Controlling a KUKA Industrial Robot Using a SIMATIC S7-1500

SIMATIC S7-1500 / TIA Portal V13 SP1 / KUKA.PLC mxAutomation

Warranty and Liability

Note

The Application Examples are not binding and do not claim to be complete regarding the circuits shown, equipping and any eventuality. The Application Examples do not represent customer-specific solutions. They are only intended to provide support for typical applications. You are responsible for ensuring that the described products are used correctly. These Application Examples do not relieve you of the responsibility to use safe practices in application, installation, operation and maintenance. When using these Application Examples, you recognize that we cannot be made liable for any damage/claims beyond the liability clause described. We reserve the right to make changes to these Application Examples at any time without prior notice. If there are any deviations between the recommendations provided in this Application Example and other Siemens publications – e.g. Catalogs – the contents of the other documents shall have priority.

We do not accept any liability for the information contained in this document. Any claims against us – based on whatever legal reason – resulting from the use of the examples, information, programs, engineering and performance data etc., described in this application example shall be excluded. Such an exclusion shall not apply in the case of mandatory liability, e.g. under the German Product Liability Act (“Produkthaftungsgesetz”), in case of intent, gross negligence, or injury of life, body or health, guarantee for the quality of a product, fraudulent concealment of a deficiency or breach of fundamental contractual obligations (“wesentliche Vertragspflichten”). The damages for a breach of a substantial contractual obligation are, however, limited to the foreseeable damage, typical for the type of contract, except in the event of intent or gross negligence or injury to life, body or health. The above provisions do not imply a change of the burden of proof to your detriment.

Any form of duplication or distribution of these Application Examples or excerpts hereof is prohibited without the expressed consent of Siemens AG.

Security information

Siemens provides products and solutions with industrial security functions that support the secure operation of plants, solutions, machines, equipment and/or networks. They are important components in a holistic industrial security concept. With this in mind, Siemens’ products and solutions undergo continuous development. Siemens recommends strongly that you regularly check for product updates.

For the secure operation of Siemens products and solutions, it is necessary to take suitable preventive action (e.g. cell protection concept) and integrate each component into a holistic, state-of-the-art industrial security concept. Third-party products that may be in use should also be considered. For more information about industrial security, visit http://www.siemens.com/industrialsecurity.

To stay informed about product updates as they occur, sign up for a product-specific newsletter. For more information, visit http://support.industry.siemens.com.
# Table of Contents

Warranty and Liability ........................................................................................................... 2

1 Task.................................................................................................................................. 5
  1.1 Introduction .................................................................................................................. 5
  1.2 Requirements ................................................................................................................ 5

2 Solution............................................................................................................................... 6
  2.1 Overview ....................................................................................................................... 6
  2.2 Available core functionality ......................................................................................... 7
  2.3 Hardware and software components .......................................................................... 8
  2.3.1 Validity .................................................................................................................... 8
  2.3.2 Components used .................................................................................................... 8

3 Basics ................................................................................................................................ 10
  3.1 Design of a KUKA industrial robot .............................................................................. 10
  3.2 KUKA.PLC mxAutomation ......................................................................................... 11
  3.2.1 Overview ................................................................................................................ 11
  3.2.2 KUKA.PLC mxAutomation block library .............................................................. 12
  3.2.3 Interpreter on the KUKA robot controller ............................................................ 12
  3.3 Block behavior according to the PLCopen standard .................................................. 12
  3.3.1 Inputs and outputs of the blocks .......................................................................... 12
  3.3.2 Chained execution of library blocks .................................................................... 13

4 Program Generation ........................................................................................................... 15
  4.1 Connecting the KUKA robot to the SIMATIC S7 ....................................................... 15
  4.1.1 Robot as a PROFINET IO device ........................................................................ 15
  4.1.2 Installing the GSDML file of the robot ................................................................. 15
  4.1.3 Integrating the robot into the hardware configuration ........................................ 15
  4.1.4 Connecting the SIMATIC S7 to the robot ............................................................ 17
  4.1.5 Controlling multiple robots using one SIMATIC S7 ........................................... 17
  4.2 Importing the block library ......................................................................................... 18
  4.2.1 Extracting the block library .................................................................................. 18
  4.2.2 Opening the block library ..................................................................................... 18
  4.2.3 Transferring blocks to the user program .............................................................. 18
  4.3 Basic program structure ............................................................................................. 19
  4.3.1 Concept .................................................................................................................. 19
  4.3.2 Structure of the program ....................................................................................... 20
  4.3.3 Read data from robot ............................................................................................ 20
  4.3.4 Process data in SIMATIC S7 .............................................................................. 21
  4.3.5 Write data to robot ............................................................................................... 21
  4.4 Programming basic functions ..................................................................................... 21
  4.4.1 Switching on the robot ......................................................................................... 21
  4.4.2 Initializing the robot ............................................................................................... 23
  4.4.3 Checking the robot status ..................................................................................... 24
  4.4.4 Acknowledging error messages ........................................................................... 27
  4.4.5 Reading out the robot position ............................................................................. 28
  4.4.6 Setting the velocity override .............................................................................. 30
  4.5 Programming motion functions .................................................................................. 30
  4.5.1 Motion types ......................................................................................................... 30
  4.5.2 Coordinate definition ............................................................................................ 31
  4.5.3 Point-to-point motion (PTP) ............................................................................... 32
  4.5.4 Linear motion ....................................................................................................... 33
  4.5.5 Circular motion ..................................................................................................... 34
  4.5.6 Approximate positioning motion function ......................................................... 35
  4.5.7 Orientation control ............................................................................................. 36
  4.5.8 Manually moving robot axes (JOG) ................................................................. 36
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.6</td>
<td>Summary program example</td>
<td>38</td>
</tr>
<tr>
<td>4.6.1</td>
<td>Complete program overview</td>
<td>38</td>
</tr>
<tr>
<td>4.6.2</td>
<td>Robot control detail view</td>
<td>39</td>
</tr>
<tr>
<td>4.6.3</td>
<td>Motion control detail view</td>
<td>40</td>
</tr>
<tr>
<td>5</td>
<td>Additional Functions</td>
<td>42</td>
</tr>
<tr>
<td>5.1</td>
<td>Tool control (gripper)</td>
<td>42</td>
</tr>
<tr>
<td>5.1.1</td>
<td>Control using the robot controller</td>
<td>42</td>
</tr>
<tr>
<td>5.1.2</td>
<td>Direct control using the SIMATIC S7</td>
<td>42</td>
</tr>
<tr>
<td>5.2</td>
<td>Safety Integrated</td>
<td>43</td>
</tr>
<tr>
<td>5.2.1</td>
<td>Components</td>
<td>43</td>
</tr>
<tr>
<td>5.2.2</td>
<td>Functionality</td>
<td>43</td>
</tr>
<tr>
<td>5.2.3</td>
<td>Control</td>
<td>44</td>
</tr>
<tr>
<td>5.3</td>
<td>PROFIenergy</td>
<td>44</td>
</tr>
<tr>
<td>5.4</td>
<td>Receiving robot messages in plain text</td>
<td>45</td>
</tr>
<tr>
<td>5.4.1</td>
<td>Principle of operation</td>
<td>45</td>
</tr>
<tr>
<td>5.4.2</td>
<td>Programming</td>
<td>46</td>
</tr>
<tr>
<td>6</td>
<td>Appendix</td>
<td>48</td>
</tr>
<tr>
<td>6.1</td>
<td>Coordinate systems</td>
<td>48</td>
</tr>
<tr>
<td>6.1.1</td>
<td>Overview</td>
<td>48</td>
</tr>
<tr>
<td>6.1.2</td>
<td>Original coordinate system of the system (WORLD)</td>
<td>48</td>
</tr>
<tr>
<td>6.1.3</td>
<td>Robot base coordinate system (ROBROOT)</td>
<td>49</td>
</tr>
<tr>
<td>6.1.4</td>
<td>Workpiece coordinate system (BASE)</td>
<td>49</td>
</tr>
<tr>
<td>6.1.5</td>
<td>Tool coordinate system (TOOL)</td>
<td>50</td>
</tr>
<tr>
<td>6.2</td>
<td>Coordinate definition</td>
<td>50</td>
</tr>
<tr>
<td>6.2.1</td>
<td>Robot axes</td>
<td>50</td>
</tr>
<tr>
<td>6.2.2</td>
<td>Axis coordinates</td>
<td>51</td>
</tr>
<tr>
<td>6.2.3</td>
<td>Cartesian coordinates</td>
<td>51</td>
</tr>
<tr>
<td>6.2.4</td>
<td>Additional data: “Status” and “Turn”</td>
<td>51</td>
</tr>
<tr>
<td>6.2.5</td>
<td>Singularities</td>
<td>54</td>
</tr>
<tr>
<td>6.2.6</td>
<td>Tool center point (TCP)</td>
<td>54</td>
</tr>
<tr>
<td>7</td>
<td>Links &amp; Literature</td>
<td>55</td>
</tr>
<tr>
<td>8</td>
<td>Contact</td>
<td>56</td>
</tr>
<tr>
<td>8.1</td>
<td>Siemens Robotics Support</td>
<td>56</td>
</tr>
<tr>
<td>8.2</td>
<td>KUKA Roboter GmbH</td>
<td>56</td>
</tr>
<tr>
<td>8.3</td>
<td>KUKA Roboter GmbH Hotline</td>
<td>57</td>
</tr>
<tr>
<td>8.4</td>
<td>KUKA,PLC mxAutomation Sales</td>
<td>57</td>
</tr>
<tr>
<td>9</td>
<td>History</td>
<td>57</td>
</tr>
</tbody>
</table>
1 Task

1.1 Introduction

Industrial robots are becoming more and more popular. Nowadays, they are increasingly used in machines and plants. Their standardized mechanical system is fully developed and highly flexible in terms of the possible motions so that a robot has become the preferred choice over expensive special mechanical equipment. This allows production from lot size one upward even without expensive modifications to machines and plants.

