



URCap installation and operating instructions – part 5.1

Table of contents

1	General information	4
1.1	Foreword	4
2	Safety instructions	4
2.1	Design of safety and warning information	4
3	Connection.....	5
4	URCap	7
4.1	Preparing to install the URCap	7
4.2	Installing the URCap	7
4.3	Enabling the URCap	9
4.4	Configuration	10
4.5	UR+ control	13
4.6	Programming.....	14
4.7	Program examples	15
4.8	Timer operation	16
4.9	Sensor operation.....	17
4.10	Expert programming – UR script commands	18
4.11	Error	21

1 General information

1.1 Foreword

The installation and operating instructions describe important information on the installation and operation of the URCap. The URCap was developed exclusively for Universal Robots e-series robots (UR3e, UR5e, UR10e, UR16e and UR20).

1.1.1 Target group

The installation instructions are part of the mk product and are intended for all persons responsible for setting up and operating the URCap. They must be read and understood by these people. In addition to these installation and operating instructions, the original operating instructions for the UR robot, the mk product and the supplier's documentation enclosed with it must be observed.

1.1.2 Safekeeping

These installation and operating instructions must always be kept close to the mk product and in a legible condition.

2 Safety instructions

2.1 Design of safety and warning information

The operator of the mk product must ensure that all safety instructions and warnings are observed and complied with. Warnings are prefaced by signal words that convey the extent of the hazard.

These operating instructions differentiate between the following hazard levels:



CAUTION

Hazard that may lead to minor or moderate injuries if the safety information is not followed.



ATTENTION

Hazard that may lead to product damage if the safety information is not followed.



NOTE

Information that helps you to better understand the work steps.

Part 5.1 – URCap installation instructions

The individual wires of the connection cable can be connected as desired to the outputs of the controller that are still free.

Wire	Frequency inverter input	Output UR robot	Signal
Black, 1	Digital input 1	Free digital output (configurable output (CO0-CO7) or digital output (DO0-DO7))	Start/stop
Black, 2	Digital input 2	Free digital output (configurable output (CO0-CO7) or digital output (DO0-DO7))	Direction (direction of transport)
Black, 3	Analogue input 1	Free analogue output (analogue output 0 (AO0) or analogue output 1 (AO1))	Speed (conveying speed)
Black, 4	0 V (GND)	Free digital input (configurable output (0 V) or digital output (0 V))	Reference potential
Green and yellow	-	-	-



NOTE

- Write down the selected outputs. They are assigned to the actual connection assignment later in the URCap.

4 URCap



NOTE

The robot software (Polyscope) must have at least software version 5.11 or higher for the URCap to function correctly.

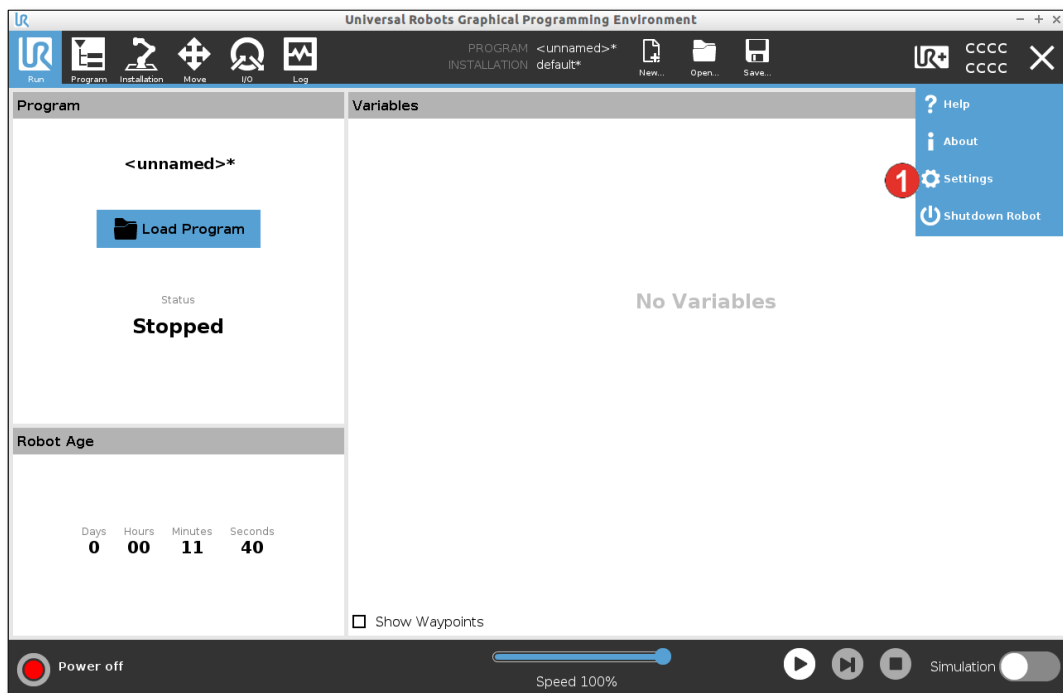
- The latest version of the robot software is available at: www.universal-robots.com/download/
- The robot arm does not have to be switched on for the URCap installation.

4.1 Preparing to install the URCap

- Download the latest version of the mk product URCap from the webpage [UR Solutions | mk Technology Group \(mk-group.com\)](http://UR Solutions | mk Technology Group (mk-group.com)) and save it to a USB stick.
- With the system on, connect the USB stick to the USB connection on the teach panel.

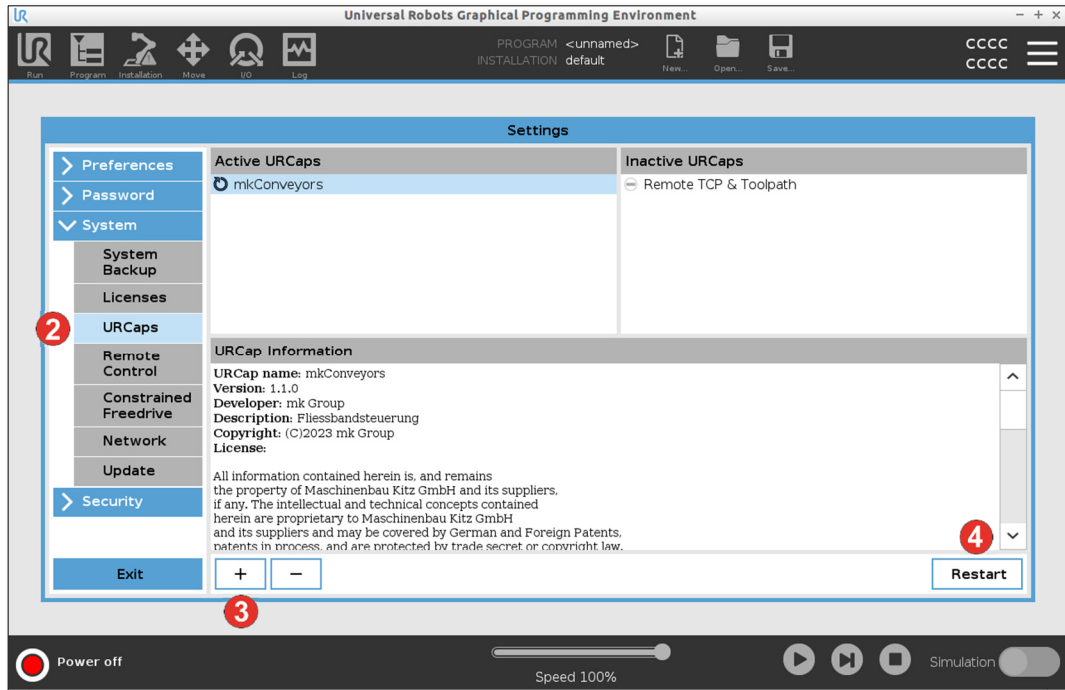
4.2 Installing the URCap

- In the main menu, **1** click on “Settings”.



