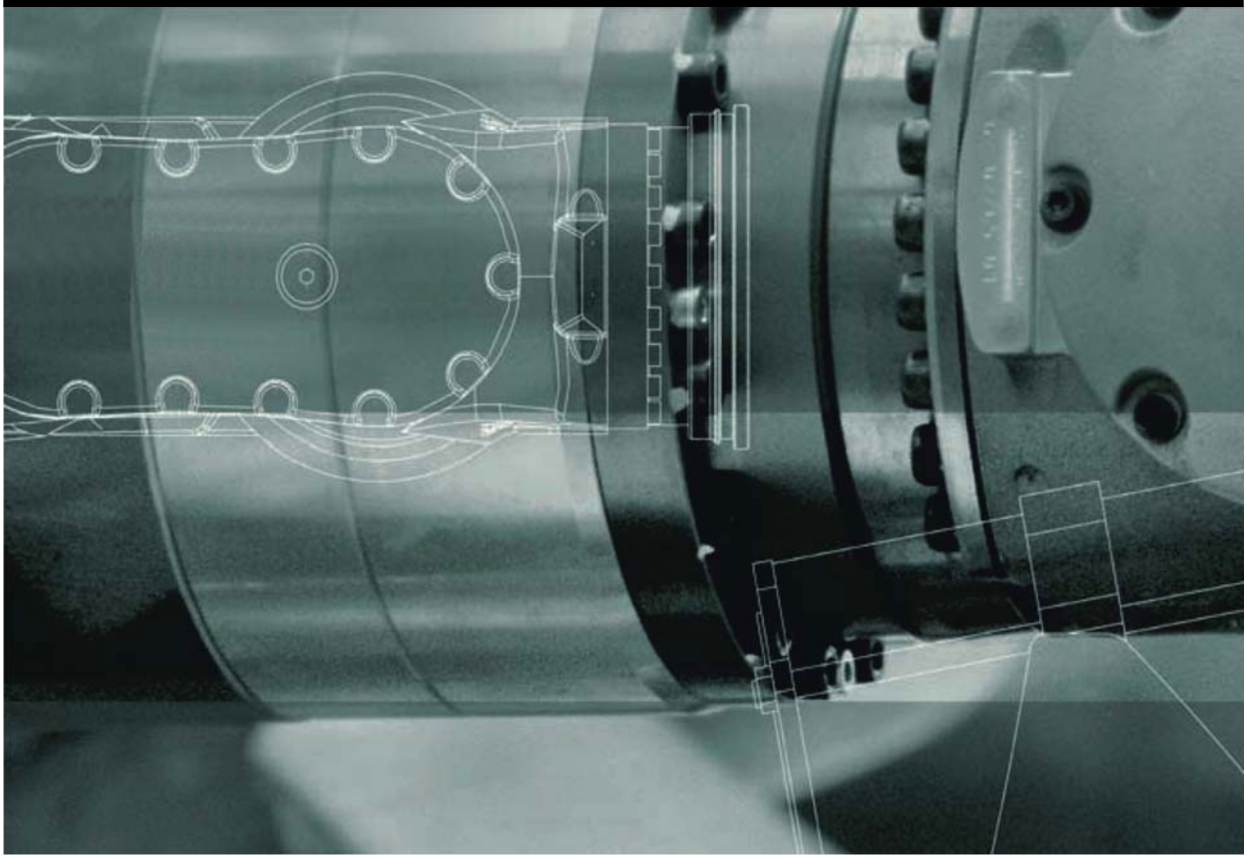
Software Option

KUKA Roboter GmbH

KUKA Sunrise.Connectivity FRI 1.5

For KUKA Sunrise.Workbench 1.5

For KUKA Sunrise.OS 1.5



Issued: 30.01.2015

Version: KUKA Sunrise.Connectivity FRI 1.5 V3

Translation of the original documentation

KIM-PS5-DOC

Publication:	Pub KUKA Sunrise.Connectivity FRI 1.5 (PDF) en
Book structure:	KUKA Sunrise.Connectivity FRI 1.5 V2.2
Version:	KUKA Sunrise.Connectivity FRI 1.5 V3

Contents

1 Introduction

1.1 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.

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

Hints

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

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

1.3 Terms used

Term	Description
API	Application Programming Interface Interface for programming applications.
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.
FRI	Fast Robot Interface Real-time interface which can be programmed by the user
Header file	Text file in the programming language C++ containing declarations and other components of a source code
Host	Computer in a network with an IP address and port number
Jitter	Standard deviation from the latency
KLI	KUKA Line Interface Connection to Ethernet network
KONI	KUKA Option Network Interface
KUKA LBR iiwa	KUKA Lightweight robot **i**ntelligent **i**ndustrial **w**ork **a**ssistant
KUKA RoboticsAPI	Java programming interface for KUKA robots RoboticsAPI is an object-oriented Java interface for controlling robots and peripheral devices.
KUKA Sunrise Cabinet	Control hardware for operating the LBR iiwa industrial robot
KUKA Sunrise.OS	KUKA Sunrise.Operating System System software for industrial robots which are operated with the robot controller KUKA Sunrise Cabinet
KUKA Sunrise. Workbench	Development environment for the start-up of a robot cell (station) and for the development of robot applications
Latency	Mean value of the response time to a signal or delay of a signal
SDK	Software Development Kit
Socket	Software module which allows the robot controller to be connected to an external system and to exchange data
UDP	User Datagram Protocol Connectionless protocol for the data exchange between the devices of a network
IP	Internet Protocol The Internet Protocol is used to define subnetworks by means of physical MAC addresses.

1.4 Trademarks

Visual Studio is a trademark of Microsoft Corporation.

Windows is a trademark of Microsoft Corporation.

2 Purpose

2.1 Target group

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

- Advanced system knowledge of robotics
- Advanced C++ programming skills
- Basic knowledge of KUKA RoboticsAPI programming
- Basic knowledge of control technology and stability
- Basic knowledge of real-time programming

> 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.

2.2 Intended use

Use

KUKA Sunrise.Connectivty FRI provides an interface for carrying out experiments with the LBR iiwa under laboratory conditions.

KUKA Sunrise.Connectivity FRI may be used exclusively in accordance with the specified system requirements for the purpose of investigating new application fields, applications and algorithms for the LBR iiwa.

Operation in accordance with the intended use also involves continuous observance of the following documentation:

- Assembly and operating instructions of the industrial robot
- Assembly and operating instructions of the robot controller
- Operating and programming instructions for the System Software

Misuse

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

Examples of such misuse include:

- Use of KUKA Sunrise.Connectivity FRI not in accordance with the specified system requirements
- Use in an industrial environment

3 Product description

3.1 Overview of KUKA Sunrise.Connectivity FRI

Sunrise.Connectivity FRI is an add-on software option for Sunrise.Workbench. FRI is an interface via which data can be exchanged continuously and in real time between a robot application on the robot controller and an FRI client application on an external system.

The real-time capability provides the FRI client application with fast cyclical access to the robot path at millisecond intervals.

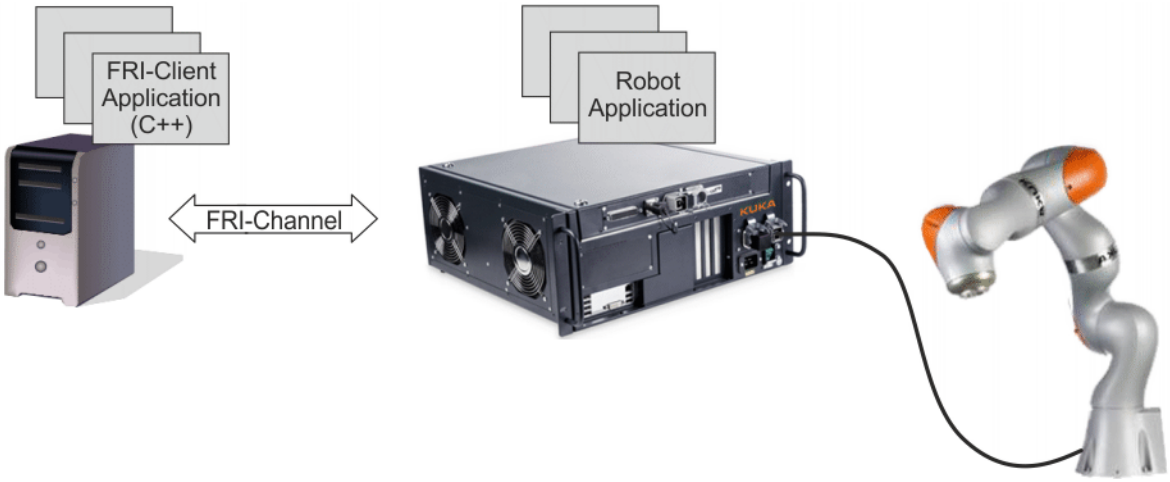


Fig. 3-1: Overview of application development with FRI

Robot application

Robot applications are programmed in Java and executed on the robot controller.

The robot application has the following functionalities:

- Sequence control for the robot
- Management of coordinate systems, objects and motions
- Programming interface for integration of external libraries and functionalities

The following functionalities are added with Sunrise.Connectivity FRI:

- Configuration and management of various FRI channels
- Activation of access by FRI client applications to selected motion sections

FRI client application

FRI client applications can be created in the programming language C++. They are executed on an external system.

The FRI client application has the following functionalities:

- Evaluation of the robot data in real time
- Continuous evaluation of the application states in real time
- Path superposition under real-time conditions

FRI channel

The robot application and FRI client application are connected via the FRI channel. The FRI channel is an Ethernet-based UDP interface with the following functionalities:

- Binary data exchange
- Management via the robot controller
- Recording of the effective cycle time for monitoring the connection quality and real-time capability

3.1.1 Funktionality of the FRI

Description

There are 2 aims of the FRI functionality:

- Observation
- Influence

Monitor mode

Monitor mode observes the robot controller and sends the current robot data to the FRI client in real time.

