Application manual

Robot Reference Interface

Controller software IRC5
RobotWare 5.12

Application manual

Robot Reference Interface

RobotWare 5.12

Document ID: 3HAC031973-001

Revision: A

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

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

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

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

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

ABB AB
Robotics Products
721 68 Västerås
Sweden

Overview

About this manual

This manual explains the basics of how and when to use the RobotWare option *Robot Reference Interface*.

Usage

This manual can be used either as a reference to find out if an option is the right choice for solving a problem, or as a description of how to use an option. Detailed information regarding syntax for RAPID routines and configuration of system parameters is not described here, but can be found in the respective reference manual.

Who should read this manual?

This manual is mainly intended for robot programmers.

Prerequisites

The reader should be:

- Familiar with industrial robots and their terminology.
- Familiar with the RAPID programming language.
- Familiar with RobotStudio and how to work with system parameters.

References

Reference	Document ID
General safety information	3HAC031045-001
Technical reference manual - RAPID Instructions, Functions and Data types	3HAC16581-1
Operating manual - RobotStudio	3HAC032104-001
Sample code available on www.abb.com/robotics (ABB Library)	

Revisions

Revision	Description
-	First edition
A	Released with RobotWare 5.12. Minor corrections. `SiConnect` has a new switch `\NoStop`. Updates in `sensor`. `SiTool` and `SiWobj` are no longer needed for calculations. Current tool and workobject are used instead.

Product documentation, M2004

Categories for robot documentation

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

All documents listed can be ordered from ABB on a DVD. The documents listed are valid for M2004 robot systems.

Product manuals

All hardware, robots and controllers, will be delivered with a **Product manual** that contains:

- Safety information.
- Installation and commissioning (descriptions of mechanical installation, electrical connections).
- Maintenance (descriptions of all required preventive maintenance procedures including intervals).
- Repair (descriptions of all recommended repair procedures including spare parts).
- Additional procedures, if any (calibration, decommissioning).
- Reference information (article numbers for documentation referred to in Product manual, procedures, lists of tools, safety standards).
- Part list.
- Foldouts or exploded views.
- Circuit diagrams.

Technical reference manuals

The technical reference manuals describe the robot software in general and contain relevant reference information.

- **RAPID Overview**: An overview of the RAPID programming language.
- **RAPID Instructions, Functions and Data types**: Description and syntax for all RAPID instructions, functions, and data types.
- **RAPID Kernel**: A formal description of the RAPID programming language.
- **System parameters**: Description of system parameters and configuration workflows.

Application manuals

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

An application manual generally contains information about:

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

Continues on next page

Operating manuals

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

The group of manuals includes (among others):

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

Safety

Safety of personnel

A robot is heavy and extremely powerful regardless of its speed. A pause or long stop in movement can be followed by a fast hazardous movement. Even if a pattern of movement is predicted, a change in operation can be triggered by an external signal resulting in an unexpected movement.

Therefore, it is important that all safety regulations are followed when entering safeguarded space.

Safety regulations

Before beginning work with the robot, make sure you are familiar with the safety regulations described in the manual *General safety information*.

1 Introduction

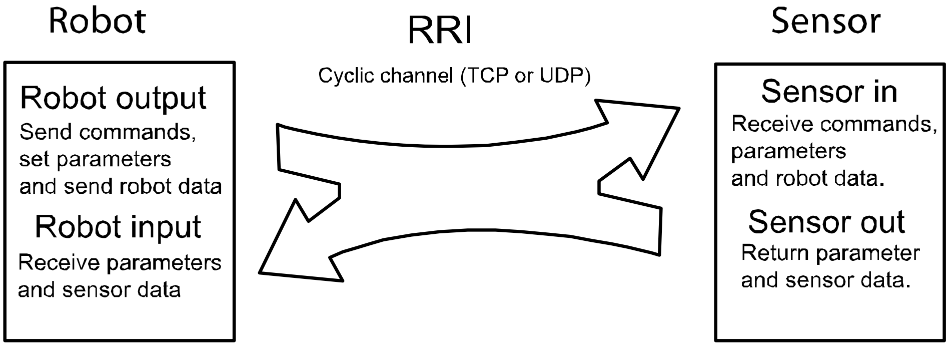
1.1. Introduction to Robot Reference Interface

Introduction

Robot Reference Interface is an option, supporting data exchange on the cyclic channel. It provides the possibility to periodically send planned and actual robot position data as well as the exchange of other RAPID variables. The message contents are represented in XML format and are configured using appropriate sensor configuration files.

Robot Reference Interface

The cyclic communication channel (TCP or UDP) can be executed in the high-priority network environment of the IRC5 Controller which ensures a stable data exchange up to 250Hz.


Robot
RRI
Sensor
Cyclic channel (TCP or UDP)
Robot output
Send commands, set parameters and send robot data
Robot input
Receive parameters and sensor data
Sensor in
Receive commands, parameters and robot data.
Sensor out
Return parameter and sensor data.

