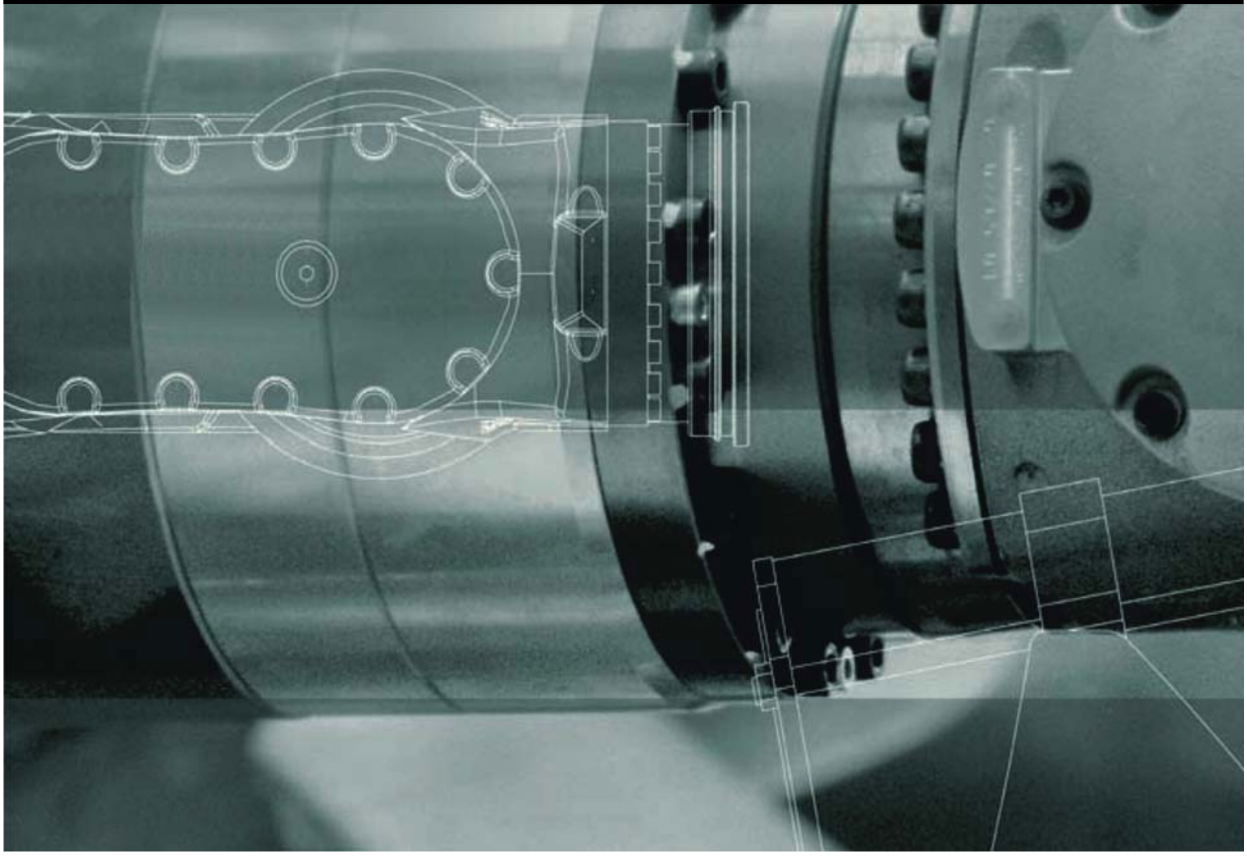
KUKA System Technology

KUKA Roboter GmbH

KUKA.PLC Multiprog 5-35 4.1

For KUKA System Software 8.2 and 8.3



Issued: 08.11.2012

Version: KST PLC Multiprog 4.1 V2 en (PDF)

KUKA Roboter GmbH
Zugspitzstraße 140
D-86165 Augsburg
Germany

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

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

Subject to technical alterations without an effect on the function.

Translation of the original documentation

KIM-PS5-DOC

Publication: Pub KST PLC Multiprog 4.1 (PDF) en
Bookstructure: KST PLC Multiprog 4.1 V2.1
Version: KST PLC Multiprog 4.1 V2 en (PDF)

Contents

1 Introduction

1.1 Target group

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

- Advanced KRL programming skills
- Advanced PLC programming skills
- Advanced knowledge of the robot controller system

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

1.2 Industrial robot documentation

The industrial robot documentation consists of the following parts:

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

Each of these sets of instructions is a separate document.

1.3 Representation of warnings and notes

Safety

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

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

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

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

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

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

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

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

Notes

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

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

1.4 Trademarks

ProConOS and **MULTIPROG** are trademarks of KW-Software GmbH.

VxWorks is a trademark of Wind River Systems Inc.

Windows and **Windows XP** are trademarks of Microsoft Corporation.

1.5 Terms used

Term	Description
Exception	Exceptional treatment for a specific event.
KLI	**K**UKA **L**ine **I**nterface: Connection to Ethernet network
KUKA.PLC Multiprog 5-35	Soft PLC for use in the robot controller.
ProConOS	Software that serves as a runtime system to execute PLC applications.
WorkVisual	Software that serves as an offline engineering system for the software of KR C4-controlled robot systems.
SPOC	Single Point of Control
Retentive data	Data that are retained even when the PLC is reset.
Time slice	Allocated time slot in which an application takes over the CPU processing time.
Watchdog	Function that monitors the max. allocated processing/response time.
Notification message	A message that is for information purposes only and does not interrupt program execution. It does not require acknowledgement. A notification message may contain general information, for example, or confirm an operator action. A notification message can only be deleted using the buttons **OK** and **Confirm all**.

Term	Description
Status message	A message that signals a status. It is generally for information purposes only, but may also interrupt program execution. The message is automatically deleted when the status that triggered it is no longer applicable. A status message cannot be deleted by the user via the smartHMI.
PDD	Process Data Directory Mechanism for exchanging process data

2 Product description

2.1 Overview of KUKA.PLC Multiprog 5-35

Description

KUKA.PLC Multiprog 5-35 is a technology package with the following functions:

- It extends the range of possible solutions for automation tasks already provided by KRL (KUKA Robot Language). KUKA.PLC Multiprog 5-35 thus represents an expanded development environment for PLC applications.
- KUKA.PLC Multiprog 5-35 is installed on a standard laptop/PC. In this case, the laptop/PC can be used as a development environment for PLC applications, which can subsequently be executed on the robot controller.

Required components

- **ProConOS runtime system:** KUKA.PLC Multiprog 5-35 uses the ProConOS runtime system to execute PLC applications.

  ProConOS must be installed on the robot controller and requires the real-time operating system VxWorks. ProConOS is compatible with the IEC standard 61131-3, and thus employs standardized syntax and semantics. ProConOS can be configured by means of an initialization file, and can be adapted flexibly to the requirements of the robot system.

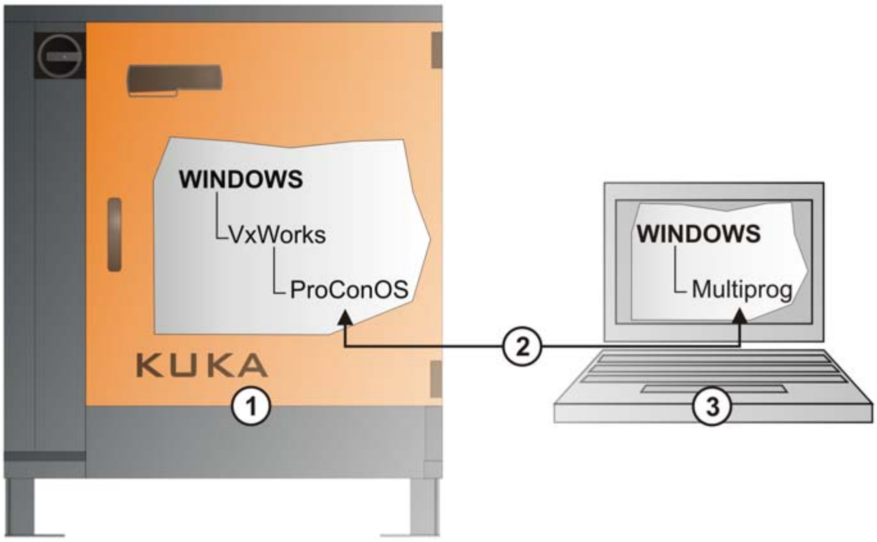


Fig. 2-1: Configuration overview

Item	Description
1	KR C4: ■ Windows operating system ■ VxWorks real-time operating system ■ ProConOS runtime system
2	Connection: KLI
3	External PC/laptop: ■ Windows operating system ■ MULTIPROG PLC development environment ■ WorkVisual offline engineering system

3 Safety

This documentation contains safety instructions which refer specifically to the software described here.