Unfortunately, however, the plant control system and the robot controller are typically two different systems. Communication between the two controllers takes place mostly only at the bit level and the robot’s motion programs are permanently stored on the robot controller and can only be called by the plant control system. Therefore, it is relatively difficult to implement a flexible response of the robot to special plant events.

Furthermore, the programming of the plant control system and the one of the robot mostly differ to a great extent so that both systems can generally not be supported by one person. Interface and coordination problems are therefore inevitable.

1.2 Requirements

The aim is to fully integrate the robot’s control and motion control into the machine and plant control system to make the use of an industrial robot in a production plant easier and more flexible.

The following requirements apply to the automation task:

- The robot should be fully programmable and controllable using the machine and plant control system (PLC).
- The robot can be operated using the same HMI of the PLC or machine (single point of operation).
- Robot diagnostics should be possible entirely using the PLC.
- It should be possible to integrate related functions such as Safety Integrated and PROFlenergy and control them using the PLC.
2 Solution

2.1 Overview

The aim is to use a SIMATIC S7-1500 controller to fully control and operate a KUKA industrial robot. To do this, the KUKA.PLC mxAutomation block library is used in TIA Portal, which provides all the necessary function blocks.

Communication between the SIMATIC S7-1500 controller and the KUKA industrial robot takes place via a PROFINET connection. All commands and status information between the SIMATIC controller and the robot are exchanged via this connection.

**Note**

Using a SIMATIC S7-1500 controller, up to five KUKA industrial robots can be simultaneously controlled with the aid of the KUKA.PLC mxAutomation block library in TIA Portal.

However, this application example confines itself to connecting only one robot to a SIMATIC S7-1500.

**Diagrammatic representation**

The diagrammatic representation below shows the most important components of the solution:

Figure 2-1 Schematic overview of the application example

Using the KUKA.PLC mxAutomation block library, the SIMATIC controller takes full control of the KUKA industrial robot.

The KUKA industrial robot consists of the KUKA KR C4 robot controller and the robot's mechanical system, the actual robot. The interpreter for the commands of the KUKA.PLC mxAutomation block library is installed on the robot controller. The interpreter receives the commands from the SIMATIC controller and executes them on the robot’s mechanical system, including kinematic transformation.
2 Solution

2.2 Available core functionality

Advantages

Controlling a KUKA industrial robot using SIMATIC S7-1500 controller with the aid of the KUKA.PLC mxAutomation block library in TIA Portal provides the following advantages:

- Only one PROFINET connection is required to connect the robot to the SIMATIC controller.
- Easy integration of the robot in TIA Portal in the SIMATIC controller hardware configuration using a GSDML file.
- Option to fully operate the most important robot functions using the SIMATIC controller by means of the commands of the KUKA.PLC mxAutomation library.
- Option to instruct event-controlled robot motions depending on the state of the sensors connected to the SIMATIC controller.
- Operation and diagnostics of the machine or plant and the robot using a common HMI.

Scope

The application example described here presents an example of using the KUKA.PLC mxAutomation block library in TIA Portal and shows which robot functions can be controlled with the aid of the function blocks using the SIMATIC controller.

The aim of the application example is to familiarize you with the basic functions and options of the KUKA.PLC mxAutomation block library in TIA Portal and provide support in decision making and planning your own projects and user programs with a KUKA industrial robot.

For a detailed description of the functions and use of the KUKA.PLC mxAutomation block library, please refer to the KUKA documentation for the KUKA.PLC mxAutomation library listed in the "Links & Literature" chapter of this documentation or available on the KUKA Roboter GmbH website.

Required knowledge

This application example does not convey but require basic knowledge of creating a user program on the SIMATIC S7-1500 in TIA Portal or setting up the hardware configuration.

Furthermore, this application example is not an introduction to robotics. Basic knowledge of the use and options of an industrial robot is also required.

2.2 Available core functionality

Here is an overview of the functions provided by the KUKA.PLC mxAutomation block library:

- **Administrative functions:**
  Switch on robot, check current robot status, read/write tool data and base data, use of robot controller peripherals, read/write limit switch data.

- **Motion programming:**
  Move robot kinematics in straight lines, circular arcs or velocity-optimized trajectories. Move robot axes in jog mode.

- **Program execution control:**
  Interrupt, continue or cancel robot programs permanently stored on the robot controller.
2.3 Hardware and software components

- **Interrupt programming:**
  Define, activate/deactivate interrupts on the robot controller and check the current status.

- **Path-related switching actions:**
  Define and activate trigger conditions for path- or distance-related switching functions.

- **Diagnostics functions:**
  Read out error messages, diagnostics signals and robot status.

- **Special functions:**
  Teach and store control points in the robot controller, perform break test and read/write system values.

**NOTICE**

Use of the KUKA SafeOperation safety option when operating the robot via a common HMI without using the KUKA smartPAD control panel.

When controlling a KUKA industrial robot using a SIMATIC S7-1500 controller with the aid of the KUKA.PLC mxAutomation block library, it should be considered to additionally use the KUKA SafeOperation safety option. However, if you want to move the robot in JOG mode via a common HMI without using the KUKA smartPAD control panel, it is mandatory to use the KUKA SafeOperation safety option.

For related information on the KUKA SafeOperation safety option, please refer to chapter 5.2 of this documentation or call the KUKA Roboter GmbH hotline.

### 2.3.1 Validity

This application is valid for:

- STEP 7 TIA Portal V13 SP1 or higher
- S7-1500 firmware version V1.5 or higher
- KUKA.PLC mxAutomation block library version V2.0.2 or higher

### 2.3.2 Components used

This application was created with the following components:

#### Hardware components

<table>
<thead>
<tr>
<th>Component</th>
<th>No.</th>
<th>Article number</th>
<th>Note</th>
</tr>
</thead>
<tbody>
<tr>
<td>SIMATIC CPU S7-1516</td>
<td>1</td>
<td>6ES7 516-3AN00-0AB0</td>
<td>Firmware version: V1.5</td>
</tr>
</tbody>
</table>
2 Solution

2.3 Hardware and software components

Table 2-2  KUKA components

<table>
<thead>
<tr>
<th>Component</th>
<th>No.</th>
<th>Article number</th>
<th>Note</th>
</tr>
</thead>
<tbody>
<tr>
<td>KR C4 compact with firmware V8.3</td>
<td>1</td>
<td></td>
<td>Robot controller</td>
</tr>
<tr>
<td>KR 6 R900 SIHX (KR AGILUS)</td>
<td>1</td>
<td></td>
<td>Robot</td>
</tr>
</tbody>
</table>

Software components

Table 2-3  SIEMENS components

<table>
<thead>
<tr>
<th>Component</th>
<th>No.</th>
<th>Article number</th>
<th>Note</th>
</tr>
</thead>
<tbody>
<tr>
<td>TIA Portal V13</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>STEP 7 Professional</td>
<td>1</td>
<td>6ES7 822-1AE03-0YA5</td>
<td>Version: V13 SP1</td>
</tr>
</tbody>
</table>

Table 2-4  KUKA components

<table>
<thead>
<tr>
<th>Component</th>
<th>No.</th>
<th>Article number</th>
<th>Note</th>
</tr>
</thead>
<tbody>
<tr>
<td>KUKA,PLC mxAutomation</td>
<td>1</td>
<td>00-233-923</td>
<td>Version: V2.0.2</td>
</tr>
<tr>
<td>KUKA,Profinet Device</td>
<td>1</td>
<td>00-244-976</td>
<td>Version: V3.2 Allows you to operate the robot as a PROFINET device.</td>
</tr>
<tr>
<td>Or alternatively: KUKA,Profinet ControllerDevice</td>
<td>1</td>
<td>00-244-984</td>
<td>Version: V3.2 Alternative to KUKA,Profinet Device: Allows you to operate the robot as a PROFINET device or as a controller.</td>
</tr>
<tr>
<td>KUKA GSD file for KR C4 with firmware V8.3</td>
<td>1</td>
<td></td>
<td>Version: V2.25 Date created: 08/08/13</td>
</tr>
</tbody>
</table>

Sample files and projects

The following list contains all files and projects that are used in this example.