The following robot data are transferred in Monitor mode:

- Connection state
- Axis-specific actual position
- Axis-specific actual torque
- Axis-specific setpoint position
- Axis-specific setpoint torque
- Axis-specific external torque
- Activity of the drives (switched on/off)
- Operating mode (T1, T2, AUT)
- Status of the safety signal “Safety stop”
- Number of axes
- Time stamp of the data transmission

The cycle time for data transfer can be configured: range from 1 ... 100 ms.

Command mode

In Command mode, the robot controller can be influenced externally.

Command mode has the following tasks:

- The robot path can be superposed by the FRI client.
- The FRI client can react to changes on the robot controller cyclically and in real time.

3.1.2 Quality of the FRI connection

The quality of the FRI connection is classified as follows:

- POOR
- FAIR
- GOOD
- EXCELLENT

A classification can only improve in a linear fashion, from the lower quality to next best classification. It is not possible to skip one or more levels. On the other hand, it is always possible for the connection quality to be assessed with any one of the lower classifications.

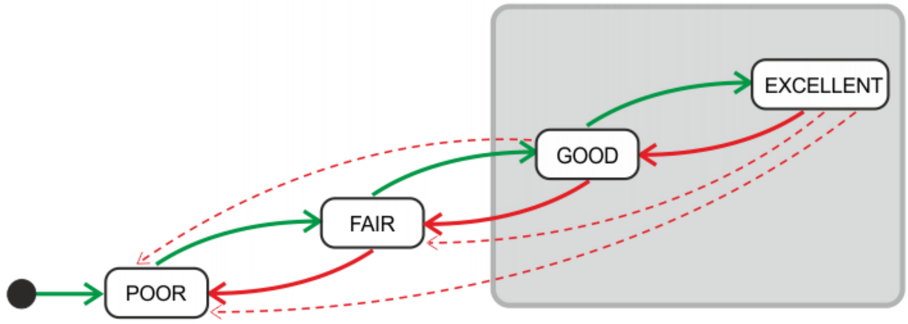


Fig. 3-2: Quality levels of the FRI connection

- Green: Improvement
- Red: Deterioration: data packet lost
- GOOD and EXCELLENT: Command mode (gray) can be used
- POOR and FAIR: only Monitor mode is available

When the FRI connection is first established, the connection quality is always classified as “POOR”. A successful handshake improves the quality of the connection quality from “POOR” to “FAIR” and then from “GOOD” to “EXCELLENT”.

> Commands from the external system to the robot controller are only accepted if the connection quality is “GOOD” or “EXCELLENT”.

(>>> 7.1.4.1 "Polling the connection quality and time statistic" Page 27)

3.1.3 Connection quality characteristics

The following characteristics determine the quality of the FRI connection:

- Latency of the FRI connection
- Jitter of the FRI connection
- Number of lost data packets
- Answer rate (compliance with the reception rate)

The characteristics can be polled.

(>>> 7.1.4.1 "Polling the connection quality and time statistic" Page 27)

3.1.4 Interaction between Monitor mode and Command mode

Monitor mode

The following FRI states are controlled by the robot controller in Monitor mode:

- MONITOR_WAIT

  The system has opened the FRI connection and is waiting for real-time capable data exchange.
- MONITOR_READY

  The system is performing real-time capable data exchange.

  It is possible to switch to Command mode in the MONITOR_READY state. The change is initiated in the robot application if a motion with path superposition is called.

Command mode

The following FRI states are controlled by the robot controller in Command mode:

- COMMAND_WAIT

  The system is waiting for the upcoming transition to Command mode via the FRI.
- COMMAND_ACTIVE

  The system applies the values specified by the FRI client application for superposing the robot path.

The external system is synchronized with the robot controller during the transition phase COMMAND_WAIT. The robot controller switches from COMMAND_WAIT to COMMAND_ACTIVE when the following conditions are met:

- All motions which precede the motion with path superposition have been completed.
- The robot controller has received at least one command message from the FRI client during the transition phase COMMAND_WAIT.
- The difference between the setpoint position of the robot and the commanded setpoint position of the FRI client is less than 0.001 rad.

The figure shows the dependencies between the different FRI states and their dependency on the connection quality.

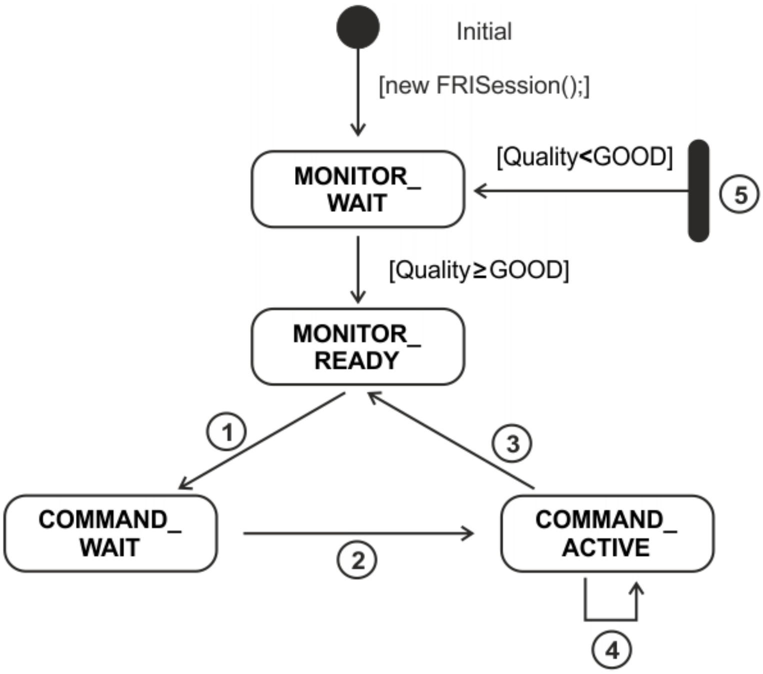


Fig. 3-3: Interaction between Monitor mode and Command mode

Item	Description of state transitions
1	Call of FRI client application The Command mode of the FRI client application can be activated in the MONITOR_READY state with `move(M.addMotionOverlay(FRIJointOverlay(…)));`. (>>> 7.1.3 "Path superposition with the FRI client" Page 26) The call causes FRI to switch to the COMMAND_WAIT state. Precondition: The connection quality is GOOD or EXCELLENT.
2	Acceptance of FRIJointOverlay Following successful synchronization in the transition phase COMMAND_WAIT, the full command authorization COMMAND_ACTIVE is issued.

Item	Description of state transitions
3	End of a superposed synchronous motion Once the synchronized motion is completed, the robot stops with exact positioning. (>>> 7.3.2 "Synchronous motion programming" Page 35) FRI switches to the MONITOR_READY state.
4	End of a superposed asynchronous motion with subsequent motion If the asynchroous motion is completed and a further superposed motion follows, the path is directly resumed. (>>> 7.3.1 "Asynchronous motion programming" Page 34) FRI remains in the COMMAND_ACTIVE state.
5	Connection error The connection quality switches to POOR or FAIR due to a connection error. FRI switches to the MONITOR_WAIT state.

3.1.5 FRI in the robot controller

The following classes are provided for using the FRI in the robot application:

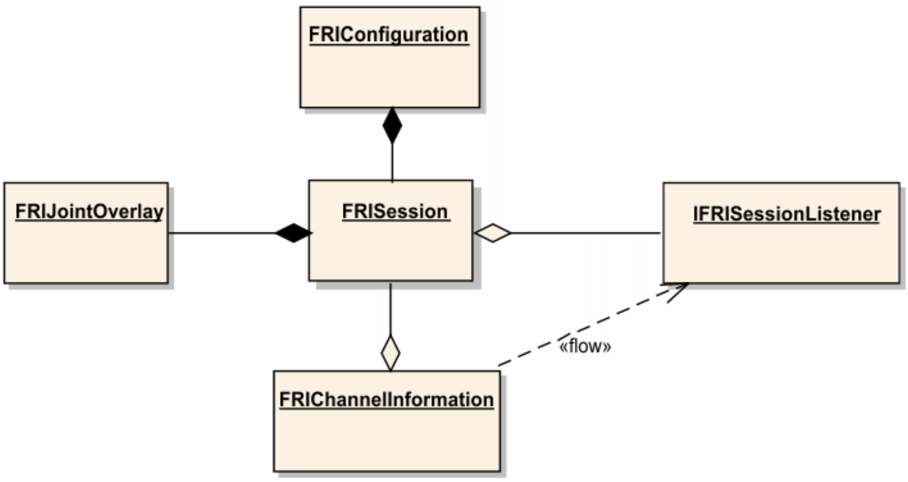


Fig. 3-4: Overview of classes

- The class FRIConfiguration enables the configuration of the FRI connection.
- The configured FRI connection is initialized and opened using the class FRISession. This causes automatic activation of Monitor mode.
- The class FRIChannelInformation can be used to poll the state of the FRI connection.
- The class IFRISessionListener provides a notification mechanism which can be used to automatically record all changes of state.
- The class FRIJointOverlay enables the activation of Command mode.

3.1.6 Application development in the FRI client

The FRI client SDK simplifies application development in the FRI client. For this reason, the network communication, state management and sequence control are encapsulated.

