ROBOTICS

Product manual

FlexLoader SC 6000

Trace back information:
Workspace Robots and Applications version a61
Checked in 2020-03-06
Skribenta version 5.3.033

Product manual

FlexLoader SC 6000

R19D

Document ID: 3HAC051768-001

Revision: D

The information in this manual is subject to change without notice and should not be construed as a commitment by ABB. ABB assumes no responsibility for any errors that may appear in this manual.

Except as may be expressly stated anywhere in this manual, nothing herein shall be construed as any kind of guarantee or warranty by ABB for losses, damages to persons or property, fitness for a specific purpose or the like.

In no event shall ABB be liable for incidental or consequential damages arising from use of this manual and products described herein.

This manual and parts thereof must not be reproduced or copied without ABB's written permission.

Keep for future reference.

Additional copies of this manual may be obtained from ABB.

Original instructions.

Table of contents

This page is intentionally left blank

Overview of this manual

About this manual

FlexLoader SC 6000 is a standard cell used for machine tool tending. This manual describes how to install, operate, maintain, service and troubleshoot the FlexLoader SC 6000.

Usage

This manual should be used during:

- installation and preparation work
- maintenance work
- repair work
- service work
- operation.

Who should read this manual?

This manual is intended for:

- installation personnel
- maintenance personnel
- repair personnel
- service personnel
- operators.

Prerequisites

Operators and maintenance, repair and installation personnel working with an ABB robot must be trained by ABB and have the required knowledge of mechanical and electrical installation, service and maintenance work.

Trademarks

FlexLoader is a trademark of ABB.

References

Reference	Document ID
Product specification - FlexLoader SC 6000	*3HAA009719-001*
Product manual - FlexLoader Vision	*3HAC051771-001*
Product manual - IRB 2600	*3HAC033453-001*
Product manual - IRB 4600	*3HAC033453-001*
Product manual - IRC5 Panel Mounted Controller	*3HAC027707-001*
Application manual - FlexLoader Standard Safety Center	*3HAC051769-001*

Revisions

Revision	Description
A	First edition.

Continues on next page

Continued

Revision	Description
B	General update. Improved RAPID description and reference.
C	Clarifications on PROFINET setup
D	Updated for R19D. Includes new RAPID program setup.

1 Getting started

Basic steps for integrators

Follow the steps below for getting started with a FlexLoader SC 6000 system.

	Action
1	Become familiar with this manual.
2	Transport the FlexLoader SC 6000 to its intended location and integrate it with machine tool(s) and other equipment. See *Installation on page 25*.
3	Pay special attention to safety and functional interface. See *Interface on page 79*.
4	Prepare the vision system. See *Vision system FlexLoader Vision on page 59* and *Product manual - FlexLoader Vision*.
5	Calibrate all camera work objects. See *Product manual - FlexLoader Vision*.
6	Calibrate all robot work objects. See *Calibration of the coordinate systems on page 61*.
7	Use a simple detail and perform a basic teachin. See *Product manual - FlexLoader Vision*.
8	Adjust the robot program to fit the needs of the robot cell. See *RAPID program on page 91*.
9	Perform a first pick with the robot at low speed. See *Operation on page 119*.
10	Make final teachin and refinements to to robot code.

This page is intentionally left blank

2 Safety

2.1 Introduction

This chapter contains safety information for procedures where there is a risk of personal injury or damage to property.

Equipment that is part of a fully- or semi-automatic system must always be treated with care regarding safety.

The users of ABB industrial robots are responsible for ensuring that the applicable safety laws and regulations in the country concerned are observed, and that the safety devices necessary to protect people working with the robot system are designed and installed correctly. Personnel working with the robot must be familiar with the operation and handling of the robot, as described in the applicable documents.

- Only personnel with required training are authorized to use the system.
- All received training must be documented in writing.

Activity	Trained operator	Trained service and maintenance technician	Qualified electrician	System integrator
Commissioning				X
Operation	X	X		
Trouble-shooting		X	X	
Correction of mechanical fault		X		
Correction of electrical fault			X	
Service		X		
Maintenance		X		

x = Authorized to perform the activity.

2.2 Safety during installation, commissioning and decommissioning

Always check that the installation and maintenance instructions are supplied together with the machine. Carefully read the safety information, before unpacking and installing the equipment.

Personnel who are involved in the installation and commissioning must have relevant training for the respective robot and corresponding safety issues.

WARNING

Maintain established good work practices with regard to robot safety.

Ensure that nobody is within the safeguarded area during automatic operation.

CAUTION

It is forbidden to enter the conveyor. Ensure that nobody is on the conveyor, especially when the belt is in motion.

CAUTION

It is forbidden to step on the illumination roof, the outer walls of the FlexLoader SC 6000 and on any FlexLoader SC 6000 option.

2.3 Safety during setup, maintenance, service and repair

Always ensure that FlexLoader SC 6000 cannot be started when working on the cell. The installed protection that belongs to the equipment must always remain installed during automatic operation.

Work on mechanical systems must only be carried out by qualified person or by a trained person under guidance and supervision of a qualified person according to the applicable technical regulations.

Work on electrical systems or operating material must only be carried out by a qualified electrician or by a trained person under guidance and supervision of a qualified electrician according to the applicable electrical technical regulations.

The following must be performed before performing work on FlexLoader SC 6000:

- Emergency stop must be activated (FlexLoader SC 6000 not in operation).
- The voltage supply to the robot cell must be disconnected.
- Action must be taken so that the voltage supply to the robot cell cannot be connected when work is carried out on the conveyor belt. Use the available lockout device.
- The air supply must be disconnected. Use the available lockout device.
- The key for resetting the protective stop must be carried when working within the safeguarded area of FlexLoader SC 6000.

Note that mechanical, pneumatic or electrical energy may be stored in FlexLoader SC 6000. Take the necessary precautions when carrying out work in the robot cell.

All safety and warning signs must be kept clean and legible. If necessary, replace them.

Always check the function of all safety devices after performing work that may have had an impact on the safety systems.

CAUTION

Ear and eye protection shall be used when working with an air cleaning box/deburring unit. The need will depend on the application and shall be risk assessed.

CAUTION

Ear and eye protection shall be used when working with an air nozzle on the gripper system.

CAUTION

Protective shoes shall be used when working with the FlexLoader SC 6000. The need will depend on the application and shall be risk assessed.

Continues on next page

CAUTION

Pneumatic energy can be stored in the gripper system. Be aware of crushing risk.

CAUTION

Details held in the gripper can fall down after an extended period of depressurized state, if the grippers are equipped with pressure retaining valves. Be aware of crushing risk.

2.4 SafeMove

Standard operation

The robot working space in the FlexLoader SC 6000 is restricted by the use of SafeMove. Due to space requirements, the SafeMove outer boundaries of restricted space are close to the FlexLoader SC 6000 enclosure.

In case of robot malfunction or erroneous programming leading to a movement out of the restricted space, the robot will begin to stop its movement. Minor collisions with the outer FlexLoader SC 6000 housing can occur, but will not lead to a personal hazard.

The working space is restricted by a set of FlexLoader SC 6000 internal zones, and an additional set of FlexLoader SC 6000 external SafeMove zones.

DANGER

The internal SafeMove configuration shall not be disabled or modified for any reason.

DANGER

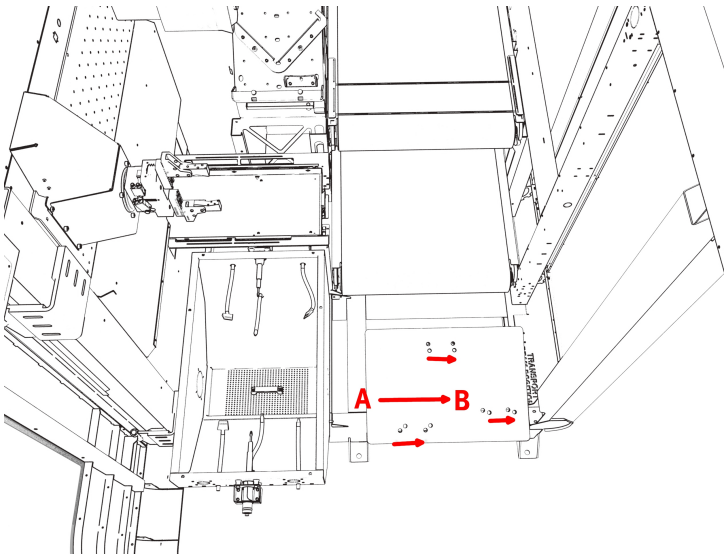
The external SafeMove configuration towards the machine tool must be adopted to the actual application.
Risk assessment must be performed, and the new SafeMove configuration must be validated.

Continues on next page

Alternate mounting position.

For some FlexLoader products, the mechanical configuration does allow an alternate mounting position of the robot. This possibility is given for the integrators convenience and shall not be considered to be a standard option.

If the wall towards the machine tool is (partly) removed, the alternate mounting position of the robot will move the robot ~100 mm closer to the machine tool.



xx1900000185

Pos.	Description
A	Standard mounting position
B	Alternative mounting position

The SafeMove configuration must be changed accordingly to the new work space. The standard ABB guidelines for configuration and validation of SafeMove configurations shall be followed.

DANGER

It is the integrators responsibility to modify and validate the SafeMove configuration for the alternate mounting position in a suitable way.

2.5 Limitation of liability

Any information given in this manual regarding safety must not be construed as a warranty by ABB that the FlexLoader SC 6000 will not cause injury or damage even if all safety instructions are complied with.

The information does not cover how to design, install and operate a complete system, nor does it cover all peripheral equipment that can influence the safety of the entire system.

In particular, liability cannot be accepted if injury or damage has been caused for any of the following reasons:

- Use of FlexLoader SC 6000 in other ways than intended.
- Incorrect operation or maintenance.
- Operation of FlexLoader SC 6000 when the safety devices are defective, not in their intended location or in any other way not working.
- When instructions for operation and maintenance are not followed.
- Non-authorized design modifications made in or around FlexLoader SC 6000.
- Repairs carried out by in-experienced or non-qualified personnel.
- Foreign objects.
- Force majeure.

2.6 Nation and region specific regulations

To protect personnel, the complete system must be designed and installed in accordance with the safety requirements set forth in the standards and regulations of the country where FlexLoader SC 6000 is installed.

Within the EU, refer to the Machinery Directive 2006/42/EC with addendum.

In the US, refer e.g. to ANSI/RIA R15.06 (Industrial Robots and Robot Systems - Safety Requirements) and UL 1740 (Standard for Robots and Robotic Equipment).

In Canada, refer e.g. to CAN/CSA-Z434-03 (Industrial Robots and Robot Systems - General Safety Requirements).

Note

The supplied equipment must not be operated until the processing machine or system in which the equipment is included, has been approved according to national and regional laws and regulations.

2.7 To be observed by the supplier of the complete system

Overview

The integrator is responsible for making sure that the safety devices necessary to protect people working with FlexLoader SC 6000 are designed and installed correctly.

When integrating FlexLoader SC 6000 with external devices and machines:

- The supplier of the complete system must ensure that all circuits used in the protective stop function are interlocked in a safe manner, in accordance with the applicable standards for the protective stop function.
- The supplier of the complete system must ensure that all circuits used in the emergency stop function are interlocked in a safe manner, in accordance with the applicable standards for the emergency stop function.
- The supplier of the complete system must ensure that all circuits used in all other safety function are interlocked in accordance with the applicable standards for that function.

The integrator of the final application is required to perform an assessment of the hazards and risks (HRA).

Note

The integrator is responsible for the safety of the final application.

Safe access

The robot system shall be designed to allow safe access to all areas where intervention is necessary during operation, adjustment, and maintenance.

Where it is necessary to perform tasks within the safeguarded space there shall be safe and adequate access to the task locations.

Safety zones, which must be crossed before admittance, must be set up in front of the robot's working space. Light beams or sensitive mats are suitable devices.

Turn-tables or the like should be used to keep the operator out of the robot's working space.

A safety fence is recommended to ensure safeguarded space. Sufficient space must be provided around the manipulator to protect those working with or on it from hazards such as crushing.

The fence or enclosure must be dimensioned to withstand the force created if the load being handled by the robot is dropped or released at maximum speed. Determine the maximum speed from the maximum velocities of the robot axes and from the position at which the robot is working in the work cell (see the *Robot product specification*).

Also consider the maximum possible impact caused by a breaking or malfunctioning rotating tool or other device fitted to the robot or positioned stationary in the cell.

The FlexLoader SC 6000 used with Eden door switches does not evaluate door locking in its safety function. This has to be considered in the risk assessment.

Continues on next page

2.7 To be observed by the supplier of the complete system
Continued

Safe handling

Make sure that the users cannot be exposed to hazards, including slipping, tripping, and falling hazards.

It must be possible to safely turn off tools, such as milling cutters, etc. Make sure that the guards remain closed until the cutters stop rotating.

It should be possible to release parts by manual operation.

Safe design

The emergency stop buttons must be positioned in easily accessible places so that the system can be stopped quickly. If any of the buttons do not stop all the robot workcell motion, each emergency stop button must be marked (if more than one is provided) to indicate its designated safety function.

Changes to Pluto program

Applicable for systems with the optional safety center.

WARNING

The Pluto program is a critical part of the automation cell's safety system.

Apart from configuration changes, no other changes may be made without qualified review and validation regarding safety risks, safety functions and the overall behavior of the safety system.

Changes to SafeMove configuration

WARNING

The SafeMove configuration is a critical part of the automation cell's safety system.

No configuration changes may be made without qualified review and validation regarding safety risks, safety functions and the overall behavior of the SafeMove system.

Note

The SafeMove configuration must take the robot stopping distances into account, see Robot stopping distances (3HAC048645).

2.8 Safety signals in the manual

Introduction to safety signals

This section specifies all safety signals used in the user manuals. Each signal consists of:

- A caption specifying the hazard level (DANGER, WARNING, or CAUTION) and the type of hazard.
- Instruction about how to reduce the hazard to an acceptable level.
- A brief description of remaining hazards, if not adequately reduced.

Hazard levels

The table below defines the captions specifying the hazard levels used throughout this manual.

For more information, see standard ISO 13849.

Symbol	Designation	Significance
	DANGER	Signal word used to indicate an imminently hazardous situation which, if not avoided, will result in serious injury.
	WARNING	Signal word used to indicate a potentially hazardous situation which, if not avoided, could result in serious injury.
	ELECTRICAL SHOCK	Signal word used to indicate a potentially hazardous situation related to electrical hazards which, if not avoided, could result in serious injury.
	CAUTION	Signal word used to indicate a potentially hazardous situation which, if not avoided, could result in slight injury.
	ELECTROSTATIC DISCHARGE (ESD)	Signal word used to indicate a potentially hazardous situation which, if not avoided, could result in severe damage to the product.
	NOTE	Signal word used to indicate important facts and conditions.

Continues on next page

2.8 Safety signals in the manual
Continued

Symbol	Designation	Significance
	TIP	Signal word used to indicate where to find additional information or how to do an operation in an easier way.

3 Installation

3.1 General

Note

Read the safety precautions and other instructions carefully before unpacking and installing the equipment.

This section describes the installation of FlexLoader SC 6000. If the installation contains further parts, safety instructions may also be found in other documentation.

Unpack the equipment and look for any signs of damage.

Prior to transportation, untreated surfaces may have been rust-proofed with a thin layer of oil which was applied before packing. Wipe off any excess oil before installation using a lint-free cloth.

3.2 Mechanical installation

When the FlexLoader SC 6000 arrives, it is bolted onto a wooden pallet. For container deliveries, the vision tower dismantled from the chassis. This section describes how to unpack the FlexLoader SC 6000 step-by-step with text and pictures.

Always check the FlexLoader SC 6000 for visible or suspected damages prior to unpacking.

Floor requirements

The floor where the FlexLoader SC 6000 is installed has to fulfill certain requirements:

- Standard concrete industry floor is typically sufficient to install the FlexLoader SC 6000.
- The surface of the floor must be flat to within the equivalent of 1 mm in 600 mm (1 inch in 50 ft).
- The floor and fastening elements must withstand the forces originating from the robot, according to the robot product manual. Use bolts with minimum dimensions M20 (3/4”).

Unloading

DANGER

All lifting devices must be properly dimensioned.
FlexLoader SC 6000 weight is ~2300 kg

CAUTION

When transporting using a forklift truck, moving parts on or in the FlexLoader SC 6000 must not be touched or damaged. The equipment must be lowered slowly. Knocks, shaking and drops can cause deformations

WARNING

Never walk or stand beneath a suspended load.

The FlexLoader SC 6000 weight is ~2300 kg and shall be lifted off the transporter with a forklift from the side. On the wooden pallet there are signs where to lift. The

Continues on next page

FlexLoader SC 6000 can be transported on the workshop floor with a forklift from the side, with 3 handheld lifts as the pictures show, or on appropriate skates.

xx1900000186

xx1900000187

Initial transport of the unit using handheld lifts (1). Look for fork lift signs

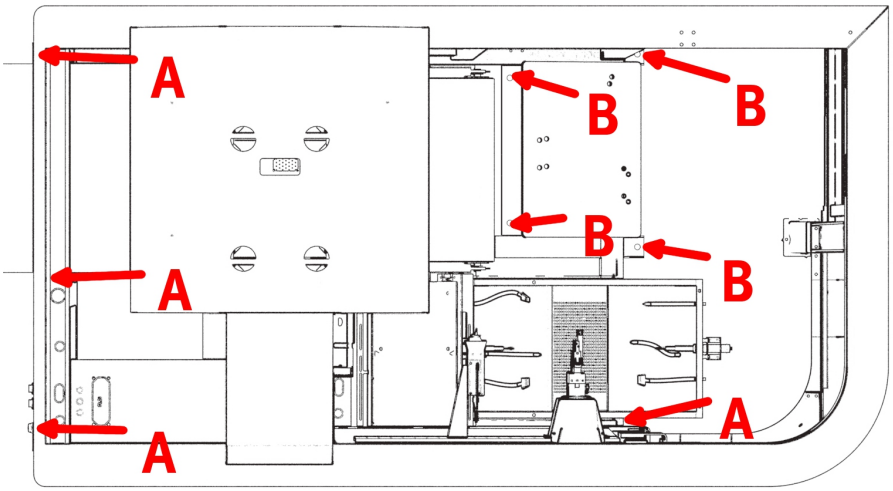
xx1900000188

Initial transport of the unit using handheld lifts (2)

Continues on next page

Unpacking and positioning

Start unpacking by removing all the stretch film. Remove all the transport safety and all the bolts from the pallet.



xx1900000195

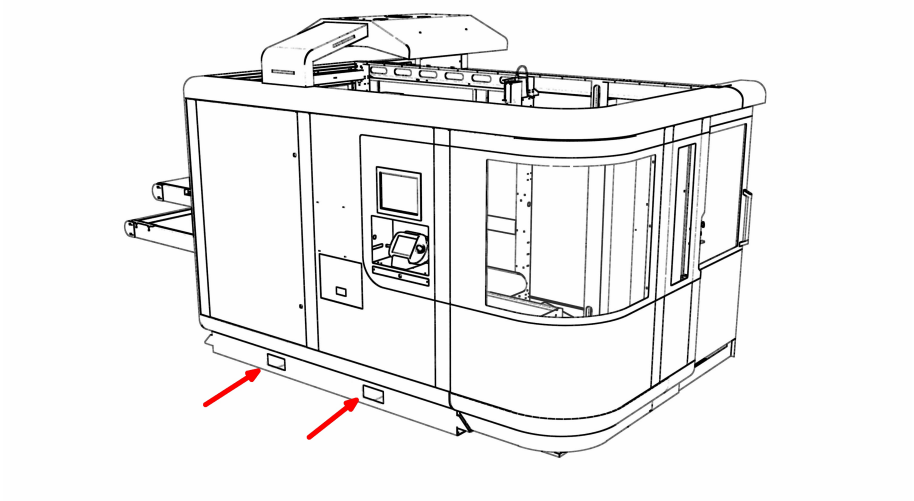
FlexLoader SC 6000 bolting positions on transport pallet

By default, the FlexLoader SC 6000 is secured by bolts at positions (B). In certain cases, additional bolts can be used at positions (A).

Note

Check all delivered parts for damage. If necessary, take appropriate actions.

The FlexLoader SC 6000 can be lifted using a fork lift with the provided fork lift slots, placed on the bottom of the machine.

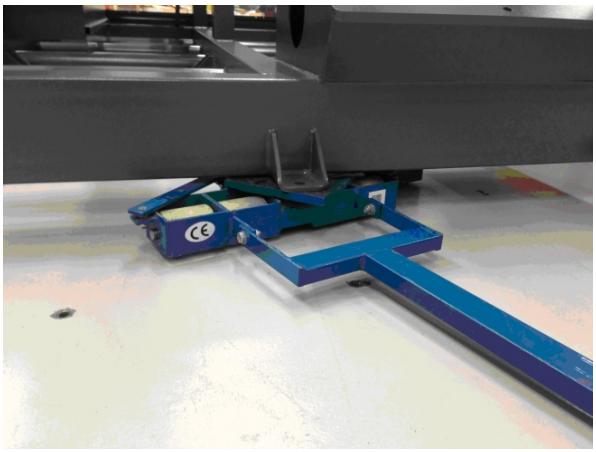


xx1900000189

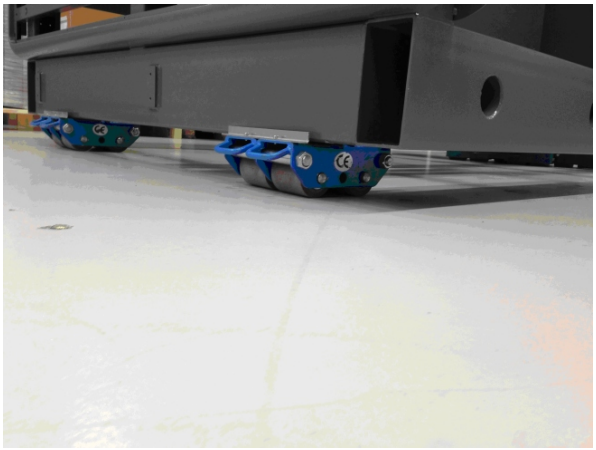
Lift of the FlexLoader SC 6000 using fork lift

Continues on next page

The FlexLoader SC 6000 can also be positioned by using skates.

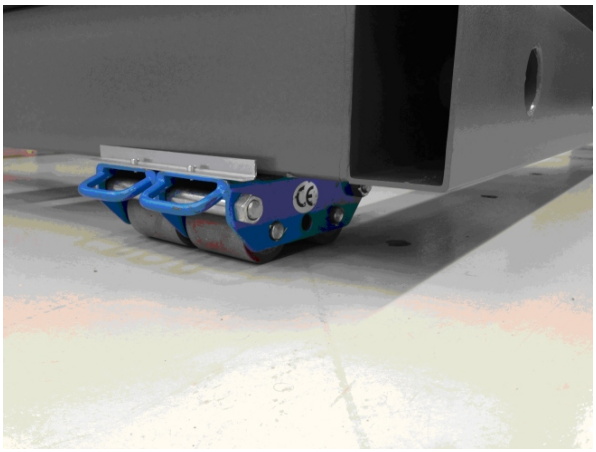


xx1900000190



xx1900000191

Transport of the FlexLoader SC 6000 using skates (1)



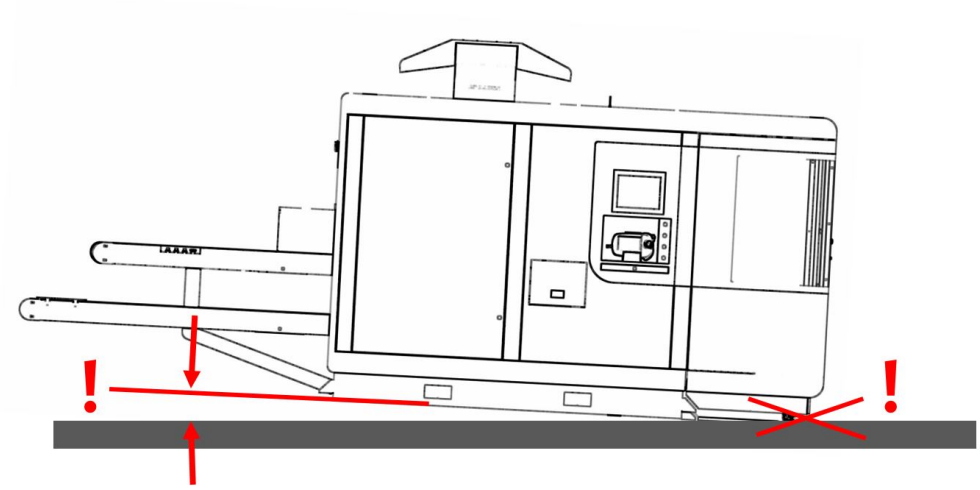
xx1900000192

Transport of the FlexLoader SC 6000 using skates (2)

Continues on next page

CAUTION

Do not lift the FlexLoader SC 6000 to high in the back only. It will damage the plates on the front door.



xx1900000193

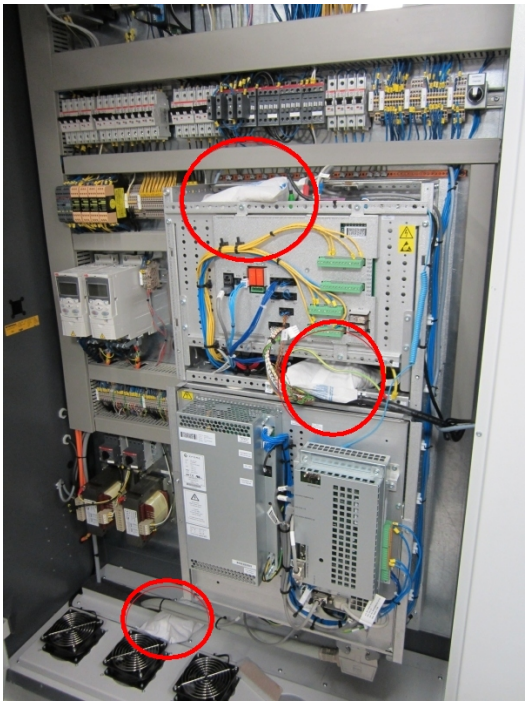
Pay attention on inclination in order to avoid damages to front plates.

Place the FlexLoader SC 6000 at a suitable position relative to the machine tool. Take especially the corridor between the FlexLoader SC 6000 and the machine tool into account.

If present, remove all the absorb bags from the electrical cabinet and the rest of the machine.

