User Manual

Content

1. The robot system

1.1 system overview

1.2 Robot description

1.3 Overview of KUKA system software

2. Safety

2.2 Workspace, safety zone, and danger zone

2.1 Mode selection

The different modes

3. Operation and coordinate system

4. Setup

4.1 PC setup

* SimPro
* WorkVisual

4.2 Mastering

4.3 Calibration

6. Robot Programming

6.1 Creating program modules

6.2 Editing program modules

6.3 KRL – guide

6.4 Motion programming

6.4.1 Axis-specific (PTP)

6.4.2 Continuous path (LIN and CIRC)

6.5 User Programming (inline form)

6.6 Expert Programming (KRL syntax)

7. Gripper Installation

**User Groups**

The robot controller offers various **user groups** with different functions. The following **user groups** can be selected.

* **Operator**

A user group for the operator. This is the default user group.

* **USER**

This option allows Start-up tasks such as mastering, tool calibration. It also enables the user to do simple application programs (programming using inline forms, motion commands). This prevent any syntax error

* **Expert**

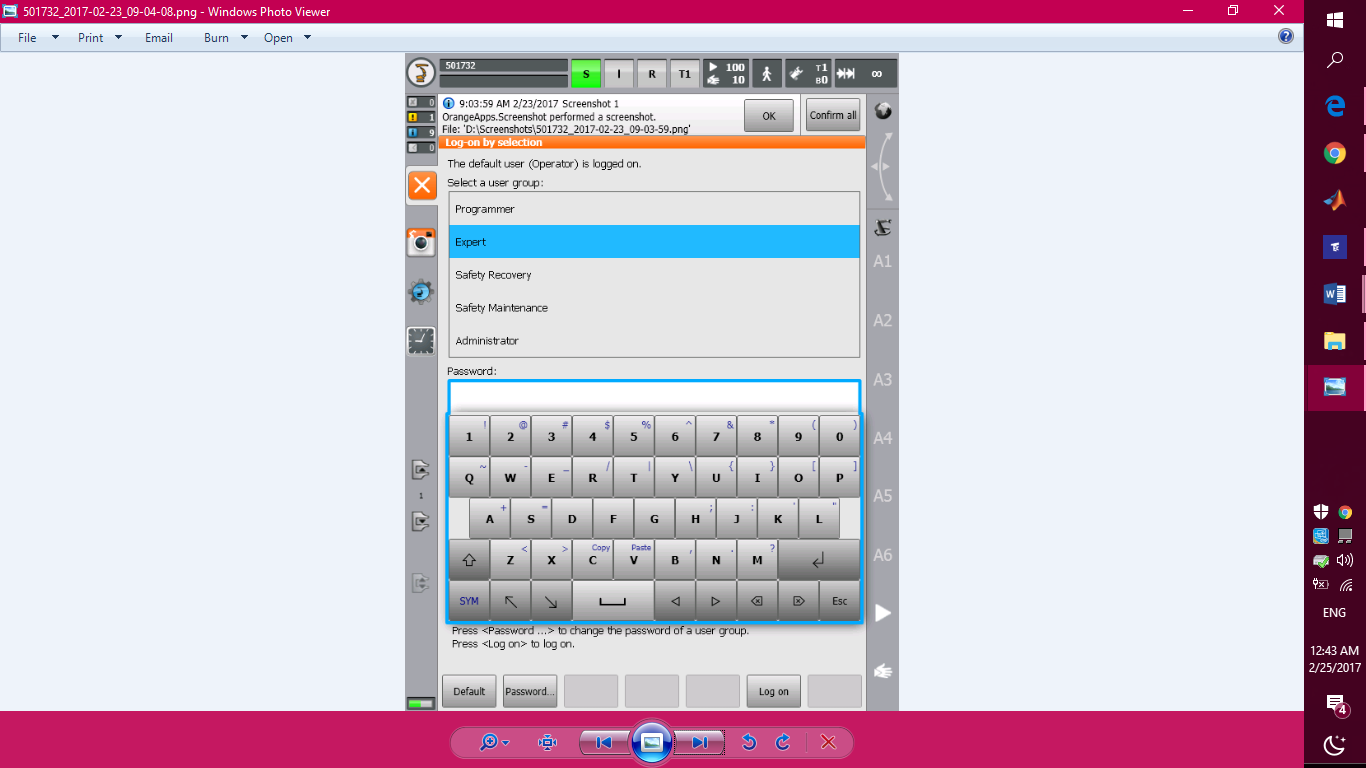
Using Expert mode can achieve advanced programming using the KRL programming language and perform complex application programs including subprograms, interrupt programming, loops, and program branches.