The fundamental safety information for the industrial robot can be found in the “Safety” chapter of the Operating and Programming Instructions for System Integrators or the Operating and Programming Instructions for End Users.

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

⚠ **WARNING** Signal states can be changed by downloading the Multiprog project, via the control dialog in Multiprog or by transferring them out of WorkVisual. It must be ensured that potentially hazardous signals (e.g. the opening/closing of a gun) can only be executed if AUT or AUT EXT mode is set and the safety gate is closed. For this, the signals must be mapped accordingly by means of the variable **bSPOC_UserSafetyActive**.

The variable **bSPOC_UserSafetyActive** corresponds to the negation of the variable **$USER_SAF**.

The following table describes which state the variable **bSPOC_UserSafetyActive** assumes with regard to operator safety, enabling switch and the set operating mode (AUT, EXT, T1, T2, KRF).

Operator safety	Enabling switch	AUT / EXT	T1 / KRF	T2
inactive	not pressed	TRUE	TRUE	TRUE
	pressed	TRUE	FALSE	FALSE
active	not pressed	FALSE	TRUE	TRUE
	pressed	FALSE	FALSE	FALSE

⚠ If the variable **bSPOC_UserSafetyActive** has the TRUE state, the safety measures for “single point of control” must be taken into account.

ℹ If operator safety is implemented in the form of a safety gate, then inactive operator safety means an open safety gate and active operator safety a closed safety gate.

3.1 Safety measures for “single point of control”

Overview

If certain components in the industrial robot are operated, safety measures must be taken to ensure complete implementation of the principle of “single point of control” (SPOC).

Components:

- Submit interpreter
- PLC
- OPC Server
- Remote control tools
- Tools for configuration of bus systems with online functionality
- KUKA.RobotSensorInterface
- External keyboard/mouse

⚠ The implementation of additional safety measures may be required. This must be clarified for each specific application; this is the responsibility of the system integrator, programmer or user of the system.

Since only the system integrator knows the safe states of actuators in the periphery of the robot controller, it is his task to set these actuators to a safe state, e.g. in the event of an EMERGENCY STOP.

T1, T2

In the test modes, the components referred to above (with the exception of the external keyboard/mouse) may only access the industrial robot if the following signal has the following state:

Signal	State required for SPOC
bSPOC_UserSafetyActive	FALSE

Submit interpreter, PLC

If motions, (e.g. drives or grippers) are controlled with the submit interpreter or the PLC via the I/O system, and if they are not safeguarded by other means, then this control will take effect even in T1 and T2 modes or while an EMERGENCY STOP is active.

If variables that affect the robot motion (e.g. override) are modified with the submit interpreter or the PLC, this takes effect even in T1 and T2 modes or while an EMERGENCY STOP is active.

Safety measures:

- In the test modes, the system variable $OV_PRO must not be written to by the submit interpreter or the PLC.
- Do not modify safety-relevant signals and variables (e.g. operating mode, EMERGENCY STOP, safety gate contact) via the submit interpreter or PLC.

  If modifications are nonetheless required, all safety-relevant signals and variables must be linked in such a way that they cannot be set to a dangerous state by the submit interpreter or PLC.

OPC server, remote control tools

These components can be used with write access to modify programs, outputs or other parameters of the robot controller, without this being noticed by any persons located inside the system.

Safety measures:

- KUKA stipulates that these components are to be used exclusively for diagnosis and visualization.

  Programs, outputs or other parameters of the robot controller must not be modified using these components.
- If these components are used, outputs that could cause a hazard must be determined in a risk assessment. These outputs must be designed in such a way that they cannot be set without being enabled. This can be done using an external enabling device, for example.

Tools for configuration of bus systems

If these components have an online functionality, they can be used with write access to modify programs, outputs or other parameters of the robot controller, without this being noticed by any persons located inside the system.

- WorkVisual from KUKA
- Tools from other manufacturers

Safety measures:

- In the test modes, programs, outputs or other parameters of the robot controller must not be modified using these components.

External keyboard/mouse

These components can be used to modify programs, outputs or other parameters of the robot controller, without this being noticed by any persons located inside the system.

Safety measures:

- Only use one operator console at each robot controller.
- If the KCP is being used for work inside the system, remove any keyboard and mouse from the robot controller beforehand.

4 Installation

4.1 System requirements

Overview

- Standard laptop/PC
  - WorkVisual 2.4 or higher

    The requirements for installation of WorkVisual are contained in the WorkVisual documentation.
- Network connections (network switch, network cable, 100 Mbit network card)

> Information about the supported Windows versions can be found in the Multiprog online help.

4.2 Installing KUKA.PLC Multiprog 5-35

Precondition

- Local administrator rights
- Software on CD/DVD or USB stick

> It is advisable to archive all relevant data before updating a software package.

> If an older version of KUKA.PLC Multiprog is installed, this must be uninstalled prior to installation of KUKA.PLC Multiprog 5-35.

Procedure

1. Place the CD/DVD in the CD/DVD drive or plug the USB stick into the laptop/PC.
2. Select the drive in Windows Explorer.
3. Start the program Setup.exe in the directory 00193181;xx; KUKA.PLC Multiprog 5-35 4.1; V__41KPM_xxxx. KUKA.PLC Multiprog 5-35 is installed.
4. Remove CD/DVD from the drive or unplug USB stick.

> Further information is contained in the relevant manufacturer documentation.

4.3 Uninstalling KUKA.PLC Multiprog 5-35

> It is advisable to archive all relevant data before updating or uninstalling a software package.

Precondition

- Local administrator rights

Procedure

1. In the Windows Start menu, select **Settings** > **Control Panel** > **Software**, and delete the entry **MULTIPROG 5.35 [...]**.
2. Reply to the request for confirmation with **Yes**. KUKA.PLC Multiprog 5-35 is uninstalled.

5 Operation

Information about operating Multiprog can be found in the Multiprog online help.

6 Configuration

6.1 Overview

Step	Description
1	■ Open Multiprog project in WorkVisual. (>>> 6.2 "Opening a Multiprog project in WorkVisual" Page 19) or ■ Import Multiprog project into WorkVisual. (>>> 6.3 "Importing a Multiprog project into WorkVisual" Page 20)
2	Map Multiprog variables. (>>> 6.4 "Mapping Multiprog variables" Page 20)

Additional information about procedures in WorkVisual is contained in the WorkVisual documentation.

Multiprog variables can be generated, edited and deleted in Multiprog. Further information about this can be found in the Multiprog online help.

6.2 Opening a Multiprog project in WorkVisual

Precondition

- Multiprog is not open.
- A project is open in WorkVisual.
- The robot controller is added.

Procedure

1. Right-click on the robot controller on the **Hardware** tab in the **Project structure** window and select **Add…** in the context menu.
2. A window opens. Select the element **PROCONOS 4-1** and confirm with **Add**.

   The element **PROCONOS 4-1** is added to the tree structure of the robot controller beneath the **Options** folder.
3. Set the robot controller as the active controller.
4. Select the menu sequence **Editors** > **Option packages** > **PLC editor**.
5. Only if the PLC editor is being started for the first time: a selection window containing templates opens. Select the desired template and confirm with **OK**.

Further information about the templates can be found in the documentation “Compatibility of Multiprog/ProConOS”. This documentation can be found on the ProConOS CD.

Multiprog is opened and a connection to WorkVisual is established. In WorkVisual, the information from the Multiprog project is displayed in the **PLC** tab in the **I/O Mapping** window.