Continues on next page

Prior to transportation untreated surfaces may have been rust proofed with a layer of oil or grease which was applied before packing. Wipe off any excess oil/grease before installation using a lint free cloth.



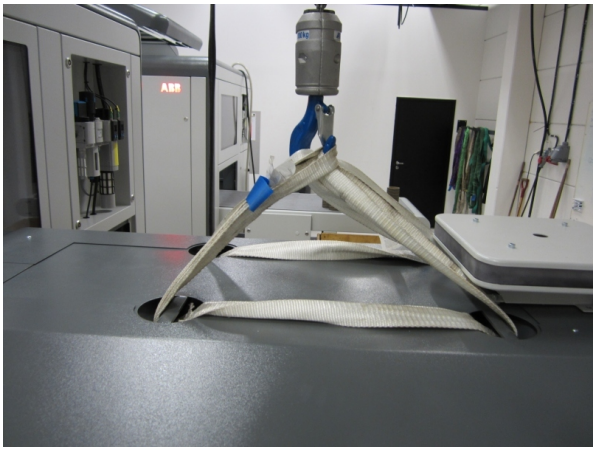
xx1900000194

Example of absorption bags - in the electrical cabinet - that must be removed.

Assemble the vision tower

If the FlexLoader SC 6000 was shipped in a container, the vision tower has to be assembled.

Use straps to lift the vision tower. The straps shall be fastened around the beams inside the tower. See image below.



xx1900000196

Lifting of the vision tower

Continues on next page

The tower shall be placed on top of the chassis at its designated position (se picture below).

xx1900000197

Mounting position vision tower

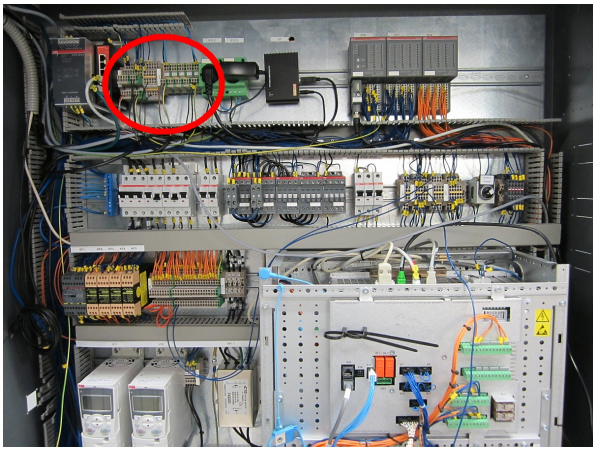
The cables that need to be connected are put inside the tower (see picture below). The cables that they shall be connected to can be found in the vision tower.

The rest of the cables need to be connected to the terminal inside the electrical cabinet according to the circuit diagram.

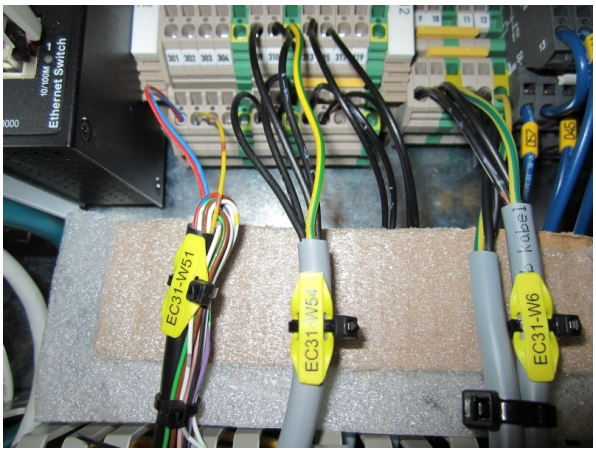
xx1900000198

Cable path from vision tower to FlexLoader SC 6000 chassis.

Continues on next page



xx1900000199



xx1900000355

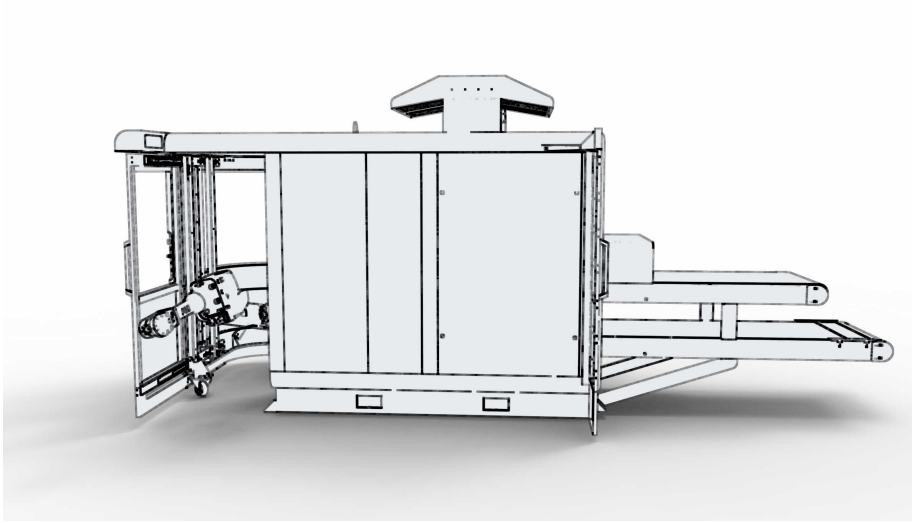
Internal connection of vision tower cables in control cabinet.

Continues on next page

Assemble corridor parts

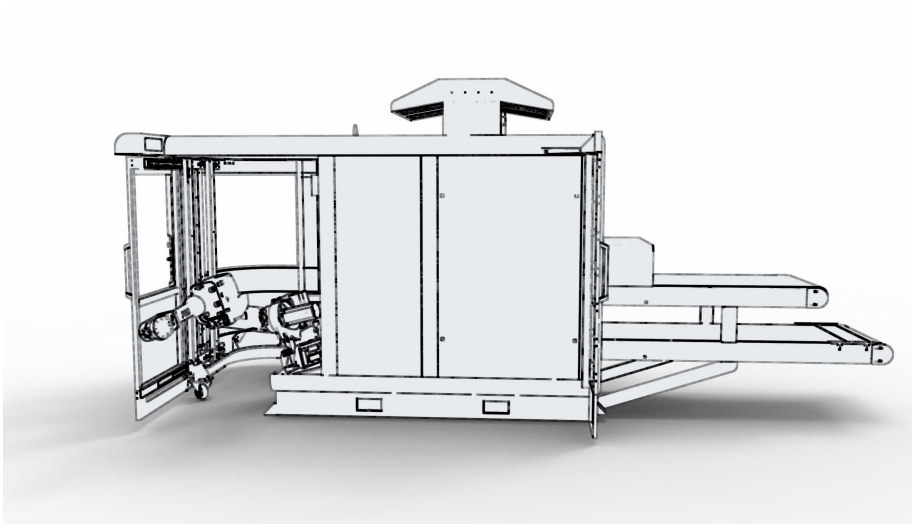
Adjust width of opening towards machine tool

The width of the opening towards the machine tool can be adjusted according to the actual needs. By selecting a suitable amount of the flexible panels on the back side of the FlexLoader SC 6000, various opening widths can be achieved (850 mm, 1300 mm, 1500 mm):



xx1900000356

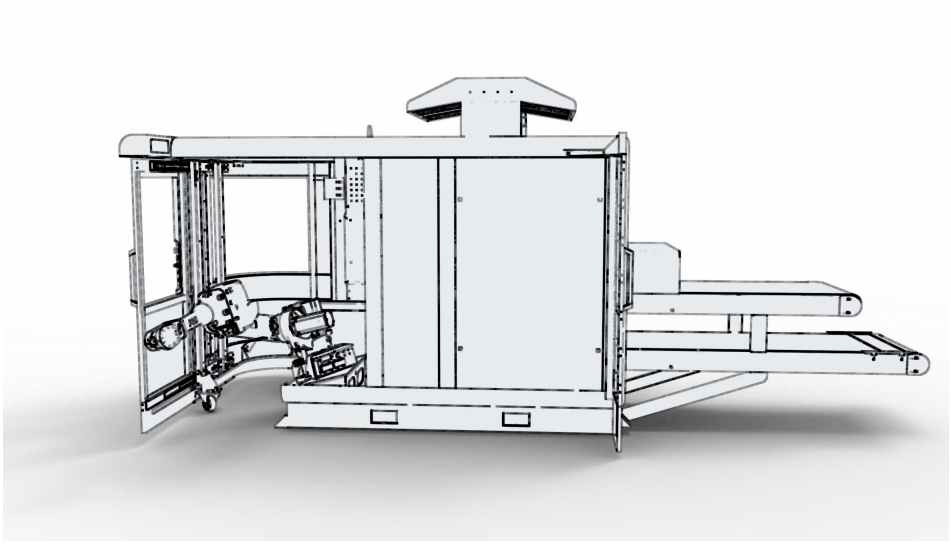
Variable opening towards machine tool: 850 mm



xx1900000357

Variable opening towards machine tool: 1300 mm

Continues on next page

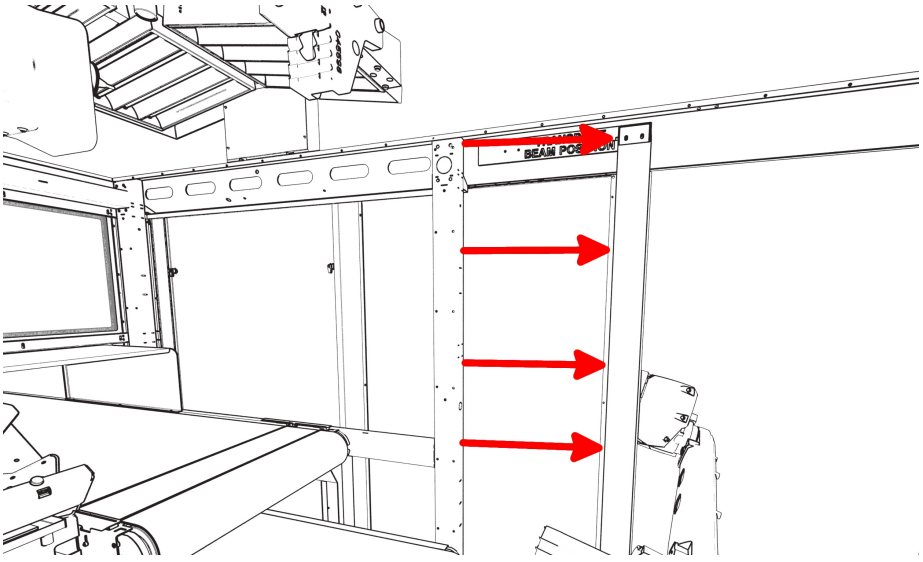


xx1900000358

Variable opening towards machine tool: 1500 mm

Tip

The stabilization support (se image below) is needed for mechanical stability during transport. Save the stabilization support when disassembling it !



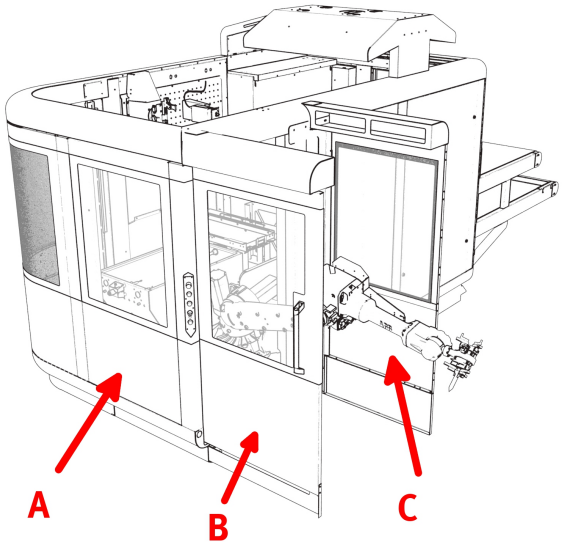
xx1900000359

Transportation stability ensured by support.

Continues on next page

Mount extension panels

The passage towards the machine tool must be protected to prevent operators from entering the robot working space. A ready made corridor part can be mounted on the side of the FlexLoader SC 6000, according to the image below. The extension panels consist of two parts, the extension panel door, and the extension panel back.

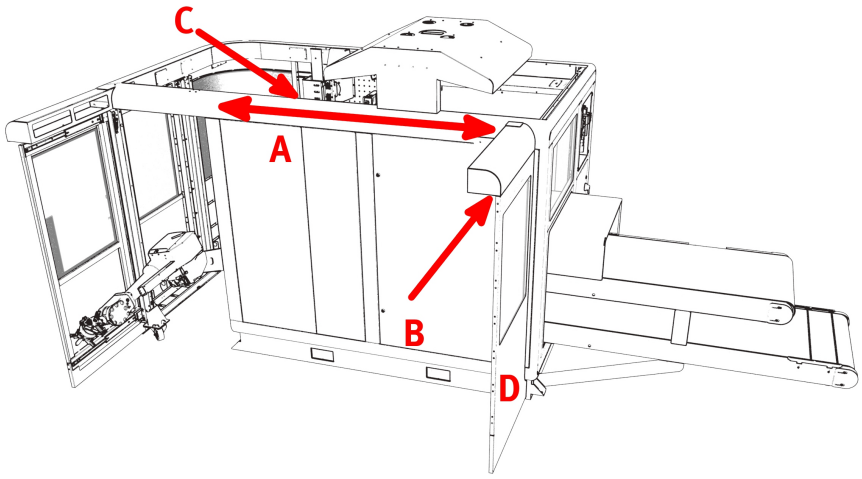


xx1900000360

Extension panel components

Pos.	Description
A	Extension panel door
B	Sliding door
C	Extension panel back

Continues on next page



xx1900000361

Variable positioning of corridor part.

Pos.	Component	Pos.	Component
A	Suitable position range	C	Safety switch connection point
B	Safety switch	D	Extension panel back

If ordered with the extension panel door, the FlexLoader SC 6000 is delivered with the extension panel mounted in place.

The extension panel back has to be mounted on the FlexLoader SC 6000: Select a suitable position along the rim and secure the extension panel back by mounting it to the FlexLoader SC 6000 outer shell (screws on upper and lower rim of the FlexLoader SC 6000).

If the corridor part is a door that can be opened, the safety switch has to be connected to the safety connection point hidden in the FlexLoader SC 6000 upper rim.

Install the FlexLoader SC 6000 to the machine

Positioning and bolting

Note

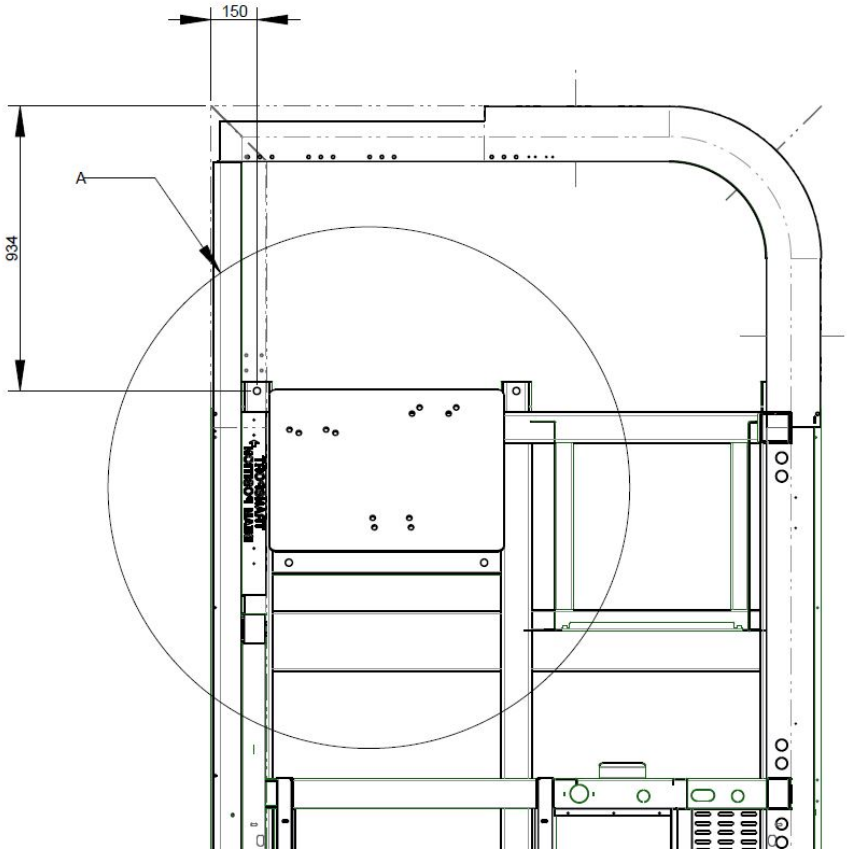
The fastening elements are not part of the delivery as they must be selected with reference to the floor material. Use only suitable fastening elements. Pay attention to the expected loads.

CAUTION

The conveyor must be levelled both in the direction of movement and across it within < 1 mm/m (< 0.5 mm/m for rolling parts). Each deviation must be corrected.

Continues on next page

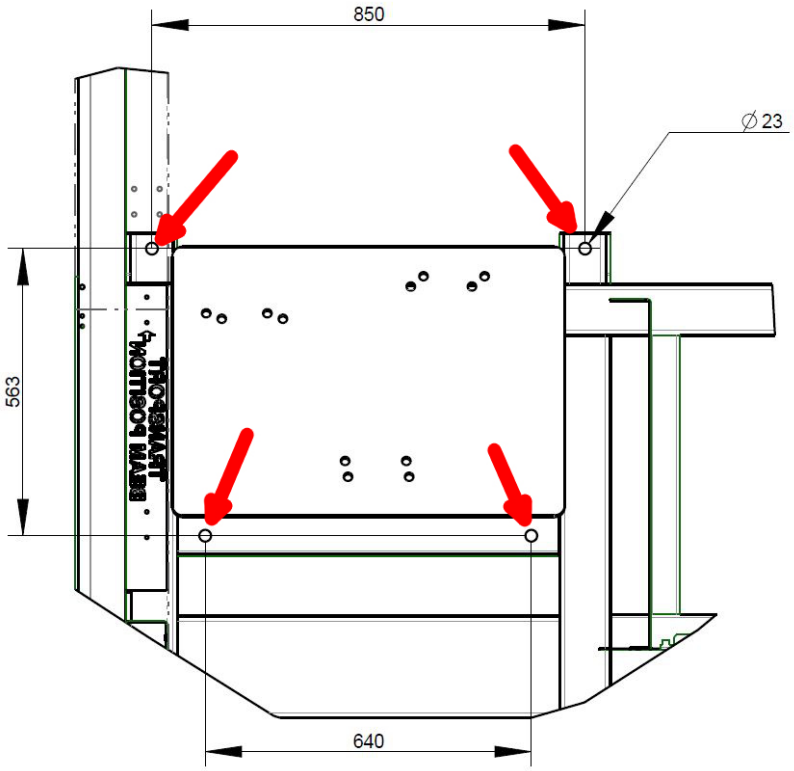
If the wholes are drilled prior to putting the FlexLoader SC 6000 into position use the following drilling plan. Pay attention to distances implied by extension panel options.:



xx1900000362

FlexLoader SC 6000 drilling template (in mm), seen from above (1)

Continues on next page



xx1900000363

Detail A from previous drawing, seen from above (2)

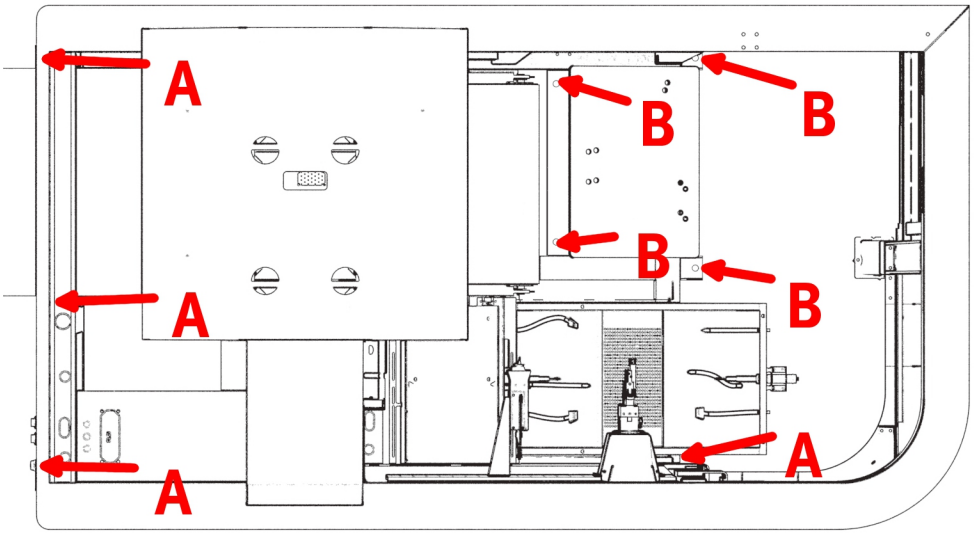
Put the FlexLoader SC 6000 at the intended position.

Level the FlexLoader SC 6000 with regard to the inconveyor, by using the adjustable machine feet (see picture below). The conveyor must be levelled both in the direction of movement and across it, within < 1 mm/m (< 0.5mm/m for rolling parts). Each deviation must be corrected.

Drill holes and bolt the machine onto the floor. There are four brackets with holes where the machine shall be bolted, around the robot.

Continues on next page

The FlexLoader SC 6000 shall be bolted with minimum bolt diameter M20 / 3/4”. Use metal plates below the brackets to avoid air gaps between floor and FlexLoader SC 6000 feet, if necessary.



xx1900000195

FlexLoader SC 6000 from above with mounting positions.

Pos.	Description
A	Machine foot positions
B	Bolting positions

Note

Note that a tightening torque of ~300 Nm / 220 lb ft is required to ensure that the equipment is well secured.

Adjust swing door leveling

The gap between the swing door upper edge and the upper rim of the FlexLoader SC 6000 shall be checked after tightening the FlexLoader SC 6000 to the floor. If swing door and FlexLoader SC 6000 upper rim do not align, the height of the floor wheel shall be adjusted.

The securing hook tension shall be checked and if necessary adjusted tightening the FlexLoader SC 6000 to the floor.

Make sure that the swing door safety switch is working properly after adjustment.

Safety precautions for long parts in turn station

DANGER

A risk assessment has to be performed when using the turn station. It has to consider if the rotated parts constitute a risk to the operator.

Continues on next page

If risk assessment indicates a risk, e.g. due to elongated or bulky parts, an extra vertical plate of suitable size needs to be mounted close to the turn station on the FlexLoader SC 6000 upper rim to avoid that the risk area can be reached from the outside.

Belt tension

The belts are generally pre-installed.

Note

Normally, the belt tension does not need to be adjusted for new machines.

The figures below refer to the tension of a new belt.

Tip

Some tips on how to tension the belts:

- Measure a distance in the middle of the untensioned belt.
- Using both screws, adjust the belt tension to the dimensions given in the table.
- Clean off the marks.

Do not tension to much, as this will cause extra stress on bearings, belts and shafts.

Short rigid belt (type PVC) (<= 1000 mm)

Normal pre-tension for a rigid belt (type PVC) is 0.3% regardless of location of the roller drives. For long rigid belts, the belt must be tensioned slightly more, according to the table.

This pre-tension is also valid for belts with steering profile.

Distance between the markings when the belt is not tensioned [mm]	250	400	700	1000
Distance between the markings when the belt is tensioned [mm]	251	401	702	1003

Long rigid belt (type PVC) (> 1000 mm)

Distance between the markings when the belt is not tensioned: 1000 mm.

Track length [mm]	1000–2500	3000–3500	4000–4500	5000–5500	6000–7500
Distance between the markings when the belt is tensioned [mm]	1003	1003	1004	1004	1005

Continues on next page

Felt belt

Normal pre-tension for a felt belt is 0.5% if the roller drives are at the front edge and 0.7% if the roller drives are at the rear edge.

Distance between the markings when the belt is not tensioned [mm]	250	400	700	1000
Roller drive at front edge: distance between the markings when the belt is tensioned [mm]	251	402	704	1005
Roller drive at rear edge: distance between the markings when the belt is tensioned [mm]	252	403	705	1007

Felt belt with steering profile

Normal pre-tension for a felt belt with steering profile is 0.5% regardless of the location of the roller drive and the length of the belt.

Distance between the markings when the belt is not tensioned [mm]	250	400	700	1000
Roller drive at front edge: distance between the markings when the belt is tensioned [mm]	250.5	401	702	1003

Belt adjustment

The belt is adjusted when FlexLoader SC 6000 is tested at manufacturing, but it may need to be readjusted after transportation or after repair.

- Check that the support rollers are installed at exactly 90° in relation to the frame extension.
- Tighten the adjustment screw that is located on the side that the belt is sliding to (maximum 1/4 turn at a time) until the belt no longer moves sideways. Wait until the belt has run around 10 times before the next adjustment.

Note

After each adjustment, the belt must be run for approximately 30 minutes while being monitored, as it takes some time for the belt to react to the new settings.

3.3 Electrical installation

Introduction

Connect the necessary wiring to the electrical cabinet and connect other media to the respective connection points, according to the wiring diagram.

Connect the wiring between the processing machine and the external equipment according to the wiring diagram.

Ensure that local rules and legislation is followed, e.g. by setting a lockable main switch if needed.

Power supply

Power supply conductors shall be connected directly on the supply disconnecting device. No cable lugs or compression sleeves are required for Cu cables. The supply disconnecting device is located inside the control panel.

The disconnecting device is only suitable for Cu cable. If Al-cable is used, an Al/Cu adaptor terminal must be used (e.g. Elpress type AKP).

If a screened or armored cable is used it shall be symmetrically designed. Chose conductor area depending on your environment and cable routing.

Power supply cable is routed through a cable flange situated in the roof of the control cabinet.

3x 380...400V 50Hz

Maximum short-circuit current 10kA. Required line fusing: Circuit breaker with minimum characteristic C or better. Max. fuse 40A

3x 440...480V 60Hz

Maximum short-circuit current 10kA. Required line fusing: Circuit breaker with trip characteristic K. Max. fuse 35A (40A if power limiting switch is used).

Earthing system

Within countries applying EU regulation, the protective earthing system shall be performed according to EN-60204-1, the PE conductor shall be dimensioned according to section 5.2, table 1.

Within countries applying US regulations, grounding and bonding equipment shall be performed according to the national electrical code, NFPA 70.