Table 2-5

<table>
<thead>
<tr>
<th>Component</th>
<th>Note</th>
</tr>
</thead>
<tbody>
<tr>
<td>109482123_S7-1500_KUKA_mxAutomation_DOKU_v10_en.pdf</td>
<td>This document.</td>
</tr>
</tbody>
</table>
3 Basics

The aim of this chapter is to describe the basic functions and provide background information for using a KUKA industrial robot in conjunction with the KUKA.PLC mxAutomation block library.

3.1 Design of a KUKA industrial robot

A KUKA industrial robot generally consists of the following components.

Figure 3-1 Design of a KUKA industrial robot

Table 3-1 Design of a KUKA industrial robot

<table>
<thead>
<tr>
<th>No.</th>
<th>Component</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Manipulator</td>
<td>The manipulator is the actual mechanical system of the robot, i.e., the kinematics that executes the commands.</td>
</tr>
<tr>
<td>2</td>
<td>smartPAD control panel</td>
<td>The smartPAD control panel allows you to make and check settings on the robot controller. Likewise, the control panel can be used to move the robot manually and automatically.</td>
</tr>
<tr>
<td>3</td>
<td>Connecting cable/smartPAD</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>Robot controller</td>
<td>The robot controller coordinates the robot motions. This controller calculates the coordinate transformation for the robot motions and controls the robot's axis motors. The robot controller can also contain the power units for the robot's axis motors.</td>
</tr>
<tr>
<td>5</td>
<td>Connecting cable/data cable</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>Connecting cable/motor cable</td>
<td></td>
</tr>
</tbody>
</table>
3.2 KUKA.PLC mxAutomation

3.2.1 Overview

The following figure provides a functional overview of the KUKA.PLC mxAutomation function package.

Figure 3-2 Overview of the functions provided by KUKA.PLC mxAutomation
The KUKA.PLC mxAutomation option package for a KUKA industrial robot consists of two parts:

- A block library for controlling a KUKA industrial robot from a SIMATIC controller.
- An interpreter on the robot controller that interprets the commands of the function blocks from the SIMATIC controller and passes them on to the robot's path planning.

### 3.2.2 KUKA.PLC mxAutomation block library

The KUKA.PLC mxAutomation block library provides different blocks for controlling a KUKA industrial robot. The desired functions of the KUKA robot can be selected by simply calling the appropriate block from the block library.

The call of a function block from the block library transfers the respective commands to the KUKA robot controller, where they are interpreted.

### 3.2.3 Interpreter on the KUKA robot controller

The interpreter on the KUKA robot controller receives the commands of the function blocks from the KUKA.PLC mxAutomation block library in the robot controller and stores them in a buffer memory. At the appropriate time, the commands are then transferred from the buffer memory to the robot's path planning so that the robot executes the functions instructed using the SIMATIC controller.

### 3.3 Block behavior according to the PLCopen standard

The behavior of the function blocks of the KUKA.PLC mxAutomation block library follows the PLCopen standard.

#### 3.3.1 Inputs and outputs of the blocks

The blocks of the KUKA.PLC mxAutomation block library always feature the following inputs and outputs:

![Example of a block of the KUKA.PLC mxAutomation block library](image-url)
### 3.3 Block behavior according to the PLCopen standard

#### Table 3-2 Block interfaces following the PLCopen standard

<table>
<thead>
<tr>
<th>Interface</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Inputs</strong></td>
<td></td>
</tr>
<tr>
<td>AxisGroupIdx</td>
<td>Selects the robot to be controlled.</td>
</tr>
<tr>
<td>ExecuteCmd</td>
<td>Starts the function via a rising edge at the block’s input.</td>
</tr>
<tr>
<td><strong>Outputs</strong></td>
<td></td>
</tr>
<tr>
<td>Busy</td>
<td>The robot function instructed via the function block was transferred to</td>
</tr>
<tr>
<td></td>
<td>the robot controller.</td>
</tr>
<tr>
<td>Active</td>
<td>The motion instructed via the function block is currently being executed</td>
</tr>
<tr>
<td></td>
<td>by the robot.</td>
</tr>
<tr>
<td>Done</td>
<td>Execution of the motion by the robot has successfully completed.</td>
</tr>
<tr>
<td>Aborted</td>
<td>Execution of the function block or the motion was aborted.</td>
</tr>
<tr>
<td>Error</td>
<td>An error has occurred while executing the function block.</td>
</tr>
<tr>
<td>ErrorID</td>
<td>Error code for exact specification of the error cause.</td>
</tr>
</tbody>
</table>

#### 3.3.2 Chained execution of library blocks

With a rising edge at the ExecuteCmd input, the selected function block transfers the command to execute the desired robot function to the robot controller. Via the Busy output of the function block, this transfer is returned to the user program.

If multiple function blocks of the block library are to be executed one after the other, they can be transferred to the robot controller in quick succession. Each time a function block started by a rising edge sets the Busy output, the command was transferred to the robot controller and the next function block can be started.

If the Busy output of the started function block is not set, either the buffer memory on the robot controller is full and the user program must wait until the robot has processed commands or processing the function block was aborted due to an error or due to the start of another function block.

Errors that have occurred are returned to the user program by the function blocks via the Error output. Aborts or replacing block processing by the start of another function block are signaled via the Aborted output.

#### Example of the chained call of function blocks

The following figure shows an example of the call of four motion functions via the function blocks of the block library, for example the KRC_MoveLinearAbsolute function block, that are buffered in the robot controller and executed one after the other.
3 Basics

3.3 Block behavior according to the PLCopen standard

Figure 3-4 Signal sequence for the chained call of function blocks

Table 3-3 Explanation of the signal sequence

<table>
<thead>
<tr>
<th>No.</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>A rising edge at the Execute input of the first motion function starts the motional sequence.</td>
</tr>
<tr>
<td>2</td>
<td>Once the motion function signals Busy, the next motion function can be started with a rising edge at the Execute input.</td>
</tr>
<tr>
<td>3</td>
<td>Due to the buffering of the motion function in the robot controller, the motion is activated when the previous motion function has completed.</td>
</tr>
<tr>
<td>4</td>
<td>If the motion function signals Done, the motion has completed and the Execute input can be reset.</td>
</tr>
<tr>
<td>5</td>
<td>If the job buffer in the robot controller cannot accept any more jobs, the Busy signal will be delayed until new jobs can be transferred to the robot controller.</td>
</tr>
<tr>
<td>6</td>
<td>With the Done signal of the last motion function of the sequence, the motional sequence has been completely processed.</td>
</tr>
</tbody>
</table>
4 Program Generation

The aim of this chapter is to provide you with a basic explanation of how to create a user program for controlling a KUKA robot using the KUKA.PLC mxAutomation library.

Note
For more detailed information on the function and parameters of the blocks of the KUKA.PLC mxAutomation library, please refer to the appropriate KUKA documentation.

4.1 Connecting the KUKA robot to the SIMATIC S7

4.1.1 Robot as a PROFINET IO device

The KUKA robot is integrated into the hardware configuration of the TIA Portal project as a PROFINET IO device. The robot is integrated via a GSDML file to be additionally integrated into TIA Portal that contains the robot’s hardware description and the possible data telegrams for data exchange between the SIMATIC controller and the robot.

4.1.2 Installing the GSDML file of the robot

The GSDML file of the robot is generally supplied with the robot or robot documentation or can be obtained from the KUKA service department. The GSDML file can be integrated into TIA Portal using the management function for GSD files.

Figure 4-1  Importing the GSDML file

4.1.3 Integrating the robot into the hardware configuration

When the GSDML file has been integrated into TIA Portal, the KUKA KR C4 robot controller with the appropriate firmware version can be integrated into the TIA Portal hardware configuration using drag and drop.

Double-clicking the robot controller in the hardware configuration allows you to configure the components for data exchange with the SIMATIC controller.

To be able to exchange the data necessary to control the robot using the SIMATIC controller, configure the required data exchange telegram with the robot controller in the hardware configuration. To do this, use drag and drop to integrate at least the telegram for 256 digital inputs and outputs from the telegram selection into the robot controller component of the hardware configuration.
4 Program Generation

4.1 Connecting the KUKA robot to the SIMATIC S7

Figure 4-2  Selecting the KUKA robot from the hardware catalog

Figure 4-3  Data exchange between the SIMATIC controller and the robot

Figure 4-4  Data telegram

Note

Optionally, additional telegrams or telegram extensions for optional functions such as Safety Integrated can be integrated into the robot controller at this point.

In the TIA Portal hardware configuration, use drag and drop to integrate the additional telegrams into the robot controller.
4.1.4 Connecting the SIMATIC S7 to the robot

Finally, use a PROFINET connection to connect the robot controller to the SIMATIC controller. To do this, assign suitable IP addresses on the individual devices or in the hardware configuration.

Optionally, an HMI device used to control and monitor the robot functions can be integrated into the configuration at this point.

Figure 4-5 Connecting the robot, SIMATIC controller and HMI

4.1.5 Controlling multiple robots using one SIMATIC S7

Using a suitable SIMATIC controller, up to five KUKA robots can be controlled separately from each other in the standard configuration with the aid of the KUKA.PLC mxAutomation block library.

