FESTO MPS Robot Station

Project Documentation

Bernhard Müllner (1225855)

July 26, 2018

The aim of this project is to set up and generate demo projects for the FESTO MPS Robot station at the TiLab. The documentation of the project starts with a short introduction and description of the FESTO MPS Robot station, followed by an introduction of setting up a project for simulating the robot in CIROS Studio 6. Additionally, there is given the source code of the two demo projects. The first one assembles a work piece of 4 parts as known from a production facility, the other one assembles and disassembles the same work piece in a loop.

Contents

1 FESTO MPS Robot Station

In this first section of the documentation, the hardware in the TiLab used for this project is described. It introduces the robot station and its building parts, followed by a description of the work pieces which can be handled by the robot.

The robot station consists of three FESTO modules:

- Mitsubishi robot and Controller
- Robot assembly module
- Robot handling module

Figure 1 shows where these parts are located.

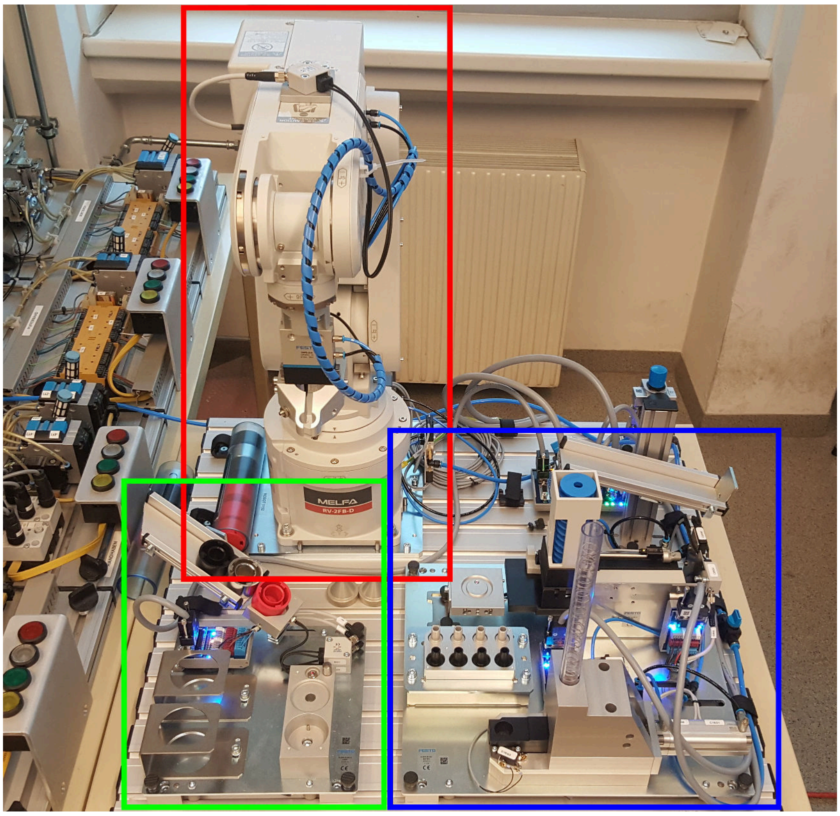


Figure 1: Mitsubishi robot, Robot handling module, Robot assembly module

1.1 Mitsubishi Robot And Controller

The robot is a 6 axis industrial robot. It executes programs stored on the controller and written in MELFA Basic V. For handling the work pieces (see section 1.4), a pneumatic gripper (see section 1.1.3) with a simple optical sensor is attached to the robot. For connecting the sensors and actuators from the two FESTO modules, an external I/O extender (figure 2) was added. The robot can be used without a program in manual operation mode, using the Teaching

Box (TB) (see section 1.1.2). This is useful to check possible movements and to gather exact positions of work pieces and tools.

Table 1 and table 2 show all inputs and respectively all outputs connected to the I/O extender. The sensors and actuators are described in section 1.2 and in section 1.3:

Input	Name
3	Sensor4HoleInBottom
4	PartAvailable
8	SpringPistonBack
9	SpringPistonFront
10	SpringAvailable
12	CapPistonBack
13	CapPistonFront
15	CapAvailable
900	ColorSensor

Table 1: Connected Inputs

Output	Name
8	SpringPiston
12	CapPiston
900	HOpen1
901	HClose1

Table 2: Connected Outputs

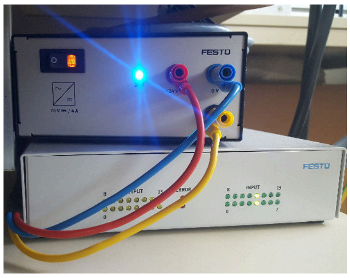


Figure 2: I/O Extender

1.1.1 Front Panel On Controller

The front panel is mainly used to start and stop a stored program on the controller. Therefore one has to switch on the power with the *Power Enable* switch (see figure 3), disable the Teaching Box by switching of the *TB Enable Switch* (see figure 7) , turn the *Manual to Automatic Switch* to "Automatic" (figure 3), turn on the servo motors (see figure 4) and finally start the program by pressing the *Start* button (figure 4).

To use a PC connected through the network with the controller, one can also follow the above steps, however, one need not to start the servo motors and the program.

To use the robot with the Teaching Box, one has to power-on the controller by switching the *Power Enable* switch to on, turning the *Manual to Automatic Switch* to "MANUAL". All other steps are described in section 1.1.2.

Figure 3: Manual to automatic switch, Emergency stop, Power enable, See figure 4

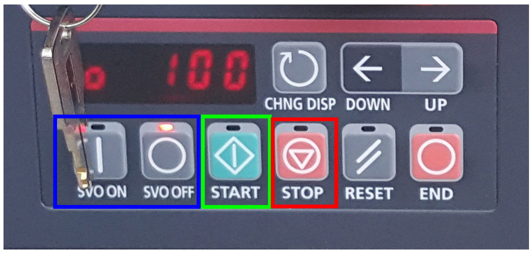


Figure 4: Turn on and off servo motors, Start current program, Stop running program

1.1.2 Teaching Box (TB)

The Teaching Box is used to control everything on the robot directly. Almost all tasks can also be done using a PC connected over a network. In this section only a few basic functions are presented, to get an idea how to work with the Teaching Box. As described in section 1.1.1, one hast to turn on the controller and switch to the manual mode to work with the Teaching Box. Here are the main tasks done with the Teaching Box during the project.

- **Move robot to position:** Press the enable button of the Teaching Box (see figure 7). Now the button lights up. To enable the servo motors, one has to grab the three position switch on the back of the Teaching Box (figure 6) and pull it to the right or to the left until one notices a click sound from the switch. This will allow the servo motors to be enabled. If one pulls the switch too strong, one will notice a second click, which means that the servo motors cannot be enabled anymore. One can check if the switch is in the right position, if one presses the *SERVO* button with the other hand while holding the three position switch in the middle position as described above. If everything is correct, one will notice the sound of the started servo motors and after about a second the *SERVO* LED will light up. During operating the robot with the Teaching Box, one has to hold the three position switch always in the middle position. If one releases the switch or presses it to strong, the servo motors will be disabled.

  By pressing the *JOG* button on the Teaching Box, one can see the current angles of each joint. In the middle of the first line one sees the current coordinate system in which the robot will move.

  - JOINT: Move each joint separately
  - TOOL: Move in the tool coordinate system specified in the *MEXTL* register
  - XYZ: Move in an absolute coordinate system where the origin is in the base of the robot.

  To move the robot one has to press *+/- XYZABC* buttons. With the *OVRD UP/DOWN* buttons one can set the speed of the robot.

  To go back to the start screen, one can use the *FUNCTION* button to change the buttons on the display controlled by the *F<1-4>* buttons. After pressing it once over the *F4* button *CLOSE* appears. By pressing it, one come back to the start screen.

- **Check input and write outputs:** To check the inputs, make sure that the I/O extender is switched on (see figure 2). Then press the *MONITOR* button on the Teaching Box. And select *INPUT* with the *EXE* button. Now one sees the inputs 0-31. Table 1 shows where each sensor is connected. Trigger each sensor and check if the status of the bit on the Teaching Box changes. To check the *ColorSensor* on the gripper one has to press *NUMBER* (*F1*) and entering the number 900. Now one sees the inputs from 900 to 931. By triggering the *ColorSensor* on the gripper one should see a change on bit 900. If no change can be observed, one has to set the register HIOTYPE to 0 to activate the sensor.

