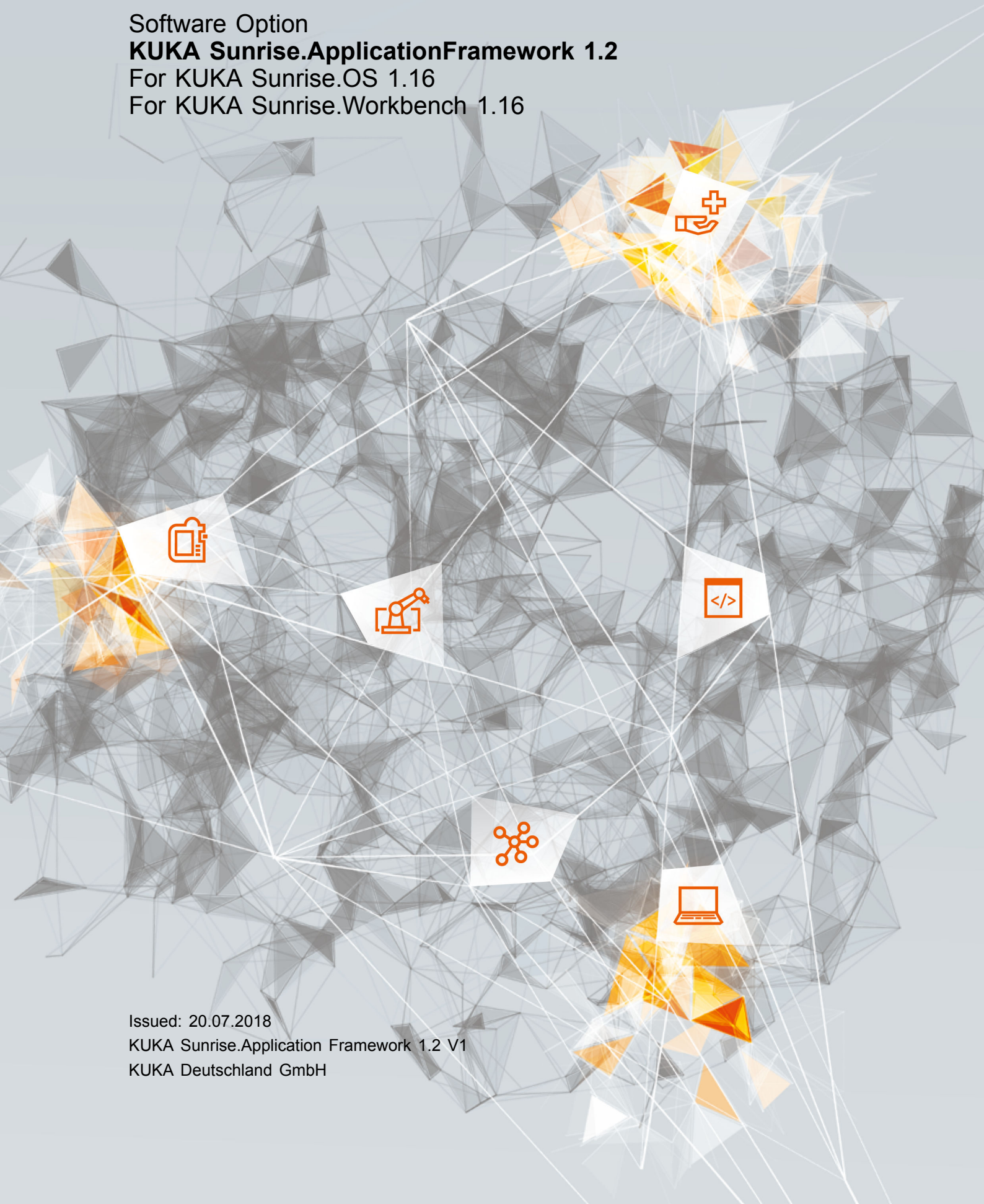
Software Option

KUKA Sunrise.ApplicationFramework 1.2

For KUKA Sunrise.OS 1.16

For KUKA Sunrise.Workbench 1.16



Issued: 20.07.2018

KUKA Sunrise.Application Framework 1.2 V1

KUKA Deutschland GmbH

KUKA Deutschland GmbH
Zugspitzstraße 140
D-86165 Augsburg
Germany

Other functions not described in this documentation may be operable in the controller. The user has no claims to these functions, however, in the case of a replacement or service work.

We have checked the content of this documentation for conformity with the hardware and software described. Nevertheless, discrepancies cannot be precluded, for which reason we are not able to guarantee total conformity. The information in this documentation is checked on a regular basis, however, and necessary corrections will be incorporated in the subsequent edition.

Subject to technical alterations without an effect on the function.

KIM-PS5-DOC

Translation of the original documentation

Publication:	Pub KUKA Sunrise.Application Framework 1.2 (PDF) en PB11490
Book structure:	KUKA Sunrise.Application Framework 1.2 V1.1 BS10527
Version:	KUKA Sunrise.Application Framework 1.2 V1

Contents

1 Introduction

1.1 Target group

This documentation is aimed at users with the following knowledge and skills:

- Advanced system knowledge of robotics
- Advanced Java programming skills

For optimal use of our products, we recommend that our customers take part in a course of training at KUKA College. Information about the training program can be found at www.kuka.com or can be obtained directly from our subsidiaries.

1.2 Industrial robot documentation

The industrial robot documentation consists of the following parts:

- Documentation for the manipulator
- Documentation for the robot controller
- Operating and programming instructions for the System Software
- Instructions for options and accessories
- Parts catalog on storage medium

Each of these sets of instructions is a separate document.

1.3 Representation of warnings and notes

Safety

These warnings are relevant to safety and **must** be observed.

DANGER

These warnings mean that it is certain or highly probable that death or severe injuries **will** occur, if no precautions are taken.

WARNING

These warnings mean that death or severe injuries **may** occur, if no precautions are taken.

CAUTION

These warnings mean that minor injuries **may** occur, if no precautions are taken.

NOTICE

These warnings mean that damage to property **may** occur, if no precautions are taken.

These warnings contain references to safety-relevant information or general safety measures.
These warnings do not refer to individual hazards or individual precautionary measures.

This warning draws attention to procedures which serve to prevent or remedy emergencies or malfunctions:

SAFETY INSTRUCTION
The following procedure must be followed exactly!

Procedures marked with this warning **must** be followed exactly.

Notices

These notices serve to make your work easier or contain references to further information.

Tip to make your work easier or reference to further information.

1.4 Terms used

Term	Description
Sequence chart	Illustrates the program sequence and forms the basis for the graphical programming.
AppFw	Application Framework
Exception	Exception or exceptional situation An exception describes a procedure for forwarding information about certain program statuses, mainly error states, to other program levels for further processing.
JRE	Java Runtime Environment Runtime environment of the Java programming language
KUKA Sunrise Cabinet	Control hardware for operating industrial robots
KUKA Sunrise.OS	KUKA Sunrise.Operating System System software for industrial robots which are operated with the robot controller KUKA Sunrise Cabinet
KUKA smartHMI	KUKA smart human-machine interface The smartHMI is the user interface of the robot controller.
KUKA smartPAD	The smartPAD is the hand-held control panel for the robot cell (station). It has all the operator control and display functions required for operation of the station.

2 Product description

2.1 OverviewKUKA Sunrise.ApplicationFramework 1.2

KUKA Sunrise.ApplicationFramework 1.2 is a software package for modeling and executing program sequences.

- Using KUKA.Sunrise Workbench, robot applications are developed in a graphical editor with the aid of a sequence chart.
- The robot applications are executed on the robot controller.
- The sequence of the robot controller is visualized on the KUKA smartPAD. The operator can intervene in the sequence via the smartPAD.

2.2 Intended use

Use

KUKA Sunrise.ApplicationFramework 1.2 is an add-on technology package for modeling, executing und visualizing program sequences.

Required components:

- Development computer with KUKA Sunrise.Workbench
- KUKA Sunrise Cabinet robot controller
- KUKA smartPAD control panel

The software must only be operated in compliance with the specified system requirements.

Misuse

Any use or application deviating from the intended use is deemed to be misuse and is not allowed. The manufacturer cannot be held liable for any damage resulting from such use. The risk lies entirely with the user.

3 Installation

3.1 System requirements

Hardware

- KUKA Sunrise Cabinet

Software

- KUKA Sunrise.Workbench 1.16
- KUKA Sunrise.OS 1.16

3.2 Installing Application Framework in Sunrise.Workbench

Description

Application Framework must be installed in Sunrise.Workbench. Through the installation, the software catalog of Sunrise.Workbench is expanded.

- If supplied jointly with Sunrise.Workbench, the software option is automatically installed during the installation of Sunrise.Workbench.
- If Sunrise.Workbench is already installed at the time of delivery of the software option, the option must be subsequently installed in Sunrise.Workbench using the following procedure.

