

SOFTWARE

KR C2 / KR C3

Configuration

KUKA System Software (KSS)

Release 5.2

Issued: 05 Aug 2005 Version: 02

© Copyright **KUKA Roboter GmbH**

This documentation or excerpts therefrom may not be reproduced or disclosed to third parties without the express permission of the publishers. Other functions not described in this documentation may be operable in the controller. The user has no claim 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 subsequent editions.

Subject to technical alterations without an effect on the function.

PD Interleaf

Contents

1	Configuring the system	7
1.1	The “Configure” menu	9
1.1.1	I/O	9
1.1.1.1	Automatic External	9
1.1.2	Drivers	9
1.1.2.1	Edit I/O Config.	10
1.1.2.2	I/O Driver Reset	10
1.1.2.3	Reconfigure I/O Driver	12
1.1.2.4	I/O State	13
1.1.2.5	Serial Communication	13
1.1.3	Submit interpreter	14
1.1.4	Status keys	14
1.1.5	Jogging (Override)	15
1.1.5.1	Program override steps (POV)	15
1.1.5.2	Jog override steps (HOV)	15
1.1.5.3	Mouse position	15
1.1.5.4	Mouse configuration	15
1.1.6	User group	15
1.1.7	Cur. tool/base	17
1.1.8	Tool definition	18
1.1.9	On/Off Options	19
1.1.9.1	Force cold startup	19
1.1.9.2	Disable PowerOff Delay	19
1.1.10	Miscellaneous	20
1.1.10.1	Language	20
1.1.10.2	Change password	21
1.1.10.3	Editor	22
1.1.10.4	Office GUI	24
1.1.10.5	Monitoring working envelope	25
1.1.10.6	Reinitialization	26
1.1.10.7	Cycle Time Optimizer	26
1.1.10.8	Event planner	27
1.2	The “File” menu	29
1.2.1	Archive	29
1.2.2	Restore	29
2	Configuring the system, Expert	31
2.1	Variable overview	31
2.1.1	Display	31
2.1.2	Configure	31
2.1.3	Edit ConfigMon.ini	34
2.2	Long text	34
2.3	Monitoring	37
2.3.1	Control cabinet external fan	37
2.3.2	PC fan	37
2.3.3	Configurable output for hardware warnings	37
2.3.4	Motor cable monitoring	37
2.4	Simulated inputs/outputs (I/O simulation)	38

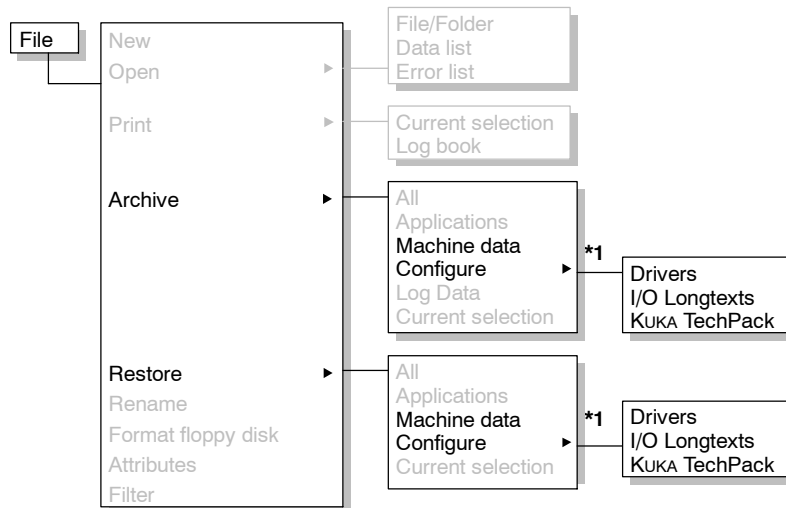
2.4.1	Function	38
2.4.2	Options	40
2.4.3	Variables used	40
2.5	5 home positions	41
2.5.1	"\R1\MaDa\$\smachine.dat" file	41
2.5.2	"\Steu\MaDa\$\smachine.dat" file	41
2.6	Workspace monitoring	42
2.6.1	Cartesian workspace monitoring	42
2.6.1.1	Configure	42
2.6.1.2	Override	45
2.6.1.3	Examples	46
2.6.2	Axis-specific workspace monitoring	51
2.6.2.1	Functional principle	51
2.6.2.2	Configure	53
2.6.2.3	Override	56
2.6.2.4	Example	57
2.7	Torque mode (Soft Servo)	64
2.7.1	General	64
2.7.1.1	Limitations and risks	64
2.7.1.2	Example of torque mode application	65
2.7.2	Functional principle	65
2.7.3	Examples for activation of soft axes	66
2.7.3.1	Axis 1 soft	66
2.7.3.2	Axis 3 soft	66
2.7.4	Example of axis with defined torque	67
2.7.5	Variables for torque mode	67
2.8	Collision monitoring	69
2.8.1	Function	69
2.8.2	Configuration	69
3	Automatic External	73
3.1	General	73
3.2	Configuring the interface	73
3.2.1	Inputs	73
3.2.1.1	Outputs	74
3.3	Automatic system start	75
3.4	Technology-specific organization program CELL.SRC	76
3.5	The P00 (AUTOMATIC EXTERNAL) module	77
3.5.1	The EXT_PGNO function	77
3.5.1.1	Request of a program number from the host computer	77
3.5.1.2	Communication of receipt of a valid program number	77
3.5.1.3	Error handling	78
3.5.2	The EXT_ERR function	78
3.6	Signal descriptions	79
3.6.1	Inputs	79
3.6.1.1	PGNO_TYPE	79
3.6.1.2	PGNO_LENGTH	80
3.6.1.3	PGNO_FBIT	80
3.6.1.4	REFLECT_PROG_NR	80
3.6.1.5	PGNO_PARITY	80

3.6.1.6	PGNO_VALID	80
3.6.1.7	EXT_START	81
3.6.1.8	MOVE_ENABLE	81
3.6.1.9	CHCK_MOVENA	81
3.6.1.10	CONF_MESS	82
3.6.1.11	DRIVES_ON	82
3.6.1.12	DRIVES_OFF	82
3.6.2	Outputs	83
3.6.2.1	STOPMESS	83
3.6.2.2	PGNO_REQ	83
3.6.2.3	PGNO_FBIT_REFL	83
3.6.2.4	APPL_RUN	83
3.6.2.5	PERI_RDY	83
3.6.2.6	ALARM_STOP	83
3.6.2.7	USER_SAF	84
3.6.2.8	T1, T2, AUT, EXTERN	84
3.6.2.9	ON_PATH	84
3.6.2.10	NEAR_POSRET	84
3.6.2.11	PRO_ACT	85
3.6.2.12	IN_HOME	85
3.6.2.13	ERR_TO_PLC	85
3.6.3	Other variables	85
3.6.3.1	PGNO	85
3.6.3.2	PGNO_ERROR	85
3.7	Signal diagrams	86
3.7.1	Automatic system start and normal operation with program number acknowledgement by means of PGNO_VALID	86
3.7.2	Automatic system start and normal operation with program number acknowledgement by means of \$EXT_START	87
3.7.3	Restart after dynamic braking (operator safety and restart)	88
3.7.4	Restart after path-maintaining EMERGENCY STOP	89
3.7.5	Restart after motion enable	90
3.7.6	Restart after user STOP	91
3.8	Other	92
3.8.1	Restart after passive stop	92
3.8.2	Step by step program execution	92
3.8.3	Velocity for returning to the programmed path	92
3.9	Configuration example	93
3.9.1	Declarations	93
3.10	Messages	95

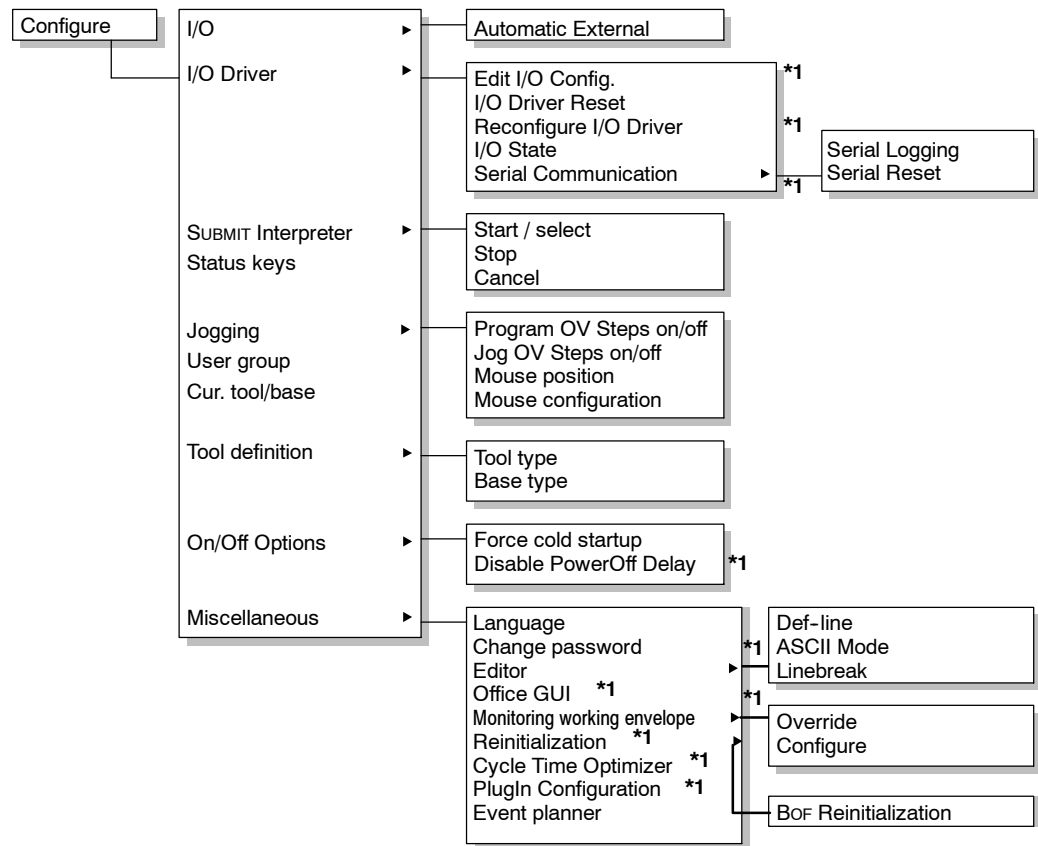


1 Configuring the system

This chapter deals with configuration of the system. The menus “File” and “Configure” are available for this purpose.



*1: Not available in the user group “User”.



*1: Not available in the user group “User”.

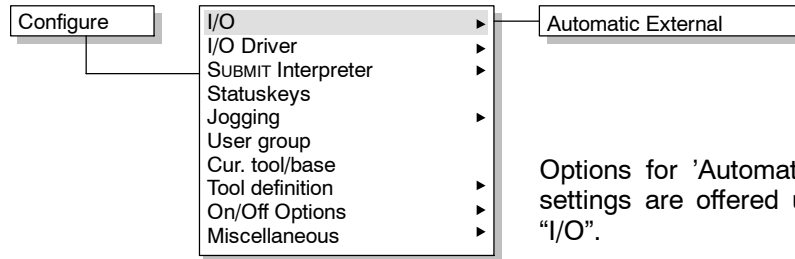


“Configure” menu	
I/O	Settings for grippers, the Automatic External interface and long texts
I/O Driver	Configures and resets the peripheral interfaces
SUBMIT Interpreter	Starts or stops the SUBMIT interpreter
Status keys	Status keys for technology packages
Jogging	Program and jog override, mouse settings
User group	Access to certain user levels via password
Cur. tool/base	Current tool, base system and external kinematic system
Tool definition	Settings for tool type, base type and external axes
On/Off Options	Cold start and PowerOff Delay
Miscellaneous	
Language	The language used in the graphical user interface
Change password	Changes the user level access password
Editor -> Def-line *1	Displays the DEF lines in a program
Editor -> ASCII Mode *1	Additional information in expert mode
Editor -> Linebreak *1	Line break in program window
Office GUI *1	Displays the KCP operator control elements for mouse operation
Monitoring working envelope -> Override *1	Switches off the monitoring of the work envelope
Monitoring working envelope -> Configure *1	Setting of Cartesian or axis-specific (joint) work envelopes
Reinitialization -> Bof Reinitialization *1	The BOF = GUI (graphical user interface) is reinitialized without rebooting the system
Cycle Time Optimizer *1	Acceleration adaptation for various technologies
Event planner	Carries out actions at specified times or under specified conditions.
“File” menu	
Archive	
Machine data	Saves machine data to floppy disk
Configure *1	Save various configuration settings to floppy
Restore	
Machine data	Restores the machine data from floppy
Configure *1	Restores configuration settings from floppy
*1 Not available in the user group “User”	

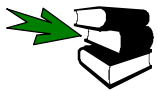
1.1 The “Configure” menu

A wide range of functions are grouped together in the menu “Configure” with which settings can be made to the robot system.

1.1.1 I/O



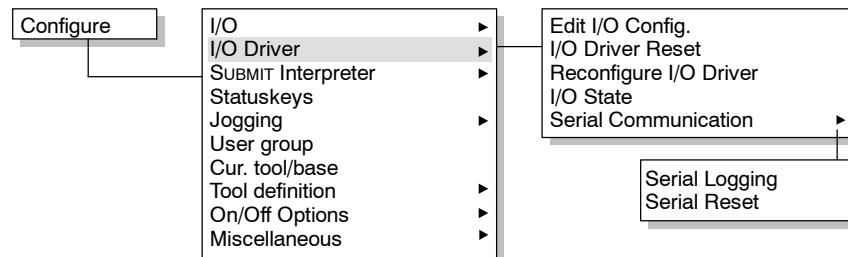
1.1.1.1 Automatic External



For information on configuring the “Automatic External” interface, please refer to the **Programming Handbook** under **[Configuration]**, chapter **[Automatic External]**.

1.1.2 Drivers

Using the functions offered here, you can configure and reset the peripheral interfaces on the robot system.



1.1.2.1 Edit I/O Config.

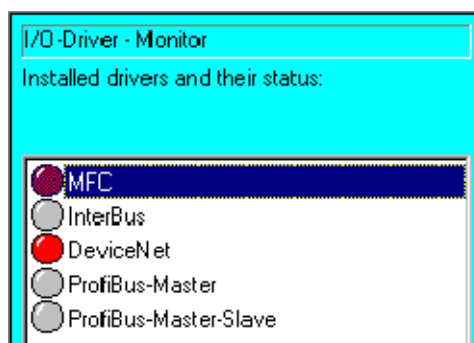
The "IOSYS.INI" file is loaded into the editor for editing. This file can be found in the directory "C:\KRC\ROBOTER\INIT\".

```

1  ;=====
2  ; IOSYS.INI - Configuration file for the I0-System
3  ;=====
4  ; For configuration help go to the end of this file.
5  ;-----
6
7  ; ATTENTION !!!! Since V5.0 Build13 we have removed the DeviceNet
8  ;                 driver "dndrv.o". Now you have to use the driver
9  ;                 "dn2drv.o" and the appropriate syntax (form 2)
10
11 [CONFIG]
12 VERSION=2.00
13
14
15 [DRIVERS]

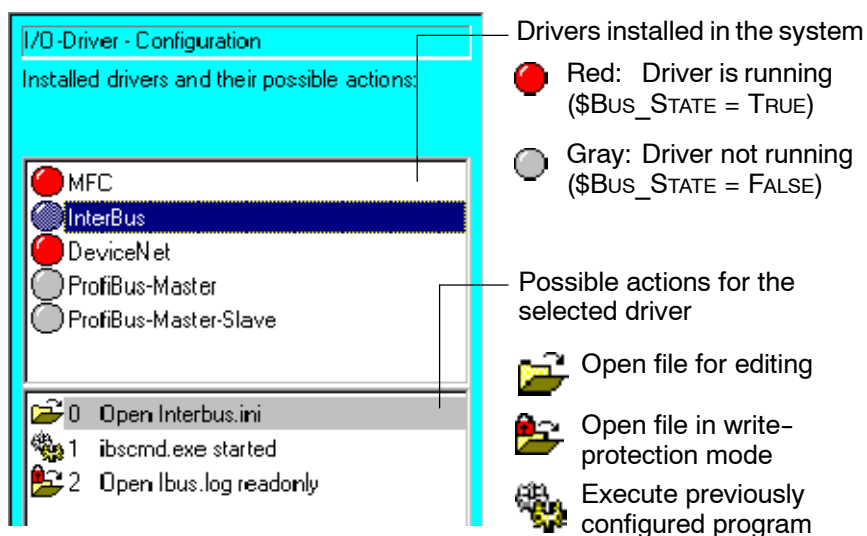
```

1.1.2.2 I/O Driver Reset

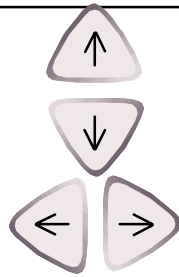


Drivers can be displayed, installed, uninstalled and modified using this menu command. When the menu item "I/O Driver" is selected, the status window illustrated here is displayed:

Configure Makes it possible to modify the I/O configuration. The status window changes for this purpose and further softkeys become available.



The status window is divided into two sections. The upper section contains the list of drivers, the lower section the list of actions.



The “↑” and “↓” arrow keys can be used to select the desired driver or action.

The “←” and “→” arrow keys are used to switch between the driver list and the list of actions. Alternatively, the “TAB” key can be used for this.

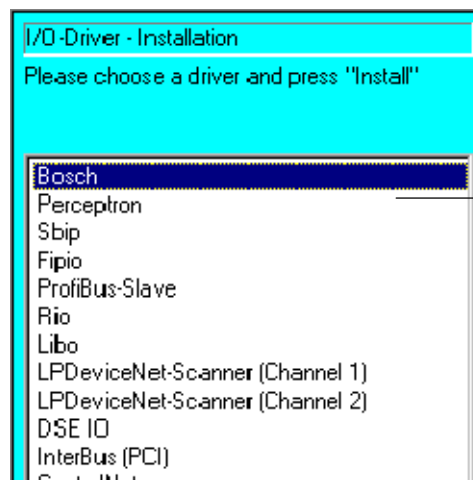


The softkeys “New inst.,” “Iosys” and “Uninstall” are not available in the user group “User”.

Driver list

New inst.

This function opens a new status window for installing additional drivers.



List of the drivers that can still be installed. Existing drivers are not listed.

Back

Reopens the main I/O configuration window.

Install

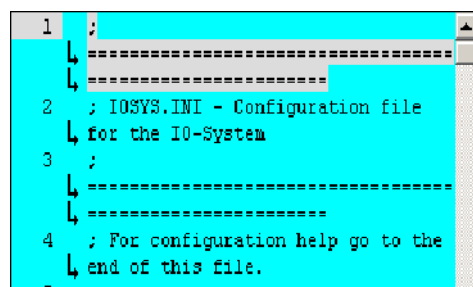
Installs the selected driver.

Close

Terminates the configuration of the I/O drivers.

Iosys

The softkey “Iosys” loads the file “IOSYS.INI” into the editor for editing.



NAVIGATOR

Brings the Navigator to the foreground.

Close

Saves the changes and closes the editor.

Reconfig.

All I/O drivers are reconfigured. No program may be selected. While this function is being executed, the functions in the softkey bar are not available.

Info

Further information can be obtained by pressing the softkey “Info” which loads the write-protected file “IOSYS.LOG” into the editor.

```

1 KUKA IOSYS LOGFILE
2 -----
3 Log Date 3.7.2 : 06:40
4
5 open ini-file successful, version
6 2.00
7 driver MFCDEV information
8 object file
9 driver ready 1
10 driver version 0200
11 function entry
12 input objects 1
13 output objects 2
  
```

NAVIGATOR Brings the Navigator to the foreground.

Close Saves the changes and closes the editor.

Uninstall The selected driver is uninstalled and the entry is deleted from the driver list.

Reset The selected driver is reset.

List of actions

If the list of actions is active, the softkeys “Uninstall” and “Reset” are not available. They are replaced by “Jump” and “OK”.

Jump The focus moves to the driver list.

OK The selected action is executed. The following actions are possible:

- Load file into the editor
- Load file into the editor in write-protection mode
- Execute defined program

Close The action can be terminated and the status window closed using the softkey “Close”.

1.1.2.3 Reconfigure I/O Driver

This menu item is used to reset the drivers to the state they had immediately after the cold start of the controller. The Ini files are then read and the bus configured accordingly.

Ti...	no.	Source	Message
16:48	202		I/O Reconfiguration started, please wait
16:48	203	/	General motion enable

The message in the message window must then be acknowledged.

Ti...	no.	Source	Message
14:57	1210	/	Ackn. general motion enable

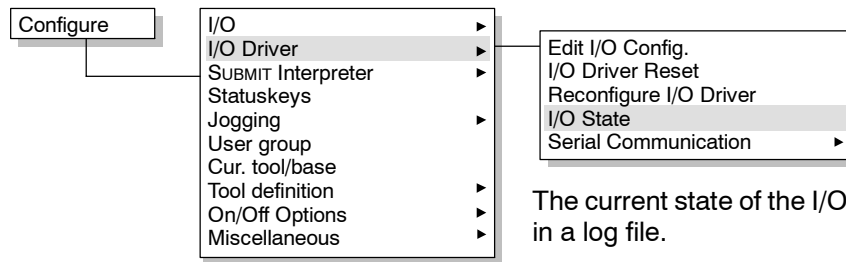
Ackn Deletes the selected message.

Ackn. All Deletes all the messages in the message window which can be deleted.



This is only possible in mode “T1”, “T2” or “AUT”.

1.1.2.4 I/O State



The current state of the I/O system is output in a log file.



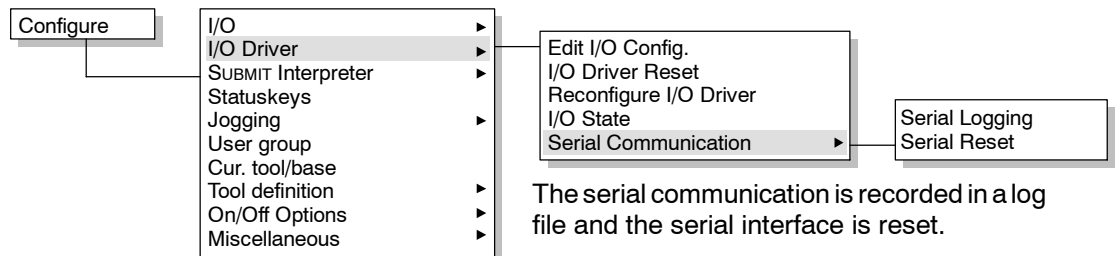
The information is saved in the file "C:\KRC\ROBOTER\LOG\IO_ACT.LOG".

The following information is saved in the log file:

- Date and time
- The I/O drivers integrated in the system, with details of version, state, etc.
- Overview of the inputs and outputs
- Overview of the analog outputs, with details of the value, resolution, type, etc.

If this item is called again, the old file is overwritten.

1.1.2.5 Serial Communication



The serial communication is recorded in a log file and the serial interface is reset.

Serial Logging

The recording of the serial communication is activated/deactivated. Calling this option the first time starts the recording. This option is now preceded by a check mark in the menu. Calling this menu item again stops the recording and writes the data to the log file. The check mark is then no longer displayed in the menu.

Next time recording is started, the old log file is overwritten.



The information is stored in the file "C:\KRC\Roboter\Log\SerialCom.LOG".

The following information is saved in the log file:

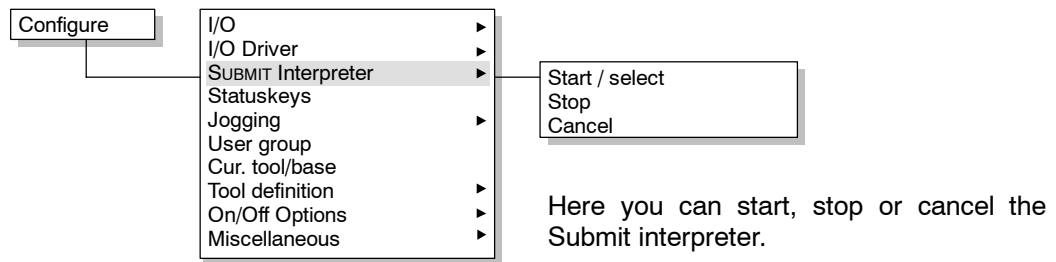
- Date and time of the recording
- Opened/closed channels
- All characters transmitted and received
- Transmission errors (e.g. "Parity")
- Driver error numbers (e.g. "errorGet()")
- Configuration parameters.

Serial Reset

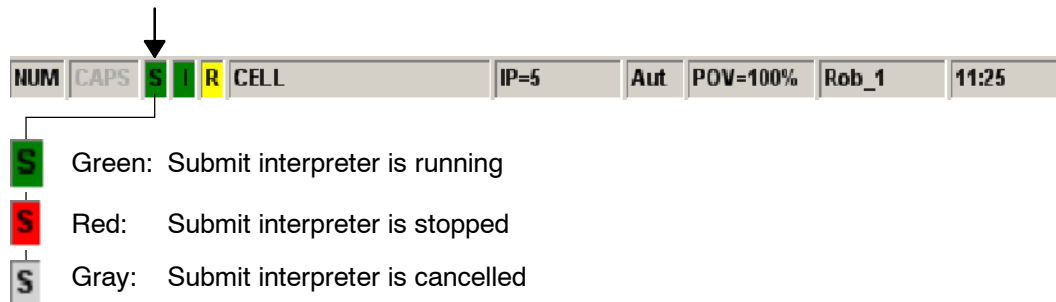
The serial interface is reset to the state it had immediately after the cold start of the system.

1.1.3 Submit interpreter

The Submit interpreter is a program which runs in the background parallel to the robot program. As this program runs entirely independently of the selected robot program, it can be used to handle all manner of different control tasks. These might include, for example, the control and monitoring of a cooling circuit, the monitoring of safety equipment or the integration of additional peripheral devices. This renders the use of an additional PLC for smaller tasks unnecessary as these tasks can be accommodated by the KR C1.



The current status of the Submit interpreter is displayed in the status line.



Only possible in mode T1 or T2.

1.1.4 Status keys

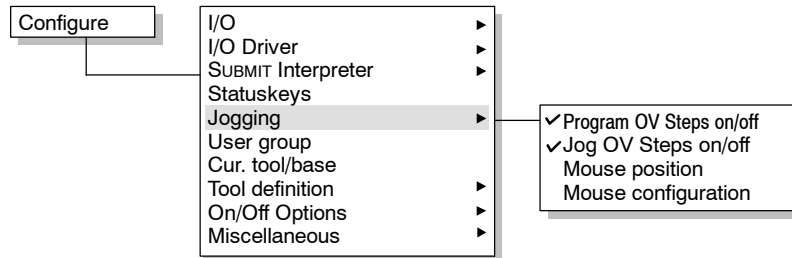
This submenu is reserved for optional technology packages. Select here the functions that are to be assigned to the freely available status keys.



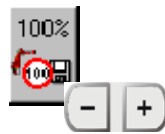
The selected option remains the same after a system reboot.

1.1.5 Jogging (Override)

This menu item enables you to set the increment of the jog override (HOV) and the program override (POV). The mouse position and configuration can also be changed.



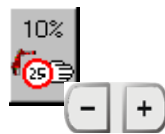
1.1.5.1 Program override steps (POV)



The POV value is usually toggled between 100, 75, 50, 30, 10, 3 and 1 percent. The option “Program OV Steps on/off” makes it possible to change the programmed velocity in increments of 1%.

Use the “+/-” key next to the status key for program override to vary the setting.

1.1.5.2 Jog override steps (HOV)



The HOV value is likewise usually toggled between 100, 75, 50, 30, 10, 3 and 1 percent. The option “Jog OV Steps on/off” makes it possible to change the jog velocity in increments of 1%.

Use the “+/-” key next to the status key for jog override to vary the setting.

1.1.5.3 Mouse position

If the robot is moved in the robot coordinate system using the Space Mouse, the operator can inform the controller of his position.

1.1.5.4 Mouse configuration

Here you can set the degrees of freedom and the dominant axis of the Space Mouse.