Part 5.1 – URCap installation instructions

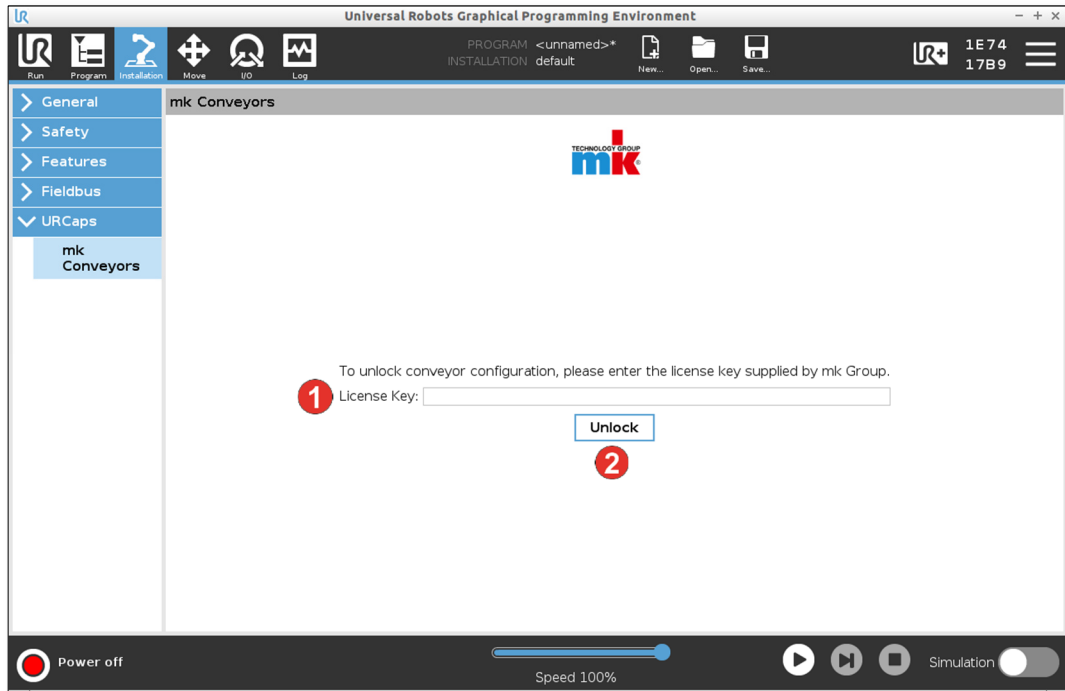
- In the new window, click on “System” ⇒ “URCaps” **2**.
- Click on “+” **3** to install the new URCap.
- On the USB stick, select the URCap “mkConveyors-1.X.X.urcap” for installation.
- Click “Restart” **4** button.



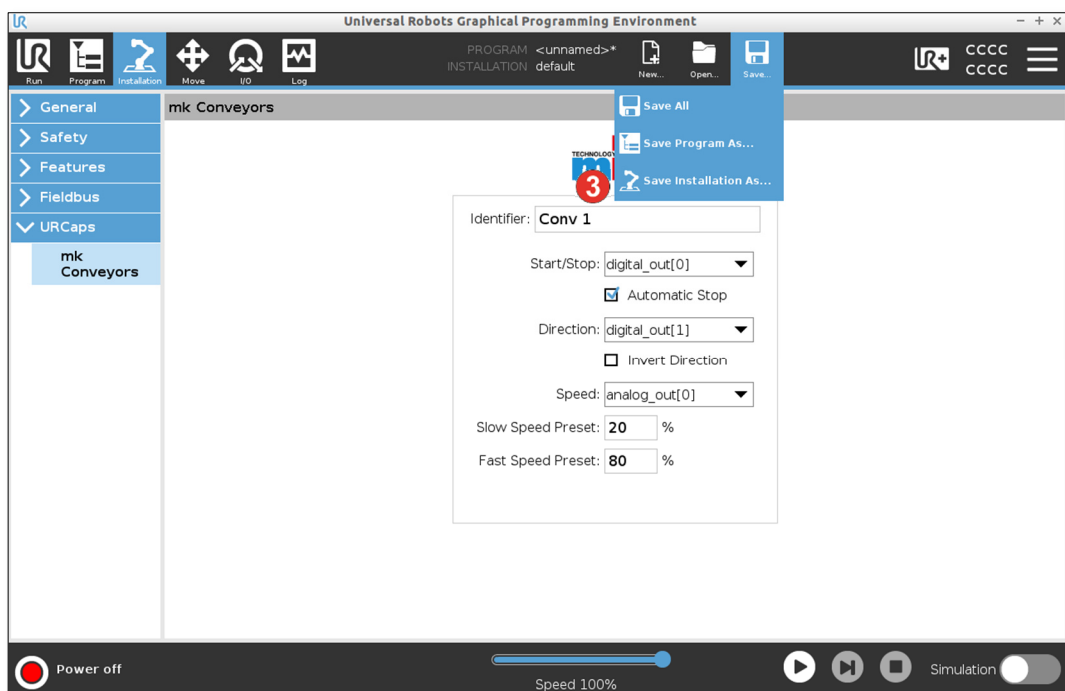
After the restart, the URCap “mkConveyors-1.X.X.urcap” is installed. After the initial configuration, the URCap can be used in the programming environment.

4.3 Enabling the URCap

- Enter the machine number of the mk product into the **1** “License Key” text field.
- Then **2** click on “Unlock”.



- Click on “Save...” ⇒ “Save Installation” **3** to save the installation.



4.4 Configuration



NOTE

- The mk product must be electrically connected to the controller (see the chapter “Connection”).
- The URCap must be installed (see the chapter “Installing the URCap”).

4.4.1 Renaming the inputs and outputs



NOTE

Renaming the inputs and outputs is not mandatory. If no custom designation is assigned, the I/Os will continue to have their default label.