The following files can be selected for installation (delivered in the **options** folder):

- File **com.kuka.appframework.swb.repository.zip**

Precondition

- Sunrise.Workbench is installed.
- Data storage medium with the software to be installed

Procedure

1. Select the menu sequence **Help** > **Install new software ...**. The **Install** window is opened.
2. To the right of the **Work with** box, click on **Add**. The **Add repository** window is opened.
3. Click on **Archive …** and navigate to the installation file:
   com.kuka.appframework.swb.repository.zip
4. Select the file and confirm with **Open**. The **Position** box now displays the path to the file.
5. Click on **OK**.
   - In the **Install** window, the **Work with** box now displays the path to the file.
   - The window now also displays the **Sunrise.OS** check box.
6. Activate the **Sunrise.OS** or **KUKA Application Framework** check box.
7. Leave the other settings in the **Install** window as they are and click on **Next >**.
8. The **Installation details** overview is displayed. Leave the settings as they are and click on **Next >**.

9. A license agreement is displayed. In order to be able to install the software, the agreement must be accepted. Then click on **Finish**.
   The installation operation begins.
10. A safety warning concerning unsigned contents is displayed. Confirm with **OK**.
11. Answer the request for confirmation with **OK**.
12. A message indicates that Sunrise.Workbench must be restarted in order to apply the changes. Click on **Restart now**.
13. Sunrise.Workbench restarts. This completes installation in Sunrise.Workbench.

4 Operation

4.1 General overview of user interface

The user interface of Application Framework consists of several views. The combination of several views is called a perspective.

The **Programming** perspective is opened as standard. After installation of Application Framework, the **AppFw** perspective can also be displayed.

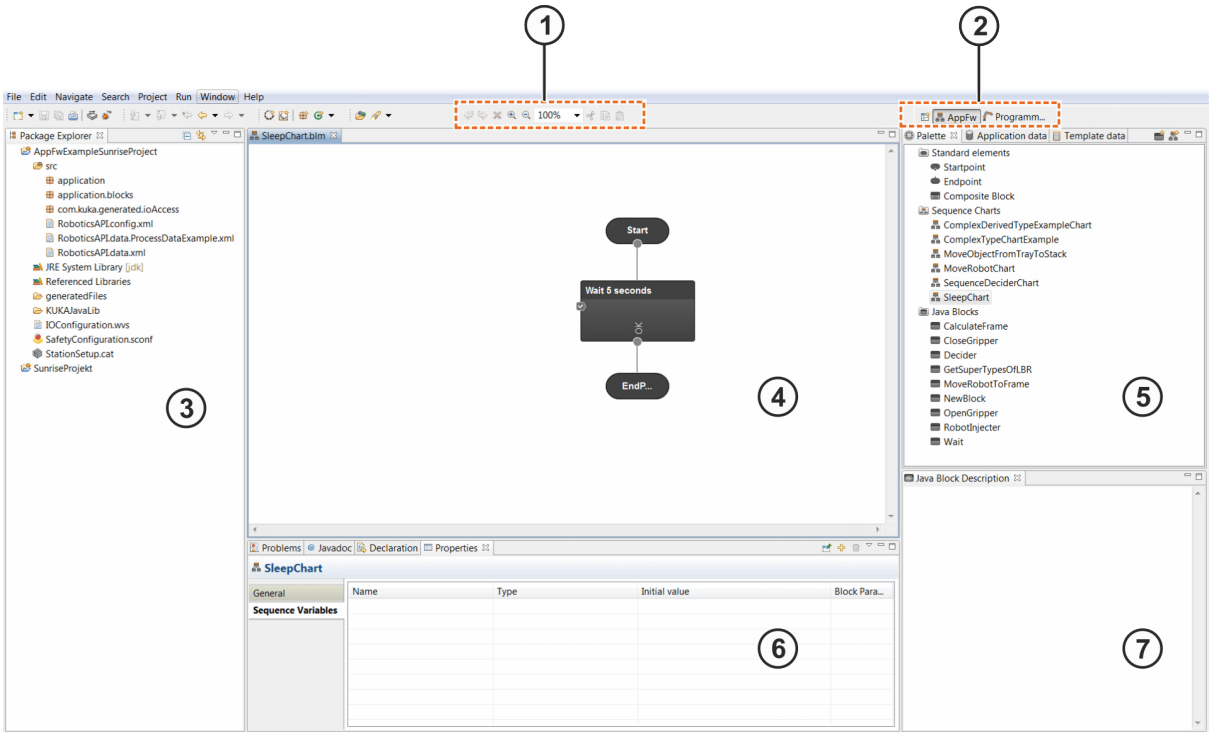


Fig. 4-1: User interface with “Application Framework” perspective

Item	Description
1	Toolbar for editing the sequence charts in the editor area (>>> *5.5.1 "Toolbar" Page 17*)
2	Perspective selection (>>> *4.2 "Activating a perspective or view" Page 11*)
3	**Package Explorer** view This view contains the projects created and their corresponding files.
4	Editor area In the editor area, the sequence charts are created, edited and displayed.
5	**Palette** view The view contains standard elements and Java blocks with which sequence charts can be created. Additionally, all available sequence charts are displayed. (>>> *5.1.1.1 "Standard elements" Page 14*) (>>> *5.1.1.2 "Sequence charts" Page 14*)

Item	Description
	(>>> *5.1.1.3 "Java blocks" Page 15*)
6	**Properties** view • Header Depending on whether a selection was made in the **Palette** view or in the Editor area, the following is displayed: – Icon – Name of the class (sequence chart or Java block) – Block name • **General**: – If a block is selected in the sequence chart, its properties are displayed here, e.g. name of the block or definition of the outputs with the name of the corresponding class. – If a block is selected in the **Palette** view, the general properties of the block class are displayed. – Clicking in the editor area next to a sequence chart causes the properties of the sequence chart to be displayed. • **Sequence variables**: The sequence variables created for a sequence chart are displayed on this tab. Only displayed on clicking in the editor area. (>>> *5.7.1 "Defining sequence variables" Page 21*) • **Block parameters**: The parameters of a block are displayed on this tab. (>>> *6.1.5 "Defining block parameters" Page 27*) (>>> *5.7.2 "Assigning block parameters" Page 22*) • **Block outputs**: The outputs of a block are displayed on this tab. (>>> *6.1.6 "Defining block outputs" Page 28*) (>>> *5.7.3 "Assigning block outputs" Page 22*)
	Errors view A summary of the errors is displayed in this view. (>>> *5.6 "Displaying errors" Page 20*)
7	**Java block description** view The documentation of the selected block is displayed in this view. This documentation is optional and must be supplemented by the application developer or system integrator.

4.2 Activating a perspective or view

Description

The user interface can be displayed in different perspectives. The default perspective is **Programming**.

Perspectives can be adapted to the requirements of the user, e.g. by activating/deactivating views or menus. It is possible to save the adapted perspective as a default setting for the perspective or under a separate name of its own.

Procedure

To activate a perspective for the first time:

- Select the menu sequence **Window** > **Open perspective** > **Other...** and select the desired perspective. Confirm with **OK**.

  The perspective is opened. Furthermore, the corresponding button is activated in the perspective selection.

To activate a perspective that has already been used:

- In the perspective selection, click on the button for this perspective.

To display views in the current perspective:

- Select the menu sequence **Window** > **Show View** and the desired view.

  Further views can be selected by clicking the menu item **Other…**.

To save user-defined perspectives:

1. Select the menu sequence **Window** > **Save Perspective As...**.
2. In the **Name** box, enter a name for the perspective and confirm it with **OK**.

If an existing perspective is selected and overwritten, the perspective will be opened with these settings in the future.

To reset the current perspective to the default setting:

- Select the menu sequence **Window** > **Reset Perspective…** and answer the request for confirmation with **Yes**.

5 Sequence programming

5.1 Sequence chart overview

Description

In a sequence chart, processing steps are represented by blocks. A block may be either a Java block that runs a stored Java code or a sub-sequence block that starts another chart. The possible result states of the block processing are defined by outputs on the blocks.

The outputs are linked to other blocks by block connections. If a processing step ends with a particular result, the block linked to this result will be executed.

(>>> *5.1.3 "Blocks and connections" Page 15*)

Before a processing step is executed, preconditions can be checked in a Java block. The Java block will only be executed if all preconditions are met.