For this purpose, the data block supplied with the KUKA.PLC mxAutomation block library provides five data arrays for data exchange between the SIMATIC controller and the robot.

The assignment of the data to the appropriate robot is made via the defined telegram addresses in the IO range of the SIMATIC controller as defined in the hardware configuration.

The assignment of the function blocks or robot functions to the individual robots in the user program is made via the AxisGroupIdx input of the blocks that represents the index of the data array in the MxADBRobots data block.

However, this functionality will be explained in detail in the following chapters.

Figure 4-6 Data pool for 5 robots in the MxADBRobots data block
4.2 Importing the block library

4.2.1 Extracting the block library

Before you can use the KUKA.PLC mxAutomation block library in your user program, you must apply the function blocks of the block library to your TIA Portal project.

The KUKA.PLC mxAutomation block library for TIA Portal is supplied as a global library from which all function blocks of the block library can be applied to a TIA Portal project using drag and drop.

To be able to use the global library for TIA Portal, please extract the global library archive to a suitable location on the hard drive of your programmer.

Note

The KUKA.PLC mxAutomation block library for TIA Portal can be requested from KUKA Roboter GmbH Service or the company’s hotline.

4.2.2 Opening the block library

After extracting, you can use the TIA Portal access functions to open the global library.

Figure 4-7 Opening the block library

The KUKA.PLC mxAutomation block library is stored in the global library as a master copy.

4.2.3 Transferring blocks to the user program

Simply use drag and drop to transfer the KUKA.PLC mxAutomation block library from the global library to your user program. Transfer both the necessary PLC data types and the function blocks to your TIA Portal project.

- First drag the object included in the master copies of the global library in PLC data types to the PLC data types folder in your project. In this folder, a subfolder is automatically created that contains all the PLC data types of the KUKA.PLC mxAutomation block library.
- Then drag the object included in Program blocks to the Program blocks folder of your TIA Portal project. In this folder, too, a subfolder is automatically created that contains all the function blocks of the KUKA.PLC mxAutomation block library.
Now your TIA Portal project includes all the necessary data types and function blocks of the KUKA.PLC mxAutomation block library. Now you can use the block library functions in your user program.

4.3 Basic program structure

4.3.1 Concept

Conceptually, data exchange between the SIMATIC controller and the KUKA robot with the aid of the KUKA.PLC mxAutomation block library follows the functionality of the SIMATIC controller. However, this concept must be implemented in the user program of the SIMATIC controller using the appropriate library blocks.
In the organization block where the robot program is to be processed, this requires that the read robot data from robot controller function be called as the first block. As the last function in the organization block, place the write calculated data to robot controller function. Between these two blocks, you can place the function blocks to implement the actual robot program in the SIMATIC controller.

For the call of the robot program in the SIMATIC controller, using the standard organization block (Main Cycle or OB 1) is generally sufficient.

### 4.3.2 Structure of the program

To structure a user program for controlling a KUKA industrial robot using the KUKA.PLC mxAutomation block library, we can give the following recommendation:

- For each robot to be controlled using the SIMATIC controller, a separate function block should be used that is called in the Program Cycle OB (OB 1).
- Within the function block of a robot, the block for reading the robot data should be called in the first network and the block for writing the robot data should be called in the last network.
- The calls – between the above calls – of the other function blocks of the KUKA.PLC mxAutomation block library can then also be divided into other function blocks structured by function.
- All calls of blocks from the KUKA.PLC mxAutomation library should be implemented as multi-instance calls in the calling blocks.

### 4.3.3 Read data from robot

The KRC_ReadAxisGroup (FB 953) block from the KUKA.PLC mxAutomation library allows you to read the data from the robot controller into the KUKA.PLC mxAutomation block library’s internal data management.

Figure 4-10  Read data from robot

This provides the data of the robot or robot controller to the user program in the SIMATIC controller for the other blocks from the KUKA.PLC mxAutomation library. The assignment of the data to the appropriate robot connected to the SIMATIC controller is made via the AxisGroupIdx input that corresponds to the index of the data array in the MxADBRobots data block and the InputStartByte input that informs the block of the robot's telegram address from the IO range of the SIMATIC controller.

Then solely the index of the AxisGroupIdx input can be used to assign the robot to the program blocks of the robot program in the SIMATIC.
4.3.4 Process data in SIMATIC S7

After reading in the data from the robot controller, all the function blocks from the KUKA.PLC mxAutomation library can be used and called. The blocks use the read-in data stored in the MxADBRobots data block.

4.3.5 Write data to robot

After processing the robot program in the SIMATIC controller, the KRC_WriteAxisGroup (FB 954) block from the KUKA.PLC mxAutomation library transfers the data from the MxADBRobots data block to the robot defined by the telegram address.

Figure 4-11 Write data to robot

NOTICE
When entering the robot’s telegram addresses for read/write, make sure that the assignment is correct. Otherwise, unexpected behavior of the robot may occur.

4.4 Programming basic functions

4.4.1 Switching on the robot

The robot is switched on using the KRC_AutoExt block. This requires that the inputs of the block be operated in a specified signal sequence and depending on the block outputs.

Note
To be able to operate the robot using the KUKA.PLC mxAutomation block library, the robot controller must be in EXT mode. It may be necessary to first select this mode using the keyswitch of the KUKA SmartPad.
Switching on the robot using the KRC_AutoExt block is implemented via the following signal sequence.

**Figure 4-12** Switching on the robot using the KRC_AutoExt function block

**Figure 4-13** Standard signal sequence for switching on the robot
### 4.4 Programming basic functions

#### 4.4.1 Explanation of the signal sequence

<table>
<thead>
<tr>
<th>No.</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>To be able to operate the robot using the KUKA.PLC mxAutomation block library, EXT mode must first be selected on the KRC_AutoExt block. This is achieved by setting the ENABLE_EXT signal on the function block. <strong>Note:</strong> Switching to EXT mode is only possible if this mode is selected on the robot.</td>
</tr>
<tr>
<td>2</td>
<td>Then set the MOVE_ENABLE and DRIVES_OFF input signals of the block. The robot responds to these signals by setting the ALARM_STOP output signal.</td>
</tr>
<tr>
<td>3</td>
<td>Then set the DRIVES_ON input signal. The robot responds to it by indicating periphery ready via the PERI_RDY output signal. Once the periphery is ready, you can reset the DRIVES_ON input signals on the block.</td>
</tr>
<tr>
<td>4</td>
<td>Then acknowledge any pending messages on the robot controller by setting the CONF_MESS input. After acknowledging the messages, the robot sets the STOPMESS output of the block as a feedback signal. The CONF_MESS input can be reset after the feedback signal by the robot.</td>
</tr>
<tr>
<td>5</td>
<td>Then start the robot by setting the EXT_START signal. The robot controller signals ready for operation and robot successfully switched on by setting the PRO_ACT output signal. Now the robot has been switched on and can be operated using the function blocks of the KUKA.PLC mxAutomation block library.</td>
</tr>
</tbody>
</table>

**Note**

A prepared block from SIEMENS Robotics Support is available for switching on the robot. It can be obtained from the address listed in the Appendix.

#### 4.4.2 Initializing the robot

Before data can be exchanged between the SIMATIC S7 and the robot, the KUKA.PLC mxAutomation interface must be initialized. It is initialized by calling the KRC.Initialize block.

Successful initialization of the interface is returned to the user program via the Done output.

In addition, the block returns the version IDs of the KUKA.PLC mxAutomation interpreter and the KUKA.PLC mxAutomation block library and the serial number of the robot controller:

- Serial number of the robot controller: KRC_Serial
- Interpreter version: KRC_MAJOR . KRC_MINOR . KRC_REVISION
- Block library version: PLC_MAJOR . PLC_MINOR . PLC_REVISION
4.4 Programming basic functions

4.4.3 Checking the robot status

The robot status or the current status of the robot must be determined using multiple function blocks of the KUKA.PLC mxAutomation block library.

Checking the robot status

The KRC_ReadMXAStatus function block allows you to determine the current status of the data interface of the KUKA.PLC mxAutomation library.

Via the MXA_STATUS PLC data type, the status information is returned at an output of the block that can have the following states:

- The function blocks of the KUKA.PLC mxAutomation library cannot exchange data with the robot controller.
- An error is active on the data interface.
- The interpreter on the robot controller is not active, in stand-by or currently processing a program.
- The robot was stopped and all instructions were aborted.

Reading out error messages of the robot controller

The KRC_ReadKRCError function block allows you to determine the current error state of the robot controller.
Figure 4-16 Reading out error messages of the robot controller

The robot controller contains a message buffer for multiple error messages. With one call of the KRC_ReadKRCError function block, up to ten messages can be simultaneously read out of the message buffer. In addition, the block contains an offset input that allows you to read out the remaining error messages in groups of ten messages.

**Note**

Error messages of the robot controller must be acknowledged via the CONF_MESS input of the KRC_AutoExt function block of the KUKA.PLC mxAutomation block library.

The robot controller contains a message buffer for multiple error messages. With one call of the KRC_ReadKRCError function block, up to ten messages can be simultaneously read out of the message buffer. In addition, the block contains an offset input that allows you to read out the remaining error messages in groups of ten messages.