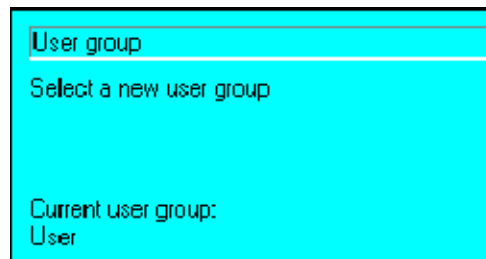
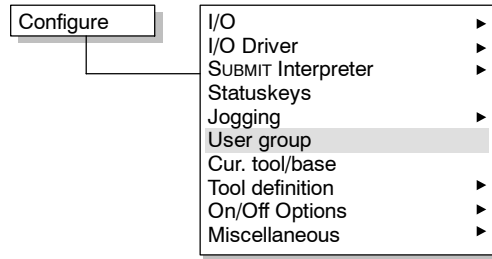
1.1.6 User group

For the purposes of increasing system security, robot controller functions and/or the programming thereof can be disabled for certain user groups. This can be done by restricting access to these functions to specific “user levels”. Access is then protected by a password.

By default, the software for the KRC controller makes a distinction between users, experts and administrators. Users do not require any knowledge of programming syntax, as they create programs by means of menus. Whenever the system is booted, the user level is automatically selected by default.

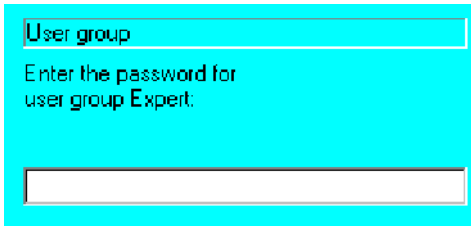
If the functions of the user level are not sufficient, it is possible to switch to the expert level. Experts can then use the ASCII keypad to program in the robot programming language KRL (KUKA Robot Language) and to edit system or initialization files (bus systems). KRL is a high-level, PASCAL-based programming language, which is thus also suitable for programming complex tasks.

Access to the expert level is protected by a password.



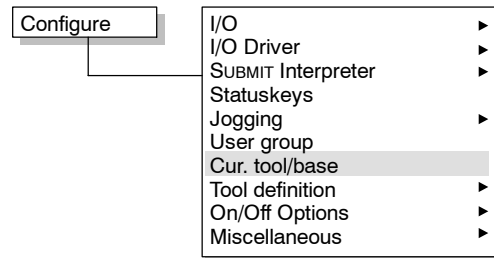
The status window illustrated here is opened.

Under normal circumstances, when the system is started, you will find yourself in the "User" group. Higher "user levels" can only be reached by entering a level-specific password.

User	Once one of the softkeys available for user group selection has been pressed, the content of the status window changes. You are asked to enter the password for the user group selected.
Expert	
Administrator	
	 <p>The image shows a cyan status window titled 'User group' with the text 'Enter the password for user group Expert:' and an empty input field below it.</p>
OK	Enter the password for the selected user group. Pay attention to the use of upper and lower case characters when entering passwords. Then press the softkey "OK".
Cancel	You can exit this function at any time. In this case the user group does <u>not</u> change.

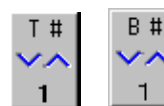
1.1.7 Cur. tool/base

Here you can select the current tool and base system to be used. The names assigned to the individual numbers can be changed using the tool definition function.

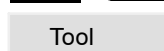


A screenshot of a configuration window titled 'current tool/base/ext. Joint'. It contains several input fields: 'Tool no.' with value '1' and a robot icon; 'Tool name:' with value 'Greifer'; 'Base No.' with value '0'; 'Base system name:' with value 'Bauteil'; 'Ext. kinematic system' with value '0'; and 'Name of ext. joint:' with an empty field. At the bottom, it says 'Select tool, base or ext. kinematics to be activated.'

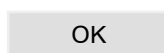
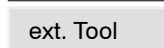
Standard tool or external tool
 Current tool [0...16]
 Name of the current tool
 Current base [0...16]
 Name of the current base system



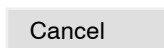
You can toggle to the next or previous input box using the “↓” and “↑” arrow keys. The numbers can either be entered using the numeric keypad or altered by means of the corresponding status key.



Toggle between a tool mounted on the robot and an external tool.



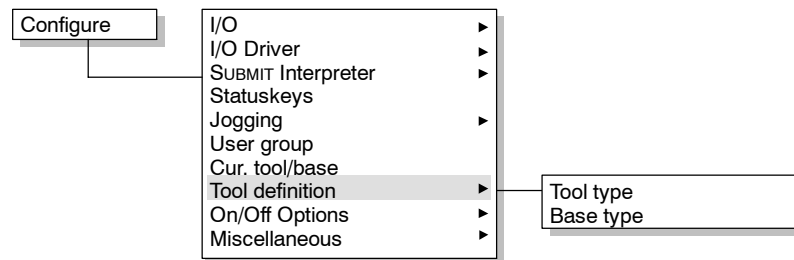
Saves the settings made. If the permissible range of values is exceeded in any of the input boxes, or if the number of an undefined tool or base is selected, a corresponding error message is generated in the message window.



Closes the status window without saving the settings made.

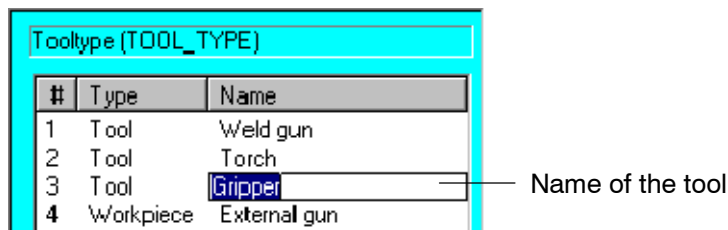
1.1.8 Tool definition

This function is used to assign names to the tool type, base type and external axis.



One of the following status windows “Tool type” or “Base type” is then opened. Various options are available in the softkey bar:

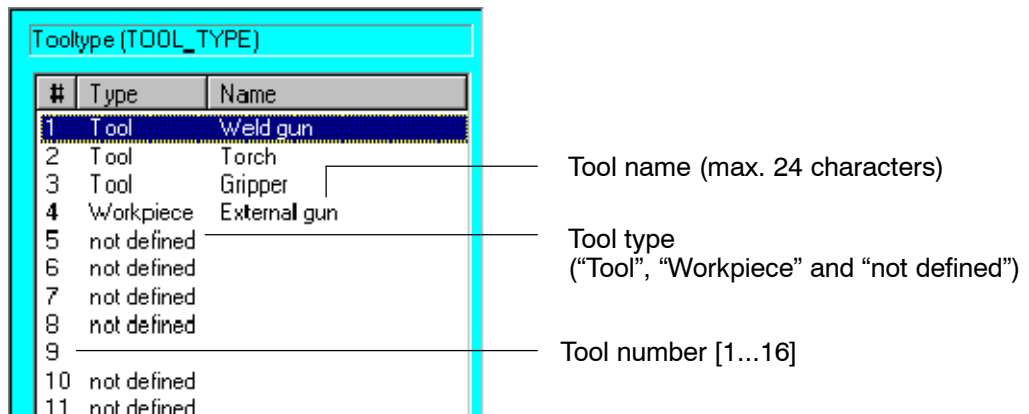
Name The name of the calibrated type can be altered. This is done by selecting the desired line using the “↓” and “↑” arrow keys and pressing the softkey “Name”.



OK Saves the changes made and closes the status window.

Cancel Closes the status window without saving the changes made.

Tool type



- Tool
Normal tool on the robot flange
- Workpiece
The robot moves the workpiece
- not defined
No tool type has yet been calibrated

Base type

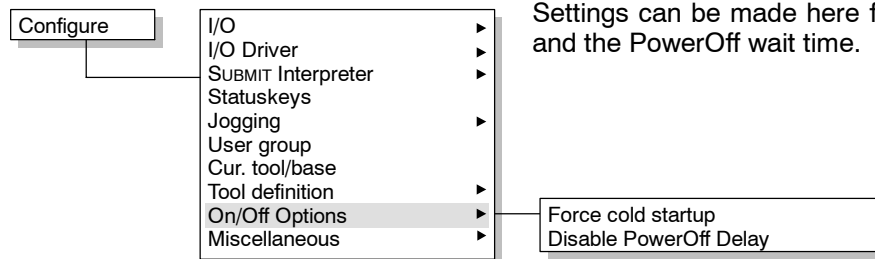
Basetype (BASE_TYPE)		
#	Type	Name
1	Offset	Part A
2	Offset	Part B
3	Tool	Part C
4	not defined	
5	not defined	
6	not defined	
7	not defined	
8		
9	not defined	
10	not defined	

Name of the base system (24 characters)

Base type
("Tool", "Offset" or "not defined")

Base system number [1...16]

- Tool
An external tool (e.g. welding gun) is mounted
- Offset
A base type has been calibrated
- not defined
No base type has yet been calibrated

1.1.9 On/Off Options**1.1.9.1 Force cold startup**

This menu item is available at both user and expert level. When a cold start has been forced, and the system has booted, the controller displays the Navigator. No program is selected; the controller is completely reinitialized.



The menu command "Force cold startup" is not retained as a default setting, i.e. it must be activated each time a cold start is required.

In the event of a warm restart, on the other hand, which the controller also initiates itself following a power failure, the robot program that was selected can be resumed. The state of the kernel system, e.g. programs, block pointer, variable contents and outputs, is completely restored. The power failure could have been caused, for example, by failure of the power supply unit or by activation of the main switch while the program was running.

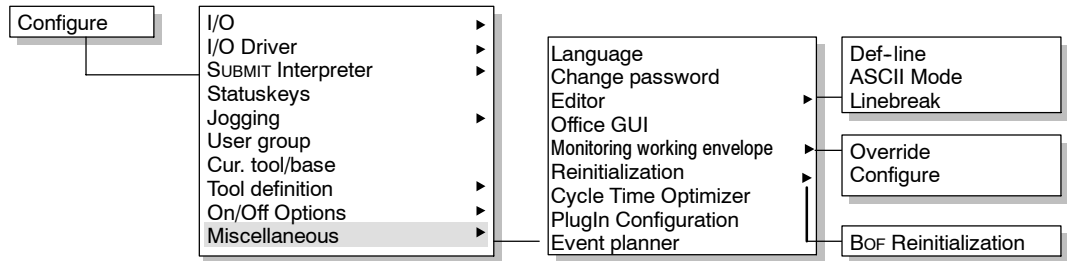
If the controller detects a system fault or altered data after the restart, it automatically forces a cold start.

1.1.9.2 Disable PowerOff Delay

This command, available in expert mode, offers the operator the possibility of reducing the preset delay time until the system is shut down.

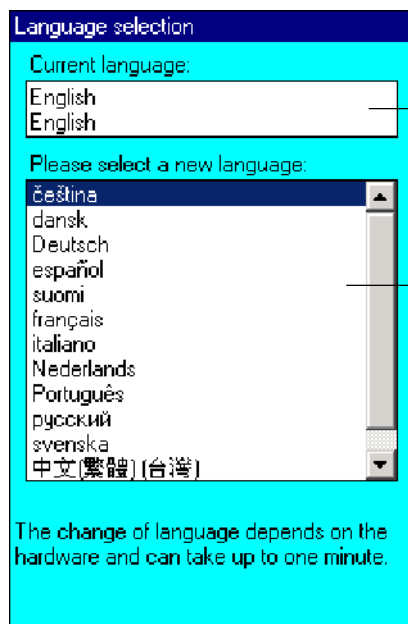
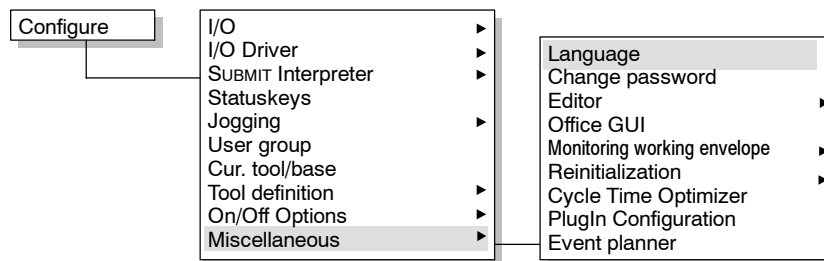
1.1.10 Miscellaneous

Further options are grouped together under this menu item and are described in more detail below.



1.1.10.1 Language

Here you can set the user interface to your language.



Language currently set

Foreign languages available



The arrow keys “↓” and “↑” can be used to select the desired language.

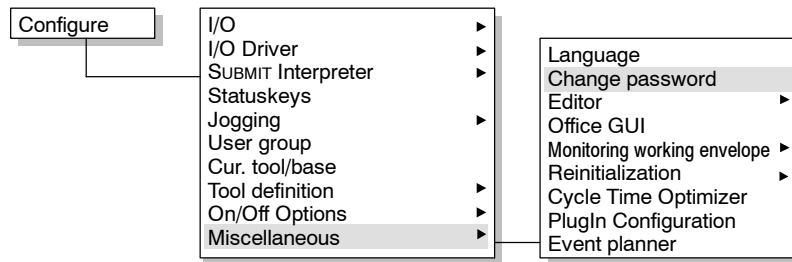
- Accepts the selection made and closes the status window. After a short wait time the user interface is then displayed in the desired language.
- Accepts the selection made and switches the user interface to the desired language. The status window stays open.
- Closes the status window without changing the user interface.



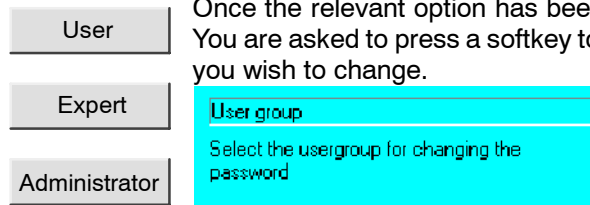
If characters are displayed incorrectly on the user interface after switching languages, the language of the operating system must also be set accordingly.

1.1.10.2 Change password

Select this option to change the access password for a user level.



Once the relevant option has been selected, a status window is opened. You are asked to press a softkey to select the user group whose password you wish to change.

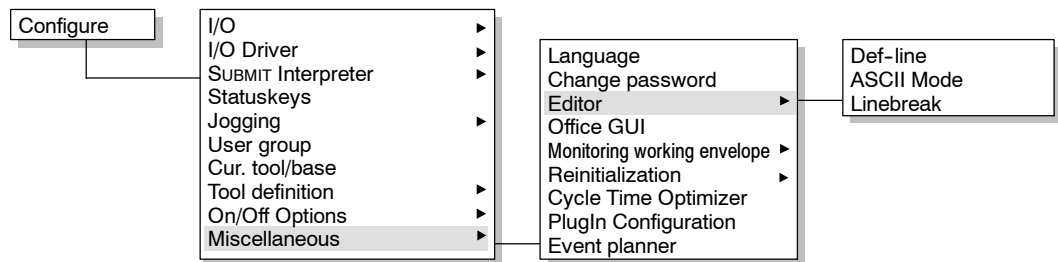


Once you have selected a user group, a further status window appears. Enter the old and new passwords, entering the new password a second time for confirmation.

- If the passwords were entered correctly, the password is changed.
- The operation is terminated without saving the data you have entered.

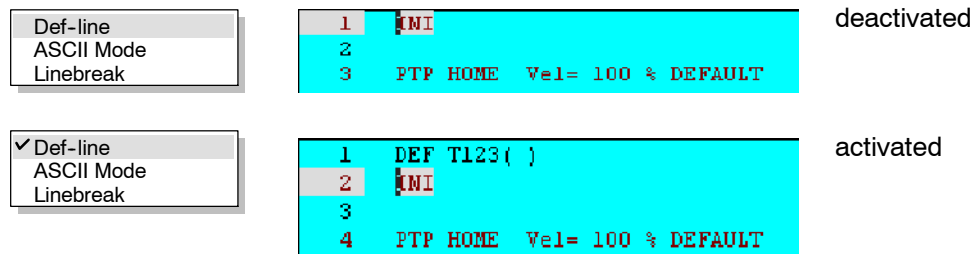
1.1.10.3 Editor

Determines the type of view in the editor or in the selected program.



Def-line

If this function is activated, the DEF line in the program, which is normally hidden, is displayed.

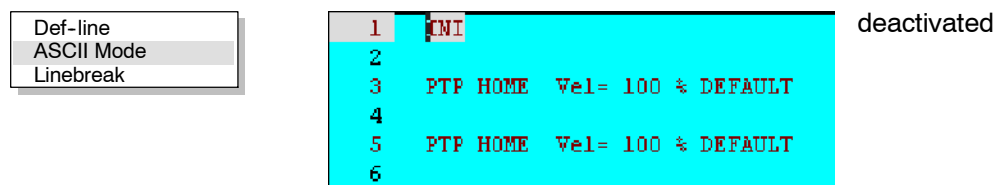


Declarations can only be made once the DEF line is visible. This function is not available, by default, below the user group “Expert”. It is automatically deactivated as soon as the operator carries out a restart or switches back to “User” mode.

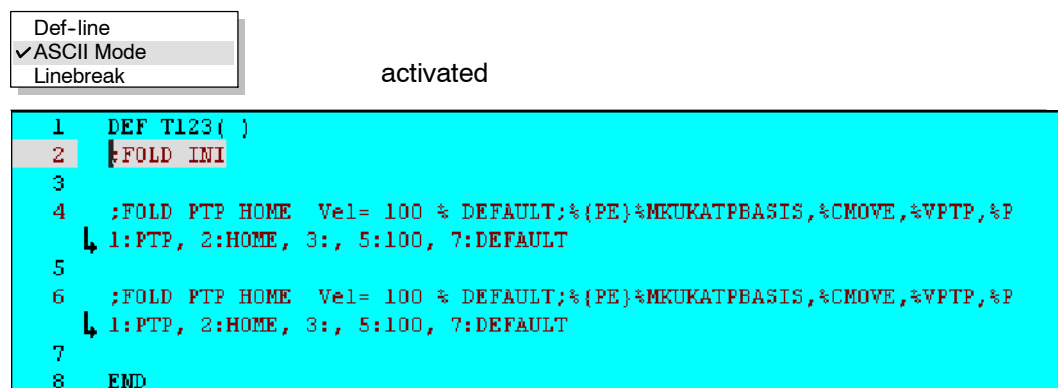
ASCII Mode (detail view)

This function is only available in Expert mode and is another aid to keeping the amount of information on the user interface as low as possible. “ASCII Mode” is deactivated by default.

If the function “ASCII Mode” is deactivated, all texts written after the “;” sign in a FOLD line, for example, are suppressed. This information is needed, however, for displaying an inline form.



When the function is activated, further information is displayed which normally remains hidden.





The programmer is only shown all available lines when all the FOLDS are open and “ASCII Mode” is activated. The display on the user interface is then equivalent to the display in a normal text editor.

```

1 DEF T123( )
2 :FOLD INI
3   :FOLD BASISTECH INI
4     GLOBAL INTERRUPT DECL 3 WHEN $STOPMESS==TRUE DO IR_STOPM ( )
5     INTERRUPT ON 3
6     BAS (#INITMOV,0 )
7   :ENDFOLD (BASISTECH INI)
8   :FOLD USER INI
9     ;Make your modifications here
10
11   :ENDFOLD (USER INI)
12 :ENDFOLD (INI)
13
14 :FOLD PTP HOME Wel= 100 % DEFAULT;#{PE}MKUKATPBASIS, %CMOVE, %VPTP, %P
  ↳ 1:PTP, 2:HOME, 3:, 5:100, 7:DEFAULT
15 $BUDSTART = FALSE
16 $H_POS=%XHOME
17 PDAT_ACT=PDEFAULT
18 BAS (#PTP_DAT )
  
```

Linebreak

You can switch the line break on or off as required in the program window.

Def-line
ASCII Mode
 Linebreak

activated

```

9 AUTOEXT INI
10 LOOP
11   POO (#EXT_PGNO,#PGNO_GET,DMY[,0 )
12   SWITCH PGNO ; Select with
  ↳ Programmnumber
13
14   CASE 1
15     POO (#EXT_PGNO,#PGNO_ACKN,DMY[,
  ↳ 0 ) ; Reset Progr.No.-Request
16     ;EXAMPLE1 ( ) ; Call
  ↳ User-Program
  
```

The line break function is activated by default; all of the available information is then displayed in the program window. If the line is too long for the program window it is broken at an appropriate point.



The part of the line after the break has no line number and is marked instead with an arrow.

Def-line
ASCII Mode
 Linebreak

deactivated



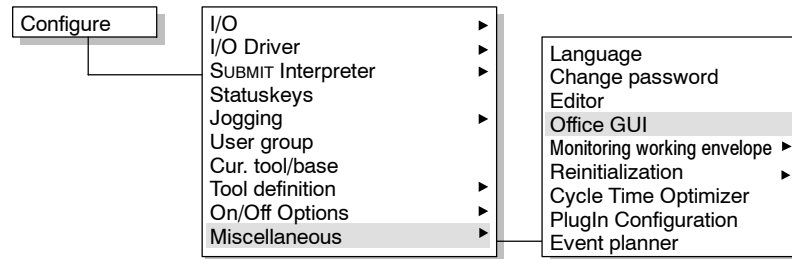
```

9 AUTOEXT INI
10 LOOP
11   POO (#EXT_PGNO,#PGNO_GET,DMY[,0 )
12   SWITCH PGNO ; Select with Program
13
14   CASE 1
15     POO (#EXT_PGNO,#PGNO_ACKN,DMY[,
16     ;EXAMPLE1 ( ) ; Call User-Program
  
```

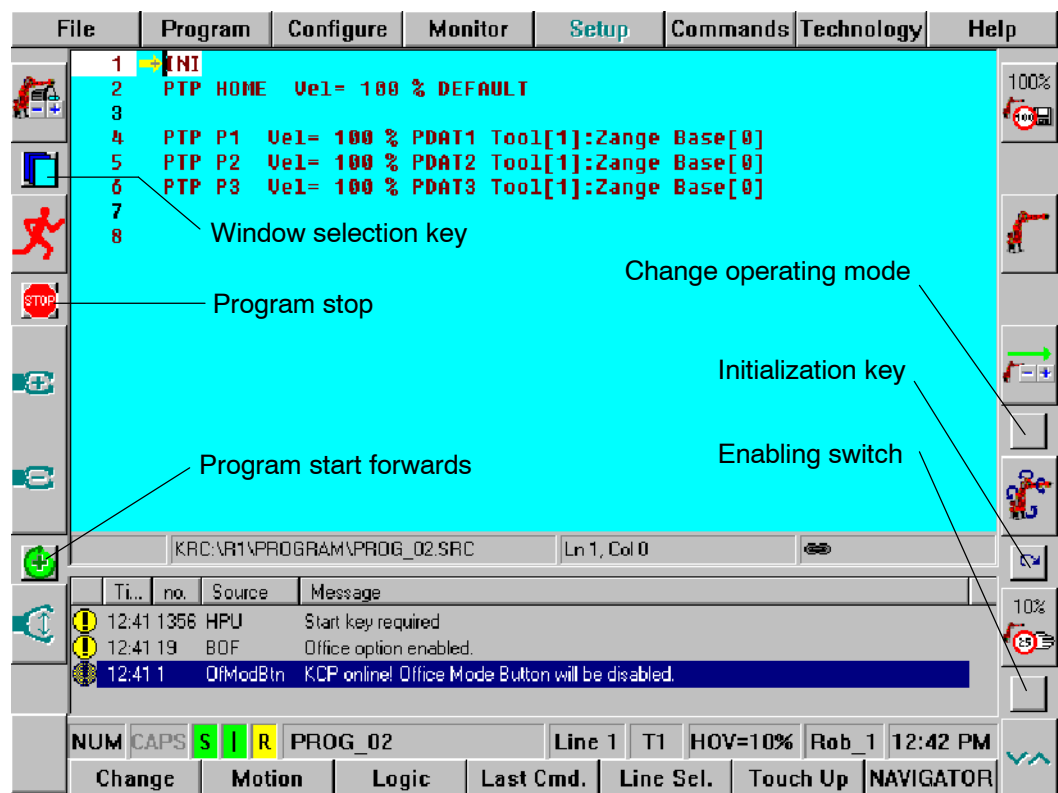
If the line break function is deactivated and the line does not fit in the program window because it is too long, the contents of the program window can be moved. This is done using the arrow keys “←” and “→”.

1.1.10.4 Office GUI

This menu item activates the KCP operator control elements “Window selection key”, “Stop”, “Program start forwards”, “initialization key” and “enabling switch” on the KCP display. This allows easy operation via the mouse.



When this menu command is selected, the “Office GUI” is activated. A corresponding message is generated in the message window.



The symbols have the following meaning:



Left-clicking with the mouse on this symbol allows you to toggle between the program, status and message windows.



If the mouse pointer is positioned over this symbol, program execution can be stopped with a left mouse-click.



Clicking on this symbol with the mouse activates program start forwards.



If no KCP is available, you can change operating mode by clicking on this symbol.



Clicking on this symbol causes the KUKA technology packages, the data list and all *.OCX files to be re-initialized, i.e. reloaded.



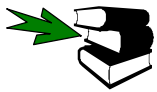
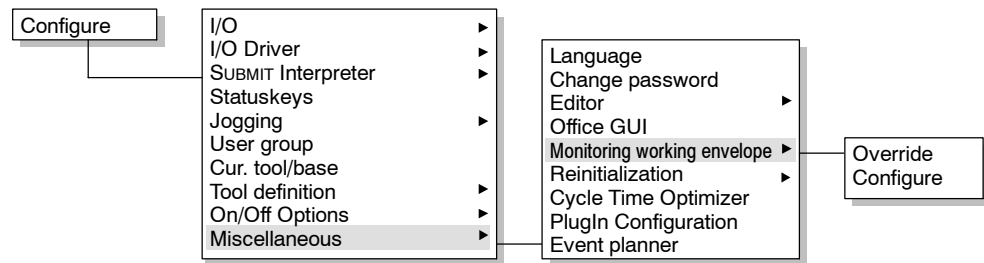
Clicking on this symbol is equivalent to pressing the enabling switch.

If this command is selected again, the option is switched back off. Here again a message is generated in the message window.

Ti...	no.	Source	Message
17:09	20	BOF	Office option disabled.

1.1.10.5 Monitoring working envelope

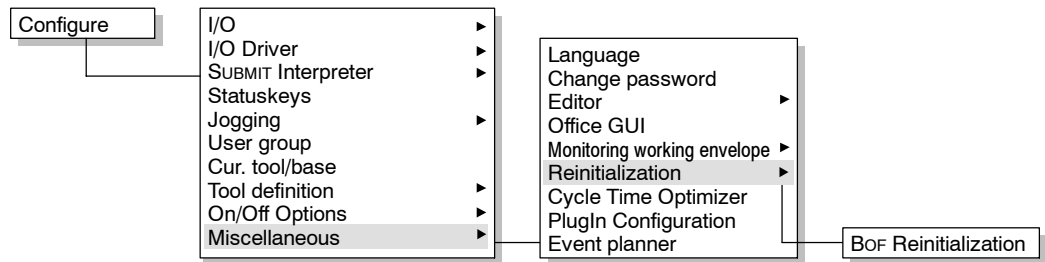
The work envelope monitoring function can be switched off, for example, in order to move a robot back out of a violated workspace.




Further information on workspaces can be found in the chapter **[Configuring the system, Expert]**, section **[Workspace monitoring]**.

1.1.10.6 Reinitialization



The graphical user interface is reinitialized without rebooting the system.



The progress of the reinitialization is then shown in the message window.

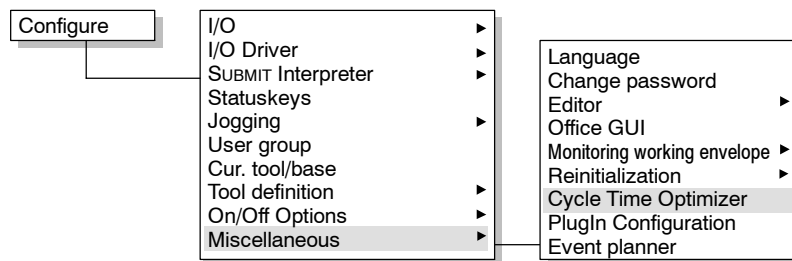
C...	Time	no.	Source	Message
	8:54:43 AM	1	HMI	Reinitialization in progress.