(>>> *6.1.4 "Programming preconditions" Page 25*)

(>>> *6.1.4.1 "Activating a cyclic check of the preconditions" Page 26*)

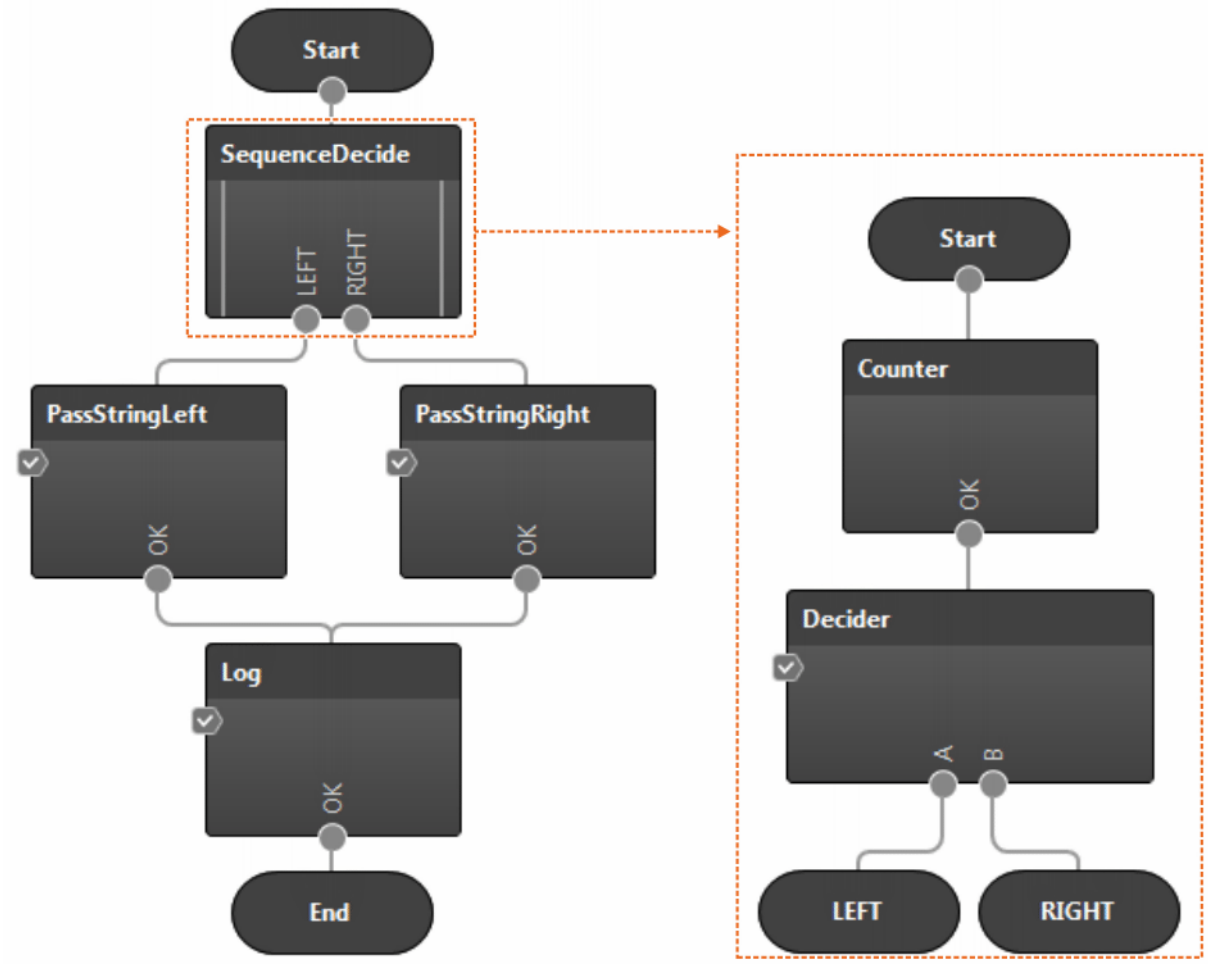


Fig. 5-1: Overview of sequence chart with open sub-sequence block

5.1.1 Standard elements and Java blocks

5.1.1.1 Standard elements

Description

The following standard elements are available:

- Start point
- End point
- Sub-sequence

Each sequence chart must have a start point and at least one end point.

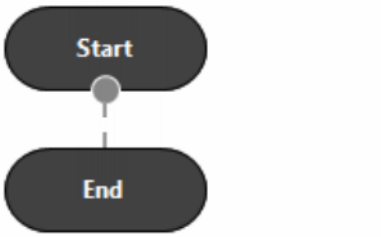


Fig. 5-2: Start and end points

The structure of Java and sub-sequence blocks is essentially identical.

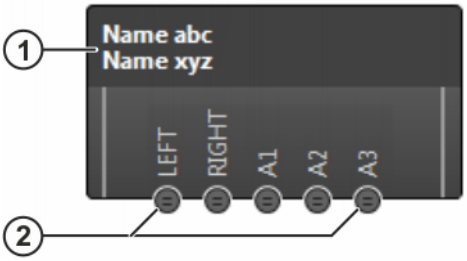


Fig. 5-3: Structure of sub-sequence block

Item	Description
1	Header with the name of the block A descriptive name makes both programming and orientation within the program easier. Block names may be used more than once within a sequence chart. A sub-sequence block is indicated by lines on the right and left and comprises a further sequence chart. The name of a block may consist of up to 2 lines.
2	Non-connected outputs

5.1.1.2 Sequence charts

Description

In the **Sequence charts** folder, all available sequence charts are displayed:

- Users can create sequence charts themselves.
- Double-clicking on a sequence chart opens the corresponding sequence chart *Name*.blm in the Editor area.

- A sequence chart can be dragged into an open sequence chart in the Editor area. A sub-sequence block is created.

5.1.1.3 Java blocks

Description

In the **Java blocks** folder, all available Java blocks are displayed. These are derived from the available Java block classes of the current project. Users can create additional Java block classes themselves.

(>>> *6.1.3 "Creating a new Java block class" Page 24*)

Double-clicking on a Java block in the **Java blocks** folder opens the corresponding *Name*.java file in the Editor area.

5.1.2 Status display

Description

If at least one parameter is defined in the referenced block class, the status of the parameter is displayed. The status indicator is visible on the corresponding block at the left-hand edge.

Status	Description
	Error A parameter is incorrect.
	Warning A parameter has not been assigned.
	Status OK All parameters have been assigned.

5.1.3 Blocks and connections

Description

The blocks in a sequence chart must be connected to each other. Each sequence chart must have a start point. The application starts with the block after the start point (start block).

A block can have any number of outputs. The outputs of the blocks can be connected to other blocks or to end points. The sequence chart is exited when the end point is reached.

The application ends when a non-connected output or an end point in the first sequence chart is reached.

An end point can be connected to any number of blocks.

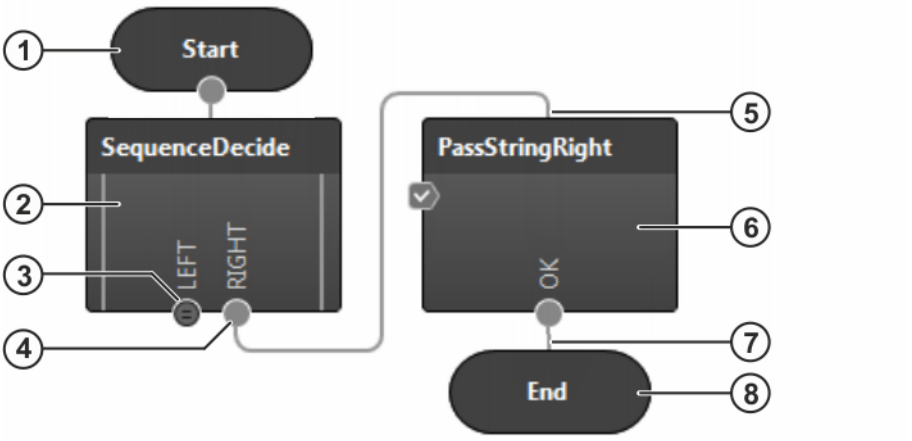


Fig. 5-4: Connecting blocks

Item	Description
1	Start point
2	Start block (here: sub-sequence block)
3	Icon for a non-connected output.
4	Icon for a connected output. An output can be assigned to no more than one block connection.
5	The block connection to an input is always placed at the middle of a block.
6	Java block
7	Block connection from an output to an end point.
8	End point

5.2 Creating a new sequence chart

Procedure

1. Select the project in the **Package Explorer**.
2. Select the menu sequence **File** > **New** > **Sequence chart**. The **New sequence chart** window is opened.
3. In the **Package** box, enter the name of the package in which the sequence chart is to be created.
4. Enter a name for the sequence chart in the **Name** box.
5. Click on **Finish**. The files *Name*.java and *Name*.blm are created and inserted into the selected package of the project.

   The sequence chart *Name*.blm is opened in the editor area. A start point is already present and does not have to be created separately.

Alternatively, a new sequence chart can be created using the additional icon in **Palette** view:

Icon	Name / description
	**Creates a new sequence chart.** Opens the **New sequence chart** window.

5.3 Inserting elements

Standard elements, sequence charts and Java blocks can be dragged from the **Palette** view into an open sequence chart in the Editor area. Standard elements, sequence charts and Java blocks are linked together by means of connections to form a complete sequence chart.

A Java block is only present once in the **Palette** view whereas the Java block can be created multiple times in a sequence chart.

5.4 Linking a sub-sequence block to a sequence chart

Description

A sequence chart can be used as a sub-sequence in another sequence chart. For this, a sequence chart must be dragged to the Editor area or a sub-sequence block must be linked to the corresponding sequence chart.

Precondition

- Sequence chart for the sub-sequence has been created.

Procedure

1. Double-click on the unlinked sub-sequence block in the editor area. The **Open resource** dialog is displayed.

   In the dialog all the blm files are shown that can be linked to the selected sub-sequence block.
2. Assign the object. Confirm with **OK**.
3. The sub-sequence block assumes the name of the linked sequence chart.

