a ControllerModeStatus control from the ToolBox on to the design surface.</td>
</tr>
<tr>
<td>2</td>
<td>You can set the values, setup events, configure data binding values, and set the application variables for a ControllerModeStatus control.</td>
</tr>
<tr>
<td></td>
<td>• To set the values of a property, see Properties window on page 17.</td>
</tr>
<tr>
<td></td>
<td>• To set up the events, see Setup Events on page 35.</td>
</tr>
<tr>
<td></td>
<td>• To configure the data binding values, see Configuring data binding on page 38.</td>
</tr>
<tr>
<td></td>
<td>• To set the application variables, see Managing application variables on page 34.</td>
</tr>
<tr>
<td>3</td>
<td>You can select the image to be displayed when the controller is in Auto mode and in Manual mode.</td>
</tr>
<tr>
<td></td>
<td>• Click AutoImage in the Properties window and browse to select the image to be displayed in Auto mode.</td>
</tr>
<tr>
<td></td>
<td>• Click ManualImage in the Properties window and browse to select the image to be displayed in Manual mode.</td>
</tr>
</tbody>
</table>
RapidExecutionStatus displays the execution status of the Controller Rapid Domain (Running - Auto). This control is used

Use this procedure to add a RapidExecutionStatus control:

<table>
<thead>
<tr>
<th>Step</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Drag a <strong>RapidExecutionStatus</strong> control from the <strong>ToolBox</strong> on to the design surface.</td>
</tr>
</tbody>
</table>
| 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. |

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

Use this procedure to add a RunRoutineButton control:

<table>
<thead>
<tr>
<th>Step</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Drag a <strong>RunRoutineButton</strong> control from the <strong>ToolBox</strong> on to the design surface.</td>
</tr>
</tbody>
</table>

Continues on next page
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.

You can perform the following RunRoutineButton tasks from the SmartTag:
- Define Actions before calling Routine
- Select Routine to call
- Define Actions after calling Routine

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.
1 Introduction

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 Managing ScreenMaker projects

2.2. Managing ScreenMaker projects

Overview

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

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**.
2 Managing ScreenMaker projects

2.2. Managing ScreenMaker projects

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.

   ![Open Screen Project dialog box]

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.
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 **Save As** 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**.

Deleting a screen

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

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.

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.
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.

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
### 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.

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.
2 Managing ScreenMaker projects

2.2. Managing ScreenMaker projects

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.

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 Managing ScreenMaker projects

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.

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 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.
The following table lists the set of predefined actions:

| Screen       | • Open Screen  
|             | • Close Screen |
| Signals      | • Set a Digital Signal  
|             | • Invert a Digital Signal  
|             | • Pulse a Digital Signal  
|             | • Read a Signal  
|             | • Write a Signal  
|             | • Reset a Digital Signal  |
| RapidData    | • Read a Rapid Data  
|             | • Write a Rapid Data  |
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 versa. 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 convertible 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.

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
2 Managing ScreenMaker projects

2.5. Data 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

<table>
<thead>
<tr>
<th>Menu</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Remove actual binding</td>
<td>Removes the existing data binding</td>
</tr>
<tr>
<td>Bind to a Controller object</td>
<td>Select available data in the controller for binding.</td>
</tr>
<tr>
<td>Bind to an Application variable</td>
<td>Select available data in project temporary data store for binding.</td>
</tr>
</tbody>
</table>
2 Managing ScreenMaker projects

2.5. Data binding

Continued

<table>
<thead>
<tr>
<th>Control</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DataEditor</td>
<td>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.</td>
</tr>
<tr>
<td>ComboBox and ListBox</td>
<td>The default index value is -1. You can enter the appropriate index value but cannot bind to a controller object or an application variable.</td>
</tr>
</tbody>
</table>
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.

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.
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.

![Application Variables Bind Form](image)

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.
2 Managing ScreenMaker projects

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 Prerequisites 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:

<table>
<thead>
<tr>
<th>Prerequisites for designing FlexArc Operator Panel on page 46</th>
<th>For creating a station with a system, defining your project, and creating your application variables.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Designing the screen on page 48</td>
<td>For defining the graphics of your screen.</td>
</tr>
<tr>
<td>Building and deploying the project on page 54</td>
<td>For building and deploying the GUI.</td>
</tr>
</tbody>
</table>
3 Tutorial

3.2. Prerequisites for designing FlexArc Operator Panel

3.2. Prerequisites 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:

   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:

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

Continues on next page
### 3.2. Prerequisites for designing FlexArc Operator Panel

*Continued*

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>JobService</td>
<td>num</td>
<td>CONST 3</td>
</tr>
</tbody>
</table>
3 Tutorial

3.3. Designing the screen

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.
- **PictureBox**. 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.