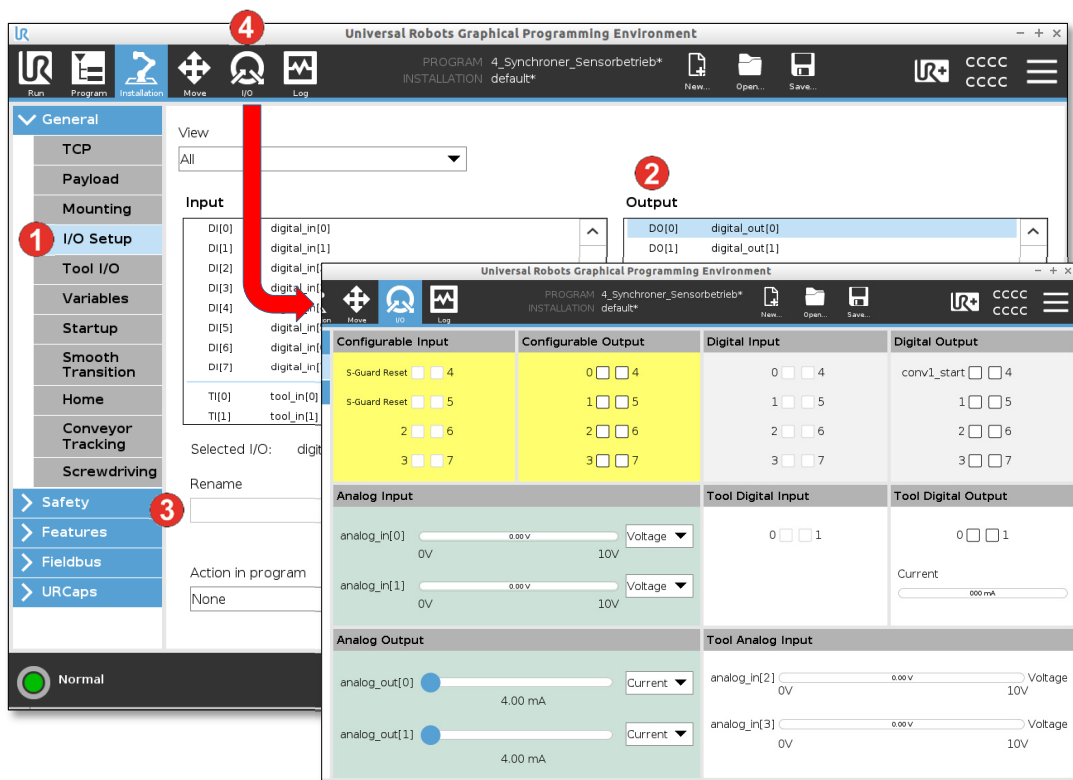
The inputs and outputs used can be labelled individually.

- Click on “Installation” ⇒ “General” ⇒ “I/O Setup” **1**.
- For “Output” **2** select the input or output to be renamed.
- Enter the new name in the “Rename” text field **3**.



NOTE

Users can click on “I/O” **4** to check all of the entries.



4.4.2 Completing the configuration

- Click on “Installation” ⇒ “URCaps” ⇒ “mk Conveyors” **5**.
- In the “Start/Stop” field **6**, select the output for the start/stop signal (depending on the signal cable connection).



NOTE

If “Automatic Stop” **7** is checked, the mk product remains standing after

- the program terminates,
- an interruption,
- or there is a collision with the robot.

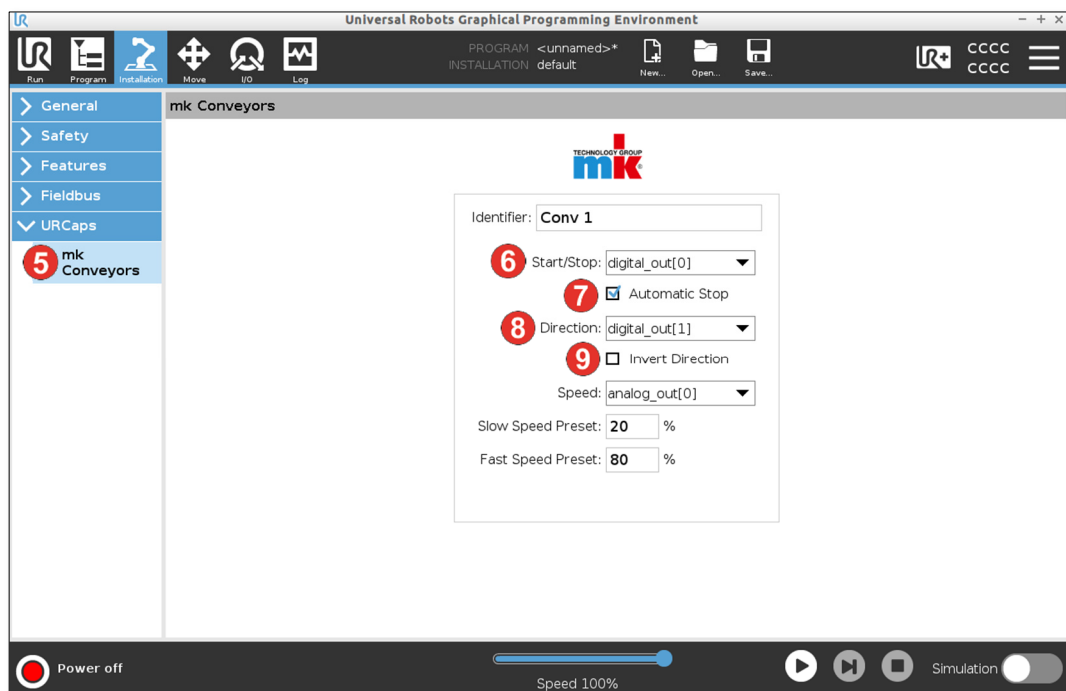
If “Automatic Stop” is not checked, the mk product continues to run.

- In the “Direction” field **8**, select the output for the conveying direction (depending on the signal cable connection).



NOTE

If the mk product does not run in the desired direction, users can invert the direction by placing a check beside “Invert Direction” **9**.



Part 5.1 – URCap installation instructions

- In the “Speed” field **10**, select the analogue output for the speed.



NOTE

For slow and fast speeds, users can enter a percentage value in the “**Slow Speed Preset**” and “**Fast Speed Preset**” field **11**, which is later used as a preset in the programs. The percentage value refers to the maximum conveying speed as the basis.