- **Write configuration registers:** The configuration registers are used to configure the robot. There are hundreds, most of them described in the *Mitsubishi Industrial Robot CR750-D/CR751-D Controller RV-2F-D Series Standard Specifications Manual* document. To modify a register first press the *F1* button on the start screen. Select *3.PARAM*. Now one can enter the name of the register (e.g *MEXTL*, *NETIP*). By pressing *DATA* one can change the value of the register and save it by pressing the *EXE* button.

- **Stopping program in automatic mode** Beside pressing the big red *Emergency Stop* button on the Teaching Box or on the front panel of the controller, one can simply press the *STOP* button on the Teaching Box to stop a program running in automatic mode. Pressing the *Stop* button has the advantage that one can resume the program by pressing the *Start* button on the front panel again. Whereas pressing the *Emergency Stop* button causes an error, leading to the acoustic warning tone which must be reset by resetting the button, followed by pressing the reset button and finally to resume the program one has to press the *Start* button on the front panel.

Figure 5: Teaching Box (TB) front view

Figure 6: Three-position enable switch

Figure 7: TB (Teaching Box) enable switch

1.1.3 Multigripper

The Muligripper is used to pick up and place the work pieces. There are five different options to pic up a part with the Multigripper. As one can see in figure 8, the gripping points have different dimensions and positions. With gripper one two and three, one can move the work piece body or the whole assembled work piece. In contrast to gripper one and two, gripper three is tilted by 90 degree. Gripper 4 is used to pick up springs and gripper 5 is used to pick up the small and the big pistons. The gripping mechanism is triggered by the outputs with number 900 and 901 called. In MELFA BASIC V it is sufficient to write HOPEN 1 and HCLOSE 1 to open and close the gripper. At the front of the gripper at the left side, there is an optical sensor called *ColorSensor*, which is connected to input 900. One can only read the register if register HIOTYPE is set to 0 (default 1).

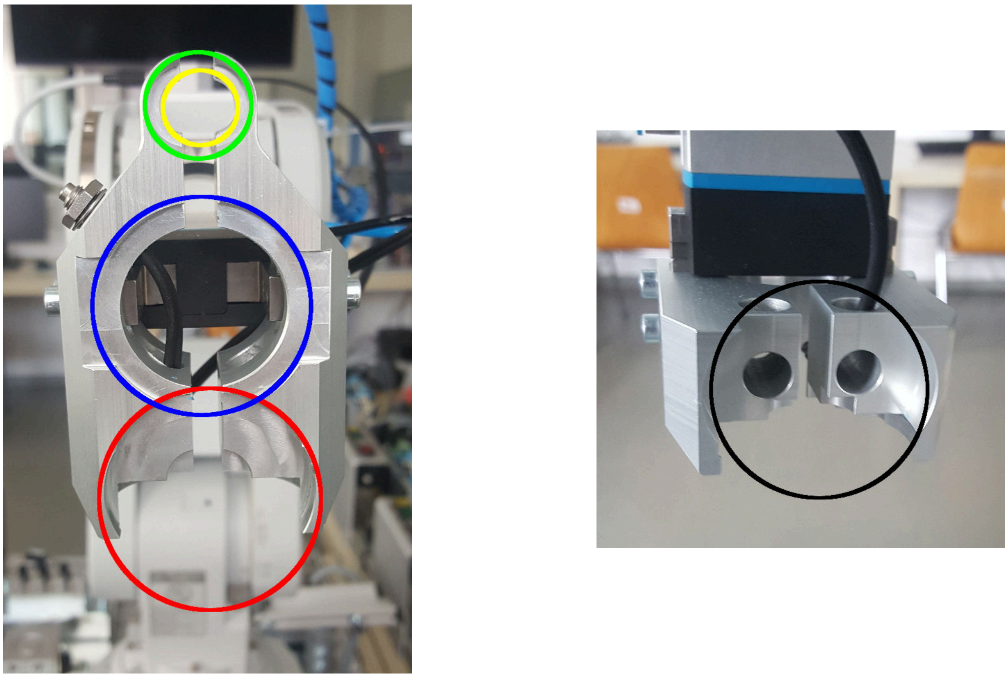


Figure 8: Multigripper: Gripper one, Gripper two, Gripper three, Gripper four, Gripper five

1.1.4 Resetting Errors

If an error occurs, the controller and the Teaching Box send out an annoying beeping sound and write an error code on the display of the controller and on the display of the Teaching Box. To get an idea of what went wrong, one can look up the error code in the *Mitsubishi Industrial Robot CR750/CR751/CR760 Series Controller INSTRUCTION MANUAL Troubleshooting* file. To reset the beeping, one has to press either the reset button on the front panel of the controller and/or the reset button on the Teaching Box. Some errors must be fixed before one can reset the beeping (e.g. a fuse is blown, Teaching Box enabled in automatic mode), and for some errors one can reset the beeping imediatly (e.g. position not defined in a program).

1.2 Robot Handling Module

In figure 9 one can see the robot handling module. Two of the four marked parts are used in this project, the input chute and the assembly socket. Unassembled work pieces can be placed on the input chute and slide to the bottom of it. At the bottom, there is an optical sensor, which signals that a new part is on the chute. This sensor is called *PartAvailable*. Its value can be read out on input number 4.

The second part used in this project from this module is the assembly socket. The first recess which is about 5mm deep can be used to place a work piece and take it with a different gripper. The sensor embedded in the assembly socket, called *Sensor4HoleInBottom* connected to input 3, can be used to detect the three holes at the bottom of each work piece. To sense the holes, one has to place the work piece exactly over the little mark left from the sensor and rotate the work piece. One a hole is found, one can use the deep recess with the small pin in it to place the work piece in it. The little pin prevents the work piece from rotating. This is useful if one wants to screw a cap on the work piece. The last function of the assembly socket is the big pin on the right upper edge. It serves two purposes. First it can be used to place a cap on it, and grab it with a different gripper. The second way to use it, is to place a cap on it, rotate the cap and read out the *Sensor4HoleInBottom* sensor. With this constellation, one can sense the locking knobs of the cap.

1.3 Robot Assembly Module

In figure 10 one can see the robot assembling module. All four parts are used in this project. The first part is the output chute. It is used to place already assembled work pieces on it. The second part is the cab stack, which is a separation unit. It has a storage tower for the caps and a piston which pushes out one cap after the other. To trigger the piston the output called *CapPiston* located at output 12, must be set. The status of the piston can be observed over the two inputs *CapPistonBack* and *CapPistonFront*. Once a cap is pushed out, it is recognized by the *CapAvailable* sensor. The next part of the robot assembly module is the piston storage. It can hold up to eight pistons, four small ones (silver) and four big ones (black). The small ones fit into the black work pieces, the big ones are for the red and silver work pieces. The last part of this module is the spring stack. It separates the springs and places one after the other on a position, reachable by the robot. The piston which pushes out a spring is operated by the output called *SpringPiston* which is connected on output 12. As at the cap stack the status of the piston can be observed with the inputs *SpringPistonFront* and *SpringPistonBack*. If the piston is in the front position, one can check with the *SpringAvailable* sensor, if a spring was outputted.

