Application manual
Production Screen

Application manual

Production Screen

RobotWare 6.03

Document ID: 3HAC050964-001

Revision: B

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.

Additional copies of this manual may be obtained from ABB.

The original language for this publication is English. Any other languages that are supplied have been translated from English.

Table of contents

This page is intentionally left blank

Overview of this manual

About this manual

This manual describes the RobotWare option Production Screen.

This manual contains instructions for

- installing the option
- adding and configuring apps and widgets
- modifying the xml-file

Usage

This manual should be used during

- installation and configuration.

Who should read this manual?

This manual is intended for:

- developers of FlexPendant applications
- installation personnel

References

Reference	Document ID
Operating manual - Emergency safety information	*3HAC027098-001*
Operating manual - IRC5 with FlexPendant	*3HAC050941-001*
Operating manual - RobotStudio	*3HAC032104-001*
Operating manual - Getting started, IRC5 and RobotStudio	*3HAC027097-001*
Operating manual - Trouble shooting IRC5	*3HAC020738-001*

Revisions

Revision	Description
-	Released with RobotWare 6.0.
A	Released with RobotWare 6.02. • Minor corrections.
A	Released with RobotWare 6.03. • The previous limitation of five app bars is removed. • Minor corrections.

Product documentation, IRC5

Categories for user documentation from ABB Robotics

The user documentation from ABB Robotics is divided into a number of categories. This listing is based on the type of information in the documents, regardless of whether the products are standard or optional.

All documents listed can be ordered from ABB on a DVD. The documents listed are valid for IRC5 robot systems.

Product manuals

Manipulators, controllers, DressPack/SpotPack, and most other hardware is delivered with a **Product manual** that generally contains:

- Safety information.
- Installation and commissioning (descriptions of mechanical installation or electrical connections).
- Maintenance (descriptions of all required preventive maintenance procedures including intervals and expected life time of parts).
- Repair (descriptions of all recommended repair procedures including spare parts).
- Calibration.
- Decommissioning.
- Reference information (safety standards, unit conversions, screw joints, lists of tools).
- Spare parts list with exploded views (or references to separate spare parts lists).
- Circuit diagrams (or references to circuit diagrams).

Technical reference manuals

The technical reference manuals describe reference information for robotics products.

- *Technical reference manual - Lubrication in gearboxes*: Description of types and volumes of lubrication for the manipulator gearboxes.
- *Technical reference manual - RAPID overview*: An overview of the RAPID programming language.
- *Technical reference manual - RAPID Instructions, Functions and Data types*: Description and syntax for all RAPID instructions, functions, and data types.
- *Technical reference manual - RAPID kernel*: A formal description of the RAPID programming language.
- *Technical reference manual - System parameters*: Description of system parameters and configuration workflows.

Application manuals

Specific applications (for example software or hardware options) are described in **Application manuals**. An application manual can describe one or several applications.

An application manual generally contains information about:

- The purpose of the application (what it does and when it is useful).
- What is included (for example cables, I/O boards, RAPID instructions, system parameters, DVD with PC software).
- How to install included or required hardware.
- How to use the application.
- Examples of how to use the application.

Operating manuals

The operating manuals describe hands-on handling of the products. The manuals are aimed at those having first-hand operational contact with the product, that is production cell operators, programmers, and trouble shooters.

The group of manuals includes (among others):

- *Operating manual - Emergency safety information*
- *Operating manual - General safety information*
- *Operating manual - Getting started, IRC5 and RobotStudio*
- *Operating manual - Introduction to RAPID*
- *Operating manual - IRC5 with FlexPendant*
- *Operating manual - RobotStudio*
- *Operating manual - Trouble shooting IRC5*, for the controller and manipulator.

This page is intentionally left blank

1 Introduction to Production Screen

1.1 System overview

What is Production Screen?

The *Production Screen* option is a framework for creating a customized GUI that can be used to present process data and status as well as execute FlexPendant applications.

