KUKA

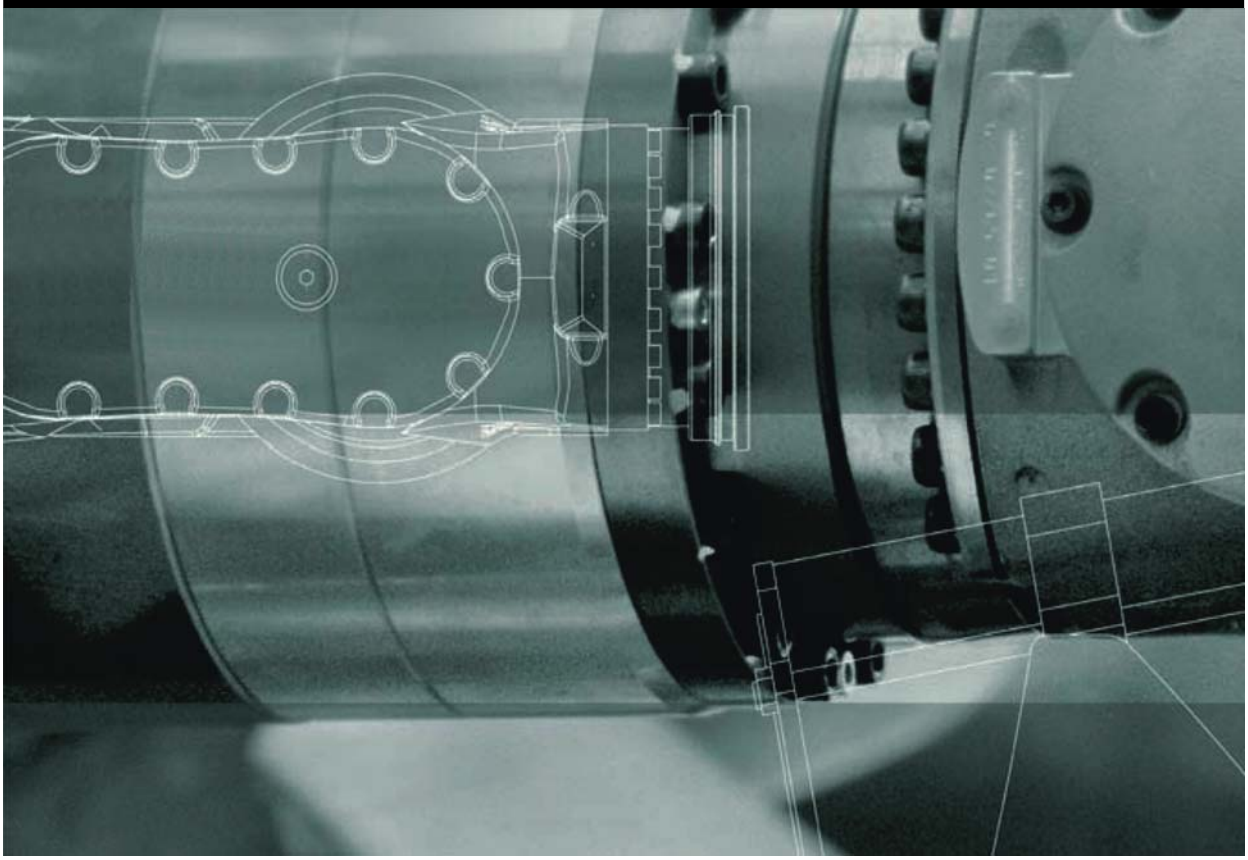
Expert Documentation

KUKA Roboter GmbH

Compatibility from 5.x to 8.x

KUKA System Software 8.x

VW System Software 8.x



Issued: 14.07.2009

Version: Kompatibilität 8.x V2 en

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 operating instructions

KIM-PS5-DOC

Publication: Pub Kompatibilität 8.x en
Book structure: Kompatibilität 8.x V4.1
Label: Kompatibilität 8.x V2

Contents

1 Introduction

1.1 Target group

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

- Expert system knowledge of the robot controller system
- Advanced knowledge of the Windows operating system
- Advanced KRL programming skills
- Basic knowledge of XML

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 Representation of warnings and notes

Safety

Warnings marked with this pictogram are relevant to safety and **must** be observed.

Danger!
This warning means that death, severe physical injury or substantial material damage **will** occur, if no precautions are taken.

Warning!
This warning means that death, severe physical injury or substantial material damage **may** occur, if no precautions are taken.

Caution!
This warning means that minor physical injuries or minor material damage **may** occur, if no precautions are taken.

Notes

Notes marked with this pictogram contain tips to make your work easier or references to further information.

Tips to make your work easier or references to further information.

1.3 Trademarks

Windows is a trademark of Microsoft Corporation.

1.4 Terms used

Term	Description
HMI	The Human-Machine Interface (HMI) is an interface which allows a human to communicate with a machine.
KSS	KUKA System Software
SmartHMI	User interface of KSS 8.x and VSS 8.x
VSS	VW System Software

2 Product description

This documentation describes new features in KSS 8.x and VSS 8.x that affect upward compatibility from 5.x to 8.x.

These primarily include:

- Elements that no longer exist in 8.x.
  e.g. certain KRL instructions or INI files.
- Elements that are used differently in 8.x than in 5.x.
  e.g. system variables whose meaning has changed.
- Modified system responses that may cause changes in program execution.

3 Communication

3.1 Modified: IP addresses and port numbers

In 8.x, IANA (Internet Assigned Numbers Authority) specifications for IP addresses and port numbers are maintained.

IP addresses

RFC 1918 from IANA is observed. The IP addresses for the shared memory network are assigned from the IP address pool for private networks.

Shared memory IP addresses:

	5.x	8.x
VxWorks	192.0.1.1	192.168.0.1
Windows	192.0.1.2	192.168.0.2

In VSS 8.x, a telnet connection to VxWorks can be opened with C:\KRC\VXWIN\PUTTYTEL.exe 192.168.0.2.

Ports

RFC 1700 from IANA is observed. The port numbers are assigned from the range 49152 to 65535 for private ports.

Important port numbers:

	5.x	8.x
Shared memory data port	4148	54100
Shared memory KCP port	5100	54101
Shared memory LTC	6000	54102

4 Configuration

4.1 INI files converted to XML files

Overview

- Most INI files from 5.x have been converted to XML files for 8.x.
- Settings that differ for KSS and VSS are contained in the same files in 8.x, but specially subdivided into different sections for KSS or VSS.
- All settings in the new path C:\KRC\ROBOTER\Config\User are maintained in the case of an update.