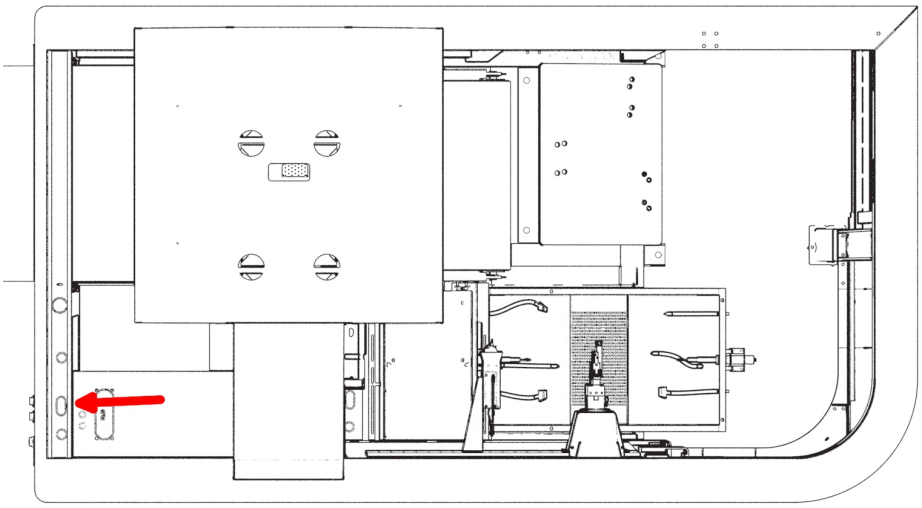
CAUTION

The FlexLoader SC 6000 is designed for use with the TNS earthing scheme.

3.4 Pneumatic installation

Connection point

Connect the incoming air connection to the FlexLoader SC 6000 at the indicated connection point at the FlexLoader SC 6000 upper rim. The incoming air must be 6 bar / 87 psi. Dry air, no oil.



xx1900000364

FlexLoader SC 6000 connection point compressed air

Air preparation adjustment

The pressure switch and the safety valve have been pre-configured with suitable settings. For detailed information, see *Air preparation units on page 160*.

3.5 Robot gripper connections

General

FlexLoader SC 6000 with base configuration has a number of predefined gripper signals that are routed to the robots **Customer Signals**/**Customer Power (CS/CP)** connector.

If the standard gripping option is chosen, these signals are already connected to the valve package.

Valve outputs

Six output signals are used for valve control (two-wire configuration). For details on **CS**/**CP** pin assignment see circuit diagrams.

Sensor inputs

Two input signals for gripper sensors are available (three-wire configuration). For details on **CS**/**CP** pin assignment see circuit diagrams.

3.6 Machine tool installation

General

FlexLoader SC 6000 connects to the machine tool by means of a safety interface and a functional interface.

The machine tool interface is normally prepared by the integrator and needs almost always customization.

Machine tool safety interface

FlexLoader SC 6000 with safety center has a safety interface that makes it possible to connect FlexLoader SC 6000 to the machine tool safety circuits.

The following standard connections are provided:

- One machine tool
  - Two-channel potential-free emergency stop from FlexLoader SC 6000 to machine tool
  - Two-channel potential-free emergency stop from machine tool to FlexLoader SC 6000.
  - Two-channel potential-free protective stop from FlexLoader SC 6000 to machine tool
  - Two-channel potential-free protective stop from machine tool (can be configured as dynamic signal) to FlexLoader SC 6000.
- Machine tool can behave as emergency stop master, emergency stop slave or emergency stop many-master (if safety center is customized to more than one external master).
- Chained connection for two doors (dynamic Eden signals) with signals for door locking.
- The physical interface is constituted by terminal blocks in the control cabinet.
- The overall safety functions in FlexLoader SC 6000 fulfill the requirements for PLd according to EN ISO 13849-1.
- The overall safety function in the machine tool must fulfill the requirements for PLd according to EN ISO 13849-1.

Refer to *Application manual - FlexLoader Standard Safety Center* for more information on installation of the safety interface.

Machine tool functional interface

The functional interface can communicate by means of any available physical interface, or using interface converters.

The user is expected to add appropriate interface signals to the robot, e.g. by ordering the robot with suitable fieldbus capabilities.

The FlexLoader SC 6000 standard configuration includes:

- Digital 24 V I/O interface with 16 IN and 16 OUT signals to be connected to the machine tool.
- Digital I/O is working with the 24 V of the machine tool.

Continues on next page

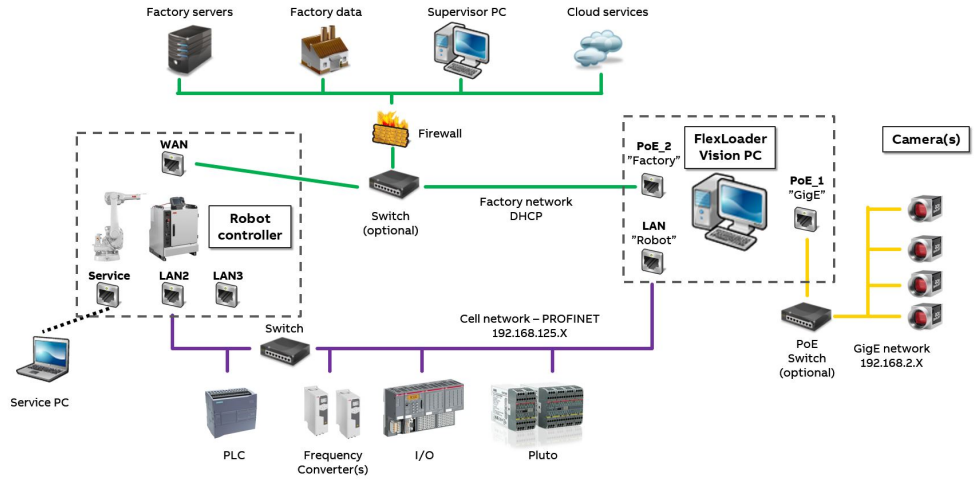
- Other physical interfaces are available by using optional robot field bus or interface options (to be ordered with robot).
- ABB DC500 series I/O node connected to the robot PROFINET network.

The functional interface must be configured and commissioned prior to use of the FlexLoader SC 6000.

3.7 Network connections

Network overview

The standard network setup with its three main networks is configured as shown below.



xx1800002353

- The factory network connects to both the FlexLoader Vision PC and the robot. This network is optional.
- The camera network is reserved for FlexLoader Vision cameras.
- The internal cell network connects all internal components, mainly through PROFINET communication (using robot option 888-2).

PROFINET configuration

The FlexLoader function package has pre-configured PROFINET settings. If any changes are needed, at least some of the steps below must be performed.

Changes to PROFINET configurations shall only be done by trained technicians. Technicians shall have a good knowledge of I/O settings both in general and in RobotStudio.

	Action
1	Set your PROFINET network in the robot to **Private network** in RobotStudio. Use **Controller** -> **Configuration**. Modify IP settings to fit the pricate network.
2	Configure devices for your robot controller in RobotStudio I/O configurator (IOC). Use **Controller** -> **Configuration**. Import all necessary GSDML files or modify the IPPNIO.xml file. Locate devices with the RobotStudio ScanTool and add relevant devices to the controller. Modify names and addresses if necessary, see *Standard network IP settings on page 51*. All sub-modules are needed when adding PLUTO devices.

Continues on next page

	Action
3	If relevant, configure Safe I/O. As general recommendation, two channel signals can be defined as separate single channel signals that are combined by PreLogic in SafeMove. Ensure correct safety checksum by using **Controller** -> **Configuration** -> **Vendor Tool**. The generated checksum is then used in the SafeMove configuration.
4	Set device names on devices with RobotStudio ScanTool. Modify the devices names if necessary, see *Standard network IP settings on page 51*. Other tools can be used if desired.

The PROFINET controllers in FlexLoader function packages normally uses DCP (Discovery Configuration Protocol), i.e. that they will configure devices when they found them.

The key property for this method is the device name (same as station name). The controller is told which devices to handle and communicate with and which device names they have. With this information, the controller finds the actual devices with the right names on the network and automatically assigns IP addresses that are pre-configured in the controller.

ABB CI502 I/O nodes will get its name depending on the switches/knobs on the module, e.g. **ci502-pn-01** if knobs are put in positions 0 (x10H) and 1 (x01H). A cold start is needed to read new name. Do not use knob positions 0 and 0, as these are used for user-defined names.

Robot PROFINET devices such as I/O nodes and Pluto gateway will get their IP address as warm start as defined in robot I/O configurator.

Internal cell network

The internal cell network connects the FlexLoader Vision PC, robot, safety devices, I/O devices, frequency converters and other internal cell components to each other.

The PC network connection is named **Robot** on physical port **LAN**.

This network is the robots private network, and both robot **LAN2** and **LAN3** as well as the service port are connected to this network. The robot is acting as PROFINET master.

Tip

The FlexLoader Vision PC can only be connected to **LAN2**.

A separate network connection to the robot network can be made by using **LAN3** network isolation.

In this case, the service port remains internally connected to **LAN2**.

See robot documentation for further information.

Note

Do not connect the internal network directly to the factory network.

Continues on next page

Limitations

Currently, the PROFINET master functionality of the robot does not allow to configure a connection to a Siemens IDevice, i.e. direct connection to another Siemens PLC acting as PROFINET master is not possible.

The robot PROFINET functionality can be used on only one of the following connections: **WAN**, private network (service port, **LAN2**, **LAN3**), or isolated **LAN3** network.

The private network of two robot controllers cannot be directly connected.

For connecting two robot controllers or an external master PLC to the cell, see *External PROFINET connections on page 52*.

Note

Robot option 841-1 Ethernet/IP™ cannot be combined with the FlexLoader function packages and standard cells.

If Ethernet/IP™ communication is needed, the robot option 840-1 Ethernet/IP™ fieldbus adapter must be used instead.

Robot service connection

A robot service PC can be connected to the robot controller through the service port of the robot controller. For more information, see the robot product manual.

Factory network

The FlexLoader Vision PC can be connected to a factory network. The PC network connection is named **Factory** on physical port **PoE2**.

Please refer to the robot controller product manual for setting the robot IP configuration on the **WAN** port.

This network is controlled by the customer, who is responsible for appropriate network setup and network protection mechanisms.

Note

Please refer to the section on cyber security in the robot integrator's guide for additional information.

The use of the FlexLoader Vision App requires an active internet connection to the FlexLoader Vision PC.

Camera network

The FlexLoader Vision PC is connected to the cameras on a separate internal network.

The PC network connection is named **GigE** on physical port **PoE1**.

Continues on next page

A single camera is connected to a PoE port of the FlexLoader Vision PC. Several cameras are typically connected by means of a PoE switch.

Tip

If the factory network is not used, the **PoE2** port can be used to connect and power a second camera.

Standard network IP settings

Internal machine network

Equipment	IP address	Gateway	Device name
PC	192.168.125.150	192.168.125.1	-/-
Robot	192.168.125.1	192.168.125.1	irc5-pnio
PLC camera 1	192.168.125.151	192.168.125.1	e.g. plc-fp100
PLC camera 2	192.168.125.152	192.168.125.1	e.g. plc-fp300
PLC camera 3	192.168.125.153	192.168.125.1	e.g. plc-fp400
PLC camera 4	192.168.125.154	192.168.125.1	e.g. plc-fp600
Pluto gateway	192.168.125.160	192.168.125.1	gatepn
I/O board 1	192.168.125.161	192.168.125.1	ci502-pn-01
...	...	...	
I/O board 9	192.168.125.169	192.168.125.1	e.g. ci502-pn-09
Frequency converter 1	192.168.125.171	192.168.125.171	e.g. bufferbelt2, separationbelt, camerabelt, pallettipper
...	...	...	...
Frequency converter 9	192.168.125.179	192.168.125.179	
Project specific equipment 1	192.168.125.180	192.168.125.1	
...	...	...	
Project specific equipment 20	192.168.125.199	192.168.125.1	

Internal camera network

Equipment	IP address	Gateway
PC	192.168.2.1	Do not specify gateway.
Camera 1	192.168.2.2	Do not specify gateway.
Camera 2	192.168.2.3	Do not specify gateway.
Camera 3	192.168.2.4	Do not specify gateway.
Camera 4	192.168.2.5	Do not specify gateway.

Continues on next page

Factory network

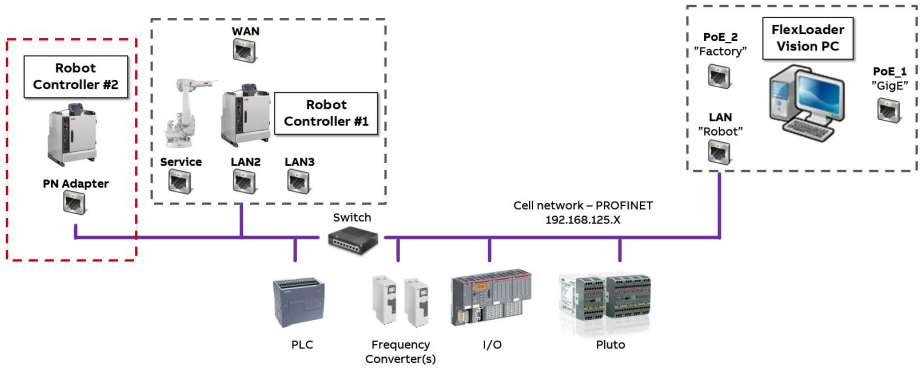
DHCP is used as standard setting.

> **Note**
>
> The network configuration on the factory network can be freely adopted to the customers network.

External PROFINET connections

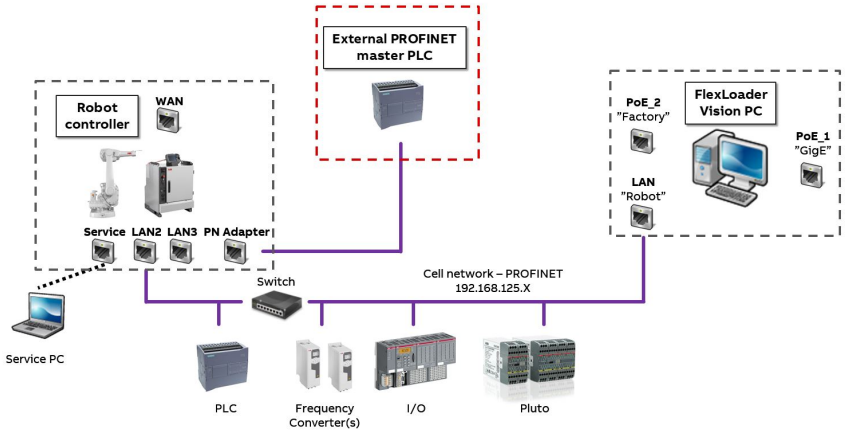
In some cases external PROFINET connections are needed during integration of a FlexLoader function package or standard cell.

A second robot controller can be connected through PROFINET by using a PROFINET fieldbus adapter in the second controller (robot option 840-3).



xx1800002906

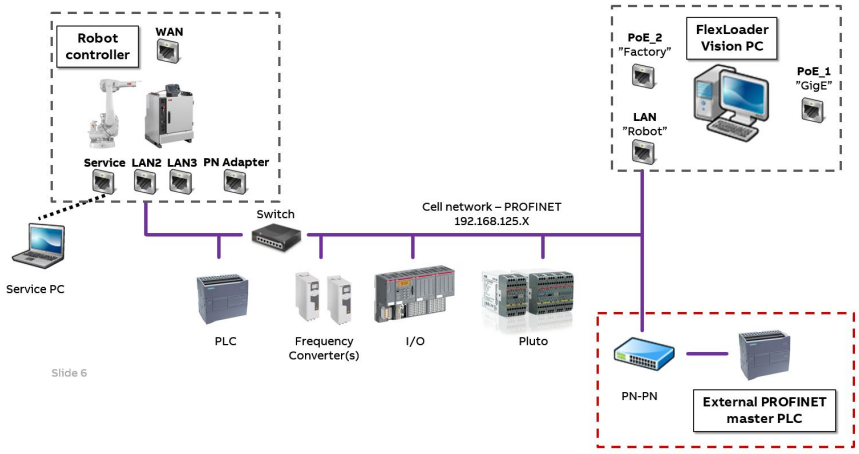
A master PLC be connected through PROFINET by using a PROFINET fieldbus adapter in the first controller (robot option 840-3).



xx1800002905

Continues on next page

Alternatively, the master PLC can be connected by using a PNPN-coupler.



xx1800002907

Note

The connections can in principle be routed through the WAN port as well, with all internal network components connected to **WAN** instead of **LAN2**.

However, this will expose all components to an external network. We recommend to avoid this system setup.

3.8 Network security

Network security

This product is designed to be connected to and to communicate information and data via a network interface. It is your sole responsibility to provide, and continuously ensure, a secure connection between the product and to your network or any other network (as the case may be).

You shall establish and maintain any appropriate measures (such as, but not limited to, the installation of firewalls, application of authentication measures, encryption of data, installation of anti-virus programs, etc) to protect the product, the network, its system and the interface against any kind of security breaches, unauthorized access, interference, intrusion, leakage and/or theft of data or information. ABB Ltd and its entities are not liable for damages and/or losses related to such security breaches, any unauthorized access, interference, intrusion, leakage and/or theft of data or information.

3.9 Licensing

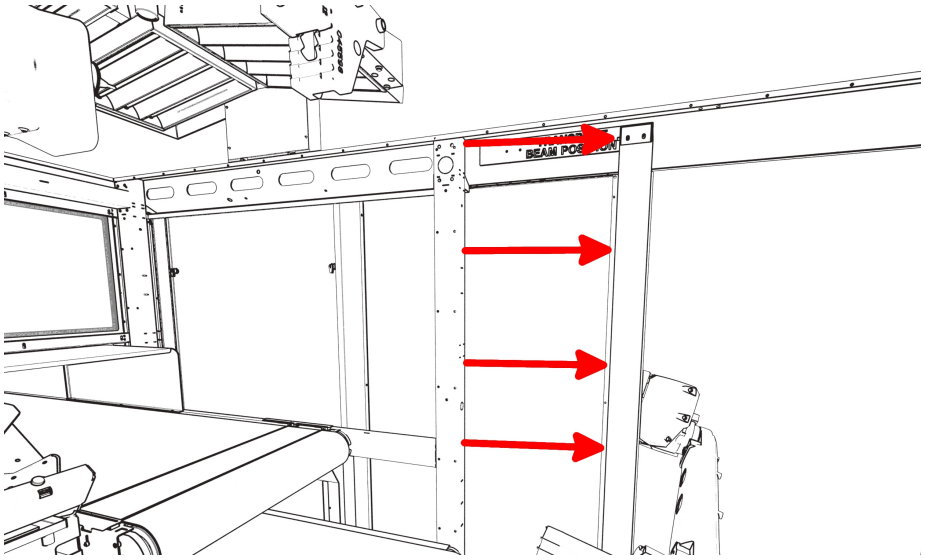
By installing and using FlexLoader SC 6000 and its FlexLoader Vision you agree to Microsoft's and Matrox's license agreements.

3.10 Moving the FlexLoader SC 6000

CAUTION

Mechanically stabilize the FlexLoader SC 6000 prior to transporting it again.

The stabilizing support close to the FlexLoader SC 6000:s machine opening has to be fitted to the FlexLoader SC 6000 prior to any transport. This support was delivered and fitted to the FlexLoader SC 6000 for original delivery.



xx1900000359

Transportation stability ensured by support.

4 Commissioning

4.1 User accounts

General

Note

There are several default user accounts used in FlexLoader SC 6000. It is strongly recommended to change the default passwords to reduce the risk of unauthorized access to important system parameters.

Follow ABB's advice for administration of user permissions.

Note

Please refer to the section on cyber security in the robot Integrator's guide for additional information.

FlexLoader Vision

The FlexLoader Vision system has two default user accounts:
For operation

User name	Password
FlexLoaderVision	FlexLoaderVision

For adminstration

User name	Password
FlexLoaderAdmin	FlexLoaderAdmin

Robot

Default credentials according to ABB default settings.

User name	Password
Default User	robotics

Safety related work (i.e. SafeMove) is performed with the SafetyUser account.

User name	Password
SafetyUser	SafetyUser

Follow ABB's advice for administration of user permissions.

Pluto

The Pluto program normally does not need to be opened or changed. Information on download and reconfiguration of the software together with the passwords needed in case it would be necessary to make changes are found in the FlexLoader Standard Safety manual.

4.2 Powering on the FlexLoader SC 6000

1 Turn on the main power supply. Wait until all devices have started up.
2 Perform a power on safety check:
   - Press and restore at least one internal FlexLoader SC 6000 emergency stop button.
   - Reset the emergency stop.
   - Open and close at least one internal FlexLoader SC 6000 cell door.
   - Reset the door status by shortly turning the reset key.

     Optionally, if a light curtain is present, a long light curtain reset may be needed (0.5 s) first, followed by a second reset for the whole cell.
   - Refer to the safety manual for more detailed information.
3 Check the function of the external emergency stop
4 Check the function of the external protective stop (if present).

DANGER

Do not continue to work until the safety circuits are functioning correctly.

4.3 Vision system FlexLoader Vision

FlexLoader SC 6000 is delivered with FlexLoader Vision pre-installed and pre-configured on the FlexLoader Vision PC.

Refer to the FlexLoader Vision manual for detailed information on teachin, configuration and problem solving etc.

Tip

Apart from Windows security updates, do not install any updates or patches if not advised from ABB or ABB:s subsupplier.

- Visually check that all parts of the lighting system are working. If not, replace illumination according to the maintenance section.
- Visually check the camera image for any signs of misalignment, blurring, dirt etc. If necessary, take appropriate actions, e.g. realignment, adjusting focus or lens cleaning, according to the maintenance section.
  - Adjust focus: Place a sheet of paper with text or information below the camera on the belt. If you intend to run FlexLoader SC 6000 with tall details, put the sheet on an appropriate height. Adjust the lens focus ring and lock it.
    For additional help, use the focus value in the large view. Normally, a higher value indicates a better focus.
  - Adjust the lens aperture: Set the exposure time in FlexLoader Vision to ~20% of the maximum value. Adjust the contrast to ~50% of the slider maximum. Place appropriate details on the belt and adjust the lens aperture ring in order to obtain a reasonable image. Lock the aperture ring.

Note

Keep in mind that the FlexLoader Vision PC has limited system resources for non-vision related tasks. Do not strain the memory or the hard disk space.

Tip

Immediately after initial commissioning, start maintaining your own system backups.

Tip

If you for some reason would lose the pre-installation and pre-configuration data of the FlexLoader Vision system, you can find a primary backup. See *Software on page 164* for detailed information.

4.4 Robot system

4.4.1 Introduction

General

FlexLoader SC 6000 is delivered with a pre-installed and pre-configured robot gripper system (option) and robot program.

For details on the structure of the FlexLoader SC 6000 robot software see *RAPID program on page 91*.

Commissioning includes the following main steps:

- Adjusting the robot coordinate systems.
- Establishing the FlexLoader SC 6000 – machine tool communication interface.
- Adapting the robot program to the individual automation needs.

Tip

FlexLoader SC 6000 commissioning uses standard robot functionality for maintaining work objects, tool data and other settings.
Refer to the robot product manual for this standard functionality.

Base coordinate system of the robot

The robot is mounted on an inclined pedestal for better reach and accessibility to the machine tool. In order to simplify handling and jogging of the robot, the robot base coordinate system shall be parallel to the floor (not angled), preferably with either the X or Y axis parallel with the front of the machine tool.

The following values can be changed in the robot configuration (cf. **MOC.CFG**) in order to take inclination and rotation into account if the robot is inclined or rotated in relation to the desired coordinate system.

- gravity_alpha, gravity_beta
- base_frame_orient_u0, base_frame_orient_u1, base_frame_orient_u2, base_frame_orient_u3
- base_frame_x, base_frame_y, base_frame_z

DANGER

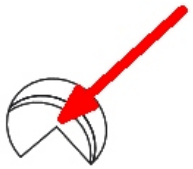
Do not change the position or orientation of the robot base coordinate system, as this will affect the SafeMove configuration.

4.4.2 Calibration of the coordinate systems

Robot coordinate systems that are used in the FlexLoader SC 6000 are pre-defined from factory, but they must be redefined to compensate for possible changes due to transportation and setup.

Calibration point markers

Most calibration points are represented by circles with a pointing mark. The tip of the calibration tool should be as close as possible to the center of the circle, as pointed out by the pointing mark.



xx1900000410

Figure 4.1: Appearance of a typical calibration mark

Calibration tool fitting

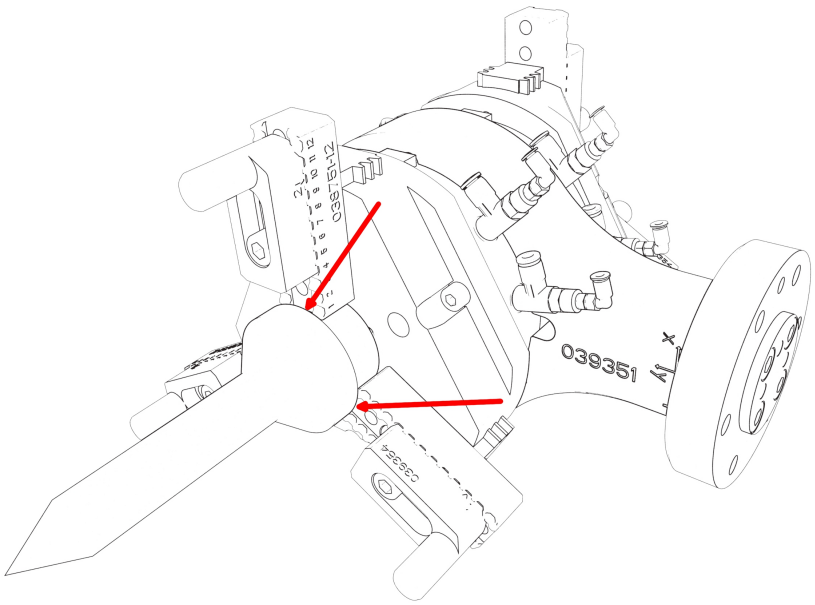
The following information applies to standard grippers only:

Place and grip the calibration tool in the gripper. Make sure that the flat surface is firmly pressed on the contact points on the adjustable finger base, as pointed out in the illustration below.