**Note**

Error messages of the robot controller must be acknowledged via the CONF_MESS input of the KRC_AutoExt function block of the KUKA.PLC mxAutomation block library.

### Scanning diagnostics signals of the robot controller

The KRC_Diag function block allows you to determine various diagnostics data of the robot controller or submit interpreter for processing the commands of the function blocks from the KUKA.PLC mxAutomation block library on the robot controller.

The following diagnostics data is provided by the function block in the SIMATIC controller:

- Number of buffered instructions of the function blocks from the KUKA.PLC mxAutomation block library in the robot controller.
- Heartbeat signal of the submit interpreter that processes the commands of the function blocks from the KUKA.PLC mxAutomation library on the robot controller as lifebeat monitoring.
- Current, shortest, longest and average cycle time of the submit interpreter.
- Error number of the robot interpreter.
- Error number of the submit interpreter.
- Error number of the PLC in the robot controller.
4 Program Generation

4.4 Programming basic functions

Figure 4-17 Scanning diagnostics signals of the robot controller

Note
For each robot, this function block may only be called or instantiated once. If the block is called multiple times, data inconsistency can occur when outputting data.

Reading out error messages of the KUKA.PLC mxAutomation interface

The KRC_ReadMXAError function block allows you to determine the current error state of an axis group. The error cause is output as an error number via the ErrorID output.

Figure 4-18 Reading out error messages of the KUKA.PLC mxAutomation interface
4.4 Programming basic functions

4.4.4 Acknowledging error messages

To acknowledge the various error messages, the KUKA.PLC mxAutomation block library provides multiple function blocks.

**Note**

Error messages of the robot controller can only be acknowledged when the robot is in stop.

Acknowledging error messages of the KUKA.PLC mxAutomation interface

The KRC_MessageReset function block allows you to acknowledge the current error state of an axis group of the KUKA.PLC mxAutomation interface.

![Figure 4-19](image1)

Figure 4-19 Acknowledging error messages of the KUKA.PLC mxAutomation interface

Acknowledging error messages of the robot controller

Using the KRC_AutomaticExternal function block, error messages of the robot controller can be acknowledged by a rising edge at the CONF_MESS input.

![Figure 4-20](image2)

Figure 4-20 Acknowledging error messages of the robot controller
4.4 Programming basic functions

Note
This block is also used to switch on the robot’s power units and select the robot controller’s modes.

4.4.5 Reading out the robot position

The KUKA,PLC mxAutomation block library provides two different function blocks to read out the current robot position.

Reading out the robot axis positions

The KRC_ReadActAxisPos block is used to cyclically read the current position of the robot axes.

Figure 4-21 Robot axis positions

Figure 4-22 Reading out the robot axis positions
The AxisPosition output is used to output the position of the robot axes in the defined E6AXIS structure. In addition, outputs A1 to A6 contain the same information as single values. Outputs E1 to E6 show the current position of the external axes that can also be influenced using the robot controller.

**Reading out the robot position in the Cartesian coordinate system**

The KRC_ReadActualPosition block is used to cyclically read the current position of the robot or tool center point (TCP) in the currently selected Cartesian coordinate system.

The Position output is used to output the current position in the defined 6POS structure. In addition, outputs X to Z and A to C contain the same information as single values for the position in space and the rotation about the coordinate axes. The Status and Turn values are used to specify the robot orientation at the current position in greater detail.

Via the Tool and Base outputs, the currently active tool and the number of the currently selected coordinate system are additionally output.
4.5 Programming motion functions

4.5.1 Motion types

There are two options to move a robot:

- **Axis-specific motion**
  The individual robot axes are moved completely independently of each other. There is no interpolation between the axes. Therefore, the robot’s trajectory is difficult to predict and depends on the current position of the robot axes. The motion is executed at the programmed maximum velocity, in percent, of the individual axes or limited by this velocity.

- **Cartesian motion**
  The individual robot axes are moved coordinated with each other via the coordinate transformation so that the robot’s reference point moves along the specified path. In this way, the robot’s reference point can be moved in a straight line or circular path with specified coordinates in the selected coordinate system. Point-to-point motion in the coordinate system is also possible.
4.5 Programming motion functions

4.5.2 Coordinate definition

Axis-specific motion requires that the end position of the individual axes be specified via the E6AXIS PLC data type. For defining the individual axis values in this data type, the KUKA.PLC mxAutomation block library provides a function that changes the single positions of the axes to the E6AXIS PLC data type.

Figure 4-26 Function for setting the E6AXIS PLC data type

Likewise, for Cartesian motion, the KUKA.PLC mxAutomation library provides a function for defining the coordinates X, Y and Z and rotating about the coordinate axes A, B and C in the E6POS PLC data type.

In addition, the Status and Turn parameters for the precise definition of the robot’s axis position can be included in this data type.

Figure 4-27 Function for setting the E6POS PLC data type
4.5 Programming motion functions

Note

The two functions for setting the PLC data types, E6AXIS and E6POS, additionally include input parameters for six external axes, E1 to E6, that can be used to control axes additionally connected to the robot controller such as conveyors, tool axes, etc.

Cartesian motion requires that, in addition to the motion’s end position, the coordinate system be selected. For this purpose, the KUKA.PLC mxAutomation block library provides an input function for the COORDSYS PLC data type.

Figure 4-28 Function for setting the COORDSYS PLC data type

Using this function, the following setting can be made for all Cartesian motion functions:

Table 4-2 Parameters of the mxA_ValuesToCOORDSYS function

<table>
<thead>
<tr>
<th>Interface</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Inputs</strong></td>
<td></td>
</tr>
<tr>
<td>Tool</td>
<td>Selects the number of the tool coordinate system or active tool.</td>
</tr>
<tr>
<td>Base</td>
<td>Selects the number of the base coordinate system to be used.</td>
</tr>
<tr>
<td>IPO_Mode</td>
<td>Selects interpolation mode, i.e., whether the tool coordinate tool coordinate system or the base coordinate system will be used as a basis for the motion.</td>
</tr>
<tr>
<td><strong>Outputs</strong></td>
<td></td>
</tr>
<tr>
<td>COORDSYS</td>
<td>Data structure for the coordinate system definition of Cartesian traversing motions.</td>
</tr>
</tbody>
</table>

4.5.3 Point-to-point motion (PTP)

In a point-to-point motion, the robot’s reference point is moved to the specified end point. The robot controller independently determines the trajectory between the current robot position and the end point so that the highest possible robot traversing velocity is achieved during the motion.

The robot controller selects the trajectory such that slow robot axes such as axes A4 to A6 preferably execute short motions while fast robot axes move longer distances.
Using the KUKA.PLC mxAutomation block library, point-to-point motions can be executed with the following commands:

- **KRC_MoveAxisAbsolute:**
  Axis-specific traversing motion of the single axes by specifying the absolute axis positions of the end point.

Figure 4-29  KRC_MoveAxisAbsolute function block

- **KRC_MoveDirectAbsolute or KRC_MoveDirectRelative:**
  Cartesian traversing motion by specifying the absolute position of the end point in the selected coordinate system or due to the relative distance between the end point and the current robot position in the selected coordinate system.

Figure 4-30  KRC_MoveDirectAbsolute function block

**Note**
Please note that it is difficult or even impossible to predict the trajectory of a point-to-point motion. Especially in limited ranges of motion, this may result in an unexpected collision of the robot with its environment.

### 4.5.4 Linear motion

With the aid of the KRC_MoveLinearAbsolute / KRC_MoveLinearRelative commands, a Cartesian robot motion can be executed in a straight line between the current robot position and the position of the end point specified as an absolute value or defined by the distance. The position of the end point is specified in the currently selected coordinate system.
4 Program Generation

4.5 Programming motion functions

Figure 4-31 KRC_MoveLineaAbsolute function block

Note
Linear robot motions are generally slower than point-to-point motions as moving the robot in a straight line requires that all robot axes be moved on a defined basis.

4.5.5 Circular motion

With the aid of the KRC_MoveCircAbsolute / KRC_MoveCircRelative commands, a Cartesian robot motion can be executed along a circular path between the current robot position and the position of the end point specified as an absolute value or defined by the distance.

The precise definition of the circular path requires that an additional absolute auxiliary point be defined. The position of the end point and auxiliary point is specified in the currently selected coordinate system.

Figure 4-32 KRC_MoveCircAbsolute function block
4.5 Programming motion functions

4.5.6 Approximate positioning motion function

The motion commands listed in the previous chapters have so-called approximation parameters that allow you to define the approach behavior of the motion’s end point. Approximate positioning allows the robot to execute paths composed of single commands in a continuous motion.

The approximation parameters for the motion commands are defined using the APO PLC data type for whose setting the KUKA.PLC mxAutomation block library provides a function.