The following classes are provided for application development in the FRI client:

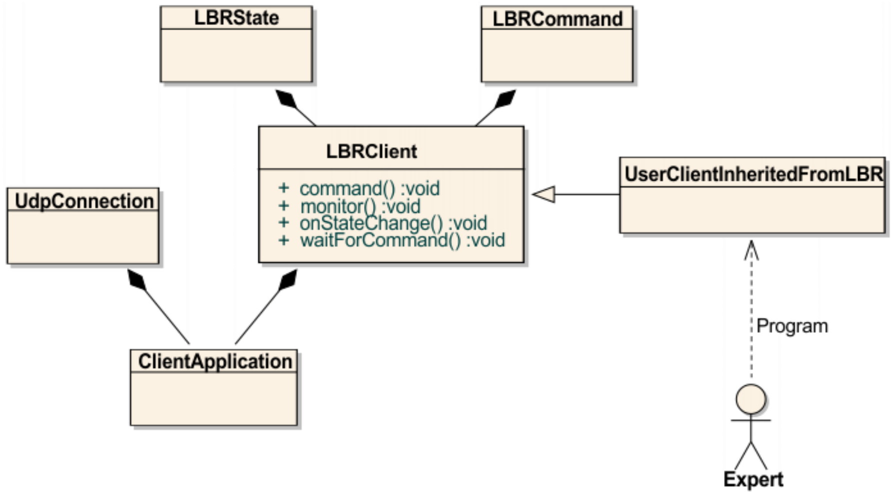


Fig. 3-5: Overview of classes

- The class LBRClient provides the interface methods for the Monitor and Command modes. Separate calculation specifications for path observation and superposition are programmed in a class derived from LBRClient.
- The class LBRState provides the current robot data.
- The class LBRCommand provides the command interface to the robot.
- The class UDPConnection encapsulates the UDP connection.
- The class ClientApplication links communication and calculation specifications.

3.1.7 Communication between the robot controller and FRI client

In the FRI client application, the FRI states are mapped directly to callback methods of the class LBRClient or classes derived from this:

- MONITOR_WAIT corresponds to `cl.monitor();`
- MONITOR_READY corresponds to `cl.monitor();`
- COMMAND_WAIT corresponds to `cl.waitForCommand();`
- COMMAND_ACTIVE corresponds to `cl.command();`
- State transitions are mapped with `cl.onStateChange();`.

Overview The overview shows how the robot controller and FRI client communicate.

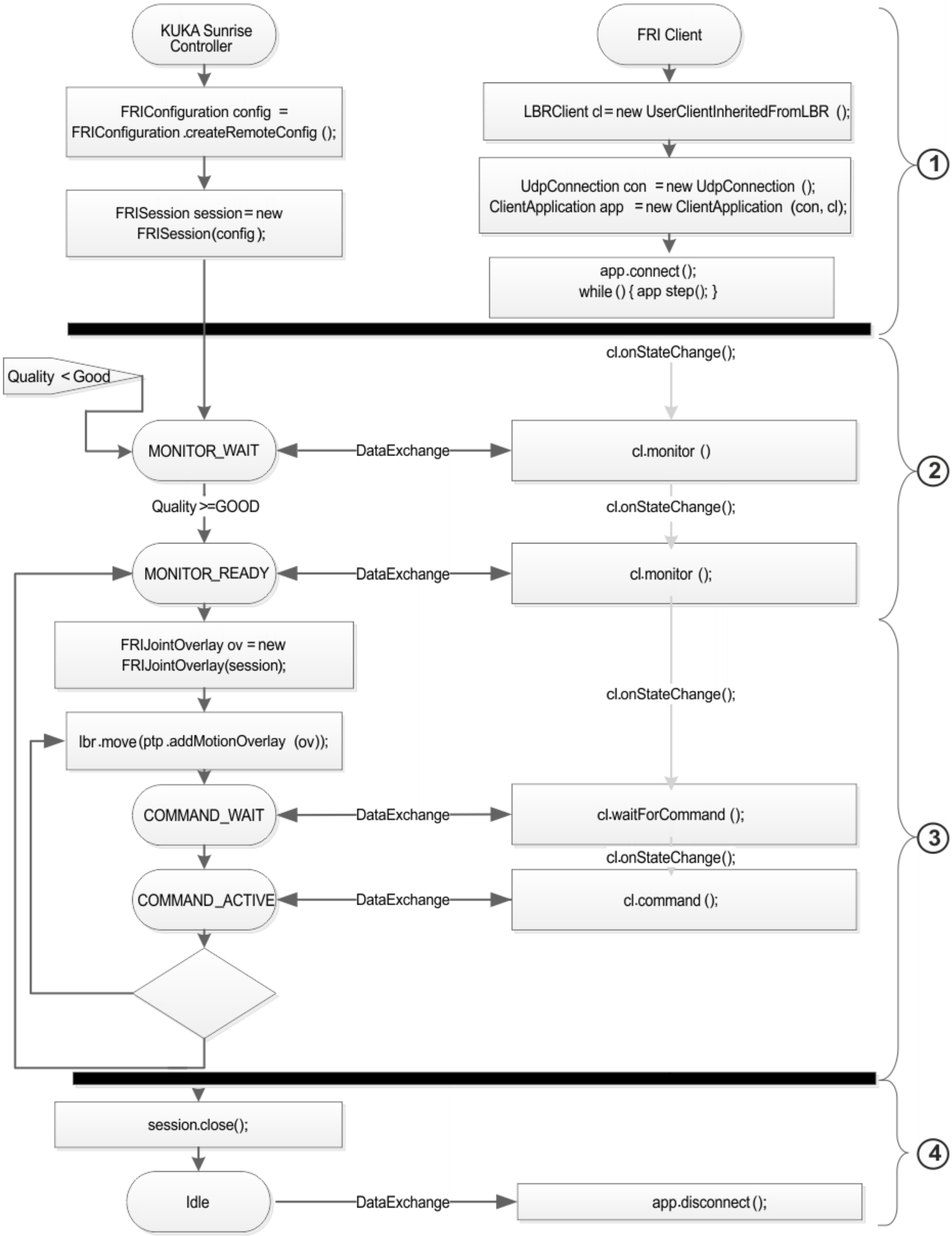


Fig. 3-6: Communication between the robot controller and FRI client

Communication can be subdivided into 4 phases:

Item	Description
1	Configuration and opening of the FRI connection (>>> 7.1.1 "Configuring the FRI connection" Page 25) (>>> 7.1.2 "Initializing and opening the FRI connection" Page 26) Details on the method app.step() can be found here: (>>> 3.1.8 "app.step() method" Page 16)
2	Monitor mode Monitoring of the robot controller by the FRI client (>>> 7.2.1 "Monitor mode application phase" Page 30)
3	Command mode Superposition of the robot path by the FRI client (>>> 7.2.2 "Command mode application phase" Page 31)
4	Closing the FRI connection and ending the FRI client application (>>> 7.1.5 "Closing the FRI connection" Page 30)

3.1.8 app.step() method

During communication between the FRI client and robot controller, the method app.step() is executed cyclically in the FRI client application at the configured send rate.

(>>> 7.1.1 "Configuring the FRI connection" Page 25)

The method app.step() comprises the data exchange with the robot controller and the call of the callback methods which correspond to the current FRI state. The user can change the system behavior with these callback methods. The callback methods are implemented via derivation from the class LBRClient.

(>>> 3.1.6 "Application development in the FRI client" Page 14)

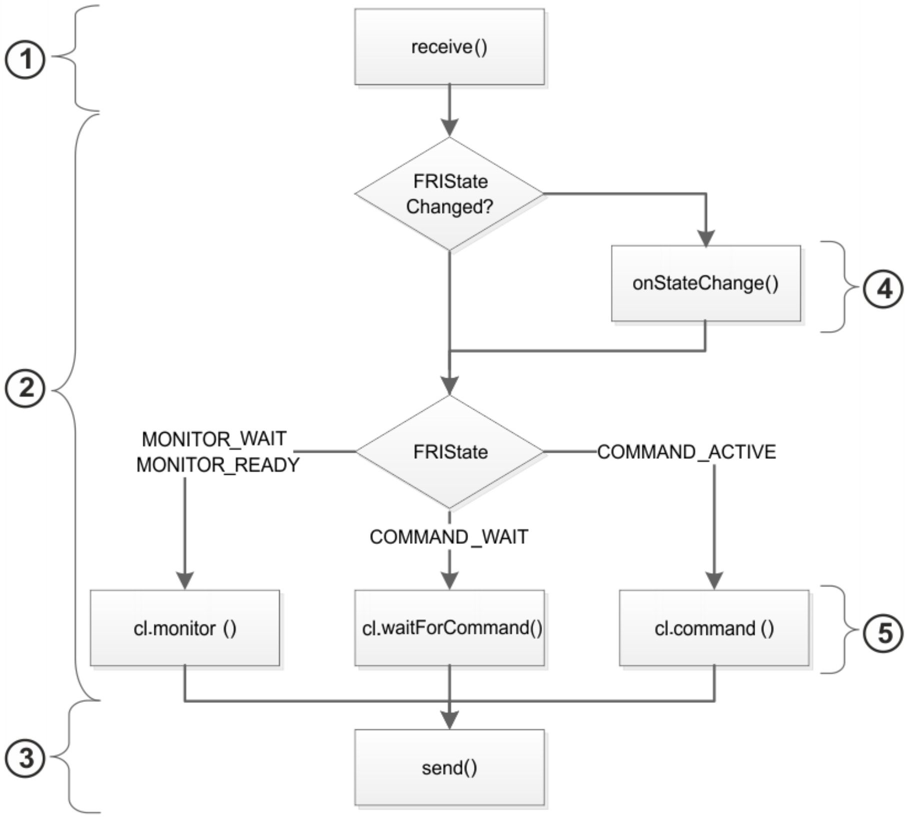


Fig. 3-7: Procedure within the method app.step()

Item	Description
1	Receipt and preparation of the message
2	Evaluation of the current FRI state and calculation of new specified values via calling of callback methods
3	Sending of message including preparation of the message
4	Callback Reaction to change of FRI state (>>> 7.2 "Programming the FRI client application" Page 30)
5	Callback Cyclical calculations in different FRI states (>>> 7.2 "Programming the FRI client application" Page 30)

4 Safety

This documentation contains safety instructions which refer specifically to the product described here. The fundamental safety information for the industrial robot can be found in the “Safety” chapter of the following documentation:

- Operating or assembly instructions of the robot
- Operating and programming instructions for the System Software

The “Safety” chapter in the robot operating or assembly instructions or in the operating and programming instructions for the system software must be observed. Death to persons, severe injuries or considerable damage to property may otherwise result.