Completion of the reinitialization is also indicated in the message window.

C...	Time	no.	Source	Message
	8:53:33 AM	0	TH	TechHandler reinitialized.
	8:53:35 AM	1	TPS	TechPack KUKATPBASIS initialized!

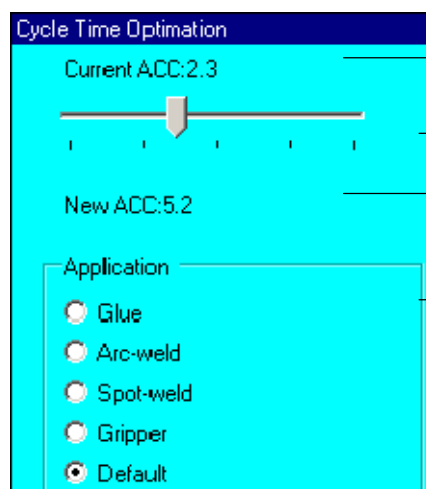
1.1.10.7 Cycle Time Optimizer

This option is used for optimizing cycle times. To do this, the value of the maximum permissible acceleration can be changed for different technology packages.



The entries offered depend on the technology package being used.

The following options are available in the status window:



The maximum acceleration value currently set
 Setting the new acceleration value
 New maximum acceleration value
 Technology package affected

The status keys "ACC" and "APP" can be used to select the new acceleration value to be used and the technology package affected.

Current ACC

The current setting for the maximum acceleration with which the robot is moved.



The value is saved in the variable "DEF_ACC_CP".

New ACC



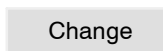
The maximum value to be used for the acceleration with which the robot may be moved. By default, the acceleration is set so as to trigger the robot controller monitoring functions as rarely as possible. In order to optimize cycle times, the maximum permissible value for the acceleration can be increased or decreased.

The higher the value is set, the more likely robot controller monitoring functions are to be triggered. In such cases, the robot is stopped and a corresponding error message is displayed in the message window. Reduce the value until the program runs without error messages.

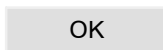
Application



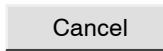
The permissible acceleration depends on the technology package being used; this can be selected here using a status key. If a new entry is selected, the predefined default value for "Current ACC" is displayed.



Accepts the setting for the currently selected technology package.



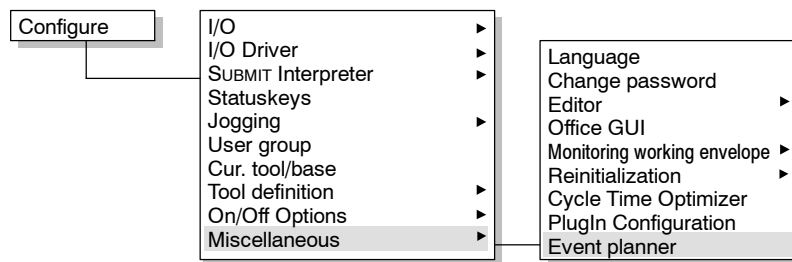
The settings are saved and the status window is closed.



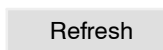
Closes the status window.

1.1.10.8 Event planner

Used for executing certain actions which are triggered according to a schedule or when certain conditions are satisfied.



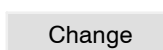
Scheduler Configuration	
Name	Description
Scheduled Actions	Scheduled Actions
Conditional Actions	Conditional Actions



Refreshes the display of the event planner.



Moves the focus to the next element.



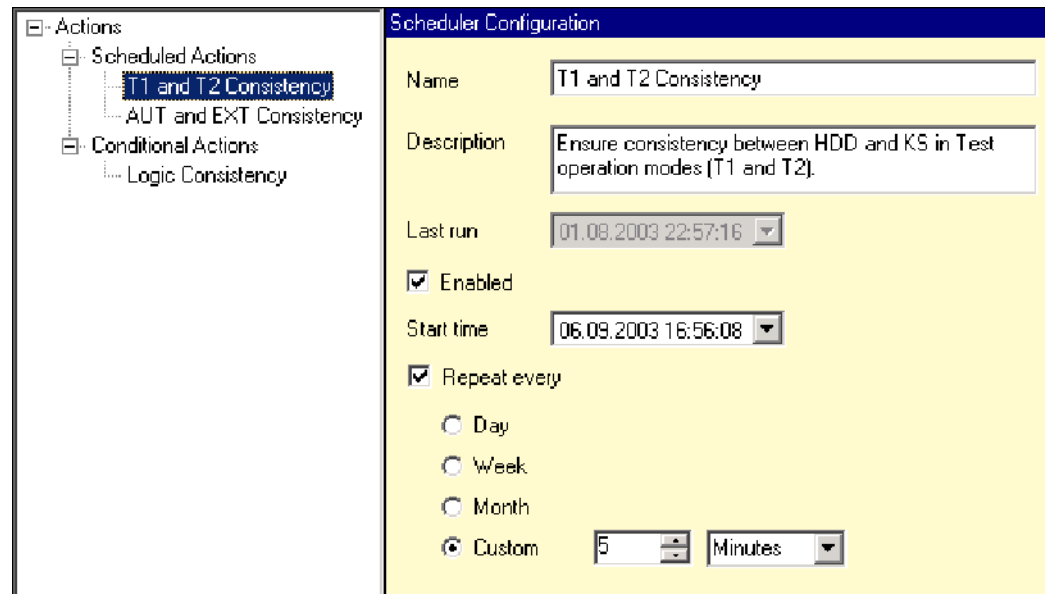
The changes are saved.

Close

Closes the event planner. Any unsaved changes will be lost.

Scheduled actions

The action can be started at a specified time on a specified day. It is also possible to specify a repeat interval.



The screenshot shows a software interface with a tree view on the left and a configuration panel on the right. The tree view shows 'Actions' expanded to 'Scheduled Actions', with 'T1 and T2 Consistency' selected. The configuration panel, titled 'Scheduler Configuration', contains the following fields and options:

- Name: T1 and T2 Consistency
- Description: Ensure consistency between HDD and KS in Test operation modes (T1 and T2).
- Last run: 01.08.2003 22:57:16
- Enabled
- Start time: 06.09.2003 16:56:08
- Repeat every
 - Day
 - Week
 - Month
 - Custom: 5 Minutes

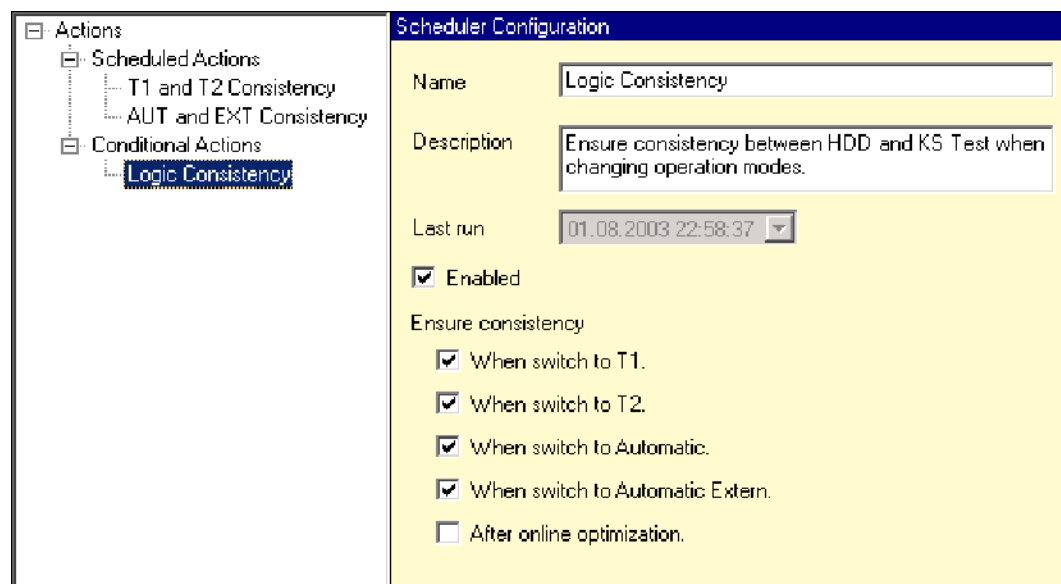
In operating modes “T1” and “T2”, a consistency check is carried out between the kernel system and the hard drive every 5 minutes.



In order not to shorten the service life of the hard drive unduly, the repeat interval for the consistency check between the kernel system and the hard drive should not be made too short.

Conditional actions

The action is executed as soon as a specified condition is fulfilled.



The screenshot shows the same software interface as above, but with 'Logic Consistency' selected in the tree view. The configuration panel, titled 'Scheduler Configuration', contains the following fields and options:

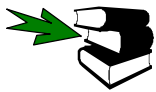
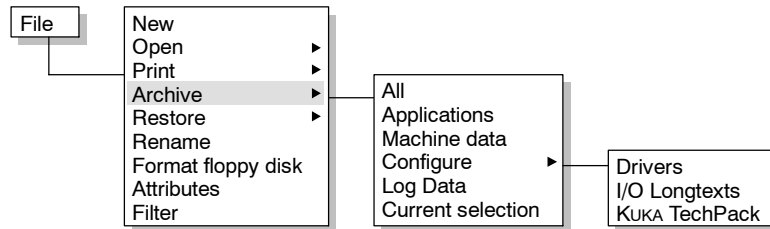
- Name: Logic Consistency
- Description: Ensure consistency between HDD and KS Test when changing operation modes.
- Last run: 01.08.2003 22:58:37
- Enabled
- Ensure consistency
 - When switch to T1.
 - When switch to T2.
 - When switch to Automatic.
 - When switch to Automatic Extern.
 - After online optimization.

When switching between operating modes, a consistency check is carried out between the kernel system and the hard drive.

1.2 The “File” menu

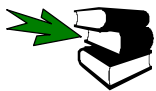
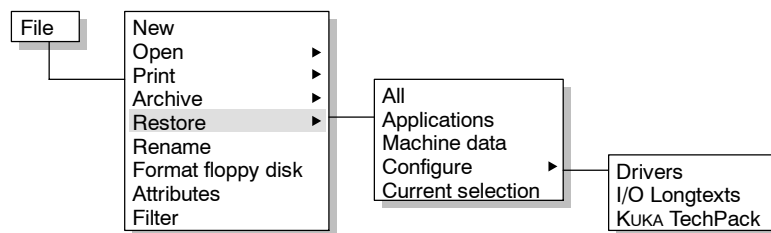
This function is used to back up certain settings to floppy or restore them from floppy.

1.2.1 Archive



Further information on this topic may be found in the **Operating Handbook**, in main chapter **[Operator Control]**, chapter **[Navigator]**, section **[“File” menu]** under **“Archive”**.

1.2.2 Restore



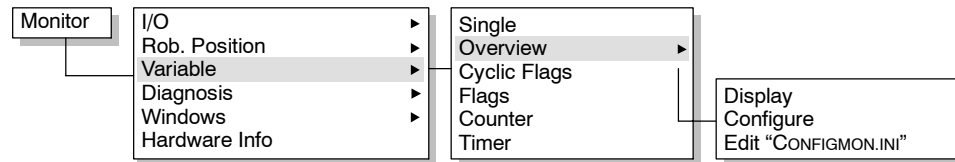
Further information on this topic may be found in the **Operating Handbook**, in main chapter **[Operator Control]**, chapter **[Navigator]**, section **[“File” menu]** under **“Restore”**.



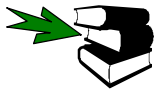
2 Configuring the system, Expert

2.1 Variable overview

The menu commands “Configure” and “Edit CONFIGMON.INI” make it simple to display more than one variable at a time.



2.1.1 Display



A description of the variable display function can be found in the **Operating Handbook** in main chapter **[Operator control]**, chapter **[Monitor]**.

2.1.2 Configure

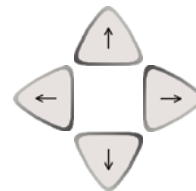
Variable overview - Configuration				
	Status	Name	Variable	Value
1		Test1	\$IN[1]	
2		Test2	\$IN[2]	

Example1 Example2

Column width: 26 Editable: User

Row height: 25 Visible: Expert

When this command is selected, the corresponding status window is displayed. Here the variables can be grouped together in different groups.



The arrow keys can be used to select a specific element in the group and make modifications.

Status The status indicates whether or not a variable is continually updated. A maximum of 12 variables per group can be monitored continuously.

Name A description of the variable, for easier identification.

Variable The variable name used by the system.

Value On selection of the menu command “Variable” -> “Overview” -> “Display”, the value of the variable is displayed here.

Example1 Ex The tab indicates which of the (max.) 10 groups is currently displayed.

Column width The desired values for the column width and row height can be entered using the numeric keypad.

Test1 Row height: 25

Column width: 26



The column width and row height can also be changed directly using a PC mouse connected to the system. Move the mouse pointer to the dividing line in the title bar or column, hold the left mouse button down and drag the line to the desired position.

Editable The user group in and above which the variable group can be displayed and modified is set via the corresponding selection menu.

Changes can be made in this user group or higher

Editable: User

Visible: Expert

The variable group is displayed in this user group or higher

Monitor Summons the variable overview.

Tab + Selects the next available group.

Jump Moves the focus in the status window to the next available input box.

Insert Allows the insertion of new groups or rows. The following options are available:

Insert group

- G. before A new group is inserted **before** the one currently selected.
- G. after A new group is inserted **after** the one currently selected.

Variable overview - Configuration

Please type in a name for the new group.

Groupname:

Editable: Expert

Visible: User

The name of the group, which is displayed in the tab (max. 25 characters)

The user group in and above which this variable group is displayed and can be edited

- Jump The focus is moved to the next element.
- OK The entries are saved.
- Cancel No new group is inserted.



A maximum of 10 groups is possible.

Insert row

R. above | A new row is inserted **above** the one currently selected.

R. below | A new row is inserted **below** the one currently selected.

The desired values can then be entered in the new row. Move the focus to the desired box and press the Enter key. It is now possible to enter values (names, variables, etc.) in the box thus selected.

	Name	Variable
1	WarTest	

Cancel | No new row is inserted.



A maximum of 25 rows per group is permissible.

Delete | Allows you to delete individual rows or a whole group. The following options are available once the softkey has been pressed:

Row | The current row is deleted.

Group | The current group is deleted.

Cancel | The action is canceled.

OK | Pressing the softkey "OK" accepts the changes and closes the status window.

Cancel | The action is terminated and the status window closed.

2.1.3 Edit CONFIGMON.INI

This command is used to load the file "CONFIGMON.INI" into the editor for editing.

```

1  :-----
2  ; KUKA Roboter GmbH
3  ; Configuration-Monitor settings for (V) KR
4  ; created automatically by upgrading an older version of
   L ConfigMon.ini:
5  ; old version: 2.0.0
6  ; new version: 2.0.0
7  ; date: 07/23/03 13:26:06
8  :-----
9  [Version]
10 Version=2.0.0
11
12 [Group1]
13 GroupTitle=Example1

```

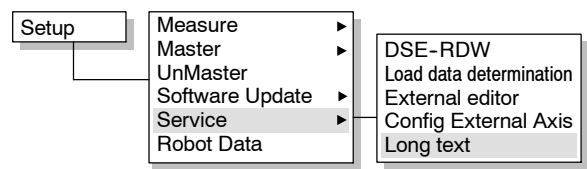
Close Accepts the changes made and closes the program window.

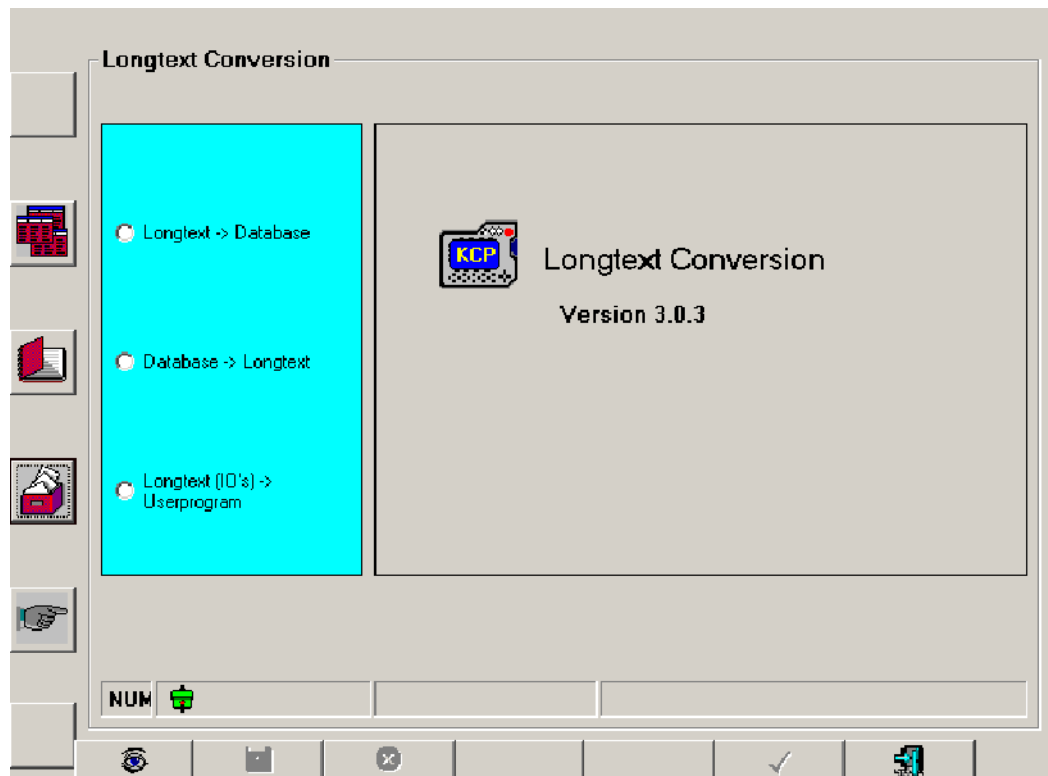
NAVIGATOR This softkey brings the Navigator to the foreground.

i If the file "CONFIGMON.INI" contains invalid entries, the attempt to load it will be canceled automatically by the system. An error message is not generated in this case.






2.2 Long text

This online program transfers an assignment list to the KUKA long text database. In this way, the long texts do not need to be re-entered manually for each robot after reinstallation. Long texts are displayed in the status windows for inputs/outputs, cyclical flags, flags, counters and timers.





The following information is displayed in the status line in conjunction with the long text conversion:

-  The connection to the database is active.
-  The database is being read.
-  The database is being written.
-  The database is being deleted.
-  An error has occurred.



The TAB key on the KCP can be used to toggle between the drives list, directory list and text input box. The "NUM" display must be deactivated for this.



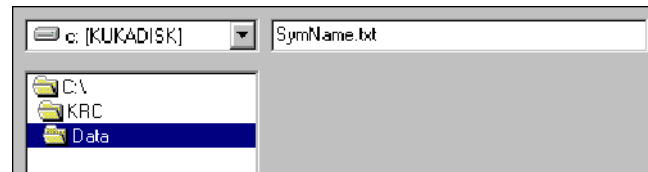
The status key "Long text -> Database" is used to load a text file and transfer it to the KUKA long text database.



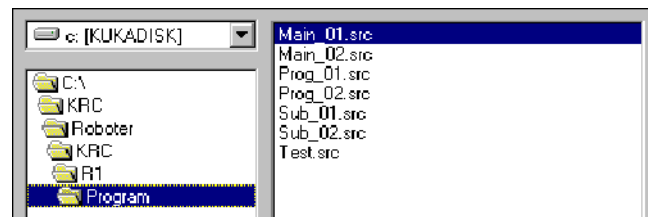
The option "Insert long text" leaves the entries already in the KUKA long text database intact and simply inserts the new entries. If this option is deactivated, the previous long text database is deleted completely.



The status key “Database -> Long text” creates a text file from the KUKA long text database. The path and the name of the file to be saved can be specified.



The status key “Longtext (IO's) -> Userprogram” updates the long texts in application programs.



The option “Select all files” applies the long text information to all application programs.



This softkey can be used to view or modify the KUKA long text database. The following options are then available:



This softkey enters the changes in the KUKA long text database.



The changes are not saved and the database display is closed.



The information is entered in the KUKA long text database or exported from it.



The online program is ended.

2.3 Monitoring

2.3.1 Control cabinet external fan

In order to increase the service life of the external fan, and also to reduce its power consumption and noise level, the external fan of the “KR C2” controller can be switched off by default. A number of different temperature values inside the control cabinet are monitored continuously. If one of these temperatures exceeds a certain value, the external fan is switched on for a defined duration.



In certain work environments, aggressive vapors are produced. In order to prevent these vapors from entering the cabinet, it is necessary to create an overpressure relative to the ambient pressure. For this, the external fan must be left to run continuously.

2.3.2 PC fan

This function checks the speed of the fan in the “KR C2” control cabinet PC against two values.

- If the speed of the fan falls below the first value, a message which cannot be acknowledged is displayed in the message window.
- If the speed falls below the second value, an error message is again generated and the robot is stopped.

The variable modification function allows you to view the current fan speed by entering the variable “\$PC_FANSPEED” in the input line. If the monitoring function has been deactivated, the value “-1” is displayed. If an Office PC is being used or there is no MFC2 present, the value “-2” is displayed.

2.3.3 Configurable output for hardware warnings

In the event of a hardware warning or “KR C2” hardware failure, a previously defined output is set.

This applies to PC fan speed monitoring, battery monitoring and motherboard temperature monitoring. If the actual value exceeds/falls below the relevant defined value, the predefined output is set.

The output to be set is defined in the file “C:\KRC\Roboter\KRC\Steu\MaDa\\$machine.dat.

```
SIGNAL $HW_WARNING $OUT[48]
```



This function is deactivated by default: “SIGNAL \$HW_WARNING **FALSE**”

2.3.4 Motor cable monitoring

For certain robot types with the “KR C2” controller, additional motor cables are required for axes 1 ... 3. The monitoring function is activated (TRUE) and deactivated (FALSE) via an entry in the file “C:\KRC\Roboter\KRC\R1\MaDa\\$machine.dat”.

```
BOOL $CABLE2_MON=FALSE
```

If the function is activated and the required cables are not connected, a message which cannot be acknowledged is displayed in the message window.



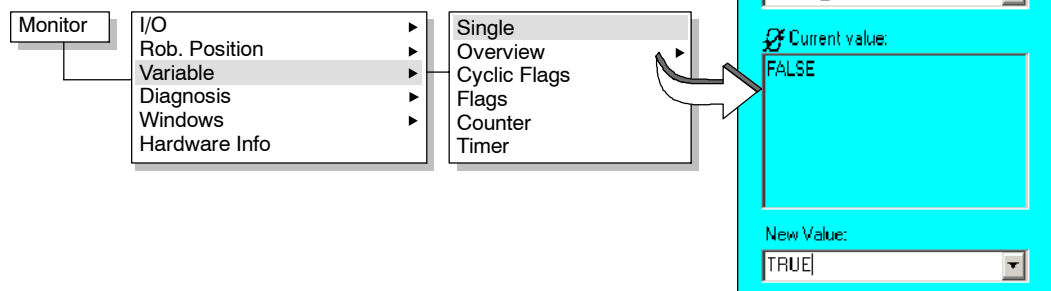
No liability will be accepted for any damage to the robot system resulting from missing additional motor cables!

2.4 Simulated inputs/outputs (I/O simulation)

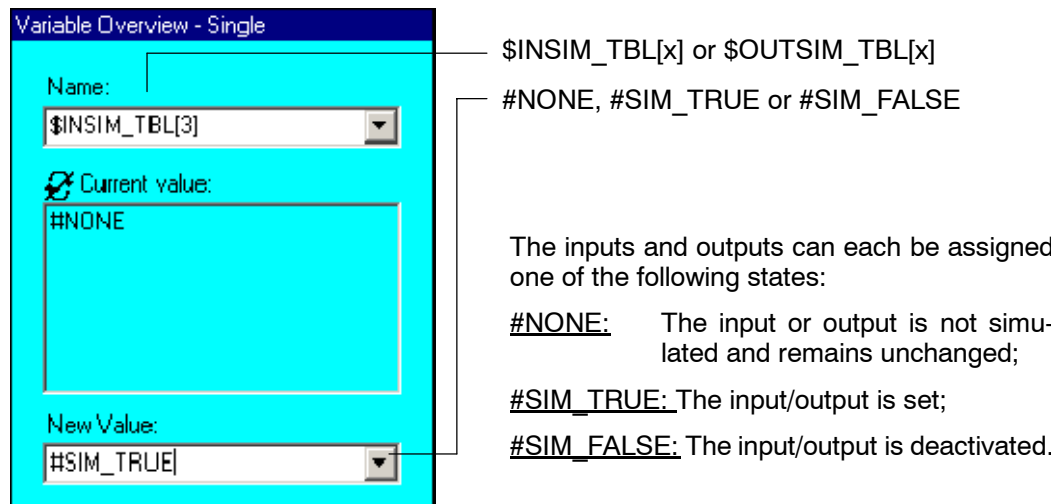
This function enables the simulation of certain inputs and outputs. If, for example, the input periphery is not yet available, the necessary inputs can be set simply to “TRUE” or “FALSE” by means of simulation. The same principle also applies to outputs.

2.4.1 Function

The simulation function is activated by means of the variable “\$IOSIM_OPT”. Open the variable correction function and change the value to “TRUE”.



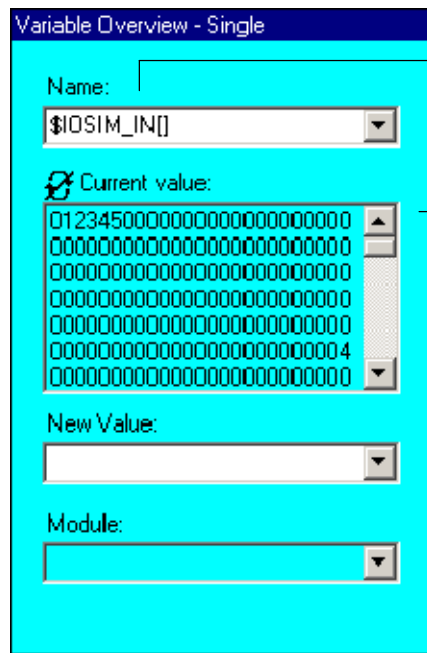
The simulation function is now activated, but the affected inputs/outputs must still be set. This is done using the variables “\$INSIM_TBL[x]” and “\$OUTSIM_TBL[x]”.



Some inputs/outputs, which are used by the system, may not be used. These are write-protected and cannot be changed.

Displaying arrays

The variables “\$IOSIM_IN[]” and “\$IOSIM_OUT[]” display all inputs/outputs in a line. The inputs and outputs are each assigned a number, the meaning of which is described below.



- \$IOSIM_IN[] or \$IOSIM_OUT[]
- Display of the assignment of all inputs/outputs
- The inputs and outputs can each take one of the following states:
 - 0 No signal (FALSE)
 - 1 Signal present (TRUE)
 - 2 Simulated signal set to "FALSE" (deactivated)
 - 3 Simulated signal set to "TRUE" (activated)
 - 4 System signal set to "FALSE" (deactivated)
 - 5 System signal set to "TRUE" (activated)



This assignment has the following appearance in the digital inputs status window:

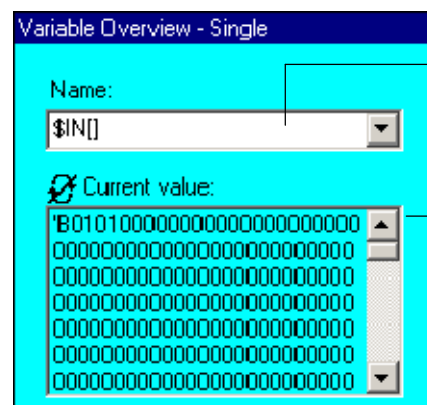
Inputs			
	SYS	SIM	Inputs
1			Input
2			Input
3		SIM	Input
4		SIM	Input
5	SYS		Input
6	SYS		Input
7			Input
8			Input
9			Input

- No signal at input 1 (FALSE)
- Signal at input 2 (TRUE)
- No signal at input 3 (FALSE)
- Signal at input 4 (TRUE)
- No signal at input 5 (FALSE)
- Signal at input 6 (TRUE)

The variables "\$IN[]" and "\$OUT[]" also display the assignment of the inputs/outputs in a line if no number is specified. No distinction is made between physical and simulated I/Os, only whether a signal is present or not.