Figure 4-33 Function for setting the APO PLC data type

Using this function, the following setting can be made for all motion functions:

Table 4-3 Parameters of the mxA_ValuesToAPO function

<table>
<thead>
<tr>
<th>Interface</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>PTP_Mode</td>
<td>Defines the approximation motion for a point-to-point motion.</td>
</tr>
<tr>
<td>CP_MODE</td>
<td>Defines the approximation motion for an interpolated motion, e.g. the robot motion in a straight line or circular path.</td>
</tr>
<tr>
<td>CPTP</td>
<td>Approximation distance, in percent, for a point-to-point motion at which – before the end point – approximate positioning starts at the earliest. The maximum distance (100%) represents half the distance between the start and end point.</td>
</tr>
<tr>
<td>CDIS</td>
<td>Distance to the end point, in millimeters, at which approximate positioning starts at the earliest.</td>
</tr>
<tr>
<td>CORI</td>
<td>Orientation parameter, in degrees, at which approximate positioning may start, at the earliest, when the dominant orientation angle falls below the value for the end point set here.</td>
</tr>
<tr>
<td>CVEL</td>
<td>Defines the velocity, in percent, at which approximate positioning in the braking phase for the end point may start at the earliest.</td>
</tr>
</tbody>
</table>
4 Program Generation

4.5 Programming motion functions

<table>
<thead>
<tr>
<th>Interface</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>Outputs</td>
<td></td>
</tr>
<tr>
<td>APO</td>
<td>Data structure for defining the approximation parameters.</td>
</tr>
</tbody>
</table>

### 4.5.7 Orientation control

Orientation control allows you to define the tool center point (TCP) orientation during the motion.

For orientation control, the OriType input parameter allows you to make the following settings on the motion control blocks of the KUKA.PLC mxAutomation library:

- OriType = 0: VAR
  The tool center point (TCP) orientation changes continuously during the motion.

- OriType = 1: CONSTANT
  The tool center point (TCP) orientation remains constant during the motion.

- OriType = 2: JOINT
  The TCP orientation changes continuously during the motion, but not uniformly. This is done by linear transformation (axis-specific motion) of the wrist axis angles.

### 4.5.8 Manually moving robot axes (JOG)

To manually move the robot axes, the KUKA.PLC mxAutomation block library provides two function blocks: KRC_JogLinearRelative and KRC_JogToolRelative. These two function blocks allow you to approach a defined Cartesian position in the WORLD coordinate system or TOOL coordinate system in a straight line relative to the current robot position.
4 Program Generation

4.5 Programming motion functions

Figure 4-35 KRC_JogLinearRelative function block

![KRC_JogLinearRelative](image)

Figure 4-36 KRC_JogToolRelative function block

![KRC_JogToolRelative](image)

Starting the jog functions always aborts all motion functions active or buffered on the robot and the robot is decelerated to a standstill before executing the jog motion.

Manually moving the robot axis using jog keys

If you want to move the robot axes using jog keys, the KRC_MoveLinearAbsolute function, for a motion in the coordinate system, or the KRC_MoveAxisAbsolute function, for a motion of the robot axes, and the KRC_Abort function have to be interconnected in a function block. Using axis-specific jog keys, the motion of the respective axis can then be started, for example, with a rising signal edge of the key. A falling signal edge of the key terminates the axis motion using the KRC_Abort function and therefore stops the axis.

When using this procedure, a position value shortly before the software limit switch of the axis has to be selected for the appropriate axis.

Note

A prepared block from SIEMENS Robotics Support is available for moving the robot using jog keys. It can be obtained from the address listed in the Appendix.

The following figure shows an example of such a jog block for moving the robot axes using jog keys.
4.6 Summary program example

The aim of this chapter is to once again illustrate the basic structure of a user program for controlling a KUKA industrial robot using the KUKA.PLC mxAutomation block library.

4.6.1 Complete program overview

The figure below shows an example of the block library function blocks, and their call sequence, that are at least necessary in order to move a KUKA industrial robot from a SIMATIC S7.

The program example is designed for controlling two separate KUKA industrial robots from the same SIMATIC S7 controller.

For each robot, the figure shows the function blocks for reading and writing the robot data. Between these blocks, two function blocks are called that contain the basic control of the robot using the administrative functions and the robot's motion control. The following two chapters provide a detailed explanation of these two blocks.
4.6.2 Robot control detail view

The Robot function block contains the robot’s administrative functions that allow you to execute the following functions:

- Switch on the robot using a separate function block.
- Initialize the KUKA.PLC mxAutomation interface.
- Check the status of the KUKA.PLC mxAutomation data interface.
- Read error messages of the robot controller, the robot’s diagnostics data and the error messages of the KUKA.PLC mxAutomation data interface.
- Acknowledge error messages on the KUKA.PLC mxAutomation data interface.
- Select the robot mode / acknowledge robot messages.
- Set override for robot motions.
4.6 Summary program example

- Read out the current robot position in axis positions or Cartesian coordinates.

Figure 4-39 Robot function block detail view

4.6.3 Motion control detail view

The RobotMove function block contains all function blocks for executing the robot motions. This block shows an example of the programming of a motional sequence using different motion blocks from the KUKA.PLC mxAutomation block library.

In this block, the following is done: First the parameters are set for each motion using the appropriate parameters and then the function block for the desired motion is called.

The beginning of the block additionally shows the functions for setting the parameters for approximate positioning of the motions and selecting the coordinate system for executing the motion.
The prepared block for implementing the robot’s axis motions using jog keys is also integrated at the beginning of the function block.

Figure 4-40  RobotMove function block detail view

Note
If you want the individual motions to be approximated, the KRC_MoveAxisAbsolute function blocks must be interconnected via the Busy output; i.e., once a function block signals Busy, the next function block can be started.
5 Additional Functions

The aim of this chapter is to introduce you to selected additional functions that go beyond basic control of a KUKA industrial robot with the aid of the KUKA.PLC mxAutomation library.

5.1 Tool control (gripper)

For an industrial robot to perform the desired activities, a tool must be installed on the robot. These tools generally feature numerous inputs and outputs or a bus port to be able to control the tool as desired or check the current tool status.

As the robot is controlled using the SIMATIC S7 controller with the aid of the KUKA.PLC mxAutomation block library, tool control has two options to transfer the desired commands to the gripper.

5.1.1 Control using the robot controller

The robot controller of a KUKA industrial robot also features a built-in PLC that allows you to control the inputs and outputs of a tool.

This allows you to connect the inputs and outputs of a tool also directly to the robot controller.

Advantages

This type of control provides the following advantages:

- Robot and tool form a hardware unit.
- The tool can be controlled by controlling the PLC inputs and outputs of the robot controller with the aid of the KRC_ReadDigitalOutput and KRC_WriteDigitalOutput blocks from the KUKA.PLC mxAutomation block library.

Disadvantages

This type of control involves the following disadvantages:

- The control data for the tool must first be transferred to the robot controller.
- If it is necessary to consistently transfer multiple data bits to the tool at the same time, the program has to ensure data consistency, which increases the programming work required in the SIMATIC S7 controller and on the robot controller.
- If the tool is controlled using a bus system from the robot controller, data transfer may be delayed due to the data transfer in the robot controller to the second bus system.

5.1.2 Direct control using the SIMATIC S7

As the SIMATIC S7 controller takes full control and motion control of the robot, the robot’s tool can also be applied directly from the SIMATIC S7 or distributed I/O connected to the controller.

If necessary, the distributed I/O of the SIMATIC S7 controller can also be integrated directly into the tool.
5 Additional Functions

5.2 Safety Integrated

Advantages
This type of control provides the following advantages:

- Easy control of the tool from the user program of the SIMATIC S7 controller using the known I/O commands integrated in the program for generating the robot’s motion commands.
- Option to integrate the distributed I/O of the SIMATIC S7 controller directly into the tool or robot arm, which significantly reduces the I/O wiring to the tool.
- Option to exchange data with the distributed I/O through wireless communication.

Disadvantages
This type of control involves the following disadvantages:

- Triggering the tool functions requires that the current robot status be checked and compared using the SIMATIC S7 controller.
- Difficult or impossible to implement motion-synchronous tool actions.

5.2 Safety Integrated

To use the Safety Integrated functionality in conjunction with a KUKA industrial robot, KUKA provides the SafeOperation safety option:

5.2.1 Components

The KUKA SafeOperation safety option includes the following components:

Software components

Table 5-1 KUKA components – SafeOperation safety option

<table>
<thead>
<tr>
<th>Component</th>
<th>No.</th>
<th>Article number</th>
<th>Note</th>
</tr>
</thead>
<tbody>
<tr>
<td>KUKA.SafeOperation</td>
<td>1</td>
<td></td>
<td>Version: V3.2</td>
</tr>
</tbody>
</table>

Hardware components

Table 5-2 KUKA components – SafeOperation safety option

<table>
<thead>
<tr>
<th>Component</th>
<th>No.</th>
<th>Article number</th>
<th>Note</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reference switch module</td>
<td>1</td>
<td></td>
<td>For automated checking of the safety areas set on the robot.</td>
</tr>
</tbody>
</table>

5.2.2 Functionality

The KUKA SafeOperation safety option contains the following functionality:

- Safe monitoring of max. 16 user-defined axis-specific or Cartesian monitoring spaces.
- Safe monitoring of a user-defined cell area.
- Safe monitoring of axis-specific velocities.
- Safe monitoring of space-specific velocities.
- Safe monitoring of Cartesian velocities.
5 Additional Functions

5.3 PROFIenergy

- Modeling of up to 16 safe tools with a safe TCP.
- Safe stop using the safety control system.
- Safe operational stop for up to 6 axis groups.
- Safe inputs to activate the monitoring functions.
- Safe outputs for status messages of the monitoring functions.
- Creating and editing the safety configuration on the robot controller.