The user is responsible for performing his own risk and hazard analysis for use of the robot. This applies in particular if personnel have to enter the robot’s workspace. We recommend having a separate supervisor with a suitable EMERGENCY STOP device. The risk analysis must give special attention to injuries caused by crushing, collision and sharp objects.

NOTICE KUKA Sunrise.Connectivity FRI enables the implementation of closed control loops. The user is responsible for ensuring their stability. Unstable control loops can cause the overall system to vibrate and the robot to move otherwise than expected, although the robot addresses its destination correctly. KUKA is not liable for damage resulting from operation of the robot with unstable control loops.

NOTICE KUKA Sunrise.Connectivity FRI enables the dynamic superposition of specified paths which can overload the robot. KUKA is not liable for damage resulting from operation of the robot with dynamic overloading.

WARNING When using the FRI, the actual system response will differ from the response specified by the manufacturer. The stopping distances and times specified in the operating and assembly instructions of the LBR iiwa are not valid. The user is responsible for determining the application-specific stopping distances and times.

Applications with the FRI in Command mode must always be carefully tested first in Manual Reduced Velocity mode (T1).

When using the FRI in T1 mode, only the velocity of the robot flange is monitored for a maximum of 250 mm/s.

5 Installation

5.1 System requirements

Hardware

- KUKA LBR iiwa
- KUKA Sunrise Cabinet
- FRI client: real-time capable computer with real-time capable operating system

  Recommended hardware: KUKA Office Programming System

Software

- KUKA Sunrise.Workbench 1.5
- KUKA Sunrise.OS 1.5

5.2 Installing Sunrise.Connectivity FRI

Procedure

1. Open the station configuration of the Sunrise project and select the **Software** tab.
2. Select the **Fast Robot Interface** software package for installation:
   - Set the check mark in the **Install** column.
3. Save the station configuration. The system asks whether the modifications to the project should be accepted. Click on **Save and apply**.

   The FRI command library is inserted in the Sunrise project and is available for programming.
4. The C++ SDK is inserted into the Sunrise project as a ZIP file under **FastRobotInterface_Client_Source/FRI-Client-SDK_Cpp.zip**.
   - Unzip the C++ SDK and save it in the C++ development environment (C++ development environment is not included in the scope of suppy).

   (>>> 7.4 "Creating the FRI client application (C++)" Page 37)
5. Install the system software on the robot controller. Changes to the station configuration do not take effect until the software is installed on the robot controller.

6 Configuration

6.1 Ethernet interfaces for the network connection

Description

The robot controller and external system must be connected via a network. The following Ethernet interfaces are available on the robot controller:

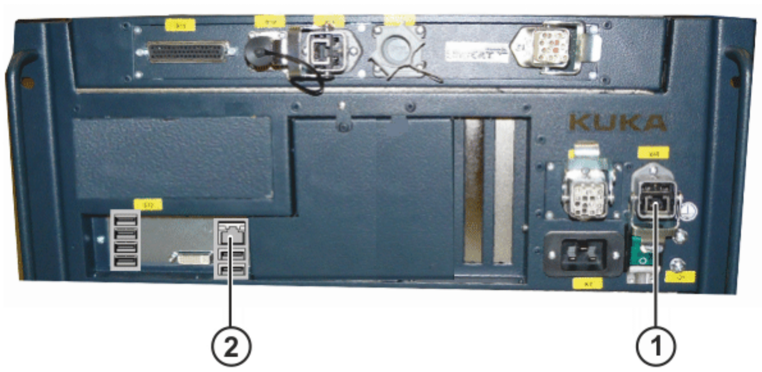


Fig. 6-1: Ethernet interfaces on the KUKA Sunrise Cabinet

Item	Description
1	X66 KUKA Line Interface (KLI) Non real-time capable Ethernet interface (possible at clock rates of >2 ms)
2	LAN Onboard – KUKA Option Network Interface (KONI) Real-time capable Ethernet interface (required at clock rates of 1…2 ms)

> The following network configuration is recommended:
> - Direct point-to-point connection (without switch)
> - Connection via the KONI of the robot controller

The FRI client on the external system is called by the robot controller. The robot controller decides which Ethernet interface is used on the basis of the target address specified in the robot application.

(>>> 7.1.1 "Configuring the FRI connection" Page 25)

6.2 Network settings in Sunrise.Workbench

Description

The network settings can be changed in the station configuration of the Sunrise project.

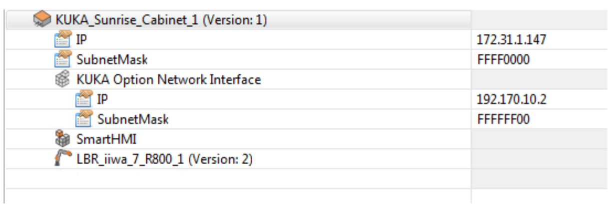


Fig. 6-2: Recommended network settings

> The IP addresses of the KLI and KONI must be in different subnets. If the IP addresses are in the same subnet, the external system and robot controller cannot communicate with each other.

Procedure

1. Open the station configuration and select the **Configuration** tab.
2. Make the desired network settings:
   - IP address and subnet of the KLI under **KUKA_Sunrise_Cabinet_1...** > **IP** / **SubnetMask**
   - IP address and subnet of the KONI under **KUKA_Sunrise_Cabinet_1...** > **KUKA Option Network Interface** > **IP** / **SubnetMask**
3. Save the station configuration. The system asks whether the modifications to the project should be accepted. Click on **Save and apply**.
4. Install the system software on the robot controller. Changes to the station configuration do not take effect until the software is installed on the robot controller.

7 Programming

7.1 Programming the robot application

The FRI connection is configured and managed in the robot application.

7.1.1 Configuring the FRI connection

Description

The FRI connection to an external system is configured via the static method createRemoteConfiguration(…). The method belongs to the class FRIConfiguration.

When the method is called, the name of the robot and the IP address of the external system (FRI client) must be transferred.

Syntax

`FRIConfiguration` *configuration* `=`

`FRIConfiguration.createRemoteConfiguration (`*robot*`,` *IP-Address*`);`

Explanation of the syntax

Element	Description
configuration	Type: FRIConfiguration Name of the FRI configuration
robot	Type: com.kuka.roboticsAPI.deviceModel.Device Robot instance in the application
IP address	Type: String IP address of the external system

Overview

The class FRIConfiguration provides the following “set” methods:

Method	Description
setPortOnController(…)	Defines the port number on the robot controller (type: int) ■ **30200 … 30209** ■ **-1** (default) If the port number **-1** is used, the same port which is configured via setPortOnRemote(…) is set on the robot controller. **Note**: Only change the port number on the robot controller if the application requires different port numbers on both sides of the FRI channel.
setPortOnRemote(…)	Defines the port number on the external system (type: int) ■ **30200 … 30209** Default: **30200** The port number **30200** is the default on both sides of the FRI channel. **Note**: The firewall and network services must ensure the secure transfer of data via the FRI channel.

Method	Description
setReceiveMultiplier(…)	Defines the answer rate factor (type: int) Default: **1** The answer rate factor defines the rate at which the external system is to send data to the robot controller: Answer rate = answer rate factor*send rate
setSendPeriodMilliSec(…)	Defines the send rate (type: int, unit: ms) ■ **1 … 100** Default: **10**

The class FRIConfiguration provides the following “get” methods:

Method	Description
getDevice()	Return value type: String Polls the device to be connected with the FRI (here LBR iiwa)
getHostName()	Return value type: String Polls the IP address of the external system
getPortOnController()	Return value type: int Polls the port number of the robot controller
getPortOnRemote()	Return value type: int Polls the port number on the external system
getReceiveMultiplier()	Return value type: int Polls the answer rate factor
getSendPeriodMilliSec()	Return value type: int Polls the send rate of the robot controller

7.1.2 Initializing and opening the FRI connection

Description

Using an instance of the class FRISession, the FRI connection between the robot controller and external system is opened and Monitor mode is automatically activated.

If Monitor mode is activated, the connection quality is continuously determined and evaluated.

Syntax

`FRISession` *session* `= new FRISession(`*configuration*`);`

Explanation of the syntax

Element	Description
configuration	Type: FRIConfiguration Name of the FRI configuration
session	Type: FRISession Name of the FRI connection

7.1.3 Path superposition with the FRI client

Description

The robot path can be superposed by the FRI client. For this, an instance of the class FRIJointOverlay must be created in the robot application.

This instance must be transferred to every motion to be superposed. The method addMotionOverlay(…) is used for this purpose.

If a linked motion is called with addMotionOverlay(…), the robot controller switches to Command mode.

(>>> 3.1.7 "Communication between the robot controller and FRI client" Page 14)

Once the motion is completed, the robot controller automatically switches back to Monitor mode. If a switch back to Monitor mode is not desired, a further motion must be programmed.