Figure 9: Robot handling module: Input chute, Calibration tool (mandrel), Sockets for buffers, Assembly socket

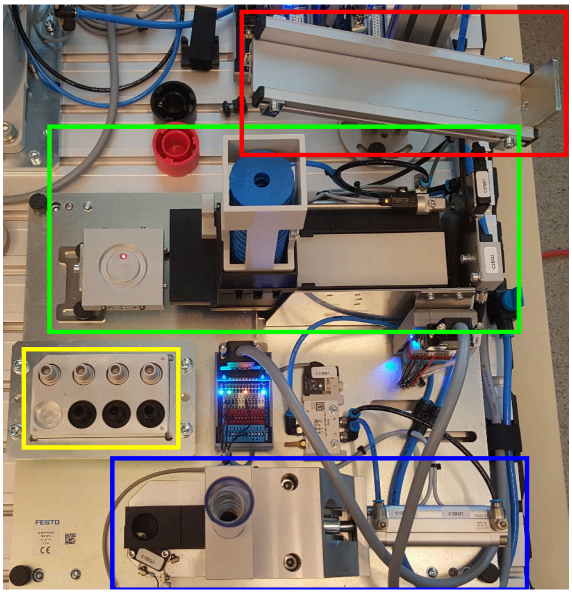


Figure 10: Robot assembly module: Output chute, Cap stack, Piston storage, Spring storage

1.4 Work Pieces

These pieces are supposed to be handled with the robot. As one can see in figure 11 four movable parts are available:

- Work Piece (Body)
- Piston
- Spring
- Cap

There are 3 colors of work pieces available, red, silver and black. They all have the same diameter, however the black one is a few millimeters shorter and has a smaller hole in the middle, allowing only the small pistons to fit in. The step by step instruction in figure 13 shows how the parts fit together to get the fully assembled work piece as shown in figure 12.

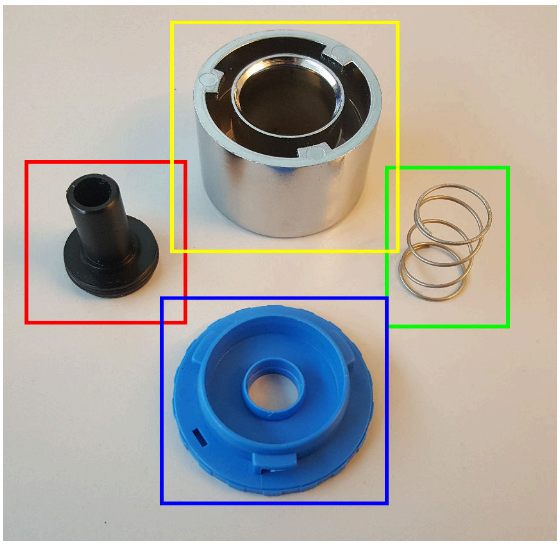


Figure 11: Disassembled work pieces: Piston, Work piece, Spring, Cap

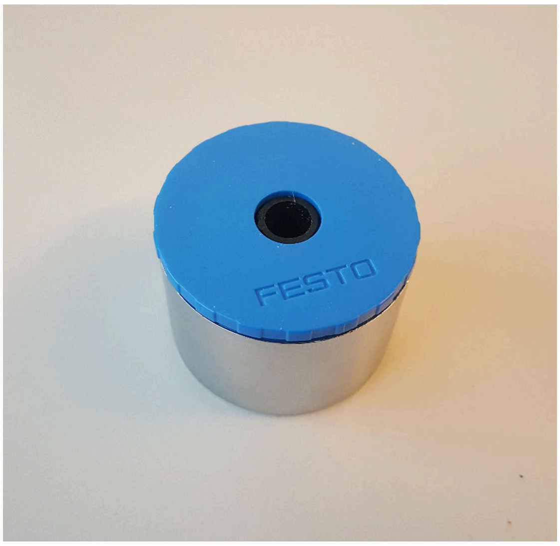


Figure 12: Assembled work piece, spring and piston inside

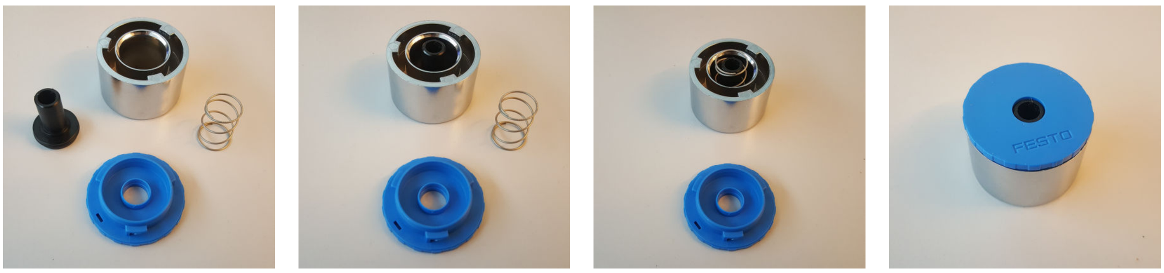


Figure 13: Step by step assembly instruction

2 CIROS Project

In this part of the documentation the setup of a new project is shown. The first part describes how to generate a simple digital twin of the robot station used in the TiLab. The second part describes how to interact and program the real robot. For this project the full version of CIROS Studio 6.0.9 was used.

2.1 Simulation

After starting the CIROS Studio software, a new project is started by "FILE → New → MPS system ...". After saving it to your preferred location, the "Model Libraries" and a 3D-View window are opened. By opening the Library "FESTO MPS" one can choose Robot assembly" and click on "Add". This adds a virtual robot station similar to the one in the TiLab. However we need to remove some parts and change the robot model to get a model close to the one in the TiLab. To do this, the "Model Libraries" can be closed and the "Model Explorer" can be opened. Figure 14 shows the three buttons where one can open these panels.

Figure 14: Left to right: Model explorer, edit mode, model libraries

In the "Model Explorer" one can open the objects lists by clicking on the arrow left of the category. By removing the following objects from the "Objects → RobotAssemblyStation" list, we get closer to our real robot platform. Remove by right clicking on the entry and selecting "Delete":

- Deposit1BoxStack
- Deposit2BoxStack
- IFObject
- IOMonitor
- Mandrel
- Panel
- PlateAssemblyAccessories
- PlateAssemblySenslink
- PlateRobotAccessories
- PlateRobotSenslink
- PopupMessages
- RV-2AJ
- S7_Assembly

For the next step we need again the ”Model Library”. By opening the ”Robots → Mitsubishi → F type” library, one can select and add the ”RV-2FB” model. Additionally add a ”Sensors → DistanceSensor” and a ”Object functions → OR gate”.

The new parts are now in the ”Object” section. By drag and drop one can place it in the ”RobotAssemblyStation”. Place the ”Or” and the ”DistanceSensor” in the ”RobotAssemblyStation → ColorSensorAtGripper”. Now we can close the ”Model Library”. Open the sublist ”RobotAssemblyStation → TransportSystem” and move ”CapStackStopper” and ”CapStackTransport” to ”RobotAssemblyStation → CapStack”, ”ChuteSlide” to ”RobotAssemblyStation → Chute”, delete ”Deposit1Transport” and ”Deposit2Transport”, move ”OutChutetSlide” to ”RobotAssemblyStation → OutChute” and ”SpringStackStopper” and ”SpringStackTransport” to ”RobotAssemblyStation → SpringStack”. Now the list of ”RobotAssemblyStation → TransportSystem” is empty and can be deleted. Open ”RobotAssemblyStation → Chute” and put ”ChuteElbow”, ”ChuteMounting” and ”ChuteSlide_1” into ”ChuteSlide”. Put also ”DistanzeSensor” and ”ReplicatorForNewParts” in ChuteBase.

Now we connect the gripping tool to the new robot. Open the sublist from the ”RV-2FB → Roll” entry, right-click on the gripper point ”RV-2SD” and select ”Properties”. In the tab ”General” one can now choose ”MultiGrip” in the Grip drop-down menu. We add a new ”Grip Point” in ”RobotAssemblyStation → ColorSensorAtGripper → DistanceSensor → Base” and name it ”DistSensorGripPoint”. In ”RobotAssemblyStation → ColorSensorAtGripper → Basis → Gripper points” we add with a right mouse click on it and ”New” a ”Gripper point”. We click with the right mouse on the new ”Gripperpoint” and select ”Properties”. In the ”Properties” Panel we select General and choose at ”Grips” the shortly generated gripping point ”DistSensorGripPoint”. We also add an additional input named ”Input2” at ”RobotStation → ColorSensorAtGripper → Or → Inputs” by clicking with the right mouse button ”New → Digital (system)...”, checking ”Assign name” and entering ”Input2”.