Expert password: **kuka**

* **Administrator**

Configuration of the robot controller (external axes, technology packages) and configuration of the robot system (field buses, vision systems)

User-defined technology commands with UserTECH



**Coordinate systems in conjunction with robots:**

The following Cartesian coordinate systems are defined in the robot controller:

.1 **WORLD coordinate system**: Fixed, rectangular coordinate system whose origin is located at the base of the robot. It is the root coordinate system for the ROBROOT and BASE coordinate systems.

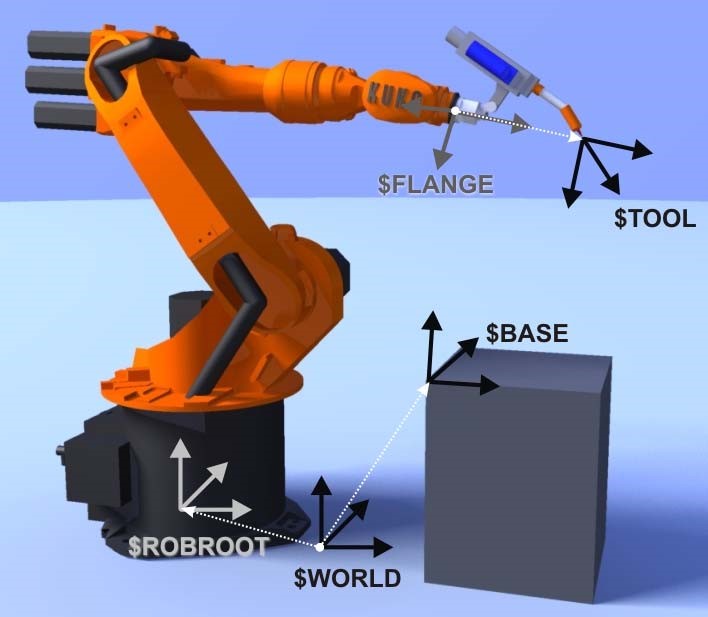
By default, the WORLD coordinate system is located at the robot base.

.2 **ROBROOT coordinate system**: Cartesian coordinate system, the robot base. It defines the position of the robot relative to the WORLD coordinate system.

By default, the ROBROOT coordinate system is identical to the WORLD coordinate system. **$ROBROOT** allows the definition of an offset of the robot relative to the WORLD coordinate system.

.3 **BASE coordinate system**: a Cartesian coordinate system that defines the position of the workpiece. It is relative to the WORLD coordinate system. It's is offset to the workpiece by the user (explained in "BASE calibration").

.4 **TOOL coordinate system: a** Cartesian coordinate system which is located at the tool center by default, the origin of the TOOL coordinate system is located at the flange center point. The TOOL coordinate system is offset to the tool center point by the user



**Starting up the robot**

**Overview**

* **start-up mode**
* **Robot mastering**
* **Tool Calibration**
* **Base Calibration**

**Displaying the current robot position**

**Options for displaying current robot position**

* **Axis-specific**

Displays the actual position of the robot in an axis-specific way. The current angle for every axis displayed is relative to the zero position of the axis.

* **Cartesian**

The current position of the current TCP (tool coordinate system) is displayed relative to the currently selected base coordinate system.