5.2.3 Control

The SafeOperation safety option is controlled using the PROFINET connection between the SIMATIC controller and the robot controller with the aid of the PROFIsafe message frame.

If you want to use this connection, you can simply integrate the PROFIsafe extension into the message frame configuration of the KUKA robot controller. For this purpose, the PROFIsafe message frame extension is included in the GSDML file of the KUKA robot controller.

Figure 5-1 Control of the SafeOperation safety option via PROFIsafe

| Note | As an alternative to data exchange via PROFIsafe, the robot’s safety functions can also be controlled using discrete hardware signals via the X11 interface of the robot controller. |

5.3 PROFIenergy

In its basic configuration, the KUKA industrial robot is already a PROFIenergy device that allows you to use the energy saving options provided by PROFIenergy. Using standardized PROFIenergy telegrams, the robot can be set to or brought back from Hibernate mode.

With regard to PROFIenergy, the KUKA industrial robot has the following possible states:

| Table 5-3 PROFIenergy states of the KUKA industrial robot |
| --- | --- |
| State | Function |
| Ready_To_Operate | The robot or robot controller is ready for operation and can be used. |
| Drive bus OFF | The robot drives are OFF, the robot cannot be used. |
5 Additional Functions

5.4 Receiving robot messages in plain text

<table>
<thead>
<tr>
<th>State</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hibernate</td>
<td>The robot controller is in Hibernate mode. There is no PROFINET communication. The robot controller only responds to the Wake-On-LAN packet to wake up the controller.</td>
</tr>
<tr>
<td>Brakes applied</td>
<td>The robot or robot controller is ready for operation, but the robot's brakes have been applied. The robot responds only to the next motion command.</td>
</tr>
</tbody>
</table>

Note

If the production break is too short to use the Drive bus OFF state, it is nevertheless possible to save energy using the Brakes applied state.

To set the robot to this state, however, it must be in programmed stop.

Note

When setting a pause time for the robot, i.e., when setting the appropriate Pause state on the robot, make sure that the total time you select is greater than the sum of the falling asleep phase (Time_to_Pause) and the wake-up phase (Time_to_operate) of the robot.

Note

If a PROFIenergy telegram is sent to the robot, this job will be directly executed in the robot controller. Therefore, it should be made sure to send the PROFIenergy telegram to the robot only when it is in standstill.

5.4 Receiving robot messages in plain text

In addition to the blocks for determining status and error messages of the robot controller and the data interface of the KUKA.PLC mxAutomation block library mentioned in the previous chapters, the user has the additional option to receive robot messages from the robot controller in plain text via a UDP interface.

5.4.1 Principle of operation

The robot controller outputs all robot messages via a UDP connection on the PROFINET connection. These messages can be received in the SIMATIC controller and, if necessary, displayed on the HMI operator panel.

In addition, it is also possible to send commands to the robot controller via the UDP connection that allow you to configure the sending of the robot messages. The following commands are possible:
### 5.4 Receiving robot messages in plain text

<table>
<thead>
<tr>
<th>Command</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>SwitchLanguage eng</td>
<td>Change output of robot messages to English.</td>
</tr>
<tr>
<td>Number of characters = 18</td>
<td></td>
</tr>
<tr>
<td>SwitchLanguage deu</td>
<td>Change output of robot messages to German.</td>
</tr>
<tr>
<td>Number of characters = 18</td>
<td></td>
</tr>
<tr>
<td>GetAllKrcMessages AllNoInfo</td>
<td>Receive pure message texts of robot.</td>
</tr>
<tr>
<td>Number of characters = 27</td>
<td>In this receive mode, a 150-character target data area is sufficient for the robot messages.</td>
</tr>
</tbody>
</table>

For transfer to the robot controller, the commands must be entered in the source data area for the commands as listed in the above table. For sending, the table also includes the command length, i.e., the number of characters.

In the SIMATIC controller, the source data area for the commands and the target data area of the robot messages are created as an Array[0..150] of Char.

#### 5.4.2 Programming

The UDP connection between the robot controller and the SIMATIC controller is established using the robot controller. Therefore, it is important to use the connection ID specified by the robot controller for the TUSEND and TURCV blocks. The connection ID of 256<sub>DEC</sub> or 16#100<sub>HEX</sub> is generally set in the robot controller. The connection ID must be transferred at the TUSEND and TURCV blocks via the ID parameter.

Furthermore, the IP address and communication port of the robot controller have to be transferred at the TUSEND and TURCV blocks via the ADDR parameter. The data is saved as the TADDR_Param PLC data type that contains an array for the IP address and port number. The robot controller sends the robot messages to the UDP nodes via port 9050<sub>DEC</sub>. For the KUKA robot controller, the same variable can be used for the ADDR parameter for the TUSEND and TURCV blocks as both the commands and the messages are communicated via the same port.

### Sending commands

Via the TUSEND block, the desired commands are sent to the robot controller with a rising edge at the REQ input. The LEN input must be provided with the number of characters to be sent. The source memory area where the command to be sent is stored must be transferred at the DATA input.

![TU_SEND function block for sending via a UDP connection](image)
5 Additional Functions

5.4 Receiving robot messages in plain text

Receiving robot messages

The robot messages are received via the TURCV block as long as the EN_R input is set to True. When receiving robot messages, it is never necessary to change the value 0 of the LEN input. The target data area to which the message is to be saved is transferred at the DATA input.

Figure 5-3  TU_RCV function block for receiving via a UDP connection

Via the NDR output, the block signals a new message and, via the RCVD_LEN output, transfers the number of characters actually received to the user program.
The aim of this chapter is to provide additional background information on handling and programming industrial robots.

### 6.1 Coordinate systems

#### 6.1.1 Overview

Multiple coordinate systems are defined in the robot controller. They determine the position of the workpiece and tool and form the basis of the individual robot motions.

Figure 6-1  Overview of the robot coordinate systems

#### 6.1.2 Original coordinate system of the system (WORLD)

The WORLD coordinate system is the original coordinate system of the entire system. This coordinate system forms the basis for all other coordinate systems that result from translation and rotation.

If no rotations and translations of the other coordinate systems are defined, the origin of the WORLD coordinate system is located at the robot base.
In the robot program, the rotations and translations of the other coordinate systems relative to the WORLD coordinate system are transferred as a robot position with six parameters and the individual parameters have the following functions:

- **X coordinate**: Translation of the coordinate system along the X axis.
- **Y coordinate**: Translation of the coordinate system along the Y axis.
- **Z coordinate**: Translation of the coordinate system along the Z axis.
- **A coordinate**: Right-hand rotation of the coordinate system about the X axis of the WORLD coordinate system.
- **B coordinate**: Right-hand rotation of the coordinate system about the Y axis of the WORLD coordinate system.
- **C coordinate**: Right-hand rotation of the coordinate system about the Z axis of the WORLD coordinate system.

**Note**

The rotation of the coordinate system about the axes of the WORLD coordinate system follows this order:

- Rotation about the X axis by the absolute value of the A coordinate
- Rotation about the Y axis by the absolute value of the B coordinate
- Rotation about the Z axis by the absolute value of the C coordinate

### 6.1.3 Robot base coordinate system (ROBROOT)

The robot base coordinate system (ROBROOT) allows you to achieve an offset of the robot's place of installation relative to the origin of the WORLD coordinate system.

**Note**

When starting up the robot, the robot base coordinate system must be defined in the robot controller data.

This data cannot be changed from the user program using blocks of the KUKA.PLC mxAutomation block library.

### 6.1.4 Workpiece coordinate system (BASE)

The workpiece coordinate system (BASE) is used to define the origin of the workpiece to be machined relative to the WORLD coordinate system.

Translation and rotation of the workpiece coordinate system (BASE) can be defined from the user program using the KRC_WriteBaseData function. The current settings of the coordinate system or coordinate system translation can be determined using the KRC_ReadBaseData function.

Up to 32 different coordinate systems or coordinate system translations for the workpiece coordinate system (BASE) can be stored in the robot controller. They can then be selected for easy definition of the robot's motion commands using the user program.
6.2 Coordinate definition

6.2.1 Robot axes

A KUKA industrial robot features up to six individual axes that allow the user to move the robot kinematics. To position the robot, these axes can be controlled either directly from the user program using axis coordinates A1 to A6 or by the coordinate transformation in the robot controller by specifying, from the user program, a position in a coordinate system defined in the robot controller via the Cartesian coordinates X, Y and Z and the rotational angles about the coordinate axes A, B and C.

Figure 6-2 Axes of a 6 axis robot
6.2 Coordinate definition

**Note**

In addition to the robot axes A1 to A6, the KUKA robot controller provides six external axes E1 to E6 that can also be influenced from the SIMATIC controller using the user program.