Sequences with recursions are invalid and cannot be transferred to the controller. References that lead to recursions are marked in red in the dialog.

If a sequence chart is linked as a sub-sequence, the end points become block outputs.

5.5 Editing a sequence chart

Description

The entire sequence chart is displayed in the editor area and can also be edited there.

5.5.1 Toolbar

The buttons available by default on the toolbar depend on the active perspective. The buttons of the **AppFw** perspective for editing the sequence chart in the editor area are described here.

Icon	Name / description
	**Undo** Undoes the last input.
	**Redo** Redoes the previously undone input.

Icon	Name / description
	**Delete** Deletes the selected element (e.g. block).
	**Zoom in** Zooms into the sequence chart.
	**Zoom out** Zooms out of the sequence chart.
	**Cut** Deletes the selected element from its original position and copies it to the clipboard.
	**Copy** Copies the selected element to the clipboard.
	**Paste** Pastes the cut or copied standard elements, sequence charts or Java blocks.

5.5.2 Aligning elements with other elements

Procedure

1. Click on an element in **Palette** view and drag it to the Editor area, or click on an element in the Editor area.
2. Hold down the left-hand mouse button and move the element.
3. If an element is moved within the vicinity of an adjacent element, a broken orange guide line is displayed. Using the guide line, the element can be aligned with the center or the edges of the adjacent element.
4. The new position is shown in the preview. This position is accepted when the mouse button is released.

5.5.3 Selecting multiple elements

Procedure

To select multiple elements:

1. Click into the editor area.
2. Hold down the left-hand mouse button. The mouse pointer changes into cross hairs.

   Drag the cross hairs over the blocks and connections that are to be selected. Blocks that are only partly touched when dragging the selection window are also selected. Connections between 2 selected blocks are also selected.
3. All the blocks and connections touched are accepted when the mouse button is released.
4. Click on the selection. Hold down the left mouse button and move the selection or align it with adjacent elements.
5. The new position is shown in the preview. This position is accepted when the mouse button is released.

To select all elements:

1. Click into the editor area.

2. All elements can be selected via the menu sequence **Edit** > **Select all**.

   Alternatively: using the keyboard shortcut, as is typical in Windows.
3. Click on the selection. Hold down the left-hand mouse button and move the selection.
4. The new position is shown in the preview. This position is accepted when the mouse button is released.

5.5.4 Copying and cutting elements

Selected elements and connections can be copied and cut. If the elements are pasted into any sequence chart, referenced Java block classes or sub-sequences are copied with them.

References and files are also copied if the selected elements and connections have been labeled completely.

Sequence variables are not copied.

5.5.5 Zooming into and out of a sequence chart

Procedure

1. Click into the sequence chart.
2. Hold down the CTRL key and scroll with the mouse wheel.
   - Mouse wheel up: zoom in
   - Mouse wheel down: zoom out

5.5.6 Moving elements

Procedure

Moving with the mouse:

1. Click on the element.
2. Hold down the left-hand mouse button and move the selection.
3. The new position is shown in the preview. This position is accepted when the mouse button is released.

Moving with the arrow keys:

1. Click on the element.
2. Move it using the arrow keys of the keyboard.
3. The new position is shown in the preview. Confirm the desired position by pressing the Enter key.

5.5.7 Enlarging elements

Procedure

1. Click on an element.
2. Use the handles at the corners or the selection edges to enlarge or shrink the element.

   If no handles are displayed, it is not possible to resize the element. The elements cannot be shrunk below the minimum size.

5.5.8 Deleting elements and connections

Procedure

1. Click into the sequence chart.
   Select the elements and connections that are to be deleted. If elements have incoming and outgoing connections, these too will be deleted.
2. Using the **Delete** button.
   Alternatively: Using the Delete key, as is typical in Windows.

5.5.9 Printing a sequence chart

Procedure

In order to print a sequence chart:

1. Open a sequence chart in the editor area.
2. Select the menu sequence **File** > **Print**. The **Printer** window is opened.
3. Select the desired printer.
   When printed, the sequence chart is automatically adapted to the page size of the paper configured in the printer.

5.6 Displaying errors

Description

The **Errors** view contains an overview of the errors in a table. The table contains the following information:

- **Description** column:
  Brief description of the nature of the error.
- **Resource** column:
  Name.blm of the sequence chart or of the application *Name*.java in which an error has occurred
- **Path** column:
  Specification of the path in the Package Explorer in which the sequence chart or application is located
- **Position** column:
  Specification of the line within the application *Name*.java in the Editor area
- **Type** column:
  Specification of the error type (e.g. sequence chart error or Java error)

Additionally, the following icons contain a round or rectangular error symbol:

- In the Package Explorer: on all folders of the path in which the sequence chart or application is stored and on the corresponding *Name*.blm or *Name*.java
- In the Editor area
- In the **Errors** view

5.7 Data flow

Description

With the aid of the block outputs and block parameters, data are passed on from one block to other blocks during program execution. Sequence variables can be linked to the block outputs and block parameters and act as buffers for the data:

- Sequence variables:
  Each sequence chart may contain any number of sequence variables. Sequence variables can be defined on the **Sequence variables** tab.
  (>>> *5.7.1 "Defining sequence variables" Page 21*)
- Block parameters:
  Block parameters can be linked to various sources. The corresponding value is assigned to the block parameter before the block is executed.
  (>>> *5.7.2 "Assigning block parameters" Page 22*)
- Block outputs:
  After it has been executed, a block can make data available via block outputs. Block outputs can be linked to sequence variables.
  (>>> *5.7.3 "Assigning block outputs" Page 22*)

Only elements of the same data type can be linked.

5.7.1 Defining sequence variables

Sequence variables that have already been created can be selected and modified by clicking on the table entry. All parameter types with the annotation @Input can be selected.

Additional icon on the **Sequence variables** tab:

Icon	Name / description
	**Create new sequence variable** Creates a new sequence variable.

Procedure

To create a new sequence variable:

1. Open a sequence chart in the editor area.
2. Click into the editor area.
3. On the **Sequence variables** tab, select the **Create new sequence variable** icon.
4. Enter a name for the new sequence variable in the **Name** box.
5. Select the type of the new sequence variable in the **Type** column:
   - Enum: The values can be selected in a window.
   - Complex parameter types: No values can be entered.
6. If required, a start value matching the type can be assigned in the **Initial value** column for the sequence variable.
7. Set the check mark in the **Block parameter** column if the sequence variable is to be activated as a block parameter.
   - Check box active: Sequence variable activated as block parameter.
     If the corresponding sequence chart is linked as a sub-sequence block, the sequence variable becomes a block parameter.

The change to the name of a sequence variable is applied wherever the sequence chart is used as a sub-sequence block.

- Check box not active: Sequence variable not activated as block parameter.

  If a check mark is removed, the corresponding block parameters are deleted, even if the sequence chart is used as a sub-sequence block. In this case, a new connection must be created.

NOTICE

No block outputs can be created for sub-sequence blocks.

5.7.2 Assigning block parameters

Procedure

1. Select a block from the sequence chart in the editor area.
2. On the **Block parameters** tab, select a parameter.
3. Depending on the parameter type, the following sources can be selected in the **Source** column:
   - Constant
   - Variable
   - Output
   - Environment
4. An appropriate entry must be made in the **Value** column according to the source:
   - Constant: Assignment of a constant value of data type Integer, Double, String, Boolean or Enum
   - Variable: Assignment of a sequence variable
   - Output: Assignment of a block output of another block in the same chart
   - Environment: Selection from the **Select environment object** list. This list contains elements of the matching type, e.g. from the **Template data** view and the **Application data** view.

In the case of sequence variables and block outputs, it is not necessary for the block output to return the type required by the block parameter.
Derived types can be assigned to more general types:

- Block outputs can be assigned to block parameters or sequence variables.
- Sequence variables can be assigned to block parameters.

5.7.3 Assigning block outputs

Procedure

1. Select a block from the sequence chart in the editor area.
2. On the **Block outputs** tab, select a parameter.
3. Select a sequence variable in the **Assignment** column.

6 Programming

6.1 Overview

Programming is subdivided into the following steps:

Step	Description
Program an application class	(>>> *6.1.2 "Programming an application class" Page 24*)
Create Java block classes	(>>> *6.1.3 "Creating a new Java block class" Page 24*)
Program preconditions	(>>> *6.1.4 "Programming preconditions" Page 25*)
Check preconditions	(>>> *6.1.4.1 "Activating a cyclic check of the preconditions" Page 26*)
Define block parameters	(>>> *6.1.5 "Defining block parameters" Page 27*)
Define block outputs	(>>> *6.1.6 "Defining block outputs" Page 28*)
Set icons for block classes	(>>> *6.1.8 "Icons for block classes" Page 29*)
Create robot applications with the aid of Application Framework	(>>> *6.1.9 "Creating robot applications" Page 30*)

6.1.1 Displaying Javadoc information

Description

Javadoc is a documentation generated from specific Java comments. The functionalities and use of classes, methods and libraries are described in Javadoc.