The variables "\$IN[]" and "\$OUT[]" also display the assignment of the inputs/outputs in a line if no number is specified. No distinction is made between physical and simulated I/Os, only whether a signal is present or not.



- \$IN[] or \$OUT[]
- Assignment of the inputs/outputs
- The inputs and outputs can each take one of the following states:
 - 0 No physical signal (FALSE)
 - 1 Physical signal present (TRUE)

2.4.2 Options

The options described below are helpful, but not vital, for the simulation of inputs/outputs.

Enabling switch (\$OUT_NODRIVE)

To toggle between the outputs, it is normally necessary to hold down one of the enabling switches. This can be bypassed by setting the variable “\$OUT_NODRIVE” to “TRUE”.

Operating mode Automatic External (\$IOBLK_EXT)

In connection with \$OUT_NODRIVE = TRUE, the variable “\$IOBLK_EXT” allows you to set outputs in Automatic External mode (#EXT), which is not normally possible. This is done by setting “\$IOBLK_EXT” to “FALSE”.

2.4.3 Variables used

Variable	Range of values	Meaning
\$IOSIM_OPT	TRUE FALSE	Simulation active Simulation inactive
\$INSIM_TBL[x] x = 1 ... 1024	#NONE #SIM_TRUE #SIM_FALSE	Input is not simulated Input is activated Input is deactivated
\$OUTSIM_TBL[x] x = 1 ... 1024	#NONE #SIM_TRUE #SIM_FALSE	Output is not simulated Output is activated Output is deactivated
\$IOBLK_EXT *1	TRUE FALSE	Outputs cannot be set in #EXT Outputs can be set
\$OUT_NODRIVE	TRUE FALSE	Enabling switch need not be pressed in order to toggle between outputs Enabling switch must be pressed
\$IOSIM_IN[]		Displays all inputs
\$IOSIM_OUT[]		Displays all outputs
\$IN[x] x = 1 ... 1026	TRUE FALSE	Input set Input not set
\$IN[]		All inputs
\$OUT[x] x = 1 ... 1024	TRUE FALSE	Output set Output not set
\$OUT[]		All outputs
*1 Only in connection with \$OUT_NODRIVE = TRUE		



If an output is simulated, it cannot, during simulation, be toggled in the variable correction function (“\$OUT[x]”) or in the status window. Simulation for this output must first be deactivated.

2.5 5 home positions

In addition to the home position, the user can define 5 more home positions. There is then a total of 6 different positions available.

In the same way as for the previous home position “\$H_POS”, where the variable “\$IN_HOME” is set to “TRUE” when the position is reached, the variables “\$IN_HOME1” ... “\$IN_HOME5” are set to “TRUE” for the positions “\$AXIS_HOME[1]” ... “\$AXIS_HOME[5]”. As with “\$H_POS”, the tolerance band defined in the variable “\$H_POS_TOL” is also valid for the 5 additional home positions. If all axes are situated within this tolerance window, the corresponding variable is set.

2.5.1 “\R1\MaDa\\$machine.dat” file

The coordinates of axes 1...6 or external axes E1...E6 are entered after the highlighted sections “\$AXIS_HOME[x]”.

```
&PARAM VERSION=3.4.0
&REL 4
DEFDAT $MACHINE PUBLIC
...
E6AXIS $H_POS={A1 0.0,A2 -90.0,A3 90.0,A4 0.0,A5 0.0,A6 0.0,E1
0.0,E2 0.0,E3 0.0,E4 0.0,E5 0.0,E6 0.0}
E6AXIS $AXIS_HOME[5]
$AXIS_HOME[1]={A1 0.0,A2 -90.0,A3 90.0,A4 0.0,A5 0.0,A6 0.0,E1
0.0,E2 0.0,E3 0.0,E4 0.0,E5 0.0,E6 0.0}
$AXIS_HOME[2]={A1 0.0,A2 -90.0,A3 90.0,A4 0.0,A5 0.0,A6 0.0,E1
0.0,E2 0.0,E3 0.0,E4 0.0,E5 0.0,E6 0.0}
$AXIS_HOME[3]={A1 0.0,A2 -90.0,A3 90.0,A4 0.0,A5 0.0,A6 0.0,E1
0.0,E2 0.0,E3 0.0,E4 0.0,E5 0.0,E6 0.0}
$AXIS_HOME[4]={A1 0.0,A2 -90.0,A3 90.0,A4 0.0,A5 0.0,A6 0.0,E1
0.0,E2 0.0,E3 0.0,E4 0.0,E5 0.0,E6 0.0}
$AXIS_HOME[5]={A1 0.0,A2 -90.0,A3 90.0,A4 0.0,A5 0.0,A6 0.0,E1
0.0,E2 0.0,E3 0.0,E4 0.0,E5 0.0,E6 0.0}
...
```



This file is located by default in the directory “C:\KRC\Roboter\KRC\R1\ MaDa”.

2.5.2 “\Steu\MaDa\\$machine.dat” file

A certain output is assigned to each of the variables “\$IN_HOME1” ... “\$IN_HOME5” in the file “\$machine.dat”.

```
&PARAM VERSION=3.4.0
DEFDAT $MACHINE PUBLIC
...
SIGNAL $IN_HOME $OUT[1000] ;ROB IN HOMEPOSITION
...
SIGNAL $IN_HOME1 $OUT[977]
SIGNAL $IN_HOME2 $OUT[978]
SIGNAL $IN_HOME3 $OUT[979]
SIGNAL $IN_HOME4 $OUT[980]
SIGNAL $IN_HOME5 $OUT[981]
...
ENDDAT
```



This file is located by default in the directory “C:\KRC\Roboter\KRC\ Steu\MaDa”.

2.6 Workspace monitoring

Up to eight cubic or axis-specific workspaces can be monitored automatically. These workspaces may also be overlapped to produce more complex shapes. If one of these defined workspaces is violated, the controller sets a predefined output. The output signal provided can then be further processed by the KRL program or by an external host computer. The robot can also be stopped and an error message generated.



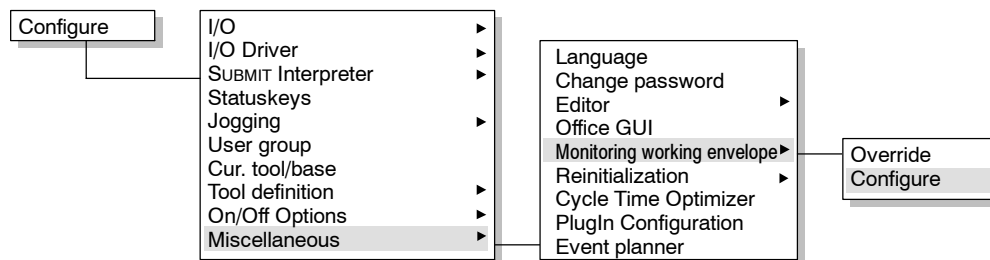
The braking distance may vary according to the velocity of the robot with the result that the robot may enter or leave the workspace before coming to a standstill! For this reason, it is necessary to leave enough space between the workspaces and the permitted/prohibited areas!

2.6.1 Cartesian workspace monitoring

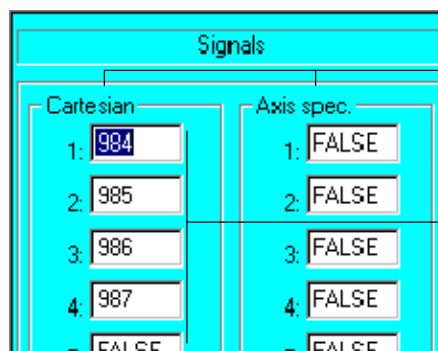
Cuboid workspaces can be defined here. Depending on the definition, the robot TCP may either not leave these areas or not enter them.

2.6.1.1 Configure

Cuboid workspaces are defined at Expert level using a number of status windows.



Signal An output for each workspace can be defined in the status window "Signal". The defined output is set if a workspace is violated.



The left-hand section of the window is for Cartesian workspace monitoring, while the right-hand section is for axis-specific workspace monitoring.

The outputs to be set are entered in the corresponding input boxes.

If no output is to be set when a workspace is violated, the corresponding signal must be set to "FALSE".

The KRL signal declarations defining the individual outputs are in turn defined in the machine data:

```

...
SIGNAL $WORKSTATE1 $OUT[984]
SIGNAL $WORKSTATE2 $OUT[985]
SIGNAL $WORKSTATE3 $OUT[986]
SIGNAL $WORKSTATE4 $OUT[987]
SIGNAL $WORKSTATE5 FALSE
SIGNAL $WORKSTATE6 FALSE
  
```

SIGNAL \$WORKSTATE7 FALSE
 SIGNAL \$WORKSTATE8 FALSE
 ...



If the signal corresponding to a workspace has been set to "FALSE", the component "\$WORKSPACE[n].STATE" can be used to read whether or not the workspace has been violated.

Cartesian

The softkey "Cartesian" takes you to the status window for definition of cubic workspaces.

Workspace name and number

Frame defining the origin and orientation of the workspace relative to the world coordinate system

Dimensions of the workspace relative to the origin defined by {X, Y, Z, A, B, C}

Workspace monitoring mode

The corresponding KRL variable "\$WORKSPACE[n]" has the following structure:

```
$WORKSPACE[n]={X 500, Y 500, Z 1000, A 0, B 0, C 0, X1 100, Y1 100, Z1 100, X2 -100, Y2 -100, Z2 -100, MODE #OUTSIDE_STOP, STATE FALSE}
```

The meaning of the components of the structure "\$WORKSPACE[n]":	
X, Y, Z:	Origin of the workspace relative to the world coordinate system
A, B, C:	Orientation of the workspace relative to the world coordinate system
X1, Y1, Z1:	Determines Δx_1 , Δy_1 , Δz_1 in relation to the origin X, Y, Z, A, B, C and opens a cuboid
X2, Y2, Z2:	Determines Δx_2 , Δy_2 , Δz_2 relative to the origin X, Y, Z, A, B, C and increases or decreases the size of the cuboid
Options for "MODE" settings:	
#OFF	Monitoring of the workspace in question is switched off.
#INSIDE	The preset output is set if the reference point (TCP) of the tool/workpiece is <u>inside</u> the workspace.
#OUTSIDE	The preset output is set if the reference point (TCP) of the tool/workpiece is <u>outside</u> the workspace.
#INSIDE_STOP	The preset output is set if the reference point (TCP) of the tool/workpiece or the wrist root point is <u>inside</u> the workspace. Furthermore, the robot is stopped and error message 114 "Work envelope no. n violated" is generated.

#OUTSIDE_STOP	The preset output is set if the reference point (TCP) of the tool/workpiece is <u>outside</u> the workspace. Furthermore, the robot is stopped and error message 114 "Work envelope no. <i>n</i> violated" is generated.
Possible states for "STATE":	
TRUE *1	The workspace has been violated
FALSE *1	The workspace has not been violated
*1 The value is merely displayed and cannot be changed	



If a workspace is violated in the modes "INSIDE_STOP" or "OUTSIDE_STOP", robot motion can only be resumed if the affected work envelope monitoring is switched off or overridden.

Variables for use with the variable correction function or in KRL programs

Variable	Meaning	Range of values
\$WORKSPACE[n] <i>n</i> = 1 ... 8	Definition of the corresponding Cartesian workspace	
\$WORKSPACE[n].MODE <i>n</i> = 1 ... 8	Definition of the type of monitoring for a defined workspace	#OFF #INSIDE #OUTSIDE #INSIDE_STOP #OUTSIDE_STOP
\$WORKSPACE[n].STATE <i>n</i> = 1 ... 8	Read-only variable indicating whether a workspace has been violated (TRUE) or not (FALSE)	TRUE FALSE
\$WORKSPACE_NAME[n] <i>n</i> = 1 ... 8	Name for a defined workspace	Max. 24 characters
\$WBOXDISABLE	Activation (TRUE) or deactivation (FALSE) of the workspace monitoring override function	TRUE FALSE



Modification of the "\$WORKSPACE" variables triggers an advance run stop.

If "\$TOOL" is invalid and at least one work envelope is active, error message 112 appears in the message window:

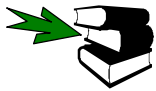
"Invalid \$Tool: Working envelope surveillance not possible"

Set outputs are reset and any displayed messages are deleted.



Using incorrect "\$TOOL" data can have unpredictable consequences!

Axis spec. | The softkey "Axis spec." takes you to the status window for definition of axis-specific workspaces.



Details can be found in Section 2.6.2.

Change

Changes are saved by pressing this softkey.



Workspaces can also be defined, or switched on and off, in *.SRC files. The values specified here are automatically entered in the file "\$CUSTOM.DAT" and are available again the next time the controller is started.

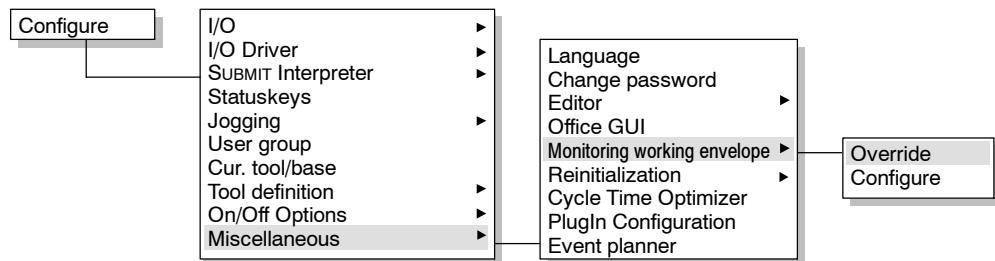
Workspace settings can also be changed by modifying variables.

Close

Closes the configuration menu. All changes are lost unless they have been saved using the "Change" softkey.

2.6.1.2 Override

Overriding the workspace monitoring via menu



This function makes it possible to move the robot back out of the violated workspace.



This is only possible in the operating mode TEST (T1).

When a workspace is violated, status message 114 appears:

"Work envelope no. *n* violated"

If the work envelope monitoring is then overridden, this message is replaced by status message 115:

"Drive free work envelope *n*"

This message is deleted once the violated workspace has been left.

Deactivating workspace monitoring via the variable correction function or in a KRL program

Alter the value of the "MODE" component for the workspace concerned, e.g.

"\$WORKSPACE[1].MODE" to "#OFF".

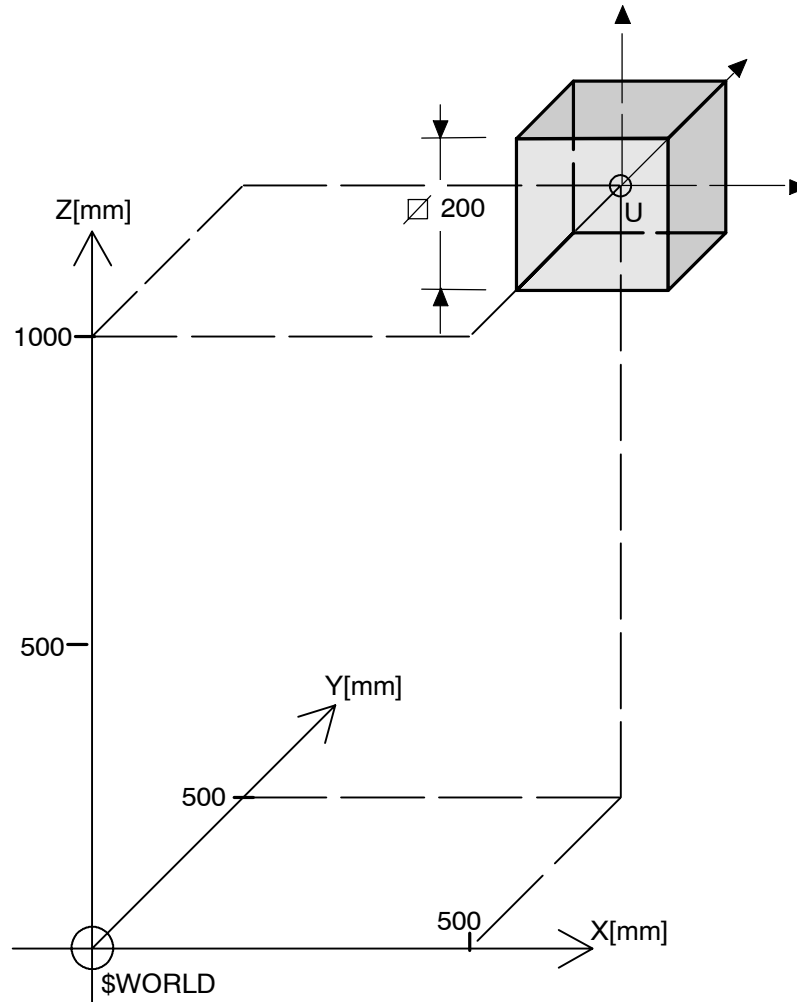


Monitoring of the affected workspace remains switched off until the component is reset to a value other than "#OFF".

2.6.1.3 Examples



A cubic workspace with sides 200 mm long is to be monitored. The center point is to be situated at $X = 500$ mm, $Y = 500$ mm und $Z = 1000$ mm. The angles A, B and C have the value "0".

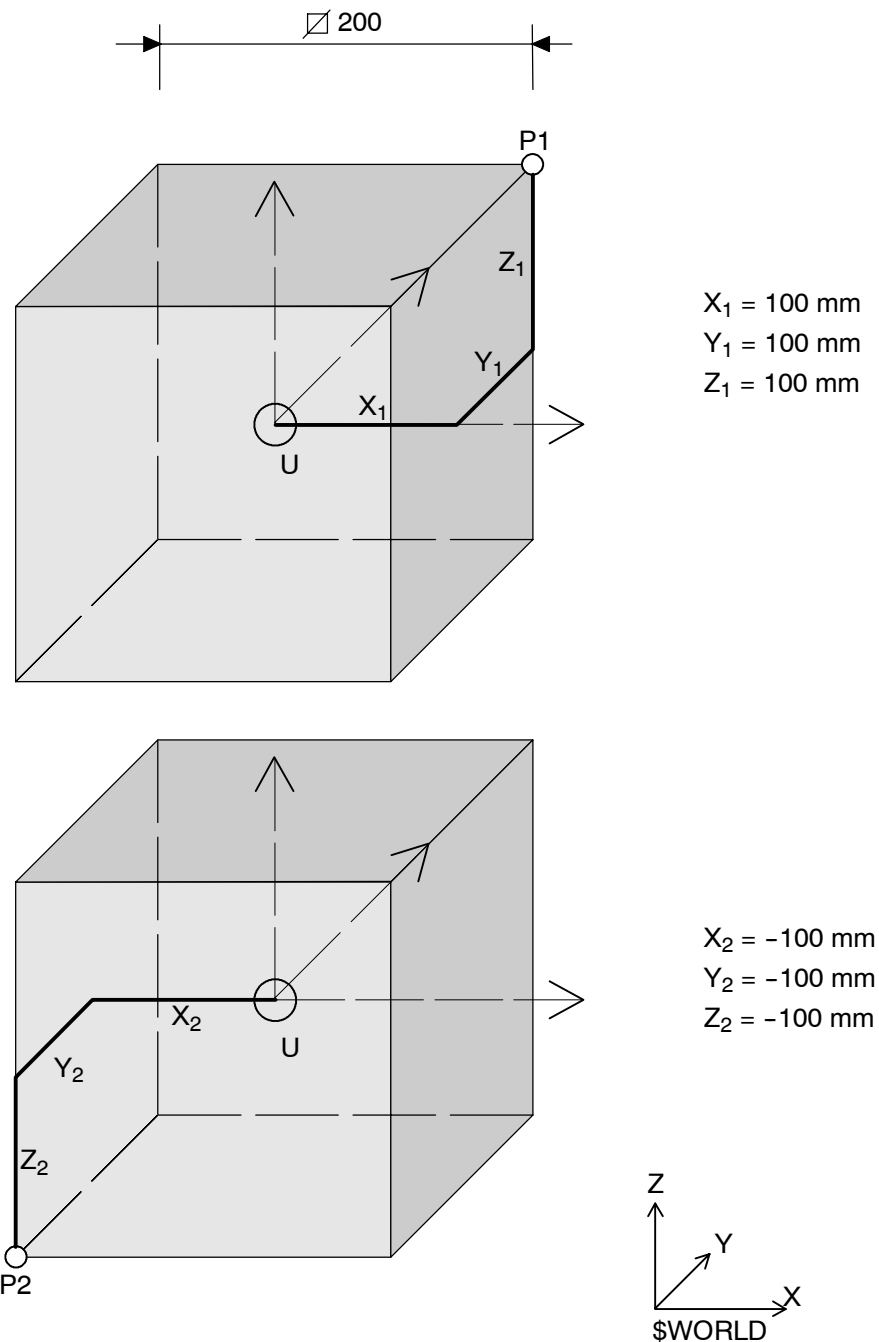


The origin of the workspace (frame "U") relative to the world coordinate system is defined as follows in the variable assignment:

```
$WORKSPACE[n]={X 500, Y 500, Z 1000, A 0, B 0, C 0, X1 100,
Y1 100, Z1 100, X2 -100, Y2 -100, Z2 -100, MODE #INSIDE,
STATE FALSE}
```

2 Configuring the system, Expert (continued)

The position and orientation of the workspace is defined by frame "U". Its size is defined by two points on opposite sides of the origin "U".



Points "P1" and "P2" can be found in the parameter line in the following position:

```
$WORKSPACE[1]={X 500, Y 500, Z 1000, A 0, B 0, C 0, x1 100,  
y1 100, z1 100, x2 -100, y2 -100, z2 -100, MODE #INSIDE,  
STATE FALSE}
```

The lengths of the sides of the cuboid are calculated from $|X_1 - X_2|$, $|Y_1 - Y_2|$ and $|Z_1 - Z_2|$.

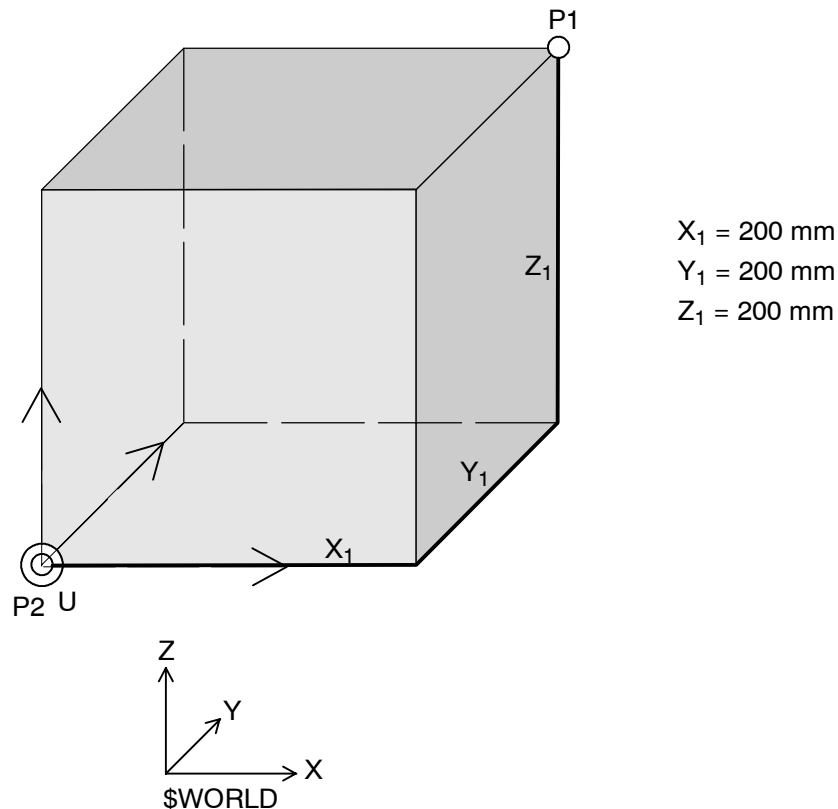
The assigned output is to be set as soon as the reference point of the tool or workpiece enters the workspace. The penultimate entry in the status window or KRL command is used for this.

Configuration

```
$WORKSPACE[n]={X 500, Y 500, Z 1000, A 0, B 0, C 0, X1 100,
Y1 100, Z1 100, X2 -100, Y2 -100, Z2 -100, MODE #INSIDE,
STATE FALSE}
```



If point "P2" is situated at the origin of the workspace, only the coordinates of "P1" need to be determined.



The assigned output is to be set if the reference point of the tool or workpiece is outside the workspace. At the same time, the robot should stop and generate an error message.

The corresponding instruction reads as follows:

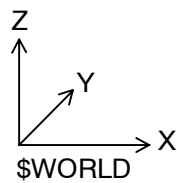
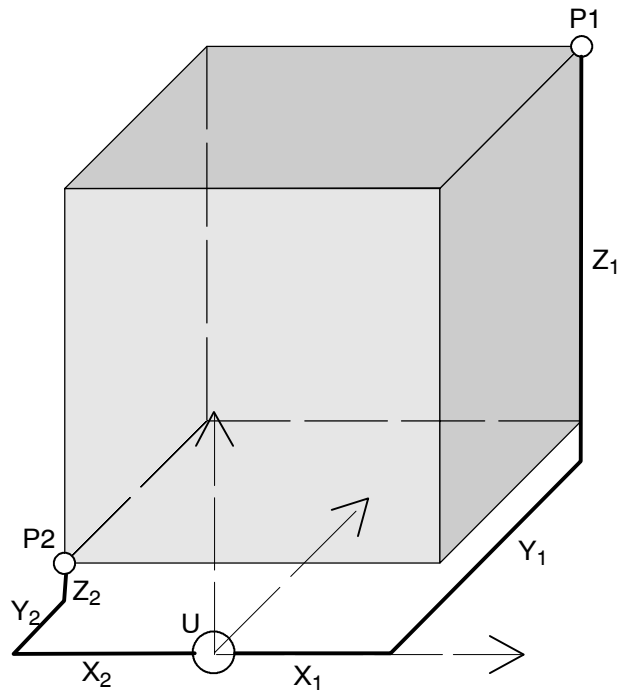
```
$WORKSPACE[2]={X 400, Y 400, Z 900, A 0, B 0, C 0, X1 200, Y1
200, Z1 200, X2 0, Y2 0, Z2 0, MODE #OUTSIDE_STOP, STATE
FALSE}
```



This is the same workspace as in the previous example. The position and orientation remain identical, but the origin is situated at one of the corners of the cuboid.



If one of the point coordinate pairs “X₁” and “X₂”, “Y₁” and “Y₂” or “Z₁” and “Z₂” has the same sign, the origin “U” lies outside the workspace.