The user interface works as a portal to launch FlexPendant SDK and ScreenMaker applications that are embedded into Production Screen.

The FlexPendant normally allows up to six applications with command bar buttons running simultaneously. Production Screen increases this limit by providing means to start applications from within the framework.

![i]	**Note**

Production Screen is only used to launch applications. A separate tool is required to create the applications.

For more information on creating FlexPendant applications, see http://developer-center.robotstudio.com, and the ScreenMaker section in *Operating manual - RobotStudio*.

The widget concept

Widgets are small applications with simple GUI:s that can run on the Production Screen desktop. The widgets can run simultaneously in the widget area and thus be used to construct a customized bigger view.

Up to 25 widgets can be placed in three different pages in the widget area.

The app concept

Apps are normal FlexPendant applications that are executable from Production Screen. Apps started from Production Screen have some extended behavior that is provided by the framework, for example error indication.

1.2 Glossary

Term list

Term	Explanation
app bar	A menu bar for apps at the bottom of the Production Screen desktop. The app bar consists of several pages with a maximum of six apps per page. At most 12 apps can run simultaneously.
app	Apps and applications started from, and running within, the Production Screen framework. The custom made apps can be created by the user as ordinary FlexPendant SDK applications.
application	Applications started from the ABB menu. Apps and applications are essentially the same. In this manual the term *apps* is used for applications started within the Production Screen framework, and the term *applications* is used for applications started from the ABB menu.
GUI	Graphical user interface.
task bar	Applications started from the ABB menu are shown in the task bar.
widget area	An area of the Production Screen desktop where widgets can be placed. It consists of three separate pages that can easily be browsed.
widgets	Small applications with a simple GUI running in one of three widget areas in Production Screen.

2 Installation

Prerequisites

The following is required to install Production Screen to a robot controller. Once installed, no specific software or tools are required.

- IRC5 controller equipped with the SxTPU3 FlexPendant (*3HAC028357-001*).
- RobotStudio.
- RobotWare build with Production Screen.
- RobotWare license key with the option *Production Screen* enabled.
- The RobotWare option *FlexPendant Interface* is required when running widgets and/or apps created in Visual Studio 2008.

The following is recommended but not required (optional).

- FlexPendant SDK installed on PC when developing widgets and apps with Visual Studio 2008.

Installing the software

Use this procedure to install the needed software to a PC.

	Action
1	Install RobotStudio.
2	Install RobotWare.
3	Optional: Install FlexPendant SDK, if using Visual Studio 2008 to develop widgets or apps.

For more information, see *Operating manual - RobotStudio*.
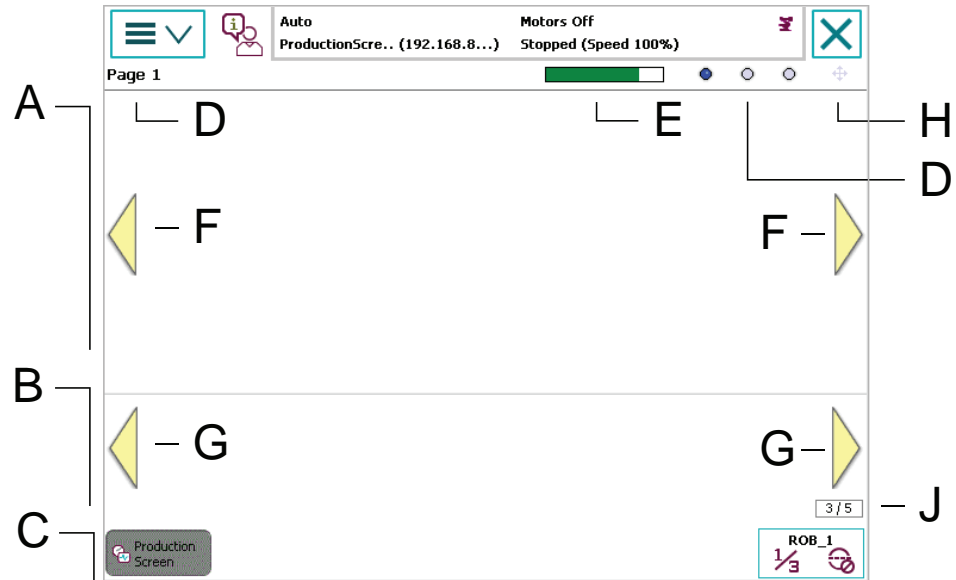
> **ⓘ Note**
>
> It is recommended to use the same release of RobotStudio, ProductionScreen (RobotWare), and FlexPendant SDK.