Multiprog can be closed again via the menu sequence **Editors** > **Option packages** > **PLC editor**.

6.3 Importing a Multiprog project into WorkVisual

Precondition

- A project is open in WorkVisual.
- The robot controller is added.

Procedure

1. Select the menu sequence **File** > **Import / Export**.
   The **Import/Export Wizard** window is opened.
2. Select **Import Multiprog project** and click on **Next >**.
3. Click on **Browse…** and specify a directory.
4. Select the file to be imported and confirm with **Open**.
5. Select the robot controller into which the project is to be imported.

> Only robot controllers that have not been set as the active controller can be selected.

6. Click on **Finish**.
   The project is imported.

6.4 Mapping Multiprog variables

Description

Multiprog variables can be mapped to KRC variables and field bus signals.

> Before mapping is carried out, variables in Multiprog are input and output variables. Only when it is mapped is a variable defined as either an input variable or an output variable.

Variables with the following data types can be mapped to one another:

IEC data type	KRC data type	Remark
DINT	INT/ENUM	
BOOL	BOOL	
REAL	REAL	
BYTE	CHAR	
STRING	CHAR[]	The array limits are checked during reading or writing. In case of doubt, the maximum permissible length is read/written and any remaining characters are simply cut off.
DINT array	INT array or ENUM array	The number of elements must match exactly.
REAL array	REAL array	The number of elements must match exactly.
BOOL array	BOOL array	
KRL_FRAME	FRAME	KRL_Frame is a newly introduced data type in Multiprog, in the folder “Data types”.
KRL_POS	POS	KRL_POS is a newly introduced data type in Multiprog, in the folder “Data types”.
KRL_E3POS	E3POS	KRL_E3POS is a newly introduced data type in Multiprog, in the folder “Data types”.
KRL_E6POS	E6POS	KRL_E6POS is a newly introduced data type in Multiprog, in the folder “Data types”.

IEC data type	KRC data type	Remark
AXIS	AXIS	AXIS is an existing data type in Multiprog.
KRL_E3AXIS	E3AXIS	KRL_E3AXIS is a newly introduced data type in Multiprog, in the folder “Data types”.
KRL_E6AXIS	E6AXIS	KRL_E6AXIS is a newly introduced data type in Multiprog, in the folder “Data types”.

> By splitting KRC variables, it is possible to map sub-elements of arrays and structures.

Precondition

- The Multiprog project has been opened in WorkVisual or imported.

Procedure

1. Select the element **Global_Variables** on the **PLC** tab in the right-hand half of the **I/O Mapping** window.

   The variables are displayed in the bottom area of the **I/O Mapping** window.
2. Select the variables/signals to be mapped and click on the **Connect** button.

   The variables/signals are now mapped.
3. In the case of KRC variables, the direction of the mapping can be changed:
   a. Right-click on the mapped variables.
   b. Select **Change mapping direction**.

7 Programming

ℹ Information about programming Multiprog can be found in the Multiprog online help.

8 Libraries

8.1 ExtensionLibV8 library

The file functions in the ExtensionLibV8 library available in earlier versions of Multiprog have been replaced with the file functions from the ProConOS library. Further information about the ProConOS library can be found in the Multiprog online help.

8.1.1 Accessing robot controller signals

Description

The function blocks KrcSignalRead and KrcSignalWrite allow the inputs and outputs of the robot controller to be accessed symbolically. This enables I/O assignment without a PLC project and without Multiprog. The assignment between the symbol and the I/O address is made by means of a robot controller signal declaration.

8.1.1.1 Reading KR C signals

Description

The function block KrcSignalRead allows KR C signals to be read symbolically.

It is not possible to read inputs and outputs of the KR C that are mapped to an I/O driver.

The function block can only be used in conjunction with communication I/Os or globally defined signal variables. The signals must not be mapped to a field bus signal.

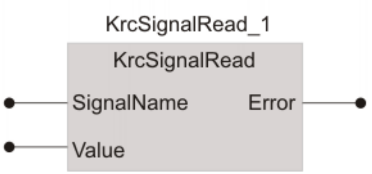


Fig. 8-1: KrcSignalRead function block symbol

Parameter	Data type	Description
SignalName	STRING	Name of the signal that is to be read **Note**: The name must be written in upper-case characters both on the robot controller and in the Multiprog project; otherwise it will not be recognized.
Value	ANY	Read value. Although the parameter appears as an input, the read value is written to the linked variable.
Error	INT	Error codes: see the error code table

Error code	Description
0	Successful
-2	Signal not found
-3	Incorrect signal type
-4	Unsupported representation width of the signal

Error code	Description
-5	Internal initialization error
-6	Buffer too small
-10	Signal not in I/O range of the KR C
-12	System I/O. Output assigned to the robot controller.
-13	Input/output is mapped to a field bus in the KRC

8.1.1.2 Writing KR C signals

Description The function block KrcSignalWrite allows KR C signals to be written symbolically.

It is not possible to write to inputs and outputs of the KR C that are mapped to an I/O driver.

The function block can only be used in conjunction with communication I/Os or globally defined signal variables. The signals must not be mapped to a field bus signal.

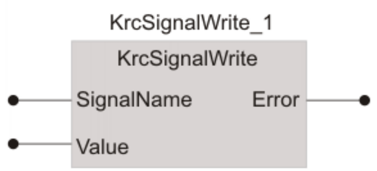


Fig. 8-2: KrcSignalWrite function block symbol

Parameter	Data type	Description
SignalName	STRING	Name of the signal that is to be written **Note**: The name must be written in upper-case characters both on the robot controller and in the Multiprog project; otherwise the name will not be recognized.
Value	ANY	Value to be written
Error	INT	Error codes: see the error code table

Error code	Description
0	Successful
-2	Signal not found
-3	Signal mapped to I/O driver in the KR C
-4	Unsupported representation width of the signal
-5	Internal initialization error
-6	Not a SPOC-safe state
-10	Signal not in I/O range of the KR C
-12	System I/O. Output assigned to the robot controller.
-13	Input/output is mapped to a field bus in the KRC

8.1.2 Saving retentive data using a program

Description The function block SaveRetain can be used to save retentive data via PLC programs.

Due to the time required for saving data (several hundred ms), this function block should be executed in the SPG2 stop task, or in a task which is not monitored by the watchdog.

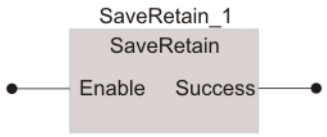


Fig. 8-3: SaveRetain function block symbol

Parameter	Data type	I/O	Description
Enable	BOOL	IN	A rising edge at this input activates the functionality of the function module.
Success	BOOL	OUT	FALSE: An error occurred when saving the retentive data.
			TRUE: The retentive data were saved successfully.

8.1.3 Provide CPU computing time

Description

The function block PlcSleep can be used to provide CPU computing time to lower-priority tasks of the lower-level operating system.

NOTICE This function can adversely affect the real-time behavior of the robot system and result in damage to the robot. The parameter may only be used in consultation with KUKA Roboter GmbH.

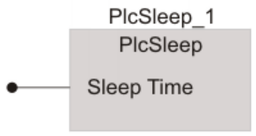


Fig. 8-4: PlcSleep function block symbol

Parameter	Data type	I/O	Description
SleepTime	INT	IN	CPU computing time (in ms) which is to be provided.

8.2 KrcLibV8 library

Description

This library carries out read and write access to robot data. During installation, the library is inserted as a project library in Multiprog.

8.2.1 Stopping the robot

Description

The function block RobStop can be used to stop the robot in one of 2 ways: path-maintaining braking or ramp stop.

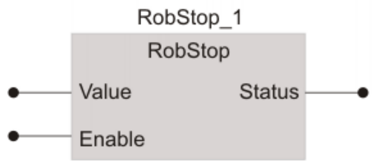


Fig. 8-5: RobStop function block symbol

Parameter	Data type	I/O	Description
Value	SINT	IN	1: Ramp stop 2: Path-maintaining braking
Enable	BOOL	IN	Trigger for execution of the stop (rising edge)
Status	SINT	OUT	0: Execution successful -1: Execution failed

8.2.2 Canceling the robot stop

Description

The function block RobStopRel can be used to reset the messages that stopped the robot with the function block RobStop.