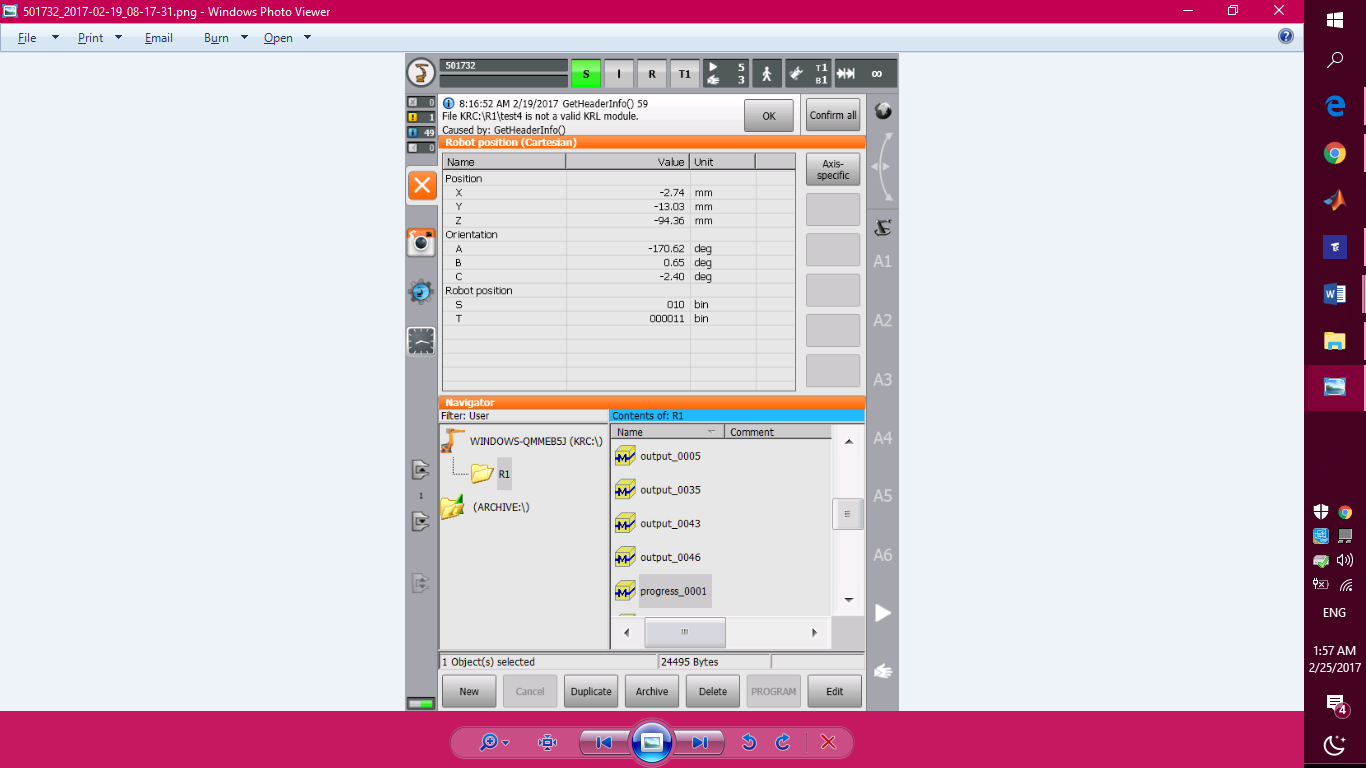
If no tool coordinate system is selected, the flange coordinate system applies!

If no base coordinate system is selected, the world coordinate system applies!

**Procedure for displaying current robot position**

1. Select **Display** > **Actual position** in the menu. The Cartesian actual position is displayed.
2. To display the axis-specific actual position, press **Axis spec.**
3. To display the Cartesian actual position again, press **Cartesian**.

Current position of the robot tool can be relative to different base coordinate system

 For selecting amongst different tools and base coordinate system

|  |  |
| --- | --- |
|  |  |
| Cartesian | **Axis-specific** |

**Working with program files**

**Creating program modules**

Using Expert group configuration, create a new module in file **KRC: \R1\.**

1. In the directory structured, select the folder in which the program to be created.
2. Press the "**New**" soft key.
3. Select "**Module**" template.
4. Enter a name for the program then confirm it with "**OK**".

|  |
| --- |
|  |
| Module structure:   * SRC file: contains the source code for the program * DAT file: contains the permanent data and point coordinates |

**Editing** **program modules**

All programs can be edited in the navigator of the KUKA smart Pad

Editing tasks include:

1. Duplicate
2. copy
3. cut
4. delete
5. Rename

The procedure applied is almost the same in all editing options.