This page is intentionally left blank

3 The FlexPendant user interface

3.1 RobotWare Production Screen

Overview

xx1200000895

	Parts	Description
A	Widget area	Widgets are shown in the widget area.
B	App bar	Apps are shown in the app bar.
C	Task bar	Applications started from the ABB menu are shown in the task bar.
D	Widget page name Widget page indicator	Indicates the currently shown widget page. There are three widget pages available. Tap one of the indicators to view a different page.
E	Memory status indicator	Indicates the FlexPendant memory consumption. Tap the memory status indicator to switch from graphical indication to numerical indication in kilobytes (kB).
F	Widget page navigation arrows	Tap the widget area once to show the widget page navigation. The widget page navigation hides after three seconds. Tap the page navigation arrows to view the next page.
G	App bar navigation arrows	There are five app bars available. Tap the page navigation arrows to view the next app bar.
H	Widget move state	Click the icon to enable *widget move state*. When enabled, a widget can be moved to a new position by tapping and holding it. **ℹ Note** When *widget move state* is enabled, buttons in the widget cannot make use of tap and hold functionality.

	Parts	Description
J	App page indicator	A page indicator that shows the current and the last app bar number.

> 💡 **Tip**
>
> Use the stylus pen when navigating on the FlexPendant. The pen is located on the back of the FlexPendant.

The widget area

The widget area is a 10 x 4 grid of cells. Each cell is 60 x 60 pixels. The smallest space a widget can occupy is one cell.

Widget area navigation

Tap the widget page indicators or the widget page navigation arrows to change widget page.

To access the widget location view, first enable *widget move state* then tap and hold the widget. The widget can now be moved by tapping any cell in the widget area.

The widget can also be moved to a different page. Tap and hold a widget to access the widget location view. Tap the desired page in the widget page indicator. Finally, tap any cell in the widget area.

xx1200000896

> ℹ️ **Note**
>
> A padlock will appear when trying access the widget location view for a locked widget.

> ℹ️ **Note**
>
> When *widget move state* is enabled, buttons in the widget cannot make use of tap and hold functionality.

The app bar

There can be several app bars available, where each app bar can hold up to six apps.

App bar navigation

Tap the app bar navigation arrows to to view the next app bar.

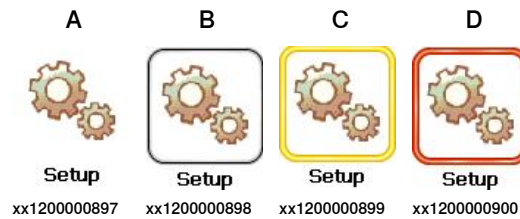Tap the button once to start the app. To close the app, first tap and hold the button, then tap the close menu item.

xx1200000902

App status

The status of the app is indicated by the border surrounding the app.

	Description	Status
A	No border.	The app is not running.
B	Black border.	The app is started.
C	Yellow border.	The app has an alert.
D	Red border.	The app has an error.

This page is intentionally left blank

4 Setting up Production Screen

4.1 Programming guidelines

Introduction

Developing an application for the FlexPendant, a device with limited process and memory resources, can be quite demanding. Issues such as performance and memory management need to be addressed.