X ₁ = 100 mm	X ₂ = -100 mm
Y ₁ = 260 mm	Y ₂ = 60 mm
Z ₁ = 220 mm	Z ₂ = 20 mm

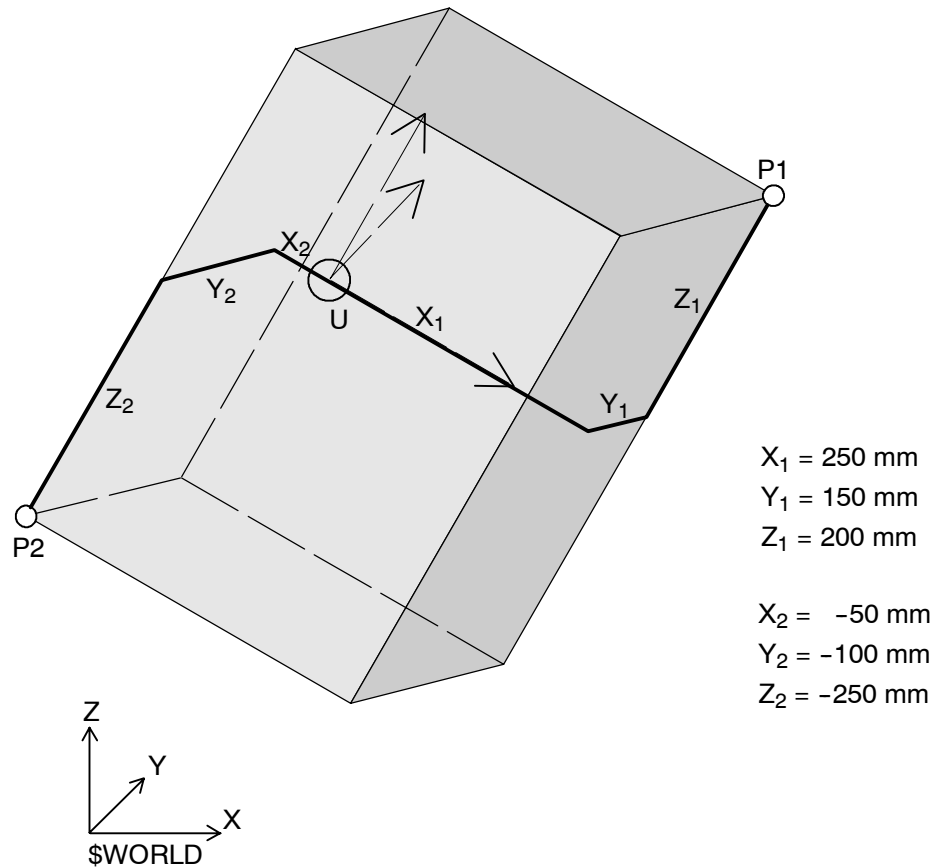
Here, the assigned output is to be set if the reference point of the tool or workpiece is outside the workspace. This does not cause the robot to stop.

The corresponding instruction reads as follows:

```
$WORKSPACE[3]={X 500, Y 500, Z 2000, A 0, B 0, C 0, X1 100,  
Y1 260, Z1 220, X2 -100, Y2 60, Z2 20, MODE #OUTSIDE, STATE  
FALSE}
```



The workspace in this example has the dimensions $x = 300$ mm, $y = 250$ mm and $z = 450$ mm. In relation to the world coordinate system, it is rotated about the Y-axis by 30 degrees. The origin "U" is not situated in the center of the cuboid.



The assigned output is to be set again if the reference point of the tool or workpiece enters the workspace. At the same time, the robot should stop and generate an error message.

The corresponding instruction reads as follows:

```

WORKSPACE[4] = {X 500, Y 500, Z 2000, A 0, B 30, C 0, X1 250,
Y1 150, Z1 200, X2 -50, Y2 -100, Z2 -250, MODE #INSIDE_STOP,
STATE FALSE}
```

2.6.2 Axis-specific workspace monitoring

The areas defined by the software limit switches can be restricted yet further using this function in order to protect the robot, tool or workpiece. The permissible range for an axis is then directly dependent on the current positions of the other axes.

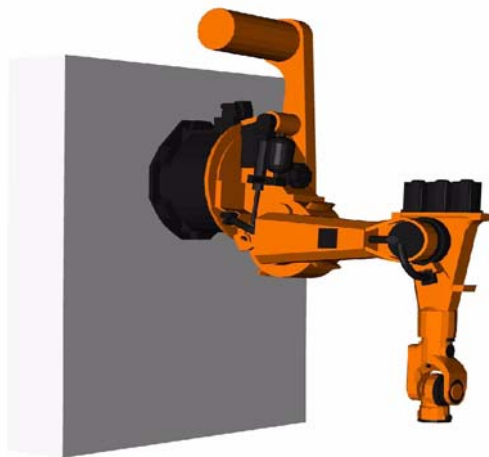
Depending on the definition, a predefined output is set when a workspace is either left or violated. Alternatively, the robot can be stopped and a message displayed in the message window.



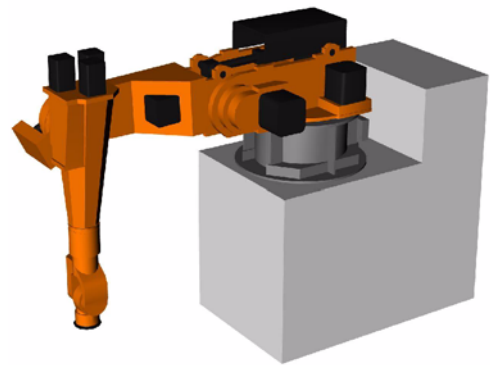
Ramp-down braking occurs if a workspace is violated in jog mode; in other modes dynamic braking is triggered.

2.6.2.1 Functional principle

With some systems, it may be sensible to restrict the work envelope of the robot further in order to avoid damage to the robot or periphery. This is particularly applicable for the following robot types:



Wall-mounted robot

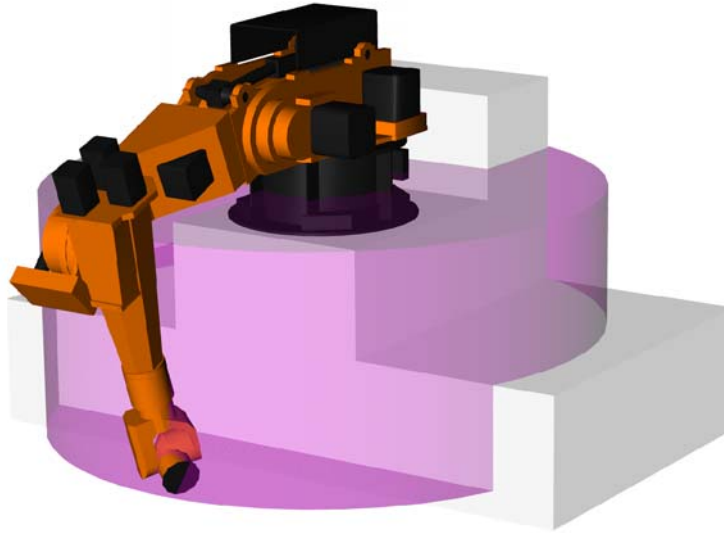


Shelf-mounted robot

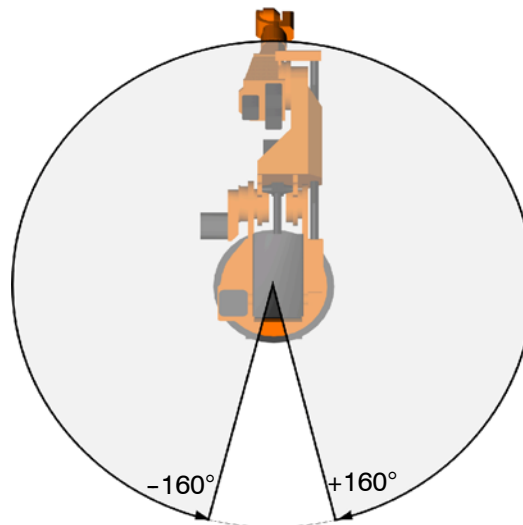


Palletizing robot

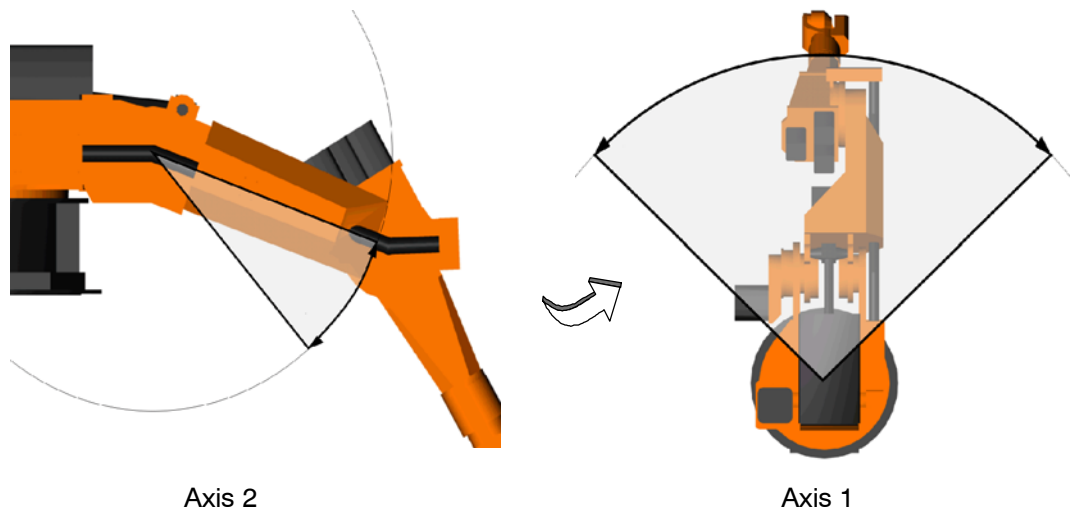
If, for example, there are supply cables, built-on accessories or other devices situated in the work envelope of the robot, these may be damaged without the corresponding workspace monitoring.



The maximum possible action range of a robot is limited by the software limit switches, the values of which depend on the particular robot type. Axis 1 of the model used in the example can move through a maximum of ± 160 degrees.



Axis-specific workspace monitoring can be used, for example, to restrict the motion of axis 1 to a greater or lesser extent dependent on the position of axis 2.



If axis 2 is situated in the shaded area, axis 1 must not leave its shaded area.

The required maximum and minimum values must be specified for each of the axes to be monitored.

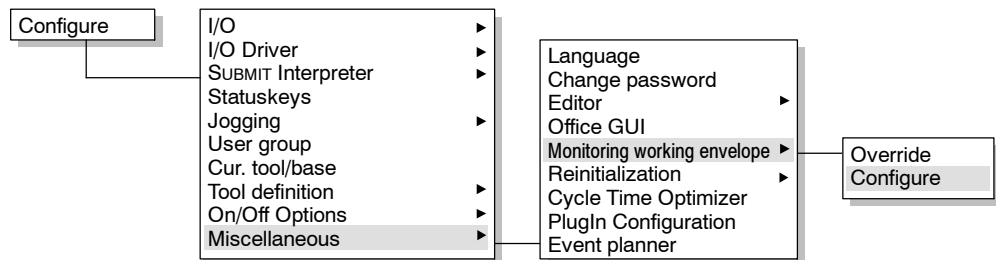


Certain machines are supplied by KUKA with predefined values for axis-specific workspace monitoring. **These values must not be changed as the robot may otherwise be damaged.**

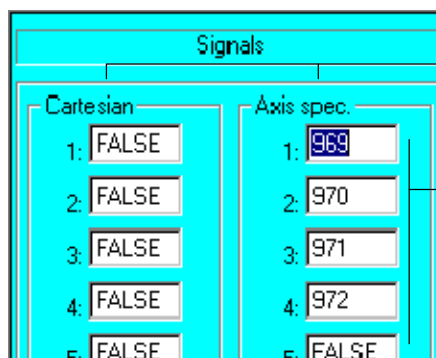
To avoid damage, take the tools and workpieces used into consideration when entering values.

2.6.2.2 Configure

Workspaces are defined at Expert level using a number of status windows.



Signal An output for each workspace can be defined in the status window "Signal". The defined output is set if a workspace is violated.



The left-hand section of the window is for Cartesian workspace monitoring, while the right-hand section is for axis-specific workspace monitoring.

The outputs to be set are entered in the corresponding input boxes.

If no output is to be set when a workspace is violated, the corresponding signal must be set to "FALSE".

The KRL signal declarations defining the individual outputs are in turn defined in the machine data:

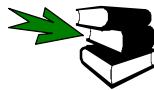
```

...
SIGNAL $AXWORKSTATE1 $OUT[969]
SIGNAL $AXWORKSTATE2 $OUT[970]
SIGNAL $AXWORKSTATE3 $OUT[971]
SIGNAL $AXWORKSTATE4 $OUT[972]
SIGNAL $AXWORKSTATE5 FALSE
SIGNAL $AXWORKSTATE6 FALSE
SIGNAL $AXWORKSTATE7 FALSE
SIGNAL $AXWORKSTATE8 FALSE
...
    
```



If the signal corresponding to a workspace has been set to “FALSE”, the component “\$AXWORKSPACE[n].STATE” can be used to read whether or not the workspace has been violated.

Cartesian The softkey “Cartesian” takes you to the status window for definition of cubic workspaces.



Details can be found in Section 2.6.1.

Axis spec. The softkey “Axis spec.” takes you to the status window for definition of axis-specific workspaces.

Axis specific workspaces					
No.	Name: AXWORKSPACE_NAME 1				
Axis	Min	Max	E1	E2	E3
A1:	-90.00	90.00	0.00	0.00	0.00
A2:	-25.00	3.00	0.00	0.00	0.00
A3:	0.00	0.00	0.00	0.00	0.00
A4:	0.00	0.00	0.00	0.00	0.00
A5:	0.00	0.00	0.00	0.00	0.00
A6:	0.00	0.00	0.00	0.00	0.00

Mode: **INSIDE_STOP**

Workspace name and number

Specification of the maximum and minimum values for individual axis angles, valid for both standard and external axes

Monitoring mode



If the max./min. boxes for an axis contain the default value “0.00”, this axis is not monitored, irrespective of the mode that is set. This applies for both standard axes (A1...A6) and external axes (E1...E6).

The KRL variable “\$AXWORKSPACE[n]” has the following structure:

```

$AXWORKSPACE[1]={A1_N -90.0,A1_P 90.0, A2_N -25.0,A2_P 3.0,A3_N
0.0,A3_P 0.0,A4_N 0.0,A4_P 0.0,A5_N 0.0,A5_P 0.0,A6_N 0.0,A6_P
0.0,E1_N 0.0,E1_P 0.0,E2_N 0.0,E2_P 0.0,E3_N 0.0,E3_P 0.0,E4_N
0.0,E4_P 0.0,E5_N 0.0,E5_P 0.0,E6_N 0.0,E6_P 0.0,MODE #INSIDE_STOP,
STATE FALSE}
    
```

The meaning of the components of the structure “\$AXWORKSPACE[n]”:	
A1_N ... A6_N	The minimum values of the individual standard axes
A1_P ... A6_P	The maximum values of the individual standard axes

E1_N ... E6_N	The minimum values of the configured external axes
E1_P ... E6_P	The maximum values of the configured external axes
Options for “MODE” settings:	
#OFF	Monitoring of the workspace in question is switched off.
#INSIDE	The specified output is set if all axes are located <u>inside</u> the defined workspace.
#OUTSIDE	The specified output is set if all axes are located <u>outside</u> the defined workspace.
#INSIDE_STOP	The specified output is set if all axes are located <u>inside</u> the defined workspace. The robot is also stopped and an error message is generated.
#OUTSIDE_STOP	The specified output is set if all axes are located <u>outside</u> the defined workspace. The robot is also stopped and an error message is generated.
Possible states for “STATE”:	
TRUE *1	The workspace has been violated
FALSE *1	The workspace has not been violated
*1 The value is merely displayed and cannot be changed	



If a workspace is violated in the modes “INSIDE_STOP” or “OUTSIDE_STOP”, robot motion can only be resumed if the affected work envelope monitoring is switched off or overridden.

Variables for use with the variable correction function or in KRL programs

Variable	Meaning	Range of values
\$AXWORKSPACE[n] <i>n</i> = 1 ... 8	Definition of the corresponding axis-specific workspace	
\$AXWORKSPACE[n].MODE <i>n</i> = 1 ... 8	Definition of the type of monitoring for a defined workspace	#OFF #INSIDE #OUTSIDE #INSIDE_STOP #OUTSIDE_STOP
\$AXWORKSPACE[n].STATE <i>n</i> = 1 ... 8	Read-only variable indicating whether a workspace has been violated (TRUE) or not (FALSE)	TRUE FALSE
\$AXWORKSPACE_NAME[n] <i>n</i> = 1 ... 8	Name for a defined workspace	Max. 24 characters
\$WBOXDISABLE	Activation (TRUE) or deactivation (FALSE) of the workspace monitoring override function	TRUE FALSE



Modification of the “\$AXWORKSPACE” variables triggers an advance run stop.

Change

Changes are saved by pressing this softkey.

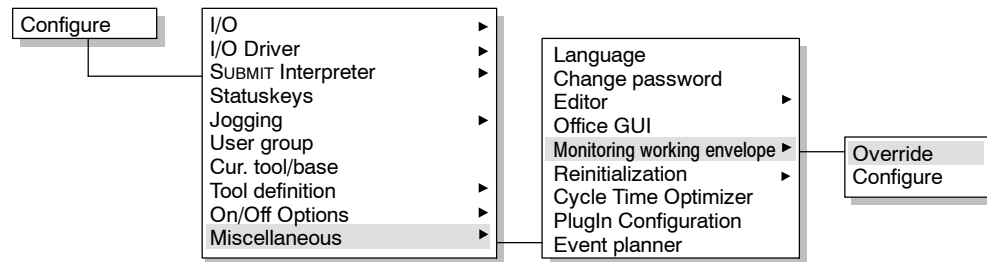


Workspaces can also be defined, or switched on and off, in *.SRC files. The values specified here are automatically entered in the file "\$MACHINE.DAT" and are available again the next time the controller is started.

Workspace settings can also be changed by modifying variables.

Close The function is terminated using the softkey "Close". All changes are lost unless they have been saved using the "Change" softkey.

2.6.2.3 Override



This function makes it possible to move the robot back out of the violated workspace.



This is only possible in the operating mode TEST (T1).

When a workspace is violated, the following status message appears:

"Axis-specific work envelope no. *n* violated"

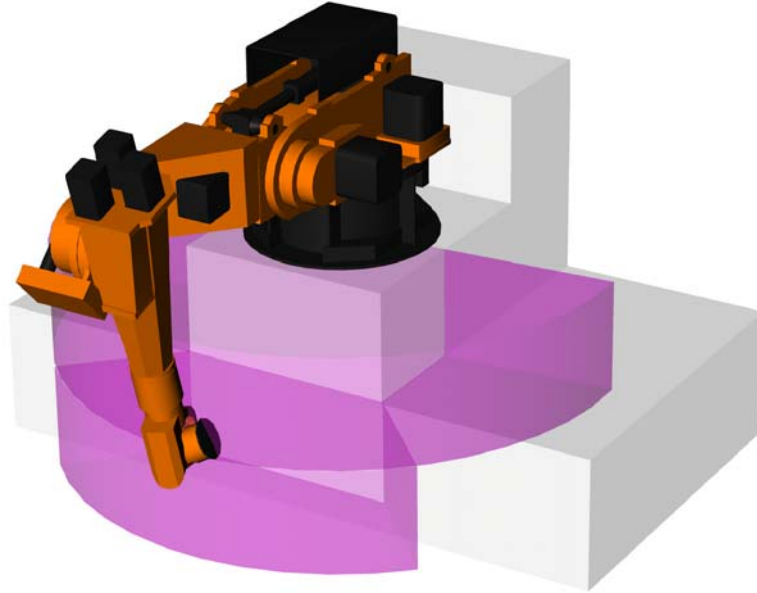
If the work envelope monitoring is then overridden, this message is replaced by the status message:

"Drive free axis-specific work envelope number *n*"

This message is deleted once the violated workspace has been left.

2.6.2.4 Example

In this example, a workspace is defined within which the robot may move. This workspace should have the following appearance:



To this end, several axis-specific workspaces must be defined. In the example, a total of 4 workspaces are required which must not be violated by the robot.



Above/below the shaded areas, the settings of the software limit switches are valid.

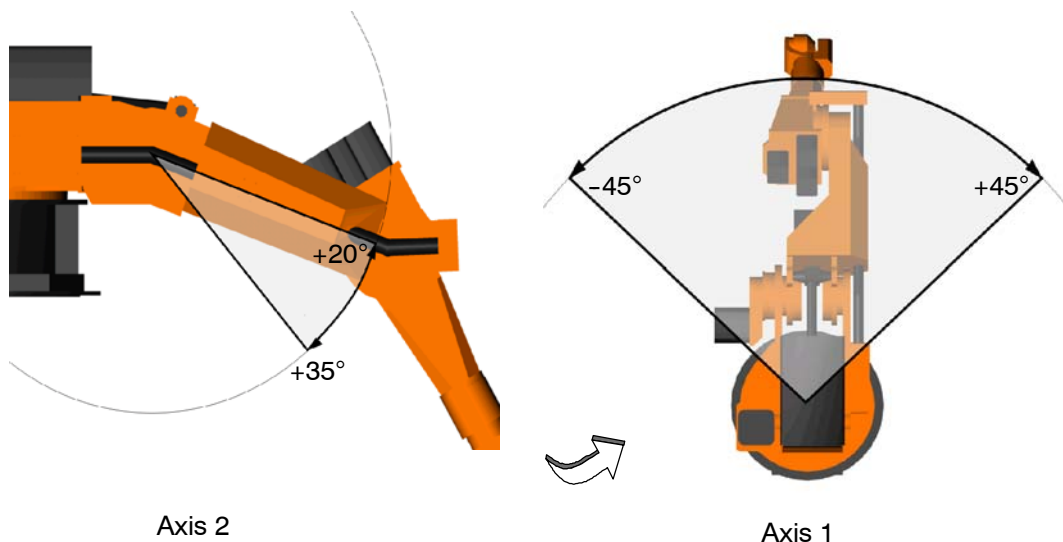


If an area is to be specified that is made up of a number of different workspaces, it is sensible to define the non-permitted areas and disable these using the MODE "#INSIDESTOP". Otherwise, the monitoring is triggered as soon as one of the defined workspaces is left.

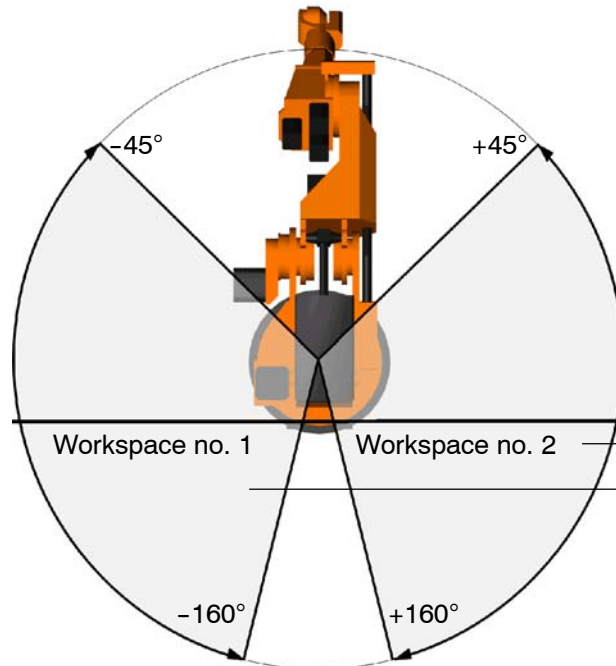
Workspaces 1 + 2



Axis 1 of a shelf-mounted robot is to be restricted as soon as axis 2 is situated in the area between +20 and +35 degrees.



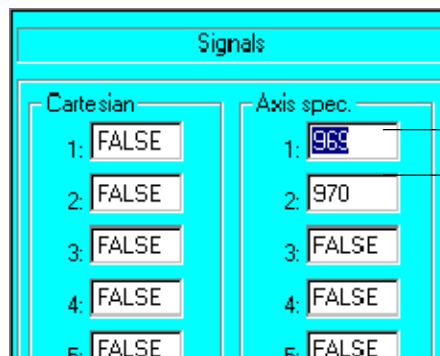
For this, two areas are defined for the first axis into which the robot may not move.



The robot may not enter these areas as long as axis 2 is situated in the area between 20 and 35 degrees.

The entries required for this in the status windows are as follows:

Signal



The individual outputs are set in accordance with the MODE setting.

The signals assigned to the workspaces are stored in the file "KRC\Steu\Mada\\$Machine.dat" and are normally defined via the user interface.

```

...
SIGNAL $AXWORKSTATE1 $OUT[969]
SIGNAL $AXWORKSTATE2 $OUT[970]
SIGNAL $AXWORKSTATE3 FALSE
SIGNAL $AXWORKSTATE4 FALSE
SIGNAL $AXWORKSTATE5 FALSE
SIGNAL $AXWORKSTATE6 FALSE
SIGNAL $AXWORKSTATE7 FALSE
SIGNAL $AXWORKSTATE8 FALSE
...

```

Axis-specific

The first workspace is defined as follows:

Axis	Min	Max	E1	E2	E3	E4	E5	E6
A1:	-160.00	-45.00	0.00	0.00	0.00	0.00	0.00	0.00
A2:	20.00	35.00	0.00	0.00	0.00	0.00	0.00	0.00
A3:	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
A4:	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
A5:	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
A6:	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00

Mode:

Axis-specific workspace no. 1 with the name "AXWORKSPACE_DOWN 1"

Angle specifications for axes 1 and 2, the other axes are not monitored

The robot is stopped and the predefined output is set as soon as axes 1 and 2 are situated in the defined workspace at the same time.

The second workspace:

Axis	Min	Max	E1	E2	E3	E4	E5	E6
A1:	45.00	160.00	0.00	0.00	0.00	0.00	0.00	0.00
A2:	20.00	35.00	0.00	0.00	0.00	0.00	0.00	0.00
A3:	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
A4:	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
A5:	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
A6:	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00

Mode:

Axis-specific workspace no. 2 with the name "AXWORKSPACE_DOWN 2"

Angle specifications for axes 1 and 2, the other axes are not monitored

The robot is stopped and the predefined output is set as soon as axes 1 and 2 are situated in the defined workspace at the same time.