There are various display options.

Precondition

- An open program code is displayed in the editor area.

Procedure

1. Move the mouse pointer to the desired element name in the program code. The associated Javadoc information is automatically displayed in a window in the editor area.

   The following elements react to the mouse pointer:

   - Methods
   - Classes (data types, not user-defined data arrays)
   - Interfaces
   - Enums
2. Select the desired element in the program code and press F2.

6.1.2 Programming an application class

Description

A sequence chart that is to be displayed on the smartPAD as an independent application requires an application class. The application class is derived from the IAppFwApplication interface and provided with the annotation @AppFwApplicationCategory.

If a sequence chart is used as a sub-sequence, the application class may be omitted.

The annotation establishes the connection between the application class and the sequence chart.

The following options are available for specifying the path of the sequence chart:

- If the sequence chart and application are in the same package, only the name of the sequence chart is required (e.g. @AppFwApplicationCategory(model = "BlockModel.blm")).
- If the sequence chart and application are in different packages, the corresponding path of the sequence chart must be specified (e.g. @AppFwApplicationCategory(model = "/differentPackage/BlockModel.blm")). Different source folders are not taken into account.

The following methods must be implemented in the application class:

Method	Description
initialize()	This method is called before the application is executed.
dispose()	This method is called after the application is executed.

Example

```
import com.kuka.appframework.IAppFwApplication;
import com.kuka.appframework.AppFwApplicationCategory;
@AppFwApplicationCategory(model = "TestBlockModel.blm")
public class TestBlockModel implements IAppFwApplication {
    @Override
    public void initialize()
    {...}
    @Override
    public void dispose()
    {...}
}
```

6.1.3 Creating a new Java block class

Description

In the **Palette** view, new Java block classes can be created.

Additional icon in the **Palette** view:

Icon	Name / description
	**Create new Java block class** Opens the **New Java block class** window.

Precondition

- **Palette** view is open.

Procedure

1. In the **Palette** view, select the **Create new Java block class** icon. The **New Java block class** window opens.
2. Enter a name for the new Java block class in the **Name** box.
3. Click on **Finish**. The new Java block class is created and inserted into the project package and the **Palette** view.

   The new Java block class *Name*.java is opened in the editor area.

The following method is available:

Method	Description
run(…)	This method contains the code that is executed when the block is executed. The value returned by the method determines the output at which the block is exited in the sequence chart.

Example

The Java block class opened in the editor area contains an Enum type “Result” with the predefined identifier “OK”. “Result” can be modified and extended as desired. Each of the identifiers, which have to be separated by a comma, defines a new block output:

```
package application.blocks;
import com.kuka.appframework.ApplicationBlock;
    public class Block extends ApplicationBlock<Block.Result>
    {
        public enum Result {
            OK,
            //...
        }
        @Override
        public Result run() throws Exception
        {
            //...
            return Result.OK;
        }
    }
```

When assigning designations (e.g. names of block classes or outputs), the rules of Java programming must be observed.

6.1.4 Programming preconditions

Description

In order to program a check of the preconditions for a block, the checkPreconditions() method must be overwritten in the Java class of the block. The method is called immediately before execution of the run() method and interrupts the further sequence if the check of the preconditions fails.

For each precondition to be checked, the check() method must be called by checker. A Boolean value and a message text must be transferred to the method:

- **true**: The precondition is fulfilled.
- **false**: The precondition is violated.
- The message text is displayed on the smartPAD if the precondition is violated.

  (>>> *7.3.7 "Ignoring failed preconditions" Page 40*)

- The method should not perform any extensive calculations or time-consuming communication so as not to delay the sequence unnecessarily.
- No actions should be executed with the method (e.g. moving the robot, switching outputs).

Example

The method checks whether the gripper is closed and the input signal is set:

```
@Override
public void checkPreconditions(IChecker checker) throws Exception {
    checker.check(isGripperClosed(),
        "Gripper must be closed.");
    checker.check(isSignalSet(),
        "Signal must be set before executing this block.");
}
```

6.1.4.1 Activating a cyclic check of the preconditions

Description

The block class can be provided with the annotation @CyclicPreconditionCheck so that the runtime environment independently carries out cyclic checks after failure of a precondition check. The checks are repeated at the specified intervals until all preconditions have been met. The non-fulfilled preconditions are displayed on the smartPAD and updated after each check. When all the preconditions have been met, the user must resume program execution by pressing the Start key.

The preconditions are checked again every time program execution is resumed. If preconditions that have already been identified as not fulfilled are to be ignored, the user can select the **Ignore precondition** option on the smartPAD.
(>>> *7.3.7 "Ignoring failed preconditions" Page 40*)

Syntax

`@CyclicPreconditionCheck(interval=`*time*`)`

Explanation of the syntax

Element	Description
time	Time between the individual checks (type: int, unit: ms) Default: 1000 If the parameter is not specified, the default value is automatically used. **Note**: An interval > 400 ms should be selected to avoid overloading the system.

6.1.5 Defining block parameters

Description

For each block parameter a method must be defined and provided with the annotation @Input. The parameter can be stored in an array of the block class and used in the checkPreconditions() and run() methods.

The following conditions must be met:

- The name of the method must begin with **set**.
- The return value is **void**.
- The method is not **static**.
- The method is **public**.
- Each method has only one parameter with one supported parameter type.
  - Simple parameter types: String, int/Integer, double/Double, boolean/Boolean
  - Enum
  - Complex parameter types: e.g. geometric types from the RoboticsAPI. These can be used to assign environment objects from the **Template data** and **Application data** views.
  - The following are not supported: Arrays, generic parameter types, byte/Byte, char/Character, float/Float, long/Long, short/Short
- Each assigned name of the @Input annotation must be unique.
- The block parameter can also be defined in a superclass.
- The block parameters for sub-sequence and Java blocks are displayed on **Block parameters** tab. Additionally, the corresponding status indicator is displayed in the left-hand column next to the name.

Example

Names of the block parameters: **value** and **force**

```
@Input
public void setValue(String value) {
    this.value = value;
}
@Input
public void setForce(double force) {
    this.force = force;
}
```

After the changes to the block parameters have been saved in the editor, the changes are shown in the **Block parameters** view and can be assigned.

(>>> *5.7.2 "Assigning block parameters" Page 22*)

If the data type can be displayed as a string, a default value can be specified. If the default value is to be used for initial parameterization when using the block in the editor, it can be specified in the @Input annotation:

```
@Input(defaultValue="1.23")
public void setForce(double force) {
    this.force = force;
}
```

6.1.6 Defining block outputs

Description

For each desired block output a method must be defined and provided with the annotation @Output.

The following conditions must be met:

- The name of the method must begin with **get**.
- The method is not **static**.
- The method is **public**.
- Each method has only one return type with one supported parameter type:
  - Simple parameter types: String, int/Integer, double/Double, boolean/Boolean
  - Enum
  - Complex parameter types
- The block output can also be defined in a superclass.

The get methods are called after the run method has been executed.

If a get method returns an invalid value or triggers an exception, the application is paused with a corresponding error message. In this case, none of the sequence variables linked to the block will be updated. Furthermore, no values will be forwarded to the block inputs connected to the block outputs.

Example

Names of the block outputs: **value** and **precision**

```
@Output
public String getValue() {
    return this.value;
}
@Output
public double getPrecision() {
    return 0.89;
}
```

6.1.7 Editing outputs

Description

In order to add or delete an output in a block, a value can be added or deleted to the Enum type “Result” in the Java block class.

If a linked output is deleted, this will be shown in red in the sequence chart after the Java block class has been saved.

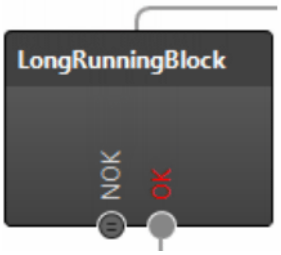


Fig. 6-1: Editing an output

Precondition

- A sequence chart is open in the editor area.

Procedure

1. Click into the sequence chart.
   Double-click on the block in which the name of the output is to be changed. The Java block class is opened in the editor area.
2. Change the name of the corresponding value of the Enum type “Result” in the Java block class.
   The change is adopted in all sequence charts after the Java block class has been saved.

6.1.8 Icons for block classes

Description

Each block class can be assigned an icon. An icon is assigned to a block class using the annotation @Icon. The icon is permanently displayed in the block of the sequence chart.

Syntax

```
@Icon (id=BlockIcon.Name)
```

Explanation of the syntax

Element	Description
Name	Type: Enum of type BlockIcon Name of the icon to be assigned to a block. Only predefined icons can be used.