It is strongly advisable to take the time to read the available manuals on this topic, before rushing into coding.

For more information on creating FlexPendant applications see http://developer-center.robotstudio.com, and the ScreenMaker section in *Operating manual - RobotStudio*.

The following sections are an addition to that information.

Creating widgets

Widgets can be created either in Visual Studio 2008, or in ScreenMaker for RobotWare 5.60 and later.

If FlexPendant SDK has been installed, a new widget or app project can be created by selecting the corresponding template in Visual Studio.

Visual Studio 2008

A widget must inherit the WidgetForm class. This class wraps the needed ITpsViewSetup and ITpsViewActivation interfaces into abstract methods which must be implemented according to FlexPendant SDK guidelines and best practices.

It is important to note that failing to deactivate subscriptions for widgets will impact performance negatively.

ScreenMaker

When creating a new widget in ScreenMaker for RobotStudio 6.0, the Production Screen widget template must be used.

When creating a new widget in ScreenMaker for RobotStudio 5.60, the type must be set to Production Screen in the screen project properties.

The binding functionality in ScreenMaker is supported by Production Screen, and the source can be configured dynamically in the *ProductionSetup.xml* configuration file. This means that a widget does not have to be rebuilt to change which RAPID variable or I/O signal is used by the binding. For more information, see *XML elements for the widget tag on page 23*.

For more information on how ScreenMaker is used to create widgets, see *Operating manual - RobotStudio*.

Creating apps

Follow the guidelines on creating FlexPendant applications.

> **ℹ️ Note**
>
> Applications created with ScreenMaker can be used in Production Screen.

TAF

The *Teach Pendant Application Framework* (TAF) is the application service provider that runs on the FlexPendant.

A FlexPendant application uses ITpsViewSetup and ITpsViewActivation to ensure proper install/uninstall and activation/deactivation.

Production Screen uses TAF to make sure that widgets and apps are installed and activated properly by forwarding the events.

Install and uninstall

It is important not to make any calls to FlexPendant or controller system objects from the constructor of the app or widget. These calls should, if they are part of the configuration of the widget or app, be made from the install routine. This way it is ensured that all system configurations and initialization have been performed.

See *Programming examples on page 21*.

Activate and deactivate

It is important that subscriptions to events, such as changes on I/O signals or RAPID variables, are deactivated when the widget goes out of scope.

This is to prevent the processor to be slowed down by execution of unnecessary code. This is done by setting up subscriptions in the activate method and tearing them down in the deactivate method.

A widget goes out of scope if the widget page is changed to one where the widget is not visible, or if the Production Screen application goes out of scope.

See *Programming examples on page 21*.

Indata

It is possible to pass configuration data from the setup file to a widget or app. This is done with the element `InData` in the configuration file.

This data will be interpreted as a string and passed as is to the install method. It is then possible to parse the file and configure the widget. The format of the in data is decided by the implementor of the widget or app.

See *Programming examples on page 21*.

4.2 Programming examples

Install and uninstall

```csharp
private AnalogMeterControl _meter;
private Controller _controller;
private AnalogSignal _signal;
private Image _bgImage;
//
public override bool Install(object sender, object data)
{
  if (_meter == null) return false;
  _controller = new Controller();
  string s = data as string;
  if (s != null)
  {
    string[] configs = s.Split(':');
    if (configs.Length >= 2)
    {
      _meter.Title = configs[0];
      _signal = _controller.IOSystem.GetSignal(configs[1]) as
          AnalogSignal;
    }
    if (_signal == null)
    {
      _meter.Title = "***Disconnected***";
      return false;
    }
    _meter.MinLevel = _signal.MinValue;
    _meter.MaxLevel = _signal.MaxValue;
    _meter.Image = _bgImage;
    return true;
  }
  return false;
}
```

Activate and deactivate

```csharp
public override void Deactivate()
{
  if (_signal != null) _signal.Changed -= new
      SignalChangedEventHandler(Signal_Changed);
}
public override void Activate()
{
  if (_signal != null)
  {
    _meter.Value = (double)_signal.Value;
    _signal.Changed += new
        SignalChangedEventHandler(Signal_Changed);
  }
}
```

4 Setting up Production Screen

4.3 The production setup XML file

Introduction

The setup of Production Screen is done in an XML file. The following chapter describes the setup of this XML file

The XML file is named *ProductionSetup.xml* and is located in the *".../HOME/ProdScr"* folder of the controller flash disk.