(>>> 7.3 "Motion programming" Page 33)

Syntax

`FRIJointOverlay` *overlay* `= new FRIJointOverlay (`*session*`);`

object`.move|moveAsync(`*motion*`.addMotionOverlay(`*overlay*`));`

Explanation of the syntax

Element	Description
overlay	Type: FRIJointOverlay Name of the superposition
session	Type: FRISession Name of the FRI connection to be used for the superposition
object	Object to be moved (robot, tool or workpiece) The object variable name which was declared and initialized in the robot application must be specified.
motion	Type: PTP, LIN, SPLINE or PositionHold Synchronously or asynchronously executed motion to be superposed

Example

```
FRIJointOverlay ov = new FRIJointOverlay(session);
lbr_iiwa_7_R800_1.moveAsync(ptp(jp1).addMotionOverlay(ov));
lbr_iiwa_7_R800_1.move(ptp(jp2).addMotionOverlay(ov));
```

7.1.4 Synchronization with the FRI client

Description

FRI serves as an interface between the robot application and the FRI client application. The FRI connection quality is used for the synchronization of the two application components. Motion commands with FRI superposition may only be sent if the connection quality is sufficient (≥ GOOD).

The robot application and FRI client application can be synchronized in different ways:

- By polling the connection quality
- By monitoring the connection quality
- Using the method await(…)

7.1.4.1 Polling the connection quality and time statistic

Description

The method getFRIChannelInformation() can be used to poll the following information and save it in a variable of type FRIChannelInformation:

- Jitter, latency and quality of the FRI connection
- Time stamp of the poll

Syntax

`FRIChannelInformation` *chanInfo* `=`

session`.getFRIChannelInformation();`

Explanation of the syntax

Element	Description
chanInfo	Type: FRIChannelInformation Variable in which the data for the FRI connection are saved
session	Type: FRISession Name of the FRI connection from which the data are polled

Overview

The class FRIChannelInformation provides the following “get” methods:

Method	Description
getLatency()	Return value type: double; unit: ms Poll of the latency
getJitter()	Return value type: double; unit: ms Poll of the jitter
getQuality()	Return value type: Enum of type FRIConnectionQuality Poll of the connection quality ■ FRIConnectionQuality.POOR ■ FRIConnectionQuality.FAIR ■ FRIConnectionQuality.GOOD ■ FRIConnectionQuality.EXCELLENT
getTimeStampMillis()	Return value type: long; unit: ms Poll of the time stamp

7.1.4.2 Monitoring the connection quality

Description

A listener of type IFRISessionListener can be used to monitor the connection quality. If the connection quality changes, the method onFRIConnectionQualityChanged(…) is called.

The listener must be registered with registerFRISessionListener(…). It can be deactivated with unregisterFRISessionListener(…).

Syntax

```
IFRISessionListener listener = new IFRISessionListener() {
@Override
public void onFRIConnectionQualityChanged(
FRIChannelInformation friChannelInformation) {
// Reaction to quality change
}
};
session.registerFRISessionListener(listener};
```

Explanation of the syntax

Element	Description
listener	Type: IFRISessionListener Name of the listener object
session	Type: FRISession Name of the FRI connection for which the listener is registered

Element	Description
Input parameter of the listener method:	
`fri ChannelIn formation`	Type: FRIChannelInformation Contains the following information about the FRI connection: ■ Jitter, latency and quality of the FRI connection ■ Time stamp at which the quality change occurred

Example

```
IFRISessionListener listener = new IFRISessionListener(){
@Override
public void onFRIConnectionQualityChanged(
   FRIChannelInformation friChannelInformation){
      System.out.println("QualityChangedEvent" +
friChannelInformation);
   }
};
session.registerFRISessionListener(listener);
```

7.1.4.3 Wait function for switching to Command mode

Description

With await(…), the robot application is stopped until it is possible to switch to Command mode (connection quality ≥ GOOD) or a specified wait time has expired.

The method belongs to the class FRISession. If the defined wait time expires without the robot controller changing to Command mode, the exception java.util.concurrent.TimeoutException is triggered.

Using a try-catch block, the exception can be executed without the application being aborted.

Syntax

`try{`

session`.await(`*timeout*`,` *timeUnit*`)`

`}`

`catch(TimeoutException toe){`

// Code for handling the timeout

`}`

`< finally{`

// Final treatment (optional)

`}>`

Explanation of the syntax

Element	Description
session	Type: FRISession Name of the FRI connection
timeout	Type: long Maximum wait time

Element	Description
timeUnit	Type: Enum of type TimeUnit Unit of the given wait time. The Enum is contained by default in the Java libraries.
`Timeout Exception toe`	The error data type `TimeoutException` is handled in the catch block. The parameter `toe` can be used to poll information about the error which occurred.

Example

```
FRISession session = new FRISession(config);

try {
   session.await(1000, TimeUnit.SECONDS);
      FRIJointOverlay ov = new FRIJointOverlay(session);

      lbr_iiwa_7_R800_1.moveAsync(ptp(jp1).addMotionOverlay(ov));
      lbr_iiwa_7_R800_1.move(ptp(jp2).addMotionOverlay(ov));
}

catch(TimeoutException toe){
   System.err.println("Timeout occured - sessionstate" +
   session.getFRIChannelInformation());
}
finally{
   session.close();
}
```

7.1.5 Closing the FRI connection

Description

The FRI connection is closed with close(). The method belongs to the class FRISession.

Syntax

`FRISession` *session*`.close();`

Explanation of the syntax

Element	Description
session	Type: FRISession Name of the FRI connection

7.2 Programming the FRI client application

In the FRI client application, Monitor mode and Command mode are implemented on an external system. For this, class methods corresponding to the different FRI states and state transitions are implemented.

(>>> 3.1.4 "Interaction between Monitor mode and Command mode" Page 11)

The class methods are automatically called during program execution as a function of the current FRI state.

7.2.1 Monitor mode application phase

In Monitor mode, the robot controller sends information about the current state of the robot in real time. This information can be used for process evaluation, for example.

Overview The following methods are available in Monitor mode (class LBRClient):

Method	Description
virtual void onStateChange (ESessionState oldState, ESessionState newState)	Callback method for displaying FRI state changes
virtual void monitor()	Cyclical callback method in the states MONITOR_WAIT and MONITOR_READY
const LBRState & robotState()	Read access to the current runtime data of the robot

7.2.2 Command mode application phase

In Command mode, the robot path can be changed from the FRI client.

The switch to Command mode is initiated in the robot application.

(>>> 7.1.3 "Path superposition with the FRI client" Page 26)

Precondition

- In order to be able to switch from Monitor mode to Command mode, the connection quality must be GOOD or EXCELLENT.

Overview The following methods are available in Command mode (class LBRClient):

Method	Description
virtual void onStateChange (ESessionState oldState, ESessionState newState)	Callback method for displaying FRI state changes
virtual void waitForCom-mand()	Cyclical callback method in the state COMMAND_WAIT
virtual void command()	Cyclical callback method in the state COMMAND_ACTIVE
const LBRState & robotState()	Read access to the current runtime data of the robot
LBRCommand & robotCommand()	Write access for changing the commandable runtime data

7.2.3 Methods of the class LBRState (polling runtime data)

The class LBRState provides the following methods for polling runtime data from the robot controller:

Method	Description	Robot controller
static int getNumberOfJoints()	Number of axes	LBR.getJointCount()
double getSampleTime() const	Cycle time in seconds	FRIConfiguration. getSendPeriodMilliSec() / 1000.0
ESessionState getSessionState() const	State of the FRI connection	———
EConnectionQuality getConnectionQuality() const	Quality of the FRI connection	FRISession. getFRIChannelInformation(). getQuality()
ESafetyState getSafetyState() const	Status of the safety signal “Safety stop”	LBR.getSafetyState()
EOperationMode getOperationMode() const	Operating mode (T1, T2, AUT)	LBR.getOperationMode()
EDriveState getDriveState() const	Activity of the drives (switched on/off)	LBR.hasActiveMotion Command()

Method	Description	Robot controller
unsigned int getTimestamp Sec() const	Time stamp of the data transmission in [s] (time of registration of the measured value)	———
unsigned int getTimestamp NanoSec() const	Fine fraction of the time stamp of the data transmission in [s] (time of registration of the measured value)	———
const double getMeasured JointPosition() const	Axis-specific actual position [rad]	LBR.getCurrentJointPosition()
const double getCommanded JointPosition() const	Axis-specific setpoint position in [rad]	LBR.getCommanded JointPosition()
const double getMeasured Torque() const	Axis-specific actual torque in [Nm]	LBR.getMeasuredTorque()
const double getCommanded Torque() const	Axis-specific setpoint torque in [Nm]	———
const double getExternal Torque() const	Axis-specific external actual torque in [Nm]	LBR.getExternalTorque()
const double getIpo JointPosition() const	Interpolated position as basis for path superposition in [rad]	———

7.2.4 Methods of the class LBRCommand (modifying runtime data)

The class LBRCommand provides the following methods for modifying runtime data on the robot controller:

Method	Description	Robot controller
void setJointPosition(const double values)	Setpoint position in [rad]	———

7.2.5 Methods of the class LBRClient (creating real-time functionalities)

In order to create a real-time application, a new class must be derived from the class LBRClient:

```
class LBRJointSineOverlayClient:public LBRClient
```

The following methods can be implemented in order to realize the real-time functionalities:

Method	Description	Default behavior
virtual void onStateChange (ESessionState oldState, ESessionState newState)	Display of the FRI state change	———
virtual void monitor()	Observation in Monitor mode MONITOR_WAIT and MONITOR_READY states	Mirroring of the commanded position
virtual void waitForCommand()	Coordination with the robot controller for activating the state COMMAND_WAIT	Mirroring of the interpolated position
virtual void command()	Path superposing in the state COMMAND_ACTIVE	Mirroring of the interpolated position

7.2.5.1 Integrating real-time functionalities into the FRI client application

In order to integrate the newly implemented real-time functionalities into the FRI client application, an instance of the new class must be transferred to the constructor of the class ClientApplication.