xx0800000128

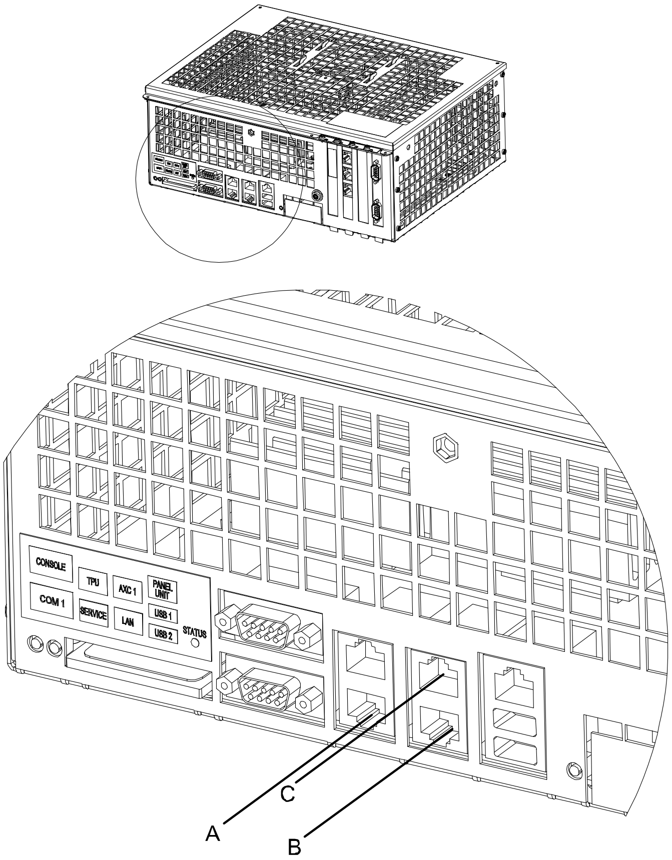
2 Installation

2.1. Connecting the communication cable

Overview

This section describes were to connect the communication cable on the controller. For further instructions, see the corresponding product manual for your robot system.

Location



xx0800000129

A	Service connection
B	Lan connection
C	Axc 1 connection (if unused)

	Action	Note
1.	Use one of these tree connections A, B, or C.	C. connection can only be used if it is free.

2.2. Prerequisites

Overview

This section describes the prerequisites for using *Robot Reference Interface*.

UDP/IP or TCP IP

Robot Reference Interface supports the communication over the standard IP protocols UDP or TCP.

UDP (HP-UDP)

In addition also a high-priority UDP (HP-UDP) implementation is supported, which bypasses the standard IP stack of the controller and directly accesses the IP packages. This ensures that the incoming and outgoing UDP messages are processed with a very short delay on the controller side, but does not imply real-time communication as such.

Recommendations

The delay in the overall communication mostly depends on the topology of the employed network. In a switched network the transmission will be delayed due to buffering of the messages in the switches. In a parallel network collisions with multiple communication partners will lead to messages being resent.

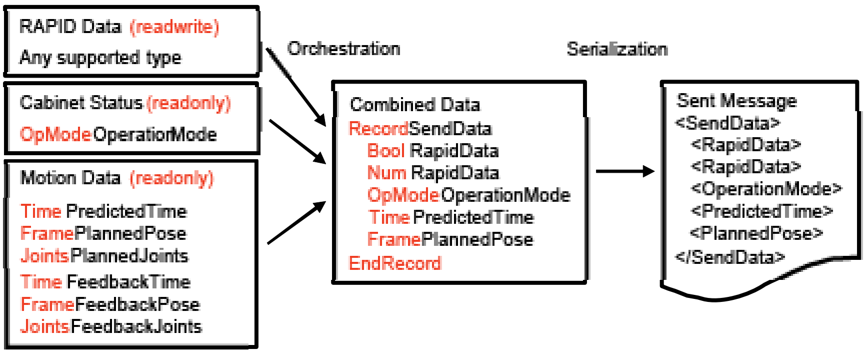
Therefore we recommended using a dedicated Ethernet link between the external system and the robot controller to provide the required performance for real-time applications. *Robot Reference Interface* can be used to communicate with any processor-based devices, that support IP via Ethernet and can serialize data into XML format.

2.3. Data orchestration

Overview

The outgoing message can be combined from any data from the RAPID level and internal data from the cabinet and motion topic. The orchestration of the data is defined in the device configuration by setting the Link attribute of internally linked data to *Intern*.

Illustration



xx0800000178

Data from the Controller topic

Name	Type	Description	Comment
OperationMode	OpMode	Operation mode of the robot.	The mapping of the members for the OpMode type can be defined in the configuration file.

Data from the Motion topic

Name	Type	Description	Comment
FeedbackTime	Time	Time stamp for the robot position from drive feedback.	There is a delay of approximately 8ms.
FeedbackPose	Frame	Robot TCP calculated from drive feedback.	Current tool and workobject are used for calculation.
FeedbackJoints	Joints	Robot joint values gathered from drive feedback.	
PredictedTime	Time	Timestamp for planned robot TCP position and joint values.	Prediction time from approximately 24ms to 60ms depending on robot type.
PlannedPose	Frame	Planned robot TCP.	Current tool and workobject are used for calculation.
PlannedJoints	Joints	Planned robot joint values.	

2.4. Supported data types

Overview

This section contains a short description of the *Robot Reference Interface* supported data types, for more detailed information about the supported data types see *References on page 5*.

Data types

Robot Reference Interface supports the following simple data types:

Data type	Description	RAPID type mapping
`bool`	Boolean value.	`bool`
`real`	Single precision, floating point value.	`num`
`time`	Time in seconds expressed as floating point value.	`num`
`string`	String with max length of 80 characters.	`string`
`frame`	Cartesian position and orientation in Euler Angles (Roll-Pitch-Jaw).	`pose`
`joints`	Robot joint values.	`robjoint`

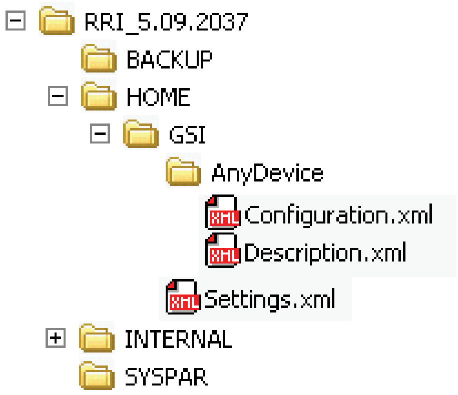
In addition, user-defined records can also be transferred from the external system to the robot controller, which are composed from the supported simple data types. User defined record types must be specified in the configuration file of the external device. See *Device configuration on page 20* for a description on how to create user-defined record types.