1. Select the file/program.
2. Select Edit soft key.
3. Choose the required action
4. In the case of selecting "Delete", confirm with "OK".

**KRL – guide**

KRC 4 controller uses **KRL** KUKA programming language.

**Variables and declarations:**

All system variables start with '$' sign, mind not starting any "user-defined" name with this sign to avoid syntax errors.

**Names in KRL:**

* Can have a maximum length of 24 characters.
* Can consist of letters (A - Z), numbers (0 - 9) and the special characters '\_' and '$'.
* Must not begin with a number.
* Must not be a keyword.

**Declaration and initialization of variables**

* Variables (simple and complex) must be declared in the SRC file before the INI line and initialized after the INI line.
* Variables can optionally also be declared and initialized in a local or global data list.
* Every variable is linked to specific data type.
* The data type must be declared before use.
* The keyword for the declaration is **DECL**. It can be omitted in case of the four simple data type
* In order to place syntax before the INI line, the DEF line must be activated:

Open file > Edit > View > DEF line

DEF program\_name ( )

**DECL** data type user\_defined\_variable ; declaration section of variables

INI

; Initialization section of user\_defined\_variables.

…

; Instruction Section

…

END

**Simple Data types**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Data Type | Integer | Real | Boolean | Character |
| Keyword | INT | REAL | BOOL | CHAR |
| Meaning | Integer | Floating point number | Logic state | 1 character |
| Range | -… -1 | ±1.1E-38 … ±3.4E+38 | TRUE, FALSE | ASCII character |
| Example | 2 | 4.23 | TRUE | c |

**Structure types**

* **AXIS:** A1 to A6 are angle values (rotational axes) or translation values (translational axes)
* **FRAME:** X, Y, and Z are space coordinates, while A, B, and C are the orientation of the coordinate system.
* **POS and E6POS**: S (Status) and T (Turn) define axis positions unambiguously

|  |  |  |
| --- | --- | --- |
| **System Variables:** | **Data Type** | **Purpose/Description** |
| $AXIS\_ACT = { A1…,A2… ,A3 …,A4 … ,A5… ,A6… ,E1…, …,E6} | structure | Displays the actual position of the robot in axis-specific method. The current angle for every axis displayed is relative to the zero position of the axis |
| $POS\_ACT = { X…,Y…,Z…,A…,B…,C…,S…,T…,E1…,…} | Structure | Displays the actual position of the robot in the Cartesian method. The current TCP position is displayed relative to the currently selected base coordinate |
| $OV\_PRO | integer | Set the velocity of robot for automatic program (program override) sequence |
| $APO.CDIS | integer | Distance parameter: Approximation starts at the earliest when distance to the end point falls to the value assigned to the variable |
| $BASE | structure | Offset and orientation of base coordinate system relative to the world coordinate system |
| $TOOL | structure | Tool coordinate system relative to flange coordinate system |
| $H\_POS | structure | Home position of the robot |
|  |  |  |
|  |  |  |
|  |  |  |

**Motion programming**

**Motion types**

The robot can move in various motion types. Paths are created according to the operation of each axis. Thus, the robot can be controlled to create either linear or circular path.

* Axis-specific motion (PTP: point-to-point)
* CP motions: LIN (linear) and CIRC(circular)

**Axis-specific motion**

The robot guides the TCP along the **fastest** path to the end point. The fastest path is generally not the shortest path and is thus not a straight line. The first motion in the program must be PTP as status and turns are only evaluated here.

The coordinates of the end point are absolute.

* **Characteristics:**

1. smooth motion
2. Robot can move from start to end singularity free. As long as both the starting and ending points are in the working envelope, the robot will get to the end point without collision or sudden movement.
3. Control is much simpler than continuous path control.