Now let us place the parts on the correct position. One can move the parts to the intended position, by right-clicking on it and selecting ”Properties”. Choose the ”Pose” tab and change the coordinates to the following value:

Set the position of the gripper point from ”RobotAssemblyStation → ColorSensorAtGripper → Basis → Gripper points → Gripperpoint” to x=-4.42, y=1.0, z=24.61, R=0, P=-90, Y=0 in the Section cooridinate system. Set in the ”Dimension” Tab from the ”RobotAssemblyStation → ColorSensorAtGripper → DistanceSensor” x=y=z=0.01.

Now we want to set the parameters for all sensors. Figure 15 shows the configuration for the sensor at ”RobotAssemblyStation → AssemblySocket → Sensor4HoleInBottom”. Figure 16 shows the settings for ”RobotAssemblyStation → ColorSensorAtGripper → DistanceSensor”.

We also have to change ”RobotAssemblyStation → ColorSensorAtGripper → Or”. Select ”Properties and switch to the ”Extended” tab. Click ”Edit...” to change the formula to ”Out_000:=(In_000 OR In_001 OR In_002)”.

In the last step we want to connect the signals from the model, such that we can interact with them. By opening ”MODELING → Manuel operation” we get a new panel for our internal signals. In the ”Model Explorer” we select the ”RV-2FB → Inputs” and deactivate all input signals by right-clicking on them and pressing ”Edit → Deactivate”. Click on the following numbers with the right mouse button, press ”Rename” and rename them as follows:

- Inactive 003 → Sensor4HoleInBottom
- Inactive 004 → PartAvailable

Object Name	X	Y	Z	R	P	Y
AssemblySocket	-489.70	487.0	771.0	-90	0	90
CapStack	-286.00	463.54	811.60	90.0	0	0
Chute	-292.27	168.70	788.08	-126.05	-19.01	0.03
CPValveTerminal	-775.00	714.80	67.00	90.00	0.00	0.00
PistonStorage	-193.29	379.88	834.36	89.04	0.30	-1.42
↳ PistonReplicators	417.82	-43.80	74.73	1.27	-1.24	-0.06
↳ ReplikatorMPiston_1	-219.09	332.56	854.55	-91.27	0.00	1.24
↳ ReplikatorMPiston_2	-218.54	357.53	854.55	-91.27	0.00	1.24
↳ ReplikatorMPiston_3	-217.99	382.50	854.55	-91.27	0.00	1.24
↳ ReplikatorMPiston_4	-217.44	407.47	854.55	-91.27	0.00	1.24
↳ ReplikatorPPiston_1	-189.13	330.22	854.20	-91.24	0.00	1.24
↳ ReplikatorPPiston_2	-188.58	355.19	854.20	-91.24	0.00	1.24
↳ ReplikatorPPiston_3	-188.03	380.16	854.20	-91.24	0.00	1.24
↳ ReplikatorPPiston_4	-187.47	405.13	854.20	-91.24	0.00	1.24
RV-2FB	-510	210	784	0	0	0
SpringStack	-74	379	809	0	0	0

Table 3: Positions of the parts in the simulation in the world coordinate system

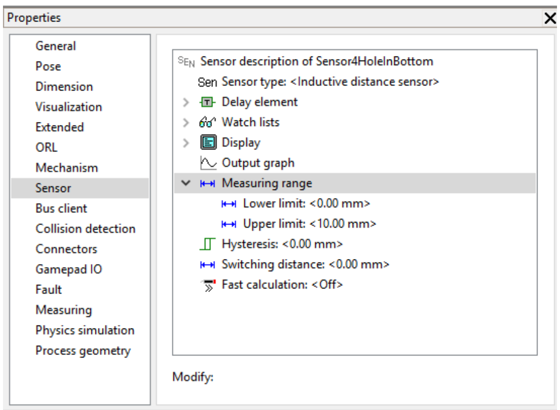


Figure 15: Sensor4HoleInBottom settings

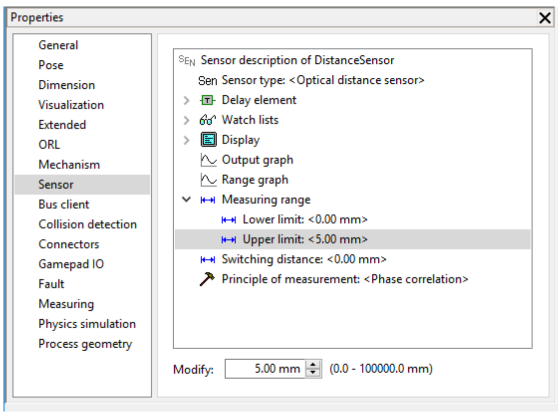


Figure 16: DistanceSensor settings

- Inactive 008 → SpringPistonBack
- Inactive 009 → SpringPistonFront
- Inactive 010 → SpringAvailable
- Inactive 012 → CapPistonBack
- Inactive 013 → CapPistonFront
- Inactive 015 → CapAvailable
- Inactive 256 → ColorSensor

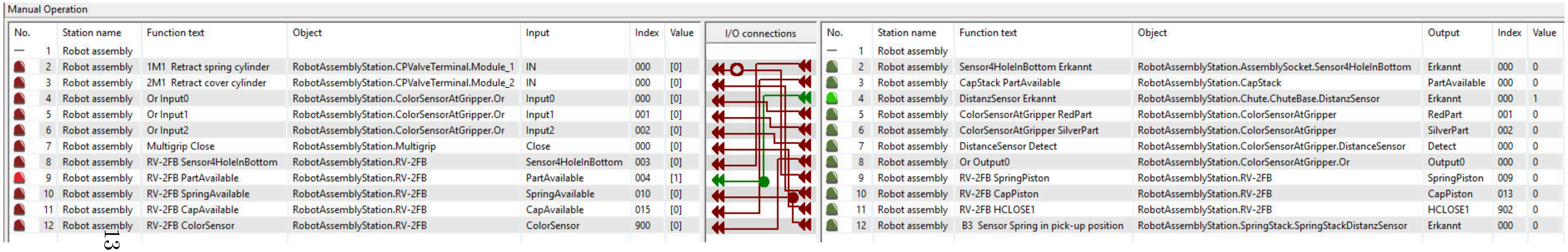
Go to "RV-2FB → Outputs" and rename the following entries:

- Inactive 008 → SpringPiston
- Inactive 012 → CapPiston

Now we connect in the "Manual Operation" Panel the following signals. Figure 17 shows a screenshot of the connected input and output signals. If you cannot see all objects, press the right mouse button and select "Show all".

2.1.1 Loading Program

To start a new programming project click on "FILE → New → File...". Select a "Melfa Basic V (MBA-V)" project, name and save it. The "Project Management" panel opens. Add a new "Program (.mb5)" and a new "Position list (.pos)" file to the project by right-clicking on "Projects → doc (MBA5) → Files" and selecting "New..." in the "Project Management" panel. After storing and naming them, one can open them through the "Project Management" by double clicking on the files. The assembling program from section 3.1 and the positions from section 3.3. Compile the program by pressing "Ctrl+F9", restarting the robot by pressing "Ctrl+Shift+F5". Now one can click on "Switch4NewPart" (see Figure 18 to generate a new part, change the color of the part by clicking on "Switch4MetalPart", "Switch4BlackPart" or "Switch4RedPart", and generate new pistons, springs and caps by pressing the "Switch4NewPistons", "Switch4NewSprings" or "Switch4NewCaps". The robot starts to move, as soon as a work piece is available in the input chute.