3 Configuration

3.1. Interface configuration

Configuration files

The configuration and settings files for the interface must be located in the folder HOME/GSI. This ensures that the configuration files are included in system backups.



```
RRI_5.09.2037
    BACKUP
    HOME
        GSI
            AnyDevice
                Configuration.xml
                Description.xml
            Settings.xml
    INTERNAL
    SYSPAR
```

xx0800000177

Related information

For more detailed information of the *Settings.xml* file see *Interface settings on page 16*.

For more detailed information of the *Description.xml* file see *Device description on page 17*.

For more detailed information of the *Configuration.xml* file see *Device configuration on page 20*.

3.2. Interface settings

Overview

This section describes the use of the xml file *Settings.xml*.

Settings.xml

The settings file Settings.xml contains the general settings for the GSI interface. It is located in the folder HOME/GSI. For the option *Robot Reference Interface* this file refers to a list of all communication clients for external systems installed in the controller. The Settings.xml file can be defined according to the XML schema Settings.xsd.

Example

For each communication client installed on the controller, the file Settings.xml must contain a Client entry in the Clients section. The Convention attribute identifies the protocol convention used by the client, for the *Robot Reference Interface* option only CDP is supported. The Name attribute identifies the name of the client and also specifies the folder with the device related configuration files.

```
<?xml version="1.0" encoding="UTF-8"?>
<Settings>
  <Clients>
     <Client Convention="CDP" Name="MySensor" />
  </Clients>
</Settings>
```

CDP stands for *cyclic data protocol* and is the internal name of the protocol, on which *Robot Reference Interface* messages are transferred.

An internal client node of the interface module will be created, which is able to connect to the external system *MySensor* that runs a data server application and can communicate via *Robot Reference Interface* with the robot.

For each sensor system, a subdirectory named with the sensor system identifier, for example *MySensor*, contains further settings.

3.3. Device description

Overview

This section describes the use of the xml file *Description.xml*.

Description.xml

The device description file Description.xml is located in the corresponding subdirectory of the device. It specifies the general device parameters, network connection and CDP specific communication settings for an installed device. A device description can be defined according to the XML schema Description.xsd.

Example

This is an example of a device description:

```
<?xml version="1.0" encoding="utf-8"?>
<Description>
  <Name>AnyDevice</Name>
  <Convention>CDP</Convention>
  <Type>IntelligentCamera</Type>
  <Class>MachineVision</Class>
  <Network  Address="10.49.65.74" Port="Service">
     <Channel Type="Cyclic" Protocol="Udp" Port="3002" />
  </Network>
  <Settings>
     <TimeOut>2000</TimeOut>
     <MaxLost>30</MaxLost>
     <DryRun>false</DryRun>
  </Settings>
</Description>
```

Name

The first section defines the general device parameters. The Name element identifies the name of the device and should correspond to the device name specified in the settings file. It must correspond to the identifier specified for the device descriptor on the RAPID level, because the descriptor name will be used initially to refer to the device in the RAPID instructions.

Element	Attribute	Description	Value	Comment
Name		Device identifier	Any string	Maximum 16 characters

Convention

The Convention element identifies the protocol that should be used by the device, for the *Robot Reference Interface* option only the Cyclic Data Protocol (CDP) is supported.

Element	Attribute	Description	Value	Comment
Convention		Protocol type	CDP	

Continues on next page

Continued

Type and Class

The Type and Class elements identifies the device type and class and are currently not validated, therefore they can also contain undefined device types or classes.

Element	Attribute	Description	Value	Comment
Type		Sensor type	Any string	Not validated
Class		Sensor class	Any string	Not validated

Network

The Network section defines the network connection settings for the device. The Address attribute specifies the IP address or host name of the device on the network. The optional Port attribute is used to specify the physical Ethernet port on the controller side that the cable is plugged into. Valid values are *Lan*, *Service*, and *Axc1* to *Axc4* if an additional robot communication card is installed. The attribute can be omitted if the Lan port is used for communication.

Element	Attribute	Description	Value	Comment
Network		Network settings	Any valid IP address or host name	
	Address	IP address or host name of the device	Lan Service Axc1 ... Axc4	10.49.65.249 DE-L-0328122
	Port	Physical Ethernet port on the controller		Optional. Can be omitted if Lan port used.

Channel

The Channel element defines the settings for the communication channel between the robot controller and the external device. The Type attribute identifies the channel type, only *Cyclic* is supported by *Robot Reference Interface*.

The Protocol attribute identifies the IP protocol used on the channel, for *Robot Reference Interface* you can specify to use *Tcp*, *Udp*, or the high-priority UDP implementation *HpUdp*. The Port attribute specifies the logical port number for the channel on the device side.

Element	Attribute	Description	Value	Comment
Channel		Channel settings		
	Type	Channel type	Cyclic	
	Protocol	The IP protocol type	Tcp Udp HpUdp	*Tcp* and *Udp* are handled by the standard IP stack. *HpUdp* uses high-priority UDP implementation.
	Port	The logical port number of the channel	uShort	Any available port number on the device, maximum 65535.