5.x	8.x
backward.ini	**KrcBackward.xml** in C:\KRC\ROBOTER\Config\User\Common $VW_BACKWARD is now write-protected, i.e. the file can no longer be written to via $VW_BACKWARD. Sections present for KSS and VSS.
IoDryRun.ini	**KrcDryRun.xml** in C:\KRC\ROBOTER\Config\User\Common Only contains user-specific settings. Sections present for KSS and VSS.
ext_conf.ini	**KrcExtConfMsg.xml** in C:\KRC\ROBOTER\Config\User\Common Only contains user-specific settings. In this file it is now possible to configure, for each acknowledgement message of the robot controller, whether it can be acknowledged externally.
memconfig.ini	Section `<MemConfig>` in **KrcConfig.xml** in C:\KRC\ROBOTER\Config\User\Common
progress.ini	User-specific settings: **KrcConfig.xml** in C:\KRC\ROBOTER\Config\User\Common For some settings, separate sections are present for KSS and VSS.
	System settings: **KRC.xml** in C:\KRC\ROBOTER\Config\System\Common
	Settings for the dry run: **KrcDryRun.xml** in C:\KRC\ROBOTER\Config\User\Common Sections present for KSS and VSS.
wsrestore.ini	Remains **wsrestore.ini**, but now in the directory C:\KRC\ROBOTER\IR_SPEC. This file only exists under the following conditions: ■ KUKA.CR ProgramCooperation or KUKA.CR MotionCooperation is installed. ■ The robot controller has been rebooted with a cold restart.
modsize.ini	Remains **modsize.ini**, but now in the directory C:\KRC\ROBOTER\IR_SPEC. The robot controller creates the file in the following cases: ■ The main switch on the robot controller is turned to OFF. ■ The robot controller is shut down by means of the menu sequence **File** > **Shut down KRC**.
serial.ini	No longer exists. Not necessary, as 8.x comes as standard without a configurable serial interface.
motiondrv.ini	Remains **motiondrv.ini**, but now in the directory C:\KRC\ROBOTER\INIT.

5.x	8.x
pl_driver.ini	Remains **pl_driver.ini**, but now in the directory C:\KRC\ROBOTER\INIT.
iosys.ini	(>>> 4.2 "Modified: I/O manager" page 12)

Example Excerpt from **KrcBackward.xml**, with sections for KSS and VSS:

```
<Version Label="">
    <Info Version="KRC V8.0"/>
    <Schema Version="1"/>
  </Version>

  <!-- ******************** -->
  <!-- Customer Version "VW" -->
  <!-- ******************** -->
  <Customer_VW>
    <TRACE ENABLE="true" FINISHED_SUB="STOP" MOVEMENTS="30"
CYCFLAGS="5"/>
    <OFC SET_TO_FALSE="true" TRACE="AT_LEAVING" RESTORE="AT_FWD"/>
    <SCAN ENABLE="false"/>
    <GENERAL BACKWARDSTART="false" IMPLICIT_BCO="true"
QUIT_M_TRACE_SCAN="true"/>
  </Customer_VW>

  <!-- ********************** -->
  <!-- Customer Version "KUKA" -->
  <!-- ********************** -->
  <Customer_KUKA>
    <TRACE ENABLE="true" FINISHED_SUB="SKIP" MOVEMENTS="30"
CYCFLAGS="0"/>
    <OFC SET_TO_FALSE="false" TRACE="AT_LEAVING" RESTORE="AT_BWD"/>
    <SCAN ENABLE="true"/>
    <GENERAL BACKWARDSTART="true" IMPLICIT_BCO="false"
QUIT_M_TRACE_SCAN="true"/>
  </Customer_KUKA>
```

4.2 Modified: I/O manager

iosys.ini The file iosys.ini no longer exists in 8.x. The corresponding menu item **I/O Driver** > **Edit I/O Config.** also no longer exists. The former contents of iosys.ini are now managed by means of XML files.

5.x	**iosys.ini** Section `[DRIVERS]` with the drivers to be loaded	**iosys.ini** Sections for the individual drivers with switching of the inputs/outputs
8.0	**Modules.xml** In C:\KRC\ROBOTER\Config\User\Cabinet (Only relevant for KUKA employees: for Office PCs in C:\KRC\ROBOTER\Config\User\Office)	**KRC_IO.xml** In C:\KRC\ROBOTER\Config\User\Common
8.1	**Modules_DevIO.xml**: driver for DevIO **Modules_Interbus.xml**: driver for Interbus **Modules_PNIODriver.xml**: driver for Profinet **Modules_ProConOS.xml**: driver for ProConos All in C:\KRC\ROBOTER\Config\User\Common	**KRC_IO.xml** In C:\KRC\ROBOTER\Config\User\Common

Registry In 8.x, there is no longer a fixed assignment of the I/O driver number to an I/O driver name. The driver names and driver numbers saved in the registry are dispensed with.

System variables

$DATA_INTEGRITY:

For 8.x, $DATA_INTEGRITY has been moved from $Option.dat to $Operate.src. This means that the value of $DATA_INTEGRITY can still be read in the interpreters; configuration is not carried out via $Option.dat, however, but via KRC_IO.xml.

$SET_IO_SIZE:

For 8.x, $SET_IO_SIZE has been moved from $Option.dat to $Operate.src. This means that the value of $SET_IO_SIZE can still be read in the interpreters; configuration is not carried out via $Option.dat, however, but via KRC_IO.xml.

$BUS_STATE and $BUS_USED:

$BUS_STATE and $BUS_USED are dispensed with in 8.x. (The reason for this is the fixed assignment of the I/O driver number from iosys.ini to the driver name in the registry.)

$IOBUS_INFO:

$BUS_STATE and $BUS_USED have been replaced by $IOBUS_INFO.

- Definition:

```
STRUCT IOBUS_INFO_T CHAR NAME[32], BOOL BUS_OK
```

- Declaration:

```
DECL IOBUS_INFO_T $IOBUS_INFO[32]
```

IOCtl

The KRL function IOCtl has been modified for 8.x so that a bus name is now transferred instead of a driver number. The remaining parameters are identical to the previous IOCtl() call.

Reading inputs

The reading of inputs in 8.x is now only dependent on the bus cycle and no longer the interpolation cycle.

5 Loading programs and files

Caution!

If the incompatibilities and restrictions described here are not taken into consideration, the following may occur:

- Error messages
- Robot controller is not operable.
- Personal injury and damage to property.

Programs

KRL programs from 5.x can be loaded in 8.x. If the programs contain elements that are no longer compatible with 8.x, the robot controller generates error messages.

Only valid for VSS:
Programs from VSS 5.x, in which motions have been programmed using inline forms, cannot be loaded in VSS 8.x! The reason for this is that the inline forms have been expanded for 8.x: they can now also be used to select a base. The programs must be created again.

CAL file

CAL files from 5.x may not be used in 8.x, as the unit “Increments” has been changed to “Motor angle”.

The robot must be remastered for all tools.

Archives

In 8.x, only archives from the same version may be restored:

- In KSS 8.0, only archives from KSS 8.0
- In VSS 8.0, only archives from VSS 8.0
- In KSS 8.1, only archives from KSS 8.1
- In VSS 8.1, etc.