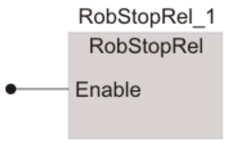


Fig. 8-6: RobStopRel function block symbol

Parameter	Data type	I/O	Description
Enable	BOOL	IN	Trigger for resetting the messages (rising edge)

8.2.3 Reading the current actual position values of axes A1 to A12

Description

The function block ReadAxisAct_Md can be used to read by means of the variable **$AXIS_ACT** the current actual robot position values of axes A1 to A12.

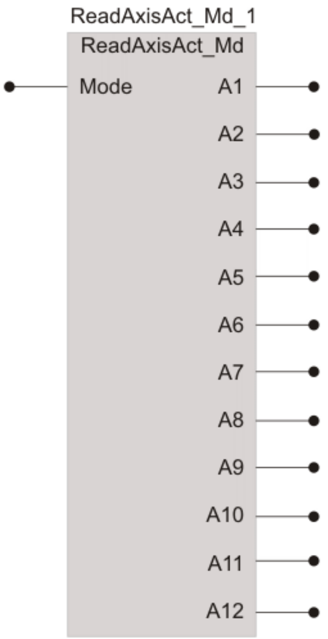


Fig. 8-7: Function block system ReadAxisAct_Md

Parameter	Data type	I/O	Element	Description
Mode	BYTE	IN	Mode 0x01	RdAXIS_ACT_MES()
			Mode 0x02	RdAXIS_ACT()
			Mode 0x03	RdAXIS_ACT_FLT_KRL_Units()
			Mode 0x04	RdAXIS_ACT_FLT()
			Mode 0x11	RdAXIS_ACT_MES() [Modulo-Calc.]
			Mode 0x12	RdAXIS_ACT() [Modulo-Calc.]
			Mode 0x13	RdAXIS_ACT_FLT_KRL_Units() [Modulo-Calc.]
			Mode 0x14	RdAXIS_ACT_FLT() [Modulo-Calc.]
A1	REAL	OUT	$AXIS_ACT.A1	Angle A1
A2	REAL	OUT	$AXIS_ACT.A2	Angle A2
A3	REAL	OUT	$AXIS_ACT.A3	Angle A3
A4	REAL	OUT	$AXIS_ACT.A4	Angle A4
A5	REAL	OUT	$AXIS_ACT.A5	Angle A5
A6	REAL	OUT	$AXIS_ACT.A6	Angle A6
A7	REAL	OUT	$AXIS_ACT.A7	Angle A7
A8	REAL	OUT	$AXIS_ACT.A8	Angle A8
A9	REAL	OUT	$AXIS_ACT.A9	Angle A9
A10	REAL	OUT	$AXIS_ACT.A10	Angle A10
A11	REAL	OUT	$AXIS_ACT.A11	Angle A11
A12	REAL	OUT	$AXIS_ACT.A12	Angle A12

8.2.4 Reading the current position of the base origin

Description

The function block ReadBaseAct can be used to read, by means of the variable **$BASE_ACT**, the current position of the base origin.

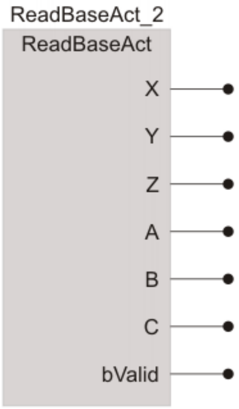


Fig. 8-8: ReadBaseAct function block symbol

Parameter	Data type	I/O	Element	Description
X	REAL	OUT	$ACT_BASE.X	X coordinate
Y	REAL	OUT	$ACT_BASE.Y	Y coordinate
Z	REAL	OUT	$ACT_BASE.Z	Z coordinate
A	REAL	OUT	$ACT_BASE.A	Orientation A
B	REAL	OUT	$ACT_BASE.B	Orientation B
C	REAL	OUT	$ACT_BASE.C	Orientation C
bValid	BOOL	OUT	-	TRUE: base coordinates are valid. FALSE: base coordinates are not valid.

8.2.5 Reading the current override value of the robot controller

Description

The function block ReadOvPro can be used to read, by means of the variable **$OV_PRO**, the current override value of the robot controller.

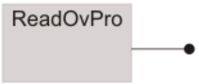


Fig. 8-9: ReadOvPro function block symbol

Parameter	Description	Range of values
$OV_PRO	The return value is the override value.	0...100

8.2.6 Setting the current override value of the robot controller

Description

The function block WriteOvPro can be used to set, by means of the variable **$OV_PRO**, the current override value of the robot controller.

The input parameter is the override value within a value range from 0 to 100.

> The variable can only be written if AUT or AUT EXT mode is set and the safety gate is closed.

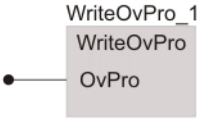


Fig. 8-10: WriteOvPro function block symbol

8.2.7 Reading the current actual values of the robot position

Description

The function block ReadPosAct_Md can be used to read, by means of the variable **$POS_ACT**, the current actual values of the robot position.

> **NOTICE** This function can adversely affect the real-time behavior of the robot system and result in damage to the robot. To prevent this, the actual position should be read by a task that is processed a maximum of once every 12 ms.

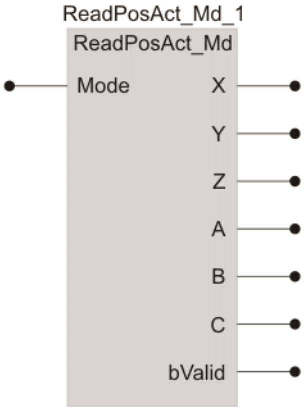


Fig. 8-11: ReadPosAct_Md function block symbol

Parameter	Data type	I/O	Description
Mode	BYTE	IN	Mode = 1: $Pos_Act in BASE
			Mode = 2: $Pos_Act in WORLD
X	REAL	OUT	X coordinate
Y	REAL	OUT	Y coordinate
Z	REAL	OUT	Z coordinate
A	REAL	OUT	Orientation A
B	REAL	OUT	Orientation B
C	REAL	OUT	Orientation C

8.2.8 Reading robot controller variables (Integer-type)

Description

The function block ReadSenInt can be used to read 20 Integer-type variables of the robot controller which are available for KRL programming. These variables must contain freely available integer values. The variables are contained in the array $SEN_PINT[].

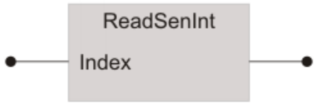


Fig. 8-12: ReadSenInt function block symbol

Parameter	Data type	I/O	Description
Index	BYTE	IN	Index of the robot controller variables from 1 to 20
Function result	DINT	OUT	Value of variable

8.2.9 Writing to robot controller variables (Integer-type)

Description

The function block WriteSenInt can be used to write to 20 Integer-type variables of the robot controller which are available for KRL programming. These variables must contain freely available integer values. The variables are contained in the array $SEN_PINT[].

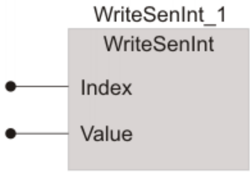


Fig. 8-13: WriteSenInt function block symbol

Parameter	Data type	I/O	Description
Index	BYTE	IN	Index of the robot controller variables from 1 to 20
Function result	DINT	OUT	Value of variable

8.2.10 Reading robot controller variables (REAL-type)

Description

The function block ReadSenReal can be used to read 20 REAL-type variables of the robot controller which are available for KRL programming. These variables must contain freely available real values. The variables are contained in the array $SEN_PREA[].

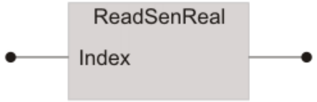


Fig. 8-14: ReadSenReal function block symbol

Parameter	Data type	I/O	Description
Index	BYTE	IN	Index of the robot controller variables from 1 to 20
Function result	REAL	OUT	Value of variable

8.2.11 Writing to robot controller variables (REAL-type)

Description

The function block WriteSenReal can be used to write to 20 REAL-type variables of the robot controller which are available for KRL programming. These variables must contain freely available real values. The variables are contained in the array $SEN_PREA[].