Manual Operation

No.		Station name	Function text	Object	Input	Index	Value
—	1	Robot assembly					
	2	Robot assembly	1M1 Retract spring cylinder	RobotAssemblyStation.CPValveTerminal.Module_1	IN	000	[0]
	3	Robot assembly	2M1 Retract cover cylinder	RobotAssemblyStation.CPValveTerminal.Module_2	IN	000	[0]
	4	Robot assembly	Or Input0	RobotAssemblyStation.ColorSensorAtGripper.Or	Input0	000	[0]
	5	Robot assembly	Or Input1	RobotAssemblyStation.ColorSensorAtGripper.Or	Input1	001	[0]
	6	Robot assembly	Or Input2	RobotAssemblyStation.ColorSensorAtGripper.Or	Input2	002	[0]
	7	Robot assembly	Multigrip Close	RobotAssemblyStation.Multigrip	Close	000	[0]
	8	Robot assembly	RV-2FB Sensor4HoleInBottom	RobotAssemblyStation.RV-2FB	Sensor4HoleInBottom	003	[0]
	9	Robot assembly	RV-2FB PartAvailable	RobotAssemblyStation.RV-2FB	PartAvailable	004	[1]
	10	Robot assembly	RV-2FB SpringAvailable	RobotAssemblyStation.RV-2FB	SpringAvailable	010	[0]
	11	Robot assembly	RV-2FB CapAvailable	RobotAssemblyStation.RV-2FB	CapAvailable	015	[0]
	12	Robot assembly	RV-2FB ColorSensor	RobotAssemblyStation.RV-2FB	ColorSensor	900	[0]

I/O connections

No.		Station name	Function text	Object	Output	Index	Value
—	1	Robot assembly					
	2	Robot assembly	Sensor4HoleInBottom Erkannt	RobotAssemblyStation.AssemblySocket.Sensor4HoleInBottom	Erkannt	000	0
	3	Robot assembly	CapStack PartAvailable	RobotAssemblyStation.CapStack	PartAvailable	000	0
	4	Robot assembly	DistanzSensor Erkannt	RobotAssemblyStation.Chute.ChuteBase.DistanzSensor	Erkannt	000	1
	5	Robot assembly	ColorSensorAtGripper RedPart	RobotAssemblyStation.ColorSensorAtGripper	RedPart	001	0
	6	Robot assembly	ColorSensorAtGripper SilverPart	RobotAssemblyStation.ColorSensorAtGripper	SilverPart	002	0
	7	Robot assembly	DistanceSensor Detect	RobotAssemblyStation.ColorSensorAtGripper.DistanceSensor	Detect	000	0
	8	Robot assembly	Or Output0	RobotAssemblyStation.ColorSensorAtGripper.Or	Output0	000	0
	9	Robot assembly	RV-2FB SpringPiston	RobotAssemblyStation.RV-2FB	SpringPiston	009	0
	10	Robot assembly	RV-2FB CapPiston	RobotAssemblyStation.RV-2FB	CapPiston	013	0
	11	Robot assembly	RV-2FB HCLOSE1	RobotAssemblyStation.RV-2FB	HCLOSE1	902	0
	12	Robot assembly	B3 Sensor Spring in pick-up position	RobotAssemblyStation.SpringStack.SpringStackDistanzSensor	Erkannt	000	0

Figure 17: Signal connections

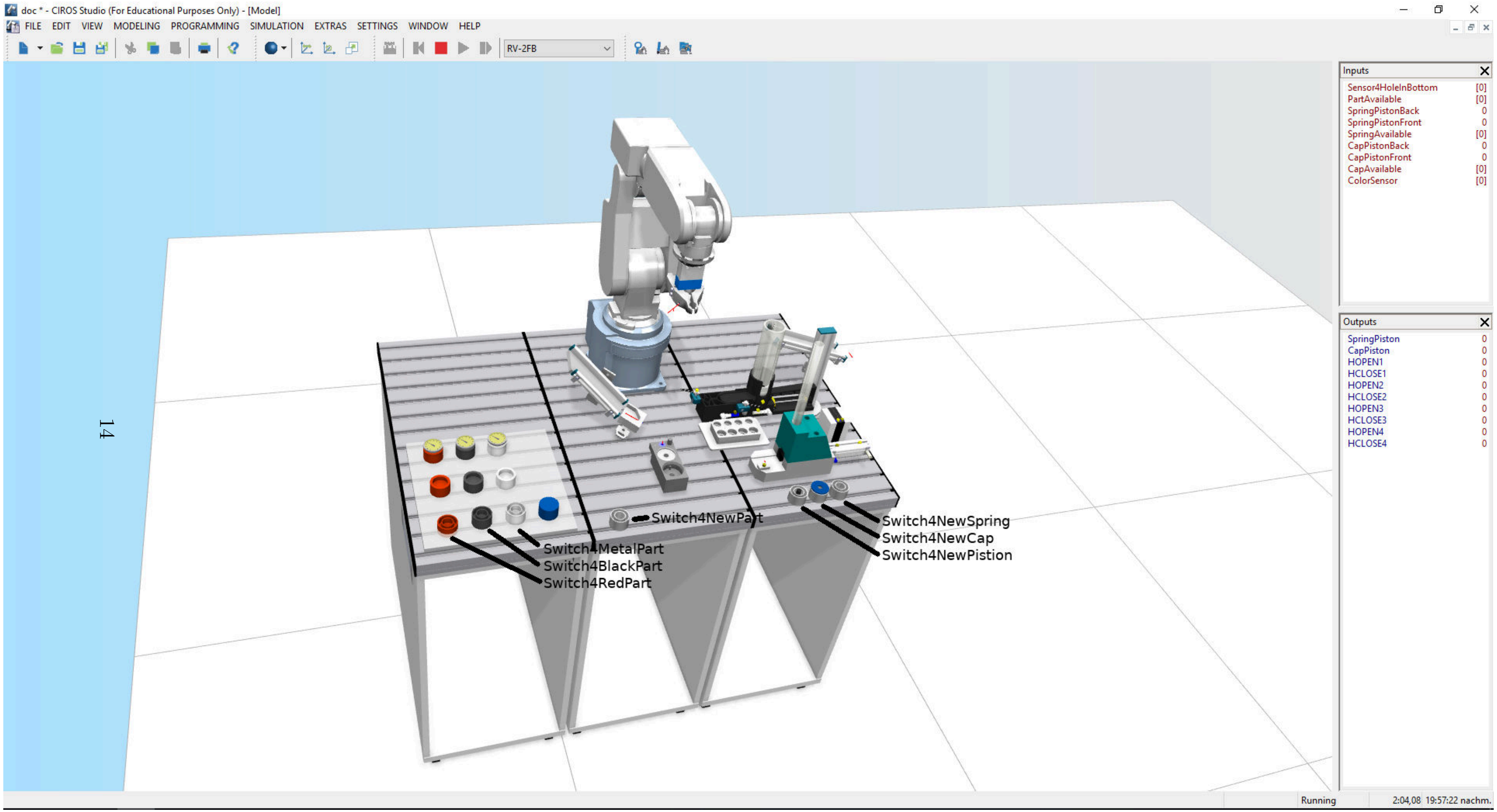


Figure 18: Robot simulation overview

2.2 Interacting With The Real Robot

To interact with the robot over the PC one needs a CIROS Project similar to the one described in section 2.1, however it is sufficient to have the correct robot in the simulation. The positions and the simulation signaling is not necessary. For the next steps it is assumed, that a project is opened in CIROS and a program like the assembling program from section 3.1 and a position file like one described in the section 3.3 is part of the project.

2.2.1 Loading Program

To load a program to the robot, one has to compile it first, by pressing "Ctrl+F5" when the program window is focused. The "Messages" panel opens and "=== Comilation: 1 successful, 0 failed, 0 skipped ===" appears in it. To connect to the robot, one has to know its IP address. It is stored in the NETIP register from the controller. One can use the Teaching Box to read it from the controller (see section 1.1.2). Then go to "SETTINGS → Communication port...", select "Network (TCP/IP) and enter the IP address form the robot. Then go to "EXTRAS → Online management → RCI explorer...". Right click on "RV-2FB → Connection" in the "RCI Explorer" panel and select "Connect". Make sure the robot is turned on and is in automatic mode. A warning window will open, press "OK" after you read the warning. Now the connection is established. Go to "EXTRAS → Online management → Download PC->robot" while the program window is focused. Press OK to download the program. Now open the position file and repeat the procedure to download the position file. Now one can start the program by clicking on "RV-2FB → Programs" in the "RCI Explorer", right click on the program and select "Start" as seen in figure 19. **ATTENTION:** If there is a work piece on the chute, the robot will start to move immediately! Make sure you have one hand always on the "Emergency Stop" of the Teaching Box!