Name	Icon	Name	Icon
CALCULATE		MOVE	
CAMERA		ON_OFF	
DECIDE		PALLETIZE	
DEPALLETIZE		PICK	
DESTACK		PLACE	
ENTER		REPEAT	
INFORM		REQUEST	
LEAVE		STACK	
MEASURE		WAIT	
MEAS-URE_BASE		-	-

Example

For greater clarity, the MOVE icon is to be added to a block with motion sequences.

```
@Icon (id=BlockIcon.MOVE)
public class First extends ApplcationBlock<First.Result>
```

6.1.9 Creating robot applications

Description

In order to create robot applications with the aid of Application Framework, the annotation @UseRoboticsAPIContext must be added to the application class. As a result, the application class and Java blocks obtain access to RoboticsAPI types via dependency injection.

The following must be taken into consideration when programming robot motions:

- The run() method may not be quit until the motion command has been finished. This precludes commanding with moveAsync() and IMotionContainer.append() as the last motion command.
- Approximation of motions between blocks is not supported.

If these points are not observed, operator errors may result due to discrepancies between display and selection of the active block. Unexpected robot motions are also possible.

Example

A robot is to address an end point with the default TCP of a tool and to issue a message when doing so. The end point and the tool are to be configurable as block parameters in Sunrise Workbench:

```
import static com.kuka.roboticsAPI.motionModel.BasicMo-
tions.ptp;
import javax.inject.Inject;
import com.kuka.appframework.ApplicationBlock;
import com.kuka.appframework.Input;
import com.kuka.roboticsAPI.geometricModel.ObjectFrame;
import com.kuka.roboticsAPI.geometricModel.Tool;
import com.kuka.task.ITaskLogger;
public class MoveTool extends ApplicationBlock<MoveTool.Re-
sult> {
    public enum Result {
        OK
    }
    private Tool tool;
    private ObjectFrame target;
    @Inject
    private ITaskLogger log;
    @Input
    public void setTool(Tool tool) {
        this.tool = tool;
    }
    @Input
    public void setTarget(ObjectFrame target) {
        this.target = target;
    }
    @Override
```

```
    public Result run() throws Exception {
        log.info("Moving tool" + tool +
            "to frame" + target.getName());
        tool.move(ptp(target));
        return Result.OK;
    }
}
```

7 KUKA smartHMI operator control

7.1 User groups

User privileges

As standard, no special authorization is required for executing the software functions on the smartHMI. If the user group “Expert” is installed, the default user may no longer execute the following functions. The user must be logged on as “Expert” or higher.

Function	Operator	Expert	Safety maintenance
Modify block parameters	✗	✓	✓
Modify sequence variables	✗	✓	✓
Ignore preconditions	✗	✓	✓
Block selection with an application running	✗	✓	✓

7.2 Applications view with sequence chart

Precondition

- The project is synchronized.
- The applications with the sequence charts are available to the robot controller.

Procedure

1. In the navigation bar, select the desired application under **Applications**.
2. Press the **Sequence chart** tile. The selected sequence chart is opened. The button for editing is displayed.

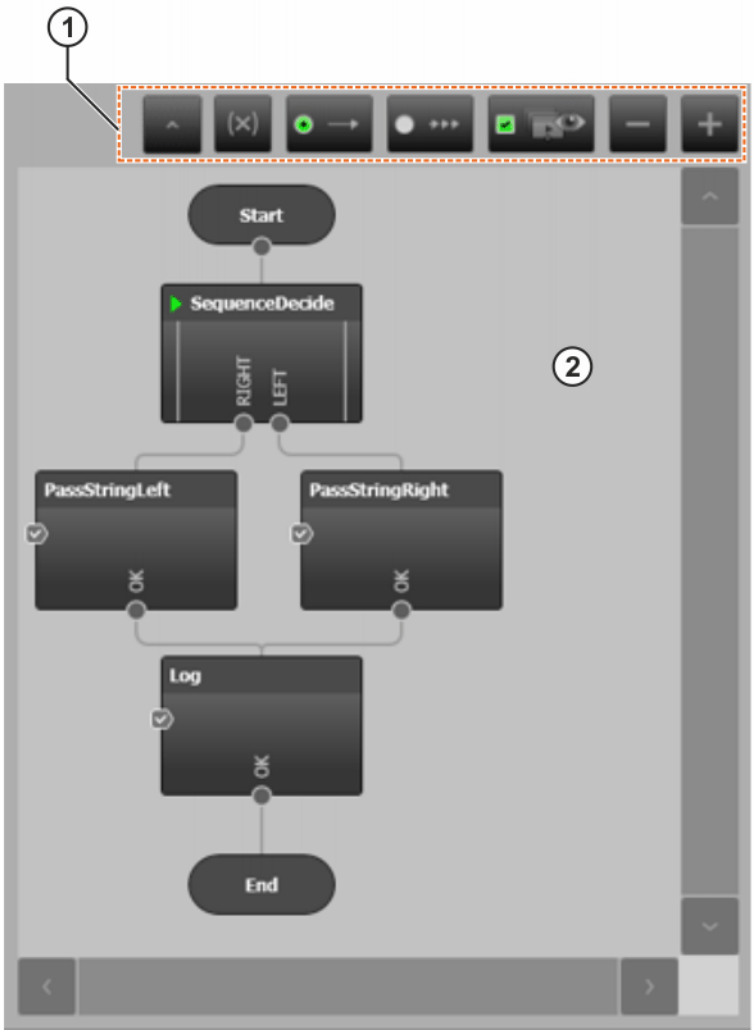


Fig. 7-1: Applications view with sequence chart

Item	Description
1	Button bar for editing
2	Display area Display of the sequence chart

Button	Description
	Returns to the higher-level sequence chart (>>> *7.2.4 "Opening a sub-sequence" Page 38*)
	Opens the “Sequence variables” view (>>> *7.2.3 "Displaying and modifying sequence variables" Page 36*)
	The application can be executed in continuous mode or in step mode. It is possible to switch from one mode to the other at any time. (>>> *7.2.5 "Selecting the program run mode" Page 38*)
	Scroll function If the scroll function is activated, the window contents are automatically moved during program execution. The block currently being executed is displayed.

Button	Description
	The scroll function can be activated or deactivated at any time.
	Zooms into and out of the sequence chart • - : Zoom is decreased • + : Zoom is increased

7.2.1 Modifying program execution by changing blocks

Description

Within a sequence chart, the sequence of execution can be modified by jumping to any selected block, start point or end point.

The selected block is executed next, regardless of the actual sequence. If a block is still active because execution was paused during a motion, the execution does not switch until this block has been completely executed.

Precondition

- The application is paused.
- The sequence chart in the display area is open.

Procedure

1. Touch the block that is to be jumped to. A dialog is opened. The block name is displayed in the header of the block selection:

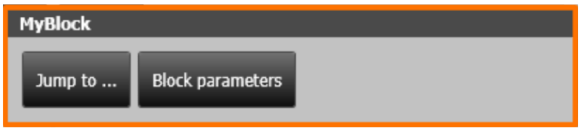


Fig. 7-2: Dialog

2. Press **Jump to ...**.
   The dialog is closed. Program execution switches to the desired block.

The switch to a sub-sequence block is only possible at the same level or to a higher level. The current level is indicated by the preview marking.
In the case of a switch to a sub-sequence block at the same level, execution jumps to the start point of the sub-sequence block:

- If the start point is not executed and a switch is made to a block, the sequence chart must be initialized.
- In the case of a switch to the end point, the sequence variables must not be initialized.

Example

While a block is being executed, an error occurs or a precondition is not met. In the case of a switch to a second block, the error is automatically deleted without the need to ignore the failed preconditions via a dialog window.

If there is a switch to a function block and back to the sub-sequence block while a sub-sequence block is being executed, execution of the sub-sequence block is started again at the start point.

NOTICE

Jumping to another block may lead to unexpected motions of the robot as a result of the associated change in the program execution.
If the effects are not known, the blocks must be tested in Manual Reduced Velocity mode (T1).

7.2.2 Displaying and modifying block parameters

Precondition

- The application is paused.
- The sequence chart in the display area is open.

Procedure

1. Touch the block. A dialog is opened. The block name is displayed in the header of the block selection:

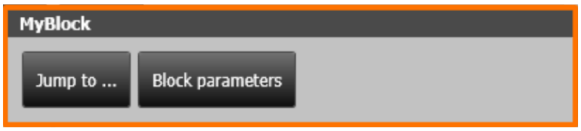


Fig. 7-3: Dialog

2. Press **Block parameters**. The **Block parameters** view opens. All the block parameters are displayed.
   (>>> *Fig. 7-4*)
3. To change a block parameter, touch the name of the block parameter.
4. Modify the value.
5. To apply the change, press **Save**.

Description

Fig. 7-4: “Change block parameters” view

Item	Description
1	Block parameter of the selected block
2	Details of the block parameters with the input box for changes
3	Opens a window with documentation about the selected block parameter.

- If the block parameter is linked to a constant (source) of data type Integer, Double or String, the value can be modified by entering letters or numbers.
- In the case of the data type Enum, the value can be selected from a list of the available values.
- In the case of the data type Boolean, the value can be modified by selecting **True**/**False**.
- No changes are possible if the block parameter is linked to another source.
- Block parameters with complex parameter types cannot be modified.

The block parameters modified on the smartHMI are saved and can be transferred to Sunrise.Workbench by means of project synchronization.