Continues on next page

4.4.2 Calibration of the coordinate systems
Continued

The calibration tool is predefined as **tCalibTool1** (if calibration tool is gripped in gripper 1), or **tCalibTool2** (if calibration tool is gripped in gripper 2). For gripper notation, check the marking on the gripper unit.



xx1900000365

Figure 4.2: Calibration tool fitted to the gripper.

Prior to the actual calibration, check the accuracy of the calibration tool by appropriate means, e.g. by reorienting with respect to a well-defined position. This can be done by reorienting the robot calibration tool around a sharp tip. If necessary, redefine the tool definition for the calibration tool.

Tip

Follow the guidelines for calibration tools and work objects as described in the robot manuals.

General work object considerations

Some standard work objects in the FlexLoader SC 6000 actively use both user frame and object frame. When doing standard 3-point calibration, the operator must calibrate by means of the user method. The object frame is used internally in order to move work objects to specific points of interest.

Calibration aiding procedures

All of the work objects below shall be calibrated by standard 3-point work object calibration. Some of them can use a point-of-interest, where parametrized movement routines will perform their work.

Continues on next page

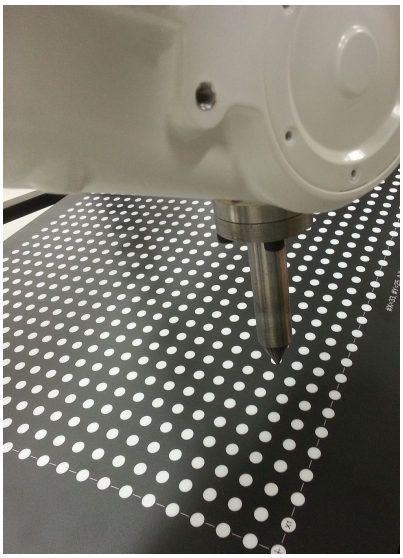
In some cases, the tool orientation in the point-of-interest can be used during later movements. Refer to each work object for more detailed information.

If a point-of-interest is defined, FlexLoader SC 6000 supplies a calibration routine that assists the operator in doing the calibration. All calibration aiding procedures have the same appearance. The operator confirms calibration of the selected point-of-interest, selects which calibration tool is used, follows the instruction on where and how to point the calibration tool, and confirms the current position.

The following coordinate systems shall be redefined (if the corresponding option is present on the FlexLoader SC 6000). The position of the calibration points can be found in the images below.

Camera coordinate system wCamera1

Coordinate system for picking from inconveyor. Follow the calibration instructions in the FlexLoader Vision manual. Perform a standard 3-point calibration by use of the FlexLoader Vision calibration plate.



xx1800000260

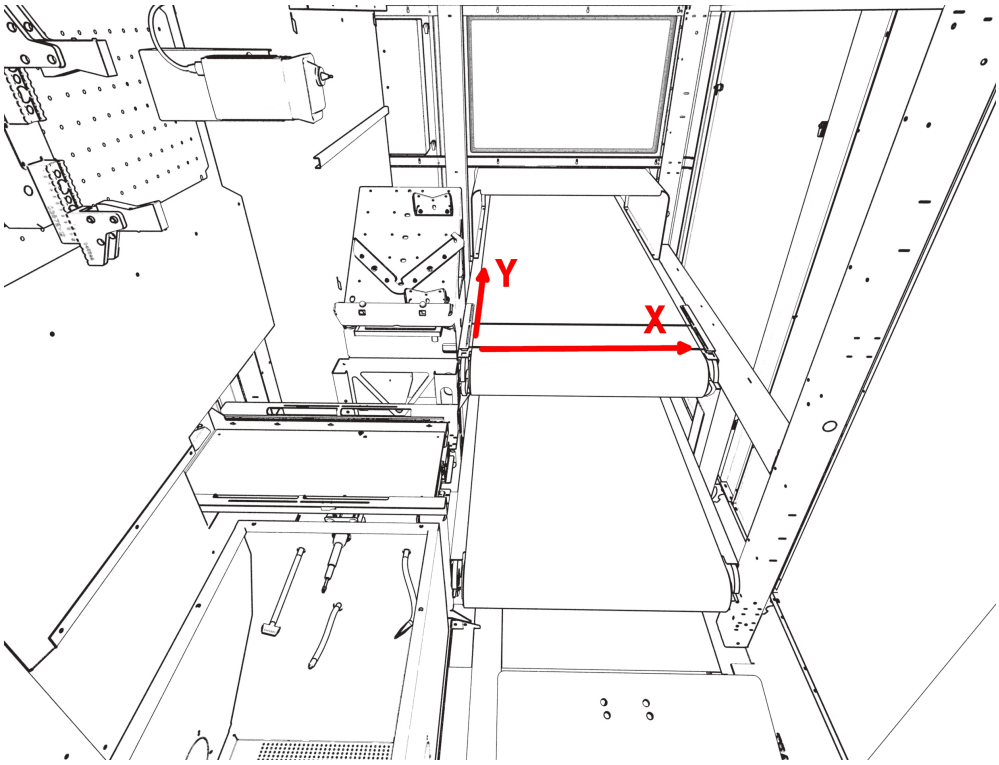
Disturbing parts outside of the calibration plate image can be suppressed by defining a region-of-interest (only valid during calibration). Click the region-of-interest button to the left of the camera image and draw a red rectangle around the valid calibration plate part of the image.

Perform the calibration as usual.

Continues on next page

4.4.2 Calibration of the coordinate systems
Continued

Approximate positioning of wCamera1 in the FlexLoader SC 6000.



xx1900000366

Finally, check the calibration by the following simple test.

Move the robot calibration tool tip to point x=0, y=0, z=0 in wCamera1. Use the mouse curser coordinate display in FlexLoader Vision and point at the image of the calibration tool tip. The displayed value should be close to x=0, y=0.

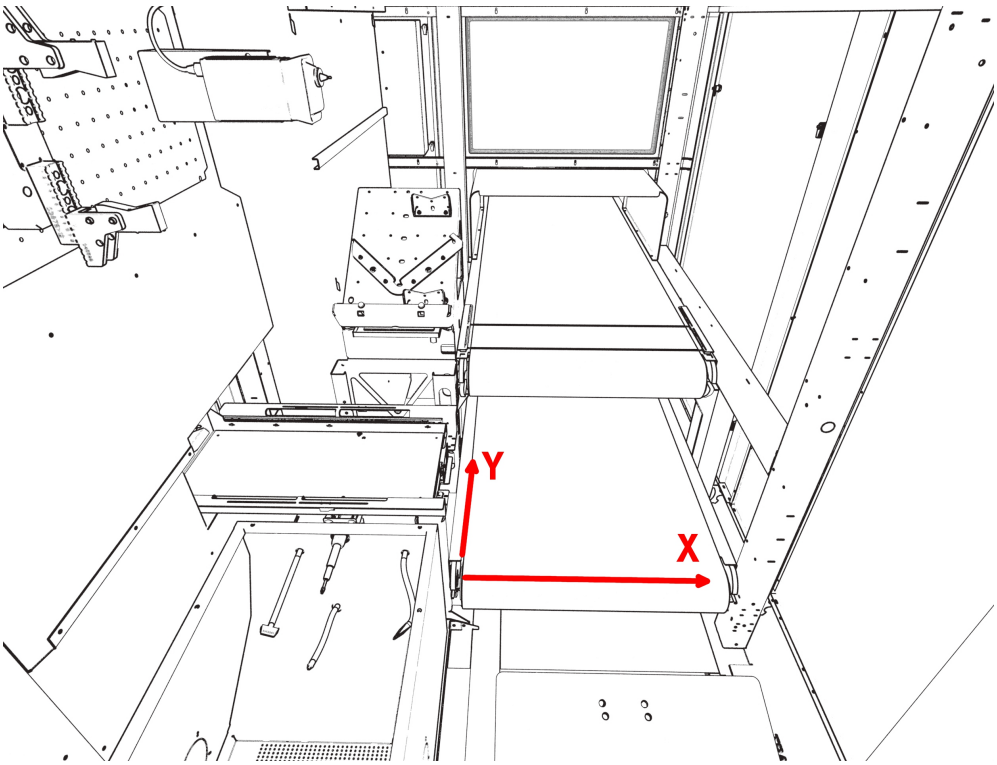
Outconveyor coordinate system wFeederOut

Coordinate system for leaving on outconveyor. The X-axis is above the idler roller shaft, the Y-axis is at the left belt edge and in the direction of movement of the belt, and the Z-axis points upwards. Perform a standard 3-point calibration.

Afterwards, call the calibration routine CalibSpeedFeederOut. Depending on the use of the system, the point-of-interest, pLeaveFeederOut, is used differently. In

Continues on next page

FlexLoader Vision Lite applications, this point is calculated from detail data. In user-defined details, the point has to be defined and displaced by user code.



xx1900000367

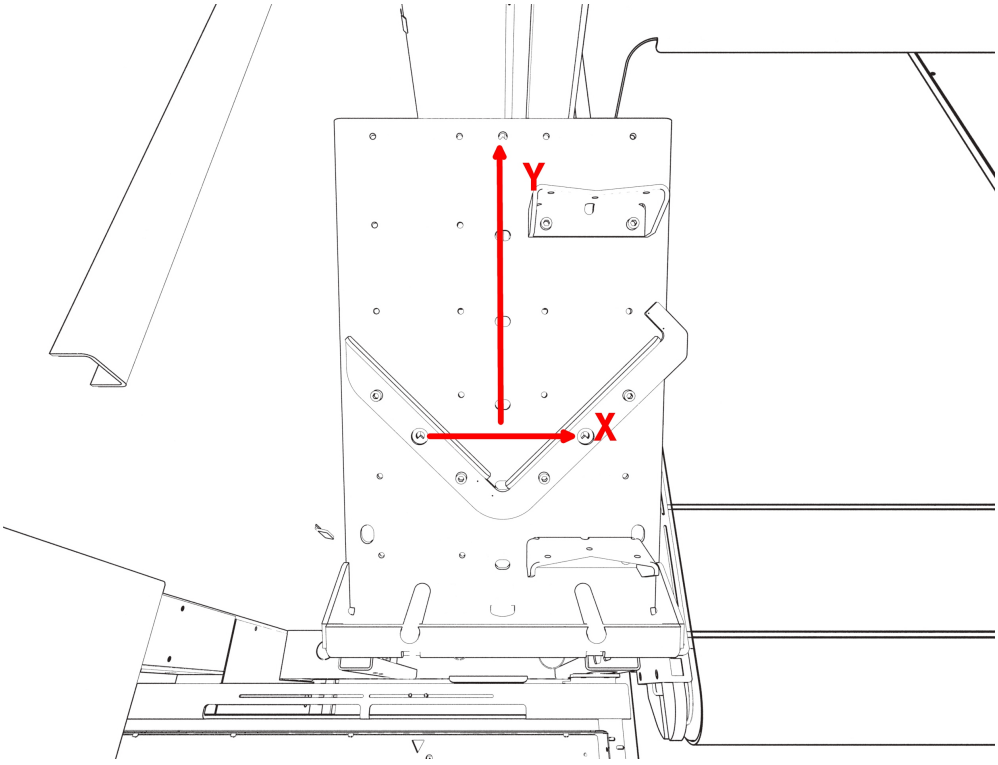
Approximate positioning of wFeederOut in the FlexLoader SC 6000

Continues on next page

4.4.2 Calibration of the coordinate systems
Continued

Re-grip table wRegrip

The re-grip table coordinate system wRegrip is oriented as follows. Standard 3-point calibration.



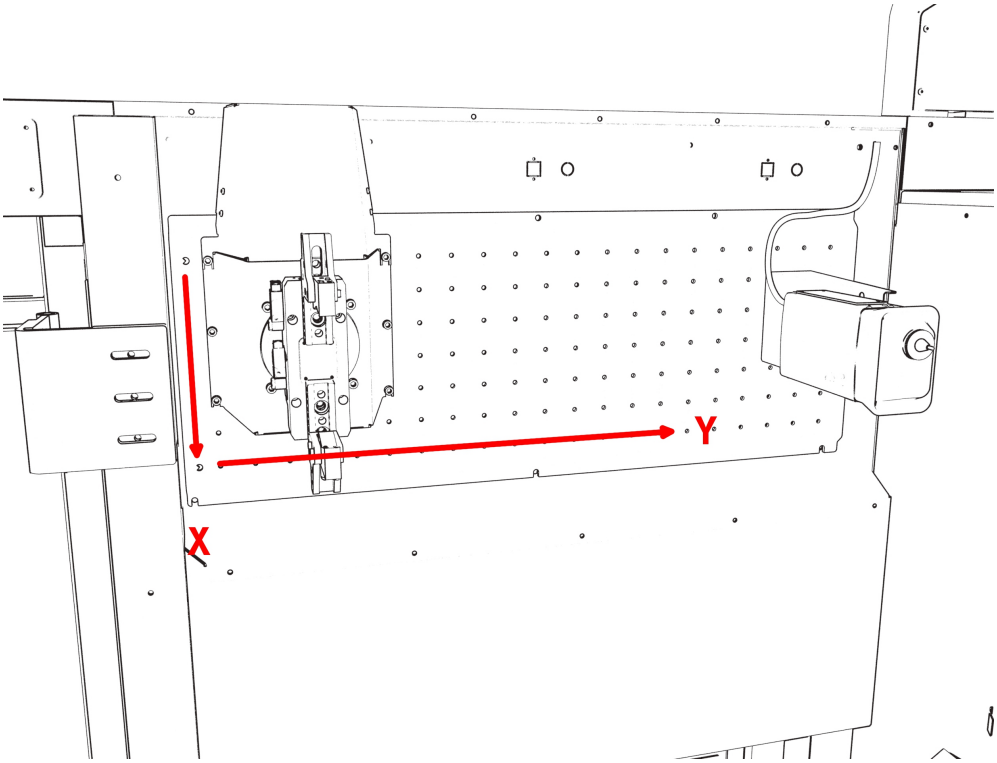
xx1900000368

Approximate positioning of wRegrip in the FlexLoader SC 6000

Continues on next page

Turning station wTurnStation

The turning station coordinate system wTurnStation is oriented as follows. Standard 3-point calibration.



xx1900000369

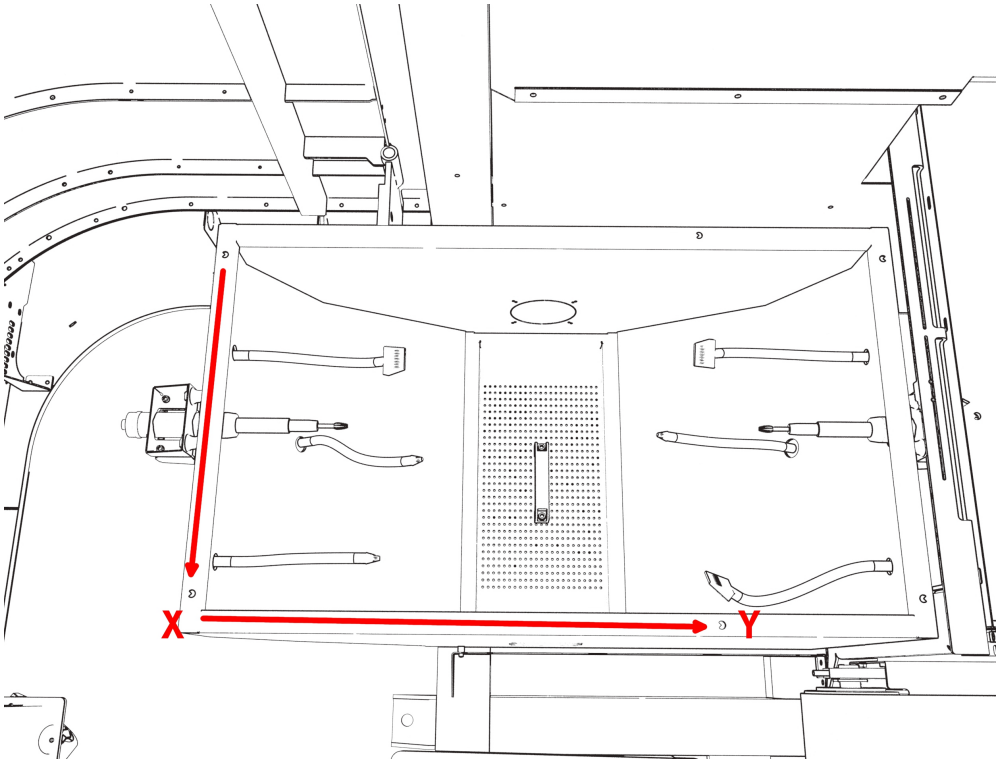
Approximate positioning of wTurnStation in the FlexLoader SC 6000

Continues on next page

4.4.2 Calibration of the coordinate systems
Continued

Air cleaning box wAirClean

The air cleaning box coordinate system wAirClean is oriented as follows. Standard 3-point calibration. Afterwards, call the calibration routine **CalibAirClean**. The point-of-interest should be in the middle of the air cleaning box, at the height where the air cleaning starts. Note that orientation of the tool is used in pre-configured movements.



xx1900000370

Approximate positioning of wAirClean in the FlexLoader SC 6000

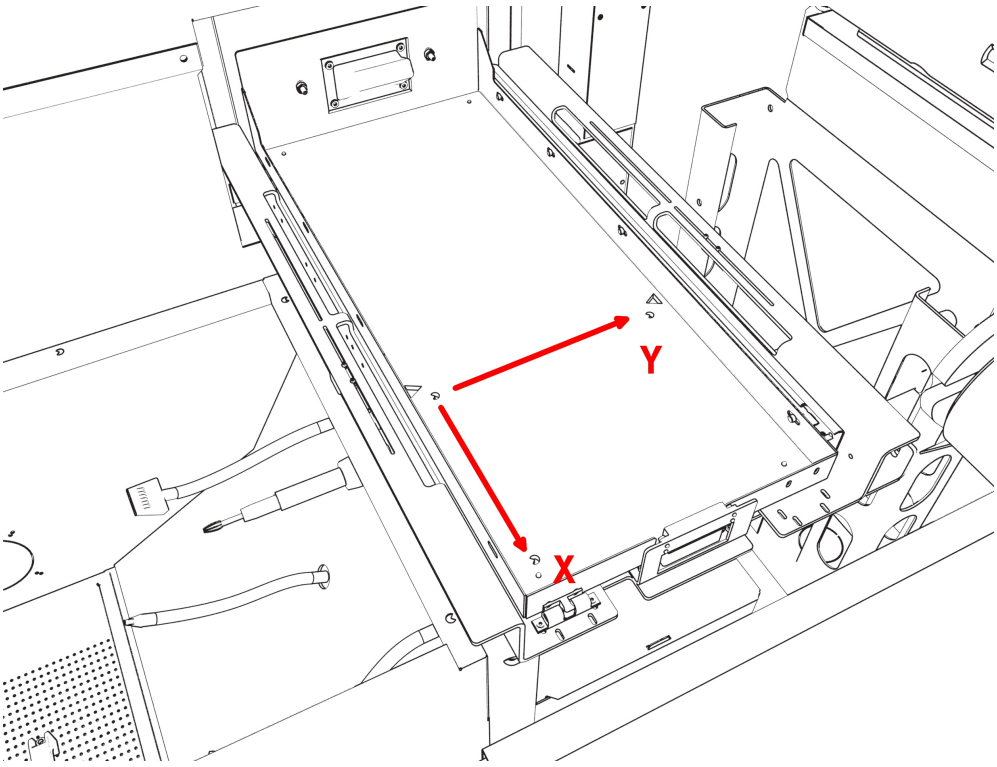
Deburring tools wDeburr

The grinding and deburring tools are accessed from their own coordinate system wDeburr. In the pre-configured setup, wDeburr is identical to wAirClean. Standard 3-point calibration.

Continues on next page

Statistical outlet wSample

The statistical outlet coordinate system wSample is oriented as follows. Standard 3-point calibration. Afterwards, call the calibration routine **CalibSampleOutlet**. The point-of-interest should be in the center of the sample outlet surface, where the sensor will detect details. Note that orientation of the tool can be used.



xx1900000371

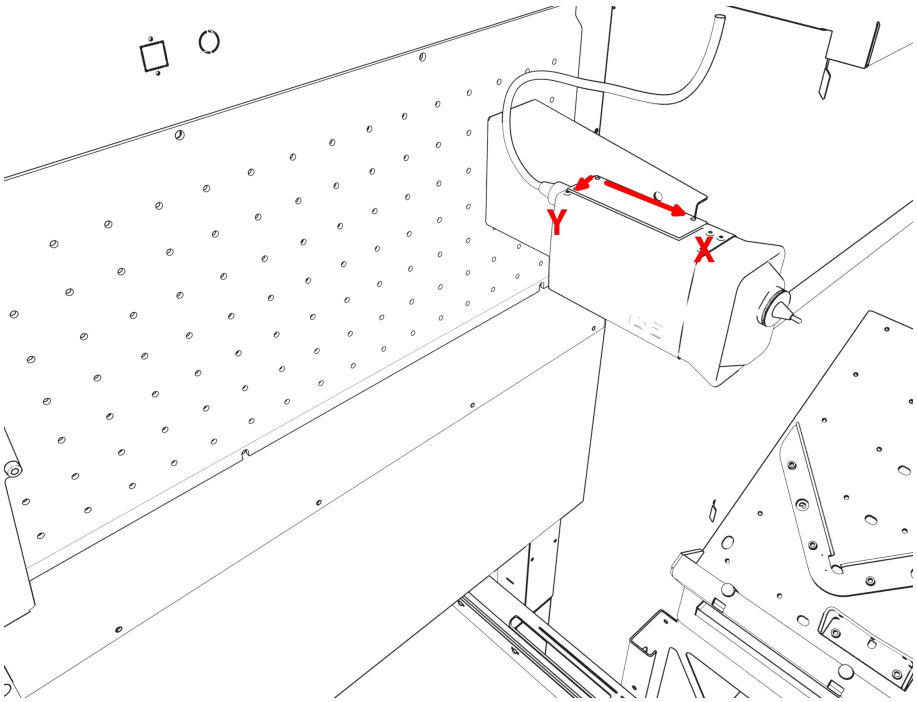
Approximate positioning of wSample in the FlexLoader SC 6000

Continues on next page

4.4.2 Calibration of the coordinate systems
Continued

Marking unit wMarker

The marking unit coordinate system wMarker is oriented as follows. Standard 3-point calibration. Use the three mounting holes on the marking unit as calibration points (X1, X2, Y1) and lower the calibration tool into the holes. No exact positioning of this coordinate system is necessary due to the nature of the marking process.



xx1900000372

Approximate positioning of wMarker in the FlexLoader SC 6000

Afterwards, call the calibration routine **CalibMarker**. The point-of-interest is where the calibration tool touches the marking needle, with the marking needle at a suitable marking distance. Note that orientation of the tool can be used.

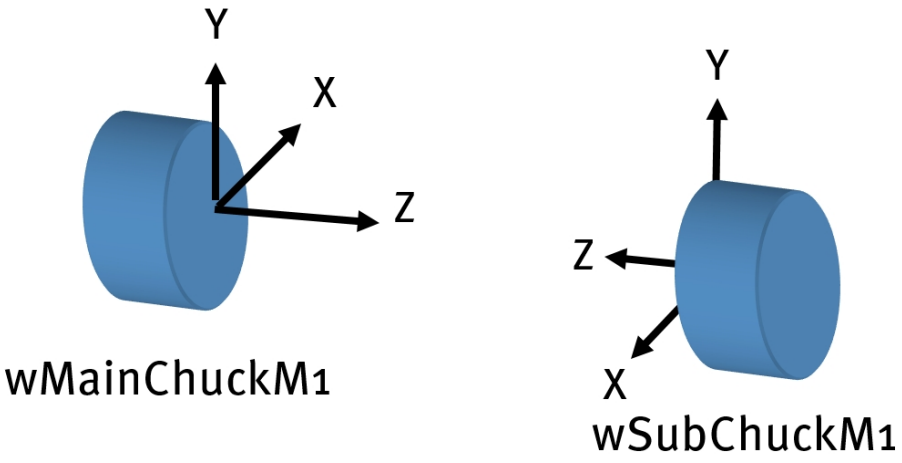
Main and sub chuck of the machine tool

When using FlexLoader Vision Lite, the following machine tool coordinate systems have to be defined.

Coordinate system **wMainChuckM1** and **SubChuckM1** for the main chuck and sub chuck of the machine tools.

Continues on next page

The X-Y plane of the coordinate systems for a chuck must be at the front plane of the main and sub chuck, with x=0,y=0 at the rotation shaft of the chuck. The typical machine tool for this application is a lathe.



xx1900000373

Perform a standard 3-point calibration of **wMainChuckM1** and **wSubChuckM1** according to the coordinate directions above. At this point, the origin of the coordinate systems does not have to be in the middle of the chuck.

Call the routine **CalibMainChuck** and follow the instructions. On request, a cylindrical detail must be gripped with the main chuck. Position the robot so that the detail can be gripped by the machine tool chuck gripper, precisely centered. Confirm to the calibration routine. From the current robot position, the object frame of **wMainChuckM1** is now moved to be located exactly in the center of the main chuck.

Call the routine **CalibSubChuck** and follow the instructions. On request, a cylindrical detail must be gripped with the sub-chuck. Position the robot so that the detail can be gripped by the machine tool chuck gripper, precisely centered. Confirm to the calibration routine. From the current robot position, the object frame of **wSubChuckM1** is now moved to be located exactly in the center of the main chuck.

If the chucks are movable, check that the z-axis in **wMainChuckM1** and **wSubChuckM1** is in line with the actual chuck movement (even for long movements).

4.4.3 Further integration steps

The pre-configured robot software for FlexLoader SC 6000 contains suggestions for a working application skeleton.

Tip

Please follow the suggestions in the section about robot program for building your own applications, see *RAPID program on page 91*. Further in-depth reading of the FlexLoader Vision manual is strongly recommended.

Note

Immediately after initial commissioning, start maintaining your own system backups.

If you for some reason would lose the pre-installation and pre-configuration data of the robot system, you can find a primary backup on the FlexLoader Vision system harddisk. See *Software on page 164* for detailed information.

4.5 Machine tool

The FlexLoader SC 6000 system provides a framework for handling the most common machine tool interface types.