<table>
<thead>
<tr>
<th>Property</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Location</td>
<td>14,45</td>
</tr>
<tr>
<td>Size</td>
<td>150,100</td>
</tr>
<tr>
<td>Title</td>
<td>Controller Status</td>
</tr>
<tr>
<td>BackColor</td>
<td>LightGray</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Property</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Location</td>
<td>14,170</td>
</tr>
<tr>
<td>Size</td>
<td>150,204</td>
</tr>
<tr>
<td>Title</td>
<td>Part Status</td>
</tr>
</tbody>
</table>

Continues on next page
3 Tutorial

3.3. Designing the screen

Continued

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:

<table>
<thead>
<tr>
<th>Property</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Location</td>
<td>19,40</td>
</tr>
<tr>
<td>Size</td>
<td>44,44</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Property</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Location</td>
<td>80,40</td>
</tr>
<tr>
<td>Size</td>
<td>44,44</td>
</tr>
</tbody>
</table>

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:

<table>
<thead>
<tr>
<th>Property</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Location</td>
<td>16,30</td>
</tr>
<tr>
<td>Size</td>
<td>131,20</td>
</tr>
<tr>
<td>Title</td>
<td>Parts Produced</td>
</tr>
<tr>
<td>BackColor</td>
<td>LightGray</td>
</tr>
<tr>
<td>Font</td>
<td>TpsFont10</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Property</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Location</td>
<td>16,56</td>
</tr>
<tr>
<td>Size</td>
<td>116,23</td>
</tr>
<tr>
<td>Value</td>
<td>Bind to a Controller object For example, RAPID num datatype.</td>
</tr>
</tbody>
</table>
3 Tutorial

3.3. Designing the screen

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:

<table>
<thead>
<tr>
<th>Property</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Location</td>
<td>16,89</td>
</tr>
<tr>
<td>Size</td>
<td>131,20</td>
</tr>
<tr>
<td>Text</td>
<td>Cycle time/part</td>
</tr>
<tr>
<td>BackColor</td>
<td>LightGray</td>
</tr>
<tr>
<td>Font</td>
<td>TpsFont10</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Property</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Location</td>
<td>16,115</td>
</tr>
<tr>
<td>Size</td>
<td>116,23</td>
</tr>
<tr>
<td>Value</td>
<td>Bind to a Controller object</td>
</tr>
<tr>
<td></td>
<td>For example, RAPID num datatype.</td>
</tr>
</tbody>
</table>

NOTE!

It is not mandatory to use only the images (.gif) mentioned in the table 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:

<table>
<thead>
<tr>
<th>Property</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Location</td>
<td>177,28</td>
</tr>
<tr>
<td>Size</td>
<td>284,359</td>
</tr>
<tr>
<td>SizeMode</td>
<td>StretchImage</td>
</tr>
<tr>
<td>Image</td>
<td>FlexArcCell.GIF</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Property</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Location</td>
<td>369,31</td>
</tr>
<tr>
<td>Size</td>
<td>48,48</td>
</tr>
<tr>
<td>SizeMode</td>
<td>StretchImage</td>
</tr>
<tr>
<td>Image</td>
<td>RobotAtBullseye.GIF</td>
</tr>
<tr>
<td>Visible</td>
<td>Link to DI_RobotAtBullseye</td>
</tr>
</tbody>
</table>
3 Tutorial

3.3. Designing the screen

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:

<table>
<thead>
<tr>
<th>Property</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Location</td>
<td>237,31</td>
</tr>
<tr>
<td>Size</td>
<td>48,48</td>
</tr>
<tr>
<td>SizeMode</td>
<td>StretchImage</td>
</tr>
<tr>
<td>Image</td>
<td>RobotAtHome.GIF</td>
</tr>
<tr>
<td>Visible</td>
<td>Link to DI_RobotAtHome</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Property</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Location</td>
<td>369,31</td>
</tr>
<tr>
<td>Size</td>
<td>48,48</td>
</tr>
<tr>
<td>SizeMode</td>
<td>StretchImage</td>
</tr>
<tr>
<td>Image</td>
<td>RobotAtService.GIF</td>
</tr>
<tr>
<td>Visible</td>
<td>Link to DI_RobotAtService</td>
</tr>
</tbody>
</table>

**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:

<table>
<thead>
<tr>
<th>Property</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Location</td>
<td>33,154</td>
</tr>
<tr>
<td>Size</td>
<td>85,34</td>
</tr>
<tr>
<td>Text</td>
<td>Reset</td>
</tr>
</tbody>
</table>

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**.