Continues on next page

Settings

The Settings section contains communication parameters specific to the CDP protocol. The TimeOut element defines the timeout for not received messages. This element identifies the time until the connection is considered broken and is only needed for bidirectional communication. The MaxLost attribute defines the maximum number of not acknowledged or lost messages allowed. The DryRun element identifies, if the acknowledgement of messages is supervised and can be used to setup an unidirectional communication.

Element	Description	Value	Comment
TimeOut	Time out for communication		Time in milliseconds, a multiple of 4 ms.
MaxLost	Maximum loss of packages allowed	Integer	
DryRun	Interface run mode	Bool	If TRUE, TimeOut and MaxLost will not be checked.

If the element DryRun in the Description.xml is set to FALSE, communication supervision is established on the protocol level of the *Robot Reference Interface*, using the settings for *TimeOut* and *MaxLost*. This supervision requires that each message that is sent out from the robot controller is answered by the connected device. The supervision generates a communication error, if the maximum response time or the maximum number of lost packages is exceeded. Each sent out message has an ID, which needs to be used for the ID in the reply too, to identify the reply message and to detect which packages have been lost. See also the example in section *Transmitted XML messages on page 27*.

3.4. Device configuration

Overview

The device configuration file *Configuration.xml* is located in the corresponding subdirectory of the device. It defines the enumerated and complex types used by the device and identifies the available parameters, which can be subscribed for cyclic transmission. The configuration file can be defined according to the XML schema Configuration.xsd. The following document shows a simplified device configuration.

Example

```
<?xml version="1.0" encoding="utf-8"?>
<Configuration>
  <Enums>
     <Enum Name="opmode" Link="Intern">
        <Member Name="ReducedSpeed" Alias="Alias"/>
     </Enum>
  </Enums>
  <Records>
     <Record Name="senddata">
        <Field Name="PlannedPose" Type="Frame" Link="Intern" />
     </Record>
  </Records>
  <Properties>
     <Property Name="DataToSend" Type="senddata" Flag="WriteOnly"
           />
  </Properties>
</Configuration>
```

Enums

In the Enums section each Enum element defines an enumerated type. The Name attribute of the Enum element specifies the name of the enumerated type, the optional Link attribute identifies if the members of the enumerated type have internal linkage.

Element	Attribute	Descriptions	Value	Comment
Enum				
	Name	Name of enumerated type	A valid RAPID symbol name	Maximum 16 characters.
	Link	Linkage of members of enumerated type	*Intern*	Optional. Can be omitted if members only have RAPID linkage.

Continues on next page

Member

Each Member element defines a member element of the enumerated type. The Name attribute specifies the name of the member on the controller side (on RAPID level). The Alias attribute identifies the name of the member on the device side (and in the transmitted message).

Element	Attribute	Descriptions	Value	Comment
Member				
	Name	Name of enumerated type member	A valid RAPID symbol name	Maximum 16 characters.Valid internal RAPID symbol names. See *Data orchestration on page 13*.
	Alias	Alias name of enumerated type member	String	Optional. The alias name is used on the device side and in message

Record

In the Records section each Record element defines a declaration of a complex type. In RAPID this complex type will be represented as a RECORD declaration. The Name attribute identifies the name of the complex type on the controller side. The Alias attribute defines the alias name of the type on the device side and in the message.

Element	Attribute	Descriptions	Value	Comment
Record				
	Name	Name of the complex type.	A valid RAPID symbol name	Maximum 16 characters.
	Alias	Alias name of complex type.	String	Optional. The alias name is used on the device side and in message.

Continues on next page

Field

Each Field element defines a field element of a complex type. The Name attribute identifies the name of the field. The Type attribute identifies the enumerated, complex or simple type associated with the field. The Size attribute defines the size of a multi-dimensional field. The Link attribute identifies if the field has internal linkage.

Element	Attribute	Descriptions	Value	Comment
Field				
	Name	Name of the complex type field	A valid RAPID symbol name	Maximum 16 characters.Valid internal RAPID symbol names. See *Data orchestration on page 13*.
	Type	Data type of the field	All supported data types	Described in section *Supported data types on page 14*.
	Size	Dimensions of the field (size of array)	Integer	Optional. Only basic types can be defined as array.
	Link	Linkage of complex type field	*Intern*	Optional. Can be omitted if field has RAPID linkage.
	Alias	Alias name of complex type field	String	Optional. The alias name is used on device side and in message.

Properties

In the Properties section each Property element defines a RAPID variable that can be used in the `SiGetCyclic` and `SiSetCyclic` instructions.

Element	Attribute	Descriptions	Value	Comment
Property				
	Name	Name of the property	An valid RAPID symbol name	Maximum 16 characters.
	Type	Data type of the property	All supported data types	Described in section *Supported data types on page 14*.
	Size	Dimension (Size of array)	Integer	Optional. Only basic types can be defined as array.
	Flag	Access Flag	None *ReadOnly* *WriteOnly* *ReadWrite*	Optional. Can be omitted if property is read and write enabled.
	Link	Linkage of property	*Intern*	Optional. Can be omitted if field has RAPID linkage.
	Alias	Alias name of the property	String	Optional. The alias name is used on device side and in message.