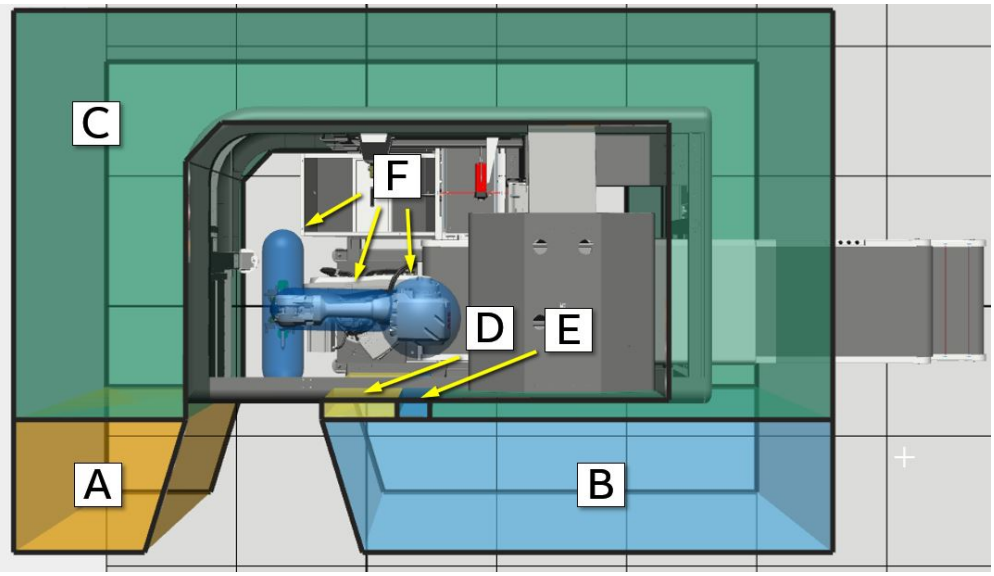
Interaction with the machine tool is handled by a machine tool specific module (**TemplateMachine.sys**). This module translates the actual machine tool communication signals into a standardized set of commands to be understood by FlexLoader SC 6000.

This standardized set of commands can be expanded and adapted. Due to the large variety of machine tool interfaces and integrator needs, it is the integrators responsibility to implement this interface module.

See *RAPID program on page 91* for further implementation suggestions.

4.6 SafeMove

The FlexLoader SC 6000 comes with a standardized setup for the SafeMove system. The delivery setup is as follows:



xx1900000414

Schematic positions of SafeMove zones within the FlexLoader SC 6000. The marked areas are forbidden zones.

Pos.	Description	Pos.	Description
A	SafeZone 1	D	SafeZone 4
B	SafeZone 2	E	SafeZone 5
C	SafeZone 3	F	Robot and tool geometry

- SafeZone3, SafeZone4 and SafeZone5 define forbidden zones based on the contours of the FlexLoader SC 6000 shell.
- SafeZone1 and SafeZone2 define forbidden zones based on the application area depending on the machine tool and the area between machine tool and FlexLoader SC 6000. These zones are predefined, but must be adopted by the integrator.
- The Stand Still Supervision zone SST ensures that the robot is kept at standstill when the statistical outlet is opened. Opening the statistical outlet while the robot is moving leads to an immediate SaveMove safety violation. SST is activated by an external sensor supervising the statistical outlet.
- The global Tool Speed Supervison zone Global_TSP limits the maximum robot speed to 1300 mm/s (-> use max v1000 in robot).

When delivered, all SafeZones (as defined above) are activated. If the opening towards the machine tool is enlarged by removing one of the flexible panels, the corresponding SafeZone should be deactivated.

- Standard opening (850 mm): SafeZone4 and SafeZone5 must remain activated.

Continues on next page

- Medium opening (1300 mm): SafeZone4 can be deactivated.
- Wide opening (1500 mm): SafeZone4 and SafeZone5 can be deactivated.

SafeZone1 and SafeZone1 must always be adopted to the actual machine tool layout.

Follow standard SafeMove configuration routines as described in the robot manual.

DANGER

If the intended use of the FlexLoader SC 6000 is changed, a new risk assessment with regard to SafeMove configuration must be done.

DANGER

SafeToolZones defined within the FlexLoader SC 6000 (STZ4, STZ5, STZ6, STZ7, SST1) shall not be modified more than described above.

DANGER

SafeToolZones depending on the machine tool must be adopted to the actual layout and machine too. Risk assessment shall be performed.

4.7 Marking unit

General

The marking unit is a standard FlexLoader SC 6000 option. The hardware consists of a SIC marking equipment, i.e. an i53 marking head with an E10 controller unit.

During operation, the robot communicates by means of serial communication with the controller unit. The robot must thus be equipped with the serial communication option (970-1).

For in-depth information refer to the section on pre-configuration data.

Configuration changes

The controller unit is pre-configured for use with the FlexLoader SC 6000. Configuration changes can be done by using the SIC controller configuration software (delivered on the FlexLoader Vision storage, to be installed on a separate PC, communication with standard USB cable).

Typical changes might be modifications of the marking setup files. FlexLoader SC 6000 is using three different marking files for different sizes: SMALL, MEDIUM, LARGE.

For in-depth information refer to the section on pre-configuration data, especially if you intend to modify the marking units configuration.

4.8 Deburring/grinding unit

The deburring/grinding unit is a standard FlexLoader SC 6000 option. The hardware consists of an Atlas Copco pneumatic die grinder LSF28 ST030E and a Nitto pneumatic filer ASH-900.

These units can be equipped by any suitable tools according to the customer needs.

DANGER

The original risk assessment allows users to run the deburring/grinding tools under the assumption that no irreversible damage can occur.
If other tools are used, this risk assessment has to be re-evaluated by the integrator/user of the FlexLoader SC 6000.

This page is intentionally left blank

5 Interface

5.1 Safety interface

Introduction

The FlexLoader SC 6000 is equipped with a well proven multi-purpose safety center that covers a large variety of standard uses. The safety center is normally connected directly to the machine tool(s).

The safety center handles all necessary connections to the door system and the robot safety.

It consists of an ABB Pluto system with programmable safety. Different usages are pre-programmed and selected by appropriate configuration.

Area of use

FlexLoader Standard Safety must only be used in cells with intended safety classification and on the condition that the applicable safety requirements and standards are followed.

Performance level

The safety functions for ABB's components (emergency stop and protective stop) according to this description fulfil the performance level PLd in accordance with EN ISO 13849:1.

Machine tool

Safety interface to machine tool

The following standard connections are provided:

- One machine tool
  - Two-channel potential-free emergency stop from FlexLoader SC 6000 to machine tool
  - Two-channel potential-free emergency stop from machine tool to FlexLoader SC 6000.
  - Two-channel potential-free protective stop from FlexLoader SC 6000 to machine tool
  - Two-channel potential-free protective stop from machine tool (can be configured as dynamic signal) to FlexLoader SC 6000.
- Machine tool can behave as emergency stop master, emergency stop slave or emergency stop many-master (if safety center is customized to more than one external master).
- Chained connection for two doors (dynamic Eden signals) with signals for door locking.
- The physical interface is constituted by terminal blocks in the control cabinet.

See *Application manual - FlexLoader Standard Safety Center*.

5.2 Function interface

General

FlexLoader SC 6000 has a function interface that is used to communicate with the robot and the rest of the robot cell.

The internal interface between robot and FlexLoader SC 6000 is described below.

Conveyor system <> Robot

Signals

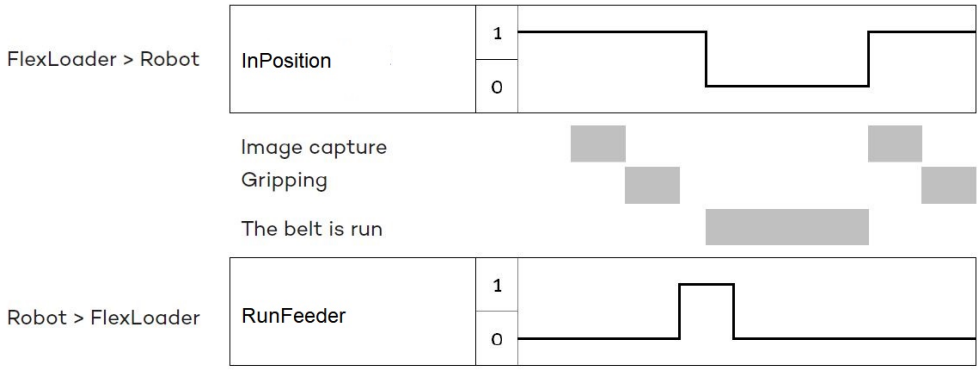
The interface for the inconveyor is described below.

Name	Description
diFeeder1InPosition	Sensor that indicates that parts are on the cameras field of view
diFeeder1TooFar	Sensor that indicates that parts are at the end of inconveyor
diFeeder1PartsLow	Optional sensor on inconveyor. Used for indication that parts on inconveyor are nearly finished.
diEmergencyStopOk	Emergency stop is not active
diFeeder1ManualForward	Knob for forward operation of inconveyor
diFeeder1ManualBackward	Knob for backward operation of inconveyor
diFeeder1SelectAutomatic	Knob for selecting method of operation (automatic – manual) of inconveyor
diOutFeeder1TooFar	Sensor that indicates that parts are at the end of outconveyor
diFeeder1SelectEmptying	Knob for selecting method of operation (automatic – emptying) of outconveyor
diAlarmU11FeederIN	Alarm from frequency inverter inconveyor
diAlarmU12FeederOUT	Alarm from frequency inverter outconveyor
doRunForwardBelt1	Runs inconveyor forward
doRunBackwardBelt1	Runs inconveyor backward
doManualLampFeeder1	Indication for operator when running inconveyor manually is permitted
doRunFeederOut1	Run outconveyor

Continues on next page

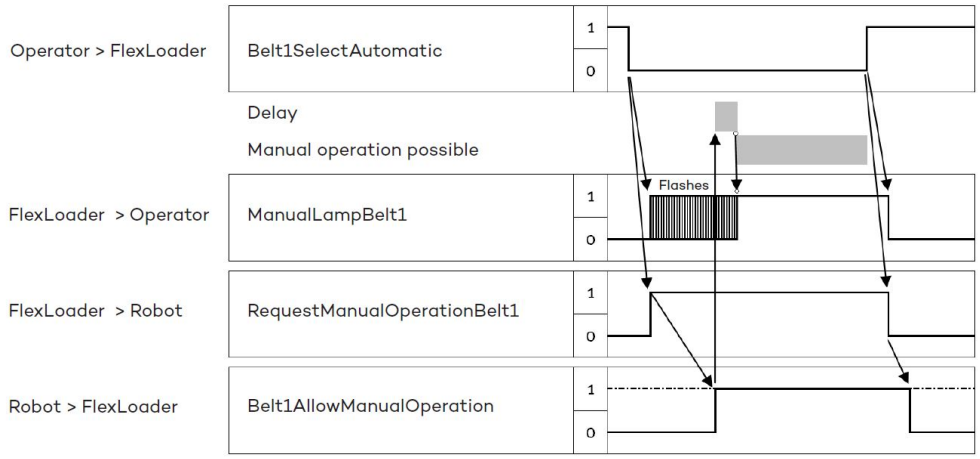
Inconveyor

During normal operation, signal exchange to robot must occur according to the following handshaking sequence:



xx1800000186

During operating mode switch, signal exchange to robot occurs according to the following handshaking sequence:



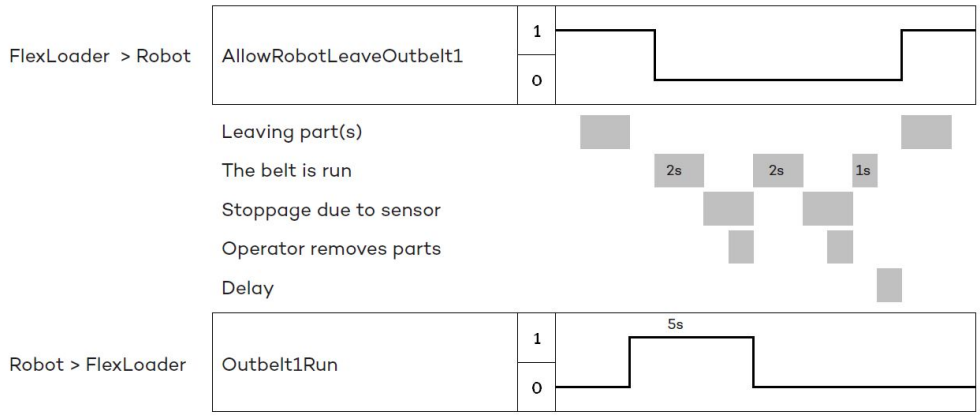
xx1800000184

Note that the robot can also set the signal Belt1AllowManualOperation without the request, that Belt1AllowManualOperation can continue to be set to 1 even if the operator selects automatic operation, and that the robot can set the signal to 0 without prior warning. This can happen for example when the robot is stopped and started.

Continues on next page

Outconveyor

During normal operation, signal exchange to robot occurs according to the following handshaking sequence:



xx1800000185

Machine tool interface

As standard, the FlexLoader SC 6000 is delivered with the following functional interface capability:

- ABB CI502 series I/O node connected to the robot PROFINET bus.
- Digital 24 V I/O interface (16 general inputs, 16 general outputs).
- The physical interface is constituted by terminal blocks in the control cabinet.
- The I/O node is physically located in the FlexLoader SC 6000 controller cabinet.

Optionally, customer specific functional interface and I/O solutions can be used.

The following signals are defined for general use.

Name	Description
diSpareInterface1	Spare input signal
...	
diSpareInterface16	Spare input signal
doSpareInterface1	Spare output signal
...	
doSpareInterface16	Spare output signal

6 Function description

6.1 Overview

WARNING

Appropriate training is required in order to use FlexLoader SC 6000. Incorrect use of the product can lead to personal injury and material damage.

Before commissioning the product, it is your responsibility to carefully read the chapter about safety and to be familiar with the safety devices, see *Safety on page 13*.

FlexLoader SC 6000 is a module based automation cell for handling parts in machine tending applications. The system can easily be adapted to the requirements of the customer.

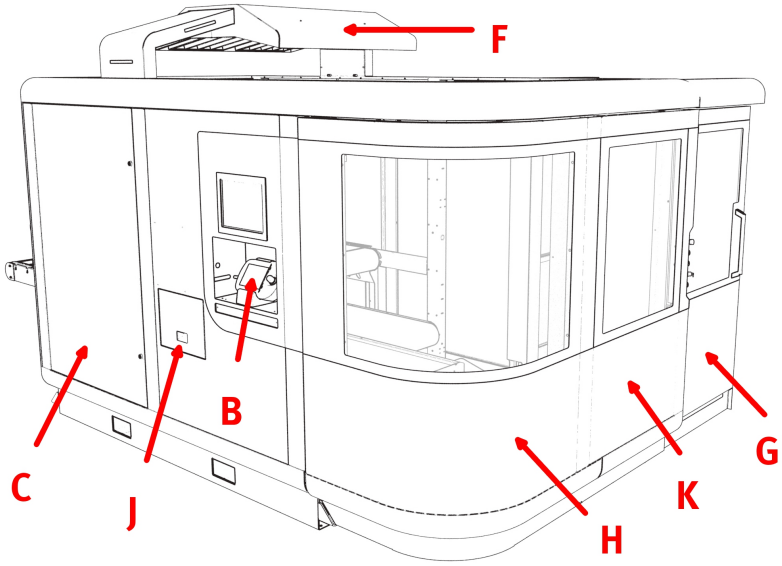
In a typical application FlexLoader SC 6000 is integrated with a machine tool. The operator places parts on a conveyor belt that acts as a buffer. The robot picks the parts from the conveyor belt. A vision system with camera that is located above the conveyor belt guides the robot to the correct grip position. After that all parts under the camera have been picked, the conveyor belt feeds new parts forward.

FlexLoader SC 6000 can be configured with a variety of useful options. Please refer to *Product specification - FlexLoader SC 6000* for further details.

Functional units

FlexLoader SC 6000 contains numerous mechanical devices and sensors that are important to know about in order to understand the function.

The FlexLoader SC 6000 consists of the following main parts:



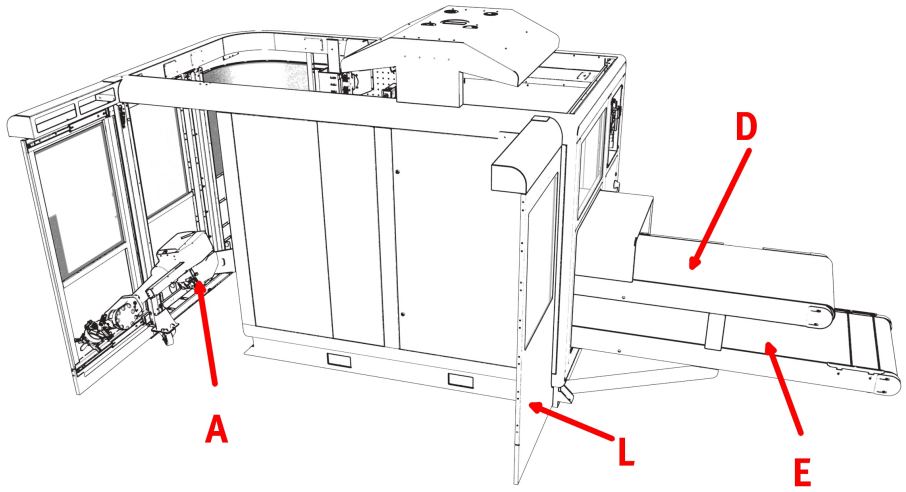
xx1900000374

Continues on next page

6.1 Overview

Continued

Pos.	Component	Pos.	Component
B	Operators panel	H	Swing door,
C	Electrical cabinet with IRC5 panel mounted controller	J	Statistical outlet
F	Vision system and illumination	K	Extension panel door
G	Retractable sliding door with entry control		

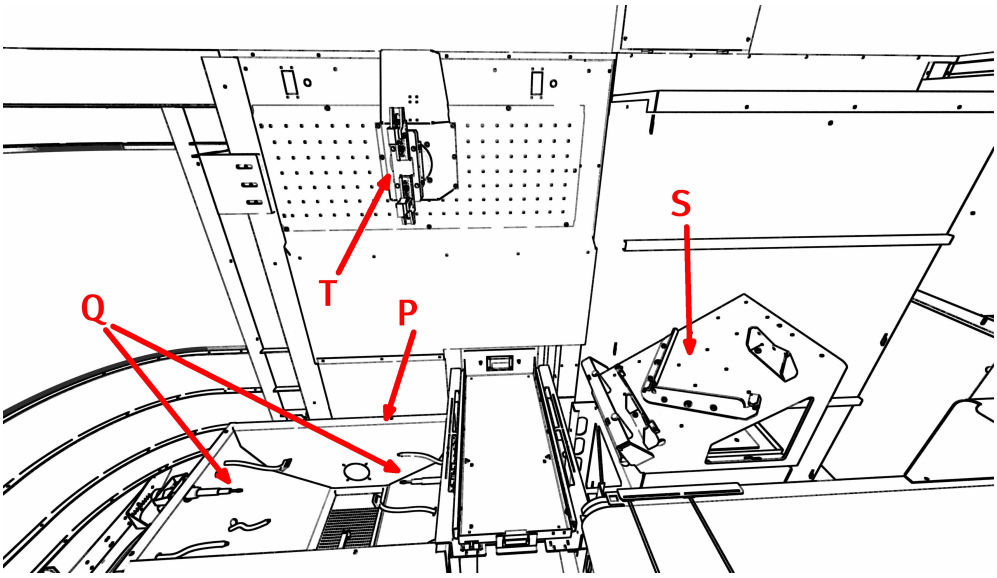


xx1900000375

Pos.	Component	Pos.	Component
A	Robot	E	Outconveyor
D	Inconveyor	K	Extension panel back

Continues on next page

The main options the FlexLoader SC 6000 can be equipped with are:



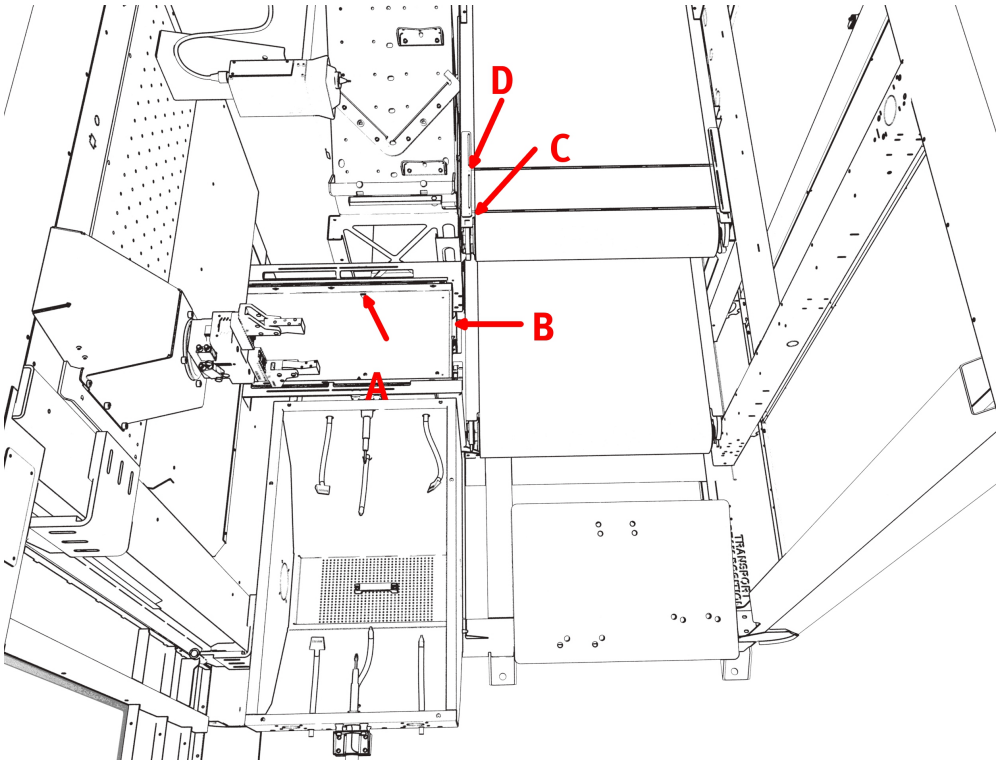
xx1900000376

Pos.	Component	Pos.	Component
M	Marking unit	S	Re-grip table
P	Air cleaning box	Q	Deburring Tools
T	Turn unit		

Continues on next page

Sensors

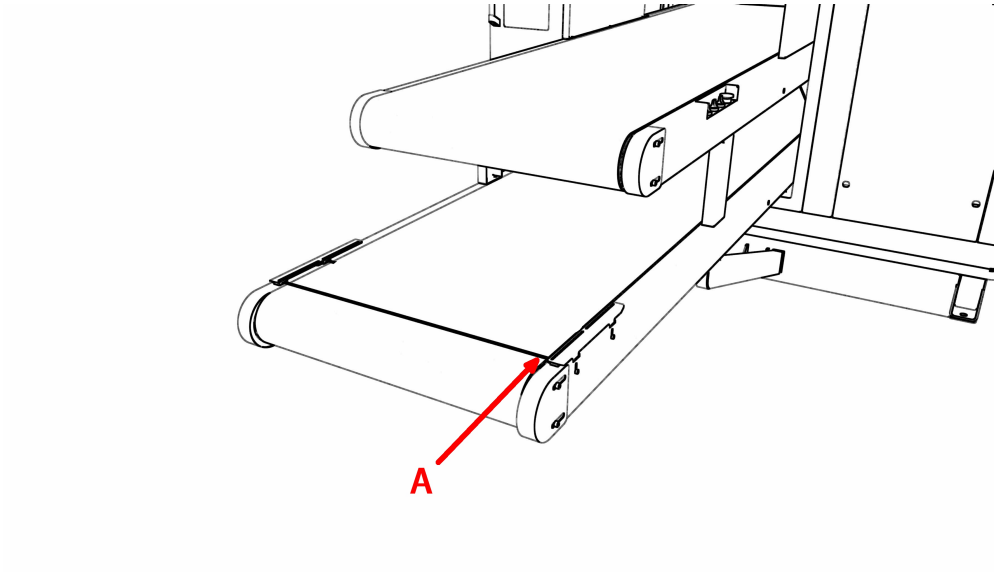
The main sensors the FlexLoader SC 6000 can be equipped with are:



xx1900000377

Pos.	Component	Pos.	Component
A	Sensor that detects if a part has been placed on the statistical outlet.	C	Inconveyor TooFar-sensor, detects if parts on the inconveyor have been transported to far. This is usually due to unidentified parts.
B	Safety sensor that ensures that robot only can move when the statistical outlet is closed.	D	Inconveyor InPosition-sensor, detects that a part has been transported into the cameras field of view.

Continues on next page



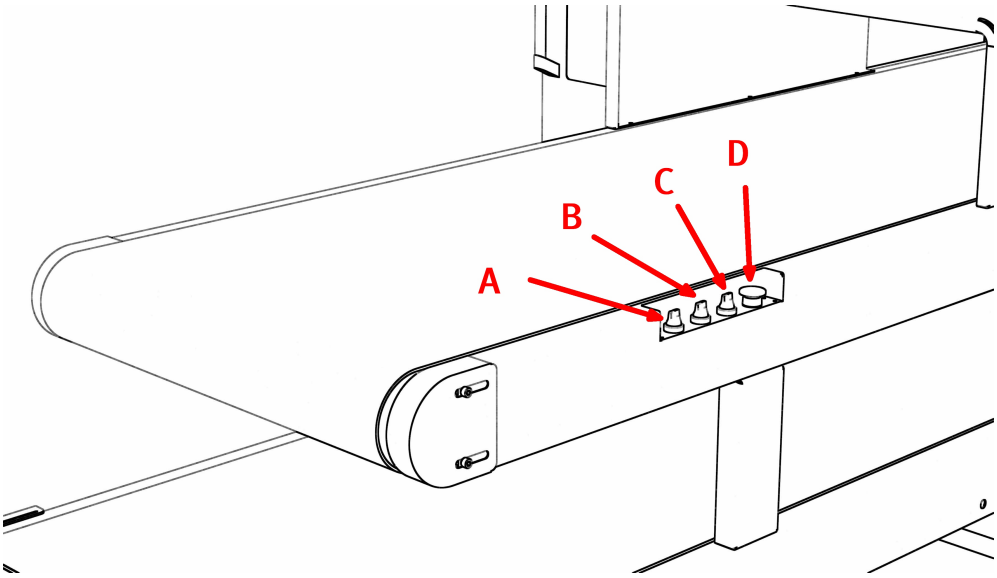
xx1900000378

Pos.	Component	Pos.	Component
A	Outconveyor ToFarOut-belt-sensor, detects when a part arrives at the end of the outconveyor.		