6 Programming

6.1 Modified (only VSS): inline forms for motion instructions

VSS 5.x No base can be selected in inline forms for motion instructions.

VSS 8.x A base can be selected in inline forms for motion instructions.

Programs from VSS 5.x, in which motions have been programmed using inline forms, cannot be used in VSS 8.x! The programs must be created again.

6.2 Removed: default data type for undeclared variables

5.x If a non-declared variable is used in a program, it is automatically assigned the default data type POS.

8.x In 8.x, all variables must be declared.

If an undeclared variable is used in a program, the robot controller displays an error message.

6.3 Removed: IMPORT … IS

The KRL instruction IMPORT … IS no longer exists in 8.x. If it is used, this results in a system error.

It was possible to use the instruction in 5.x to import variables from external data lists. This is not necessary, however, as variables can be made globally available using the keyword GLOBAL.

6.4 Restriction: global data types in $CONFIG.DAT

8.x No data types defined using the keyword GLOBAL may be used in $CONFIG.DAT.

Example In DEFDAT PROG(), the enumeration type SWITCH_TYP has been defined with the keyword GLOBAL:

```
DEFDAT PROG()

GLOBAL ENUM SWITCH_TYP ON, OFF
...
```

If this data type is used in $CONFIG.DAT, the compiler signals the error "*Type unknown: *** DECL SWITCH_TYP MY_VAR*".

```
DEFDAT $CONFIG

DECL SWITCH_TYP MY_VAR
...
```

6.5 Modified: transfer parameters in KRL

6.5.1 Specifying the transfer type

5.x In 5.x, it is permissible to transfer parameters to subprograms without specifying the transfer type (IN or OUT) in the subprogram.

Example main program:

```
DEF Main ()
DECL INT my_var
...
UP(my_var)
END
```

Example subprogram:

```
DEF UP (J)
DECL INT J
...
END
```

The subprogram is permissible in 5.x. (The parameter J is interpreted by default as an OUT parameter.)

8.x

In 8.x, it is not permissible to transfer parameters without specifying the transfer type in the subprogram. If the type is not specified, saving the program triggers error message 2091: "*IN or OUT expected*".

The example subprogram illustrated above is only permissible in 8.x in the following forms:

```
DEF UP (J: IN)
DECL INT J
...
END
```

Or:

```
DEF UP (J: OUT)
DECL INT J
...
END
```

6.5.2 Transferring constants and expressions as IN parameters

5.x

In 5.x, constants and all types of expressions can be transferred as OUT parameters. Expressions include comparisons, function calls, arithmetic expressions, logic expressions, etc.

Example main program:

```
DEFDAT Main
DECL INT i
...
UP(13)
UP(i+20)
TRIGGER WHEN DISTANCE=0 DELAY=0 DO UP(FCT())
INTERRUPT DECL 5 WHEN ... DO UP(3)
...
END
```

Example subprogram:

```
DEF UP (par: OUT)
DECL INT par
...
END
```

Example function:

```
DEFFCT INT FCT()
RETURN 23
ENDFCT
```

8.x

In 8.x, constants and all types of expressions must be transferred as IN parameters. If they are transferred as OUT parameters, saving the program triggers error message 2367: "*Expression as OUT parameter not permissible*".

The example subprogram illustrated above is only permissible in 8.x in the following form:

Example, subprogram:

```
DEF UP (par: IN)
DECL INT par
...
END
```

6.5.3 Transferring arrays

5.x

In 5.x, only arrays of type CHAR can be transferred as IN parameters. All other arrays must be transferred as OUT parameters.

If, however, a different array is transferred as an IN parameter, the robot controller does not generate an error message when the program is saved. It is only during program execution that the interpreter generates error message 1536 "*Array parameter inadmissible*", as soon as the corresponding line is reached.

Example main program:

```
DEF Main ()
...
DECL INT my_array[3]
...
UP(my_array[ ])
...
END
```

Example subprogram:

```
DEF UP (par: IN)
DECL INT par[ ]
...
END
```

The example subprogram can be saved without an error message, but it cannot be executed. As soon as the interpreter reaches the line `UP(my_array[ ])`, error message 1536 is generated.

8.x

In 8.x, as in 5.x, only arrays of type CHAR can be transferred as IN parameters. All other arrays must be transferred as OUT parameters.

If, however, a different array is transferred as an IN parameter, the robot controller already displays an error message when the program is saved and does not wait until the program is executed. This is error message 2369: "*Passing arrays as IN parameter is only valid vor arrays of type CHAR*".

6.5.4 Size of IN parameters with trigger and interrupt subprograms

5.x

There is no size limit for IN parameters transferred in trigger or interrupt subprograms.

8.x

There is a size limit for IN parameters transferred in trigger or interrupt subprograms. In this way, the subprograms are called approx. 19% faster.

If the size has been exceeded, error message 2802 is generated during program execution: "*IN parameter too big for trigger-/interrupt- subroutine call*". Larger parameters can only be transferred as OUT parameters.

Guide value: Arrays of type CHAR with a size of 780 or greater should be transferred as OUT parameters. (Example: DECL CHAR c_arr[780])

A maximum of 10 parameters can be transferred per call of a trigger or interrupt subprogram (as in 5.x).

6.6 Modified: CP-PTP approximate positioning with linear external axes

5.x

If the higher motion profile is active, the approximate positioning radius of linear external axes is calculated incorrectly in the case of CP-PTP approximate positioning.

In the calculation, the linear external axes are regarded as rotational axes. This results in an approximate positioning radius that is too great. The approximate positioning radius actually used is derived from the minimum approximate positioning radii of the robot axes and external axes. This means that the incorrectly calculated, excessively large approximate positioning radius of the external axes cannot be used.

8.x

If the higher motion profile is active, the approximate positioning radius of linear external axes is calculated correctly in the case of CP-PTP approximate positioning.

The approximate positioning contour can thus be modified compared with 5.x.

6.7 Modified: trigger response with PRIO=-1

6.7.1 Fundamentals

If `PRIO=-1` is programmed for a trigger, the robot controller automatically issues the priority. Priorities 40 to 80 are available for this. The robot controller issues the lowest free number from this range. The lower the number, the higher the priority (1 = highest priority). Triggers with higher priorities are processed first.

Example:

```
LIN P1
TRIGGER WHEN DISTANCE=1 DELAY=-30 DO SUBPROG_1() PRIO=-1
TRIGGER WHEN DISTANCE=1 DELAY=-40 DO SUBPROG_2() PRIO=-1
TRIGGER WHEN DISTANCE=1 DELAY=-50 DO SUBPROG_3() PRIO=-1
LIN P2
```

For these 3 triggers, the robot controller issues the priorities automatically.

6.7.2 Reciprocal interruption of triggers with subprogram calls

Response in 5.x

Triggers with PRIO=-1 and subprogram calls can reciprocally interrupt one another.