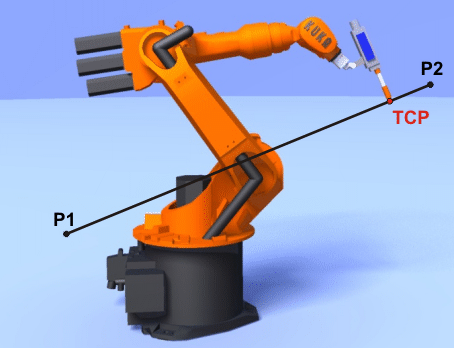
**Applications**: Pick-and-place applications, spot welding, etc.



**CP motion**

**LIN – Linear –Motion**

Motion at a defined velocity and acceleration along a straight line.  This motion requires the programmer to “teach” one point.  The robot uses the point defined in the previous move as the start point and the point defined in the current command as the end point and interpolates a straight line in between the two points.



**CIRC – Circular – Motion**

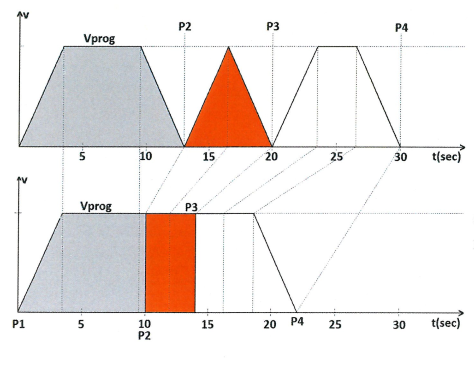
Motion at a defined velocity and acceleration along a circular path or a portion of a circular path.  This motion requires the programmer to “teach” two points, the mid-point and the end point.  Using the start point of the robot (defined as the end point in the previous motion command) the robot interpolates a circular path through the mid-point and to the end point



**Approximate Positioning**

Approximate positioning of motion means that the next programmed point will not be exactly reached. This can help to shorten cycle times.