4 Configuration examples

4.1. RAPID programming

RAPID module

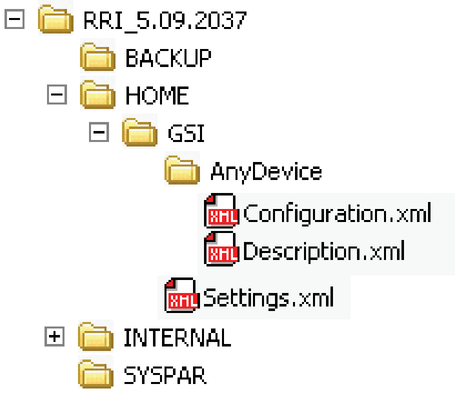
A RAPID module containing the corresponding RAPID record declarations and variable declarations must be created and loaded.

The FlexPendant user interface is not included in RobotWare.

4.2. Example configuration

Overview

The files Settings.xml, Description.xml, and Configuration.xml are located in the folder HOME\GSI\



```
⊟ RRI_5.09.2037
    BACKUP
  ⊟ HOME
    ⊟ GSI
        AnyDevice
          Configuration.xml
          Description.xml
        Settings.xml
  ⊞ INTERNAL
    SYSPAR
```

xx0800000177

Settings.xml

```
<?xml version="1.0" encoding="utf-8"?>
<Settings>
  <Servers/>
  <Clients>
     <Client Convention="CDP" Name="AnyDevice" />
  </Clients>
</Settings
```

Description.xml

```
<?xml version="1.0" encoding="utf-8"?>
<Description>
  <Name>AnyDevice</Name>
  <Convention>CDP</Convention>
  <Type>IntelligentCamera</Type>
  <Class>MachineVision</Class>
  <Network  Address="10.49.65.74" Port="Service">
     <Channel Type="Cyclic" Protocol="Udp" Port="3002" />
  </Network>
  <Settings>
     <TimeOut>2000</TimeOut>
     <MaxLost>30</MaxLost>
     <DryRun>false</DryRun>
  </Settings>
</Description>
```

Continues on next page

Configuration.xml

```
<?xml version="1.0" encoding="utf-8" ?>
<Configuration>
  <Enums>
    <Enum Name="OperationMode" Link="Intern">
      <Member Name="Automatic" Alias="Auto" />
      <Member Name="ReducedSpeed" Alias="ManRS" />
      <Member Name="FullSpeed" Alias="ManFS" />
    </Enum>
  </Enums>
  <Records>
    <Record Name="RobotData">
      <Field Name="OperationMode" Type="OperationMode"
          Link="Intern" Alias="RobMode" />
      Alias="RobMode" />
    <Field Name="FeedbackTime" Type="Time" Link="Intern"
        Alias="Ts_act" />
    <Field Name="FeedbackPose" Type="Pose" Link="Intern"
        Alias="P_act" />
    <Field Name="FeedbackJoints" Type="Joint" Link="Intern"
        Alias="J_act" />
    <Field Name="PredictedTime" Type="Time" Link="Intern"
        Alias="Ts_des" />
    <Field Name="PlannedPose" Type="Pose" Link="Intern"
        Alias="P_des" />
    <Field Name="PlannedJoints" Type="Joint" Link="Intern"
        Alias="J_des" />
    <Field Name="ApplicationData" Type="Num" Size="18"
        Alias="AppData" />
    </Record>
    <Record Name="SensorData">
      <Field Name="ErrorString" Type="String" Alias="EStr" />
      <Field Name="ApplicationData" Type="Num" Size="18"
          Alias="AppData" />
    </Record>
  </Records>
  <Properties>
    <Property Name="RobData" Type="RobotData" Flag="WriteOnly"/>
    <Property Name="SensData" Type="SensorData" Flag="ReadOnly"/>
  </Properties>
</Configuration>
```

Continues on next page

Continued

RAPID configuration

```
! example for a RRI implementation
! out data using an array of 18 num (robdata)
! in data receiving a string and an array of 18 num (sensdata)
! This needs to defined according the configuration.xml file
!
RECORD applicationdata
  num Item1;
  num Item2;
  num Item3;
  num Item4;
  num Item5;
  num Item6;
  num Item7;
  num Item8;
  num Item9;
  num Item10;
  num Item11;
  num Item12;
  num Item13;
  num Item14;
  num Item15;
  num Item16;
  num Item17;
  num Item18;
ENDRECORD
RECORD robdata
  applicationdata AppData;
ENDRECORD
RECORD sensdata
  string ErrString;applicationdata AppData;
ENDRECORD
! Sensor Declarations
PERS sensor AnyDevice := [1,4,0];
PERS robdata DataOut := [[0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0]];
PERS sensdata DataIn :=
     ["No",[0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0]];
! Setup Interface Procedure
PROC RRI_Open()
  SiConnect AnyDevice;
  ! Send and receive data cyclic with 64 ms rate
  SiGetCyclic AnyDevice, DataIn, 64;
  SiSetCyclic AnyDevice, DataOut, 64;
ENDPROC
```

Continues on next page

```
  ! Close Interface Procedure
  PROC RRI_Close()
    ! Close the connection
    SiClose RsMaster;
  ENDPROC
ENDMODULE
```

Transmitted XML messages

Each XML message has the data variable name as root element with the attributes `Id` (the message ID) and `Ts` (the timestamp of the message). The subelements are then the record fields. The values of a multiple value field (array or record) are expressed as attributes.

Message sent out from robot controller

The time unit is second (float) with a resolution of 1 ms, and the position (length) unit is millimeter (float), position (angle) unit is degrees.