Example

```
UDPConnection conn;
LBRJointSineOverlayClient cl;
ClientApplication app(conn, cl);
```

7.3 Motion programming

The FRI client application can directly influence the robot motion in Command mode. This can lead to an intentional deviation of the robot motion from the originally programmed path. The path deviation also influences subsequent motion sections and the pause behavior.

> A programmed path consisting of several motion commands (`moveAsync` with FRIJointOverlay) is represented as a single continuous motion sequence in the FRI client application. It is advisable to complete such motion sequences with a synchronous motion command (`move` with FRIJointOverlay).

The figure shows the geometric effects on the programmed path when approximated asynchronous or synchronous motion commands with FRIJointOverlay are used:

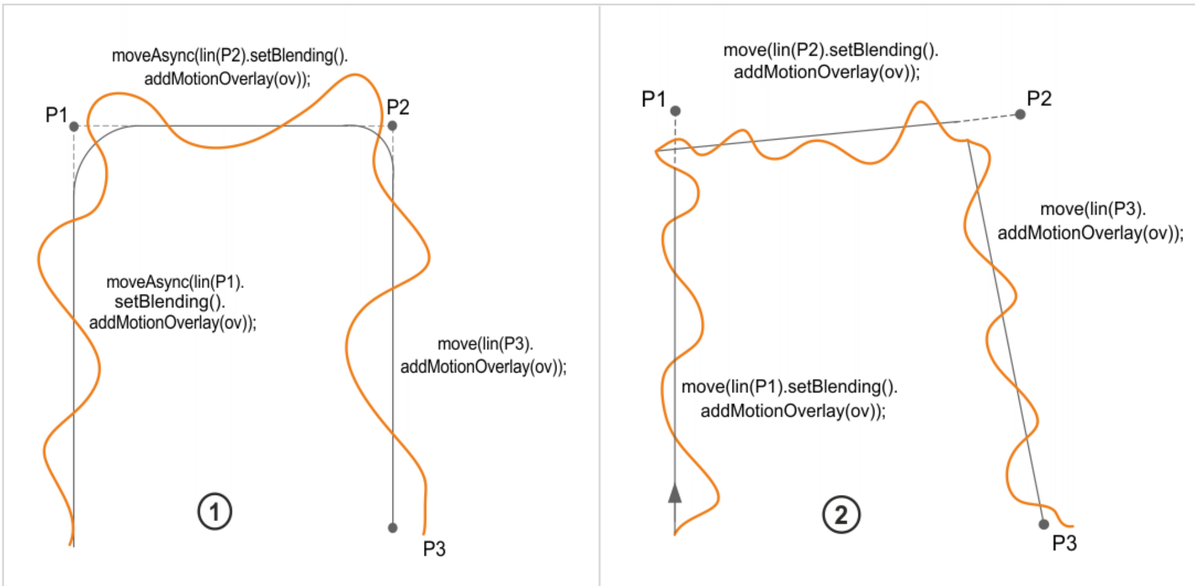


Fig. 7-1: Comparison of the geometric effects of the motions

Item	Description
1	FRIJointOverlay when using the commands `moveAsync` and `move`: ■ The motion is continuously executed and approximated. ■ Command mode is not interrupted.
2	FRIJointOverlay without `moveAsync`: ■ In each case, the programmed motion stops at the approximation point with exact positioning. ■ The subsequent motion starts from the last commanded position. ■ Command mode is ended after each motion section and reinitialized for the subsequent motion.

7.3.1 Asynchronous motion programming

If an asynchronous motion command with FRIJointOverlay is replaced by a further motion command with FRIJointOverlay, the command flow in the FRI client application is not interrupted. The motion commands are regarded as one motion sequence. FRI remains in the state COMMAND_ACTIVE during the entire motion.

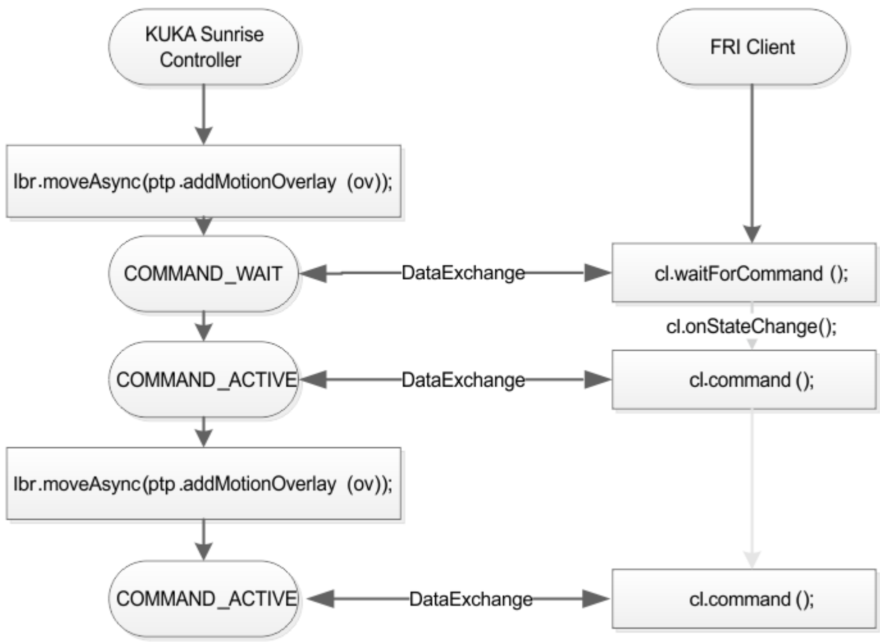


Fig. 7-2: State sequence with asynchronous motion programming

Path

The figure (>>> Fig. 7-3) shows the fundamental path in the case of approximation between 2 asynchronous motions. The gray line shows the path of the interpolator and the orange line shows a possible path due to the FRI superposition by the external system.

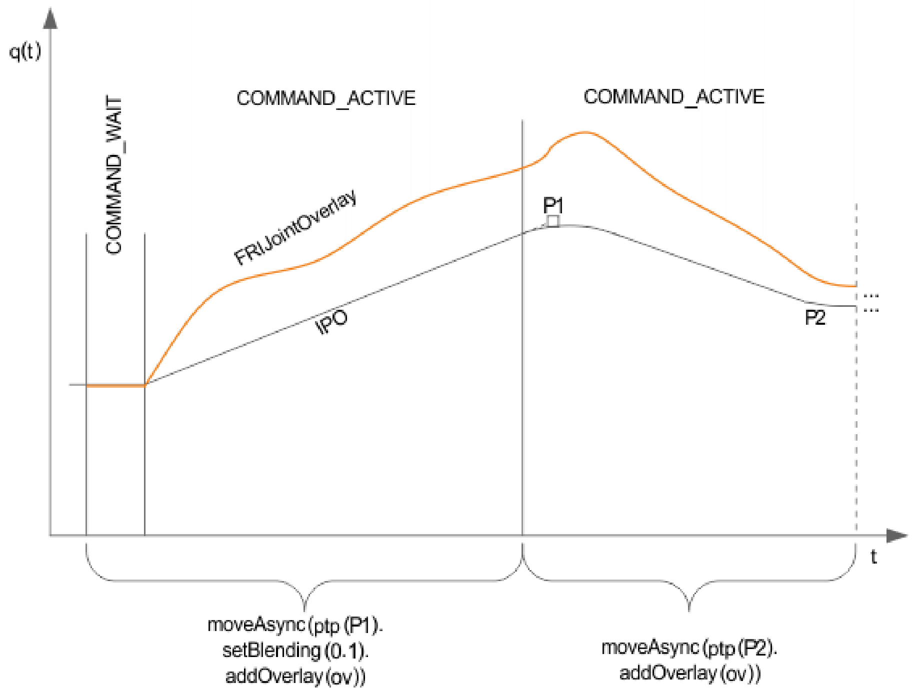


Fig. 7-3: Path with asynchronous motion programming

Approximate positioning

Asynchronous motions can be approximated such that a continuous motion is performed without exact positioning. Approximate positioning also works in Command mode (>>> Fig. 7-3).

Command mode can be activated in the middle of an approximated motion sequence without the need for exact positioning. However, leaving Command mode always results in exact positioning, even if the programmed path would permit approximation.

7.3.2 Synchronous motion programming

A synchronous motion command with FRIJointOverlay is always followed by exact positioning and a switch from Command mode to Monitor mode (regardless of the subsequent motion type). The following motions start from the last position commanded by the FRI client and not from the programmed end point of the previous motion.

If a further motion command with FRIJointOverlay follows, the state COMMAND_WAIT is executed again and the FRI client application must synchronize with the programmed path once again. Only after the synchronization has been performed is it possible to switch to COMMAND_ACTIVE.