Description

The robot controller assigns the priority when the advance run pointer has reached the line with the trigger. The trigger positioned uppermost in the program thus receives the lowest number, i.e. the highest priority (as long as there is no overflow of the priorities (>>> 6.7.3 "Sequence of the triggers" page 21)).

The trigger in the example could receive the following priorities:

- SUBPROG_1: Priority 40
- SUBPROG_2: Priority 41
- SUBPROG_3: Priority 42

The order in which the triggers are triggered depends on the trigger parameters. It thus does not necessarily correspond to the order of the priorities. When triggers call subprograms, it is thus possible that the subprograms interrupt themselves.

Subprograms from the basic example with the example priorities:

- SUBPROG_3 is called first because of DELAY=-50.
- SUBPROG_2 is called second because of DELAY=-40. The trigger has a higher priority than that of SUBPROG_3 and thus interrupts SUBPROG_3.

  SUBPROG_3 is not executed further until SUBPROG_2 has been completed.
- SUBPROG_1 is called last because of DELAY=-30. The trigger has a higher priority than that of SUBPROG_2 and thus interrupts SUBPROG_2.

  SUBPROG_2 is not executed further until SUBPROG_1 has been completed.

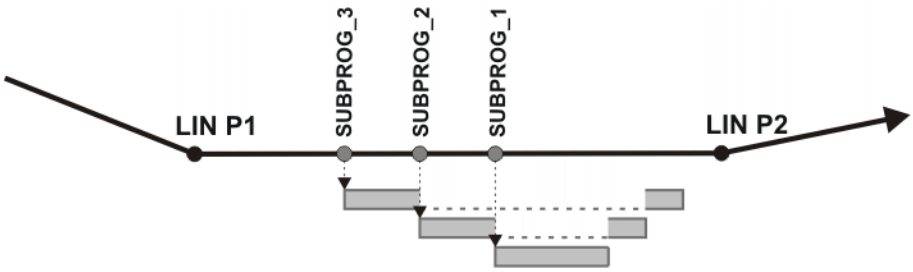


Fig. 6-1: Triggers interrupt one another during execution of the subprograms

Response in 8.x Triggers with PRIO=-1 and subprogram calls cannot reciprocally interrupt one another.

Description The robot controller assigns the priority when the trigger event is triggered. This means that the trigger event that occurs first receives the lowest number that is still free and thus the highest available priority.

Subprograms called by triggers with `PRIO=-1` can thus no longer interrupt themselves.

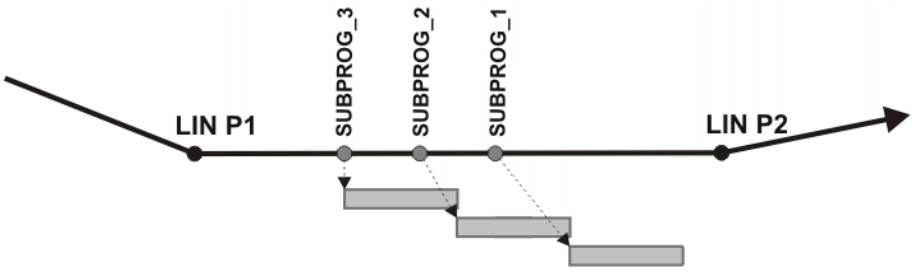


Fig. 6-2: Subprograms are executed completely

6.7.3 Sequence of the triggers

Response in 5.x Triggers with PRIO=-1 are not always executed in the same order. Cause: Overflow of the priority numbers.

Description The trigger in the basic example could receive the following priorities:

- SUBPROG_1: Priority 40
- SUBPROG_2: Priority 41

- SUBPROG_3: Priority 42

If the robot controller has assigned priority 80, it next assigns priority 40 again. The trigger in the basic example could thus also receive the following priorities:

- SUBPROG_1() = priority 79
- SUBPROG_2() = priority 80
- SUBPROG_3() = priority 40

The hierarchy of the priorities is different in this case. This means that the subprograms are executed in a different order.

The user can only tell with great difficulty when the priority numbers overflow and when the response changes accordingly.

Subprograms from the basic example with the changed priorities:

- The subprogram SUBPROG_3 is called first because of DELAY=-50. Since the trigger has the highest priority, it is terminated before SUBPROG_2 can be called.
- SUBPROG_2 is called second because of DELAY=-40.
- SUBPROG_1 is called last because of DELAY=-30. The trigger has a higher priority than that of SUBPROG_2 and thus interrupts SUBPROG_2.

  SUBPROG_2 is not executed further until SUBPROG_1 has been completed.

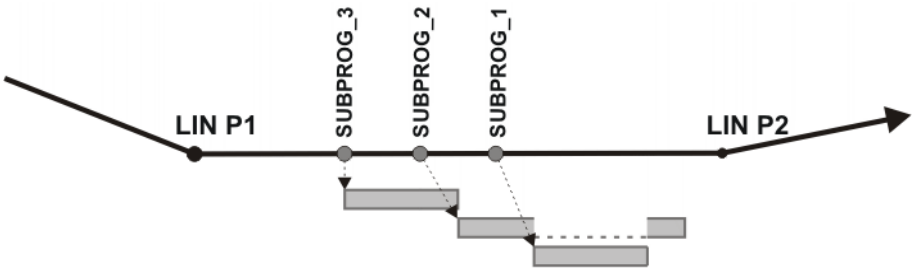


Fig. 6-3: Priorities changed

Response in 8.x Triggers with PRIO=-1 are always executed in the same order.

Description If the robot controller has assigned priority 80, it cannot assign priority 40 again until no trigger subprograms are being executed.

If a trigger subprogram with priority 80 is being executed and another trigger with PRIO=-1 is to be triggered, error message 1432 appears: *Max. no. of interrupts defined*. (If a trigger calls a subprogram, it counts as an active interrupt for as long as the subprogram is being executed.)

Remedy: Optimize the KRL program.

6.8 Modified: runtime variables in triggers

5.x In 5.x, runtime variables cannot be used in triggers.

8.x In 8.x, runtime variables can be used in triggers if they are on the right-hand side of the assignment.

It is still not possible to use runtime variables on the left-hand side of an assignment. If a runtime variable is used there, this results in error message 1316: *Runtime values for Trigger in subroutines inadmissible*. This is also the case if the runtime variable has been transferred to the subprogram as an IN or OUT parameter (even if this has occurred over several call levels).

Example

```
DEF TEST ()
  DECL BOOL my_var
  INI
  my_var=TRUE
  PTP HOME
  TRIGGER WHEN DISTANCE=0 DELAY=0 DO $OUT[2]= my_var
  PTP HOME
END
```

Response in 5.x:

The example results in error message 1316: *Runtime values for Trigger in subroutines inadmissible*.

Response in 8.x:

- No error message.
- The robot controller converts the runtime variable to a constant. At the moment the trigger is triggered, the constant of the variable is assigned on the left-hand side.