<table>
<thead>
<tr>
<th>Rapid Data to Write</th>
<th>Value to Write</th>
</tr>
</thead>
<tbody>
<tr>
<td>T_ROB1.MainModule.partsReady</td>
<td>MyResetValue.Value</td>
</tr>
<tr>
<td>T_ROB1.MainModule.cycleTime</td>
<td>MyResetValue.Value</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Property</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Location</td>
<td>486,66</td>
</tr>
<tr>
<td>Size</td>
<td>116,105</td>
</tr>
</tbody>
</table>

Continues on next page
3 Tutorial

3.3. Designing the screen

Continued

<table>
<thead>
<tr>
<th>Property</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Text</td>
<td>Start</td>
</tr>
<tr>
<td>Font</td>
<td>TpsFont20b</td>
</tr>
<tr>
<td>BackColor</td>
<td>LimeGreen</td>
</tr>
<tr>
<td>Enabled</td>
<td>Link to DI_RobotAtHome</td>
</tr>
</tbody>
</table>

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.

<table>
<thead>
<tr>
<th>Rapid Data to Write</th>
<th>Value to Write</th>
</tr>
</thead>
<tbody>
<tr>
<td>T_ROB1.MainModule.JobProduce</td>
<td>JobProduce</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Property</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Location</td>
<td>486,226</td>
</tr>
<tr>
<td>Size</td>
<td>116,105</td>
</tr>
<tr>
<td>Text</td>
<td>Stop</td>
</tr>
<tr>
<td>Font</td>
<td>TpsFont20b</td>
</tr>
<tr>
<td>BackColor</td>
<td>LimeGreen</td>
</tr>
<tr>
<td>Enabled</td>
<td>Link to DI_PRODUCE</td>
</tr>
</tbody>
</table>

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.

<table>
<thead>
<tr>
<th>Rapid Data to Write</th>
<th>Value to Write</th>
</tr>
</thead>
<tbody>
<tr>
<td>T_ROB1.MainModule.JobIdle</td>
<td>JobIdle</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Property</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Location</td>
<td>274,246</td>
</tr>
<tr>
<td>Size</td>
<td>111,47</td>
</tr>
<tr>
<td>Text</td>
<td>BullsEye</td>
</tr>
<tr>
<td>Font</td>
<td>TpsFont14b</td>
</tr>
<tr>
<td>BackColor</td>
<td>LimeGreen</td>
</tr>
</tbody>
</table>

Continues on next page
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**.

<table>
<thead>
<tr>
<th>Rapid Data to Write</th>
<th>Value to Write</th>
</tr>
</thead>
<tbody>
<tr>
<td>T_ROB1.MainModule.JobBulls</td>
<td>JobBulls</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Property</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Location</td>
<td>274,324</td>
</tr>
<tr>
<td>Size</td>
<td>111,47</td>
</tr>
<tr>
<td>Text</td>
<td>Service</td>
</tr>
<tr>
<td>Font</td>
<td>TpsFont14b</td>
</tr>
<tr>
<td>BackColor</td>
<td>LimeGreen</td>
</tr>
<tr>
<td>Enabled</td>
<td>Link to DI_RobotAtHome</td>
</tr>
</tbody>
</table>

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**.

<table>
<thead>
<tr>
<th>Rapid Data to Write</th>
<th>Value to Write</th>
</tr>
</thead>
<tbody>
<tr>
<td>T_ROB1.MainModule.JobService</td>
<td>JobService</td>
</tr>
</tbody>
</table>
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
4 Frequently asked questions

4.1. Frequently asked questions

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**.

The **Project Properties** dialog box appears showing how the project appears on the FlexPendant.
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 AllowMultipleState to TRUE and set the Image state.

Create two states and add images for each state:
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:

Using CommandBar and the menu items

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

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.
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.

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

Continues on next page
Select **Gate** menu item. The following dialog appears.

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.
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.
4 Frequently asked questions

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.
Index

A
Application Variables 34
  create, delete, rename application variables 34
  FlexPendant memory 34
  RAPID variable 34

C
Configure data binding 38
  Using Binding menu 38
  Using Smart tag 38
Connecting a controller 32

D
Data Binding 38
  Application variable data binding 42
  Controller object data binding 41
Design surface 13
Development Environment 13

L
LED 16

M
Manage ScreenMaker Project 26
  Close project 29
  Create project 27
  Load project 28
  Save project 29
Manage Screens 29
  Create screen 29
  Delete screen 29
  Edit screen 30
  Rename screen 30
Managing ScreenMaker Projects
  Build project 33
  Modify project properties 31

P
Properties Window
  Event Help panel 17
  Graphical Component Name panel 17
  Properties window toolbar 17
  Table panel 17

S
Screen navigation 43
  switch 16

T
ToolBox
  ActionTrigger 15
  BarGraph 15
  CheckBox 15
  ComboBox 15
  CommandBar 15
  ControllerModeStatus 15
  DataEditor 15
  Graph 15
  GroupBox 15
  ListBox 16
  NumEditor 16
  NumericUpDown 16
  Panel 16
  PictureBox 16
  RapidExecutionStatus 16
  RunRoutineButton 16
  TabControl 16
  TpsLabel 16