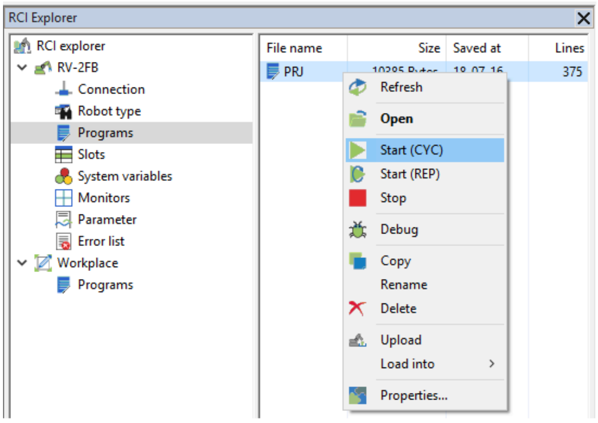


Figure 19: Starting the program

2.2.2 Read And Write Registers

To modify the registers from the controller, power on the controller (see figure 3), switch to "AUTOMATIC" (see also figure 3), connect to the controller by "EXTRAS → Online management → Init connection". After reading the warning, press OK. Open the "RCI explorer" by clicking on "EXTRAS → Online management → RCI explorer...". Double click on "RV-2FB

→ Parameter". After a few seconds all registers are loaded and one can read and modify them by double clicking on them. Notice, writing some registers cause a restart of the controller.

2.2.3 Get/Set Positions

Make sure you are connected to the robot in "AUTOMATIC" mode and you have opened the "RCI explorer" like described in section 2.2.2. Now click in the "RCI explorer" with the right mouse button on "RV-2FB" and select "Get position (robot -> PC)". This will cause the robot in the simulation to move to the position in which the real robot is. Notice, that one has to select the same TCP in the simulation as selected on the robot to get the same position (see section 2.2.4). By pressing "Ctrl+F5" one can reset the robot in the simulation. Before moving the real robot to the same position, make sure again that the same TCP is selected in the simulation and on the robot and that the speed overwrite is set to a safe value.

2.2.4 Setting TCP

The TCP is the coordinate system in which the tool of the robot moves. The standard TCP has its origin at the center of connection between the robot and the gripper.

Setting TCP in the simulation
To change the TCP in the simulation, open the "Model Explorer", and right click on "RobotAssemblyStation → RV-2FB" and select "Properties". Switch to the "TCPs" tab. With "New TCP" one can add all gripper positons as TCPs. For using a TCP click on it and select "Use TCP". Notice, a simple double click does not work.

Setting TCP on the robot
The TCP on the robot can be set by writing the MEXTL register. Do not mistake it with the MEXDTL register! Writing the MEXTL register can be done as described in section 2.2.2, or in section 1.1.2, or by loading a program which includes the line "TOOL (X,Y,Z,Y,P,R)", where the values for XYZYPR come from the TCPs in the simulation. Notice, that in the TCP properties panel the sequence of the parameters is XYZRPY, whereas in the MEXTL register and the TOOL command the corresponding sequence is XYZYPR.

2.2.5 Setting Speed Override

AUTOMATIC Mode:
In "AUTOMATIC" mode one can change the speed settings of the robot by simply pressing the "DOWN" and "UP" buttons on the front panel which are not marked but can be seen in figure 4. Start with value 10 and slowly increase it. The robot will always be to fast in case of an error.

MANUAL Mode:
In "MANUAL" mode one can change the speed settings by using the Teaching Box. Enable the TB and press the "JOG" button. Now you can use the "OVRD ↑/↓" buttons to set the speed overwrite. The display shows the current value in the first line.

3 Demo Programs

The programs listed in section 3.1 and in section ?? were programmed during the project. There are a few things one should mention programing in MELFA BASIC V.

Lessens Learned:

- This language is not case sensitive
- Do not use names (variables, subprograms, etc) longer than eight characters
- Do not use underlines. Using them as second character makes the variable a global variable.
- Gripper positions used in CIROS must not be used in XYZPRY sequence, rather