IN and OUT parameters are also converted to constants if the underlying variable is a runtime variable.

7 System variables

7.1 Modified: $PRO_IP

5.x

In 5.x, access to the overall structure of $PRO_IP gave different results from access to a component.

In the following example, there were different results for mPArrived1 and mPArrived2. The example illustrates a Submit program:

```
1  Decl PRO_IO mProIP
2  Decl INT mPArrived1, mPArrived2

3  mProIP=$Pro_IP
4  mPArrived1=mProIP.P_Arrived
5  mPArrived2=$Pro_IP.P_Arrived
```

Line	Description
4	Reads P_Arrived from the Submit interpreter (always 0)
5	Reads P_Arrived from the robot interpreter

8.x

In 8.x, access to the overall structure of $PRO_IP and access to a component have the same result.

Explicit access to the interpreter status:

The following new variables exist in 8.x:

- $PRO_IP0

  $PRO_IP0 can be used to access the status of the Submit interpreter.
- $PRO_IP1

  $PRO_IP1 can be used to access the status of the robot interpreter.

$PRO_IP0 and $PRO_IP1 can be read by a program. They can also be written to using the variable correction function.

Implicit access to the interpreter status:

Depending on the specific interpreter, access for $PRO_IP is as follows:

- Reading the variable in a robot program refers to the status of the robot interpreter.
- Reading the variable in a Submit program refers to the status of the Submit interpreter.
- Reading/writing to the variable by means of the variable correction function refers to the current value of $INTERPRETER.

  $INTERPRETER = 0: the Submit interpreter is selected.

  $INTERPRETER = 1: the robot interpreter is selected.

Access to P_Arrived in Submit program:

$PRO_IP contains the following initialized components in the Submit interpreter:

- $PRO_IP.SNR
- $PRO_IP.Name[]
- $PRO_IP.I_Executed

This means: P_Arrived is not initialized in the Submit interpreter. Reading P_Arrived in a Submit program triggers error message 1422 "*… value invalid*".

In order to be able to read the robot interpreter component P_Arrived in a Submit program, the instruction must be modified as follows:

`IF ($PRO_IP1.P_Arrived == 1) THEN …`

(In 5.x, this was: `IF ($PRO_IP.P_Arrived == 1) THEN ...`)

7.2 Removed: $BIOS_VERSION

The system variable $BIOS_VERSION no longer exists in 8.x. If it is used, this results in a system error.

$BIOS_VERSION was used in 5.x to indicate the BIOS version on the user interface. The BIOS version is still indicated there in 8.x, but no longer read using $BIOS_VERSION.

7.3 Removed: MSG_T

The structure type MSG_T and the system variable $MSG_T no longer exist in 8.x. If they are used, this results in a system error.

MSG_T and $MSG_T were used in 5.x for programming messages. A different syntax, in which MSG_T and $MSG_T are no longer used, is available for user-defined messages in 8.x.

7.4 Removed: $DIGIN and associated elements

The following elements no longer exist in 8.x. If they are used, this results in a system error.

Name	Description
$DIGIN1 … $DIGIN6	System variable
$DIGIN1CODE … $DIGIN6CODE	System variable
$STROBE1 … $STROBE6	System variable
$STROBE1LEV … $STROBE6LEV	System variable
$DIGIN_FILT	System variable
DIGIN ON	KRL instruction
DIGIN OFF	KRL instruction
DIGINCODE	Enumeration type

7.5 Removed: system variables for path planning

The following system variables from 5.x no longer exist in 8.x. The system response in 8.x corresponds to the default setting in 5.x, with the exception of $ASYNC_OPT and $TECH_CONT.

System variable	Response in 8.x
$ANA_DEL_FLT	The analog output filter is activated. Corresponds in 5.x to: $ANA_DEL_FLT = ON
$ASYNC_OPT	Asynchronous axes are possible. Corresponds in 5.x to: $ASYNC_OPT = TRUE If $VEL_FLT_OFF = TRUE, then $ANA_DEL_FLT has no effect.
$CPVELREDMELD	Corresponds in 5.x to: $CPVELREDMELD = TRUE
$DRIVE_CART	PTP points can have Cartesian coordinates Corresponds in 5.x to: $DRIVE_CART = TRUE

System variable	Response in 8.x
$DRIVE_CP	Cartesian robot motion is possible. Corresponds in 5.x to: $DRIVE_CP = TRUE
$ENDLESS	Infinitely rotating axes are possible. Corresponds in 5.x to: $ENDLESS = TRUE
$EXT_AXIS	Corresponds in 5.x to: $EXT_AXIS = TRUE
$IMPROVEDCP-BLENDING	Corresponds in 5.x to: $IMPROVEDCPBLEND-ING = TRUE
$IMPROVEDMIXED-BLENDING	Corresponds in 5.x to: $IMPROVEDMIXED-BLENDING = TRUE
$TECH_CONT	Corresponds in 5.x to: $TECH_CONT = TRUE
$VEL_FLT_OFF	The velocity is always calculated from filtered setpoint values. The velocity values need no longer be filtered. Corresponds in 5.x to: $VEL_FLT_OFF = TRUE

7.6 Modified: machine data

Description

Numerous machine data for the drive technology which were system variables in 5.x no longer exist in 8.x. They have either been moved to XML files, for example, or dispensed with entirely.

- System variables that have been moved to XML files can no longer be read from KRL.

  Reading/writing via GET_SYSTEM_DATA() or SET_SYSTEM_DATA() will only be possible for controller parameters ($I_LG_PTP, etc.).
- Some of the units have changed since 5.x.
- System variables in 5.x that refer to axes and are arrays with one array element per axis are replaced in 8.x by axis-specific files, e.g. CtrlA<*Drive no.*>.xml.

  5.x: a system variable consists of 1 machine datum that refers to the different axes.

  8.x: an XML file contains several machine data that refer to 1 axis.
- If there are external axes in the system, additional axis-specific files exist for them, e.g. CtrlE<*Drive no.*>.xml.
- Entries with a slash in the column **Name 8.x** indicate the position within the XML file.

  Example: ToolMotor/Monitoring/PositionLag = section **ToolMotor**, subsection **Monitoring**, Parameter **PositionLag**

```
- <Axis
    xsi:noNamespaceSchemaLocation="/Roboter/Config/System/Common/Schemes/NGAxis.xsd"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  - <Version Label="">
      <Info Version="KUKA V8.0" />
      <Schema Version="2" />
    </Version>
    <Machine Name="#KR16 NG FLR ZH16" />
    <AxisData Type="Rotator" Mode="StandAlone" Simulation="Off" SingleMotorCoupling="Off"
      MessageDisable="0" />
  - <ToolMotor EmergencyStopTorque="20" ExtTorqueOfInertia="0.000408"
      MaximalEnergy="988" RampUpTimeUnderLoad="400" RampStopTime="211"
      MotorFile="Motor/00-115-925_H.xml" ServoFile="Servofile/Servo_32_00-115-
      925_H_V8.xml" ParameterFile="Mada/NGAxis/CtrlA1.xml">
      <Monitoring PositionLag="250" PositionLagSoft="1000" Speed="3300" SpeedLag="0" />
      <Limit Speed="3250" Current="0" />
    </ToolMotor>
    <Inversion Position="false" Velocity="false" Torque="false" />
    <Delay ActValue="2" CmdValue="42" />
  </Axis>
```

Fig. 7-1: Example: ToolMotor/Monitoring/PositionLag

Overview

The overview displays important machine data (drive technology) from 5.x and their status in 8.x.