XML tags for the project

Following tags are used for the setup of the project.

Tag	Description
<ProjectSettings ... Grid = "false" Separator = "true" RemoveEmptyAppSlots = "true" >	Project settings. Show/hide the widget grid. Show/hide a graphical line between the widget area and the app bar. Remove all empty gaps in the app bar (default value is true).
<Apps>	Collection of apps.
<App>	Individual app.
<Widgets>	Collection of widgets.
<Widget>	Individual widget.
<WidgetPages> <Page> <Page> <Page>	The name of the widget pages as shown in the widget area. The name of page 1. The name of page 2. The name of page 3.

XML elements for the app tag

Following elements can be found in the <App> tag.

Elements	Description
<Name>	The name of the app.
<SystemApp>	Set to FALSE for own apps. Set to TRUE for BaseWare applications.
<Index>	The position of the app in the app bar. If the attribute *RemoveEmptyAppSlots* is set to true, the values are regarded as a sort order. Only one app can occupy a position and if several are configured to use the same position, the first will get the position and the rest will be moved to a free position. If *RemoveEmptyAppSlots* is false, the values are considered fixed and an error is raised if any conflicts are detected. There can be up to six apps in each app bar. Empty app bars are removed.
<Assembly>	The name of the *.dll* file including the file type. The file must be located in the *".../HOME/ProdScr/tps"* folder.
<Type>	The NameSpace name together with the main class name.

Continues on next page

Elements	Description
<Image>	The name of the icon image file including the file type. The image shall be 64 x 64 pixels and the allowed file types are *.png*, *.bmp*, *.gif*, and *.jpg*. The file must be located in the *".../HOME/ProdScr/tps"* folder.
<AlertSignal>	Digital output signal that can be set from RAPID or another application to indicate that this app wants the user to activate it and perform some task, indicated by a yellow frame around the app. The signal must be declared in the system (it will not be created).
<ErrorSignal>	Digital output signal that can be set from RAPID or another application to indicate that this app is in an error state and that the user should activate it and perform some task, indicated by a red frame around the app. The signal must be declared in the system (it will not be created).
<InData>	Optional string of input data to be interpreted by the app.

XML elements for the widget tag

Following elements can be found in the <Widget> tag.

Elements	Description
<WidgetLocked = "true">	Set to true if the widget shall be locked. Default value is false.
<Name>	The name of the widget. Must be unique.
<Page>	Widget page number.
<Assembly>	The name of the *.dll* file including the file type. The file must be located in the *".../HOME/ProdScr/tps"* folder.
<Type>	The NameSpace name together with the main class name.

<Bindings>
<Binding PropertyName="xxx" BindingType="yyy" DataName="zzz">
	A list of mappings between GUI controls and data on the controller, ie RAPID data and/or I/O-signals. Used by widgets made in ScreenMaker to dynamically redirect a ScreenMaker binding at startup. This enables reuse of widgets for different variables or I/O without having to recompile the widget.
	Example:
	<pre><Bindings>
	    <Binding PropertyName="meter1.Value"
	        BindingType="SIGNAL"
	        DataName="aoMeterSignal" />
	    <Binding PropertyName="meter1.Title"
	        BindingType="RAPID"
	        DataName="Flow1Title" />
	</Bindings></pre>

<Position> <X> <Y>	The position of the widget in the 10 x 4 grid of cells. The upper left corner is position (1, 1) and the bottom right corner is position (10,4).
<ZIndex>	The z-index is used when placing widgets on top of each other. Default value is 0. Allowed values are 0 to 99.
<InData>	Optional string of input data to be interpreted by the widget.