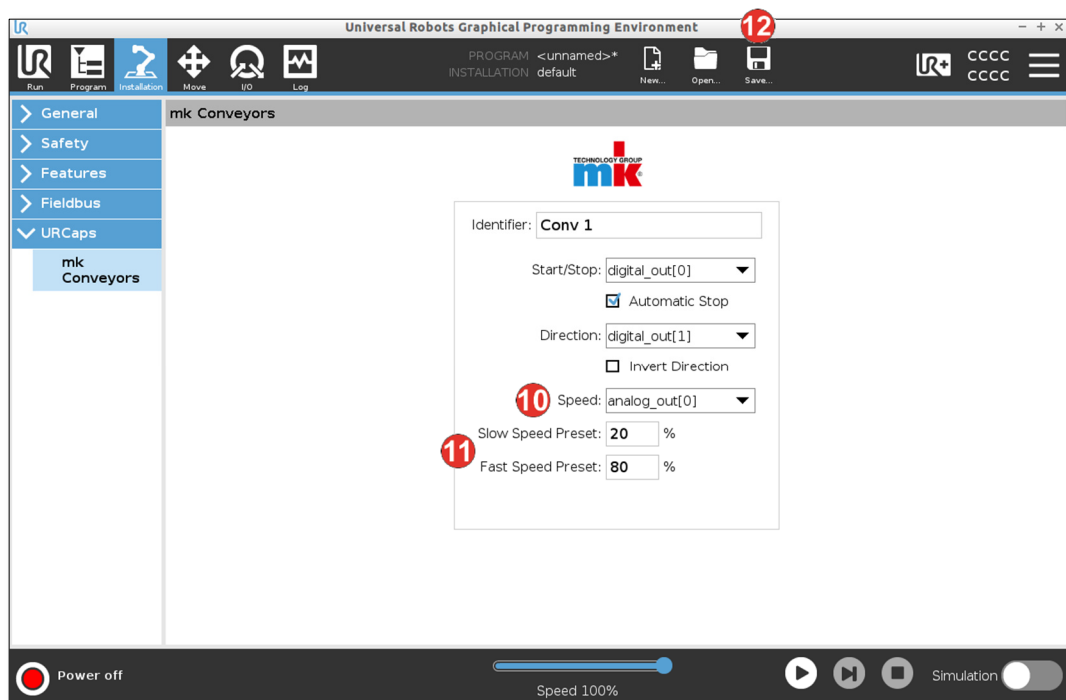
The values 20% and 80% are preset and can be changed individually.

- Click on “Save...” **12** to save the configuration.



NOTE

If users make changes to a configuration, programs that have already been created must be checked again and adjusted, if necessary.



4.5 UR+ control



CAUTION

Risk of injury!

If the mk product continues running after the UR+ control has been terminated, this can result in minor to moderate injury.

If a check mark is set for “**Continuous Operation**” **1**, the mk product continues to run even after the UR+ control has been closed.

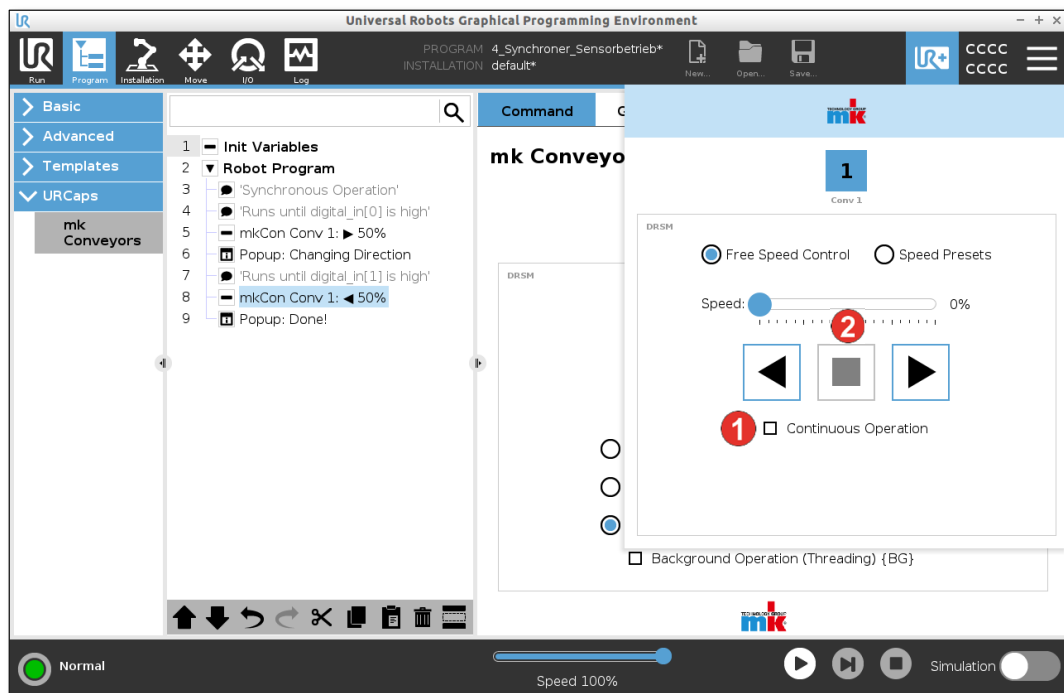
- Set the check mark for “**Continuous Operation**” **1** only if there is appropriate protection.

The mk product can be stopped by clicking on **2** with the UR+ control open.

- In an emergency, the frequency inverter’s main emergency stop must be actuated.

In the UR+ control, the most important functions can also be tested without a robot program.

- Click the symbol  to launch the UR+ control.



4.6 Programming



CAUTION

Risk of injury!

If the mk product continues running after the program has been terminated and this is not intended, this can result in minor to moderate injury.

- Be sure to turn off the mk product in the event of an emergency stop and unforeseen program stops.



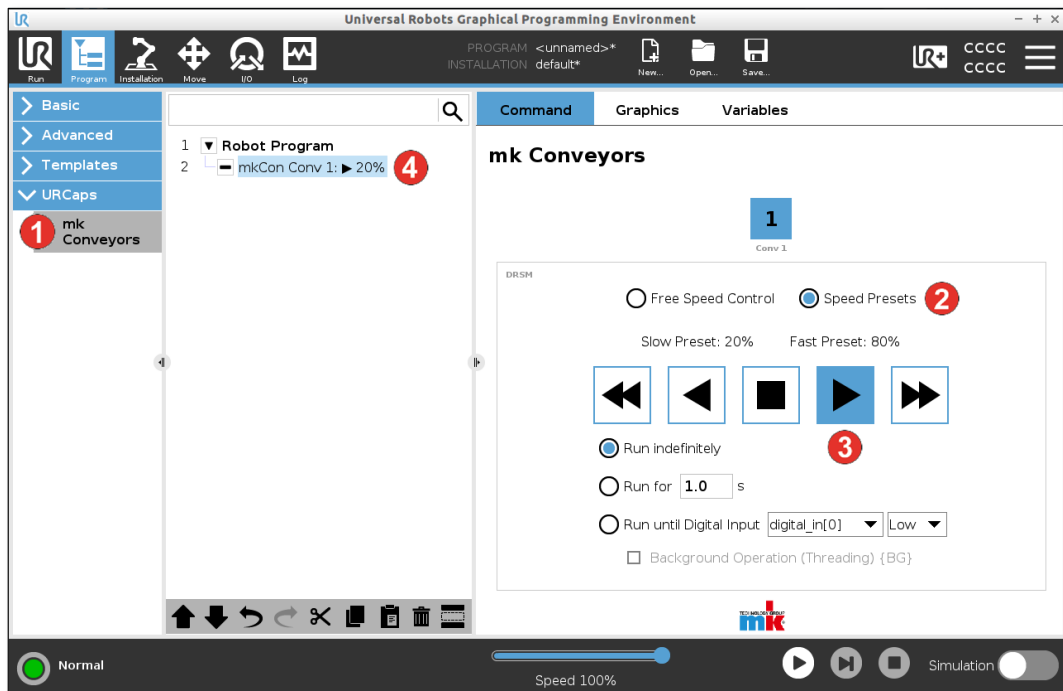
NOTE

- Before the end of the program, the mk product must be stopped using an appropriate command.
- The mk product must be electrically connected to the controller (see the chapter “**Connection**”).
- The URcap must be installed (see the chapter “**Installing the URcap**”).
- The robot arm must be activated (robot status is green).