Warning!
The machine data may only be edited by KUKA personnel.

- Machine data for which no file is specified in the column **Name 5.x** are located in the file $machine.dat.
- Machine data for which “Open” has been entered in the column **Status 8.x** are still being worked on.

Name 5.x	Status 8.x	File 8.x	Name 8.x (unit)
$ASR_ERROR	N/A	-	-
$AX_ENERGY_MAX	Moved	NGAxis\ A<*Drive no.*>.xml	ToolMotor/MaximalEnergy (J)
$AX_SIM_ON	Moved	NGAxis\ A<*Drive no.*>.xml	Simulation
$AXIS_RESO	N/A	-	-
$BOUNCE_TIME	N/A	-	-
$BRK_COOL_OFF_COEFF	Moved	Motor\<*Motor name*>.xml	BrakeData/CoolOffCoeff
$BRK_ENERGY_MAX	Moved	Motor\<*Motor name*>.xml	BrakeData/MaximalEnergy (J)
$BRK_OPENTM	Moved	Motor\<*Motor name*>.xml	BrakeData/OpenTime (ms)
$BRK_TORQUE	Moved	Motor\<*Motor name*>.xml	BrakeData/Torque (Nm)
$CABLE2_MON	Open	-	-
$CURR_CAL	N/A	-	-
$CURR_LIM	Moved	NGAxis\ A<*Drive no.*>.xml and Motor\<*Motor name*>.xml	ToolMotor/Limit/Current or MotorData/MaximalCurrent
$CURR_MAX	N/A	-	-
$CURR_MON	N/A	-	-
$DECEL_MB	Moved	NGAxis\ CtrlA<*Drive no.*>.xml	ToolMotor/RampStopTime (ms)
$DSECHANNEL	Moved	CFCore.xml	Defined via the overall configuration of the drive technology.
$EMSTOP_MOTTORQ $robcor.dat	Moved	NGAxis\ A<*Drive no.*>.xml	ToolMotor/EmergencyStopTorque (Nm)
$EXT_ACCU_MON $steu/option.dat	N/A	-	-
$FFC_TORQ	N/A	-	-
$FFC_TORQ_AXIS	Moved	NGAxis\ CtrlA<*Drive no.*>.xml	GlobalParameter/FFCTorque
$FFC_VEL	Moved	NGAxis\ CtrlA<*Drive no.*>.xml	GlobalParameter/ FFCVelocity
$FOL_ERR_MA	Moved	NGAxis\ CtrlA<*Drive no.*>.xml	ToolMotor/Monitoring/ PositionLag (°)
$G_COE_CUR	N/A	-	-
$G_VEL_CAL	N/A	-	-
$G_VEL_CP	Moved	NGAxis\ CtrlA<*Drive no.*>.xml	VelGain.CP (Nms/°)

Name 5.x	Status 8.x	File 8.x	Name 8.x (unit)
$G_VEL_PTP	Moved	NGAxis\ CtrlA<*Drive no.*>.xml	VelGain.PTP (Nms/°)
$I_LG_CP	Moved	NGAxis\ CtrlA<*Drive no.*>.xml	PosIntTime.CP (s)
$I_LG_PTP	Moved	NGAxis\ CtrlA<*Drive no.*>.xml	PosIntTime.PTP (s)
$I_VEL_CP	Moved	NGAxis\ CtrlA<*Drive no.*>.xml	VelIntTime.CP (s)
$I_VEL_PTP	Moved	NGAxis\ CtrlA<*Drive no.*>.xml	VelIntTime.PTP (s)
$IN_POS_MA	Remains	-	-
$KPS_CURR_MAX	Moved	Supply\<*Supply type*>.xml	Current/MaxCurrent1s (A)
$KPS_CURR_RATED	Moved	Supply\<*Supply type*>.xml	Current/MaxCurrent60s (A)
$KT_MOT	Moved	Motor\<*Motor name*>.xml	MotorData/KT-Factor (Nm/A)
$KT0_MOT	Moved	Motor\<*Motor name*>.xml	MotorData/KT0-Factor (Nm/A)
$LG_CP	Moved	NGAxis\ CtrlA<*Drive no.*>.xml	PosGain.CP (1/s)
$LG_PTP	Moved	NGAxis\ CtrlA<*Drive no.*>.xml	PosGain.PTP (1/s)
$LOOP_DIRECTION	Moved	NGAxis\ A<*Drive no.*>.xml	Inversion/Position or Velocity
$LOOP_G_VEL_PTP	N/A	-	-
$LOOP_I_LG_PTP	N/A	-	-
$LOOP_I_VEL_PTP	N/A	-	-
$LOOP_LG_PTP	N/A	-	-
$LOOP_TYPE	Moved	NGAxis\ A<*Drive no.*>.xml	AxisData/Mode
$LOOP_RAT_EXTPOS_AX	N/A	-	-
$LOOP_RAT_MOT_AX	N/A	-	-
$MOTOR_POLE_NUMBER	Moved	Motor\<*Motor name*>.xml	MotorData/PolePairNumber
$MS_DA	Moved	NGAxis\ A<*Drive no.*>.xml	AxisData/ MessageDisable (bit array)
$NINPUT_SENSORCHANNEL	Moved	CFCore.xml	Defined via the overall configuration of the drive technology.
$NINPUT_SENSORTYPE	Moved	CFCoreWaggonDriver-Config.xml	Defined via the overall configuration of the drive technology.
$NINPUT_SUBCHANNEL	N/A	-	-
$PHASE_MONITORING $steu/option.dat	N/A	-	-
$PMCHANNEL	Moved	NextGenDriveTech.xml	AxisConfig/Axis/Supply-Channel
$POSINPUT_SENSORCHANNEL	Moved	CFCore.xml	Defined via the overall configuration of the drive technology.