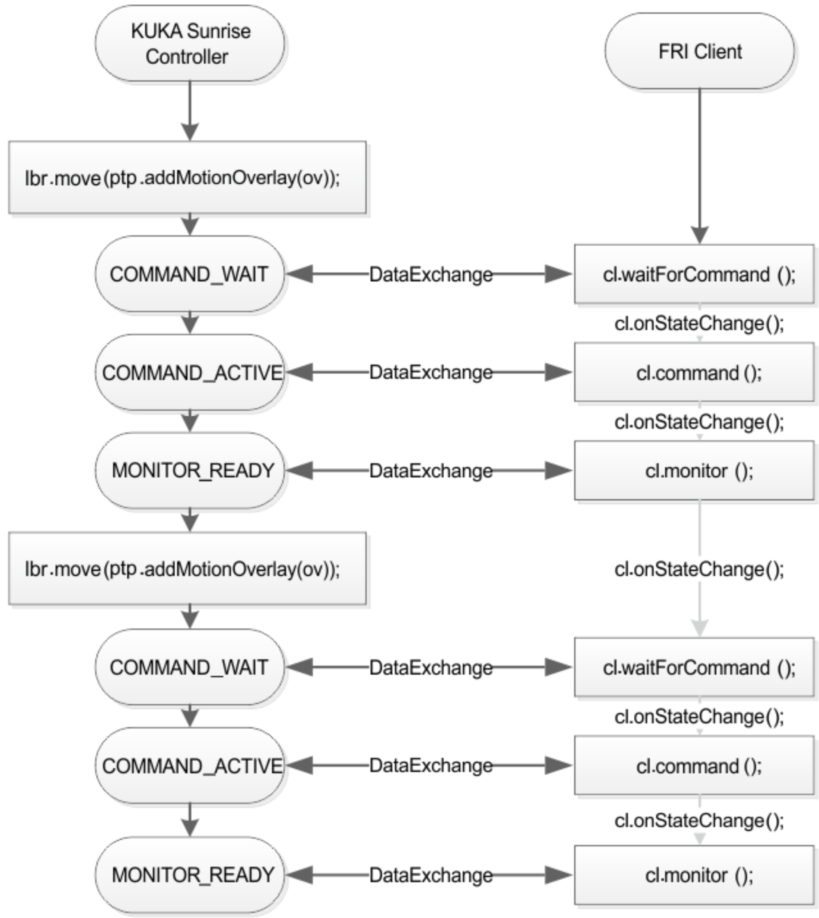


Fig. 7-4: State sequence with synchronous motion programming

Path

The figure shows the fundamental path between 2 synchronous motions. The gray line shows the path of the interpolator and the orange line shows a possible path due to the FRI superposition by the external system. Exact positioning is executed between the motions.

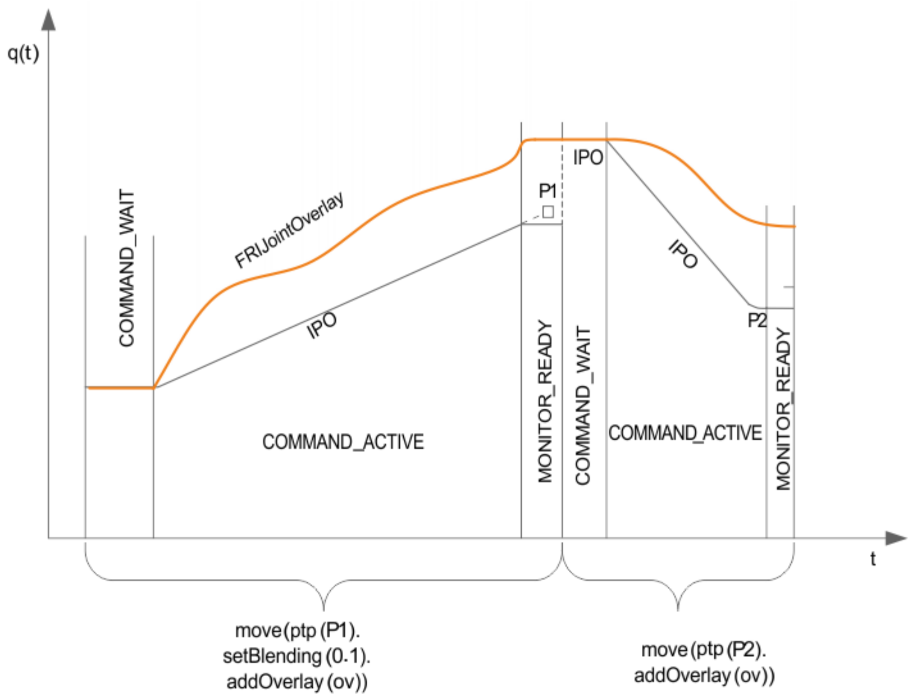


Fig. 7-5: Path with synchronous motion programming

7.3.3 Pause behavior in Command mode

If the robot application is paused while Command mode is active (for example by pressing the STOP key), the FRI client will not accept any further position specifications. The motion is stopped and the FRI switches to the state MONITOR_READY.

Pressing the Start key causes the robot to be automatically repositioned on the originally programmed path. The motion then restarts from the programmed path. The state COMMAND_WAIT is executed again and the FRI client application must synchronize with the programmed path. Only then is it possible to switch to the state COMMAND_ACTIVE and resume the motion.

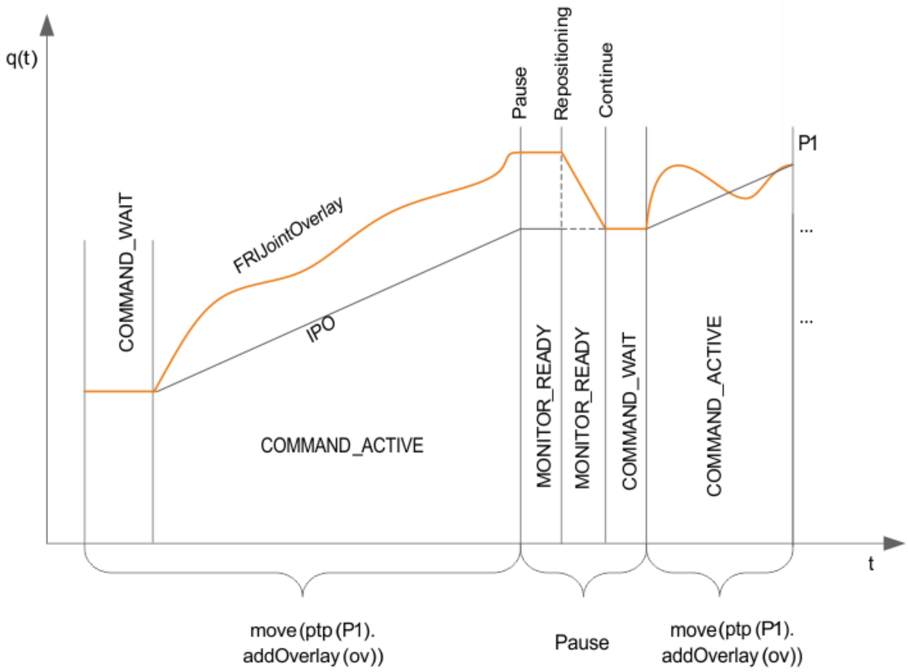


Fig. 7-6: Pause behavior in Command mode

7.4 Creating the FRI client application (C++)

Description

To create an FRI client application, the C++ Software Development Kit (SDK) is required.

The C++ SDK is located in the Sunrise project as a ZIP file under **FastRobotInterface_Client_Source/FRI-Client-SDK_Cpp.zip**.

The following directories are created during unzipping:

- **build**: Build environments for the SDK and example programs
- **doc**: Documentation for the programming interface
- **example**: FRI client sample application LBRJointSineOverlay
- **include**: Programming interface for the FRI client application
- **src**: Source code required for building the SDK library

In order to ensure that the SDK has the greatest possible platform independence, KUKA supplies the library sources as source code. The library must be compiled once for the desired target platform before the application is created. The SDK provides sample build environments for Windows and Linux.

It is advisable to always use the supplied SDK programming interface for FRI client applications. Only those FRI client applications created in this way will remain compatible with future FRI developments.

A detailed description of the programming interface is stored in the HTML documentation **doc/html/index.html**.

7.4.1 Build process with Visual Studio (Windows)

Description

The SDK library and sample applications exist as independent projects and can be opened in Visual Studio.

Precondition

- Windows operating system
- Microsoft Visual Studio 2010 SP1

Procedure

1. Open the Visual Studio solution **build/MSVisualStudio2010/FRIClientSDK.sln**.
2. Build the SDK library as a binary for Win32 in the solution (project **libFRIClient**).

7.4.2 Build process with GNUMake (Linux)

Description

Generic makefiles for building the SDK library are located in the directory **build/GNUMake**. The build process is initiated using the program `make`.

New applications can be added to the automatic build process by expanding the variable EXAMPLE_DIRS in the file **build/GNUMake/Makefile**. Alternatively, the applications can be compiled directly from their source directories by calling `make`.

Precondition

- Linux operating system

Procedure

- Call `make` in the directory **build/GNUMake**.

  The SDK library and the sample applications are built one after the other. Once the build process has been successfully completed, the compiled sample applications can be found in the directory **bin**.

7.4.3 Build process for further platforms

Description

The supplied GNU makefiles are optimized for Linux. In order to build on other platforms, the file **build/GNUMake/tools.mak** can be adapted.

Procedure

1. Enter the build tools belonging to the platform, e.g. compiler, linker, etc., in the file "tools.mak".
2. Set the required compiler and linker options.

Platform-dependent adaptations

The source code of the SDK is as C++-compliant as possible. Depending on the platform, it may be necessary to make adaptations in certain files.

UdpConnection.h, UdpConnection.cpp:

If UDP sockets are used for the FRI channel, it may be necessary to use platform-specific headers. Furthermore, depending on the platform, the compiler option **–DHAVE_SOCKLEN_T** must be set or removed in the file **build/GNUMake/tools.mak**.

pb_syshdr.h:

The header files from nanopb sources are not available on all platforms:

- stdint.h
- stddef.h
- stdlib.h
- stdbool.h
- string.h

For this reason, the nanopb sourcs contain the file "pb_syshdr.h". This file provides the appropriate functionalities for platforms which do not recognize these headers. The availability of the headers is configured via compiler options.