- Click on “**Program**” ⇒ “**URCaps**” ⇒ “**mk Conveyors**” **1**.

Users can set up the robot sequence program with various functions by selecting several. For example, clicking on “**Speed Preset**” **2** and **▶** **3** makes the display “**mkCon Conv1: 20%**” **4** appear in the program tree.

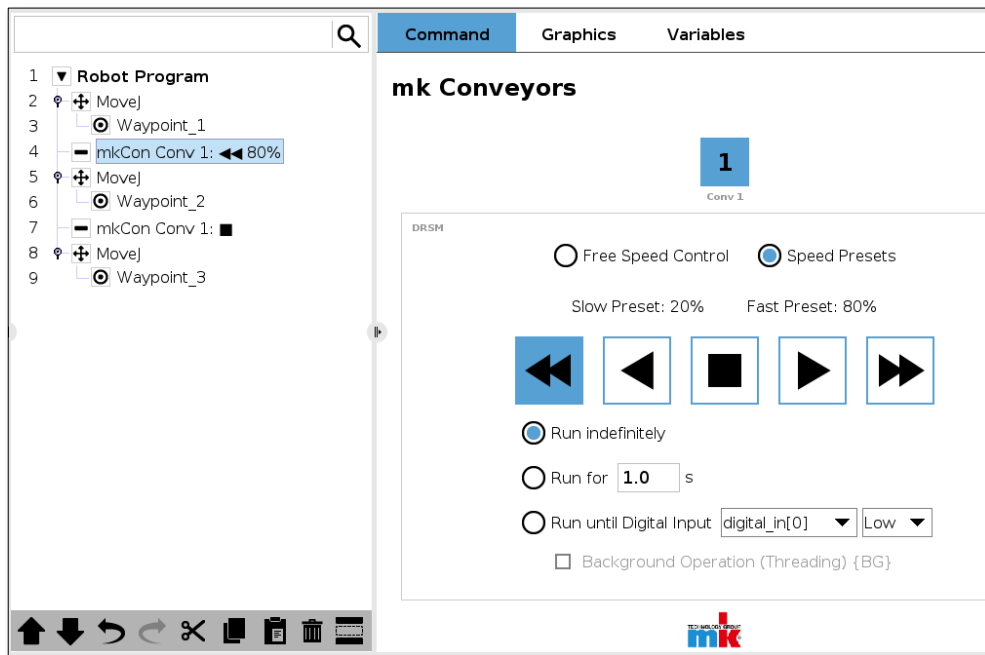
The mk product moves slowly to the right from this program point.



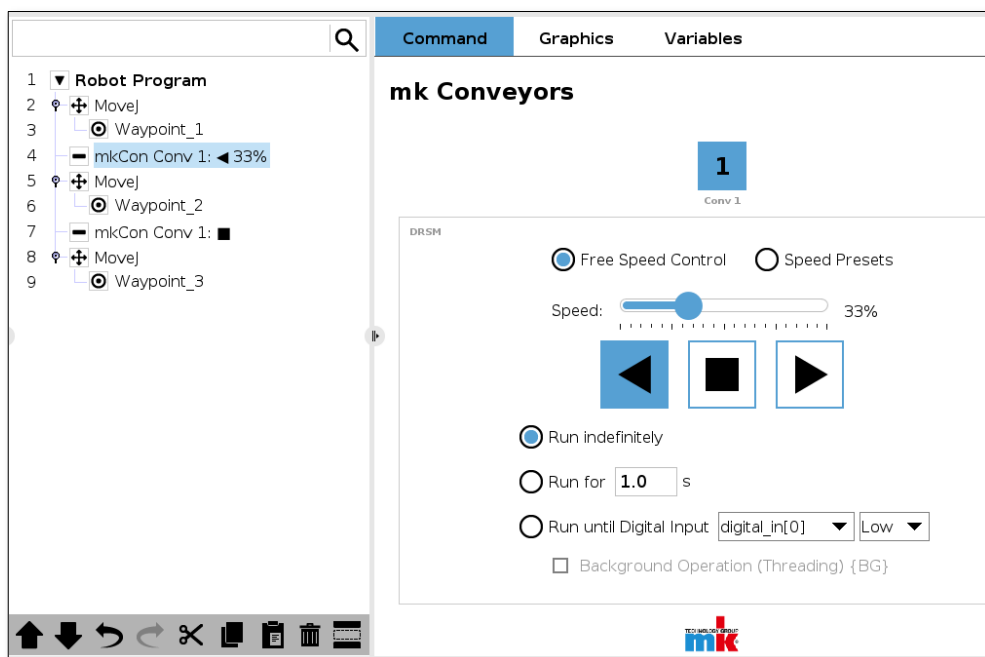
4.7 Program examples

4.7.1 Normal operation

In the following program example, the mk product is switched on after reaching waypoint 1 (fast anti-clockwise rotation, predefined to 80%). The robot moves to waypoint 2, the mk product is stopped, and the robot moves to waypoint 3.

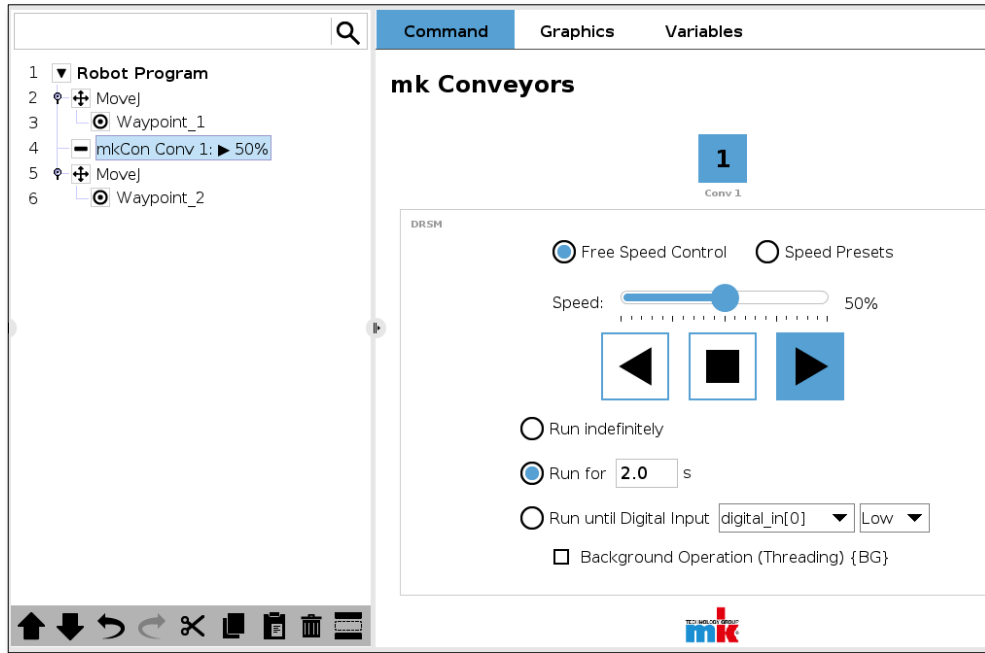


Same function as above, only anti-clockwise rotation with free speed setting to 33%.