Speed profile if all points approached exactly

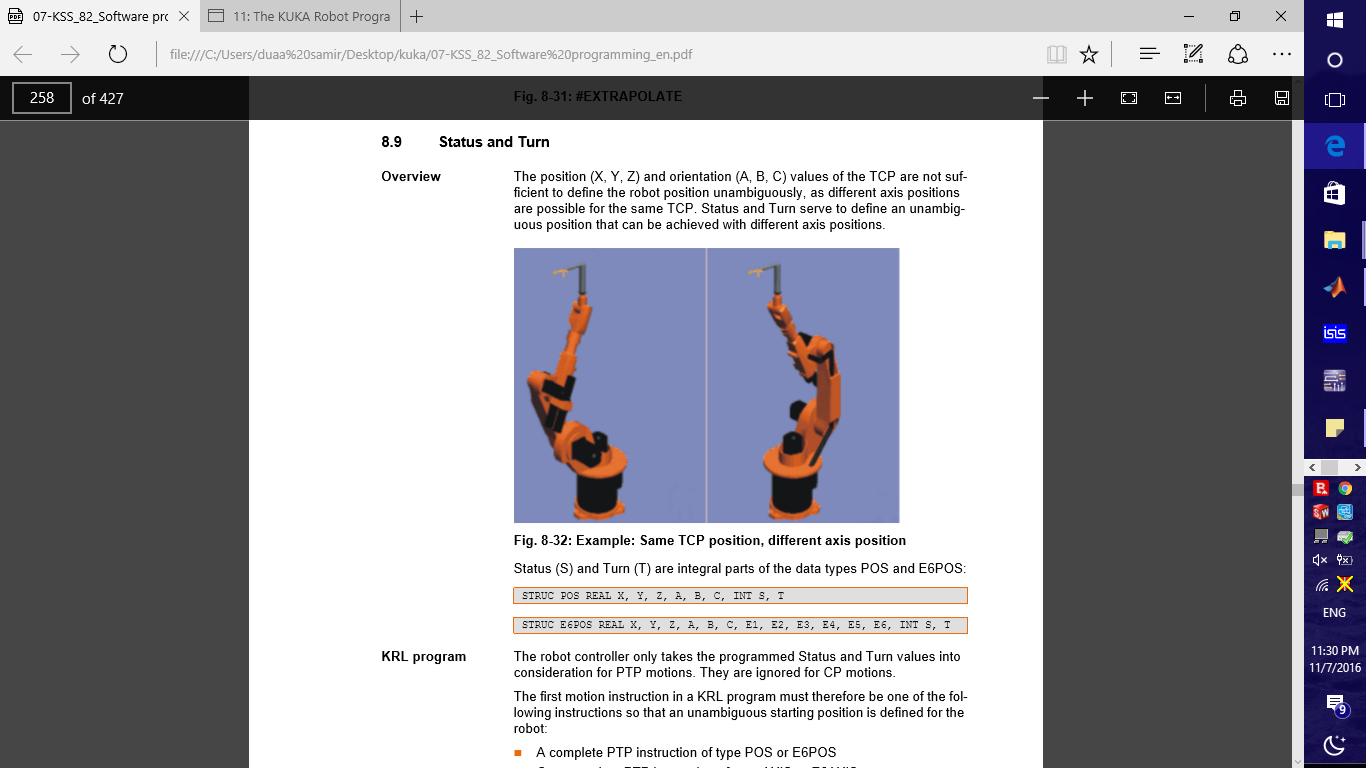


Speed profile in case of approximate positioning of the points

**Status and Turns**

The position of x,y,z and orientation A,B,C values of TCP are not sufficient to define the robot position ,as different axis positioning are possible for the same TCP .

Status and turns serve to define the position that can be achieved with different axis positions.



Same TCP, different axis position

**Singularities**

KUKA robots with 6 degrees of freedom have 3 different singularity positions.

A singularity position is characterized by the fact that unambiguous reverse

Transformation (conversion of Cartesian coordinates to axis-specific values) is not possible, even though Status and Turn are specified. In this case, or if very slight Cartesian changes cause very large changes to the axis angles, one speaks of singularity positions.

**Overhead singularity**

In the overhead singularity, the wrist root point (intersection of axes A4, A5 and A6) is located vertically above axis 1.

The position of axis A1 cannot be determined unambiguously by means of reverse transformation and can thus take any value. If the end point of a PTP motion is situated in this overhead singularity position, the robot controller may react as follows by means of the system variable

$SINGUL\_POS[1]:

N 0: The angle for axis A1 is defined as 0 degrees (default setting).

N 1: The angle for axis A1 remains the same from the start point to the end point.

**Extended position singularity**

In the extended position singularity, the wrist root point (intersection of axes

A4, A5 and A6) is located in the extension of axes A2 and A3 of the robot.

The robot is at the limit of its work envelope.

Although reverse transformation does provide unambiguous axis angles, low

Cartesian velocities result in high axis velocities for axes A2 and A3.

* Wrist axis singularity
* In the wrist axis singularity position, the axes A4 and A6 are parallel to one another and axis A5 is within the range ±0.01812°.

The position of the two axes cannot be determined unambiguously by reverse transformation. There is an infinite number of possible axis positions for axes A4 and A6 with identical axis angle sums.

**Gripper Installation**

**Product Description**



GP404NC – C gripper

2- Jaw Parallel Pneumatic Gripper GP404NC – C with maximum stroke 4 mm per jaw.

**Gripper connection**

**Procedure**

1. Connect the output from the compressor to AIR 1 in A1 interface.
2. Connect 2A - XPN41 connection to the left air supply of the gripper.

**Gripper Operation Manually**

**Procedure**

1. In the main menu, select **Display**> **inputs/outputs > Digital I/O**
2. Select output NO.4, and press **Value** to open gripper
3. Select output NO.1, and press **Value** to close gripper

**Gripper operation with KUKA.GripperTech**

KUKA.GripperTech is a technology package installed on KSS – KUKA system software- to facilitate the use of a gripper in programming, configuration, and operator control.