The corresponding KRL instructions read as follows:

Name of the workspaces:

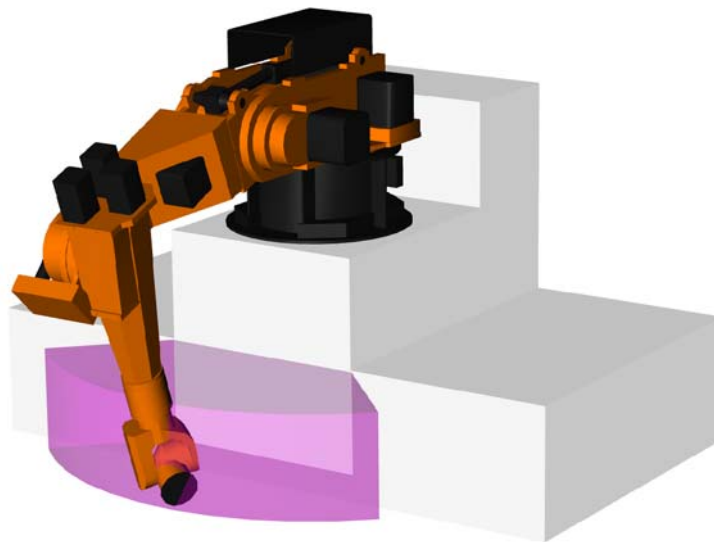
```
$AXWORKSPACE_NAME1[ ]="AXWORKSPACE_DOWN 1"
$AXWORKSPACE_NAME2[ ]="AXWORKSPACE_DOWN 2"
```

Axes to monitor:

```
$AXWORKSPACE[1]={A1_N -160.0,A1_P -45.0, A2_N 20.0,A2_P 35.0,A3_N
0.0,A3_P 0.0,A4_N 0.0,A4_P 0.0,A5_N 0.0,A5_P 0.0,A6_N 0.0,A6_P
0.0,E1_N 0.0,E1_P 0.0,E2_N 0.0,E2_P 0.0,E3_N 0.0,E3_P 0.0,E4_N
0.0,E4_P 0.0,E5_N 0.0,E5_P 0.0,E6_N 0.0,E6_P 0.0,MODE #INSIDE_STOP,
STATE FALSE}
```

```
$AXWORKSPACE[2]={A1_N 45.0,A1_P 160.0, A2_N 20.0,A2_P 35.0,A3_N  
0.0,A3_P 0.0,A4_N 0.0,A4_P 0.0,A5_N 0.0,A5_P 0.0,A6_N 0.0,A6_P  
0.0,E1_N 0.0,E1_P 0.0,E2_N 0.0,E2_P 0.0,E3_N 0.0,E3_P 0.0,E4_N  
0.0,E4_P 0.0,E5_N 0.0,E5_P 0.0,E6_N 0.0,E6_P 0.0,MODE #INSIDE_STOP,  
STATE FALSE}
```

Ignoring the other axes, the resulting area in which the robot is free to move has roughly the following appearance:

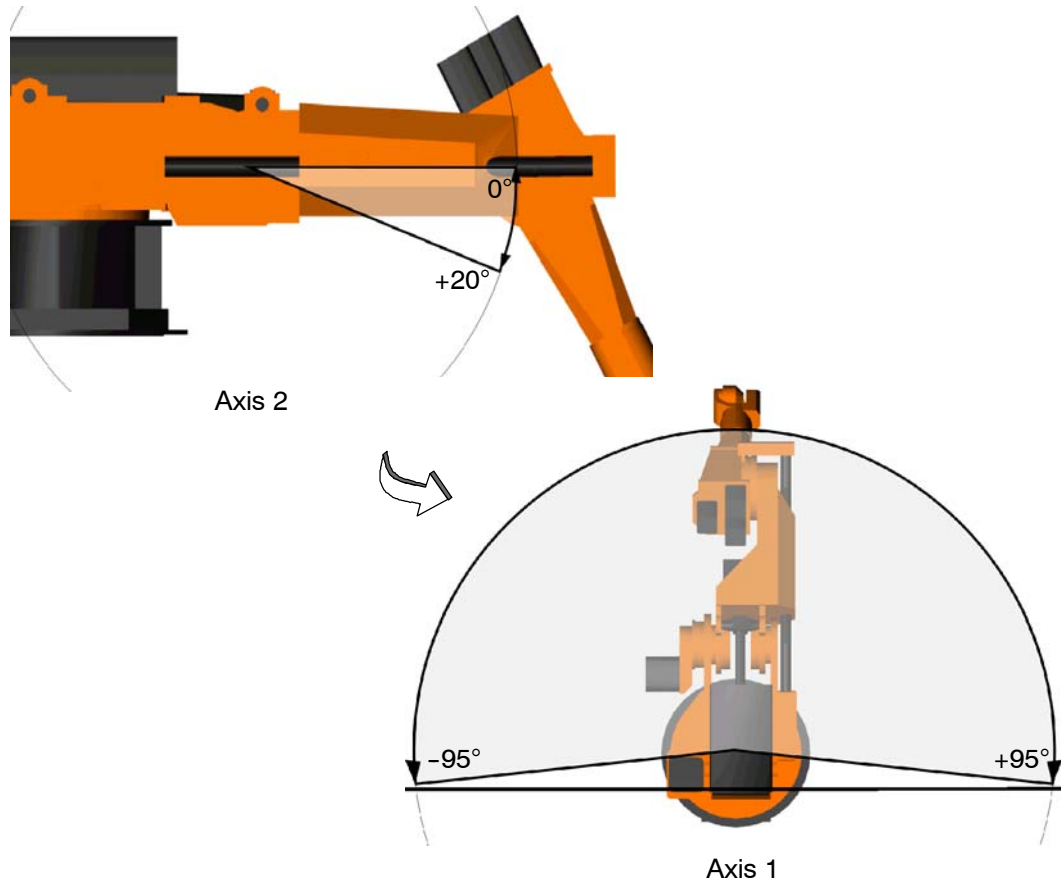


The next step is to define an area within which the robot may move.

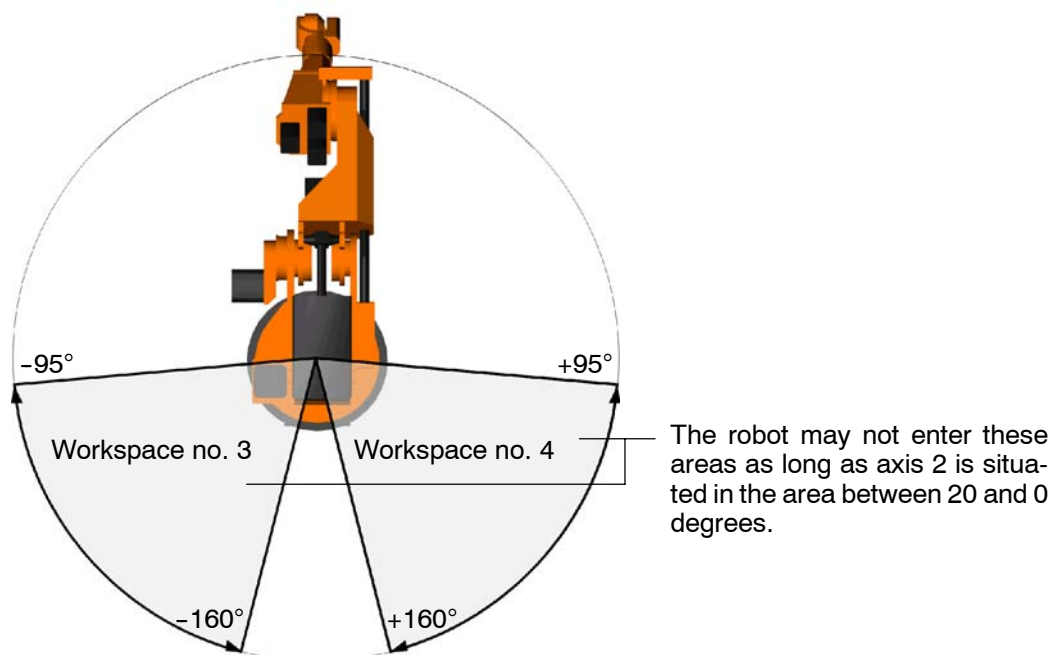
Workspaces 3 + 4



Axis 1 is to be less restricted as long as axis 2 is situated in the area between 0 and +20 degrees.

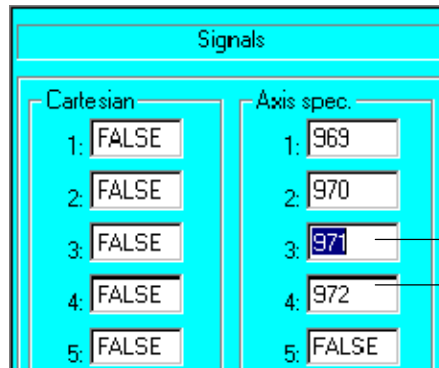


Here again, the permissible range for axis 1, taking the software limit switches into consideration, is divided into two separate areas which the robot may not enter.



The entries required for this in the status windows are as follows:

Signal



The 'Signals' window is divided into two columns: 'Cartesian' and 'Axis spec.'. The 'Cartesian' column contains five 'FALSE' buttons. The 'Axis spec.' column contains five input fields with values: 1: 969, 2: 970, 3: 971, 4: 972, 5: FALSE. A callout line points to the '971' value in the third row.

The relevant output is set in accordance with the condition.

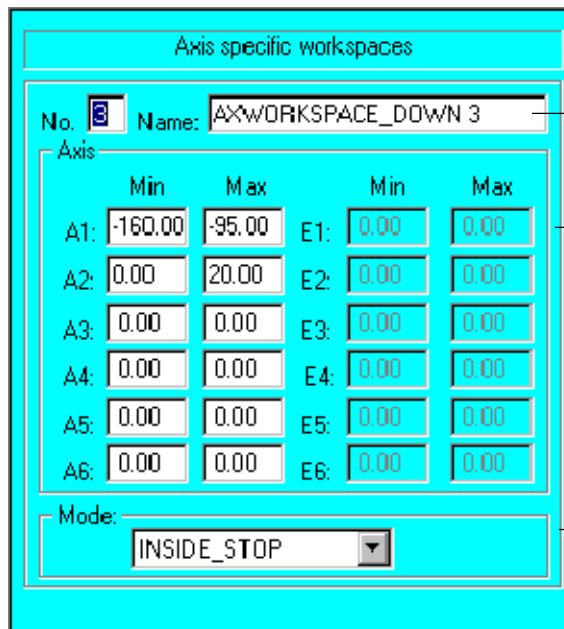
The outputs assigned to the workspaces are stored in the file “KRC\Steu\Mada\ \$Machine.dat” and are normally configured via the user interface.

```

...
SIGNAL $AXWORKSTATE1 $OUT[ 969 ]
SIGNAL $AXWORKSTATE2 $OUT[ 970 ]
SIGNAL $AXWORKSTATE3 $OUT[ 971 ]
SIGNAL $AXWORKSTATE4 $OUT[ 972 ]
SIGNAL $AXWORKSTATE5 FALSE
SIGNAL $AXWORKSTATE6 FALSE
SIGNAL $AXWORKSTATE7 FALSE
SIGNAL $AXWORKSTATE8 FALSE
...
    
```

Axis-specific

The third workspace is defined as follows:



The 'Axis specific workspaces' window shows workspace No. 3 with the name 'AXWORKSPACE_DOWN 3'. It features a table for axis specifications and a mode dropdown.

Axis	Min	Max	E1	Min	Max
A1:	-160.00	-95.00	E1:	0.00	0.00
A2:	0.00	20.00	E2:	0.00	0.00
A3:	0.00	0.00	E3:	0.00	0.00
A4:	0.00	0.00	E4:	0.00	0.00
A5:	0.00	0.00	E5:	0.00	0.00
A6:	0.00	0.00	E6:	0.00	0.00

Mode:

Axis-specific workspace no. 3 with the name “AXWORKSPACE_DOWN 3”

Angle specifications for axes 1 and 2, the other axes are not monitored

The robot is stopped and the predefined output is set as soon as axes 1 and 2 are situated in the defined workspace at the same time.

The fourth workspace:

Axis	Min	Max	Min	Max
A1:	95.00	160.00	E1:	0.00
A2:	0.00	20.00	E2:	0.00
A3:	0.00	0.00	E3:	0.00
A4:	0.00	0.00	E4:	0.00
A5:	0.00	0.00	E5:	0.00
A6:	0.00	0.00	E6:	0.00

Mode:

Axis-specific workspace no. 4 with the name "AXWORKSPACE_DOWN 4"

Angle specifications for axes 1 and 2, the other axes are not monitored

The robot is stopped and the predefined output is set as soon as axes 1 and 2 are situated in the defined workspace at the same time.

The corresponding KRL instructions read as follows:

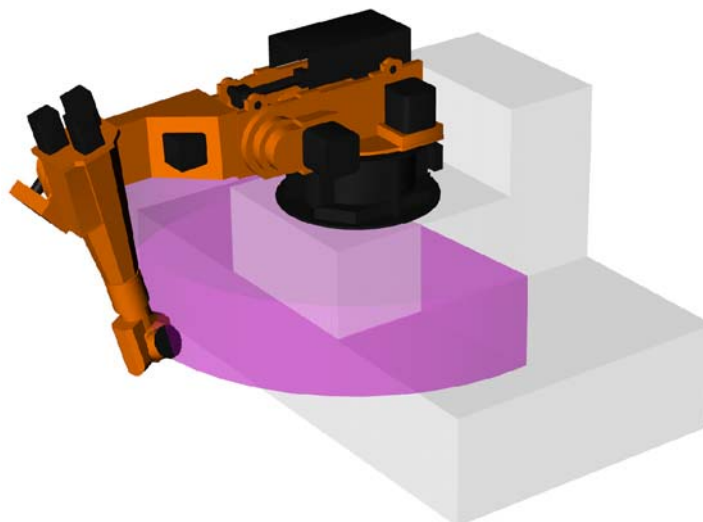
Name of the workspaces:

```
$AXWORKSPACE_NAME3 [ ]="AXWORKSPACE_DOWN 3"  
$AXWORKSPACE_NAME4 [ ]="AXWORKSPACE_DOWN 4"
```

Axes to monitor:

```
$AXWORKSPACE[3]={A1_N -160.0,A1_P -95.0, A2_N 0.0,A2_P 20.0,A3_N  
0.0,A3_P 0.0,A4_N 0.0,A4_P 0.0,A5_N 0.0,A5_P 0.0,A6_N 0.0,A6_P  
0.0,E1_N 0.0,E1_P 0.0,E2_N 0.0,E2_P 0.0,E3_N 0.0,E3_P 0.0,E4_N  
0.0,E4_P 0.0,E5_N 0.0,E5_P 0.0,E6_N 0.0,E6_P 0.0,MODE #INSIDE_STOP,  
STATE FALSE}  
  
$AXWORKSPACE[4]={A1_N 95.0,A1_P 160.0, A2_N 0.0,A2_P 20.0,A3_N  
0.0,A3_P 0.0,A4_N 0.0,A4_P 0.0,A5_N 0.0,A5_P 0.0,A6_N 0.0,A6_P  
0.0,E1_N 0.0,E1_P 0.0,E2_N 0.0,E2_P 0.0,E3_N 0.0,E3_P 0.0,E4_N  
0.0,E4_P 0.0,E5_N 0.0,E5_P 0.0,E6_N 0.0,E6_P 0.0,MODE #INSIDE_STOP,  
STATE FALSE}
```

The resulting area has roughly the following appearance:



2.7 Torque mode (Soft Servo)

2.7.1 General

It is possible to switch either individual axes or several axes to torque mode.



In normal operation, the robot counters external forces in order to remain on the programmed path.

If the forces involved are too great, this can cause damage to the component, tool or robot.

The same applies if, within the programmed path, the robot collides with the component or some other obstacle.

For certain applications, however, e.g. unloading die casting machines or working with electric motor-driven spot welding guns, it is necessary for the robot to “yield” to the external force.

In torque mode, an axis can either apply a defined torque (push or pull) against a resistance, where the torque is defined by the limitation of the speed controller output (positive and negative limits), or the axis can be moved by an external force (pushing the robot wrist or attached tool away).



Axes in torque mode **cannot** be moved in conjunction with other axes.

When torque mode is deactivated, the axis moves from the current position to the next position in the program.

2.7.1.1 Limitations and risks

Torque mode is technically possible for every robot axis, but it is not sensible for all axes due to technical constraints and the dangers arising, in part, from these constraints. The following table gives information about the possibilities and limitations.

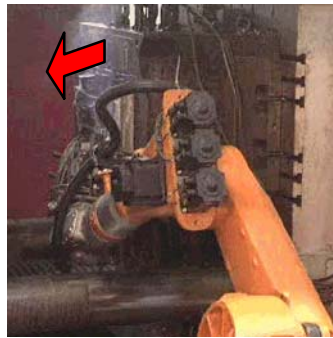
Axis	Torque mode, limitations, risks
1	Possible for floor or ceiling-mounted robots without restriction and without particular risks. With wall-mounted robots , on the other hand, torque mode is not possible because of the danger of the axis sagging.
2	Torque mode not possible. There is a risk of the axis sagging due to differing torques resulting from the position of the robot arm and, if applicable, the counterbalancing system.
3	Torque mode possible , but should be avoided because of the risk of the axis sagging due to differing torques dependent on the position of the link arm.
4	Torque mode possible , but not sensible.
5	Due to the low reaction of the wrist axis gear units, very great forces are required.
6	

2.7.1.2 Example of torque mode application

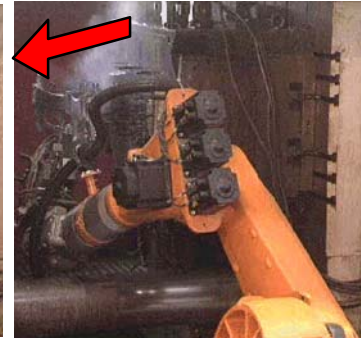
Unloading a die casting machine



The robot grips the component.



The component is removed from the mold. The ejector stroke is absorbed by a rotation of the robot about axis 1.



The robot takes over the component and moves it away from the die casting machine.

- The robot grips the component in one half of the mold.
- The part is released from the mold by a hydraulic ejector.
- The ejector stroke cannot be quantified and measured by the robot controller in the short space of time in which it occurs.
- If the robot is correctly aligned, it will “give” with axis 1 in torque mode. The ejector stroke is absorbed by a rotation of axis 1, without any damaging forces being exerted on the robot.



A diagonal ejector motion cannot be carried out, as this would require a programmed path involving several axes in torque mode.

If, however, the robot is positioned in such a way that the mounting flange of the robot base is parallel to the ejector motion (e.g. inclined installation of the robot), torque mode is possible for axis 1.



Inclination of the robot is only permissible within certain limits. See Section 2.7.1.

By setting the negative and positive limits of the speed controller output accordingly, it is possible to make the robot compensate for the torque resulting from gravitational force, and even pull the component, but allow itself to be pushed away by the ejector without applying a counterforce.

2.7.2 Functional principle

The speed controller output limits can be modified using the system variable “\$CURR_RED[x, x]”. An index is available for both the positive and negative limits. The axis in question is activated in the bit field “\$TORQUE_AXIS”.

By manipulating the speed controller limits it is possible to define whether the axis is “soft” or exerts a defined force.

Once an axis has been switched to torque mode, the monitoring functions “Regulator limit exceeded”, “Stopped”, “Positioning time” and “Motor blocked” are deactivated for this axis. Only the actual velocity continues to be monitored for this axis. In mode “Test 1”, the velocity must not exceed

$$\text{\$RED_T1} * \text{\$VEL_ACT_MA}$$

of the maximum axis velocity (machine datum 7% * 110% = 7,7%), otherwise a monitoring function is tripped and the drives are disconnected.

In modes Test 2 and Automatic, the monitoring limits can be influenced in the program by means of the variable “\$TORQ_VEL[]”. In this way, velocities of up to 150% are permissible.

When torque mode is deactivated, the axis moves from the current position to the next position in the program. A change in the value of the variable “\$TORQUE_AXIS” triggers an advance run stop.



If the program is deselected or reset, torque mode is automatically deactivated.
 In the event of manual traversing, torque mode is deactivated for the duration of the manual traversing and reactivated when the robot is repositioned.
 During repositioning, the axis does not move if it has been set to “soft”. The system moves internally, however, to the programmed path (BCO), i.e. the start key must be held down until BCO is reached.

2.7.3 Examples for activation of soft axes

2.7.3.1 Axis 1 soft

To set axis 1 as soft, the speed controller output is limited to 0%. This has the result that the axis is no longer monitored by the position and speed controllers, and can thus be moved by external forces.

```

$TORQUE_AXIS = 'B000001'           ;switch axis 1 to torque mode
$CURR_RED[1,1] = 0                  ;positive torque limit set to 0
$CURR_RED[1,2] = 0                  ;negative torque limit set to 0
PTP {A1 90}                         ;motion to release brakes
                                     ;axis is soft and can be moved

WAIT FOR $IN[17]                    ;wait for signal to cancel torque mode
                                     ;(e.g. ejector)

$TORQUE_AXIS = 0                    ;set axis “hard” again
PTP {A1 -20}                         ;move to next position
    
```

2.7.3.2 Axis 3 soft

Axis 3 can be set to “soft” by limiting the speed controller output to the current holding torque. The axis is now only monitored by the position and speed controllers near the holding torque, and the axis can be moved by external forces.

The greatest holding torque is reached when axis 3 is horizontal.



If the axis is loaded with an impermissibly high load or if the current limitation settings are incorrect, the robot arm will sag!

```

PTP {A3 90}                         ;axis 3 horizontal
$TORQUE_AXIS = 'B000100'           ;switch axis 3 to torque mode
$CURR_RED[3,1] = ABS($CURR_ACT[3]) ;positive torque limit to
                                     ;holding torque
$CURR_RED[3,2] = ABS($CURR_ACT[3]) ;negative torque limit to
                                     ;holding torque
PTP {A3 0}                          ;motion A3 vertical
                                     ;axis is soft and can be moved
    
```

```

WAIT FOR $IN[17]                ;wait for signal to cancel torque mode
                                ;(e.g. ejector)

$TORQUE_AXIS = 0                ;set axis "hard" again

PTP {A1 -20, A3 80}            ;move to next position

```

Axis 3 must, in any case, be moved to a position outside the pressure range (in the example vertical), otherwise the axis cannot be pushed upwards, as the position controller attempts to maintain the axis position using the holding torque; if you try to push the axis away, you have to overcome the weight of the axis.

The wrist axes can safely be set to "soft" with a torque of 0%; however, the reaction of the wrist axis gear units is not ideally suited to this; these axes cannot be moved very easily by external forces.



Axis 2 must never be switched to "soft" because of the pressure compensation and the fact that this axis has the lowest torque when it is in the vertical position. The risk of the axis sagging is very great!

2.7.4 Example of axis with defined torque



The following example illustrates the use of an electric motor-driven spot welding gun:

The electric motor-driven spot welding gun (external axis E1) exerts a defined torque on the component. This is done by moving to a position inside the component and setting the current resulting from the defined torque.

As this "axis" cannot reach the programmed position, because the gun's electrode touches the metal first, it presses against the component with the defined torque. If the current limitation for the inward motion is reduced, and if the second current limitation is set to 100%, it is possible to move back away from the component at full velocity.

```

PTP {E1 0}                      ;move gun to just before contact with metal

$TORQUE_AXIS = 'B1000000'       ;activate torque mode

$CURR_RED[7,1] = 20             ;set defined torque
                                ;positive limit builds up torque

PTP {E1 -10}                   ;"move" 10 mm into component
                                ;torque exerted

WAIT FOR WELD END               ;wait for "weld end" signal
                                ;set positive limit high again "on the fly"

TRIGGER WHEN DISTANCE=0 DELAY=50 DO $CURR_RED[7,1]=100

PTP {E1 20}                    ;open gun

$TORQUE_AXIS = 0                ;deactivate torque mode

```

2.7.5 Variables for torque mode

The following variables are available for torque mode:

REAL \$CURR_ACT[12]

Current value of current of axes 1 to 12 as percentage of amplifier current
 $\$CURR_MAX * \$CURR_LIM$ (-100% to +100%).

REAL \$CURR_RED[12,2]

Current limitation of axes 1 to 12 as percentage of maximum current (0% to +100%).

Index 1 is here the positive limit and index 2 the negative limit. These limits are absolute values between 0% and 100%.



**Current limitation carries the risk of no longer being able to achieve the necessary torque for holding, braking or moving the axis.
This can be dangerous for both people and machines!
Current limitation may only be used in conjunction with “\$TORQUE_AXIS”.**

INT \$TORQUE_AXIS

This is a bit array for the torque-driven robot axes A1 - A6 and external axes E1 - E6.

Torque mode is activated for an axis by setting a corresponding bit for that axis. The monitoring functions for this axis are disabled.



Changing the value of this variable triggers an advance run stop.

REAL \$TORQ_VEL[12]

Velocity limit as percentage of the maximum velocity for monitoring the torque-driven axis.



When an axis is in torque mode, all monitoring functions for this axis are disabled.

In order to ensure that a defect or fault in the hardware or the sagging of an axis is detected, the velocity is monitored.

The maximum permissible velocity in operating modes T2 and Automatic can be set in the program using the variable “\$TORQ_VEL”.

In operating mode T1, the velocity set in the machine data applies. If this velocity is exceeded, the drives are switched off and a corresponding error message is generated.

2.8 Collision monitoring

2.8.1 Function

If the robot collides with a component or if the tool gets caught on a component, the position controller and speed controller react accordingly. The command torques of the axes involved are automatically increased. Depending on the application, the robot may be able to overcome the resistance and continue its motion. This may, however, damage the tool or component.



External axes are not monitored.

The user can configure both the torque limits and the response time. A KRL variable is used to define an area (a so-called “monitoring tunnel”) around the particular torque. If the torque leaves this monitoring tunnel, a path-maintaining stop reaction is triggered after the specified response time and a corresponding message is generated in the message window.

Monitoring is switched off by default, but the default value for the monitoring tunnel is 200%. The monitoring can be made more sensitive for individual motions or program sections.



In the event of a reset, block selection or program deselection, the limits that were set are reset to the default value in Custom.dat.



The collision monitoring function does not provide any guarantee against damage, but it can reduce the extent of any damage. Both the traversing velocity and the torque values arising from the collision play an important role in determining the type of damage caused.

The signal outputs \$COLL_ENABLE and \$COLL_ALARM remain available in KRC:\STEU\MACHINE.DAT. COLL_ENABLE is set if the current monitoring limit of an axis in program mode, \$torqmon, is less than 200%. \$COLL_ALARM is set if the message “117 Torque exceeded axis Ax” is generated for an axis. This output remains set as long as \$STOPMESS is active.

2.8.2 Configuration

In order to be able to use the collision monitoring function, acceleration adaptation must also be switched on. This is the case if the variable “\$ADAP_ACC” has the value “#STEP1”. This variable can be found in the file “C:\KRC\Roboter\KRC\R1\MaDa\$ROBCOR.DAT”:

```
...
DECL ADAP_ACC $ADAP_ACC=#STEP1 ; ACCELERATION ADAPTATION
...
```



The load data must be calculated accurately for the purposes of collision monitoring. The monitoring tunnel must also be adapted to the particular application.

The response time of the monitoring in [ms] can be set using the variable “\$TORQMON_TIME”. This variable can be found in the file “C:\KRC\Roboter\KRC\Steu\MaDa\$CUSTOM.DAT”:

```
:
REAL $TORQMON_TIME=0.0 ; TIME FOR TORQUE MONITORING
:
```

The size of the monitoring tunnel can be specified by means of the following variables in the file “C:\KRC\Roboter\KRC\MaDa\$CUSTOM.DAT”:

- Default values for the monitoring tunnel for program mode in percent
 - :
 - \$TORQMON_DEF [1]=200
 - \$TORQMON_DEF [2]=200
 - \$TORQMON_DEF [3]=200
 - \$TORQMON_DEF [4]=200
 - \$TORQMON_DEF [5]=200
 - \$TORQMON_DEF [6]=200
 - :
- Default values for the monitoring tunnel for command mode in percent
 - :
 - \$TORQMON_COM_DEF [1]=200
 - \$TORQMON_COM_DEF [2]=200
 - \$TORQMON_COM_DEF [3]=200
 - \$TORQMON_COM_DEF [4]=200
 - \$TORQMON_COM_DEF [5]=200
 - \$TORQMON_COM_DEF [6]=200
 - :



The width of the tolerance band is equal to the maximum torque [Nm] multiplied by the percentage in “\$TORQMON_...”.

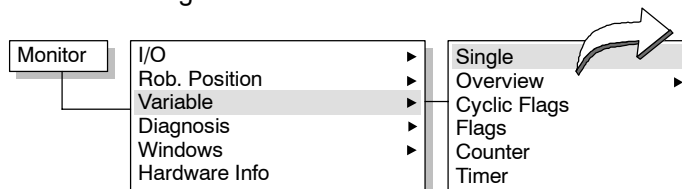
Torques are monitored in

- Program mode
 - Advance run:
Within the KRL program, a block-specific tolerance band for the torque can be specified in the variable “\$TORQMON[]”.
 - Main run:
The monitoring is activated or deactivated immediately when the variable is written to in the interrupt program.
- Command mode
 - The values of the variables “\$TORQMON_COM_DEF[1]...[6]” in the file “\$CUSTOM.DAT” are valid by default for torque monitoring. By means of variable modification, the user can change the monitoring limits at any time by altering the values of “\$TORQMON_COM[1]...[6]”.

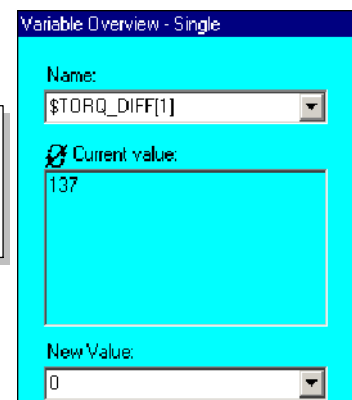
The maximum torque deviation that has occurred can be read as a percentage in the system variable “\$TORQ_DIFF[1]...[6]”. This variable can thus be used to optimize the torque monitoring.



Before a motion command or motion section, set the variable to “0” using the variable modification function.



Now execute the motion command using suitable path parameters (velocity, acceleration). Then read the variable again. The value given corresponds to the maximum torque deviation that has occurred.



Now set the monitoring tunnel to the value of “\$TORQ_DIFF[]” plus a safety margin of 5–10%.



Only the value "0" can be assigned to the variable "\$TORQ_DIFF[]".



No liability will be accepted for any damage resulting from incorrect settings!