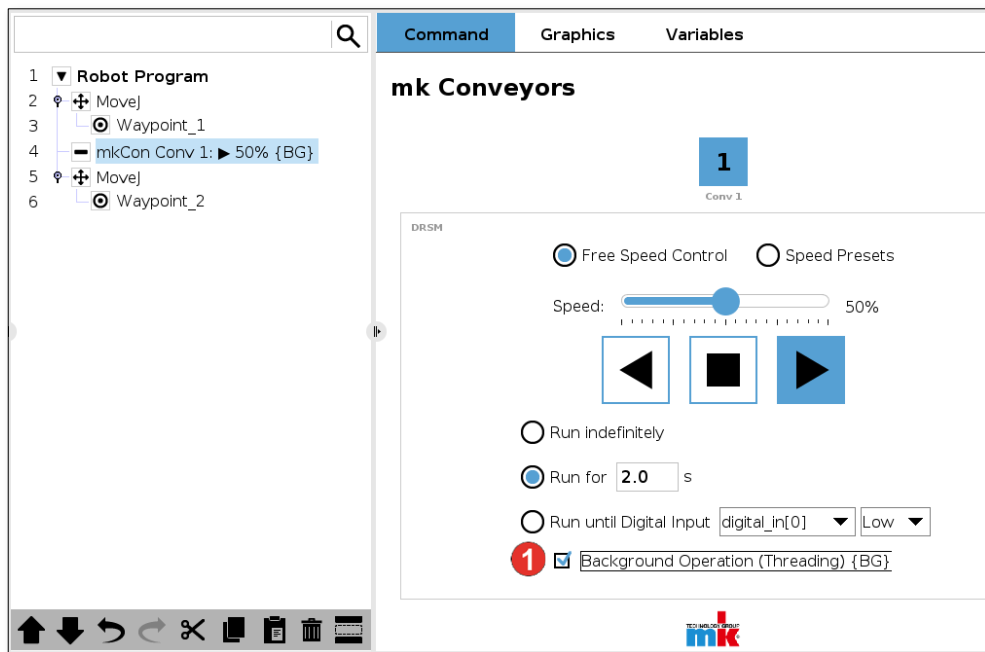


4.8 Timer operation

In the following program example, the mk product is switched on after reaching waypoint 1 (clockwise rotation with free speed setting to 50%). The robot waits 2 seconds, turns off the mk product and moves to waypoint 2.

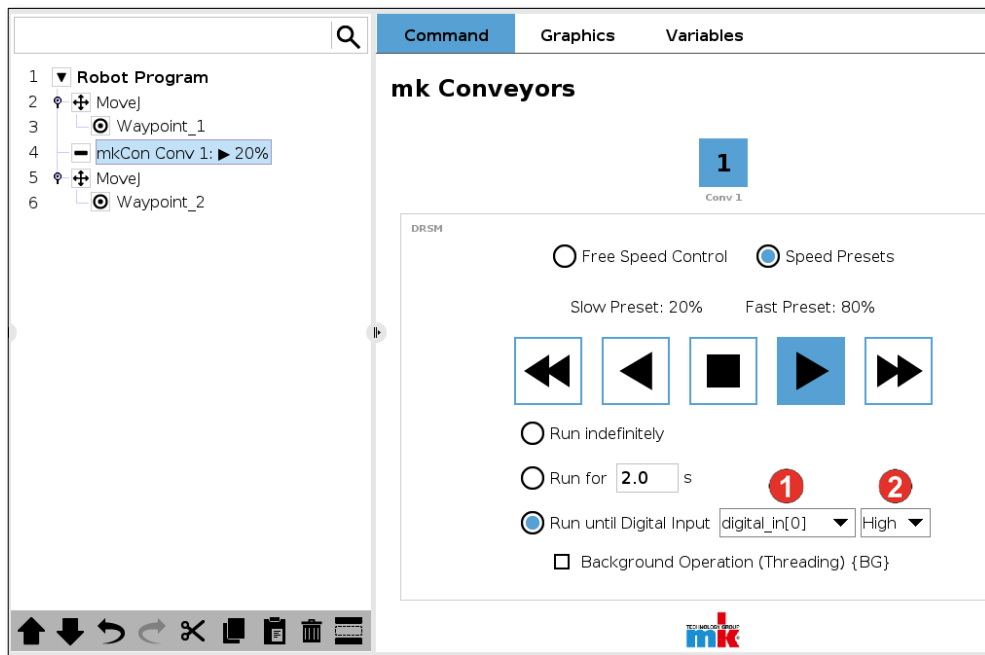


When the “**Background Operation**” function ① is selected, the mk product is started for 2 seconds after reaching waypoint 1, but the robot continues to travel to waypoint 2 without waiting.

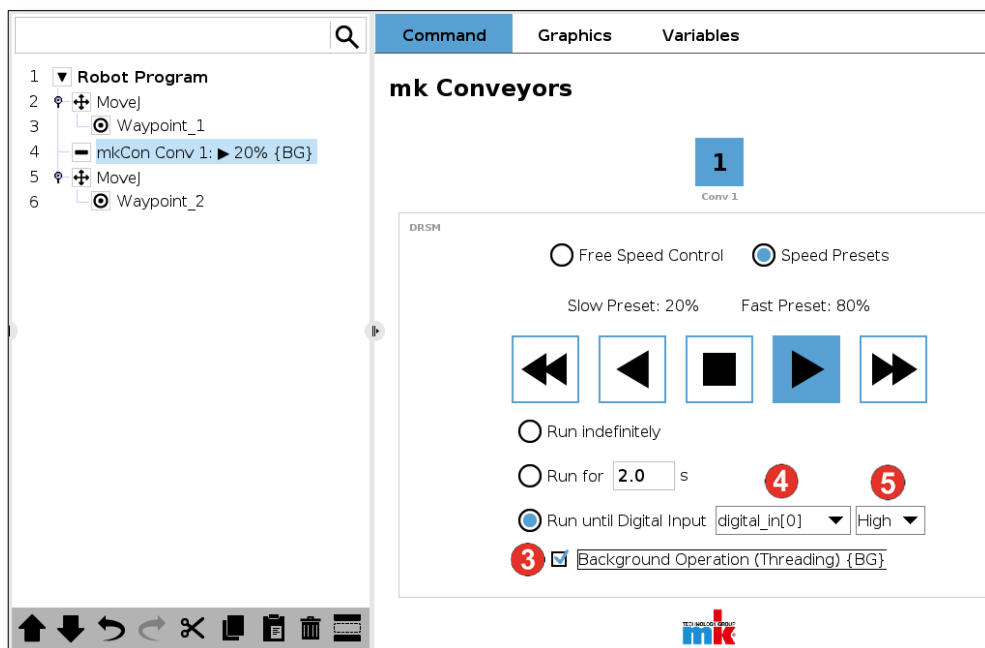


4.9 Sensor operation

In the following program example, the mk product is switched on after reaching waypoint 1 (slow clockwise rotation, predefined to 20%). The robot waits until the sensor signal “DI[0]” **1** changes to “High” **2**, turns off the mk product and moves to waypoint 2.