7.2.3 Displaying and modifying sequence variables

Precondition

- The application is running or paused.
- The sequence chart in the display area is open. The button for editing is displayed.

Procedure

1. Press the **Open sequence variables** button.
   (>>> *Fig. 7-1*)
2. The **Sequence variables** view opens. All the sequence variables of the selected sequence chart are displayed.
   (>>> *Fig. 7-5*)
3. To change a sequence variable, touch the name of the sequence variable.
4. Modify the value. Sequence variables with complex parameter types cannot be modified.
5. To apply the change, press **Save**.

Description

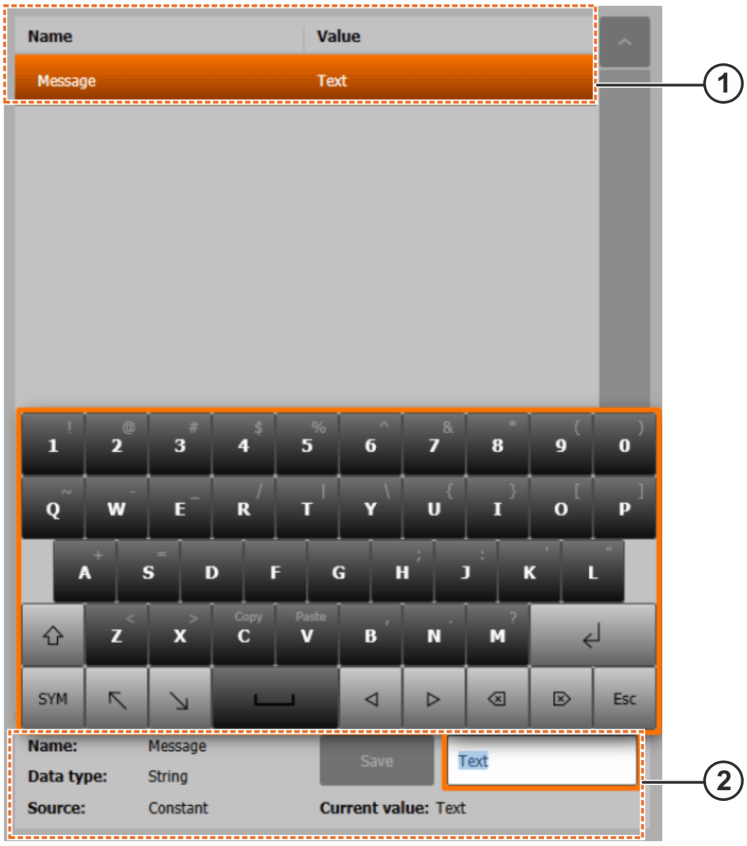


Fig. 7-5: “Change sequence variable” view

Item	Description
1	Sequence variables of the selected sequence chart
2	Details of the sequence variables with the input box for changes

- If the sequence variable is linked to a constant (source) of data type Integer, Double or String, the value can be modified by entering letters or numbers.
- In the case of the data type Boolean, the value can be modified by selecting **True**/**False**.
- If the sequence variable is linked to a complex parameter type, the data type is displayed as a string. This means that the method toString() is called.

The sequence variables modified on the smartHMI in a sub-sequence block or Java block apply to the current execution and are not saved.

7.2.4 Opening a sub-sequence

Description

Sub-sequence blocks can be opened in order to display the details inside a sequence chart.

Preconditions

- The application is selected.
- A sequence chart with a sub-sequence block is open.

Procedure

1. Touch the sub-sequence block. A dialog is opened. The block name is displayed in the header:

Fig. 7-6: Dialog

2. Press **Details**. The view with the sub-sequence plan opens.
3. In order to return to the higher-level sequence chart, press the button with the arrow icon in the button bar.

7.2.5 Selecting the program run mode

The application can be executed in continuous mode or in step mode. It is possible to switch from one mode to the other at any time.

(>>> *7.2 "Applications view with sequence chart" Page 32*)

Depending on the selected program run mode:

- Step mode

  In step mode, the program executes each block individually. The application is paused after each block. The Start key must be pressed again in order to execute the next block.
- Continuous mode

  In continuous mode, the application is executed without interruption. The headers of the blocks in which the corresponding program sections are currently being executed are automatically indicated one after the other.

7.3 Overview of runtime behavior

On execution of the application, the blocks in a sequence chart are successively executed beginning at the start point. The result of execution of a block determines the selection of the output and thus the next block in each case. The application ends as soon as an end point or a non-connected output is reached.

- Behavior before and during execution

  (>>> *7.3.1 "Behavior before and after execution" Page 39*)

- Behavior in the case of sub-sequences
  (>>> *7.3.2 "Behavior in the case of sub-sequences" Page 39*)
- Behavior during an application pause
  (>>> *7.3.3 "Behavior during an application pause" Page 39*)
- Behavior in the case of an error
  (>>> *7.3.4 "Behavior in the case of an error" Page 40*)
- Behavior in the case of non-assigned block parameters
  (>>> *7.3.5 "Behavior in the case of non-assigned block parameters" Page 40*)

7.3.1 Behavior before and after execution

Before execution of the application, the start block of the sequence chart is marked with a green triangle (preview marking).

After the start, the header of the start block is marked with a green background (execution marking).

In continued program execution, the header of the block currently being executed is always marked.

When an end point is reached, the execution is duly terminated and the end point is marked with a green background.

7.3.2 Behavior in the case of sub-sequences

On reaching a sub-sequence block, the header of the block is marked green. The contents of the sub-sequence block (e.g. other sub-sequences) are not displayed on the smartPAD. Irrespective of the program run mode selected, the header of the block is marked green until the sub-sequence block has been fully executed.

If a sequence variable of a sub-sequence block has a default value, the sequence variable is set as a block parameter. If the block parameter of the sub-sequence block becomes an output of a different block, the block parameter is overwritten with the default value of the sequence variable.

If a sub-sequence is open during program execution, only the headers of those blocks that are part of this sub-sequence are marked green. The view of the sub-sequence is not automatically quit even if the scroll function is activated.

(>>> *7.2.4 "Opening a sub-sequence" Page 38*)

7.3.3 Behavior during an application pause

If an application pause occurs, the execution marking of the current block is replaced with the preview marking:

- The current block contains one or more motion commands:
  Execution is paused before the next motion command. The current block is indicated by the preview marking until the application is resumed. Execution of the block is then continued.
- The current block does not contain a motion command:
  The current block is executed completely. Execution is paused before the next block. The next block is selected depending on the result of the execution and the connections of the current block. The next block is indicated by the preview marking.

7.3.4 Behavior in the case of an error

If a program error occurs during execution of a block, the application is paused. The block in which the error is triggered is indicated by the preview marking and the error symbol: 

Touching the indicated block opens a window with details on the error.

When the application is resumed, the block is executed again.

If an error occurs in a block within a sub-sequence, the corresponding status indicator is displayed on the sub-sequence block. By opening the sub-sequence block, it is possible to identify the block that has caused the error.

NOTICE

Before the error occurs, the block may already have carried out permanent changes that could result in consequential errors when the block is executed again.
If the effects are not known, the blocks must be tested in Manual Reduced Velocity mode (T1).

7.3.5 Behavior in the case of non-assigned block parameters

If a block parameter has not been assigned a value, the block is marked with the yellow status indicator. If a block without assigned block parameters is reached in the program sequence, the execution is paused before the block.

Block parameters can be modified on the smartPAD.

(>>> *7.2.2 "Displaying and modifying block parameters" Page 35*)

If a block parameter within a sub-sequence has not been assigned a value, the corresponding status indicator is displayed on the sub-sequence block. By opening the sub-sequence block, it is possible to identify the block that has caused the error. The missing block parameter can only be assigned in Sunrise Workbench and not on the smartPAD.

7.3.6 Behavior in the case of failed preconditions

Existing preconditions are checked before a block is executed. The application pauses as soon as a precondition fails. The block is additionally indicated by a red symbol for violated preconditions: 

Touching the indicated block opens the dialog with details on the violated precondition.

The preconditions are checked again when the program is resumed.

In order to execute a block despite non-fulfilled preconditions, activate the **Ignore preconditions** check box in the window with the details.

(>>> *7.3.7 "Ignoring failed preconditions" Page 40*)

If a precondition has failed in a block within a sub-sequence block, the corresponding status indicator is displayed on the sub-sequence block. By opening the sub-sequence block, it is possible to identify the block that has caused the error.

7.3.7 Ignoring failed preconditions

If the check of the preconditions of a block has failed, the block can be executed despite non-fulfilled preconditions. In this case, the preconditions

will be ignored. The block is additionally indicated by a yellow symbol:

Touching the indicated block opens a dialog:

- Check box not active: Failed preconditions are not ignored

  Failed preconditions prevent the block from being executed. The preconditions are checked again when the Start key is pressed. Program execution is resumed if the preconditions are fulfilled.
- Check box active: Failed preconditions are ignored

  The block can be executed despite failed preconditions. Program execution is resumed.