Operator panels

The main operator panels of FlexLoader SC 6000 are situated at the start of the inconveyor, below the vision system screen, and at the sliding door towards the machine tool.

The operating panel at the start of the inconveyor controls basic conveyor operation.

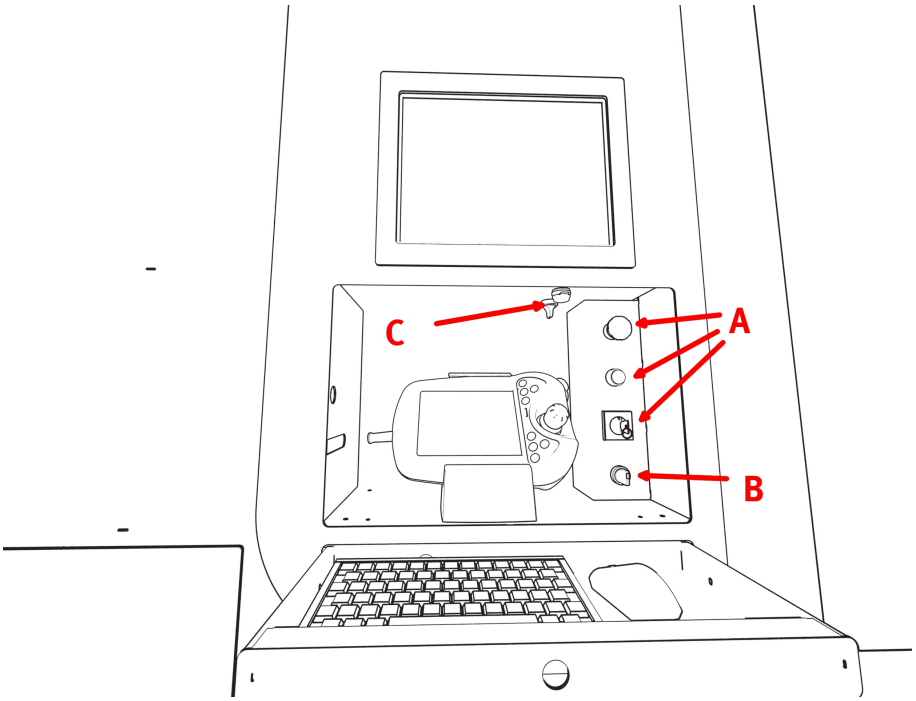


xx1900000379

Continues on next page

Pos.	Component	Pos.	Component
A	Move inconveyor forward and backward, reset errors by shortly switching the knob to move Forward.	C	Function selector for OutConveyor: Automatic mode or emptying mode.
B	Function selector inconveyor: Manual or Automatic, with visual indicator.	D	Emergency stop button.

The operator panel below the vision system screen contains the robot operator panel, i.e. FlexPendant, safety and motor controls, and a USB connection for data transfer to and from the vision system.

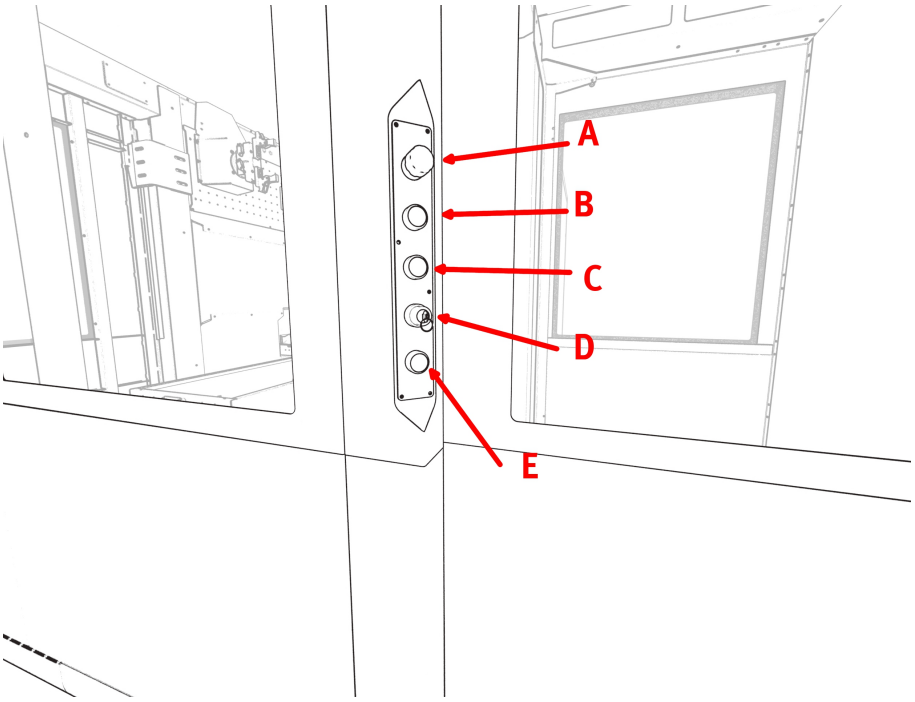


xx1900000380

Pos.	Component	Pos.	Component
A	Robot operator panel (Emergency stop, Motors On, Operating mode witch Auto/Manual with reduced speed)	C	USB connector, located below the vision system screen
B	Illumination switch, turns illumination on or off	D	Robot service port, located below the vision system screen

Continues on next page

The operator panel at the sliding door combines the following elements:



xx1900000381

Pos.	Component	Pos.	Component
A	Emergency stop	D	Door reset (after cell entry)
B	Start robot (if in home position)	E	Reset of emergency stop
C	Request cell entry and request confirmation lamp		

6.2 General function description

The purpose of FlexLoader SC 6000 is to feed parts to the cameras field of view where they are to be identified by a camera and picked by a robot. The inconveyor is filled with parts manually. Finished parts are left on the outconveyor. These are then picked manually by the operator.

The belt system of the FlexLoader SC 6000 is controlled by a number of background tasks (FlexLoader SC 6000 core functions).

Functions for inconveyor

- Two operation modes, either manual or automatic operation. Changing operation mode by operator and after confirmation from robot.
- Forward and backward operation using a switch.
- Sensor to stop the belt when the part reaches the cameras field of view.
- Sensor to stop the belt when the part reaches the end of the belt.
- Adjustable belt speed, changed at frequency inverter.
- Time monitored belt stop with manual reset.
- After powering up the FlexLoader SC 6000 and after emergency stop, the inconveyor is stationary.

Functions for outconveyor

- Two operation positions, either automatic operation or emptying. Changing operation mode via operator.
- Sensor to stop the belt when the part reaches the end of the belt.
- Time monitored belt stop with manual reset.
- In order for outconveyor in emptying mode to start running after powering up or emergency stop, a brief switch to automatic operation and back to emptying mode is required.

Functions for marking unit

This manual describes the basic operations controlled during automatic operation, control operation, and basic setup operation. In-depth information is given in the manual of the marking unit. This manual can be found in the documentation folder on the FlexLoader Vision PC.

During automatic operation, the robot controls the start and stop of marking. Communication is performed by means of standard serial channel communication.

Functions for other options

Functions for other options, e.g. air cleaning box, deburring, turn station and re-grip table, are controlled by various output signals directly from the robot.

7 RAPID program

7.1 Overview

WARNING

Applicable training is required in order to use the product. Incorrect use of the product can lead to personal injury and material damage.
Before programming the FlexLoader SC 6000, it is your responsibility to carefully read the chapter on safety, to become familiar with the FlexLoader SC 6000 and its options, and to become familiar with the safety devices.

This chapter is used for commissioning and adapting the code skeleton to the actual application. The information covers the basic structure of the robot program. More or less extensive additions and adjustments are required depending on the design and the functionality of the robot cell.

For more information on robot programming see the programming manual of the robot.

The FlexLoader RAPID code consists of several main blocks that supports the functionality of FlexLoader systems:

- FlexLoader application functionality
- FlexLoader Vision interface
- FlexLoader Vision Lite functionality
- FlexLoader RW Add-in
- FlexLoader assistance and utility functionality
- FlexLoader machine tool interface functionality
- FlexLoader options functionality

7.2 FlexLoader application functionality

Overview

The FlexLoader SC 6000 application code handles the complete workflow of parts in the cell (e.g. unloading and loading of parts, option handling, picking from inconveyor and leaving on outconveyor).

The application code consists of a basic executable skeleton, which must be modified by the integrator in order to match the actual application. However, the basic structure should be followed to ensure smooth operation.

Main structure

Main.pgf and MainModule.mod are automatically loaded into the controller when a part is started from FlexLoader Vision. During the following initialization, project modules with part specific information are loaded. This happens upon every start (if not configured otherwise, see FlexLoader Vision ConfigurationEditor).

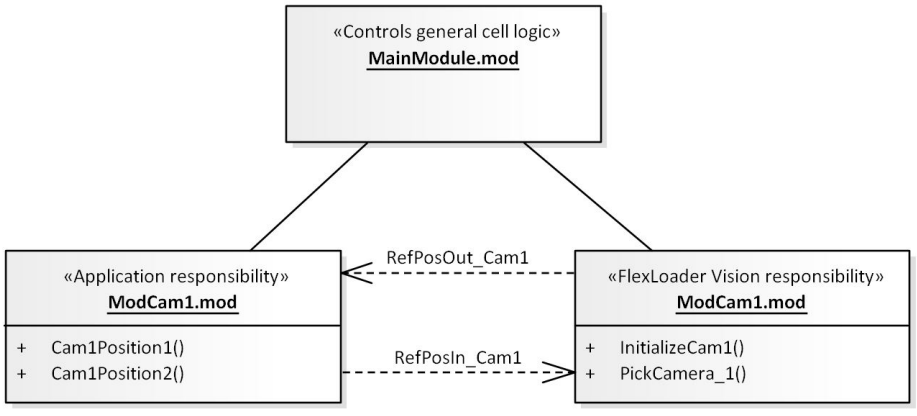
MainModule.mod is always the same for all parts and all operating modes of the robot cell. MainModule.mod is responsible for loading the modules that are linked to the parts to be run. Below, these are called PartCamX for a general reference to any PartCam module, or specifically PartCam1 for parts to be run using camera 1, PartCam2 for parts to be run using camera 2, and so on. These files can be renamed according to project needs.

Cell definitions like tool data, work objects and so on along with some event routines and help functions are placed in Common.sys and CommonCell / CommonSC3000 / CommonSC6000.

Application code vs FlexLoader Vision code

FlexLoader RAPID code has RAPID system code responsible for picking parts from the feeder. The interface between application code and system code for picking are so called reference positions.

RefPosIn is a robot position from which picking of parts is possible. After picking a part, the system code returns the robot to RefPosOut. The application part is responsible to handle parts, starting from RefPosOut, and handling all RAPID code until the robot can be returned to RefPosIn.



xx1800002289

Continues on next page

WARNING

After changing MainModule.mod or any PartCamX module, these modules MUST be saved, otherwise the changes will be lost upon next start due to the automatic program loading.

Example files

The following example programs show the principle structure of the robot code and how the robot program should communicate with FlexLoader Vision.

There is no guarantee that any of the examples work unless the program has been adjusted and tested for the specific robot cell.

RAPID example MainModule.mod

MainModule.mod contains the main routine which is the standard entry point. Some typical example application skeleton for MainModule.mod are shown below. All examples use the same initialization code.

```
PROC Main()
  ! set system specific control parameters
  InitializeMain;
  ! set vision specific control parameters
  InitVision;
  WaitTime 2;
  WHILE TRUE DO
    ! control end-of-cyle or entry requests
    CheckSystem;
    ! ---------------------------
    ! see example codes below
    ! ---------------------------
  WaitTime 0.05;
  ENDWHILE
ENDPROC
```

Generic solution

Many standard feeding solutions can be handled by the following schematic code. This is the most simple code example.

It can be used for picking from up to four feeders, e.g. FP 100, FP 300 and FP 400 or any FlexLoader function package.

```
! loop through desired cameras to perform picks and leave parts.
FOR i FROM nStartAcceptCamera TO nStopAcceptCamera DO
  ! This call will make the robot pick a part from camera i.
  ! When using desired position as in example, only parts found
  ! in position 1 is picked.
  PickPartAtCamera i,\DesiredPosition:=1;
  ! call part specific routine for the found position
  IF bDetailInGripper=TRUE THEN
    %"Cam"+ValToStr(i)+"Position_"+ValToStr(nPosition)%;
  ENDIF
ENDFOR
```

Continues on next page

For ease of use, the PickPartAtCamera routine combines some operations like grab image, wait for result, do retry if nothing found and evaluate which action vision proposed for the robot. The image grabbing could also be handled outside of the routine, either with autograb or with manual calls. Below is part of the code used in this routine.

Note the handling of flags for CHANGE_BIN, PICK_INTERLAYER and PICK_INTERLAYER_AND_CHANGE_BIN relevant for picking with 3D camera. Also note that there is another routine for handling picking several parts from same image. Read routine comments for PickMultiPartsAtCamera for information about this.

```
VisionResult{CameraNumber}:=GetNewVisionResult(CameraNumber
     \DesiredPosition?DesiredPosition,\MaxImageRetries?MaxImageRetries,
     \TimeoutTimeVision?TimeoutTimeVision);
TEST VisionResult{CameraNumber}.Action
CASE ABORTED:
  ! No valid coordinate from vision
  MT_SetAlarm "flv_ImageAborted",\WildCardTexts:=
     [VisionResult{CameraNumber}.AbortMsg];
CASE NO_DETAILS:
  ! Allow manual mode again if there is a feeder of that type
  IF nFeederType{CameraNumber}=FEEDER_NO_RETURN SetDOSignal
     "DOF_AllowManualModeFeeder"+ValToStr(CameraNumber),1;
  ! Nothing found in image, even after potential retries
  MT_SetAlarm "flv_NoPartFoundError";
  Stop;
  EXIT;
  ! TODO -------- ONLY FOR CELLS WITH NO BELT, EXAMPLE PALLET
     PICKER. REMOVE DOWN TO NEXT DASHED LINE IF NOT NEEDED!
     --------------
CASE CHANGE_BIN:
  ! No valid Pick position in pTarget
  ! TODO - Make sure pallet is switched
  Stop;
  ! Call ConfirmPick or RefPosOut to move out of camera area and
     confirm pick to vision.
  ! In cells with auto grab, this will also trigg a new image. Use
     RefPosOut_ to move out of camera area at the same time as
     confirming.
  CallByVar "ConfirmPick",CameraNumber;
CASE PICK_INTERLAYER:
  ! pTarget holds coordinates for an interlayer
  ! TODO - Pick interlayer with robot or ask operator to remove it
  Stop;
  ! Call ConfirmPick or RefPosOut to move out of camera area and
     confirm pick to vision.
  ! In cells with auto grab, this will also trig a new image. Use
     RefPosOut_ to move out of camera area at the same time as
     confirming.
  CallByVar "ConfirmPick",CameraNumber;
CASE PICK_INTERLAYER_AND_CHANGE_BIN:
  ! Action only in cells with no feeder
```

Continues on next page

```
    ! VisionResult{CAMERA_NO_1}.Target holds coordinates for an
       interlayer
    ! TODO - Pick interlayer with robot or ask operator to remove it
    ! TODO - Pallet is also empty, so make sure it's switched out
    Stop;
    ! Call ConfirmPick or RefPosOut to move out of camera area and
       confirm pick to vision.
    ! In cells with auto grab, this will also trig a new image. Use
       RefPosOut_ to move out of camera area at the same time as
       confirming.
    CallByVar "ConfirmPick",CameraNumber;
    ! ------------------------------------------------------
  DEFAULT:
    ! Move robot to camera area
    CallByVar "RefPosInCam_",CameraNumber;
    ! Call routine to pick the part
    CallByVar "PickCam_",CameraNumber;
    ! Move robot out of camera area and confirm pick
    CallByVar "RefPosOutCam_",CameraNumber;
  ENDTEST
```

Single camera feeder with post-pick adjustments

This example is valid for a feeder with a single camera, where the same camera is used for the normal pick and for post-pick adjustments. This can be used for more exact positioning of the part in the gripper fingers.

```
! Belt setting: setup how belt should be controlled
nBeltAction:=[RUN_ONE_DETAIL,RUN_NEVER,RUN_NEVER,RUN_NEVER];
! Camera setting: grab images automatically or only from user
     program
bAllowAutoGrab:=[TRUE,FALSE,FALSE,FALSE];
```

In order to take the post-pick image, use the prepared routines for this. Note that the RAPID modules for this are optional and might not be included in the RAPID package if not asked for. See comments in RAPID code for more information about the functions in the example code.

```
VAR visionres vrResult;
! Call to Control image function to prepare for second image, grab
     that image and then reset camera to normal image settings.
vrResult:=ControlImage(CAMERA_NO_1,\ControlImagePositions:=[3,4],
     \NormalImagePositions:=[1,2],\CameraSettingsControl:=[80,70,5],
     \CameraSettingsNormal:=[60,60,10]);
```

The ControlImage routine will depend on the user having created or modified a couple of movement routines to help robot get to correct position for the post-pick image. Routines in example below must exist and have movements according to the comments.

```
PROC MoveToControlImage1()
! TODO - if control image is used add code here to move robot in
     position for control image
ENDPROC
```

Continues on next page

```
PROC MoveFromControlImage1()
! TODO - if control image is used add code here to move robot away
     and out of camera field of view
ENDPROC
```

Single camera feeder with additional camera for post-pick adjustments

This example is valid for a feeder, e.g. a FP 800 with an additional camera for post-pick adjustments. All camera image acquisitions are controlled manually due to process requirements.

```
! Belt setting: setup how belt should be controlled
nBeltAction:=[RUN_NEVER,RUN_NEVER,RUN_NEVER,RUN_NEVER];
! Camera setting: grab images automatically or only from user
     program
bAllowAutoGrab:=[FALSE,FALSE,FALSE,FALSE];
```

The main loop is handled according to code shown below. This example shows an example of how to handle that parts occasionally are not found.

```
! do initial image acquisition and pick of part
GrabImage CAMERA_NO_1;
RefPosInCam_1
WaitUntil bCoordReceived{CAMERA_NO_1}=TRUE;
SetNextTarget CAMERA_NO_1;

! handle that parts occasionally are not found.
WHILE nAction{CAMERA_NO_1} <= NO_DETAIL DO
  ! change camera settings if necessary
  DiscardAndTakeNewImage CAMERA_NO_1;
  WaitUntil bCoordReceived{CAMERA_NO_1}=TRUE;
  SetNextTarget CAMERA_NO_1;
  Incr nImageRetries{CAMERA_NO_1};
  IF nImageRetries{CAMERA_NO_1} > MAX_IMAGE_RETRIES{CAMERA_NO_1}
     THEN
    ! Create alarm or inform user in some other way
  EXIT;
  ENDIF
ENDWHILE

IF nAction{nCameraNumber}=CHANGE_BIN THEN
  ! TODO - Do something to take care of pallet change, can be
     removed if not a pallet picker cell
ELSEIF nAction{nCameraNumber}=PICK_INTERLAYER THEN
  ! TODO - Do something to take care of interlayer, can be removed
     if not a pallet picker cell
ELSEIF nAction{nCameraNumber}=PICK_INTERLAYER_AND_CHANGE_BIN THEN
  ! TODO - Do something to take care of interlayer AND pallet
     change, can be removed if not a pallet picker cell
ELSE
  PickCam_1;
  RefPosOutCam_1;
```

Continues on next page

```
    ! start process of post-pick processing
    MoveRobotToCamera2;
    RefPosInCam_2;
    GrabImage CAMERA_NO_2;
    WaitUntil bCoordReceived{CAMERA_NO_2}=TRUE;
    SetNextTarget CAMERA_NO_2;

    ! handle that parts occasionally are not found.
    WHILE nAction{CAMERA_NO_2} <= NO_DETAIL DO
      ! change camera settings if necessary
      DiscardAndTakeNewImage CAMERA_NO_2;
      WaitUntil bCoordReceived{CAMERA_NO_2}=TRUE;
      SetNextTarget CAMERA_NO_2;
      Incr nImageRetries{CAMERA_NO_2};
      IF nImageRetries{CAMERA_NO_1} > MAX_IMAGE_RETRIES{CAMERA_NO_1}
        THEN
        ! Create alarm or inform user in some other way
        EXIT;
      ENDIF
    ENDWHILE

    ! handle parts according to results from vision data
    RefPosOutCam_2;
    %"Cam"+ValToStr(CAMERA_NO_1)+"Position_"+ValToStr(nPosition)%;
  ENDIF
```

Multiple robot with camera handling

If the robot controller has several attached robots which should interact with FlexLoader Vision there are some additional settings and variables that needs to be handled. First, FlexLoader Vision will load Main.pgf to T_ROB1 just as normal. If there is a second robot task and there is a MainRob2.pgf placed in robot HOME folder, then it is loaded to T_ROB2. In the same way MainRob3.pgf is loaded to T_ROB3 if it exists and MainRob4.pgf to T_ROB4 if that task exists.

Some communication between robot and FlexLoader Vision is general and should be communicated once for one robot controller. Most of the communication is done per camera and is communicated to the robot task that is in control of the specific camera. To handle all this there are a couple of variables that must be set correctly. Normally those are set in InitializeMain.

```
! Communication to FlexLoader Vision, the other robots are setup
     with TRUE for this value.
bIsSlaveVisionRobot:=FALSE;

! TODO - Set cameras to use in this robot task. Risk for reference
     errors if accepting cameras that is not there in reality.
nStartAcceptCamera:=1;
nStopAcceptCamera:=1;
```

In the example above, this robot task has set TASK PERS bIsSlaveVisionRobot to FALSE. This means that all general communication that is needed is handled by

Continues on next page

this task. It's not so important which robot task is Master as long as it is only one task. The other robot tasks set this value to TRUE to make them slaves.

The other variables define which camera or cameras this robot should handle. Note that it must be in order, robot 1 could handle camera 1-2 but not camera 1 and 3. Be careful when deciding camera numbers!

Normally there is one PartCamX.mod module for each camera. This module is selected during teach-in in the mechanical tab of FlexLoader Vision and later on loaded into the robot task that controls that camera. See example below.

```
! Code in InitializeMain in robot task T_ROB2
bIsSlaveVisionRobot:=TRUE;
! Assign camera 3-4 to robot 2.
nStartAcceptCamera:=3;
nStopAcceptCamera:=4;

! Code in main program of T_ROB2
GrabImage CAMERA_NO_3;
! ...normal handling to of receiving coordinates
! Then call to pick part, code in the PartCam3.mod in T_ROB2.
PickCam_3;
```

At start up FlexLoader Vision and RAPID code will co-work to make the PartCam3.mod and PartCam4.mod load into T_ROB2 task according to the variable setting.

Single camera feeder with multiple coordinate transfer for multiple picking

In certain applications, the robot is expected to pick more than one part in one pick cycle. This can reduce cycle times.

FlexLoader Vision needs to be configured for multiple coordinate transfer (see FlexLoader Vision ConfigurationEditor).

The main code uses following instructions to handle this case.

```
VAR num nPartsToPick;
! This call will make the robot pick 3 parts from the same image,
     first and second pick must be a part found in position 1 and
     third pick must be found in position 2.
! If this combination is not found vision system will try a new
     image twice and then give up.
nPartsToPick:=PickMultiPartsAtCamera(CAMERA_NO_1,
     \NumberOfPartsToPick:=3,
     \DesiredPositions:=[1,1,2],\MaxImageRetries:=2);
```

In the part module file there must be a special pick routine named as in example below. It should handle how to pick the different parts.

```
PROC PickMultiPartsCam_1(num PartNo,num Position)
  VAR dnum dnGripper;
  ! Switch off configuration control to be sure to reach the
     position, those will switch on in MoveFromCamera1.
  ConfJ\Off;
  ConfL\Off;
  ! TODO - set correct gripper to use depending on which part to
     pick
```

Continues on next page

```
    TEST PartNo
    CASE 1:
      ! TODO - Set correct gripper to grab first part from vision
       result
      tPickingGripper:=tGripper1;
      dnGripper:=GRIPPER_1;
    CASE 2:
      ! TODO - Set correct gripper to grab second part from vision
       result
      tPickingGripper:=tGripper2;
      dnGripper:=GRIPPER_2;
    CASE 3:
      ! TODO - Set correct gripper to grab third part from vision
       result
      tPickingGripper:=tGripper3;
      dnGripper:=GRIPPER_3;
    ENDTEST
    ! Update grip load based on currently used gripper
    GripLoadUpdate dnGripper;
    ! Set correct data in part tracker for feeder in
    MT_SetPartInfo\Station:=sdFeederIn,\ProgCode:=dnProgCodePartCam1,
       \State:=psRAW,\UserDef:="PickNo: " +ValToStr(PartNo)+" Pos:
       "+ ValToStr(Position);
    MoveJ RelTool(VisionResult{CAMERA_NO_1}.Target,0,0,-nPZ),v1000,
       z10,tPickingGripper\WObj:=wCamera1;
    MoveL VisionResult{CAMERA_NO_1}.Target,v200,fine,
       tPickingGripper\WObj:=wCamera1;
    ! TODO - Close gripper to pick part
    ! TODO - If needed use your own solution to keep track of what's
       held in gripper
    PickPartAt sdFeederIn,dnGripper;
    MT_Log MT_LVL_INFO,"PickMultiPartsCam_1",["Picked part no: ",
       ValToStr(PartNo)," pos: ",ValToStr(Position)];
    ! TODO - If several parts is picked, the VisionResult will be
       overwritten between each one.
    ! If the result for each part is needed later, store it somewhere!
    MoveL Offs(VisionResult{CAMERA_NO_1}.Target,0,0,nPZ),v1000,z10,
       tPickingGripper\WObj:=wCamera1;
  ENDPROC
```

RAPID example PartCam1.mod

The provided detail specific application code offers various possibilities.

Generic teachin – Everything is part specific

Application code for a general teachin.

Robot tasks as picking, machine tending, or leaving on the outbelt, have to be programmed manually. A skeleton is provided, at location HOME:\PartModules\PartCam1.mod.

However, no general robot program can be provided for unknown parts, so the skeleton has to be adapted to each new part. Positions, movements and option

Continues on next page

handling have to be modified or generated for each new part. This is the most general and open approach to machine tending.

> **Note**
>
> These routines are provided as example code. Always ensure that they are working with the actual parts and processes.

Programming of e.g. Cam1Position_1, Cam1Position_2, Cam2Position_1... for all used cameras and positions must be done if those routines should be used.