**Gripper configuration**

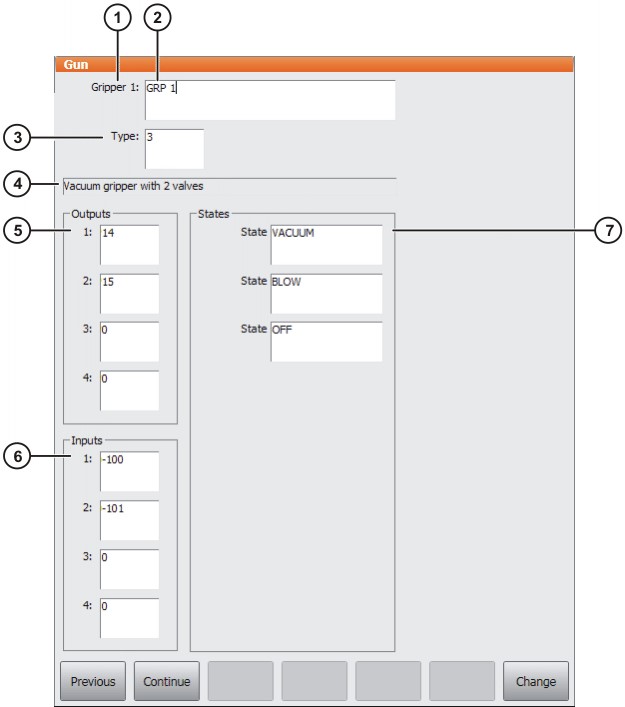
There are 5 predefined gripper types in GripperTech. If these types are not sufficient, additional gripper functions can be programmed.In our case, we are going to use **Type 1.**

**Type 1** described as Single-element gripper, static, open/closed**.** It has 2 states: Opened, Closed

**Procedure**

1. In the main menu, select **Configure** > **I/O** > **Gripper**. A window opens.
2. In the main menu, select **Configure** > **I/O** > **Gripper**. A window opens.
3. Assign one of the types 1 to 5 to the gripper.
4. Assign the inputs and outputs
5. Save the configuration with **Change**

If desired, you may change default names of the states or gripper name



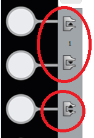
Predefined gripper configuration

|  |  |
| --- | --- |
| Description | Item No. |
| Number of the gripper  1 … 16 | 1 |
| Name of the gripper  The name is displayed in the inline form. The default name can be changed.  1 … 24 characters | 2 |
| Type  For predefined grippers: 1 … 5 (See “Gripper types” table) | 3 |
| Designation of the gripper type (not updated until saved) The designation cannot be changed. | 4 |
| Assignment of the output numbers  Outputs that are not required can be assigned the value “0”. In this way, they are immediately identifiable as unused. If they are nonetheless assigned a number, this has no effect. | 5 |
| Assignment of the input numbers  Inputs that are not required can be assigned the value “0”. In this way, they are immediately identifiable as unused. If they are nonetheless assigned a number, this has no effect. | 6 |
| Assignment of the input numbers  Inputs that are not required can be assigned the value “0”. In this way, they are immediately identifiable as unused. If they are nonetheless assigned a number, this has no effect. | 7 |

**Status Keys**

Status keys used to operate the gripper manually using the smart pad. These buttons are shown in fig - - .

**Procedure**



Status Keys on smart pad

* + 1. Select the gripper using the status key.
    2. Activate operating mode T1 or T2.
    3. Press enabling switch.
    4. Control the gripper using the status key.

Gripper datasheet:

1. <http://rva-tw.com/wp-content/uploads/2013/05/s_gp400_gb1.pdf>
2. <http://www.sommerautomatic.com/store/media/pdf/GP400.pdf>

References:

1. KSS KUKA Robot Programming 1 Training -KSS8\_v4.
2. KST-Expert\_Programming\_manual KSS 5.2.
3. Notes\_ Bochum KUKA Programming.
4. KRL Quickguide Syntax 8x.
5. 07-KSS\_82\_Software programming.
6. KST\_GripperSpotTech\_32.
7. Spez\_KR\_AGILUS\_sixx\_CR.
8. <http://javakuka.com/xyzabc>
9. [www.globalrobots.com](http://www.globalrobots.com)