Multiple Configurations

Several products that use the Production Screen framework may want to add their widgets and apps to the configuration. To assist this process the framework allows new configurations to be added in the *.../HOME/ProdScr/config* folder. Any valid setup file with the extension *.xml*, will be merged into the existing *ProductionSetup.xml* file at the startup of the Production Screen application. This process will warn the user about conflicts, such as duplicated or overlapping apps or widgets.

> ![Note icon] **Note**
>
> The files will be removed when they have been merged into the setup.

5 RAPID reference information

5.1 Functions

5.1.1 PS_ChangeAppPage - Change the currently displayed app page

Usage

`PS_ChangeAppPage` *(change app page)* is a function that changes the currently displayed app page.

Basic examples

The following example illustrates the function `PS_ChangeAppPage`.

Example 1

```
PS_ChangeAppPage(1);
```
Displays the first app page.

Arguments

```
PS_ChangeAppPage (PageNumber)
```

`PageNumber`

Data type: `num`

The page number.

Program execution

The function `PS_ChangeAppPage` displays the selected app page.

Syntax

```
PS_ChangeAppPage '('
  [ PageNumber ':=' ] < variable (VAR) of num > ')'
```

5 RAPID reference information

5.1.2 PS_ChangeWidgetPage - Change the currently displayed widget page

Usage

PS_ChangeWidgetPage *(change widget page)* is a function that changes the currently displayed widget page.

Basic examples

The following example illustrates the function PS_ChangeWidgetPage.

Example 1

```
PS_ChangeWidgetPage(2);
```
Displays the second widget page.

Arguments

```
PS_ChangeWidgetPage (PageNumber)
```

PageNumber

Data type: num

The page number.

Program execution

The function PS_ChangeWidgetPage displays the selected widget page.

Syntax

```
PS_ChangeWidgetPage '('
  [ PageNumber ':=' ] < variable (VAR) of num > ')'
```

5.1.3 PS_IsRunning - Check if Production Screen is running

Usage

PS_IsRunning *(Production Screen is running)* is a function that checks if the Production Screen application is running.

Basic examples

The following example illustrates the function PS_IsRunning.

Example 1

```
VAR bool psrunning;
psrunning := PS_IsRunning();
```

psrunning will be TRUE if Production Screen is running, else it will be FALSE.

Return value

Data type: bool

The return value will be TRUE if Production Screen is running, else it will be FALSE.

Arguments

```
PS_IsRunning ( [\ForceUpdate] )
```

[\ForceUpdate]

Data type: switch

The ForceUpdate flag indicates that PS_IsRunning shall perform an actual handshake with the Production Screen framework.

Program execution

The function PS_IsRunning is used to check if the Production Screen framework is running.

Error handling

The following recoverable errors can be generated. The errors can be handled in an ERROR handler. The system variable ERRNO will be set to:

Name	Cause of error
ERR_WAIT_MAXTIME	Time-out when accessing Production Screen.

Syntax

```
PS_IsRunning '('
    ['\'ForceUpdate] ')'
```

A function with a return value of the data type bool.

This page is intentionally left blank

Index

Contact us

ABB AB
Discrete Automation and Motion
Robotics
S-721 68 VÄSTERÅS, Sweden
Telephone +46 (0) 21 344 400

ABB AS, Robotics
Discrete Automation and Motion
Nordlysvegen 7, N-4340 BRYNE, Norway
Box 265, N-4349 BRYNE, Norway
Telephone: +47 51489000

ABB Engineering (Shanghai) Ltd.
No. 4528 Kangxin Hingway
PuDong District
SHANGHAI 201319, China
Telephone: +86 21 6105 6666

www.abb.com/robotics

Power and productivity
for a better world™

ABB