RAPID example for Initialization

General initialization that is not part specific is placed in the InitializeMain in the MainModule. This includes the following variables which will affect the functionality of how and when images are grabbed.

```
! Belt setting: setup how belt should be controlled
nBeltAction{CAMERA_NO_1}:=RUN_ONE_DETAIL;

! Camera setting: grab images automatically or only from user
     program
bAllowAutoGrab{CAMERA_NO_1}:=TRUE;

! Set image grab delay (after stop at sensor)
nImageGrabDelay{CAMERA_NO_1}:=0.2;
```

Initialization that is part specific is placed in the PartCam1 module (if part is run at camera 1) in the InitializeCam1. This could include part dimensions, weight and which options should be used for the part.

```
! Options - Set value TRUE for the options that should be used for
     this part
bUseAirClean:=FALSE;
bUseMarking:=FALSE;
bUseWash:=FALSE;
bUseSamplePart:=FALSE;
bUseDeburr:=FALSE;
bUseReGrip:=FALSE;
bUseTurnStation:=FALSE;

! Initiate options (those which need initialization)
IF (bHasMarking=TRUE AND bUseMarking=TRUE) SetMarker
     sMarkingFontSize,sMarkingText;

! ------------------------------------------
! Weight of part [kg]
! ------------\ /----------------------------
nWeight_PART:=1;

! ------------------------------------------
! Distance [mm] from TCP to center of gravity in Z direction of
     tool for a part
! -----------------\ /-----------------------
nCOGFromTCP_PART:=30;
```

Continues on next page

RAPID example MT_MoveRobotTo

The principal motion in the FlexLoader SC 6000 is controlled by the MT_MoveRobotTo routine, which moves the robot from known zone positions to a target zone position. From these zone positions (via positions), movement within certain areas of the FlexLoader SC 6000 can be initiated.

For detailed reference on zone and via positions, see *MoveRobotVia.mod on page 206*

```
! order robot to another zone
MT_MoveRobotTo ZONE_OUTBELT;
```

CAUTION

The MT_MoveRobotTo routine is defined and verified for standard parts (as defined by the product specification) and standard 3-finger grippers in the standard FlexLoader SC 6000 configuration.

When zones, via positions, parts, grippers or any other conditions are changed, the integrator has to ensure that the MT_MoveRobotTo routine can execute without risk for collisions. The MT_MoveRobotTo routine is by default general. But due to variations it might need to be made part specific which is possible to handle within a project.

When starting the robot from unknown positions, collisions can occur.

7.3 FlexLoader Vision interface

Overview

The FlexLoader Vision interface code handles the interface and the communication to and from FlexLoader Vision. No changes should be made to this code.

The basic application related use of the FlexLoader Vision interface is described above, see *FlexLoader application functionality on page 92*. This section describes additional functionality that can be used for a large variety of purposes.

For further reference, see *FlexLoader Vision interface on page 194*.

RAPID examples using Vision.sys

Camera exposure time, contrast and gain can be controlled from the robot.

```
! set contrast for camera 1 to 50%
CameraContrast CAMERA_NO_1, 50;

! set exposure time for camera 2 to 10%
CameraExposureTime CAMERA_NO_2, 10;

! set gain for camera 3 to 30%
CameraContrast CAMERA_NO_3, 30;
```

Positions for the currently used part can be switched on and off, if required by the application.

```
! enable all positions for camera 1
EnablePosition CAMERA_NO_1, 1000;
! disable position 1 for camera 1
DisablePosition CAMERA_NO_1, 1;

! disable all positions for camera 1
DisablePosition CAMERA_NO_1, 1000;
! enable position 1 for camera 1
EnablePosition CAMERA_NO_1, 1;
```

Set alarm, warning and information in FlexLoader Vision alarm list.

```
! Set an alarm
SetAlarm "This is an alarm text";
! Set a warning
SetWarning "This is a warning text";
!Set an information message
SetInformation "This is an information";

!Reset the alarm
ResetAlarm "This is an alarm text";
! Reset the warning
ResetWarning "This is a warning text";
!Reset the information message
ResetInformation "This is an information";
```

Continues on next page

There is a lot of more functions, look into the Vision.sys module and the routine comments for more information.

Limitations

Note

Please note the following system limitation regarding robot system events.

It is not possible to use elements of the FlexLoader Vision interface in system event routines (e.g. STOP, QUICKSTOP, START).

Do not use of e.g. master/slave functionality, alarms handling, initiating image acquisition or setting other parameters in such system event RAPID code. Non-compliance may result in undesired system behavior.

Note

Please note that some code relies on predefined I/O signal names. If any of those signal names is renamed the code can't execute as desired. If a signal name should change always search the code and make sure it is possible first.

7.4 FlexLoader Vision Lite functionality

Overview

The application code for FlexLoader Vision Lite provides an extension to the general teachin described previously.

This extension is used for parameter-based simplified teachin for upright standing cylindrical details, the so called FlexLoader Vision Lite. Normally, this code needs no changes.

For further reference, see *FlexLoader Vision Lite functionality on page 209*

The RAPID code follows the same structure as described in *FlexLoader application functionality on page 92*. It contains a preconfigured MainModule.mod, preconfigured machine tool sequences, and a parameter interface to FlexLoader Vision.

RAPID example LitePartCam1.mod

A specialized LitePartCam1.mod contains the application code for parameter-based simplified teachin for upright standing cylindrical parts (i.e. FlexLoader Vision Lite). A generic robot program based on geometrical part data is executed. No robot programming is necessary for different parts, and teachin is reduced to entering a few geometrical parameters in FlexLoader Vision. But note that some work objects and configurations needs to be handled.

The module contains parametric movement instructions, option handling and machine tool handling.

In this case, the LitePartCam1.mod is used for all parts, and only parametrized code can be executed. It is located in HOME\FlexLoaderVisionLite\.

Note

The handling routines are called for each part. Make sure your code works with all parts. If not, activate the special robot program option in FlexLoader Vision.

Special robot program option for part specific tasks

A part specific PartCam1.mod can be used and stored at a different location. Option handling, e.g. deburring, air cleaning or marking, can be handled on part level.

RAPID example for Initialization

Belt handling and grab setting is done in InitializeMain. Note that this is an example, other settings for those are also possible.

```
! Belt setting: setup how belt should be controlled
nBeltAction{CAMERA_NO_1}:=RUN_ONE_DETAIL;

! Camera setting: grab images automatically or only from user
     program
bAllowAutoGrab{CAMERA_NO_1}:=TRUE;

! Set image grab delay (after stop at sensor)
nImageGrabDelay{CAMERA_NO_1}:=0.2;
```

Continues on next page

Other part specific initialization could include option activation, outbelt setup, part data and load and unload positions. Those type of initialization is placed in InitializeCam1.

```
! Activate/deactivate options
bUseRegrip:=FALSE;
bUseTurnStation:=FALSE;

! Initiate options (those which need initialization)
IF (bHasMarking=TRUE AND bUseMarking=TRUE) SetMarker
     sMarkingFontSize,sMarkingText;

! Set up outbelt, see manual for documentation about all arguments.
SetUpFeederOutBelt FEEDER_WIDTH,nFeederOutSpeed,SPACE_BETWEEN_PARTS,
     SPACE_TO_EDGE,Y_DISTANCE_TO_LEAVE;
SetUpFeederOutDetail maxNumValue(nFinishedDetailDiameter,
     nFinishedDetailOuterDiameter),maxNumValue(nFinishedDetailDiameter,
     nFinishedDetailOuterDiameter),nGripHeightFinishedDetail,
     \allowedDetailsInHeight:=getAllowedNumberToStack(),\detailHeight:=
     nFinishedDetailHeight,\hasLeftOverPart:=bIsThereLeftOverPart,
     \leftOverPartWidth:=nLeftOverPartDiameter,\leftOverPartLength:=
     nLeftOverPartDiameter,\leftOverPartHeightToTCP:=nGripHeightLeftOverPart;

! Setup possible robot part loads for this detail program
lPartLite:=[nRawDetailWeight,[0,0,nGripHeightRawDetail-
     (nRawDetailHeight/2)],[1,0,0,0],0,0,0];

! Calculations for positions
pLoadMachine1_MAIN.trans:=calcPosLoadMachine();
pLoadMachine1_SUB.trans:=calcPosLoadMachine(\nSpindleOffset:=
     nUnloadSubSpindleOffset);
pUnLoadMachine1_MAIN.trans:=calcPosUnloadMachine();
pUnLoadMachine1_SUB.trans:=calcPosUnloadMachine(\nSpindleOffset:=
     nUnloadSubSpindleOffset);
pUnLoadLeftOverPart_MAIN.trans:=calcPosUnloadLeftOverPart();
pUnLoadLeftOverPart_SUB.trans:=calcPosUnloadLeftOverPart
     (\nSpindleOffset:=nUnloadSubSpindleOffset);
```

Cell emptying

In some cases of machine tending, the cell must be emptied from parts, i.e. the machine shall be unloaded without loading the next part.

For FlexLoader Vision Lite, this behavior is pre-programmed. By setting the output DOF_EmptyCell to “1”, the robot will finish a cycle by unloading a finished part without loading the next raw part.

The mechanism for changing the signal state (external signal, button, FlexPendant, ...) must be decided and implemented by the integrator.

7.5 FlexLoader Library Add-in

Overview

The FlexLoader Library Add-in package includes code that is installed into the RobotWare and hidden to end user. The following sections describe the different FlexLoader Library Add-in code blocks in short. For more information of all included instructions, functions and data types, see appendix.

Alarms & messages

Alarms & Messages helps user to create alarms, warnings, informations and other messages shown to the user in the local language. At robot start up, all texts will be read into the system from the source files holding texts of the currently used language in the Flexpendant.

The source files for English are found under HOME\Language\en. If the FlexPendant language is changed to Swedish, the files are expected to be in the folder HOME\Language\sv. If the standard text files don't exist for the selected language, a translation must be done and new files must be created.

Possible languages are the same as the languages that can be used for the FlexPendant itself. The table below shows selectable languages and return values that will be used in the robot code for each language. These return values should also be the name on the language folder to allow RAPID code to use that language.

Return value	Language
cs	Czech
zh	Chinese (simplified Chinese, mainland Chinese)
da	Danish
nl	Dutch
en	English
fi	Finnish
fr	French
de	German
hu	Hungarian
it	Italian
ja	Japanese
ko	Korean
pl	Polish
pt	Portuguese (Brazilian Portuguese)
ro	Romanian
ru	Russian
sl	Slovenian
es	Spanish
sv	Swedish
tr	Turkish

xx1900001037

Continues on next page

To create a new text in the cell, add a row to an existing file or create a new csv file. There is also a possibility to add project specific alarms directly from code using special RAPID instructions(see *FlexLoader Vision interface on page 102*). All csv files in the selected language folder will be read into the robot. Inside the csv file there are 7 columns to be filled to define the alarm or message.

Element	Description
Identifier	Unique string through the system. This is later used in routine calls to point to this message.
Category	“Error” = this is an alarm, “Warning” = this is a warning, “Information” = this is an information. Leave this blank for other messages used in input boxes and so on. If defined as any of the three this message will show in event log and alarm lists.
Station	If wanted, name the station to which this message is connected. Leave blank if not needed.
Number	If wanted, give an alarm number. This will be the last part of the alarm number showing in alarm lists.
Domain	If wanted, group texts and alarms in domains.
Header	Header for the message. Depending on where you see this message, the header might be the only line you see.
Text	Body text for the message. Possible to give a more detailed description of the message.

In the Header and Text wild cards can be used. A wild card is a place holder that later can be changed to wanted text through RAPID code. For example Header: “Emergency stop {1} machine number {2}”.

The placeholder numbers can be replaced by customer defined text when RAPID calls are made in the program. If the example has the identifier "EMStop" then a call in the program might look like this:

```
! Creating alarm: Emergency stop from machine number 3
MT_SetAlarm "EMStop",\WildCardTexts:=["from",ValToStr(3)];
```

The text shown in alarm list will be “Emergency stop from machine number 3”.

A maximum number of 5 wild cards per message is possible, which can be used freely in both Header and Text.

Example of tasks that are handled by the alarms & messages are setting and resetting of alarms, keeping track of active alarms and language handled UI instructions.

Predefined data

There is some predefined data in the alarms & messages module that could be good to know about.

```
MT_nMsgReadFromCSV
```

This variable will be set to 1 during the time that csv file information is read and stored into the system. This could take some time and the alarms & messages system will not work as intended during this time. If needed, make sure this variable is set to 0 before starting normal execution and messaging usage in your user program.

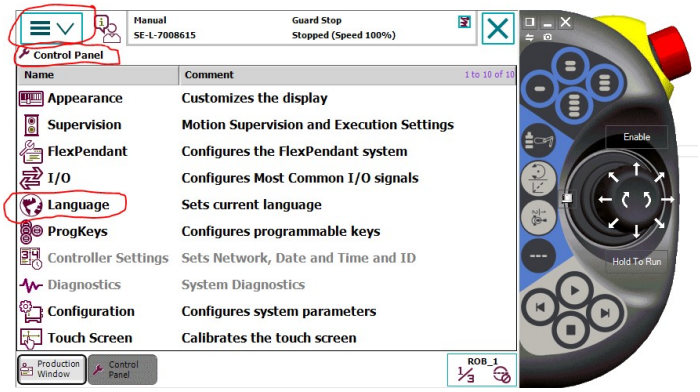
```
MT_bUIDialogActive
```

Continues on next page

This variable will be TRUE as long as there is any type of user question/input box active on the FlexPendant (any active UI instruction). This means that as long as this is TRUE robot is waiting for some kind of user input through the Flexpendant. This could be used to indicate for the user that input is needed.

Language selection

To select a new language select the main menu in top left corner of the FlexPendant screen. Then select **Control Panel** and then **Language**, see image below.



xx1900001039

Find your desired language in the list and press **Ok**. The change will become active after a restart and you will be asked to restart directly when you choose a new language.

When system is restarted the FlexPendant language has changed. The new text file is also read upon restart and your new application text will also be updated now.

Note that if the correct language file for the application texts was not found the system will search for and use the English file instead. There will be a log in the FlexPendant of that in the case this happens.

Stations

Stations enable a good object-oriented structure of the cell. They open possibilities to visualize station level information in a user-friendly way on a station level in the user interface.

Most modules or objects in a cell can be defined as a station. The station concept is especially useful for objects that perform a certain task or function or could hold a part. Typical examples are machine tools, buffer stations, in- and out-feeders and testing units.

Continues on next page

The part tracker concept (described below) depends on stations. A station can include up to 20 part trackers. Everywhere the robot can store parts or does work with a part could be a good candidate for a station.

Tip

If a buffer could hold more than 20 parts it is probably a good idea to create two different stations to be able to create part trackers for all the positions at the buffer.

Part trackers

Part trackers are used to keep track of where parts are located in the cell. Part trackers can also carry some specific data for each part, for example a part state like raw, machined, or deburred. It's also possible to assign user data for a part that could hold an index number or some other kind of data specific for to part.

To handle this, the parts need to be coming into the cell somewhere and the code needs to create a new part tracker object which is connected to this in-feeder station. One tracker object can hold one part and a tracker is always connected to a station which could have up to 20 part trackers.

When the robot moves parts in the cell, instructions should be used to move the part data between the trackers to have it match the cell status. Then the user program can check which part is in the machine part tracker or the robot gripper part tracker for example.

For more detailed information about the instructions, see appendix.

MoveVia

The MoveVia functionality helps to move the robot between different stations in the cell. Normally, in every cell there is one robot position in front of each station where robot is doing some kind of work.

Those points in front of the different stations in the cell are used when navigating between the different stations and are defined by the user and packaged into a mttargetdata. This data is defined in the user program and gives the MoveVia more input to find which station is the closest when the robot is started in an undefined position.

In the user main program the normal instruction calls will be to MT_MoveRobotTo to which the mttargetdata of the target position is passed, no matter where the robot is positioned right know.

Find the MoveRobotVia example module to get a good picture of how to use this functionality. One key is to define all mvXXXXX routines where XXXXX is the name of each defined mttargetdata. These mv-instructions should move the robot to that corresponding target or zone.

Example

A small project might define these zones: HOME, MACHINE and FEEDER. Within the MoveVia concept you define in the user program a position and a mttargetdata:

```
PERS robtarget pHome:=[[400,0,600], .....
PERS robtarget pViaMachine:=[[-1500,300,1000], .....
```

Continues on next page

```
PERS robtarget pViaFeeder:=[[1500,300,1000], .....

CONST mttargetdata ZONE_HOME:=[1,"pHome",[-80,70], .....
CONST mttargetdata ZONE_MACHINE:=[2,"pViaMachine",[-100,-80], .....
CONST mttargetdata ZONE_FEEDER:=[5,"pViaFeeder",[-100,170], .....
```

Note that in each mttargetdata, there is an indirect pointer to the actual robtarget defined by a string value. Furthermore, a couple of routines should be created:

```
PROC MT_MoveToVia(mttargetdata Target)
  ! In this routine add code to move to via position (position
     connected to the targetdata) of the zone. This will be called
     if robot is somewhere in the zone but not at a defined
     position and needs to come back. The easiest way could be to
     always go straight to that position:
  MoveJ MT_GetViaPosition(target),v500,fine,tTool0;

  ! But in some cases that might not be possible and there it could
     be needed to add some additional smart code to navigate to a
     free area before moving directly to the via position. This
     added code must be created by the integrator who have
     knowledge of the specific cell.
ENDPROC
```

In the mv-instructions define how to navigate from any other position to this position/zone. Note that the routine name must be mv followed by the exact name of the corresponding mttargetdata. mvZONE_HOME could in this example look like this. Note that there also must be similar routines for mvZONE_MACHINE and mvZONE_FEEDER too.

```
PROC mvZONE_HOME()
  VAR mttargetdata FromZone;

  FromZone:=MT_GetCurrentZone();
  TEST FromZone
  CASE ZONE_MACHINE:
    !TODO - add movement instructions here
  CASE ZONE_FEEDER:
    !TODO - add movement instructions here
  DEFAULT:
    !No path available
    !TODO - error handling in a way that fits your project
    TPWrite "Movement to ZONE_FEEDLINE from this position is not
     defined";
    Stop;
    EXIT;
  ENDTEST
  MoveJ pHome,v7000,z200,tTool0;
ENDPROC
```

Continues on next page

Predefined data

The MT_SAFE_START_DISTANCE is default 100 and is used internally as the safe start distance. This means that the robot is free to start without any extra user warning or anything if it is within this distance to a via position.

```
CONST num MT_SAFE_START_DISTANCE:=100;
```

For more detailed information about the instructions, see appendix.

Logging

Sometimes it's a good idea to create some logs to keep track of what is happening in the robot code. This module gives the possibility to keep some order by having different log levels that can be enabled and disabled independently.

The log messages are written to a csv file on a USB stick that has to be inserted to the robot controller computer before starting to log. If no USB memory is inserted, user will be notified of this.

Note

The USB port in the FlexPendant is not usable for this logging purpose.

Use the USB ports on the main computer unit instead

The log message created with a call to MT_Log will include date and time, robot task, log level, source and message. Each log will occupy one row in the log file. When the log file is getting big, the content is moved to an old log file and the current file is removed. This is to make sure the latest log history is always there even if the USB is full.

For more detailed information about the instructions, see appendix.

7.6 FlexLoader assistance and utility functionality

Overview

Assistance and utility functionality enables efficient and user friendly programming of the robot cell. It provides means for tool handling, feeder handling, entry control, indication lights and several global utility functions. Some of these function packages like options and feeder handling will be included in the RAPID code package if those kind of equipment exist in the cell.

For further reference, see *FlexLoader assistance and utility functionality on page 211*.

RAPID example tool handling

The following example is typical use of the tool handling system.

```
! Call to define all robot tools in cell. This call is normally
     done in the PowerOn event.
SetupTools;

! Example of how SetupTool could look like. This is placed in
     CommonXXXX.sys
PROC SetupTools()
  MT_Log MT_LVL_INFO,"CommonCell",["Performing Tool setup"];
  ! General tool data setup
  tGripper0:=tool0;
  !----- Define gripper data -----------
  !!TODO - set correct values for you grippers tool data, then
     remove alarm row below
  !tGripper1:=
  !tGripper2:=
  MT_SetAlarm "tool_NoToolData";
  !-------------------------------------
  !Modify cog and mass for tGripper0, default to use data for
     tGripper1
  !!TODO - If necessary modify those cog and mass data to something
     that fits even better for tGripper0
  tGripper0.tload.cog:=tGripper1.tload.cog;
  tGripper0.tload.mass:=tGripper1.tload.mass;
  !!TODO - modify gripper setups to fit the current cell. Read
     comments in ToolSetup for more information about this.
  ClearToolData;
  ToolSetup GRIPPER_0,"Gripper 0",tGripper0,sdRobot,\StnIndex:=10;
  ToolSetup GRIPPER_1,MT_GetText("msgGripper1"),tGripper1,sdRobot,
     \StnIndex:=1,\OpenOrOnSignal:="DOF_OpenGripper1",
     \WaitTimeOpenOrOn:=0.5,\LabelOpenOrOn:=MT_GetText("msgOpened"),
     \WaitTimeCloseOrOff:=0.5,\LabelCloseOrOff:=MT_GetText("msgClosed");
  ToolSetup GRIPPER_2,MT_GetText("msgGripper2"),tGripper2,sdRobot,
     \StnIndex:=2,\OpenOrOnSignal:="DOF_OpenGripper2",\WaitTimeOpenOrOn:=0.5,
     \LabelOpenOrOn:=MT_GetText("msgOpened"),\WaitTimeCloseOrOff:=0.5,
     \LabelCloseOrOff:=MT_GetText("msgClosed");
  ToolSetup COMPRESSED_AIR_1,MT_GetText("msgCompressedAir1"),
     tGripper0,sdRobot,\StnIndex:=3,\OpenOrOnSignal:="doValve5Gripper",
     \WaitTimeOpenOrOn:=0.5,\LabelOpenOrOn:=MT_GetText("msgOn"),
     \WaitTimeCloseOrOff:=0.5,\LabelCloseOrOff:=MT_GetText("msgOff");
```

Continues on next page

```
ENDPROC

!Sets correct gripload depending on all currently held parts (in
     all grippers). This should normally be called every time the
     robot is about to start working with another gripper (ie.
     gripper 1, gripper 2, tool0 and so on)
GripLoadUpdate GRIPPER_1;

!Makes sure gripper is open before going in to pick
ToolAction\Release,GRIPPER_1,\InsideGrip:=bInsideGrip;

!Pick part by using standard function that handles both picking
     and moving part tracker information in one
!Picks the part with gripper 1 from feeder in (includes moving of
     tracker data from feeder in to robot gripper). Uses inside
     grip if boolean value is set.
PickPartAt sdFeederIn,GRIPPER_1,\InsideGrip:=bInsideGrip;

!How to check what is in a gripper and act on that information
IF getToolPartState(GRIPPER_1)=psRAW THEN
  !use regrip if needed
  IF (bHasRegrip=TRUE AND bUseRegrip=TRUE) RegripDetail;
  !Load machine
  LoadMachine;
ENDIF
```

All tools that are defined through the tool handling system can be access through the tool control menu. This menu is used in manual mode to open, close, turn on or off grippers for example if a part should be released.

RAPID example feeder handling

Feeder handling includes in-feeder control in cases no PLC is taking care of it and some help functions to handle leaving and running the out belt. It also includes additional functions to monitor if camera belt has been in manual mode since this might indicate someone entered the cell and might have touched parts which was already found and prepared for robot picking.

RAPID example entry control

Entry requests are handled in a background task. The main task uses the CheckSystem routine to test and react to entry and stop requests.

RAPID example indication lights

The indication light is controlled automatically through the user messaging system. The indication lamps are switched on and off depending on the current alarm, warning, question and information status. Indication light control is executed in a background task.

Conditions and indication light colors are defined during program initialization. A color is assigned to all conditions, and the integrator selects flashing or static lights. In most cases RGB lighting is used.

```
! predefined colors: RED, GREEN, BLUE, WHITE, YELLOW
```

Continues on next page

```
IndicationLightsSetupRGBLights \AlarmFlashingColor:=RED
     \CycleOnColor:=GREEN \InformationFlashingColor:=WHITE
     \QuestionFlashingColor:=BLUE \WarningFlashingColor:=YELLOW;

! static light instead
IndicationLightsSetupRGBLights \AlarmStaticColor=RED
     \CycleOnColor:=GREEN \InformationStaticColor=WHITE
     \QuestionStaticColor=BLUE \WarningStaticColor=YELLOW;
```

Sometimes a traditional light tower with discrete signals for every color is used. In this case, a different setup can be used:

```
! predefined lamps: BLUE_LAMP, GREEN_LAMP, ORANGE_LAMP,
! RED_LAMP, WHITE_LAMP, YELLOW_LAMP
IndicationLightsSetupLightTower \AlarmStaticColor:=RED
     \CycleOnColor:=GREEN \QuestionStaticColor:=BLUE
     \WarningStaticColor:=YELLOW;
```

If needed, the lighting can be switched on by integrator RAPID code by using dedicated signals.

```
! force alarm indication on
Set DOF_Alarm;
! switch back to automatic control
Reset DOF_Alarm;

! force warning indication on
Set DOF_Warning;
! switch back to automatic control
Reset DOF_Warning;

! force information indication on
Set DOF_Information;
! switch back to automatic control
Reset DOF_Information;

! force question indication on
Set DOF_Question;
! switch back to automatic control
Reset DOF_Question;
```

Light indications should always be accompanied with user messages for explanation.

RAPID example global utility functions

Look into the module GlobalCodeAndConfig and read routine header comments to see some of the utility routines that could be good to have. This module is loaded as shared which makes those routines accessible through all tasks.

7.7 FlexLoader machine tool interface functionality

Overview

The FlexLoader SC 6000 standard modules communicate with the machine tool by means of the interface module TemplateMachine_Lathe.sys or TemplateMachine_CNC.sys, which must be modified and adapted to the actual machine tool by the integrator. Those modules are template modules that describes how the machine communication should be handled.

Read the module and routine comments, in order to add machine tool specific code where needed.

For further reference, see *FlexLoader machine tool interface functionality on page 218*

States and actions

This module contains a number of predefined states and actions that are consistently used throughout the FlexLoader SC 6000.

For further reference, see *FlexLoader machine tool interface functionality on page 218*

RAPID example

Depending on the type of the machine tool and the type of application, a variety of communication flows are possible.