When the “Background Operation” function **3** is selected, the mk product is started after waypoint 1 is reached until the sensor signal “DI[0]” **4** changes to “High” **5**. However, the robot continues to waypoint 2 without waiting.



4.10 Expert programming – UR script commands

4.10.1 Normal operation

mk1_run_forward(speed)

Starts the mk product in forward mode at the specified speed.

Parameter: `speed` (the relative speed of the mk product [0;1](float))

Example: `mk1_run_forward(0.8)`

Example parameters: `speed = 0.8 (80%)`

mk1_run_backward(speed)

Starts the mk product in reverse mode at the specified speed.

Parameter: `speed` (the relative speed of the mk product [0;1](float))

Example: `mk1_run_backward(0.2)`

Example parameters: `speed = 0.2 (20%)`

mk1_stop

Stops the mk product in normal operation and prematurely when the sensor or timer is operated in concurrent mode.

Example: `mk1_stop()`

4.10.2 Timer operation

mk1_run_fwd_timer(speed, millis, inBackground)

Starts the mk product in forward mode with the specified speed and duration. After the specified time has elapsed, the mk product stops automatically. The parameter `inBackground` determines whether this operation is performed concurrently.

`inBackground=False` Synchronous/blocking operation. The next command is executed only after the timer has expired.

`inBackground=True` Concurrent operation. The next command is executed immediately. The timer is running in a thread in the background. The automatically generated global variable `mk1ConRunning` can be used to determine whether the mk product is still active.

Parameter: `speed` (the relative speed of the mk product [0;1](float))

`millis` (the duration of operation [ms](integer))

`inBackground` (concurrent execution [True, False](boolean))

Example: `mk1_run_fwd_timer(0.5, 2000, False)`

Example parameters: speed = 0.5 (50%)
 millis = 2000 (2 seconds)
 inBackground = False (synchronous operation)

mk1_run_fwd_timer(speed, millis, inBackground)

Starts the mk product in reverse mode with the specified speed and duration. After the specified time has elapsed, the mk product stops automatically. The parameter `inBackground` determines whether this operation is performed concurrently.

`inBackground=False` Synchronous/blocking operation. The next command is executed only after the timer has expired.

`inBackground=True` Concurrent operation. The next command is executed immediately. The timer is running in a thread in the background. The automatically generated global variable `mk1ConRunning` can be used to determine whether the mk product is still active.

Parameter: speed (the relative speed of the mk product [0;1](float))
 millis (the duration of operation [ms](integer))
 inBackground (concurrent execution [True, False](boolean))

Example: `mk1_run_bwd_timer(0.3, 10000, True)`

Example parameters: speed = 0.3 (30%)
 millis = 10000 (10 seconds)
 inBackground = True (concurrent operation)

4.10.3 Sensor operation

mk1_run_fwd_sensor(speed, port, inputType, signalLevel, inBackground)

Starts the mk product in forward mode at the specified speed. The mk product runs until the expected signal level is at the specified input. The parameter `inBackground` determines whether this operation is performed concurrently.

`inBackground=False` Synchronous/blocking operation. The next command is not executed until the expected signal level is applied to the input.

`inBackground=True` Concurrent operation. The next command is executed immediately. The wait for the signal level at the input is executed in a background thread. The automatically generated global variable `mk1ConRunning` can be used to determine whether the mk product is still active.

Parameter: `speed` (the relative speed of the mk product [0;1](float))

`inputType` 0 = standard digital input
1 = configurable digital input
2 = tool digital input

`port` (the ID of the input. Range of values depends on `inputType`)

- Standard digital input: [0:7] (integer)
- Configurable digital input [0:7] (integer)
- Tool digital input [0:1] (integer)

`signalLevel` (the signal level that initiates the mk product stop (True = high, False = low))

`inBackground` (concurrent execution [True, False](boolean))

Example: `mk1_run_fwd_sensor(0.8, 0, 2, True, False)`

Example parameters:

`speed` = 0.8 (80%)

`inputType` = 0 (standard digital input)

`port` = 2 (`digital_in[2]`)

`signalLevel` = True (waiting for high signal)

`inBackground` = False (synchronous operation)

mk1_run_bwd_sensor(speed, port, inputType, signalLevel, inBackground)

Starts the mk product in reverse mode at the specified speed. The mk product runs until the expected signal level is at the specified input. The parameter `inBackground` determines whether this operation is performed concurrently.

`inBackground=False` Synchronous/blocking operation. The next command is not executed until the expected signal level is applied to the input.

`inBackground=True` Concurrent operation. The next command is executed immediately. The wait for the signal level at the input is executed in a background thread. The automatically generated global variable `mk1ConRunning` can be used to determine whether the mk product is still active.

Parameter: `speed` (the relative speed of the mk product [0;1](float))

`inputType` 0 = standard digital input
1 = configurable digital input
2 = tool digital input

`port` (the ID of the input. Range of values depends on `inputType`)

- Standard digital input: [0:7] (integer)
- Configurable digital input [0:7] (integer)
- Tool digital input [0:1] (integer)

`signalLevel` (the signal level that initiates the mk product stop (True = high, False = low))

`inBackground` (concurrent execution [True, False](boolean))