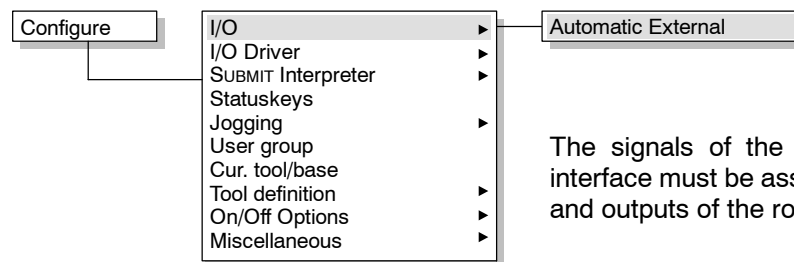
3 Automatic External

3.1 General

When using interlinked production lines, it is necessary to be able to start robot processes from a central position. A host computer can communicate with the robot controller via the “Automatic External” interface and activate various robot processes. Similarly, the robot controller can send information about operating states and fault signals to the host computer.

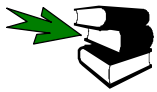
In the KR C..., this is all done by the automatic system start, the technology-specific organization program `CELL.SRC` and the functions of the `P00` module.

3.2 Configuring the interface



The signals of the “Automatic External” interface must be assigned physical inputs and outputs of the robot controller.

The status window for the Automatic External interface inputs is then opened.



An explanation of the inputs/outputs for Automatic External can be found in Section 3.6.



Using the “↓” and “↑” arrow keys, move the focus (the dark blue highlight) to the desired box. Then use the numeric keypad to enter the number of the interface to which the signal is to be assigned.



3.2.1 Inputs

This status window displays the inputs of the Automatic External interface.

Automatic External - Configuration: Inputs				
	Term	Type	Name	Value
1	Type programno.	Var	PGNO_TYPE	1
2	REFLECT_PROG_NR	Var	REFLECT_PROG_M 0	
3	Bitwidth programno.	Var	PGNO_LENGTH	8
4	First bit programno.	I/O	PGNO_FBIT	33
5	Parity bit	I/O	PGNO_PARITY	41
6	Programno. valid	I/O	PGNO_VALID	42
7	Programstart	I/O	\$EXT_START	1026
8	Move enable	I/O	\$MOVE_ENABLE	1025
9	Error confirmation	I/O	\$CONF_MESS	1026
10	Drives off (invers)	I/O	\$DRIVES_OFF	1025
11	Drives on	I/O	\$DRIVES_ON	140
12	Activate interface	I/O	\$I_O_ACT	1025

Configuration

Term	A functional description of the specific variable or input.
Var IO	The type can be a variable (yellow) or an input (green).
Name	The variable name of the corresponding input.
Value	The value of the input or the channel number.
Display	Corresponds to the function "Monitor" -> "I/O" -> Automatic External".
Outputs	The softkey "Outputs" toggles to the corresponding status window.
Value	This softkey can be used to change the value of the input or the channel number.
	<input type="text" value="140"/>
OK	When "OK" is pressed, the validity of the changed value is checked; if the input is permissible, the new value is accepted.
Cancel	"Cancel" is used to reject the change.
Close	Closes the status window.

3.2.1.1 Outputs

This status window displays the outputs of the Automatic External interface.

Automatic External - Configuration: Outputs				
	Term	Type	Name	Value
1	Control ready	IO	\$RC_RDY1	137
2	Alarm stop active	IO	\$ALARM_STOP	1013
3	User safety switch closed	IO	\$USER_SAF	1011
4	Drives ready	IO	\$PERI_RDY	1012
5	Robot calibrated	IO	\$ROB_CAL	1001
6	Interface active	IO	\$I_O_ACTCONF	140
7	Error collection	IO	\$STOPMESS	1010
8	PGNO_FBIT_REFL	IO	PGNO_FBIT_REFL	999

Start conditions | Program state | Robot position | Operat

Term	A functional description of the specific variable or output.
Var IO	The type can be a variable (yellow) or an output (green).
Name	The variable name of the corresponding output.

Value	The value of the output or the channel number.
Display	Corresponds to the function "Monitor" -> "I/O" -> Automatic External".
Inputs	Switches to the status window for the inputs of the Automatic External interface.
Tab +	The softkeys "Tab+" and "Tab-" allow you to move up or down to the next group of variables.
Tab -	
Value	This softkey can be used to change the value of the output or the channel number.
<div style="border: 1px solid black; display: inline-block; padding: 2px;">137</div>	
OK	When "OK" is pressed, the validity of the changed value is checked; if the input is permissible, the new value is accepted.
Cancel	"Cancel" is used to reject the change.
Close	Closes the status window.



If modifications are made to the "\$MACHINE.DAT" file, the Submit interpreter is briefly deselected while the data are transferred and then automatically reselected once they are saved.

3.3 Automatic system start

If the I/O interface has been activated by setting the system variable \$I_O_ACT to the value TRUE, the output \$_I_O_ACTCONF is also switched to TRUE as a feedback signal. If all other start conditions have been met, the program CELL.SRC can be started by a signal on the line \$EXT_START.

The CELL.SRC program can also, of course, be opened at any time from the user interface.

For automatic system start the following value must be assigned to the system variable \$PRO_I_O in the file "C:\KRC\Roboter\KRC\Steu\MaDa\\$_CUSTOM.DAT" :

```
CHAR $PRO_I_O[ ]="/R1/SPS( )"
```



After the controller has run up, it always tries to execute the program designated in \$PRO_I_O.



Before the I/O interface can be activated at all, the input assigned using the signal declaration of the variable \$I_O_ACT must first be set.

3.4 Technology-specific organization program CELL.SRC

Instruction for linking the user-defined, external subprograms ...

```
;EXT EXAMPLE1 ( )
;EXT EXAMPLE2 ( )
;EXT EXAMPLE3 ( )
```

Initialization sequence ...

```
INIT
BAS INI
CHECK HOME
PTP HOME Vel= 100 % DEFAULT
AUTOEXT INI
```

Start of loop ...

```
LOOP
```

The P00 module is called to get the program number from the host computer ...

```
P00 (#EXT_PGNO, #PGNO_GET, DMY [ ], 0 )
```

Control structure dependent on the program number received ...

```
SWITCH PGNO
```

If program number PGNO = 1 ...

```
CASE 1
```

... communicate receipt of the program number to the host computer ...

```
P00 (#EXT_PGNO, #PGNO_ACKN, DMY [ ], 0 )
```

... and call the user-defined program EXAMPLE1

```
;EXAMPLE1 ( )
```

If program number PGNO = 2 ...

```
CASE 2
```

... communicate receipt of the program number to the host computer ...

```
P00 (#EXT_PGNO, #PGNO_ACKN, DMY [ ], 0 )
```

... and call the user-defined program EXAMPLE2

```
;EXAMPLE2 ( )
```

If program number PGNO = 3 ...

```
CASE 3
```

... communicate receipt of the program number to the host computer ...

```
P00 (#EXT_PGNO, #PGNO_ACKN, DMY [ ], 0 )
```

... and call the user-defined program EXAMPLE3

```
;EXAMPLE3 ( )
```

If no CASE branch is found for the program number communicated by the host computer,

```
DEFAULT
```

error treatment is carried out ...

P00 (#EXT_PGNO, #PGNO_FAULT, DMY[], 0)

End of the control structure ...

ENDSWITCH

End of the loop ...

ENDLOOP

Programmende ...

END

3.5 The P00 (AUTOMATIC EXTERNAL) module

The P00 module contains the functions for the transfer of program numbers via a host computer. The functions `INIT_EXT`, `EXT_PGNO`, `CHK_HOME` and `EXT_ERR` are grouped together in this global subprogram.

3.5.1 The `EXT_PGNO` function

This function takes over the complete signal handling process for the transfer of program numbers via a host computer.

It can be called by one of the three following parameters:

#PGNO_GET	Program number request
#PGNO_ACKN	Communication of receipt of a program number
#PGNO_FAULT	Error treatment

3.5.1.1 Request of a program number from the host computer

EXT_PGNO (#PGNO_GET)

If the host computer detects a program number request on the `PGNO_REQ` line, it sets the program number as a binary value at the robot controller inputs provided for this purpose.

In order to increase reliability of the transfer, a parity bit, `PGNO_PARITY`, can be transferred from the host computer in addition to the program number. If the signal levels remain stable, the host computer sets the `PGNO_VALID` or `EXT_START` line in order to request the robot controller to read the program number. The `EXT_PGNO` function now calculates the parity of the program number received and compares it with the parity bit. If the result is positive, the function returns the received program number as an integer value. If, however, the received and calculated parities do not agree, the program number is set to the value "0". An error message will be displayed in the status window of the KCP.



Since the program number is always set to zero in the event of a parity error, this value cannot, of course, be used as a valid program number in `CELL.SRC`.

3.5.1.2 Communication of receipt of a valid program number

EXT_PGNO (#PGNO_ACKN)

Once the program number has been transferred correctly, the control structure in `CELL.SRC` attempts to assign this program number to an application program. If this is successful, the function automatically cancels the request for a program number. It signals this to the host computer by setting the line `APPL_RUN`.

If, on the other hand, it is unsuccessful, the error treatment functions described below are called.

3.5.1.3 Error handling

EXT_PGNO (#PGNO_FAULT)

If the program number was not transferred correctly, i.e.

- (7) the parity check was not successful, or
- (8) the BCD encoding was incorrect, or more precisely: the decoding did not produce a valid result, or
- (9) no application program was assigned to this program number,

the EXT_PGNO function displays a transmission error in the message window of the KCP. The PGNO_REQ line is reset. This informs the host computer that the transmission contained errors.



A faulty transmission can be recognized by the host computer on the basis of a timeout. This timeout is started when the PGNO_VALID line is set. If the program number request on the PGNO_REQ line is not cancelled after a specified period of time (about 200 ms), an error must have occurred in the transfer. The host computer can now react to the error.

3.5.2 The EXT_ERR function

This function can be used to transfer to the host computer, via eight specified outputs, a declared error number between 1 ... 255. Furthermore, the 64 most recent errors are stored in the ERR_FILE ring memory for closer analysis.

In order to be able to use the EXT_ERR function, you need to edit the p00.dat file as described below:

```

&ACCESS R
&COMMENT EXTERNAL package
DEFDAT P00

BOOL PLC_ENABLE = TRUE Set this value to TRUE

INT I
INT F_NO=1

INT MAXERR_C = 1 Enter here the number of controller errors for the transmission of which you have defined parameters

INT MAXERR_A = 1 Enter here the number of application errors for the transmission of which you have defined parameters

DECL STOPMESS MLD

SIGNAL ERR $OUT[25] TO $OUT[32] Specify here which robot controller outputs the host computer should use to read the error number  
In the example, these are outputs 25 to 32
    
```

```

BOOL FOUND
STRUC PRESET INT OUT,CHAR PKG[3],INT ERR
DECL PRESET P[255]

```

In the following section you must enter the **parameters** of the errors:

OUT -

Error number to transfer to the host computer

PKG[] -

Technology package

ERR -

Error number in the selected technology package

```

P[1]={OUT 2,PKG[]"P00",ERR 10}
...
P[127]={OUT 27,PKG[]"S00",ERR 11}
P[128]={OUT 12,PKG[]"CTL",ERR 1}
...
P[255]={OUT 25,PKG[]"CTL",ERR 10}

```

In the range **P[1] ... P[127]** you can only enter **application errors**

In the range **P[128] ... P[255]** you can only enter **controller errors**

```

STRUC ERR_MESS CHAR P[3],INT E
DECL ERR_MESS ERR_FILE[64]
ERR_FILE[1]={P[] "XXX",E 0}
...
ERR_FILE[64]={P[] "XXX",E 0}
ENDDAT

```

3.6 Signal descriptions

The signals are write-protected, but can be read or used in programs at any time.

3.6.1 Inputs

3.6.1.1 PGNO_TYPE

This is neither an input nor a signal, but a variable. Its value determines the format in which the program number sent by the host computer is read.

PGNO_TYPE	Read as ...	Meaning	Examples
1	Binary number	The program number is transmitted by the higher-level controller as a binary coded integer	0 0 1 0 0 1 1 1 => PGNO = 39
2	BCD value	The program number is transmitted by the higher-level controller as a binary coded decimal	0 0 1 0 0 1 1 1 └──┬──┘ └──┬──┘ 2 7 => PGNO = 27
3	"1 of n" *1	The program number is transmitted by the higher-level controller or the periphery as a "1 of n" coded value	0 0 0 0 0 0 0 1 => PGNO = 1 0 0 0 0 1 0 0 0 => PGNO = 4
*1 When using this transmission format, the values of PGNO_REQ, PGNO_PARITY and PGNO_VALID are not evaluated and are thus of no significance.			

3.6.1.2 PGNO_LENGTH

This is also neither an input nor a signal, but again a variable. Its value determines the number of bits defining the program number sent by the host computer.

PGNO_LENGTH = 1...16

Example:

PGNO_LENGTH = 6 => the external program number is six bits long



While PGNO_TYPE has the value 2 (program number read as BCD value), only 4, 8, 12 and 16 are permissible values for the number of bits.

3.6.1.3 PGNO_FBIT

Input representing the first bit of the program number.

PGNO_FBIT = 1...1024 (PGNO_LENGTH)

Example:

PGNO_FBIT = 5 => the external program number begins with \$IN[5]

3.6.1.4 REFLECT_PROG_NR

This option allows you to decide whether or not the program number should be mirrored in a definable output area. The value of the variable can be changed via the configuration of the Automatic External interface.

REFLECT_PROG_NR	Function
0	deactivated
1	activated



The output of the signal starts with the output defined using "PGNO_FBIT_REFL".

3.6.1.5 PGNO_PARITY

Input to which the parity bit is transferred from the host computer.

Input	Function
Negative value	Odd parity
0	No evaluation
Positive value	Even parity



While PGNO_TYPE has the value 3 (program number read as "1 of n" value), PGNO_PARITY is **NOT** evaluated.

3.6.1.6 PGNO_VALID

Input to which the command to read the program number is transferred from the host computer.

Input	Function
Negative value	Number is transferred at the falling edge of the signal

0	Number is transferred at the rising edge of the signal on the EXT_START line
Positive value	Number is transferred at the rising edge of the signal



While PGNO_TYPE has the value 3 (program number read as “1 of n” value), PGNO_VALID is **NOT** evaluated.

3.6.1.7 EXT_START

If the I/O interface is active, this input can be set to start or continue a program.



Only the rising edge of the signal is evaluated.



There is no BCO run in Automatic External mode, so there is no program stop at the first programmed position. This applies both after dynamic braking with deviation from the path (e.g. operator safety) and after manual movement of the robot off the path.

The first position to be addressed in such cases is the position saved in \$POS_RET before the interruption. This means that when EXT_START is set, the operator must make sure that the robot is either already situated in this position or can safely reach it.

The first motion block must be a PTP block with absolute specification of the target point. The robot always moves to this target point exactly and at **maximum** velocity, with any **approximate positioning instruction** programmed being **ignored!**

3.6.1.8 MOVE_ENABLE

This input is used by the host computer to check the robot drives.

Signal	Function
TRUE	Manual motion and program execution are possible
FALSE	All drives are stopped and all active commands inhibited



If the drives have been switched off by the host computer, the message “GENERAL MOTION ENABLE” appears in the message window of the KCP. It is only possible to move the robot again once this message has been reset and another external start signal has been given.



During commissioning, the motion enable variable “\$MOVE_ENABLE” is often configured with the value “\$IN[1025]”. If a different input is not subsequently configured, no external start is possible.

3.6.1.9 CHCK_MOVENA

If the variable \$CHCK_MOVENA has the value “FALSE”, MOVE_ENABLE can be bypassed. The value of the variable can only be changed in the file “C:\KRC\Roboter\KRC\Steu\MaDa\OPTION.DAT”.

Signal	Function
TRUE	MOVE_ENABLE monitoring is active
FALSE	MOVE_ENABLE monitoring is deactivated



In order to be able to use MOVE_ENABLE monitoring, \$MOVE_ENABLE must have been configured with the input "\$IN[1025]". Otherwise, "\$CHCK_MOVENA" has no effect whatsoever.

3.6.1.10 CONF_MESS

Setting this input enables the host computer to reset (acknowledge) error messages automatically.



Only the rising edge of the signal is evaluated.

Acknowledgement of the error messages is, of course, only possible once the cause of the error has been eliminated.

3.6.1.11 DRIVES_ON

With a high-level pulse of at least 20 ms duration at this input, the host computer can switch on the robot drives.



The signal "DRIVES_ON" must be set 500 ms after the signal "USER_SAF". In this way, a possible ESC circuit error message is avoided.

3.6.1.12 DRIVES_OFF

With a low-level pulse of at least 20 ms duration at this input, the host computer can switch off the robot drives.

3.6.2 Outputs

3.6.2.1 STOPMESS

This output is set by the robot controller in order to communicate to the host computer any message occurring which requires the robot to be stopped.

(e.g. EMERGENCY STOP, Driving condition, Operator safety, Command velocity, etc.)

3.6.2.2 PGNO_REQ

A change of signal at this output requests the host computer to send a program number.



Both edges of the signal are evaluated.

While PGNO_TYPE has the value 3 (program number read as "1 of n" value), PGNO_REQ is **NOT** evaluated.

3.6.2.3 PGNO_FBIT_REFL

Mirrored output representing the first bit of the program number. To use this option, the variable "REFLECT_PROG_NR" must be assigned the value "1".

If a program selected by the PLC is deselected by the user, the output area starting with PGNO_FBIT_REFL is set to "FALSE". In this way, the PLC can prevent a program from being restarted manually.



PGNO_FBIT_REFL is also set to "FALSE" if the interpreter is situated in the CELL program.

$PGNO_FBIT_REFL = 1 \dots 1024$ (PGNO_LENGTH)

The size of the output area depends on the number of bits defining the program number (PGNO_LENGTH).

Example:

$PGNO_FBIT_REFL = 5 \Rightarrow$ the program number begins with \$OUT[5]

3.6.2.4 APPL_RUN

By setting this output, the robot controller communicates to the host computer the fact that a program is currently being executed.



The value of APPL_RUN may not be less than "0".

3.6.2.5 PERI_RDY

By setting this output, the robot controller communicates to the host computer the fact that the robot drives are switched on.

3.6.2.6 ALARM_STOP

This output is reset in the event of an EMERGENCY STOP event.

3.6.2.7 USER_SAF

This output is reset if the safety fence monitoring switch is opened (in AUTO mode) or an enabling switch is released (in TEST mode).

3.6.2.8 T1, T2, AUT, EXTERN

These outputs are set when the corresponding operating mode is selected.

3.6.2.9 ON_PATH

This output remains set as long as the robot stays on its programmed path.

The output `ON_PATH` is set after the BCO run. This output remains set until the robot leaves the path; the program is reset or block selection is carried out. The `ON_PATH` signal has no tolerance window, however; as soon as the robot leaves the path the signal is reset.

3.6.2.10 NEAR_POSRET

A second signal, **NEAR_POSRET**, allows the host computer to determine whether or not the robot is situated within a sphere about the position saved in `$POS_RET`. The user can set the radius of the sphere in the file `$CUSTOM.DAT` using the system variable `$NEARPATHTOL`.

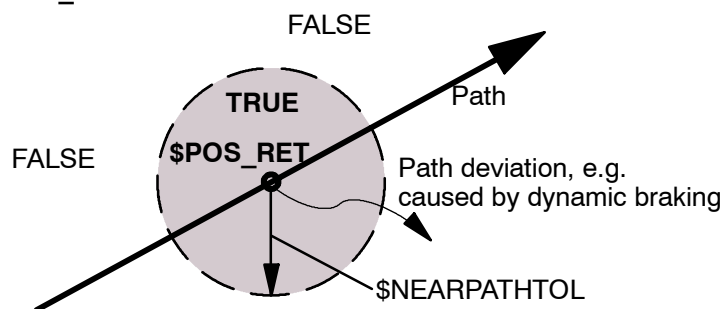


The host computer can use this information to decide whether or not the program may be restarted.

The return position `$POS_RET` is the position at which the robot has left the path.

When switching to “Automatic External” mode, a check is made to see if the variable “`$NEAR_POSRET`” is set to “TRUE”. If this is not the case, a corresponding error message will be displayed in the message window.

`$NEAR_POSRET`



Possible states for **NEAR_POSRET**:

TRUE:

`ON_PATH` is set, or if `ON_PATH` is not set: `$POS_RET` is valid and the position is within the sphere about `$POS_RET`.

FALSE:

`ON_PATH` is reset and `$POS_RET` is invalid or the position is outside the sphere about `$POS_RET`.

Setting:

File: \$MACHINE.DAT

```
SIGNAL $NEAR_POSRET $OUT[xxx]
```

3.6.2.11 PRO_ACT

This output is always set if a process or the program execution is active at robot level.

Its signal state is derived from the system variable \$PRO_STATE1:

```
$PRO_STATE1=#P_ACTIVE → $PRO_ACT=TRUE
```

```
all other process states → $PRO_ACT=FALSE
```

The process is therefore active as long as a program or an interrupt is being processed. Program processing is set to the inactive state at the end of the program only after all pulse outputs and all triggers have been processed. There are three possible situations in the event of an error stop:

- If interrupts have been activated but not processed at the time of the error stop, the process is regarded as inactive ($PRO_ACT=FALSE$)
- If interrupts have been activated and processed at the time of the error stop, the process is regarded as active ($PRO_ACT=TRUE$) until the interrupt program is completed or a STOP occurs in it ($PRO_ACT=FALSE$)
- If interrupts have been activated and a STOP occurs in the application program, the process is regarded as inactive ($PRO_ACT=FALSE$). If an interrupt condition is fulfilled after this time, the process is regarded as active ($PRO_ACT=TRUE$) until the interrupt program is completed or a STOP occurs in it ($PRO_ACT=FALSE$)

3.6.2.12 IN_HOME

This output communicates to the host computer whether or not the robot is in its HOME position.

3.6.2.13 ERR_TO_PLC

By setting this output, the robot controller communicates to the host computer the fact that a controller or application error has occurred.



This function is only active if PLC_ENABLE has the value TRUE.

3.6.3 Other variables

3.6.3.1 PGNO

The program EXT_PGNO.SRC stores the program number received from the host computer in this variable (irrespective of the parameterized data type) as an integer value.

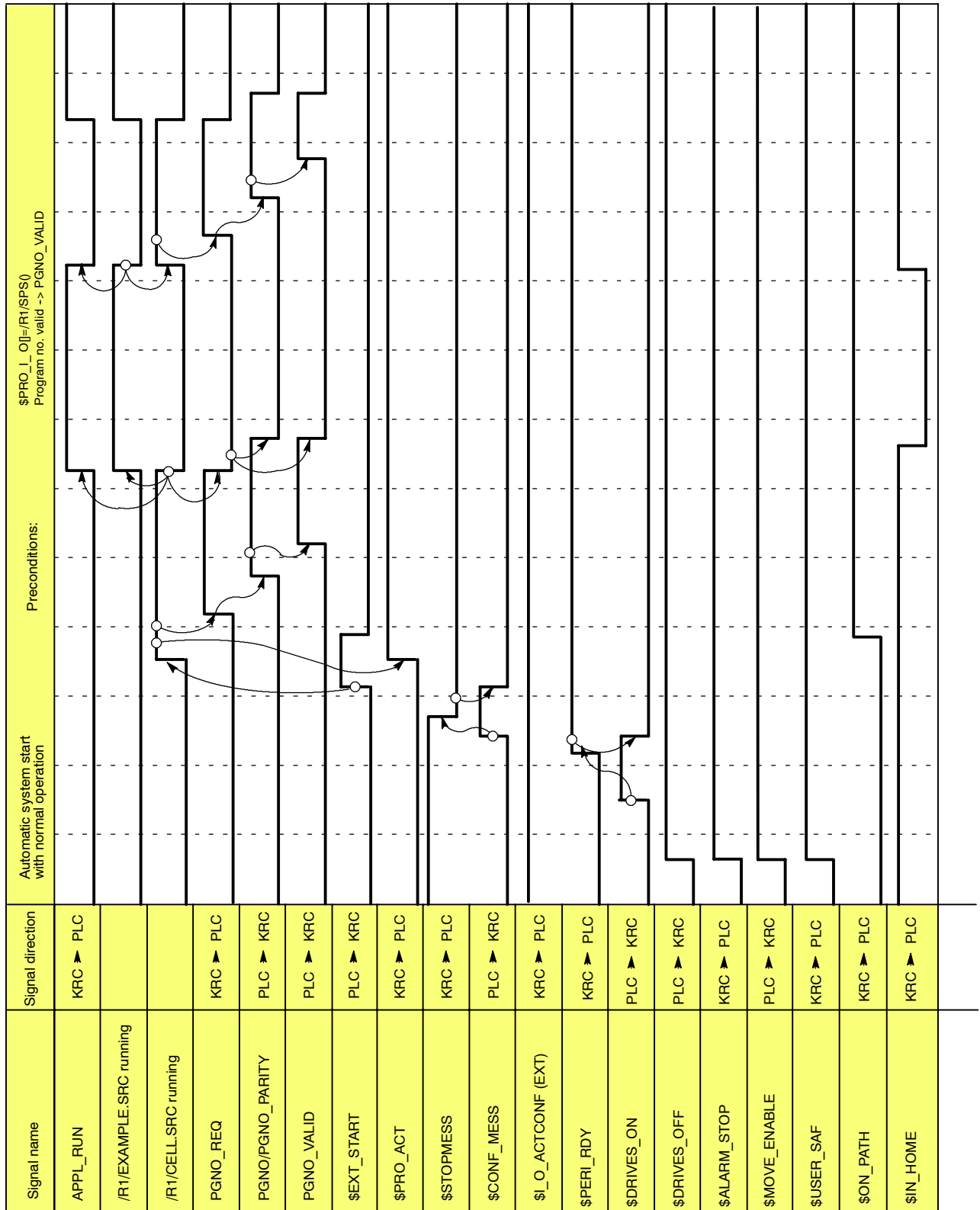
The technology-specific organization program CELL.SRC uses this variable to assign the corresponding application program to the program number.

3.6.3.2 PGNO_ERROR

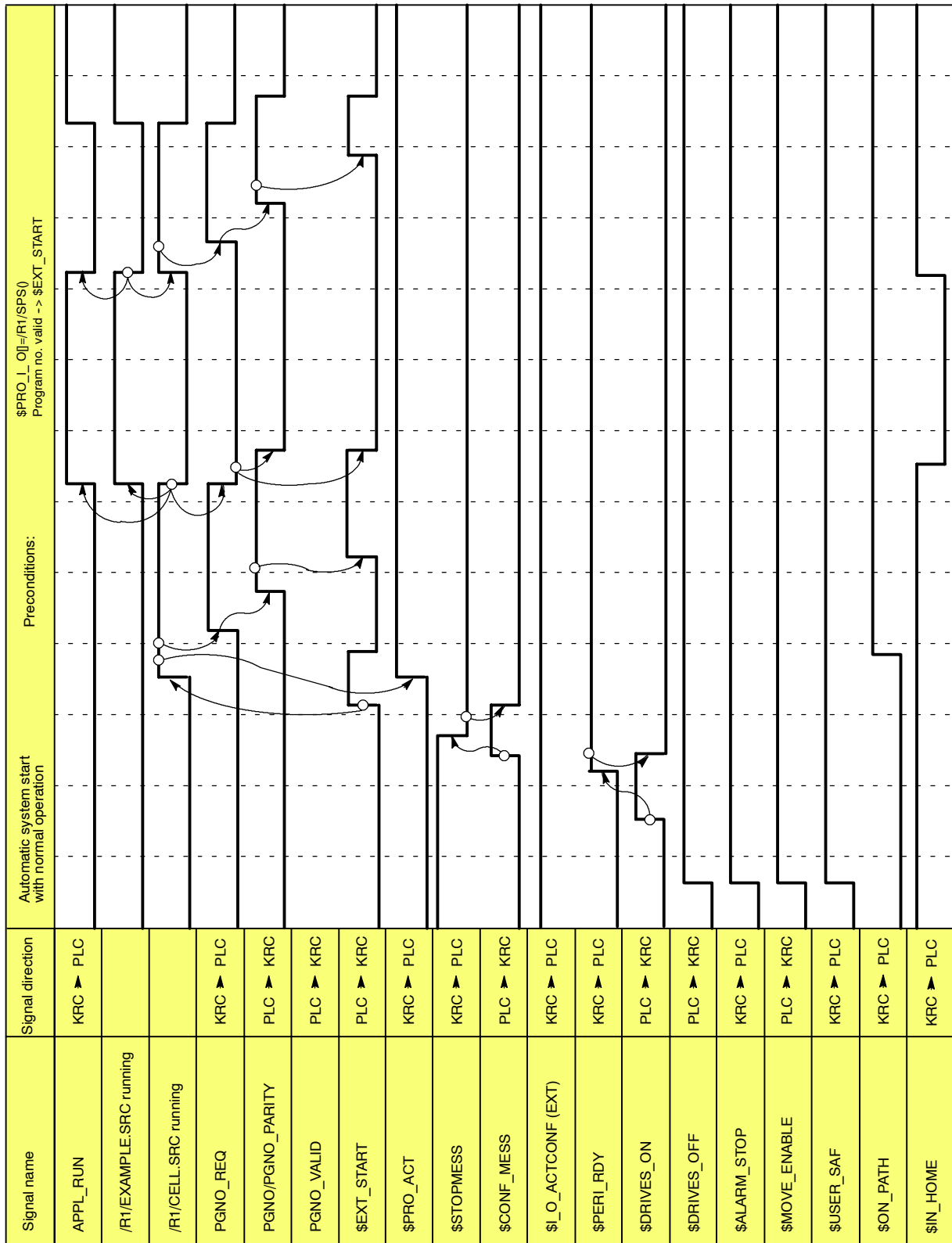
This variable is used for the internal error management of the program EXT_PGNO.SRC and must not be used or altered!