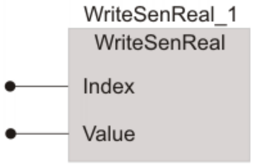


Fig. 8-15: WriteSenReal function block symbol

Parameter	Data type	I/O	Description
Index	BYTE	IN	Index of the robot controller variables from 1 to 20
Function result	REAL	IN	Value of variable

8.2.12 Reading the current position of the tool origin

Description

The function block ReadToolAct can be used to read, by means of the variable **$POS_TOOL**, the current position of the tool origin.

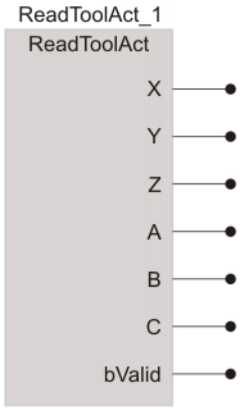


Fig. 8-16: ReadToolAct function block symbol

Parameter	Data type	I/O	Element	Description
X	REAL	OUT	$ACT_BASE.X	X coordinate
Y	REAL	OUT	$ACT_BASE.Y	Y coordinate
Z	REAL	OUT	$ACT_BASE.Z	Z coordinate
A	REAL	OUT	$ACT_BASE.A	Orientation A
B	REAL	OUT	$ACT_BASE.B	Orientation B
C	REAL	OUT	$ACT_BASE.C	Orientation C
bValid	BOOL	OUT	-	TRUE: tool present. FALSE: no tool present.

8.2.13 Reading the current operating mode

Description

The function block ReadModeOp can be used to read, by means of the variable **$MODE_OP**, the current operating mode.

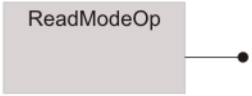


Fig. 8-17: ReadModeOp function block symbol

Parameter	Data type	I/O	Description
Return value	BYTE	OUT	1: T1
			2: T2
			3: AUT
			4: EXT
			5: Invalid

8.2.14 Reading the current state of the submit and robot interpreters

Description

The function block ReadProState can be used to read, by means of the variable **$PRO_STATE**, the current state of the submit and robot interpreters.

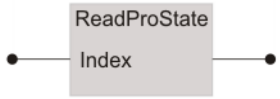


Fig. 8-18: ReadProState function block symbol

Parameter	Data type	I/O	Description
Index	BYTE	IN	0: Submit interpreter
			1: Robot interpreter
Return value	BYTE	OUT	0: Invalid index
			1: No program selected
			2: Program selected but not yet started
			3: Program is being executed
			4: Program stopped
			5: Program has been completely executed

8.2.15 Reading the interpolation mode of the robot

Description

The function block ReadIpoMode has the following characteristics:

- The return value of the variable **$IPO_MODE_C** is the valid interpolation mode in the robot main run.
- If the robot is stopped and then jogged manually, the value of **$IPO_MODE_C** remains set to the last valid program mode value.
- If the variable **$IPO_MODE** is modified by the user while the robot is stopped in program interpolation mode, the variable **$IPO_MODE_C** also assumes the modified value. If the program interpolator is restarted, the system implicitly switches back to the value that was previously valid in the interpolator.
- In the case of command motions in the interrupt, **$IPO_MODE_C** is set to the value of **$IPO_MODE** that is valid in the interrupt level.
- When a program is deselected, the last valid interpolation mode in the interpolator remains set.

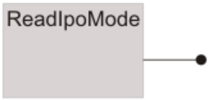


Fig. 8-19: ReadIpoMode function block symbol

Parameter	Data type	I/O	Description
Return value	BYTE	OUT	1: #BASE
			2: #TCP

8.2.16 Displaying notification messages

Description The function block DisplayKCPNotifyMsg can be used to display notification messages to the user in the PLC application.

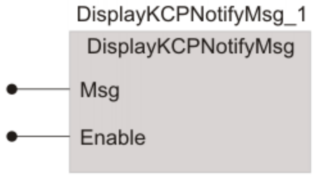


Fig. 8-20: DisplayKCPNotifyMsg function block symbol

Parameter	Data type	I/O	Description
Msg	STRING	IN	Notification message to be displayed (max. 44 characters)
Enable	BOOL	IN	Trigger to start displaying message (rising edge)

8.2.17 Displaying status messages

Description The function block DisplayKCPStatusMsg can be used to display status messages to the user in the PLC application. The function returns an unambiguous message handle. The message handle can be used to have the specific status message cleared by the PLC program.

⚠ This function should not be used to generate cyclical messages. Otherwise there is a risk that individual messages can no longer be displayed, or that the robot controller message buffer will overflow.

⚠ If an identical status message is generated repeatedly, without being cleared first, the system stops the program that is causing this after the 10th identical message.

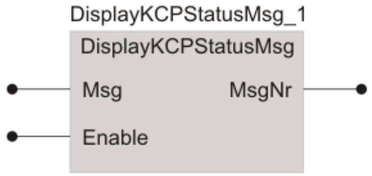


Fig. 8-21: DisplayKCPStatusMsg function block symbol

Parameter	Data type	I/O	Description
Msg	STRING	IN	Message to be displayed (max. 44 characters)
Enable	BOOL	IN	Trigger to start displaying message (rising edge)
MsgNr	DINT	OUT	Unambiguous message handle 0: Message could not be displayed.

Timing diagram

The figure shows the timing diagram for the status message “External EMERGENCY STOP”.

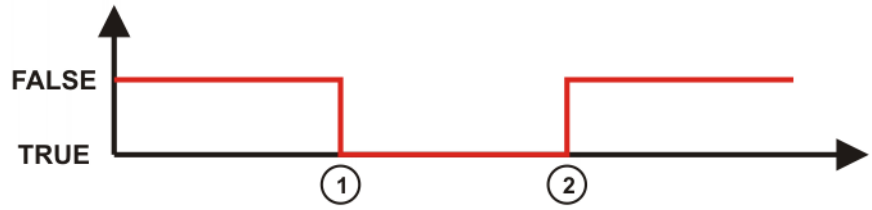


Fig. 8-22: Timing diagram “External EMERGENCY STOP”

1 Message is generated.

2 Message is cleared.

8.2.18 Clearing a status message

Description

The function block ClearKCPStatusMsg can be used to have a status message cleared by the PLC program. Status messages cannot be acknowledged by the user.

⚠ Only message handles returned by the function block DisplayKCPStatusMsg as a return value may be used.

ℹ To prevent a message handle from being used a second time, it should be set to 0 after a message has been cleared.

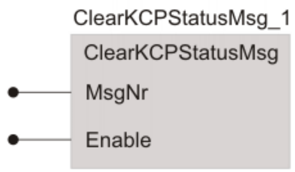


Fig. 8-23: ClearKCPStatusMsg function block symbol

Parameter	Data type	I/O	Description
MsgNr	DINT	IN	Unambiguous message handle from previous call of DisplayKCPStatusMsg. (>>> 8.2.17 "Displaying status messages" Page 35)
Enable	BOOL	IN	Trigger to start clearing message (rising edge)

8.2.19 Clearing all status messages

Description

The function block ClearAllKCPStatusMsg can be used to clear all messages that have been displayed by means of the function block DisplayKCPStatusMsg.

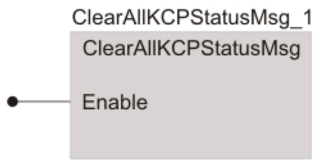


Fig. 8-24: ClearAllKCPStatusMsg function block symbol

Parameter	Data type	I/O	Description
Enable	BOOL	IN	Trigger to start clearing message (rising edge)

8.2.20 Converting the message number of the ISG core into plain text and displaying it

Description The function block DisplayIsgMsg converts message numbers of the ISG core into plain text and displays the corresponding notification messages in the message window of the smartPAD.

This module is only required if KUKA.CNC and Extended Motion are used.