3.1 Assembling Program

This program takes all parts for a work piece and assembles them together. It is assumed that all parts are always available and that there are no faults, e.g. lost work pieces, wrong parts at wrong places, etc.

```
def pos GBIG
def pos GBIGCAP
def pos GBIGLOW
def pos GCENTER
def pos GCENTERL
def pos GCENTCAP
def pos GSMALL
def pos GSMALLLO
def pos GVert
'
'X Y Z Y P R
GBIG = (39.97,0,127,-180,-0,90)
GBIGCAP = (39.69,-0.01,108.38,-180,-0,90)
GBIGLOW = (39.97,0,124.50,-180,-0,90)
GCENTER = (0,-0,129,-180,-0,90)
GCENTERL = (0,-0,126.5,-180,-0,90)
GCENTCAP = (1.49,-0.13,110.55,-180,-0,90)
GSMALL = (-41.5,0.06,122.83,-180,-0,90)
GSMALLLO = (-41.5,0.07,109.95,-180,-0,90)
GVert = (69,0,89,90,-0,90)

def pos prot
def pos pori
Def Pos pact
Def Io colorSensor=Bit,900
Def Io PARTAV=Bit,4
Def Io foundHole=Bit,3

OVRD 10
While 1
    Tool GSMALL
    Cnt 0
```

```
'wait until a part is available
While PARTAV=0
WEnd
HOpen 1
'check color
Dly 1
Mov P50,40
Dly 1
Mvs P50
Dly 1
Def Inte isColor
isColor% = colorSensor
Dly 1
Mvs P50,40
Dly 1
'move to chute and grab with big gripper
If(isColor% = 1) Then
 Tool GBIG
Else
 Tool GBIGLOW
EndIf
Mov P14,40
Mvs P14
Dly 1
HClose 1
Dly 1
'move to assembly socket and place it there
Mvs P14,40
Mov P10, 40
Mvs P10
Dly 1
HOpen 1
Dly 1
Mvs P10,20
'pick up with middle gripper
If(isColor% = 1) Then
 Tool GCENTER
Else
 Tool GCENTERL
EndIf
Mov P10, 40
Mvs P10
Dly 1
HClose 1
Dly 1
Mvs P10,40
'move to scanning place and find hole in bottom
Mov P16, 40
pact = P16
Mvs pact
Dly 1

prot = (+0.00,+0.00,+0.00,+0.00,+0.00,+1.00)
```

```
pori = (+0.00,+0.00,+0.00,+0.00,+0.00,+0.00)
'find hole
While foundHole=1
 pori=pori+prot
 Mov P16+pori
WEnd
Dly 1
While foundHole=0
 pori=pori+prot
 Mov P16+pori
WEnd
Dly 1
pori=pori-(+0.00,+0.00,+0.00,+0.00,+0.00,+10.00)
Mvs P16+pori
Dly 1
Mov P11+pori+(+0.00,+0.00,+0.00,+0.00,+0.00,+90.00),200
Dly 5
Mvs P11+pori+(+0.00,+0.00,+0.00,+0.00,+0.00,+90.00)
Dly 1
HOpen 1
Dly 1
Mvs P11+pori+(+0.00,+0.00,+0.00,+0.00,+0.00,+90.00),160
Dly 1
'take piston
Tool GSMALL
'check if first piston is available
If (isColor% = 1) Then
 Dly 1
 Mov P40,40
 Dly 1
 Mvs P40
 Dly 1
 If (colorSensor = 1) Then
  Mvs P40,40
  Dly 1
  Mov P30,40
  Mvs P30
  Dly 1
  HClose 1
  Dly 1
  Mvs P30,40
  Dly 1
 Else
  Mov P41,40
  Dly 1
  Mvs P41
  Dly 1
  If (colorSensor = 1) Then
   Mvs P41,40
   Dly 1
   Mov P31,40
   Mvs P31
   Dly 1
```

```
        HClose 1
        Dly 1
        Mvs P31,40
        Dly 1
       Else
        Mov P42,40
        Dly 1
        Mvs P42
        Dly 1
        If (colorSensor = 1) Then
         Mvs P42,40
         Dly 1
         Mov P32,40
         Mvs P32
         Dly 1
         HClose 1
         Dly 1
         Mvs P32,40
         Dly 1
        Else
         Mov P43,40
         Dly 1
         Mvs P43
         Dly 1
         If (colorSensor = 1) Then
          Mvs P43,40
          Dly 1
          Mov P33,40
          Mvs P33
          Dly 1
          HClose 1
          Dly 1
          Mvs P33,40
          Dly 1
         Else
          While(1)
          'Loop until reset
          WEnd
         EndIf
        EndIf
       EndIf
      EndIf
     Else
     'black work piece
      Dly 1
      Mov P44,40
      Dly 1
      Mvs P44
      Dly 1
      If (colorSensor = 1) Then
       Mvs P44,40
       Dly 1
       Mov P34,40
```

```
        Mvs P34
        Dly 1
        HClose 1
        Dly 1
        Mvs P34,40
        Dly 1
       Else
        Mov P45,40
        Dly 1
        Mvs P45
        Dly 1
        If (colorSensor = 1) Then
         Mvs P45,40
         Dly 1
         Mov P35,40
         Mvs P35
         Dly 1
         HClose 1
         Dly 1
         Mvs P35,40
         Dly 1
        Else
         Mov P46,40
         Dly 1
         Mvs P46
         Dly 1
         If (colorSensor = 1) Then
          Mvs P46,40
          Dly 1
          Mov P36,40
          Mvs P36
          Dly 1
          HClose 1
          Dly 1
          Mvs P36,40
          Dly 1
         Else
          Mov P47,40
          Dly 1
          Mvs P47
          Dly 1
          If (colorSensor = 1) Then
           Mvs P47,40
           Dly 1
           Mov P37,40
           Mvs P37
           Dly 1
           HClose 1
           Dly 1
           Mvs P37,40
           Dly 1
          Else
           While(1)
```

```
          'No piston available, wait for reset
          WEnd
        EndIf
      EndIf
    EndIf
  EndIf
EndIf
'put it in the work piece
Mov P18,50
Dly 1
Mvs P18
Dly 1
HOpen 1
Dly 1
Mvs P18,80
'output a spring
Def Io spring=Bit,8
spring = 1
'move to spring and grab it
Tool GSMALLLO
Mov P19,40
Dly 1
Mvs P19
Dly 1
HClose 1
Dly 1
Mvs P19,40
Dly 1
spring = 0
'put spring in work piece
Tool GSMALL
Mov P18,50
Dly 1
Mvs P18
Dly 1
HOpen 1
Dly 1
Mvs P18,80
'output cap
Def Io cap=Bit,12
cap = 1
Dly 1
cap=0
'set tool to big grabber
TOOL GBIGCAP
'move to cab an grab it
Mov P20,40
Dly 1
Mvs P20
Dly 1
HClose 1
Dly 1
Mvs P20,40
```

```
    Dly 1
    'move to big nop
    Mov P21,40
    Dly 1
    Mvs P21
    Dly 1
    HOpen 1
    Dly 1
    Mvs P21,40
    'change to center tool
    Tool (+0.00,+0.00,+110.55,-180.00,+0.00,+90.00)
    'grab it with the center tool
    Mov P21,40
    Dly 1
    Mvs P21
    Dly 1
    HClose 1
    Dly 1
    'find knops
    prot = (+0.00,+0.00,+0.00,+0.00,+0.00,+1.00)
    pori = (+0.00,+0.00,+0.00,+0.00,+0.00,+0.00)
    pact = (+0.00,+0.00,+0.00,+0.00,+0.00,+0.00)
    While(foundHole = 0)
     pori = pori - prot
     Mvs P21+pori
    WEnd
    While(foundHole = 1)
     pori = pori - prot
     Mvs P21+pori
    WEnd
    Dly 1
    'OVRD 5
    'move up and to work piece
    If (isColor% = 0) Then
     'black
     pori = pori + (+0.00,+0.00,-2.50,+0.00,+0.00,+0.00)
    EndIf
    'OVRD 10
    Dly 1
    Mvs P21+pori,40
    'Dly 1
    'OVRD 20
    Dly 1
    Mvs P18+pori,40
    Dly 1
    Mvs P18+pori
    Dly 1
    Mvs P18+pori+(+0.00,+0.00,+0.00,+0.00,+0.00,-75.00)
    Dly 1
    HOpen 1
    Dly 1
    Mvs P18+pori+(+0.00,+0.00,+0.00,+0.00,+0.00,-75.00),40
    Dly 1
```

```
    'set tool to big grabber
    If (isColor% = 1) Then
     Tool (+40.00,+0.00,+108.38,-180.00,+0.00,+90.00)
    Else
     Tool (+40.00,+0.00,+105.88,-180.00,+0.00,+90.00)
    EndIf
    Dly 1
    Mvs P18+(+0.00,+0.00,+0.00,+0.00,+0.00,+90.00),40
    Dly 1
    Mvs P18+(+0.00,+0.00,+0.00,+0.00,+0.00,+90.00)
    Dly 1
    HClose 1
    Dly 1
    Mvs P_Curr,250
    Dly 1
    Mov P15
    Mov P23,50
    Dly 1
    Mvs P23
    Dly 1
    HOpen 1
    Dly 1
    Mov P23,100
    Dly 1
    Mov P15
    Dly 1
    WEnd
End
```