Name 5.x	Status 8.x	File 8.x	Name 8.x (unit)
$POSINPUT_SENSORTYPE	Moved	CFCoreWaggonDriver-Config.xml	Defined via the overall configuration of the drive technology.
$POSINPUT_SUBCHANNEL	N/A	-	-
$RAISE_T_MOT	Moved	Motor\<*Motor name*>.xml	RampUpTime (ms)
$RAISE_TIME	Moved	NGAxis\ A<*Drive no.*>.xml	TooMotor/RampUpTime-UnderLoad (ms)
$RAT_EXT_ENC	N/A	-	-
$RAT_MOT_ENC	Moved	Motor\<*Motor name*>.xml	EncoderData/RatioMo-torEncoder
$RDC2_PHASE_SHIFT_1	N/A	-	-
$RDC2_PHASE_SHIFT_2	N/A	-	-
$SEN_DEL	N/A	-	-
$SERV_OFF_TM	Moved	Motor\<*Motor name*>.xml	BrakeData/CloseTime (ms)
$SERVOFILE	Moved	NGAxis\ A<*Drive no.*>.xml	ToolMotor/Servofile
$SERVOFILE_CONFIG	N/A	-	-
$SERVOFILEKPS	Moved	Supply\<*Supply type*>.xml	Config/ServoFile
$SERVOMODE	N/A	-	-
$SLAVE_LOOP_PMCHANNEL	N/A	-	-
$VEL_AXIS_MA	Remains	-	-

8 Messages

8.1 Modified: $STOPNOAPROX

5.x

- In modes T1 and T2, $STOPNOAPROX determines which message is generated in the case of “Approximation not possible”:

  1123 (notification message) or 1128 (acknowledgement message that triggers a stop)

  (1155 and 1442 are always notification messages.)
- In AUT and AUT EXT modes, $STOPNOAPROX can suppress messages 1123, 1155 and 1442.

$STOPNOAPROX ==	Message type
FALSE	Operating mode T1 or T2: notification message
	Operating mode AUT or AUT EXT: no message
TRUE	Operating mode T1 or T2: acknowledgement message that triggers a stop
	Operating mode AUT or AUT EXT: notification message

8.x

- In modes T1 and T2, $STOPNOAPROX determines the message type for messages 1123, 1442 and 2920:
  - Either notification message that does not trigger a stop
  - Or acknowledgement message that triggers a stop
- $STOPNOAPROX cannot be used to suppress messages. (If messages are to be suppressed in AUT and AUT EXT, this must be configured in the file C:\KRC\USER\SmartHMI.User.config.)

  (>>> 8.3 "Modified: suppressing messages" page 33)

$STOPNOAPROX ==	Message type
FALSE	All operating modes: notification message
TRUE	Operating mode T1 or T2: acknowledgment message
	Operating mode AUT or AUT EXT: notification message

8.2 Modified: message generation in the case of “Approximate positioning not possible”

T1/T2

In 8.x, in modes T1 and T2, $STOPNOAPROX determines the message type for messages 1123, 1442 and 2920.

(>>> 8.1 "Modified: $STOPNOAPROX" page 31)

AUT / AUT EXT

In 8.x, $STOPNOAPROX has no effect in AUT and AUT EXT modes.

In 8.x, the following notification messages are suppressed by default in AUT and AUT EXT modes:

No.	Message
1123	*Approximate positioning not possible (module* <module name>*, line* <line number>*)*
1155	*Approximate positioning not possible, torque too high*

No.	Message
1442	*Instruction prevents approximate positioning (module* <module name>*, line* <motion line number>*)*
2920	*Approximate positioning in interrupt program not possible (module* <module name>*, line* <line number>*)*

(>>> 8.3 "Modified: suppressing messages" page 33)

Modified/new messages

No.	Message / description
1123	*Approximate positioning not possible (module* <module name>*, line* <line number>*)* The message has been extended to include the module name and line number. If the reason why approximate positioning is not possible is known, it is referred to at the end of the text of 1123 by means of *(=> 1155)* or *(=> 1442)*. (This is also the case if the display of message 1155 or 1442 has been suppressed.)
1128	*Stop, approximate positioning not possible* This message no longer exists. The function has been replaced by message 1123.
1155	*Approximate positioning not possible, torque too high* If, additionally, approximate positioning was not possible due to reasons of time, a "+" is displayed at the end of the text of 1155.
1442	*Instruction prevents approximate positioning (module* <module name>*, line* <motion line number>*)* The block pointer indicates the instruction that cannot be approximated. If, additionally, approximate positioning was not possible due to reasons of time, a "+" is displayed at the end of the text of 1442.
2920	*Approximate positioning in interrupt program not possible (module* <module name>*, line* <line number>*)* The message is new in 8.x. It appears if an approximate positioning block has been programmed in an interrupt program. This also applies to programs called from an interrupt program.

- The KRL program must be displayed in detail view with open folds so that the line number in the message corresponds to the line number in the program.
- Whenever a message refers to a KRL program with the attribute **Hidden**, the module name and line number refer to the hidden program. The block pointer indicates the point in the non-hidden program from which the hidden program was called.

Example 1

The example is valid for operating mode T1 or T2 and $STOPNOAPROX == TRUE.

```
   ...
7  LIN P4 C_DIS
8  WAIT FOR $IN[27]= TRUE
9  LIN P5
   ...
```

Line	Description
7	The block pointer stops because approximate positioning is not possible from point P4. Acknowledgement message 1123 *Approximate positioning not possible (module XY, line 7) (=> M1442)* is displayed. The block pointer remains stopped until the message is acknowledged.
8	The block pointer stops because WAIT FOR stops the advance run and so approximate positioning cannot be carried out. Acknowledgement message 1442 *Instruction prevents approximate positioning (module XY, line 8)* is displayed. The block pointer remains stopped until the message is acknowledged.

It is stopped twice to show the user that the problem can be resolved in two ways:

- Either convert P4 to an exact positioning point.
- Or remove the WAIT FOR instruction that stops the advance run.

Example 2

Examples of messages with module, line number, etc., specified:

No.	Example
1123	*Approximate positioning not possible (module X, line 5)*
1123	*Approximate positioning not possible (module Y, line 25) (=> 1155)*
1123	*Approximate positioning not possible (module Z, line 15) (=> 1442)*
1155	*Approximate positioning not possible, torque too high*
1155	*Approximate positioning not possible, torque too high +*
1442	*Instruction prevents approximate positioning (module Z, line 16)*
1442	*Instruction prevents approximate positioning (module Z, line 16) +*
2920	*Approximate positioning in interrupt program not possible (module N, line 59)*

8.3 Modified: suppressing messages

Description

5.x: To suppress notification messages in AUT and AUT EXT modes, $STOPNOAPROX == FALSE was set.

8.x: If messages are to be suppressed in AUT and AUT EXT, this must be configured in the file C:\KRC\USER\SmartHMI.User.config.

($STOPNOAPROX no longer suppresses messages, but merely determines the message type.)

(>>> 8.1 "Modified: $STOPNOAPROX" page 31)

Precondition

- Windows interface

Procedure

1. Open the file SmartHMI.User.config in the directory C:\KRC\USER.
2. Make the desired changes.
3. Save and close the file.
4. Reboot SmartHMI.

SmartHMI.User.config

The messages are configured by means of so-called MessageSets.