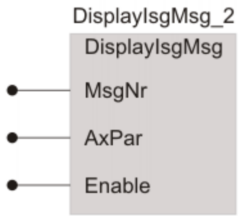


Fig. 8-25: DisplayIsgMsg function block symbol

Parameter	Data type	I/O	Description
MsgNr	DINT	IN	Message number of the ISG core
AxPar	STRING	IN	Additional string parameter before the displayed message
Enable	BOOL	IN	Trigger for displaying the message (rising edge)

8.2.21 Displaying/clearing status messages (simplified)

Description The function block KRCStateMsg is a simplified function block for displaying and clearing a status message. The message handles are used internally by the function block. Access to the message handles is thus no longer required.

A rising edge at the input EnableMsg causes the status message to be displayed. A falling edge at the input EnableMsg causes the status message to be cleared again.

In the following cases, the messages displayed by this function block are implicitly cleared by the system software:

- Reboot of the PLC after a stop in cold start mode
- Reboot of the PLC after a stop in warm start mode
- System reconfiguration

This function block can only be used with KUKA System Software 8.3 or higher.

The messages displayed by this function block cannot be cleared by the function blocks ClearKCPStatusMsg and ClearAllKCPStatusMsg.

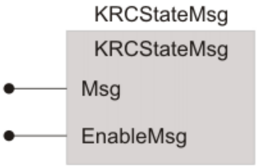


Fig. 8-26: KRCStateMsg function block symbol

Parameter	Data type	I/O	Description
Msg	STRING	IN	Message to be displayed (max. 44 characters)
EnableMsg	BOOL	IN	Trigger to start displaying message (rising edge)

8.2.22 Reading the current state of the drives

Description

The function block RdPeriReady can be used to read, by means of the variable **$PERI_RDY**, the current state of the drives.

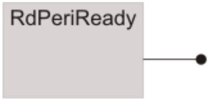


Fig. 8-27: RdPeriReady function block symbol

Parameter	Data type	I/O	Description
$PERI_RDY	BOOL	OUT	■ **TRUE**: Drives are switched on. ■ **FALSE**: Drives are switched off.

8.2.23 Reading the current state of the robot brakes

Description

The function block RdBrakeSig can be used to read, by means of the variable **$BRAKE_SIG**, the current state of the robot brakes.

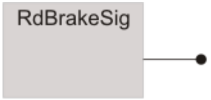


Fig. 8-28: RdBrakeSig function block symbol

Parameter	Data type	I/O	Description
$BRAKE_SIG	UINT	OUT	Bit array for the brake signals of the robot axes of the robot controller. Each bit stands for a robot axis: Bit_0 = axis 1, Bit_1 = axis 2, etc. The bit values have the following meaning: ■ 0: The robot axis is under servo-control. ■ 1: The holding brake for the robot axis is activated. **Examples:** ■ $Brake_Sig = 0 (binary: 000000000000): All robot axes are under servo-control. ■ $Brake_Sig = 63 (binary: 000000111111): The holding brakes for axis 1 to axis 6 are activated.

8.2.24 Displaying/clearing messages

Description

The KrcUserMsg function block is used for displaying notification, status, acknowledgement and wait messages, as well as for clearing status and wait messages. Notification and acknowledgement messages are not cleared. In addition, the function block ensures the correct handling of acknowledgement checkback signals and simulation checkback signals.

The message handles are used internally by the function block. Access to the message handles is thus not required.

A rising edge at the input EnableMsg causes the message to be displayed. A falling edge at the input EnableMsg causes the status or wait message to be cleared again.

In the following cases, the status and wait messages displayed by this function block are implicitly cleared by the system software:

- Reboot of the PLC after a stop in cold start mode
- Reboot of the PLC after a stop in warm start mode
- System reconfiguration

> The messages displayed by this function block cannot be cleared by the function block ClearAllKCPStatusMsg.

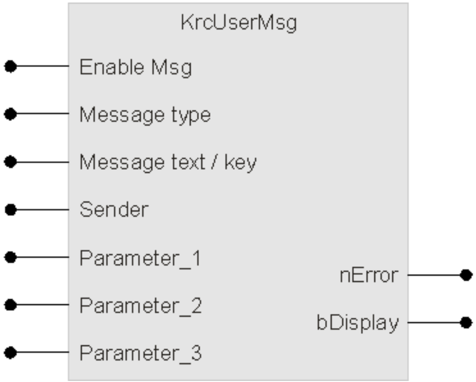


Fig. 8-29: KrcUserMsg function block symbol

Parameter	Data type	I/O	Description
Enable Msg	BOOL	IN	- **TRUE**: Message is displayed. - **FALSE**: Message is cleared (status and wait messages only).
Message type	BYTE	IN	- 0: Notification message - 1: Status message - 2: Acknowledgement message - 3: Wait message
Message text / key	STRING	IN	Message text with placeholders for the parameters, or database key for the KXR file. Maximum length: 80 characters **Note**: Message texts or keys which are longer than 80 characters result in an error reaction at the output nError.
Message number	USINT	IN	Message number on the HMI
Sender	STRING	IN	Sender ID on the HMI **Note**: ProConOS messages are identified by means of square brackets as the sender.
Parameter_1	ANY	IN	1st parameter for the message, placeholder: %1 - STRING: Is transferred directly to the message. - SINT, INT, DINT, USINT, UINT, UDINT, WORD, DWORD, BYTE: Are converted into the string representation of the integer value and then sent as a message parameter to the robot controller. - BOOL: Converted into the string representation TRUE/FALSE and displayed as a parameter. - REAL, LREAL: Converted into the string representation (e.g. 0.01234) and displayed as a parameter. Maximum length: 44 characters If the maximum length is exceeded, the parameter is implicitly shortened to the maximum length. **Note**: If the parameter is not required, omit the value or the variable.
Parameter_2	ANY	IN	2nd parameter for the message, placeholder: %2 **Note**: The data type has the same effects and the same maximum length as Parameter_1. If the parameter is not required, omit the value or the variable.

Parameter	Data type	I/O	Description
Parameter_3	ANY	IN	3rd parameter for the message, placeholder: %3 **Note**: The data type has the same effects and the same maximum length as Parameter_1. If the parameter is not required, omit the value or the variable.
nError	INT	OUT	Error code if a problem occurs on displaying the message: ■ 0: No error ■ -1: Message type is invalid ■ -2: Message key is too long (max. 80 characters allowed) ■ -3: Maximum number of messages has been reached (max. 20 status, wait and acknowledgement messages are possible) ■ -4: Maximum number of messages per second has been reached (max. 20 messages every 4 seconds are possible) ■ -5: General error on sending the message to the robot controller ■ -6: Interface to the robot controller is not available (e.g. if ProConOS is operated without a robot controller) ■ -11: Invalid data type in the 1st message parameter ■ -12: Invalid data type in the 2nd message parameter ■ -13: Invalid data type in the 3rd message parameter ■ -14: Error during generation of the ProConOS message in the ProConOS message management facility ■ -15: Invalid reference to the ProConOS message management facility ■ -16: General ProConOS error in the message function block
bDisplay	BOOL	OUT	Indicates that the message has been displayed in the message window of the HMI. ■ Notification messages: The parameter has no meaning and is therefore always FALSE. ■ Status messages: The parameter is TRUE for as long as the message is set. ■ Acknowledgement messages: The parameter is TRUE as long as the message has not been acknowledged. ■ Wait messages: The parameter is TRUE as long as the message has not been simulated in the message window.

Examples

Message text	Connection to device %1, device %2 and device %3 is established.
Parameter	■ Parameter_1: String “Hugo” ■ Parameter_2: String “Otto” ■ Parameter_3: Integer “55”
Result	Connection to device Hugo, device Otto and device 55 is established.

Message text	Parameter 1 is %1 and parameter 2 has the value %2.
Parameter	■ Parameter_1: Bool “TRUE” ■ Parameter_2: Real “0.0123”
Result	Parameter 1 is TRUE and parameter 2 has the value 0.0123.