By setting or removing the following compiler options in the file **build/GNU-Make/tools.mak**, it is possible to determine which header files are available on the platform used:

- -DHAVE_STDINT_H (stdint.h)
- -DHAVE_STDDEF_H (stddef.h)
- -DHAVE_STDLIB_H (stdlib.h)
- -DHAVE_STDBOOL_H (stdbool.h)
- -DHAVE_STRING_H (string.h)

8 Troubleshooting

8.1 Problems with connection quality and real-time capability

Error	Cause	Remedy
The quality of the FRI connection is consistently POOR (state MONITOR_WAIT).	The network is incorrectly configured.	■ Check the network configuration. ■ Check the connection between the robot controller and the external system (ping function).
	Plug connection of the KLI or KONI does not match the assigned IP address.	■ Check the plug connection. ■ Check the network configuration.
	The firewall is blocking the connection.	Check the firewall and set it correctly.
The quality of the FRI connection is consistently FAIR (state MONITOR_WAIT).	Send/receive timing is not complied with.	Check the timing on the FRI client and user slower clock rates if necessary.
The quality of the FRI connection changes constantly between FAIR, GOOD and EXCELLENT.	The switch used is not real-time capable.	Use a direct point-to-point connection with a Cat6 cable.
	FRI client application is not real-time capable.	■ Check the FRI client application. ■ Relocate processing-intensive operations to the background task. ■ Use slower clock rates.
	The external computer used is not real-time capable.	■ Check BIOS settings. ■ Use a real-time capable computer (recommended hardware: KUKA Office Programming System).
	The operating system on the external computer used is not real-time capable.	Use a real-time capable operating system.
	The network driver of the operating system on the external computer used is not real-time capable.	Use a real-time capable network driver.
	The network connection via the KLI is not real-time capable.	Network connection via the KONI
The robot controller is no longer available in the network following installation of the system software.	The KLI and KONI are in the same subnet.	Contact KUKA Service. (>>> 9 "KUKA Service" Page 45)

8.2 Runtime problems in Command mode

Error	Cause	Remedy
The makes jerky motions or sags during the transition to COMMAND_ACTIVE.	The FRI client application is not correctly installed.	Check the implementation of the following methods in the FRI client application: ■ onStateChange() ■ waitForCommand()
Runtime Exception CK_COMPOUND_RETURN_ERROR: ■ Connection quality switches to POOR. ■ State switches from COMMAND_ACTIVE to MONITOR_WAIT.	Computing time of the FRI client in `command()` is too long.	It is advisable to call the calculations performed in the COMMAND_ACTIVE phase while still in the MONITOR_READY phase. This ensures that an increased computing time requirement is already recognized in Monitor mode. 1. Estimate the computing time requirement via FRIChannelInformation.getLatency(). 2. Reduce the send timing accordingly. 3. Reduce the computing time requirement.
	The computing time in the FRI client is temporarily too long.	In the FRI client application, check the following methods for potential deadlocks (e.g. new, malloc, semaphores): ■ onStateChange() ■ command()
	The quality of the FRI connection is insufficient or unstable.	(>>> 8.1 "Problems with connection quality and real-time capability" Page 41)
Runtime Exception CK_COMPOUND_RETURN_ERROR: ■ The connection quality is GOOD or EXCELLENT. ■ State MONITOR_READY	The specified value is outside the allowed axis range or the allowed velocity and acceleration.	Check the specified value.
The robot stops abruptly, FRI connection is no longer available.	FRI connection was closed prematurely.	■ Check the program sequence. ■ Wait for asynchronous motions. ■ Complete the motion sequence with a synchronous motion command.

8.3 Exceptions

Java exception	Method	Cause / remedy
IllegalArgumentException	FRIConfiguration.create RemoteConfiguration(…)	Invalid parameterization (>>> 7.1.1 "Configuring the FRI connection" Page 25)
	■ FRIConfiguration.set PortOnController(…) ■ FRIConfiguration.set PortOnRemote(…) ■ FRIConfiguration.set ReceiveMultiplier(…) ■ FRIConfiguration.set SendPeriodMilliSec(…)	Invalid parameterization (>>> 7.1.1 "Configuring the FRI connection" Page 25)
	FRISession.await(…)	Invalid parameterization (>>> 7.1.4.3 "Wait function for switching to Command mode" Page 29)
	FRISession.FRISession(…)	FRI configuration invalid (>>> 7.1.2 "Initializing and opening the FRI connection" Page 26)
	■ FRISession.register FRISessionListener(…) ■ FRISession.unregister FRISessionListener(…)	Invalid listener (>>> 7.1.4.2 "Monitoring the connection quality" Page 28)
	FRISession.close()	FRI connection has already been closed. Remedy: Check the program sequence.
IllegalStateException	FRISession.await(…)	FRI connection is closed. Remedy: Check the program sequence.
	FRISession.FRISession(…)	FRI connection could not be opened. Remedy: Check the network connection (IP address, port) (>>> 8.1 "Problems with connection quality and real-time capability" Page 41)
TimeoutException	FRISession.await(…)	Wait time has expired before the state MONITOR_READY was reached. ■ Extend the wait time. ■ Start the FRI client. (>>> 8.1 "Problems with connection quality and real-time capability" Page 41)

Java exception	Method	Cause / remedy
NotSupportedException	addMotionOverlay(FRIJointOverlay)	The motion contains more than one FRIJointOverlay.
		A control type not supported by the FRI has been used, e.g. an impedance controller. Only position-controlled motions can be superposed (control type PositionControl-Mode).
		A motion type not supported by the FRI has been used, e.g. CIRC or MotionBatch. The following motion types are supported: PTP, LIN, SPLINE, PositionHold.
IllegalArgumentException	addMotionOverlay(FRIJointOverlay)	The FRI connection used does not match the robot being moved. Remedy: When configuring the FRI connection, select the appropriate robot instance. (>>> 7.1.1 "Configuring the FRI connection" Page 25) (>>> 7.1.3 "Path superposition with the FRI client" Page 26)

8.4 Errors in the FRI client C++ application

Fault	Cause	Remedy
The FRI client application library does not compile.	The compiler settings (defines) do not match the environment.	Determine and set the compiler settings for "nanopb". See also C++ SDK "src/ nanopb-0.2.8/pb_syshdr.h".
The FRI client cannot open the connection.	The network is incorrectly configured.	Check the network configuration. (>>> 8.1 "Problems with connection quality and real-time capability" Page 41)
	The firewall is preventing the connection.	Modify the firewall settings.
	The port is already assigned.	Enable the port, e.g. by ending the running application.
The user-specific FRI client application does not work, but the supplied FRI client sample application **example/LBRSineOverlay** does.	There is a real-time violation.	Check the user-specific FRI client application.
	The specified value is outside the permitted axis range.	
	The specified value is outside the permitted velocity or acceleration.	

9 KUKA Service

9.1 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

9.2 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 Roboter GmbH Zugspitzstr. 140 86165 Augsburg Germany Tel. +49 821 797-4000 Fax +49 821 797-1616 info@kuka-roboter.de www.kuka-roboter.de

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 Robotics 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 Robotics Japan K.K. YBP Technical Center 134 Godo-cho, Hodogaya-ku Yokohama, Kanagawa 240 0005 Japan Tel. +81 45 744 7691 Fax +81 45 744 7696 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 Roboter CEE GmbH Gruberstraße 2-4 4020 Linz Austria Tel. +43 7 32 78 47 52 Fax +43 7 32 79 38 80 office@kuka-roboter.at www.kuka.at

Poland	KUKA Roboter Austria GmbH 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 Sistemas de Automatización S.A. Rua do Alto da Guerra n° 50 Armazém 04 2910 011 Setúbal Portugal Tel. +351 265 729780 Fax +351 265 729782 kuka@mail.telepac.pt
Russia	KUKA Robotics RUS Werbnaja ul. 8A 107143 Moskau Russia Tel. +7 495 781-31-20 Fax +7 495 781-31-19 info@kuka-robotics.ru www.kuka-robotics.ru
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 Schweiz AG Industriestr. 9 5432 Neuenhof Switzerland Tel. +41 44 74490-90 Fax +41 44 74490-91 info@kuka-roboter.ch www.kuka-roboter.ch

Spain	KUKA Robots IBÉRICA, S.A. Pol. Industrial Torrent de la Pastera Carrer del Bages s/n 08800 Vilanova i la Geltrú (Barcelona) Spain Tel. +34 93 8142-353 Fax +34 93 8142-950 Comercial@kuka-e.com www.kuka-e.com
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 Robot Automation Taiwan Co., Ltd. No. 249 Pujong Road Jungli City, Taoyuan County 320 Taiwan, R. O. C. Tel. +886 3 4331988 Fax +886 3 4331948 info@kuka.com.tw www.kuka.com.tw
Thailand	KUKA Robot Automation (M)SdnBhd Thailand Office c/o Maccall System Co. Ltd. 49/9-10 Soi Kingkaew 30 Kingkaew Road Tt. Rachatheva, A. Bangpli Samutprakarn 10540 Thailand Tel. +66 2 7502737 Fax +66 2 6612355 atika@ji-net.com www.kuka-roboter.de
Czech Republic	KUKA Roboter Austria GmbH Organisation Tschechien und Slowakei Sezemická 2757/2 193 00 Praha Horní Počernice Czech Republic Tel. +420 22 62 12 27 2 Fax +420 22 62 12 27 0 support@kuka.cz

Hungary	KUKA Robotics Hungaria 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 Automation + Robotics Hereward Rise Halesowen B62 8AN UK Tel. +44 121 585-0800 Fax +44 121 585-0900 sales@kuka.co.uk

Index

W