Example: `mk1_run_bwd_sensor(0.6, 2, 1, True, False)`

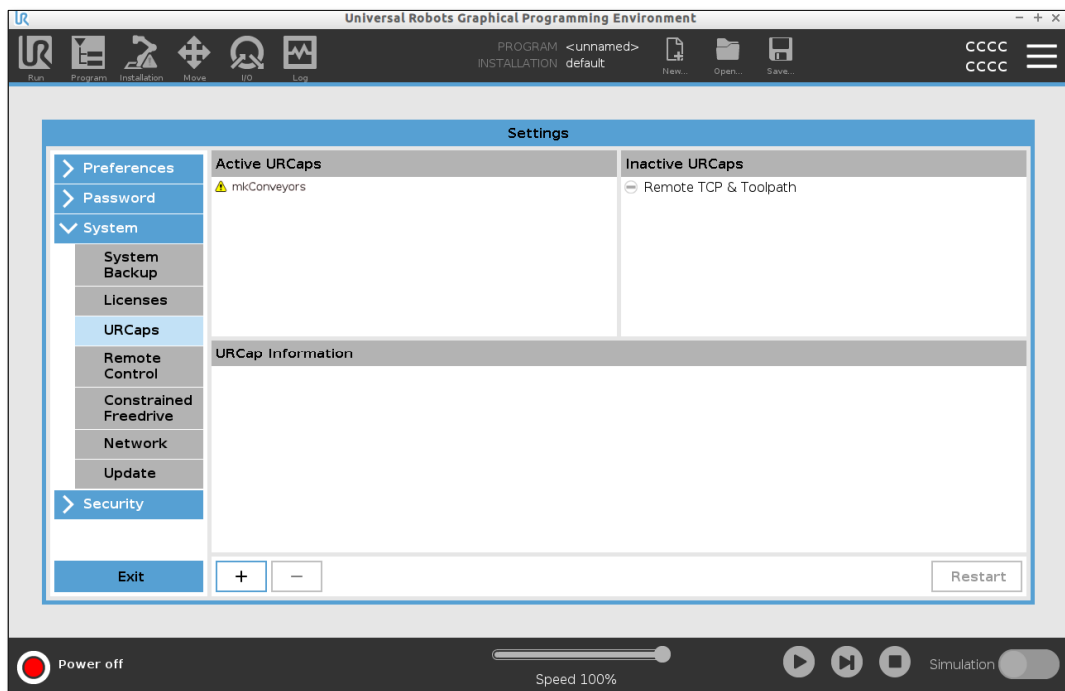
Example parameters:
 speed = 0.6 (60%)
 inputType = 2 (tool digital input)
 port = 1 (digital_in[1])
 signalLevel = True (waiting for low signal)
 inBackground = True (concurrent operation)

4.11 Error

After installation, the URCap is not active.

An old or incorrect version of Polyscope earlier than version 5.11 is being used. The URCap can be used only with Polyscope 5.11 or later.

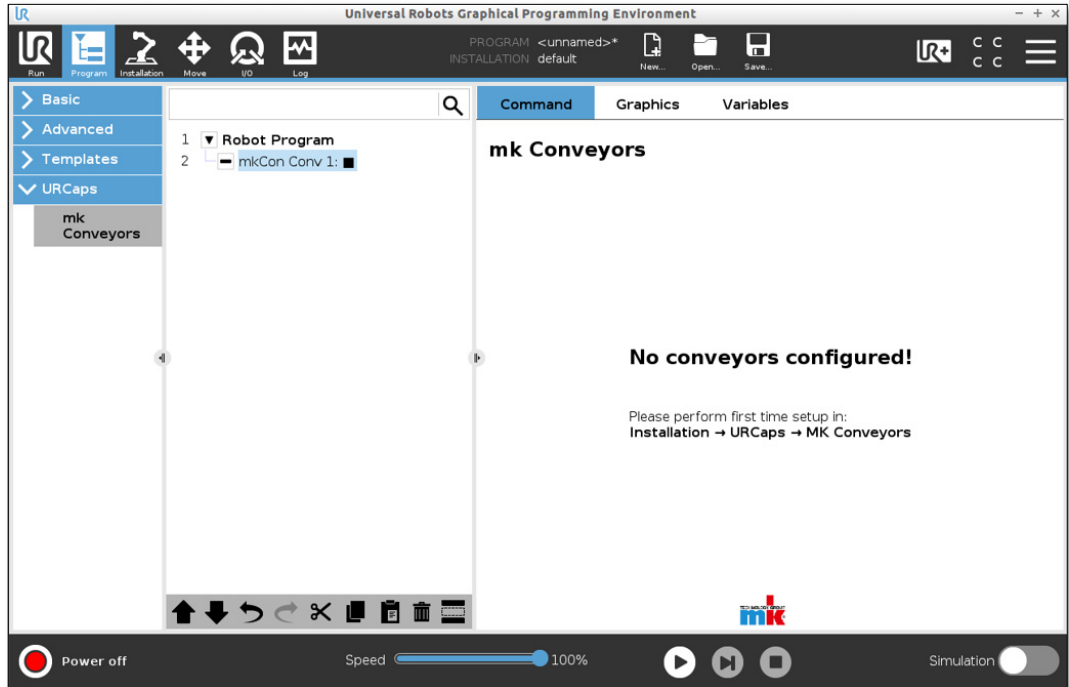
- Update the robot software to be able to use the URCap.



Part 5.1 – URCap installation instructions

“No conveyors configured”

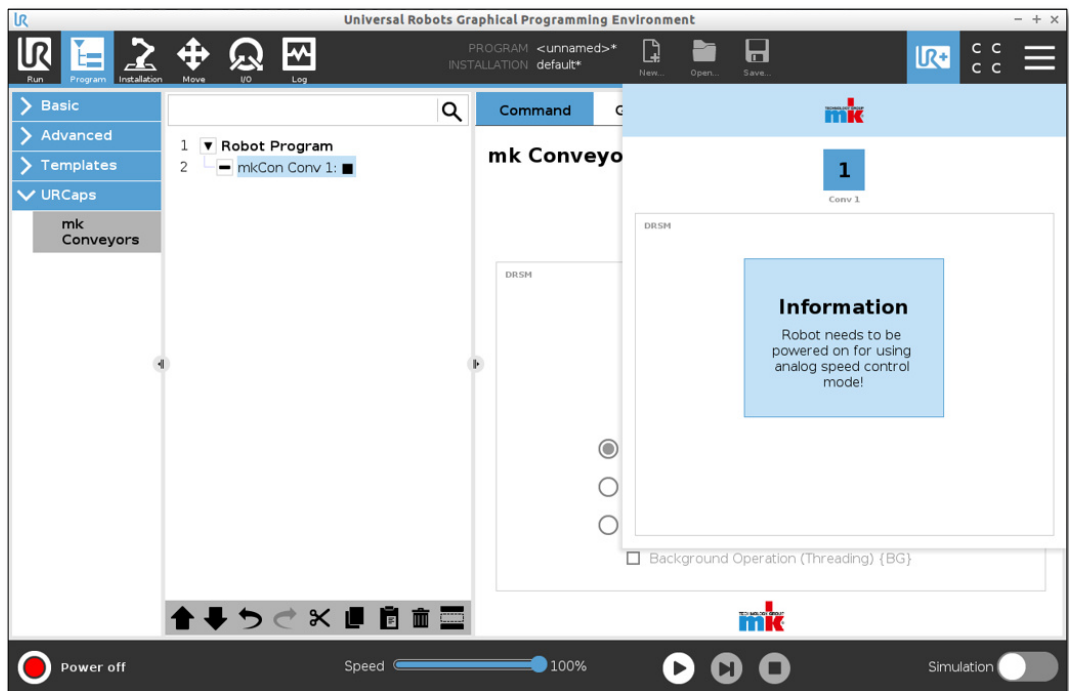
- Configuring the URCap (see chapter “Configuration”)



“Information – robot needs to be powered on”

The robot is not switched on yet.

- Switch on the robot.



“Warning – Multiple use of same IO”

The same outputs cannot be used for the signals “Start/Stop” and “Direction”.

- Check the outputs and change them, if necessary.