Name	Data type	Description
`Id`	Integer	Last received robot data message ID
`Ts`	Float	Time stamp (message)
`RobMode`	Operationmode	Operation mode
`TS_act`	Float	Time stamp (actual position)
`P_act`	Pose	Actual cartesian position
`J_act`	Joint	Actual joint position
`TS_des`	Float	Time stamp (desired position)
`P_des`	Pose	Desired cartesian position
`J_des`	Joint	Desired joint position
`AppData`	Array of 18 Floats	Free defined application data

```
<RobData Id="111" Ts="1.202" >
  <RobMode>Auto</RobMode>
  <Ts_act>1.200</Ts_act>
  <P_act X="1620.0" Y="1620.0" Z="1620.0" Rx="100.0" Ry="100.0"
        Rz="100.0" />
  <J_act J1="1.0" J2="1.0" J3="1.0" J4="1.0" J5="1.0" J6="1.0" />
  <Ts_des>1.200</Ts_des>
  <P_des X="1620.0" Y="1620.0" Z="1620.0" Rx="100.0" Ry="100.0"
        Rz="100.0" />
  <J_des J1="1.0" J2="1.0" J3="1.0" J4="1.0" J5="1.0" J6="1.0" />
  <AppData X1="1" X2="1620.000" X3="1620.000" X4="1620.000"
        X5="1620.000" X6="1620.000" X7="1620.000" X8="1620.000"
        X9="1620.000" X10="1620.000" X11="1620.000" X12="1620.000"
        X13="1620.000" X14="1620.000" X15="1620.000"
        X16="1620.000" X17="1620.000" X18="1620.000" />
</RobData>
```

Continues on next page

Continued

Message received from robot controller

The time unit is seconds (float).

Name	Data type	Description
`Id`	Integer	Last received data message ID. This ID must correspond to the ID sent from the robot controller.
`Ts`	Float	Time stamp
`EStr`	String	Error message
`AppData`	Array of 18 floats	Free defined application data

The corresponding XML message on the network would look like this:

```
<SensData Id="111" Ts="1.234">
  <EStr>xxxx</Estr>
     <AppData X1="232.661" X2="1620.293" X3="463.932"
      X4="1231.053" X5="735.874" X6="948.263" X7="2103.584"
      X8="574.228" X9="65.406" X10="2372.633" X11="20.475"
      X12="96.729" X13="884.382" X14="927.954" X15="748.294"
      X16="3285.574" X17="583.293" X18="684.338" />
</SensData>
```

5 RAPID reference information

5.1 RAPID instructions

5.1.1. SiConnect - Sensor Interface Connect

Usage

`SiConnect` is used to establish a connection to an external device.

Basic examples

A basic example of the instruction `SiConnect` is illustrated below.

See also *More examples on page 30*.

Example 1

```
PERS sensor AnyDevice;
...
  SiConnect AnyDevice;
```

Establish a connection to the device called `AnyDevice`.

Arguments

```
SiConnect Sensor [\NoStop]
```

Sensor

Data type: `sensor`

The descriptor for the external device to connect to. The argument is a persistent variable and its name must be the same as the name specified as the client in setup file `Settings.xml`.

[\NoStop]

Data type: `switch`

`\NoStop` will prevent system stop when a communication error with the sensor is detected. It can be useful if no robot movements are depending on the sensor. When `\Nostop` is used, movements in the system will continue even if the communication with the sensor is lost.

If using `\NoStop` it is possible to do error handling in a `TRAP` routine, with the use of `IError` or `IPers`.

Program execution

Loads the current sensor configuration and establishes the connection to the external device.

The sensor stays connected, even if the program pointer is set to main.

Continues on next page

More examples

More examples of how to use the instruction `SiConnect` are illustrated below.

Example 1

```
PERS sensor AnyDevice;
PERS robdata DataOut := [[0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0]];
PERS sensdata DataIn :=
     ["No",[0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0]];
VAR num SampleRate:=64;
...
! Setup Interface Procedure
PROC RRI_Open()
  SiConnect AnyDevice;
  ! Send and receive data cyclic with 64 ms rate
  SiGetCyclic AnyDevice, DataIn, SampleRate;
  SiSetCyclic AnyDevice, DataOut, SampleRate;
ENDPROC
```

When calling routine `RRI_Open`, first a connection to the device with name `AnyDevice` is opened. Then, cyclic transmission is started at rate `SampleRate`.

Example 2

```
PERS sensor AnyDevice;
...
  SiConnect AnyDevice \NoStop;
  ! Send and receive data cyclic with 64 ms rate
  SiGetCyclic AnyDevice, DataIn, SampleRate;
  SiSetCyclic AnyDevice, DataOut, SampleRate;
...
TRAP sensorChange
  IF AnyDevice.state = STATE_ERROR THEN
  ...
  ENDIF
ENDTRAP
```

Establish a connection to the device called `AnyDevice` with the optional argument \NoStop preventing the system to stop if the connection to `AnyDevice` is broken. Handle error states in the `TRAP` routine.

Continues on next page

Error handling

If UDP is used as communication protocol no guarantees are given regarding the success of the connect operation and therefore no error handling is possible at the connect moment.

If TCP is used as communication protocol the system variable `ERRNO` is set to `ERR_COMM_INIT` if the connect operation fails. This error can then be handled in an error handler.

The switch `\NoStop` makes it possible to handle communication errors detected after a successful connect. `\NoStop` means that movements and execution of RAPID continues and that a `TRAP` routine can be used to handle specific errors using `IError` or specific state changes using `IPers`.

NOTE: `IPers` and `IError` are not safe interrupts, so if an error is detected after a stop, no `TRAP` will be executed. A way to handle this problem is to have a `SiConnect \NoStop` in the restart shelf, to be sure that the application tries to reestablish the connection to the client.