3.2 Loop Program

This program is supposed for a demo run. It takes a fully assembled work piece, partly disassembles it, and reassembles it afterward. After finishing, it feeds the work piece again into the input chute and the program starts again.

```
Def Pos prot
Def Pos pori
Def Pos pact
Def Io colorSensor=Bit,900
Def Io PARTAV=Bit,4
Def Io foundHole=Bit,3
def pos GBIG
def pos GBIGCAP
def pos GBIGLOW
def pos GCENTER
def pos GCENTERL
def pos GCENTCAP
def pos GSMALL
def pos GSMALLLO
def pos GVert
'
'X Y Z Y P R
```

```
GBIG = (39.97,0,127,-180,-0,90)
GBIGCAP = (39.69,-0.01,108.38,-180,-0,90)
GBIGLOW = (39.97,0,124.50,-180,-0,90)
GCENTER = (0,-0,129,-180,-0,90)
GCENTERL = (0,-0,126.5,-180,-0,90)
GCENTCAP = (1.49,-0.13,110.55,-180,-0,90)
GSMALL = (-41.5,0.06,122.83,-180,-0,90)
GSMALLLO = (-41.5,0.07,109.95,-180,-0,90)
GVert = (69,0,89,90,-0,90)
Ovrd 20
While 1
 Tool GBIG
 Mov P15 'HOME
 Dly 1
 While PARTAV = 0 'Wait for assembled work piece
 WEnd
 HOpen 1
 'output spring piston
  Def Io spring=Bit,8
  spring = 1
 'farbe checken
 Dly 1
 Tool GSMALL
 Mov P50,40
 Dly 1
 Mvs P50
 Dly 1
 Def Inte isColor
 isColor% = colorSensor
 Dly 1
 Mvs P50,40
  Dly 1
  'move to chute and wait for work piece
  If(isColor% = 1) Then
   Tool GBIG
  Else
   Tool GBIGLOW
  EndIf
 GoSub *TSLIDE 'TAKE From SLIDE
 Dly 1
 GoSub *PWBENH 'Put on work bench high
 Dly 1
 GoSub *TWBENH 'TAKE from work bench high
 Dly 1
 GoSub *FINDHOLE 'go to sensor and turn until a hole is found
 Dly 1
 GoSub *DECAP 'remove cap from WE
 Dly 1
 GoSub *RSPRING 'remove spring
 Dly 1
 GoSub *ISPRING 'insert spring
 Dly 1
 GoSub *MOCAP 'mount cap from WE
```

```
 Dly 1
 GoSub *PSLIDE 'put on slide
 Dly 1
WEnd
End
'TAKE from SLIDE
*TSLIDE
 Tool GBIG
  Mov P14,40
  Dly 1
  Mvs P14,2 'Cap added
  Dly 1
  HClose 1
  Dly 1
  Mvs P14,40
  Dly 1
Return
'Put on work bench high
*PWBENH
 Tool GBIG
  Mov P10,40
  Dly 1
  Mvs P10,2 'Cap added
  Dly 1
  HOpen 1
  Dly 1
  Mvs P10,40
  Dly 1
Return
'Take from work bench high
*TWBENH
 Tool GCENTER
  Mov P10,40
  Dly 1
  Mvs P10,2 'Cap added
  Dly 1
  HClose 1
  Dly 1
  Mvs P10,40
  Dly 1
Return
*FINDHOLE
 Tool GCENTER
 Mov P16, 40
 pact = P16
 Mvs pact,2 'Cap added
 Dly 1
  prot = (+0.00,+0.00,+0.00,+0.00,+0.00,+1.00)
  pori = (+0.00,+0.00,+0.00,+0.00,+0.00,+0.00)
  'lochposition finden
  While foundHole=1
   pori=pori+prot
   Mov P16+pori,2 'Cap added
```

```
  WEnd
  Dly 1
  While foundHole=0
   pori=pori+prot
   Mov P16+pori,2 'Cap added
  WEnd
  Dly 1
  pori=pori-(+0.00,+0.00,+0.00,+0.00,+0.00,+10.00)
  Mvs P16+pori,2 'Cap added
  Dly 1
  Mov P11+pori+(+0.00,+0.00,+0.00,+0.00,+0.00,+90.00),200
  Dly 5
  Mvs P11+pori+(+0.00,+0.00,+0.00,+0.00,+0.00,+90.00),2
  Dly 1
  HOpen 1
  Dly 1
  Mvs P11+pori+(+0.00,+0.00,+0.00,+0.00,+0.00,+90.00),160
  Dly 1
Return
*DECAP
 Tool GCENTER
 Mov P11,40
 Dly 1
 Mvs P11,2
 Dly 1
 HClose 1
 Dly 1
 Mvs P11+(+0.00,+0.00,+0.00,+0.00,+0.00,+40.00),2
 Dly 1
 Mov P11+(+0.00,+0.00,+0.00,+0.00,+0.00,+40.00),40
 Dly 1
 Mov P15 'HOME
 Dly 1
 Mov P20,40 'CAP Lager
 Dly 1
 Mvs P20,-15 'CAP Lager
 Dly 1
 HOpen 1
 Dly 1
 Mov P20,40 'CAP Lager
 HOpen 1
 Dly 1
 Mov P15 'HOME
 Dly 1
Return
*RSPRING
 Tool GSMALL
 Mov P11+(0,0,0,0,0,-90),40
 Dly 1
 Mvs P11+(0,0,0,0,0,-90),10
 Dly 1
 HClose 1
 Dly 1
```

```
 Mov P11+(0,0,0,0,0,-90),40
 Dly 1
  'move to spring and store it
  Mov P34,40
  Dly 1
  Mvs P34
  Dly 1
  HOpen 1
  Dly 1
  Mvs P34,40
  Dly 1
  Mov P15 'HOME
Return
*ISPRING
 Tool GSMALL
 'move to spring and grab it
  Mov P34,40
  Dly 1
  Mvs P34
  Dly 1
  HClose 1
  Dly 1
  Mvs P34,40
  Dly 1
 Mov P11+(0,0,0,0,0,-90),40
 Dly 1
 Mvs P11+(0,0,0,0,0,-90),10
 Dly 1
 HOpen 1
 Dly 1
 Mov P11+(0,0,0,0,0,-90),40
 Dly 1
Return
*MOCAP
  'set tool to big grabber
  Tool GBIGCAP
  'move to cab an grab it
  Mov P20,40
  Dly 1
  Mvs P20
  Dly 1
  HClose 1
  Dly 1
  Mvs P20,40
  Dly 1
  'move to big knob
  Mov P21,40
  Dly 1
  Mvs P21
  Dly 1
  HOpen 1
  Dly 1
  Mvs P21,40
```

```
'change to center tool
Tool (+0.00,+0.00,+110.55,-180.00,+0.00,+90.00)
'grab it with the center tool
Mov P21,40
Dly 1
Mvs P21
Dly 1
HClose 1
Dly 1
'find knobs
prot = (+0.00,+0.00,+0.00,+0.00,+0.00,+1.00)
pori = (+0.00,+0.00,+0.00,+0.00,+0.00,+0.00)
pact = (+0.00,+0.00,+0.00,+0.00,+0.00,+0.00)
While(foundHole = 0)
 pori = pori - prot
 Mvs P21+pori
WEnd
While(foundHole = 1)
 pori = pori - prot
 Mvs P21+pori
WEnd
Dly 1
'OVRD 5
'move up and to work piece
If (isColor% = 0) Then
 'black
 pori = pori + (+0.00,+0.00,-2.50,+0.00,+0.00,+0.00)
EndIf
Mvs P21+pori,40
Dly 1
Mvs P22+pori,40
Dly 1
Mvs P22+pori
Dly 1
Mvs P22+pori+(+0.00,+0.00,+0.00,+0.00,+0.00,-75.00)
Dly 1
HOpen 1
Dly 1
Mvs P22+pori+(+0.00,+0.00,+0.00,+0.00,+0.00,-75.00),40
Dly 1
'set tool to big grabber
'If (isColor% = 1) Then
 'Tool (+40.00,+0.00,+108.38,-180.00,+0.00,+90.00)
'Else
 Tool (+40.00,+0.00,+105.88,-180.00,+0.00,+90.00)
'EndIf
Dly 1
Mvs P22+(+0.00,+0.00,+0.00,+0.00,+0.00,+90.00),40
Dly 1
Mvs P22+(+0.00,+0.00,+0.00,+0.00,+0.00,+90.00)
Dly 1
HClose 1
Dly 1
```

```
   Mvs P_Curr,100
   Dly 1
   Mov P15 'HOME
   Dly 1
 Return
 *PSLIDE
  Tool GBIG
  Mov P51,40
   Dly 1
   Mvs P51,2 'Cap added
   Dly 1
   HOpen 1
   Dly 1
   Mvs P51,40
   Dly 1
   Mov P15
   Dly 1
 Return
 'END
```

3.3 Positions

This positions are used for both programs:

No	Position	Orientation	Comment
P15	230.0,0.0,378.0	-0,0,90,R,A,N	HOME
P10	351.1,-14.4,57.7	-0,0,90,R,A,N	Assembly Socket High
P14	234.6,-69.1,70.3	21,0,153,R,A,N	Chute
P11	397.7,-14.5,45.1	-0,0,90,R,A,N	Workbench Low
P18	397.7,-14.5,66.2	-0,0,0,R,A,N	Workbench Low cap
P16	320.9,-30.9,62.7	-0,-0,90,R,A,N	Search for Hole in Bot
P21	319.0,1.7,62.1	0,-0,0,R,A,N	BIG KNOB
P19	400.2,193.8,70.0	0,0,-136,R,A,N	Spring Storage
P30	319.5,120.3,70.3	0,0,-90,R,A,N	Big Piston 1
P31	319.5,145.3,70.3	0,0,-90,R,A,N	Big Piston 2
P32	319.5,170.3,70.3	0,0,-90,R,A,N	Big Piston 3
P33	319.5,195.3,70.3	0,0,-90,R,A,N	Big Piston 4
P34	292.5,122.7,70.3	0,0,90,R,A,N	Small Piston 1
P35	292.5,147.3,70.3	0,0,90,R,A,N	Small Piston 2
P36	292.5,172.3,70.3	0,0,90,R,A,N	Small Piston 3
P37	292.5,197.3,70.3	0,0,90,R,A,N	Small Piston 4
P40	327.1,85.1,70.7	0,-0,-58,R,A,N	Scan Big Piston 1
P41	337.6,114.5,69.5	0,0,-41,R,A,N	Scan Big Piston 2
P42	337.1,138.5,69.5	-0,0,-41,R,A,N	Scan Big Piston 3
P43	331.3,161.4,69.4	-0,0,-51,R,A,N	Scan Big Piston 4
P44	256.8,133.7,67.7	-0,0,178,R,A,N	Scan Small Piston 1
P45	256.8,158.7,67.7	-0,0,178,R,A,N	Scan Small Piston 2
P46	256.8,183.7,67.7	-0,0,178,R,A,N	Scan Small Piston 3
P47	257.0,210.3,67.7	-0,0,178,R,A,N	Scan Small Piston 4
P50	238.6,-17.2,60.8	-0,0,90,R,A,N	Color Measurement
P20	201.2,171.5,67.1	0,-0,1,R,A,N	Cap Storage
P23	99.7,314.8,192.2	-19,-1,1,R,A,N	Out Chute

Figure 20: Real positions of the robot station