However, the external axes (E1 to E6) are not the subject of this example. For more information, please refer to the KUKA.PLC mxAutomation documentation or the robot controller documentation.

6.2.2 Axis coordinates

Direct positioning of the robot axes can be achieved by specifying the desired axis positions in the user program using the E6AXIS data type. The KRC_MoveAxisAbsolute command is available for this purpose.

This command directly moves the robot axes from the current position to the position specified using E6AXIS. In this process, there is no coordination of the individual axis motions. Each robot axis moves to the specified position at the specified percentage of its maximum axis velocity.

6.2.3 Cartesian coordinates

The E6POS data type and various motion commands are available to position the robot axes with the aid of the robot controller’s coordinate transformation.

Depending on the selected coordinate system such as the workpiece coordinate system (BASE) or tool coordinate system (TOOL), the robot axes can be positioned by specifying a Cartesian position and the orientation vector at this position via the coordinate transformation.

Depending on the selected motion command, the robot controller’s coordinate transformation coordinates the robot axes, for example, for the motion of the robot reference point in a straight line or circular path.

6.2.4 Additional data: “Status” and “Turn”

If the robot is positioned via the robot controller’s coordinate transformation, specifying the position (X, Y and Z) and orientation (A, B and C) is not sufficient to unambiguously define the robot’s axis position at each position. The additional data for Status (S) and Turn (T) is available for this purpose.

**Note**

The robot controller considers specified Status and Turn values only for point-to-point (PTP) motions. Therefore, the first motion in a motion program of the robot must always be a fully defined PTP motion so that a unique starting position is defined for the robot. For all other motion commands, it is not necessary to change the value 0 of Status and Turn. The robot controller itself defines the axis position.
6 Appendix

6.2 Coordinate definition

Additional data: Status

Bits 0 to 4 of Status allow you to define the position of the robot axes relative to the coordinate system on axis A1 at the robot base.

Table 6-1 Additional data: Status

<table>
<thead>
<tr>
<th>Status</th>
<th>Bit state</th>
<th>Bit 0</th>
<th>Bit 1</th>
<th>Bit 2</th>
<th>Bit 3</th>
<th>Bit 4</th>
<th>Bit 5</th>
<th>Bit 6</th>
<th>Bit 7</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Bit 0</td>
<td>Basic area</td>
<td>The intersection of wrist axes A4, A5 and A6 is above the positive X axis of the coordinate system relative to axis A1.</td>
<td>0</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Bit 1</td>
<td>The position of axis A3 is less than zero.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Bit 2</td>
<td>The position of axis A5 is less than or equal to zero.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Bit 3</td>
<td>Unused</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Bit 4</td>
<td>The point was not taught with a robot with absolutely accurate calibration.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Bit 5</td>
<td>Unused</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Bit 6</td>
<td>Unused</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Bit 7</td>
<td>Unused</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Figure 6-3 Example of the Status additional data – bit 0

Intersection of wrist axes A4, A5 and A6

Projection of the intersection to the coordinate axis

Coordinate axis perpendicular to A1 and A2

Bit 0 = 0

Bit 0 = 1

© Siemens AG 2016. All rights reserved
6 Appendix

6.2 Coordinate definition

Additional data: Turn

Bits 0 to 5 of Turn allow you to define the direct position of the robot axes in the axes’ ranges of motion.

Table 6-2 Additional data: Turn

<table>
<thead>
<tr>
<th>Turn</th>
<th>Bit state</th>
<th>Bit state</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>Bit 0</td>
<td>The position of axis A1 is greater than or equal to zero.</td>
<td>The position of axis A1 is less than zero.</td>
</tr>
<tr>
<td>Bit 1</td>
<td>The position of axis A2 is greater than or equal to zero.</td>
<td>The position of axis A2 is less than zero.</td>
</tr>
<tr>
<td>Bit 2</td>
<td>The position of axis A3 is greater than or equal to zero.</td>
<td>The position of axis A3 is less than zero.</td>
</tr>
<tr>
<td>Bit 3</td>
<td>The position of axis A4 is greater than or equal to zero.</td>
<td>The position of axis A4 is less than zero.</td>
</tr>
<tr>
<td>Bit 4</td>
<td>The position of axis A5 is greater than or equal to zero.</td>
<td>The position of axis A5 is less than zero.</td>
</tr>
<tr>
<td>Bit 5</td>
<td>The position of axis A6 is greater than or equal to zero.</td>
<td>The position of axis A6 is less than zero.</td>
</tr>
<tr>
<td>Bit 6</td>
<td>Unused</td>
<td>Unused</td>
</tr>
<tr>
<td>Bit 7</td>
<td>Unused</td>
<td>Unused</td>
</tr>
</tbody>
</table>

Figure 6-4 Example of the Turn additional data
6 Appendix

6.2 Coordinate definition

**Note**
If Status and Turn are not specified when positioning the robot, the robot controller itself decides on the transition from one robot axis position to the next.

### 6.2.5 Singularities

KUKA six axis industrial robots have the following three different singularities:

- **Overhead singularity**
  Axis A5 is perpendicular to axis A1.

- **Extended position singularity**
  Axis A5 is in the extension of robot axis A2 and A3.

- **Wrist axis singularity**
  Axes A4 and A6 are parallel to one another and axis A5 moves to the extended position A5, i.e., axis A5 is in the range of ±0.01812°.

When programming or manually approaching points, consider and, where possible, avoid these singularities or use different programming – for example, directly move the robot axes instead of moving them using a Cartesian motion command – to bypass them.

**NOTICE**
If there is at least one singularity in the desired workspace of the robot, it may be necessary to check and correct the robot’s position in the system so that the singularity no longer occurs in the workspace.

### 6.2.6 Tool center point (TCP)

All Cartesian motion commands of the robot refer to the tool center point (TCP). The reference point of a tool is defined by writing a tool offset relative to the WORLD coordinate system using the KRC_WriteToolData command.

If no tool is selected, the TCP is on the robot flange, i.e., a tool without translation and rotation is assumed.
## 7 Links & Literature

### Table 7-1

<table>
<thead>
<tr>
<th>Topic</th>
<th>Title</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Siemens Industry Online Support</td>
<td><a href="http://support.industry.siemens.com">http://support.industry.siemens.com</a></td>
</tr>
</tbody>
</table>

**Siemens SIMATIC S7 / Safety Integrated / PROFIenergy**

| 3. SIMATIC STEP 7 Professional                             | SIMATIC STEP 7 Professional V13 SP1 System Manual                       |
|                                                           | Edition: 12/2014 Document ID: Online help printout Order number: -     |

| 4. SIMATIC PROFINET                                       | SIMATIC PROFINET System Description System Manual                       |
|                                                           | Edition: 03/2012 Document ID: A5E00298287-06 Order number: -           |

| 5. SIMATIC Safety                                         | SIMATIC Industrial Software SIMATIC Safety – Configuring and Programming |

| 6. PROFIenergy                                           | Application Guideline for Implementing the Switch-Off Concepts With PROFIenergy Application Example |

| 7. PROFIenergy                                           | Which modules of the Siemens portfolio support PROFIenergy functions? FAQ |

| 8. PROFIenergy                                           | SIMATIC S7 Library for the Simple Configuration of PROFIenergy Application Example / Block Library |

**KUKA.PLC mxAutomation**

| 9. KUKA home page                                        | KUKA Robotics                                                                 |
|                                                          | [http://www.kuka-robotics.com](http://www.kuka-robotics.com)                  |
|                                                          | KUKA Download Center – Brochures                                               |
8 Contact

If you have any questions regarding KUKA industrial robots or the KUKA.PLC mxAutomation block library or want to order the necessary KUKA documentation, please use the below contact details.

8.1 Siemens Robotics Support

To contact SIEMENS Robotics Support, please use the following email address:
Email: robotic.industry@siemens.com

8.2 KUKA Roboter GmbH

KUKA Roboter GmbH – Headquarters
Zugspitzstraße 140
86165 Augsburg, Germany
Phone: +49 821 797-4000
Fax: +49 821 797-4040
Email: info@kuka-roboter.de
8.3 **KUKA Roboter GmbH Hotline**

KUKA Roboter GmbH – Hotline
In Germany, the hotline is available 24 hours a day and 365 days a year:
Phone: +49 821 797-1926

8.4 **KUKA.PLC mxAutomation Sales**

KUKA Roboter GmbH – Global Sales Center
Hery-Park 3000
86368 Gersthofen, Germany
Phone: +49 821 4533-0
Fax: +49 821 4533-1616
Email: sales@kuka-roboter.de

**Contact:**
Philipp Kremer
Product Manager
Phone: +49 821 4533-3734
Fax: +49 821 797-41 3734
Philipp.Kremer@kuka.com

9 **History**

<table>
<thead>
<tr>
<th>Version</th>
<th>Date</th>
<th>Modifications</th>
</tr>
</thead>
<tbody>
<tr>
<td>V1.0</td>
<td>01/2016</td>
<td>First version</td>
</tr>
<tr>
<td>V1.1</td>
<td>01/2016</td>
<td>Text added/modified</td>
</tr>
</tbody>
</table>