Syntax

```
SiConnect
  [ Sensor ':=' ] < persistent (PERS) of sensor >
  [ '\' NoStop ] ';'
```

Related information

For information about	See
Close connection to an external system.	*SiClose - Sensor Interface Close on page 32*.
Register data for cyclic transmission.	*SiSetCyclic - Sensor Interface Set Cyclic on page 35*.
Subscribe on cyclic data transmission.	*SiGetCyclic - Sensor Interface Get Cyclic on page 33*
Descriptor to the external device.	*sensor - External device descriptor on page 37*.
Communication state of a device.	*sensorstate - Communication state of the device on page 39*.

5.1.2. SiClose - Sensor Interface Close

Usage

`SiClose` closes an existing connection to an external device.

Basic examples

Basic example of the instruction `SiClose` is illustrated below.

Example 1

```
PERS sensor AnyDevice;
...
  SiClose AnyDevice;
```

Close the connection to the device called `AnyDevice`.

Arguments

```
SiClose Sensor
```

Sensor

Data type: `sensor`

The descriptor for the external device that should be closed. The argument is a persistent variable, and its name must be the same as the name specified as the client in the setup file `Settings.xml`.

Program execution

Closes an existing connection to the external device.

Error handling

If UDP is used as communication protocol then there is no guarantee regarding the success of the close operation and therefore no error handling is possible.

If TCP is used as communication protocol the system variable `ERRNO` is set to `ERR_COMM_INIT` if the close operation fails. This error can then be handled in an error handler.

Syntax

```
SiClose
  [ Sensor ':=' ] < persistent (PERS) of sensor > ';'
```

Related information

For information about	See
Establish a connection to an external system.	*SiConnect - Sensor Interface Connect on page 29*.
Register data for cyclic transmission.	*SiSetCyclic - Sensor Interface Set Cyclic on page 35*.
Subscribe on cyclic data transmission.	*SiGetCyclic - Sensor Interface Get Cyclic on page 33*.
Descriptor to the external device.	*sensor - External device descriptor on page 37*.
Communication state of a device.	*sensorstate - Communication state of the device on page 39*.

5.1.3. SiGetCyclic - Sensor Interface Get Cyclic

Usage

`SiGetCyclic` subscribes data for cyclic transmission from an external device.

Basic examples

A basic example of the instruction `SiGetCyclic` is illustrated below.

See also *More examples on page 34*.

Example 1

```
SiConnect AnyDevice;
! Receive data cyclic with 64 ms rate
SiGetCyclic AnyDevice, DataIn, 64;
```

The example shows how to establish connection to an external device and set up a cyclic transmission from the device `AnyDevice`.

Arguments

```
SiGetCyclic Sensor Data Rate
```

Sensor

Data type: `sensor`

A descriptor for the external device to receive cyclic data from. The argument is a persistent variable, and its name must be the same as the name specified as the client in setup file `Settings.xml`.

Data

Data type: `anytype`

Reference to a persistent containing the data to receive from the client specified in argument `Sensor`. The variable must be defined as *Readable* in the file Configuration.xml.

Rate

Data type: `num`

Transfer rate in milliseconds (only multiples of 4ms are supported).

Program execution

Instruction `SiGetCyclic` subscribes data for cyclic transmission from an external device.

For `SiGetCyclic` and `SiSetCyclic` instructions, a transfer rate of 0 stops (unregisters / unsubscribes) the cyclic transmission of the given data or data set.

Continues on next page

More examples

More examples of how to use the instruction `SiGetCyclic` are illustrated below.

Example 1

```
PERS sensor AnyDevice;
PERS robdata DataOut := [[0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0]];
PERS sensdata DataIn :=
     ["No",[0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0]];
VAR num SampleRate:=64;
...
! Setup Interface Procedure
PROC RRI_Open()
  SiConnect AnyDevice;
  ! Send and receive data cyclic with 64 ms rate
  SiGetCyclic AnyDevice, DataIn, SampleRate;
  SiSetCyclic AnyDevice, DataOut, SampleRate;
ENDPROC
```

When calling routine `RRI_Open`, first a connection to the device with name `AnyDevice` is opened. Then, cyclic transmission is started at rate `SampleRate`.

Syntax

```
SiGetCyclic
  [ Sensor ':=' ] < persistent (PERS) of sensor > ','
  [ Data ':=' ] < persistent (PERS) of anytype > ','
  [ Rate ':=' ] < expression (IN) of num > ] ';'
```

Related information

For information about	See
Establish a connection to an external system.	*SiConnect - Sensor Interface Connect on page 29*.
Close connection to an external system.	*SiClose - Sensor Interface Close on page 32*.
Register data for cyclic transmission.	*SiSetCyclic - Sensor Interface Set Cyclic on page 35*.
Descriptor to the external device.	*sensor - External device descriptor on page 37*.
Communication state of a device.	*sensorstate - Communication state of the device on page 39*.

5.1.4. SiSetCyclic - Sensor Interface Set Cyclic

Usage

`SiSetCyclic` registers data for cyclic transmission to an external device.

Basic examples

A basic example of the instruction `SiSetCyclic` is illustrated below.

See also *More examples on page 36*.

Example 1

```
PERS sensor AnyDevice;
PERS robdata DataOut := [[0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0]];
...
  SiConnect AnyDevice;
  SiSetCyclic AnyDevice, DataOut, 40;
```

Establish a connection to the device called `AnyDevice`. Then register data for cyclic transmission to the external device `AnyDevice` every 40 ms.

Arguments

```
SiSetCyclic Sensor Data Rate
```

Sensor

Data type: `sensor`

A descriptor for the external device to send data to.