Structure of a MessageSet:

```
<MessageSet Name="..." MergeOrder="...">
         <Rule Type ="..." Source="..." Module="..." Number="..." />
         <IgnoreInAutomatic />
      </MessageSet>
```

Excerpt from SmartHMI.User.config with the predefined MessageSet:

```
<!-- Rule: Ignore some specific kernel system messages in
"Automatic" and "Automatic Extern"-->
      <MessageSet Name="IgnoreSomeKernelMessages" MergeOrder="100">
         <Rule Module="CrossMeld"
Number="1123|3078|3087|470|1053|1155|1442|2920|3131|3132" />
         <IgnoreInAutomatic />
      </MessageSet>
```

The predefined MessageSet causes the following notification messages to be suppressed by default in AUT and AUT EXT modes:

No.	Message
470	Safe robot override reduction active
1053	CP-Vel. reduction point %1 %2 by %3
1123	Approximate positioning not possible
1155	Approximate positioning not possible, torque too high
1442	Instruction prevents approximate positioning (module %1, line %2)
2920	Approximate positioning in interrupt program not possible
3078	String Velocity 0 m/sec in Spline Point %1, Line %2, Reason %3
3087	Trigger: %1-value %2 changed to: %3
3131	KRL System Diagnostics not started yet.
3132	KRL System Diagnostics hasn't been stopped yet.

<MessageSet />

A MessageSet defines a set of messages. The number of MessageSets can be increased as required.

Attribute	Description
`Name`	Type: String Name of the MessageSet. Used internally for outputs to the SmartHMI.log. If a name is issued, it must be unique. Optional
`MergeOrder`	Type: INT ≥ 100 One message can belong to several MessageSets. For this reason, the order in which the MessageSets are evaluated is relevant. The MessageSets are sorted accordingly `MergeOrder` before they are executed. (The range 0 … 99 is reserved for internal system purposes.)

<Rule />

`<Rule />` defines which messages belong to a MessageSet. One message can belong to several MessageSets.

- A MessageSet without the element `<Rule />` defines the set of all messages.
- All attributes permissible for `<Rule />` are optional. If `<Rule />` is used, however, it must contain at least one attribute.

Attribute	Description
Type	Type: String Message type: Info, State, Ackn., Wait or Dialog
Source	Type: String Language database key of the originator of the message
Module	Type: String Module name of the originator of the message
Number	Type: String Message number

The attributes of type String are regular expressions and are used in accordance with the syntax for regular Microsoft expressions. The description of this syntax is not part of the scope of this documentation.

Examples

Suppress all user-specific messages in AUT and AUT EXT:

```
<MessageSet Name="IgnoreAllUserMessages" MergeOrder="101">
     <Rule Source="&lt;.*&gt;" />
     <IgnoreInAutomatic />
</MessageSet>
```

Suppress all user-specific notification messages in AUT and AUT EXT:

```
<MessageSet Name="IgnoreAllInfoUserMessages" MergeOrder="101">
      <Rule Type="Info" Source="&lt;.*&gt;" />
      <IgnoreInAutomatic />
    </MessageSet>
```

The originator (=Source) of a user-specific message is generally a freely entered text in angle brackets. The regular expression for this is “<.*>”.

Characters in XML	Meaning
<	“<” character
.	Any character
*	Any number
&gt	“>” character

In XML, certain characters must be masked for reasons of well-formedness. If the file SmartHMI.User.config is edited in a text editor, the characters “<” and “>” must be replaced in the attribute values.

9 KUKA Service

9.1 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.

Faults leading to production downtime should be reported to the local KUKA subsidiary within one hour of their occurrence.

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)
- Version of the KUKA System Software
- Optional software or modifications
- Archive of the software
- Application used
- Any external axes used
- Description of the problem, duration and frequency of the fault

9.2 KUKA Customer Support

Availability

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

Argentina

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

Australia

Marand Precision Engineering Pty. Ltd. (Agency)
153 Keys Road
Moorabbin
Victoria 31 89
Australia
Tel. +61 3 8552-0600
Fax +61 3 8552-0605
robotics@marand.com.au

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

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.
Avenida Franz Liszt, 80
Parque Novo Mundo
Jd. Guançã
CEP 02151 900 São Paulo
SP Brazil
Tel. +55 11 69844900
Fax +55 11 62017883
info@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 Flexible Manufacturing Equipment (Shanghai) Co., Ltd.
Shanghai Qingpu Industrial Zone
No. 502 Tianying Rd.
201712 Shanghai
P.R. China
Tel. +86 21 5922-8652
Fax +86 21 5922-8538
Franz.Poeckl@kuka-sha.com.cn
www.kuka.cn

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

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

Hungary

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

India

KUKA Robotics, Private Limited
621 Galleria Towers
DLF Phase IV
122 002 Gurgaon
Haryana
India
Tel. +91 124 4148574
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.
Daiba Garden City Building 1F
2-3-5 Daiba, Minato-ku
Tokio
135-0091
Japan
Tel. +81 3 6380-7311
Fax +81 3 6380-7312
info@kuka.co.jp

Korea

KUKA Robot Automation Korea Co. Ltd.
4 Ba 806 Sihwa Ind. Complex
Sung-Gok Dong, Ansan City
Kyunggi Do
425-110
Korea
Tel. +82 31 496-9937 or -9938
Fax +82 31 496-9939
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 Mexico S. de R.L. de C.V.
Rio San Joaquin #339, Local 5
Colonia Pensil Sur
C.P. 11490 Mexico D.F.
Mexico
Tel. +52 55 5203-8407
Fax +52 55 5203-8148
info@kuka.com.mx

Norway

KUKA Sveiseanlegg + Roboter
Bryggeveien 9
2821 Gjövik
Norway
Tel. +47 61 133422
Fax +47 61 186200
geir.ulsrud@kuka.no

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

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

Spain

KUKA Sistemas de Automatización S.A.
Pol. Industrial
Torrent de la Pastera
Carrer del Bages s/n
08800 Vilanova i la Geltrú (Barcelona)
Spain
Tel. +34 93 814-2353
Fax +34 93 814-2950
Comercial@kuka-e.com
www.kuka-e.com

Sweden

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

Switzerland

KUKA Roboter Schweiz AG
Riedstr. 7
8953 Dietikon
Switzerland
Tel. +41 44 74490-90
Fax +41 44 74490-91
info@kuka-roboter.ch
www.kuka-roboter.ch

Taiwan

KUKA Robot Automation Taiwan Co. Ltd.
136, Section 2, Huanjung E. Road
Jungli City, Taoyuan
Taiwan 320
Tel. +886 3 4371902
Fax +886 3 2830023
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

UK

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

USA

KUKA Robotics Corp.
22500 Key Drive
Clinton Township
48036 Michigan
USA
Tel. +1 866 8735852
Fax +1 586 5692087
info@kukarobotics.com
www.kukarobotics.com

Index