8.3 KrcExVarLib library

Description

This library makes it possible to read and write in an array. The robot controller makes available the following array variables, which can be used for data exchange between ProConOS and the robot controller software:

- $SOFTPLCBOOL[1...n]
- $SOFTPLCINT[1...n]
- $SOFTPLCREAL[1...n]

8.3.1 Reading a value from an array

Description

The function blocks ReadPLCBool, ReadPLCInt and ReadPLCReal can each be used to read a single value from an array.

Example

The example shows the function block ReadPLCBool, which reads the value of the KR C variable $SOFTPLCBOOL[x] and assigns it to the ProConOS variable at the output “Value”. The result is saved in the variable at the output “Result”.

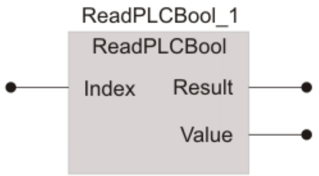


Fig. 8-30: RealPLCBool function block symbol

Parameter	Data type	I/O	Description
Index	INT	IN	Index valid from 1 to n
Value	BOOL	OUT	Value from the array
	DINT	OUT	
	REAL	OUT	
Result	BOOL	OUT	TRUE: Error
			FALSE: OK

8.3.2 Writing a value to an array

Description

The function blocks WritePLCBool, WritePLCInt and WritePLCReal can each be used to write a single value to an array.

Example

The example shows the function block WritePLCInt, which reads the value of the KR C variable $SOFTPLCBOOL[x] and assigns it to the ProConOS variable at the output “Value”. The result is saved in the variable at the output “Result”.

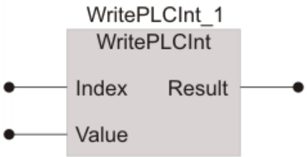


Fig. 8-31: WritePLCInt function block symbol

Parameter	Data type	I/O	Description
Index	INT	IN	Index valid from 1 to n
Value	BOOL	OUT	Value from the array
	DINT	OUT	
	REAL	OUT	
Result	BOOL	OUT	TRUE: Error
			FALSE: OK

8.3.3 Reading multiple values from an array

Description

The function blocks ReadPLCBoolEx, ReadPLCIntEx and ReadPLCRealEx can be used to read a number of values from an array.

Example

The example shows the function block ReadPLCBoolEx, which reads the value of the KR C variable $SOFTPLCBOOL[x...y] and assigns it to the ProConOS array at the output “BoolArray”. The result is saved in the variable at the output “Result”.

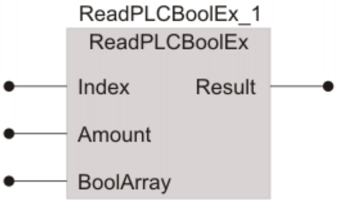


Fig. 8-32: ReadPLCBoolEx function block symbol

Parameter	Data type	I/O	Description
Index	INT	IN	Start index for reading within a range of values from 1 to 28.
Amount	BYTE	IN	Number of variables to be read
BoolArray	BOOL	OUT	ProConOS ARRAY[1 to n] which receives the data from $SOFTPLCBOOL[1 to 128]. Used with ReadPLCBoolEx.
DintArray	DINT	OUT	ProConOS ARRAY[1 to n] which receives the data from $SOFTPLCBOOL[1 to 128]. Used with ReadPLCIntEx.
RealArray	REAL	OUT	ProConOS ARRAY[1 to n] which receives the data from $SOFTPLCBOOL[1 to 128]. Used with ReadPLCRealEx.
Result	BOOL	OUT	TRUE: OK
			FALSE: Error

8.3.4 Writing multiple values to an array

Description

The function blocks WritePLCBoolEx, WritePLCIntEx and WritePLCRealEx can be used to write a number of values to an array.

Example

The example shows the function block WritePLCRealEx, which writes the values of the ProConOS array at the input “RealArray” to the KR C variables $SOFTPLCBOOL[x...y]. The result is saved in the variable at the output “Result”.

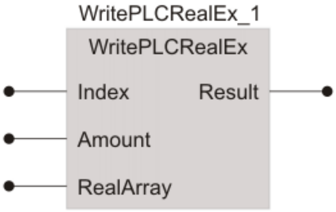


Fig. 8-33: WritePLCRealEx function block symbol

Parameter	Data type	I/O	Description
Index	INT	IN	Start index for reading within a range of values from 1 to 1024.
Amount	BYTE	IN	Number of variables to be read
BoolArray	BOOL	IN	ProConOS ARRAY[1 to n] from which the data are written to $SOFTPLCBOOL[1 to n]. Used with WritePLCBoolEx.
DintArray	DINT	IN	ProConOS ARRAY[1 to n] from which the data are written to $SOFTPLCBOOL[1 to n]. Used with WritePLCIntEx.
RealArray	REAL	IN	ProConOS ARRAY[1 to n] from which the data are written to $SOFTPLCBOOL[1 to n]. Used with WritePLCRealEx.
Result	BOOL	OUT	TRUE: OK
			FALSE: Error

8.4 AutoExtLib library

The function blocks KRC_AutoExt and VKRC_AutoExt from the ProConOS library AutoExtLib make it possible to operate the Automatic External interface of the robot controller directly, without the need to use ProConOS I/Os. If the wrong block on the robot controller is used, an error message is generated when the project is downloaded in Multiprog.

The signal declarations are read from the following files:

- C:\KRC\Roboter\KRC\R1\System\$Config.dat
- C:\KRC\Roboter\KRC\Steu\Mada\$Machine.dat

The associated I/Os can be read and written directly by the function blocks.

⚠ The entry SIGNALFILES in the file ProConOS.xml is automatically set by the setup program and should not be modified.

The signal files are read in accordance with the entry SIGNALFILEREAD. If this entry is set to BOOT, the signal files are read once when ProConOS or the robot controller is started. Otherwise, the signal files are reloaded every time

ProConOS program execution is started. If Automatic External interface signals are reconfigured, ProConOS must be stopped and restarted.

Further information is contained in the operating and programming instructions for the KUKA System Software (KSS) or VW System Software (VSS).

8.4.1 Operating the Automatic External interface (KR C)

Description

The function block KRC_AutoExt can be used in the robot controller to operate the Automatic External interface.

To select a KRL program, its program number must be applied to the input PGNO and acknowledged in accordance with the value of the variable **PGNO_VALID**. The actual program number and parity are formed according to the following values:

- PGNO_FBIT
- PGNO_LENGTH
- PGNO_TYPE
- PGNO_PARITY

If the program number at input PGNO cannot be displayed with the bits defined in PGNO_LENGTH, program number 0 is written to the input map.