If failed preconditions are ignored, unexpected system behavior or unexpected robot motions may occur during execution of the block. This application should therefore only be used if the effects are known.
If the effects are not known, the blocks in which this condition has been activated must be tested in Manual Reduced Velocity mode (T1).

8 KUKA Service

8.1 Requesting support

Introduction

This documentation provides information on operation and operator control, and provides assistance with troubleshooting. For further assistance, please contact your local KUKA subsidiary.

Information

The following information is required for processing a support request:

- Description of the problem, including information about the duration and frequency of the fault
- As comprehensive information as possible about the hardware and software components of the overall system

  The following list gives an indication of the information which is relevant in many cases:

  – Model and serial number of the kinematic system, e.g. the manipulator
  – Model and serial number of the controller
  – Model and serial number of the energy supply system
  – Designation and version of the system software
  – Designations and versions of other software components or modifications
  – Diagnostic package KRCDiag

    Additionally for KUKA Sunrise: Existing projects including applications

    For versions of KUKA System Software older than V8: Archive of the software (KRCDiag is not yet available here.)
  – Application used
  – External axes used

8.2 KUKA Customer Support

Availability

KUKA Customer Support is available in many countries. Please do not hesitate to contact us if you have any questions.

Argentina

Ruben Costantini S.A. (Agency)
Luis Angel Huergo 13 20
Parque Industrial
2400 San Francisco (CBA)
Argentina
Tel. +54 3564 421033
Fax +54 3564 428877
ventas@costantini-sa.com

Australia

KUKA Robotics Australia Pty Ltd
45 Fennell Street
Port Melbourne VIC 3207
Australia
Tel. +61 3 9939 9656
info@kuka-robotics.com.au
www.kuka-robotics.com.au

Belgium

KUKA Automatisering + Robots N.V.
Centrum Zuid 1031
3530 Houthalen
Belgium
Tel. +32 11 516160
Fax +32 11 526794
info@kuka.be
www.kuka.be

Brazil

KUKA Roboter do Brasil Ltda.
Travessa Claudio Armando, nº 171
Bloco 5 - Galpões 51/52
Bairro Assunção
CEP 09861-7630 São Bernardo do Campo - SP
Brazil
Tel. +55 11 4942-8299
Fax +55 11 2201-7883
info@kuka-roboter.com.br
www.kuka-roboter.com.br

Chile

Robotec S.A. (Agency)
Santiago de Chile
Chile
Tel. +56 2 331-5951
Fax +56 2 331-5952
robotec@robotec.cl
www.robotec.cl

China

KUKA Robotics China Co., Ltd.
No. 889 Kungang Road
Xiaokunshan Town
Songjiang District
201614 Shanghai
P. R. China
Tel. +86 21 5707 2688
Fax +86 21 5707 2603
info@kuka-robotics.cn
www.kuka-robotics.com

Germany

KUKA Deutschland GmbH
Zugspitzstr. 140
86165 Augsburg
Germany
Tel. +49 821 797-1926
Fax +49 821 797-41 1926
Hotline.robotics.de@kuka.com
www.kuka.com

France

KUKA Automatisme + Robotique SAS
Techvallée
6, Avenue du Parc
91140 Villebon S/Yvette
France
Tel. +33 1 6931660-0
Fax +33 1 6931660-1
commercial@kuka.fr
www.kuka.fr

India

KUKA India Pvt. Ltd.
Office Number-7, German Centre,
Level 12, Building No. - 9B
DLF Cyber City Phase III
122 002 Gurgaon
Haryana
India
Tel. +91 124 4635774
Fax +91 124 4635773
info@kuka.in
www.kuka.in

Italy

KUKA Roboter Italia S.p.A.
Via Pavia 9/a - int.6
10098 Rivoli (TO)
Italy
Tel. +39 011 959-5013
Fax +39 011 959-5141
kuka@kuka.it
www.kuka.it

Japan

KUKA Japan K.K.
YBP Technical Center
134 Godo-cho, Hodogaya-ku
Yokohama, Kanagawa
240 0005
Japan
Tel. +81 45 744 7531
Fax +81 45 744 7541
info@kuka.co.jp

Canada

KUKA Robotics Canada Ltd.
6710 Maritz Drive - Unit 4
Mississauga
L5W 0A1
Ontario
Canada
Tel. +1 905 670-8600
Fax +1 905 670-8604
info@kukarobotics.com
www.kuka-robotics.com/canada

Korea

KUKA Robotics Korea Co. Ltd.
RIT Center 306, Gyeonggi Technopark
1271-11 Sa 3-dong, Sangnok-gu
Ansan City, Gyeonggi Do
426-901
Korea
Tel. +82 31 501-1451
Fax +82 31 501-1461
info@kukakorea.com

Malaysia

KUKA Robot Automation (M) Sdn Bhd
South East Asia Regional Office
No. 7, Jalan TPP 6/6
Taman Perindustrian Puchong
47100 Puchong
Selangor
Malaysia
Tel. +60 (03) 8063-1792
Fax +60 (03) 8060-7386
info@kuka.com.my

Mexico

KUKA de México S. de R.L. de C.V.
Progreso #8
Col. Centro Industrial Puente de Vigas
Tlalnepantla de Baz
54020 Estado de México
Mexico
Tel. +52 55 5203-8407
Fax +52 55 5203-8148
info@kuka.com.mx
www.kuka-robotics.com/mexico

Norway

KUKA Sveiseanlegg + Roboter
Sentrumsvegen 5
2867 Hov
Norway
Tel. +47 61 18 91 30
Fax +47 61 18 62 00
info@kuka.no

Austria

KUKA CEE GmbH
Gruberstraße 2-4
4020 Linz
Austria
Tel. +43 732 784 752 0
Fax +43 732 793 880
KUKAAustriaOffice@kuka.com
www.kuka.at

Poland

KUKA CEE GmbH Poland
Spółka z ograniczoną odpowiedzialnością
Oddział w Polsce
Ul. Porcelanowa 10
40-246 Katowice
Poland
Tel. +48 327 30 32 13 or -14
Fax +48 327 30 32 26
ServicePL@kuka-roboter.de

Portugal

KUKA Robots IBÉRICA, S.A.
Rua do Alto da Guerra n° 50
Armazém 04
2910 011 Setúbal
Portugal
Tel. +351 265 729 780
Fax +351 265 729 782
info.portugal@kukapt.com
www.kuka.com

Russia

KUKA Russia OOO
1-y Nagatinskiy pr-d, 2
117105 Moskau
Russia
Tel. +7 495 665-6241
support.robotics.ru@kuka.com

Sweden

KUKA Svetsanläggningar + Robotar AB
A. Odhners gata 15
421 30 Västra Frölunda
Sweden
Tel. +46 31 7266-200
Fax +46 31 7266-201
info@kuka.se

Switzerland

KUKA Roboter CEE GmbH
Linz, Zweigniederlassung Schweiz
Heinrich Wehrli-Strasse 27
5033 Buchs
Switzerland
Tel. +41 62 837 43 20
info@kuka-roboter.ch

Slovakia

KUKA CEE GmbH
organizačná zložka
Bojnická 3
831 04 Bratislava
Slovakia
Tel. +420 226 212 273
support.robotics.cz@kuka.com

Spain

KUKA Iberia, S.A.U.
Pol. Industrial
Torrent de la Pastera
Carrer del Bages s/n
08800 Vilanova i la Geltrú (Barcelona)
Spain
Tel. +34 93 8142-353
comercial@kukarob.es

South Africa

Jendamark Automation LTD (Agency)
76a York Road
North End
6000 Port Elizabeth
South Africa
Tel. +27 41 391 4700
Fax +27 41 373 3869
www.jendamark.co.za

Taiwan

KUKA Automation Taiwan Co. Ltd.
1F, No. 298 Yangguang ST.,
Nei Hu Dist., Taipei City, Taiwan 114
Taiwan
Tel. +886 2 8978 1188
Fax +886 2 8797 5118
info@kuka.com.tw

Thailand

KUKA (Thailand) Co. Ltd.
No 22/11-12 H-Cape Biz Sector Onnut
Sukhaphiban 2 road, Prawet
Bangkok 10250
Thailand
Tel. +66 (0) 90-940-8950
HelpdeskTH@kuka.com

Czech Republic

KUKA Roboter CEE GmbH
organizační složka
Pražská 239
25066 Zdiby
Czech Republic
Tel. +420 226 212 273
support.robotics.cz@kuka.com

Hungary

KUKA HUNGÁRIA Kft.
Fö út 140
2335 Taksony
Hungary
Tel. +36 24 501609
Fax +36 24 477031
info@kuka-robotics.hu

USA

KUKA Robotics Corporation
51870 Shelby Parkway
Shelby Township
48315-1787
Michigan
USA
Tel. +1 866 873-5852
Fax +1 866 329-5852
info@kukarobotics.com
www.kukarobotics.com

UK

KUKA Robotics UK Ltd
Great Western Street
Wednesbury West Midlands
WS10 7LL
UK
Tel. +44 121 505 9970
Fax +44 121 505 6589
service@kuka-robotics.co.uk
www.kuka-robotics.co.uk

Index

R

S

T

U

V

W