Data

Data type: `anytype`

Reference to a persistent of any complex or supported simple type containing the data to be sent to the client specified in argument `Sensor`. The variable must be defined as *Writable* in the `Configuration.xml` file.

Rate

Data type: `num`

Transfer rate in milliseconds (only multiples of 4 ms are supported).

Program execution

Instruction `SiSetCyclic` registers data for cyclic transmission to an external device.

For `SiGetCyclic` and `SiSetCyclic` instructions, a transfer rate of 0 stops (unregisters / unsubscribes) the cyclic transmission of the given data or data set.

Continues on next page

More examples

More examples of how to use the instruction `SiSetCyclic` are illustrated below.

Example 1

```
PERS sensor AnyDevice;
PERS robdata DataOut := [[0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0]];
PERS sensdata DataIn :=
     ["No",[0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0]];
VAR num SampleRate:=64;
...
! Setup Interface Procedure
PROC RRI_Open()
  SiConnect AnyDevice;
  ! Send and receive data cyclic with 64 ms rate
  SiGetCyclic AnyDevice, DataIn, SampleRate;
  SiSetCyclic AnyDevice, DataOut, SampleRate;
ENDPROC
```

When calling routine `RRI_Open`, first a connection to the device with name `AnyDevice` is opened. Then, cyclic transmission is started at rate `SampleRate`.

Syntax

```
SiSetCyclic
  [ Sensor ':=' ] < persistent (PERS) of sensor > ','
  [ Data ':=' ] < persistent (PERS) of anytype >
  [ Rate ':=' ] < expression (IN) of num > ] ';'
```

Related information

For information about	See
Establish a connection to an external system.	*SiConnect - Sensor Interface Connect on page 29*.
Close connection to an external system.	*SiClose - Sensor Interface Close on page 32*.
Subscribe on cyclic data transmission.	*SiGetCyclic - Sensor Interface Get Cyclic on page 33*.
Descriptor to the external device.	*sensor - External device descriptor on page 37*.
Communication state of a device.	*sensorstate - Communication state of the device on page 39*.

5.2 RAPID data types

5.2.1. sensor - External device descriptor

Usage

`sensor` is a descriptor to the external device to connect to.

Description

The descriptor for a device on the RAPID level is encapsulated in the record data type `sensor`. It holds information about the sensor device such as id, error code and sensor communication state.

Components

id

Data type: `num`

The internal identifier of the device, which will be set on the first operation with the device from RAPID level. (Not implemented yet).

error

Data type: `num`

The `error` parameter is set when parameter `state` is set to `STATE_ERROR`. When `state` goes from `STATE_ERROR` to `STATE_CONNECTED` parameter `error` is set to 0.

Error number	Error
0	No error.
112600	Communication interface initialization failed.
112602	Communication interface error.

state

Data type: `sensorstate`

Reflects the actual communication state of the device.

Continues on next page

Examples

Example of the data type `sensor` is shown below.

Example 1

```
PERS sensor AnyDevice;
PERS robdata DataOut := [[0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0]];
PERS sensdata DataIn :=
     ["No",[0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0]];
VAR num SampleRate:=64;
...
! Setup Interface Procedure
PROC RRI_Open()
  SiConnect AnyDevice;
  ! Send and receive data cyclic with 64 ms rate
  SiGetCyclic AnyDevice, DataIn, SampleRate;
  SiSetCyclic AnyDevice, DataOut, SampleRate;
ENDPROC
```

When calling routine `RRI_Open`, first a connection to the device `AnyDevice` is opened. Then, cyclic transmission is started at rate `SampleRate`.

Structure

```
<dataobject of sensor>
  <id of num>
  <error of num>
  <state of sensorstate>
```

Related information

For information about	See
Establish a connection to an external system.	*SiConnect - Sensor Interface Connect on page 29*.
Close connection to an external system.	*SiClose - Sensor Interface Close on page 32*.
Register data for cyclic transmission.	*SiSetCyclic - Sensor Interface Set Cyclic on page 35*.
Subscribe on cyclic data transmission.	*SiGetCyclic - Sensor Interface Get Cyclic on page 33*
Communication state of a device.	*sensorstate - Communication state of the device on page 39*.

5.2.2. sensorstate - Communication state of the device

Usage

`sensorstate` is used to represent an actual communication state of a device.

Description

A `sensorstate` constant is used to reflect the actual communication state of a device. It can be used from RAPID to evaluate the state of the connection with the sensor.

Predefined data

The following symbolic constants of the data type `sensorstate` are predefined and can be used to evaluate what communication state the device is in.

Constant	Value
STATE_ERROR	-1
STATE_UNDEFINED	0
STATE_CONNECTED	1
STATE_OPERATING	2
STATE_CLOSED	3

Characteristics

`sensorstate` is an alias data type for `num` and consequently inherits its characteristics.

Related information

For information about	See
Establish a connection to an external system.	*SiConnect - Sensor Interface Connect on page 29*.
Close connection to an external system.	*SiClose - Sensor Interface Close on page 32*.
Register data for cyclic transmission.	*SiSetCyclic - Sensor Interface Set Cyclic on page 35*.
Subscribe on cyclic data transmission.	*SiGetCyclic - Sensor Interface Get Cyclic on page 33*
Descriptor to the external device.	*sensor - External device descriptor on page 37*.

Continues on next page

ABB AB
Robotics Products
S-721 68 VÄSTERÅS
SWEDEN
Telephone: +46 (0) 21 344000
Telefax: +46 (0) 21 132592