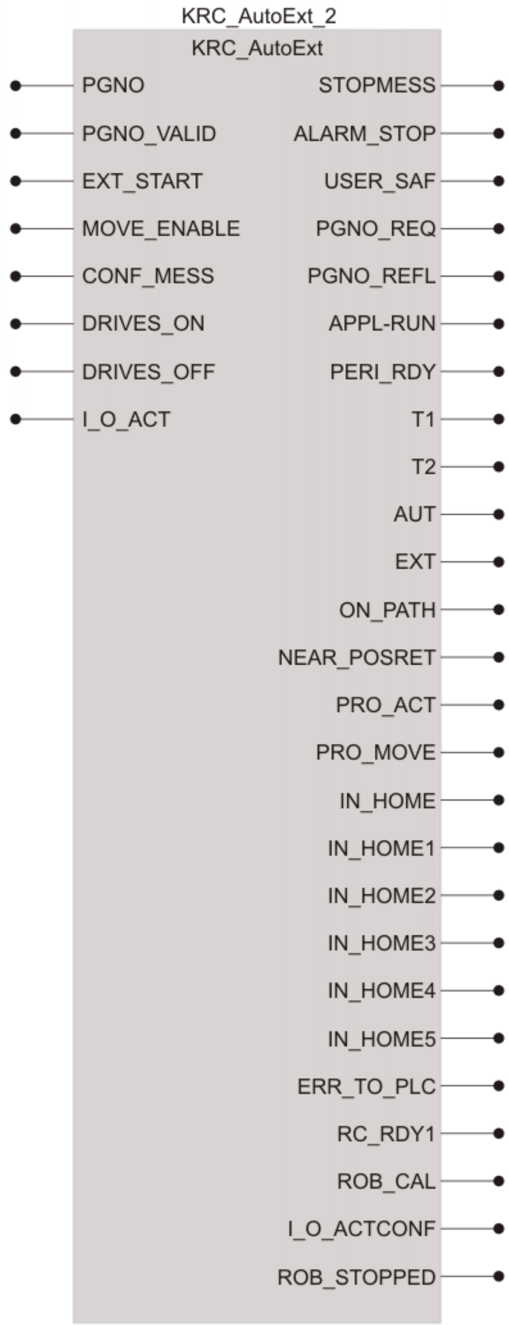


Fig. 8-34: Function block symbol KRC_AutoExt

Parameter	Data type	KRL variable	Meaning
PGNO	INT	PGNO_FBIT	Program number
PGNO_VALID	BOOL	PGNO_VALID	Program number valid
EXT_START	BOOL	$EXT_START	Start of program
MOVE_ENABLE	BOOL	$MOVE_ENABLE	Drive enable
CONF_MESS	BOOL	$CONF_MESS	Acknowledge messages
DRIVES_ON	BOOL	$DRIVES_ON	Switch on drives
DRIVES_OFF	BOOL	$DRIVES_OFF	Switch off drives
I_O_ACT	BOOL	$I_O_ACT	Activate interface
STOPMESS	BOOL	$STOPMESS	Robot collective fault
ALARM_STOP	BOOL	$ALARM_STOP	E-STOP of robot
USER_SAF	BOOL	$USER_SAF	Operator safety

Parameter	Data type	KRL variable	Meaning
PGNO_REQ	BOOL	PGNO_REQ	Program number request
PGNO_REFL	INT	PGNO_FBIT_REFL	Reflected program number, -1 if deactivated (REFLECT_PROG_NR=0)
APPL_RUN	BOOL	APPL_RUN	Robot program running
PERI_RDY	BOOL	$PERI_RDY	Drives are activated
T1	BOOL	$T1	T1 mode
T2	BOOL	$T2	T2 mode
AUT	BOOL	$AUT	Automatic mode
EXT	BOOL	$EXT	External mode
ON_PATH	BOOL	$ON_PATH	Robot on path
NEAR_POSRET	BOOL	$NEAR_POSRET	Robot near path
PRO_ACT	BOOL	$PRO_ACT	Robot program execution active
PRO_MOVE	BOOL	$PROMOVE	Robot program motion active
IN_HOME	BOOL	$IN_HOME	Robot in home position 1
IN_HOME1	BOOL	$IN_HOME1	Robot in home position 2
IN_HOME2	BOOL	$IN_HOME2	Robot in home position 3
IN_HOME3	BOOL	$IN_HOME3	Robot in home position 4
IN_HOME4	BOOL	$IN_HOME4	Robot in home position 5
IN_HOME5	BOOL	$IN_HOME5	Robot in home position 6
ERR_TO_PLC	BOOL	ERR_TO_PLC	Controller or technology fault
RC_RDY1	BOOL	$RC_RDY1	Robot controller ready
ROB_CAL	BOOL	$ROB_CAL	Robot mastered
I_O_ACTCONF	BOOL	$I_O_ACTCONF	Interface active
ROB_STOPPED	BOOL	$ROB_STOPPED	Robot stopped

8.4.2 Operating the Automatic External interface (VKR C)

Description

The function block VKRC_AutoExt can be used in the VKR C to operate the Automatic External interface.

> The function block is only available for the VW System Software (VSS).

To select a KRL Folge (program), its Folge number must be applied to the input of the FOLGE and acknowledged at the input SRB.

The Folge number is formed according to the following values:

- P_FBIT
- P_LEN
- P_TYPE

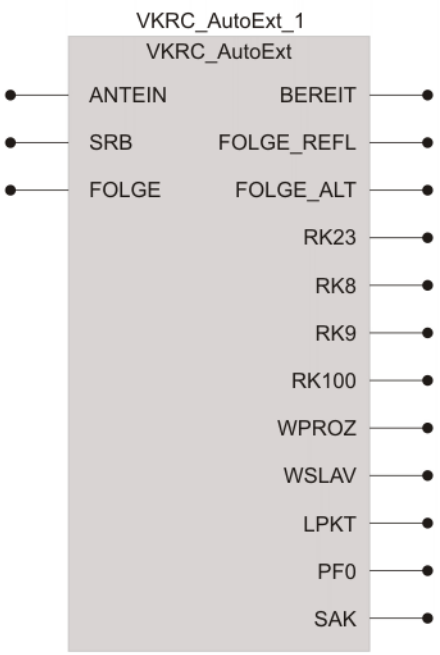


Fig. 8-35: VKRC_AutoExt function block symbol

Parameter	Data type	KRL variant	Meaning
ANTEIN	BOOL	$DRIVES_ON	Drive enable
SRB	BOOL	SRB	Start Folge
FOLGE	INT	P_FBIT	Folge number
BEREIT	BOOL	$RC_RDY1	Operating mode
FOLGE_REFL	INT	R_FBIT	Reflected Folge number
FOLGE_ALT	INT	P_OLD	Last Folge in Automatic mode
RK23	BOOL	$PR_MODE	Programming mode
RK8	BOOL	$SS_MODE	Single Step mode
RK9	BOOL	$EXT	Automatic mode
RK100	BOOL	RK100	Enable start actuators
WPROZ	BOOL	WPROZ	Wait for process
WSLAV	BOOL	WSLAV	Wait for slave
LPKT	BOOL	LPKT	Last point reached
PF0	BOOL	PF0	Home position
SAK	BOOL	$NEAR_POSRET	Block coincidence, robot on path

9 Diagnosis

9.1 Global variables

The status of global variables can provide information for diagnosis.

Variable	Description
bRetainValid	**TRUE**: Reading of the Retain data was successful last time ProConOS was started.
bSPOC_UserSafeActive	The variables are described in the chapter “Safety”. (>>> 3 "Safety" Page 11)
bSPOC_MotionEnabled	
PDD_KRC_RW_SUSPEND	**TRUE**: Access to PDD variables is temporarily suspended, e.g. due to a file download of KRL programs.
PDD_KRC_READ_NO_INIT	**TRUE**: Access to PDD variables is not possible because a KRL variable is not initialized (e.g. a variable such as $POS_INT which can temporarily become invalid).
PDD_KRC_WRITE_FAILED_SPOC	**TRUE**: Access to PDD variables is not possible because the previous SPOC state is blocking access.

10 KUKA Service

10.1 Requesting support

Introduction

The KUKA Roboter GmbH documentation offers information on operation 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:

- Model and serial number of the robot
- Model and serial number of the controller
- Model and serial number of the linear unit (if applicable)
- Model and serial number of the energy supply system (if applicable)
- Version of the KUKA System Software
- Optional software or modifications
- Archive of the software

  For KUKA System Software V8: instead of a conventional archive, generate the special data package for fault analysis (via **KrcDiag**).
- Application used
- Any external axes used
- Description of the problem, duration and frequency of the fault

10.2 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

Headland Machinery Pty. Ltd.
Victoria (Head Office & Showroom)
95 Highbury Road
Burwood
Victoria 31 25
Australia
Tel. +61 3 9244-3500
Fax +61 3 9244-3501
vic@headland.com.au
www.headland.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.
Songjiang Industrial Zone
No. 388 Minshen Road
201612 Shanghai
China
Tel. +86 21 6787-1888
Fax +86 21 6787-1803
www.kuka-robotics.cn

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 Sdn Bhd
South East Asia Regional Office
No. 24, Jalan TPP 1/10
Taman Industri Puchong
47100 Puchong
Selangor
Malaysia
Tel. +60 3 8061-0613 or -0614
Fax +60 3 8061-7386
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 Austria GmbH
Vertriebsbüro Österreich
Regensburger Strasse 9/1
4020 Linz
Austria
Tel. +43 732 784752
Fax +43 732 793880
office@kuka-roboter.at
www.kuka-roboter.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

OOO KUKA Robotics Rus
Webnaja ul. 8A
107143 Moskau
Russia
Tel. +7 495 781-31-20
Fax +7 495 781-31-19
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