3.7 Signal diagrams

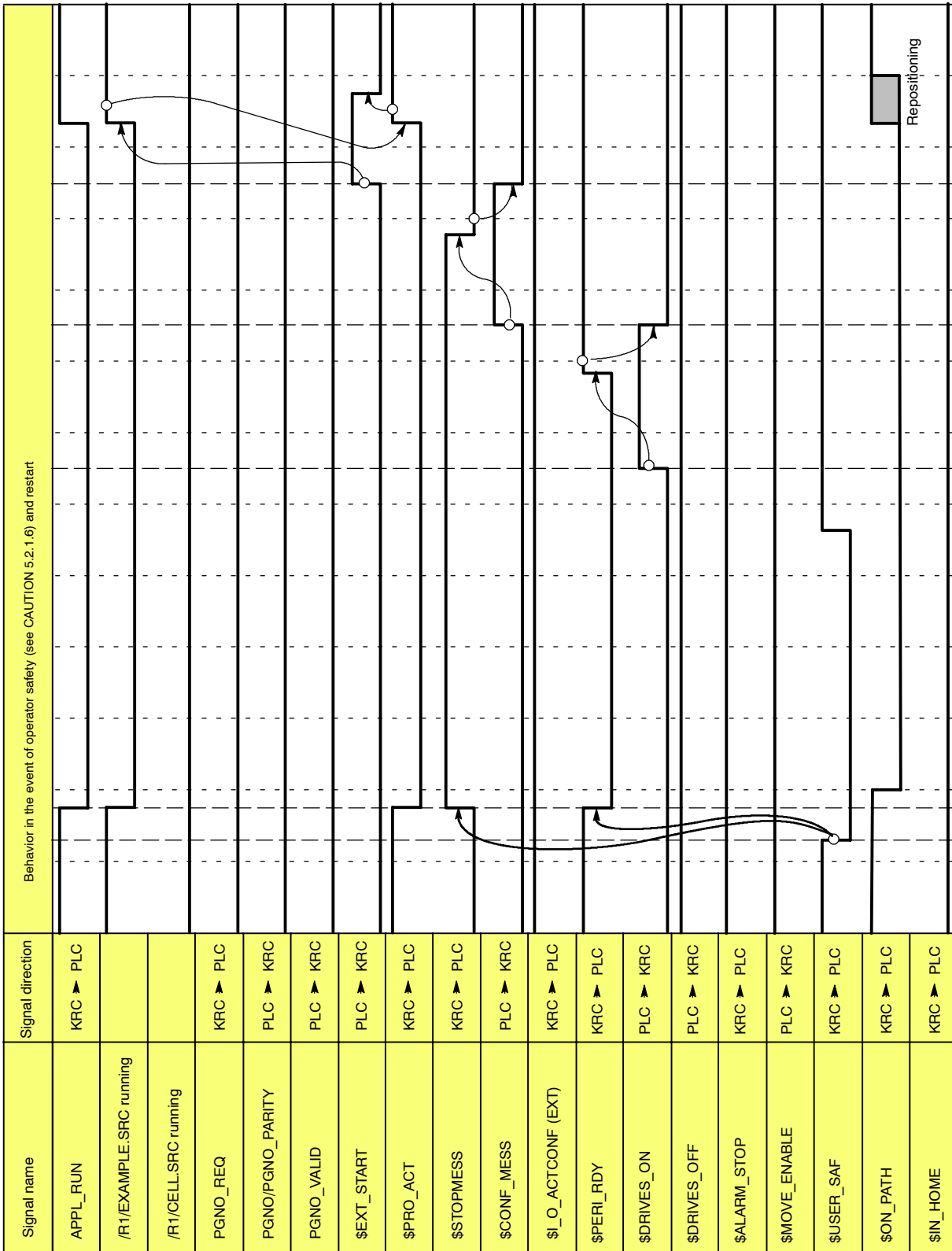
3.7.1 Automatic system start and normal operation with program number acknowledgment by means of PGNO_VALID



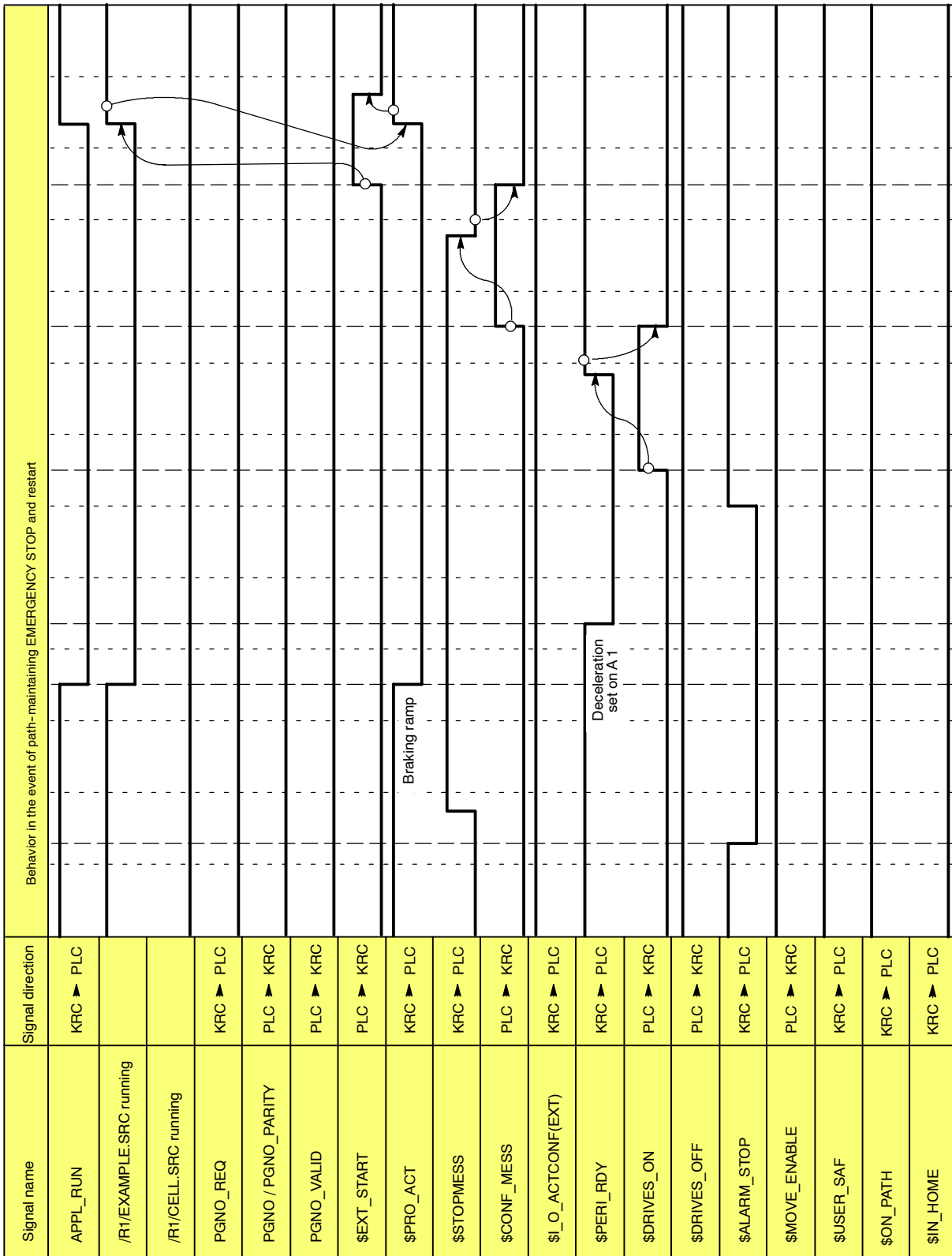
3.7.2 Automatic system start and normal operation with program number acknowledgment by means of \$EXT_START



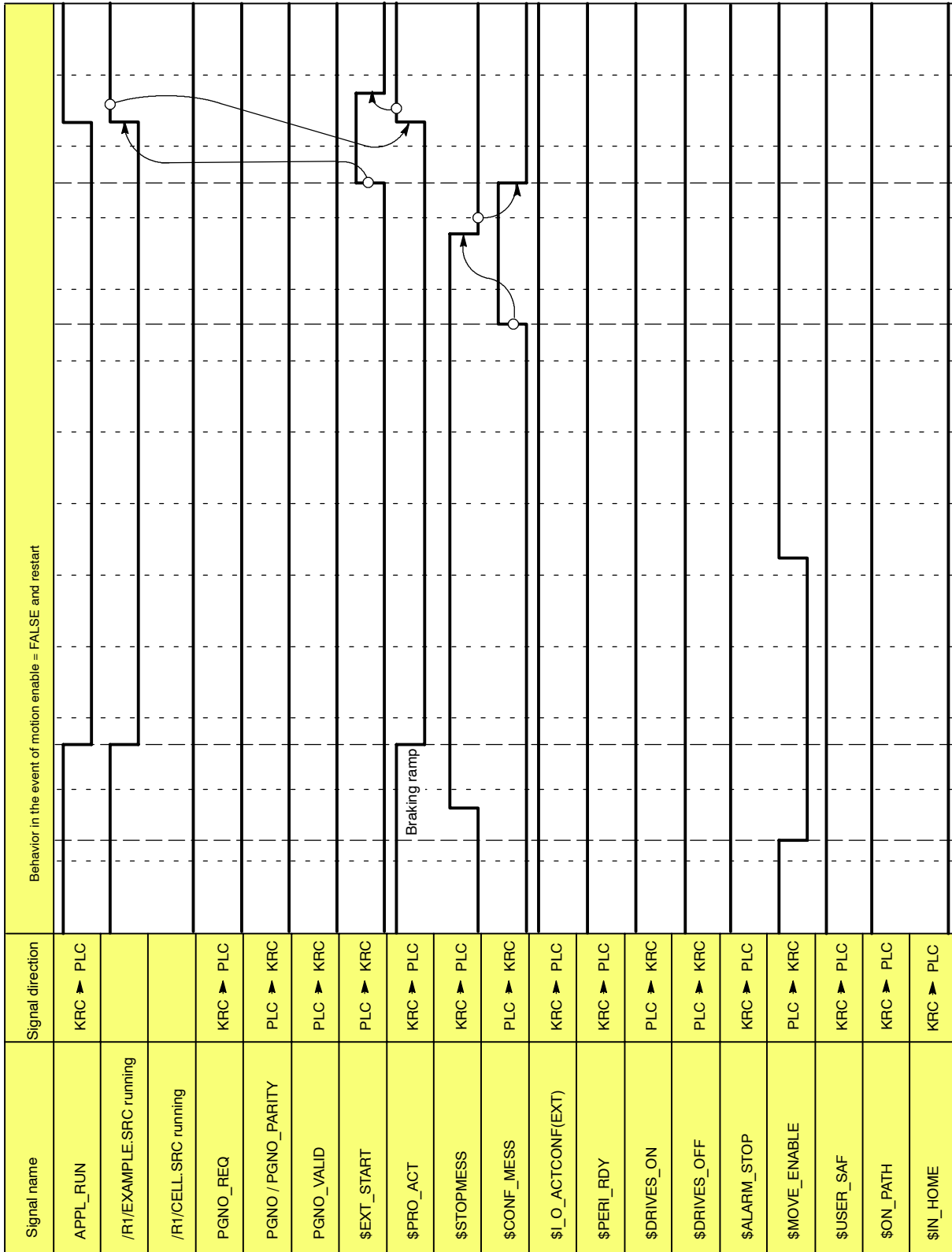
3.7.3 Restart after dynamic braking (operator safety and restart)



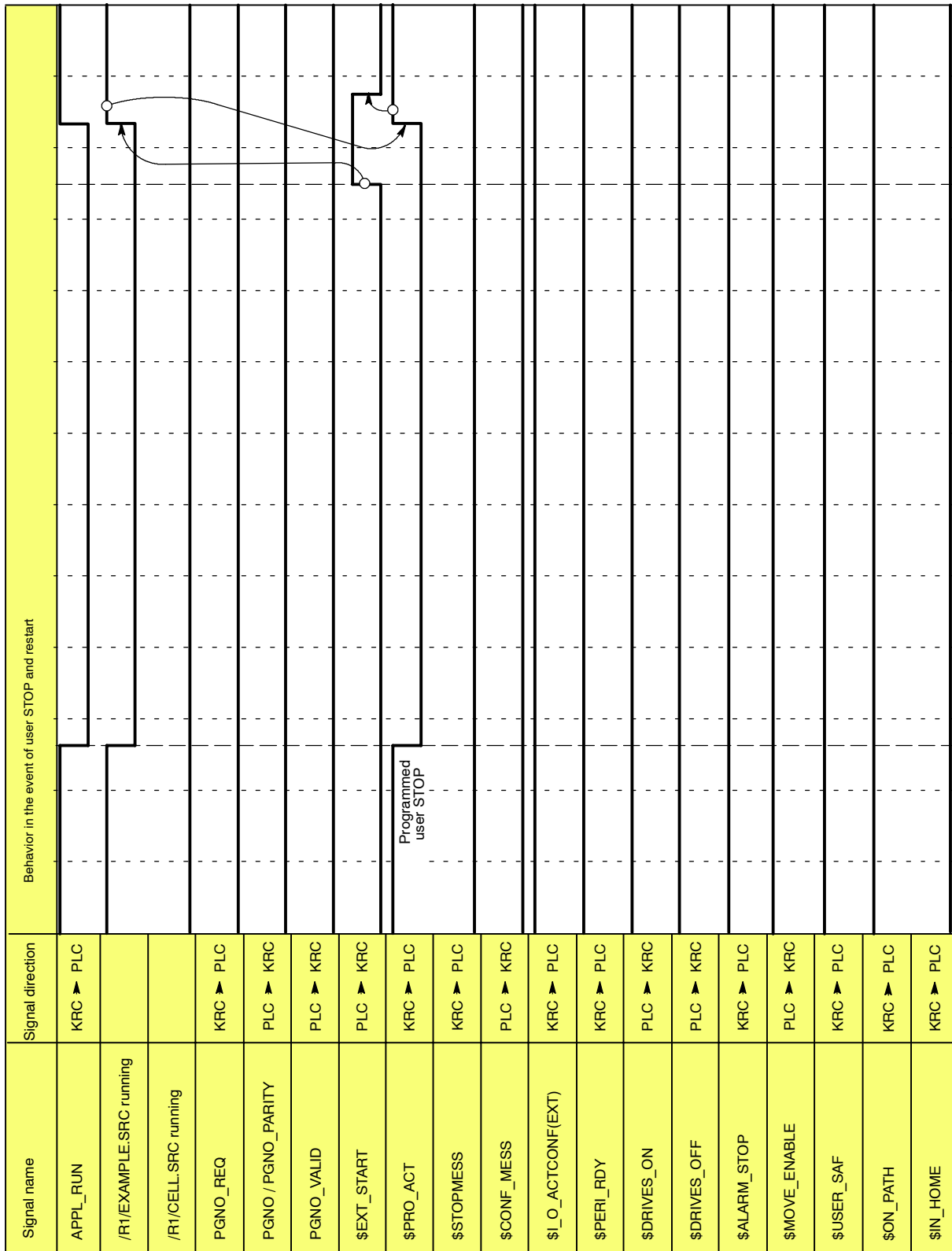
3.7.4 Restart after path-maintaining EMERGENCY STOP



3.7.5 Restart after motion enable



3.7.6 Restart after user STOP



3.8 Other

3.8.1 Restart after passive stop

If a passive stop is initiated from the KCP without a change of operating mode, the error message “Q1370: Passive STOP” must be acknowledged on the KCP. The program can then be continued with an external start.



In the event of a passive stop from the KCP and a change of operating mode, the robot must be manually repositioned.

3.8.2 Step by step program execution

Normally, only the program run mode “#GO” is allowed in “Automatic External” mode. In certain cases, the operating mode “#MSTEP” is available for step by step program execution.

For this, the following line must be modified in the file “C:\KRC\Roboter\Init\Progress.ini”:

```
[ FEATURES ]  
...  
MSTEP_IN_EXT = TRUE  
...
```

This option is then available after the next system reboot. You can toggle simply between the program run modes by pressing the status key.

3.8.3 Velocity for returning to the programmed path

If the robot has left the programmed path, it is moved back to the point at which this occurred with reduced velocity. The same applies if the robot executes a BCO run. The velocity at which this positioning is carried out corresponds with the manual traversing velocity.

A precondition for this is that the following line is modified in the file “C:\KRC\Roboter\Init\Progress.ini”:

```
[ FEATURES ]  
...  
SLOW_BCO_EXT = TRUE  
...
```

The variables \$RED_VEL_AXC[1]...[12], which can only be modified in the file “C:\KRC\Roboter\KRC\R1\MaDa\Machine.dat”, are required for reducing the axis traversing velocity.



Where possible, avoid modifying the variables “\$RED_VEL_AXC[1]...[12]. **No liability will be accepted for any damage or downtime resulting from modified variable values.**

3.9 Configuration example

3.9.1 Declarations

- The program number is to be sent as a binary number.
- The program number is 7 bits wide and is received at input 1.
- The parity bit is received at input 8 and checked for odd parity.
- The request for a new program number is indicated via the rising edge of the signal at output 1.
- The host computer communicates the presence of a program number with a rising edge at input 9.
- The fact that a program is running is communicated to the host computer via output 2.
- If the I/O interface is active, this is indicated to the host computer via output 3.
- The host computer uses input 10 to carry out an external start.
- A group error is communicated to the host computer via output 4.
- The host computer acknowledges errors via input 11.



Necessary entries in the file “C:\KRC\Roboter\KRC\R1\System\\$CONFIG.DAT” (configuration example)

PGNO=0	Pre-assignment of the program number
PGNO_TYPE=1	Data format of the program number: binary number
PGNO_FBIT=1	First bit of the program number: input 1
PGNO_LENGTH=7	Length of the program number: 7 bits
PGNO_PARITY=-8	Odd parity, parity bit at input 8
PGNO_REQ=1	Request for a new program number via output 1
PGNO_VALID=9	Confirmation that the program number has been transferred; sent to input 9
APPL_RUN=2	Notification that a program is being executed: output 2 set
PGNO_ERROR=0	Pre-assignment of the error flag

Entries in file \$MACHINE.DAT (configuration example)

\$EXT_START\$IN[10]	; External start
\$I_O_ACTCONF \$OUT[3]	; I/O interface active
\$STOPMESS \$OUT[4]	; Stop error
\$CONF_MESS \$IN[11]	; Group acknowledgement



Interface assignment (configuration example)

Controller	Signal name	Host computer
\$IN[1]	PGNO bit 1	A 20.0
\$IN[2]	PGNO bit 2	A 20.1
\$IN[3]	PGNO bit 3	A 20.2
\$IN[4]	PGNO bit 4	A 20.3
\$IN[5]	PGNO bit 5	A 20.4
\$IN[6]	PGNO bit 6	A 20.5
\$IN[7]	PGNO bit 7	A 20.6
\$IN[8]	PGNO_PARITY	A 20.7
\$IN[9]	PGNO_VALID	A 21.0
\$IN[10]	\$EXT_START	A 21.1
\$IN[11]	\$CONF_MESS	A 21.2
\$IN[12]	\$DRIVES_OFF	A 21.3
\$IN[13]	\$DRIVES_ON	A 21.4
\$IN[14]	\$MOVE_ENABLE	A 21.5
\$OUT[1]	PGNO_REQ	E 20.0
\$OUT[2]	APPL_RUN	E 20.1
\$OUT[3]	\$I_O_ACTCONF	E 20.2
\$OUT[4]	\$STOPMESS	E 20.3
\$OUT[5]	\$PERI_RDY	E 20.4
\$OUT[6]	\$PRO_ACT	E 20.5

3.10 Messages

This section contains a description of the error messages which can arise in conjunction with the "Automatic External" interface.

Message number	Message text	Cause
P00:1	PGNO_TYPE incorrect value permissible values (1,2,3)	The data type for the program number was entered incorrectly.
P00:2	PGNO_LENGTH incorrect value Range of values $1 \leq \text{PGNO_LENGTH} \leq 16$	The selected program number length in bits was too high.
P00:3	PGNO_LENGTH incorrect value permissible values (4,8,12,16)	If BCD format was selected for reading the program number, a corresponding number of bits must also be set.
P00:4	PGNO_FBIT incorrect value not in the \$IN range	The value "0" or a non-existent input was specified for the first bit of the program number.
P00:7	PGNO_REQ incorrect value not in the \$OUT range	The value "0" or a non-existent output was specified for the output via which the program number is to be requested.
P00:10	Transmission error incorrect parity	Discrepancy detected when checking parity. A transmission error must have occurred.
P00:11	Transmission error incorrect program number	A program number was sent by the host computer for which no branch for execution has (yet) been created in the CELL.SRC control structure.
P00:12	Transmission error incorrect BCD encoding	The attempt to read the program number in BCD format led to an invalid result.
P00:13	Incorrect operating mode	The I/O interface output has not been activated, i.e. the system variable \$I_O_ACTCONF currently has the value FALSE. This can have the following causes: The keyswitch is not in the "Ext." position. The signal \$I_O_ACT currently has the value FALSE.
P00:14	Move to Home position in operating mode T1	The robot has not reached the HOME position
P00:15	Incorrect program number	More than one input set with "1 of n".



NOTES:

Symbols

#INSIDE, 43, 55
#INSIDE_STOP, 43, 55
#OUTSIDE, 43, 55
#OUTSIDE_STOP, 44, 55
#PGNO_ACKN, 77
#PGNO_FAULT, 78
#PGNO_GET, 77
#STEP1, 69
\$_I_O_ACTCONF, 75
\$ADAP_ACC, 69
\$CONFIG.DAT, 93
\$CURR_LIM, 67
\$CURR_MAX, 67
\$CURR_RED[x,x], 65, 66, 67
\$CUSTOM.DAT, 69
\$EXT_START, 75, 87
\$EXT_START\$IN[,], 93
\$_I_O_ACTCONF, 75
\$_I_O_ACTCONF \$OUT[,], 93
\$IN[x], 40
\$INSIM_TBL[x], 40
\$IOBLK_EXT, 40
\$IOSIM_IN[,], 40
\$IOSIM_OPT, 40
\$IOSIM_OUT[,], 40
\$MACHINE.DAT, 93
\$NEARPATHTOL, 84
\$OUT[x], 40
\$OUT_NODRIVE, 40
\$OUTSIM_TBL[x], 40
\$POS_RET, 81, 84
\$PRO_I_O[,], 75
\$RED_T1, 65
\$ROBCOR.DAT, 69
\$STOPMESS \$OUT[,], 93
\$TOOL, 44
\$TORQ_DIFF, 70
\$TORQ_VEL[,], 66, 68
\$TORQMON_COM, 70
\$TORQMON_COM_DEF, 70
\$TORQMON_TIME, 69
\$TORQUE_AXIS, 66, 67

Numbers

5 home positions, 41

A

Access password, 21
Advance run stop, 68
ALARM_STOP, 83
APPL_RUN, 83, 93
ASCII, 22
AUT, 84
Automatic External, 9, 73
Automatic system start, 75
Axis 1 soft, 66
Axis 3 soft, 66
Axis with defined torque, 67

B

BCD value, 79
Binary number, 79

C

Ceiling-mounted, 64
CELL.SRC, 75, 76, 85
Change password, 21
CHCK_MOVENA, 81
Collision monitoring, 69
CONF_MESS, 82
Configurable output for hardware warnings, 37
"Configure" menu, 9
Configuring the interface, 73
Control cabinet external fan, 37
Cycle Time Optimizer, 26

D

Def-line, 22
Detail view, 22
Drivers, 9
DRIVES_OFF, 82
DRIVES_ON, 82
Dynamic braking, 88

E

Edit "ConfigMon.ini", 34
Edit I/O Config., 10
Editor, 22
EMERGENCY STOP, 89
Enabling switch, 40
ERR_FILE, 78
ERR_TO_PLC, 85

Event planner, 27
Example of torque mode application, 65
Expert, 16
EXT_ERR, 78
EXT_PGNO, 77, 78
EXT_PGNO.SRC, 85
EXT_START, 81
EXTERN, 84

F

Floor-mounted, 64
Focus, 73
Force cold startup, 19

G

General, 64

H

Hardware warning, 37
Host computer, 73
HOV, 15

I

I/O, 9
I/O Driver Reset, 10
I/O interface, 75
I/O simulation, 38
IN_HOME, 85
Inclination of the robot, 65
INSIDE_STOP, 44, 55
INT \$TORQUE_AXIS, 68
Interface assignment, 94

J

Jog override, 15
Jogging, 15

K

KRL, 16
KUKA long text database, 34

L

Language, 20
Linebreak, 23
Long text, 34

M

Macro assignment list, 34
Miscellaneous, 20
Monitoring tunnel, 69
Monitoring working envelope, 25
Motion enable, 90
Motor cable monitoring, 37
MOVE_ENABLE, 81

N

NEAR_POSRET, 84

O

Office GUI, 24
ON_PATH, 84
Operating mode Automatic External, 40
Operator safety, 88
OUTSIDE_STOP, 44, 55
Override, 15
Overriding the workspace monitoring, 45

P

P00 module, 77
Passive stop, 92
Password, 16
PC fan, 37
PERI_RDY, 83
Peripheral interfaces, 9
PGNO, 85, 93
PGNO_ERROR, 85, 93
PGNO_FBIT, 80, 93
PGNO_FBIT_REFL, 83
PGNO_LENGTH, 80, 93
PGNO_PARITY, 80, 93
PGNO_REQ, 77, 83, 93
PGNO_TYPE, 79, 93
PGNO_VALID, 80, 86
POV, 15
PRO_ACT, 85
Program number acknowledgement, 86
Program override, 15

R

Reaction, 64, 67
REAL \$CURR_ACT[12], 67
REAL \$CURR_RED[12,2], 67
REFLECT_PROG_NR, 80

Reinitialization, 26
Response time, 69
Restart, 88, 92
Restrictions, 64
Risks, 64

S

Sagging axes, 64
Signal descriptions, 79
Simulated inputs/outputs, 38
Speed controller output, 65
Status keys, 14
Step by step program execution, 92
STOPMESS, 83
Submit interpreter, 14

T

T1, 84
T2, 84
Technology-specific organization program, 76
Torque limits, 69
Torque mode, 64
Torque mode not possible, 64
Torque mode possible, 64

U

User, 15
User group, 15
User level, 16
User levels, 15
User STOP, 91
USER_SAF, 84

V

Variables for torque mode, 67
Velocity for returning, 92

W

WAIT FOR \$IN[], 66
Wall-mounted robots, 64
Workspace monitoring, 42
Wrist axes, 67
Wrist axis gear units, 64