A typical machine tool communication flow for a load-unload sequence as realized in the standard setup could be as follows. Note that no movement is included in this schematic code.

```
! Prepare for machine tool interaction
Machine1Action INIT_MACHINE;
WaitUntil Machine1Check(LOAD_MAIN_OK)=TRUE;
Machine1Action OPEN_DOOR;
WaitUntil Machine1Wait(DOOR_OPENED,10);

! Enter machine tool and load part in main chuck
Machine1Action CLOSE_CHUCK_MAIN;
! grip detail in sub chuck
Machine1Action OPEN_CHUCK_SUB;

! leave machine
Machine1Action CLOSE_DOOR;
Machine1Action CYCLE_START;
```

Different machine tools have different behavior. As described in the previous section, FlexLoader SC 6000 supplies a comprehensive set of tools and adjustment possibilities to handle these behaviors.

CAUTION

All predefined states and actions that are used in the template module shall be implemented or deleted. This minimizes the risk of unexpected machine behavior.

Continues on next page

An example implementation of the routine Machine1Action is shown below.

```
PROC Machine1Action(string sAction)
  TEST sAction
    CASE CLOSE_CHUCK_MAIN:
      ! TODO - Send signal to machine
      Set doMachineSpare01;
      WaitDI diMachineSpare06,1;
      Reset doMachineSpare01;
    CASE CLOSE_CHUCK_SUB:
      ! TODO - Send signal to machine
      Set doMachineSpare05;
      WaitDI diMachineSpare08,1;
      Reset doMachineSpare05;
    CASE OPEN_CHUCK_MAIN:
      ! TODO - Send signal to machine
      Set doMachineSpare02;
      WaitDI diMachineSpare05,1;
      Reset doMachineSpare02;
    CASE OPEN_CHUCK_SUB:
      ! TODO - Send signal to machine
      Set doMachineSpare03;
      WaitDI diMachineSpare07,1;
      Reset doMachineSpare03;
    CASE CLOSE_DOOR:
      ! TODO - Send signal to machine
    CASE PREPARE_LOAD_MAIN:
      ! TODO - Send signal to machine
      WaitUntil Machine1Wait(MAIN_CHUCK_OPENED,30)=TRUE;
    CASE PREPARE_UNLOAD_MAIN:
      ! TODO - Send signal to machine
    CASE PREPARE_UNLOAD_SUB:
      ! TODO - Send signal to machine
    CASE OPEN_DOOR:
      ! TODO - Send signal to machine
    CASE CYCLE_START:
      ! TODO - Send signal to machine
      PulseDO\High\PLength:=0.5,doMachineSpare04;
    CASE INIT_MACHINE:
      ! TODO - Send signal to machine
    CASE STOP_MACHINE:
      ! TODO - Send signal to machine
    DEFAULT:
      ! TODO - No valid action to perform,
      ! create alarm or do something if needed
  ENDTEST
ENDPROC
```

7.8 FlexLoader options functionality

Overview

The option modules provide methods for easy integration and control of various options in the RAPID code of the application.

Not all options are present in all machines. Options may need to be adapted to specific project needs.

For further reference, see *FlexLoader options functionality on page 222*

RAPID example

Air cleaning option

```
! Move robot to suitable position
StartAirClean NOZZLE_3;
! Make cleaning movements if necessary
StartAirClean ALL_NOZZLES;
! Make cleaning movements if necessary
StopAirClean NOZZLE_2;
! Make cleaning movements if necessary
StopAirClean ALL_NOZZLES;
! Continue process
```

Marking unit option

```
! Move robot to suitable position
SetMarker "Medium", "ExampleText";
StartMarker;
! Continue process
```

Statistical outlet

The statistical outlet is most often used in FlexLoader Vision Lite applications. The outlet handling is based on a few RAPID calls.

```
! based on predefined cycles and sample frequency
LeaveSamplePartIfNeeded;

! based on manual request controlled by integrator
PrepareLeaveInStatisticalOutlet;
LeaveSampleOutlet; (in ModCam1.mod)

! inform operator if part needs to be removed from outlet
PrepareSamplePartRemoval;
```

7.9 WorldZones

Some integrators prefer the use of WorldZones to verify the robots presence in certain areas/volumes, and/or to block certain movements. FlexLoader SC 6000 offers predefined WorldZone handling in the module Common.sys.

The FlexLoader SC 6000 defined WorldZones are active in both jogging mode and automatic mode.

The following zone is predefined and must be configured during commissioning.

- forbiddenVolume: A zone that e.g. lies above the volume that the robot needs for movements. Prevents the robot from moving through the forbidden volume.

The following zones are predefined for integrator convenience and can be configured during commissioning.

- safeVolume: The robot could be started from here directly. Connected to DOF_SafeZone.
- machineVolume: Robot is outside of machine tool. Connected to DOF_LoaderOut.

The definition of these WorldZones can be changed by geometrical data defined in Common.sys. After changing these values, the robot controller must be restarted in order to activate the new definitions.

For further details, see *Common.sys on page 205*.

8 Operation

8.1 Overview

During operation, the FlexLoader SC 6000 operator workstation is located where new material is added. During teachin, the operator workstation is located at the FlexLoader Vision monitor.

WARNING

FlexLoader SC 6000 may only be operated by trained operators with the required knowledge of the functions and risks related to FlexLoader SC 6000.

DANGER

Before FlexLoader SC 6000 is started or restarted, always check that all safety devices are working properly and that no damage can be caused at startup.

DANGER

Make sure that the air pressure to the gripper is safely shut down before replacing or repositioning the gripping fingers.

8.2 Powering on the FlexLoader SC 6000

1 Turn on the main power supply. Wait until all devices have started up.
2 Perform a power on safety check:
   - Press and restore at least one internal FlexLoader SC 6000 emergency stop button.
   - Reset the emergency stop.
   - Open and close at least one internal FlexLoader SC 6000 cell door.
   - Reset the door status by shortly turning the reset key.

     Optionally, if a light curtain is present, a long light curtain reset may be needed (0.5 s) first, followed by a second reset for the whole cell.
   - Refer to the safety manual for more detailed information.
3 Check the function of the external emergency stop
4 Check the function of the external protective stop (if present).

DANGER

Do not continue to work until the safety circuits are functioning correctly.

8.3 Shutting down the FlexLoader SC 6000

1 Stop FlexLoader SC 6000 and await a complete cycle stop.
2 Exit the FlexLoader Vision application.
3 Turn off the computer.
4 Turn off the main switch.

8.4 Operating FlexLoader SC 6000 with FlexLoader Vision

Standard start/stop operation

Cycle start

1. Ensure that the machine tool is ready for automatic operation and that the correct program is loaded.
2. Ensure that the correct details are supplied to the FlexLoader SC 6000.
3. Ensure that the doors are properly closed and that the emergency stop is not activated
4. If necessary, acknowledge any remaining error messages on the robot FlexPendant.
5. If necessary, turn the robot operating mode selector to automatic mode. Confirm the operation mode change.
6. Select the correct detail in the FlexLoader Vision user interface by choosing the correct group and detail.
7. Click **Start** in the FlexLoader Vision user interface. The selected robot program for the detail will now be loaded in the robot (if defined during teachin) and the FlexLoader SC 6000 will start.

Cycle stop

To stop the system, click on **Stop** in the FlexLoader Vision user interface.

Note that clicking **Stop** will not make the robot stop immediately. It will stop at the end of the cycle or after a certain maximum time.

Clicking **Stop** a second time will make the robot stop immediately. See FlexLoader Vision product manual for details.

If an emergency situation has arisen, use the emergency stop.

Robot stop

Perform a robot stop by pressing the stop button on the robot FlexPendant.

The robot will finish its current movement instruction and then stop. The external equipment will finish its current work cycle.

Restart after robot stop

Check that the robot can be restarted directly without any risk of damage. Then press the start button on the robot FlexPendant.

Starting with the robot in manual mode

DANGER

Running FlexLoader SC 6000 in manual mode with an operator close to the robot, the machine tool and FlexLoader SC 6000 options leads to increased risks for the operator.

Be careful and observant in order to avoid accidents.

Continues on next page

FlexLoader SC 6000 can be run with open doors. The robot operating mode must be manual with reduced speed.

Upon clicking **Start**, FlexLoader Vision notifies the operator about the manual mode. The following restrictions apply:

- The robot program will not be loaded automatically when FlexLoader Vision is started with the robot in manual mode.
- Do not forget to save the robot program after making changes. If FlexLoader SC 6000 starts in automatic mode, the robot program is reloaded and any changes made are lost.

Tip

A convenient way to automatically load the right modules is to start FlexLoader Vision with the robot in automatic mode.

As soon as the program start is completed, all modules are loaded correctly. The robot can then be put into manual mode for further work.

Continuing work in automatic mode after working in manual mode

1. Save the robot program if you have made any changes to it.
2. Reset the cell safety (see *Entry control on page 125*).
3. Set the robot operating mode selector to automatic.
4. Acknowledge the message on the FlexPendant.
5. Press **Motors On** on the robot operating panel.
6. Press the start button on the FlexPendant.

Note

When you have made changes to the system we recommend running the first production cycles with reduced speed so that you can react on any erroneous behavior.

Emptying cell

In some cases of machine tending, the cell must be emptied from parts, i.e. the machine shall be unloaded without loading the next part.

For FlexLoader Vision Lite, this behavior is pre-programmed. By setting the output **DOF_EmptyCell** to 1, the robot will finish a cycle by unloading a finished part without loading another raw part.

For details refer to the section on robot program.

8.5 Emergency stop

Emergency stopping the entire robot cell

Press any emergency stop button to stop the entire robot cell. All devices stop as quickly as possible. The blue reset button lights up on the operating panel, indicating that the emergency stop button has been activated. The blue button stays lit until the emergency stop has been reset.

Note

The emergency stop buttons must only be used in emergency situations.

Do not use emergency stop buttons to stop FlexLoader SC 6000 in normal operation.

WARNING

Note that grippers and other pneumatic equipment may contain stored pneumatic energy.

Restarting after emergency stop

WARNING

Ensure that a restart of FlexLoader SC 6000 does not cause any risks of damage to equipment or personal injury.

1. Twist (or pull out, depending on button model) all activated emergency stop buttons to reset them. The blue reset button on the operating panel starts flashing.
2. Reset the emergency stop mode by pressing the blue reset button so that the flashing blue light goes off.
3. Acknowledge the emergency stop message on the robot FlexPendant.
4. Ensure that the external equipment is ready for automatic operation.
5. If the robot cannot be automatically run back to a safe position (for example while loading or unloading the processing machine):

   Set the robot operating mode selector to manual. Jog the robot close to its safe position. Switch back to automatic mode. If the robot cell is restarted without any manual intervention, unexpected risk situations may occur.
6. Press **Motors On** on the operating panel.
7. Perform a standard cycle start.

8.6 Entry control

Introduction

The FlexLoader SC 6000 is well prepared for typical entry control handling.

The application RAPID code is expected to handle entry requests and operator interaction through it's I/O system. The optional FlexLoader Standard Safety handles the protective stop and manual operation of equipment.

The entry control prevents access to the cell unless the system is stationary. The protective stop mode prevents the cell from being started when someone is working inside the cell.

For more information on the FlexLoader Standard Safety, see *Application manual - FlexLoader Standard Safety Center*.

The example below describes operator interaction with the operator panel.

Standard operation

The entry control to the cell during automatic operation works as follows:

Stop the system in a controlled way by pressing the button for an entry request (C) at the door. The green lamp (C) starts to flash. The system stops at a suitable moment, so that the restart can occur without problems. As soon as the entry is permitted, the door is unlocked and the green lamp (C) is lit without flashing.

Take the key (D) and bring it with you when you carry out the work in the cell.

xx1900000172

Pos.	Description
A	Emergency stop
B	Start button (option)
C	Entry control button
D	Reset key switch

Continues on next page

DANGER

When entering the cell, the operator must, under all circumstances, bring the key that is used to confirm the protection door. This is to prevent somebody else from confirming and restarting FlexLoader SC 6000 when the operator is inside the cell.

If possible use only one key for the entire cell.

Close the door and turn the key switch (D) to confirm that the work in the cell is finished. The system can then be restarted with the start button (B) (option), or from the robot FlexPendant. Upon restart, the door is locked again.

The robot cell is in protective stop mode from the moment the door opens until the key switch (D) has been turned.

DANGER

Always make sure that nobody is inside the cell and that the system can be restarted without any risk of injury before confirming with the key switch and restarting the system.

During manual operation, the lock is automatically opened when the robot program has stopped. As soon as the door has been unlocked, the green lamp is lit without flashing.

Manual operation of dangerous equipment

By default the protective stop is activated as soon as any FlexLoader SC 6000 door is opened. This deactivates dangerous equipment in the cell. Sometimes, such equipment must be activated for test operation.

Therefore the FlexLoader SC 6000 electrical cabinet allows for using an alternate protective stop configuration (protective stop with override-function). This possibility should only be used by integrators and never in normal operation.

Press the enabling device on the FlexPendant and switch the door reset switch. The protective stop sent to the dangerous equipment is now overridden until the enabling device on the FlexPendant is released.

DANGER

It is the integrators responsibility to perform a suitable risk assessment for this mode of operation.

8.7 SafeMove operation

SafeMove stop

A SafeMove stop occurs when a predefined robot geometry or a predefined gripper geometry interferes with the predefined SafeZone volumes. The SafeMove supervision is active in both automatic mode and manual mode.

The main reasons for SafeMove stop are:

- Jogging the robot manually out of the safe areas.
- The robot is running faster than the allowed speed.
- Pulling out the statistical outlet while robot is moving.

Restart after SafeMove stop

- SafeMove stop occurred due to speed restriction:
  - Acknowledge the error message on the FlexPendant and press **Motors On**.
  - Check the robot program, reduce the speed in the relevant movement instructions, and save the robot program.
  - Perform a standard cycle start.
- SafeMove stop occurred due to jogging out of the safe areas:
  - Acknowledge the error message on the FlexPendant and press **Motors On**.
  - Jog the robot back into the safe area.
- SafeMove stop occurred because the statistical outlet was pulled out while robot was moving:
  - Push the statistical outlet back in place.
  - Acknowledge error message on FlexPendant and press **Motors On**,
  - Press **Motors On** once more.
  - Press the start button on the FlexPendant.

As soon as the operating mode switch is set to manual mode, the SafeMove manual operation (override) is enabled.

Cyclic brake check

The FlexLoader SC 6000 is programmed to automatically perform the cyclic brake check that is needed for the secure operation with SafeMove. The safety break check is normally performed at the home position of the robot.

Sometimes, it is necessary to perform this procedure manually. Please adhere to the robots guidelines in how to perform a cyclic break check.

In short, move the robot to a suitable position, preferably its home position. Then, call the service routine CyclicBrakeCheck.

Manual software sync

The FlexLoader SC 6000 is programmed to automatically initiate the software synchronization that is needed for the secure operation with SafeMove. The software synchronization is normally performed at the robots home position.

Continues on next page

Sometimes, it can become necessary to perform this procedure manually. Please adhere to the robots guidelines in how to perform a software synchronization.

In short, move the robot to a suitable position, preferably its home position. Then, call the service routine SoftwareSync.

8.8 Manual gripper control

WARNING

When replacing gripper fingers, the air pressure to the gripper must be disconnected.

For safety reasons, the manual control of the gripper is found in a separate menu (**Tool control menu**) in the FlexPendant. Press the function key  in order to start the menu (this function key must be connected signal **DOF_ToolMenuButton**).

Note that the gripper menu can only be opened when the robot is in manual mode.

In the **Tool control menu**, the operator can select between different actions:

- Open or close gripper 1.
- Open or close gripper 2.
- Open or close the turn station (option) gripper.
- Toggle turn station (option) rotation between rotated and non-rotated.

If the installation is equipped with an air vent valve, the air supply is checked at every system startup and every time the gripper is opened or closed manually. If the door is opened or the safety chain is interrupted for some reason, the air supply has to be reactivated.

To do so, press the enabling device on the FlexPendant. Turn the key to reset the door safety and to activate the air supply. As soon as the enabling device is released, the air supply is stopped.

8.9 Conveyor operations

Operator panel inconveyor

xx1900000382

Pos.	Component	Pos.	Component
A	Run inconveyor forward/backward	C	Switch outconveyor operation mode
B	Switch inconveyor operation mode	D	Emergency stop

Forward / backward inconveyor

When the button for running the inconveyor is turned in one direction during manual operation, the belt starts to run in that direction. The belt stops when either the sensor at the cameras field of view or the sensor at the end of the inconveyor is affected or when the button is quickly turned in another direction.

Backward operation is, however, only possible as long as the knob is held in position for backward operation.

Function selection inconveyor

For being able to run the inconveyor manually, the knob for selecting operating mode must be turned to manual operation. This requests permission for manual operation from the robot, and the lamp in the knob starts to flash. The lamp lights continuously as soon as FlexLoader SC 6000 can be run in manual mode.

WARNING

If you have stopped the robot in the middle of a started gripping manoeuvre, then run the belt manually, and then allow the robot to continue the grip manoeuvre, there is a risk of a collision.

Function selection outconveyor

When the knob is in position for emptying, the belt is run until the sensor at the end of the outconveyor is affected, otherwise the outconveyor is controlled by the robot.

If the parts have reached the end of the outconveyor, the belt stops and the yellow lamp (warning) is switched on. The operator must then pick off the parts. The belt

Continues on next page

either starts automatically when the parts are picked off, or the operating mode knob must be turned briefly (1s) to automatic mode and back.

Note

Running in automatic mode, avoid removing a part from the outconveyor just before the robot places another part.

Loading inconveyor

The conveyor can be loaded during manual or automatic operation.

Parts are placed on the conveyors loading area for processing with a distance that allows sufficient clearance for the grippers claw.

If the machine is in manual mode and the desired number of parts is loaded, the operator can run the belt forward using the manual mode button until the belt can be filled with further parts. Otherwise, the belt runs automatically until the first parts reach the cameras field of view.

During manual operation the conveyor can be run forward until the sensor at the cameras field of view is affected.

WARNING

Avoid placing raw parts directly on the link that holds the belt together. This could lead to unidentified parts or parts that are not picked due to false-identified gripper collision.

Emptying in- or outconveyor

The inconveyor can be emptied in manual operation mode. If the operator does not reach all parts on the conveyor, the belt can be run backwards using the backward operation button.

The outconveyor must be emptied in emptying operation mode. This is selected via the associated operation mode knob.

WARNING

If you have stopped the robot, and then run the outconveyor manually, and then let the robot continue, this may cause a collision risk, depending on which position pattern is used on the outconveyor.

Automatic operation

Inconveyor

During automatic operation, the inconveyor is controlled based on the robots requirement. Parts can be placed on the inconveyor during operation.

Outconveyor

During automatic operation, the outconveyor is controlled based on the robot being able to leave parts. Parts can be picked from the outconveyor during automatic operation.

Continues on next page

If the parts have reached the end of the outconveyor, the belt stops and the yellow lamp (warning) is switched on. The operator must then pick off the parts. The belt either starts automatically when the parts are picked off, or the operating mode knob must be turned briefly (1s) to emptying and back.

8.10 Indicator lamps

The indicator lamps at the front and back of FlexLoader SC 6000 are used to indicate the status of the system and notify the operator of any errors or warnings. Any error messages and warnings are also displayed on the FlexPendant.

The following functions and alarms are included in the default robot code of FlexLoader SC 6000. These can be changed by the programmer for project specific adjustments.

Green light: In operation

The indicator lamps are green when the robot is in operation.

Red light: Error

The indicator lamps are red when an error has occurred in the system. Example of possible causes:

- Unlocked door/s at startup.
- The robot cannot leave details on the outconveyor.
- Low air pressure.
- A detail has been transported too far on the inconveyor.

Yellow light: Warning

The indicator lamps are yellow when there is a warning message for the operator on the FlexPendant, e.g. there are no details on the inconveyor.

White light: Question

The indicator lamps are white when FlexLoader SC 6000 is awaiting information from the operator on the FlexPendant.

Blue light: Information

The indicator lamps are blue when there is information for the operator to read on the FlexPendant.

8.11 Statistical outlet

If configured so, the robot will leave sample parts in the statistical outlet with a specified frequency. The robot will wait for a previous sample part still in place to be removed.

Due to the size of the statistical outlet opening (which accommodates quite large details) and safety regulations, the robot must be in a safe standstill while the outlet is opened.

Take out sample part already in place

In order to take out a sample part that is already in place, press function key  on the FlexPendant. The robot will come to a safe standstill, and the sample outlet can be opened. When the sample outlet is closed again, the operator can confirm, and the robot will continue.

The FlexLoader Vision screen will show information, e.g. acknowledge of opening request, information on safe standstill, request to press P4 again to confirm program continuation, and so on.

If the robot is waiting to be able to place the next sample part, it already is at standstill, thus the sample outlet can be opened directly.

Order new sample anytime

In order to take a sample part independently of the specified sampling frequency, press function key  on the FlexPendant. The robot will place the next part in the statistical outlet and come to standstill, and the sample outlet can be opened. As soon as the sample outlet is closed again, the operator can confirm, and the robot will continue.

The FlexLoader Vision screen will show information, e.g. acknowledge of opening request, information on safe standstill, request to press  again to confirm program continuation, and so on.

Restart after inappropriate opening of outlet

If the outlet was opened while the robot was moving, please follow instructions in section on Emergency and SafeMove stop, see *Emergency stop on page 124* and *SafeMove operation on page 127*.

9 Maintenance

9.1 Maintenance schedule

WARNING

Always follow the general safety instructions, see *Safety on page 13*.

This section covers maintenance of both FlexLoader SC 6000 and options that are not included in all installations. If the installation contains further parts, maintenance instructions may also be found in other documentation.

FlexLoader SC 6000 is made of components that have a minimal maintenance requirement. However, there are some components that require scheduled maintenance.

Always make sure that FlexLoader SC 6000 is turned off during maintenance.

Below is an overview of the maintenance intervals and the corresponding corrective actions.

Maintenance intervals	Corrective action
Daily	Clean the conveyor belt.
Every week	Check the conveyor belt. FlexLoader SC 6000 overall cleaning. Clean the air cleaning box (option). Check the oil level of the compressed air lubricator (option).
Every month	Check the electrical functions. Check the gripper system. Clean marking unit stylus pin guide and assembly. Check the safety functions.
Every 3 months	Check the cables and the cable rack. Check the connections.
Every 6 months	Replace the compressed air filter cartridge
When necessary	Adjust the conveyor belt. Replace the illumination tower lamps. Check, clean and replace the sensors and sensor reflectors. Clean the camera optics. Clean the PC screen. Drain the air preparation condensate
If the system has been changed	Backup the robot program. Backup the vision system. Create recovery image of vision PC.
Automatic	Windows updates (performed as long as PC has internet connectivity)
After 10 million cycles.	Lubricate the gripper system.

9.2 Mechanical maintenance

FlexLoader SC 6000 overall cleaning

What: Overall cleaning of the FlexLoader SC 6000.

When: Once / week. More often if needed.

How: Let the robot move to its home position (-> entry request). Clean all surfaces from chips, oil and other material that might disturb correct function. Vacuum clean with industrial grade vacuum cleaner. Be careful when cleaning inside.

Robot, mechanical maintenance

What: Refer to the product manual of the robot for more information.

Clean the conveyor belt

What: Clean the conveyor belt.

When: Vacuum the felt belts at least once a week. Clean all other belts every day.

How: Felt belts should be vacuumed. All other belts are cleaned with a soap solution and a cloth. Run the belt to be able to clean the entire belt.

Check the conveyor belt

What: Check the conveyor belt.

When: Every week.

How: Run the conveyor belt and check that the belt does not slip at the motor drum nor moves sideways.

Adjust the conveyor belt

What: Adjust the conveyor belt.

When: When it starts to move sideways or slacken.

How: Refer to the installation sections for belt adjustment and belt tension.

Clean the air cleaning box (option)

What: Clean the air cleaning box.

When: Every week. More often if needed, for example when performing intense deburring.

How: Remove all chips and material from the deburring units. Vacuum clean with an industrial grade vacuum cleaner. Remove oil and dirt from deburring tools and their holders.

Check the gripper system

What: Check the gripper system.

When: Every month.

How: Visually check the gripper system, including gripper holder, gripper finger holders and gripper fingers, for signs of mechanical wear. Replace damaged or worn parts.

Continues on next page

Lubricate the gripper system

What: Lubricate the gripper system.

When: After 10 million cycles. At ambient temperature above 60 °C the lubricants can harden faster. Decrease the interval accordingly.

How: During maintenance, treat all greased areas with lubricant. Thinly apply lubricant with a lint-free cloth.

Surface	Recommended lubricant
Metallic sliding surfaces	Schunk LINOMAX or Schunk microGLEIT GP 360 or equivalent.
All seals	Schunk Renolit HLT 2 or equivalent.
Bores on the piston	Schunk Renolit HLT 2 or equivalent.

In addition to the described maintenance, the guides of the gripper can be re-lubricated as needed by means of lubricating nipples. The lubricating nipples can be used instead of the air purge connection. Remove the two adjustment screws for the air purge connection and replace them with two conical grease nipples.

For detailed information on gripper system lubrication, see Schunk manuals.

9.3 Electrical maintenance

Robot, control cabinet

What: Refer to the robot product manual for more information.

Illumination

What: Change the illumination units in the illumination tower.

When: When the illumination units are approaching the end of their service life, or when the first illumination unit breaks, it's recommended to replace all the remaining ones at the same time.

How: Remove the illumination unit from the holder and replace it by a new component.

If only one illumination unit breaks after a maximum of six months, it is sufficient to replace that particular unit only.

Sensors and reflectors

What: Check that all sensors and sensor reflectors (if any) are intact, clean and secured properly.

When: Check and clean when necessary, depending on the amount of dirt that comes from the details and the surrounding. Replace if necessary.

Check the safety functions

What: Check all emergency stops and emergency and safety switches.

When: Check every month and fix when necessary.

Electrical function check

What: Check all electrical functions, breakers and limit position functions

When: Once / month

Check the cables and cable racks

What: Check the cables and cable racks.

When: Every 3 months.

How: Inspect the entire cable rack (fixing points, dirt deposits, wear) and remedy any breaks. Check all cables. Replace any damaged cables. Extend cables that rub against sharp edges.

Check the connections

What: Check the connections.

When: Every 3 months.

How: Inspect all connections and make sure that they are secured properly and that there is no risk of play.

9.4 Pneumatic maintenance

Drain the air preparation condensate

What: Condensate drain.

When: When the condensate level is approximately 10 mm below the filter element.

How: Turn the drainage screw at the bottom of the bowl in an counterclockwise direction as seen from below. The condensate will then flow out.

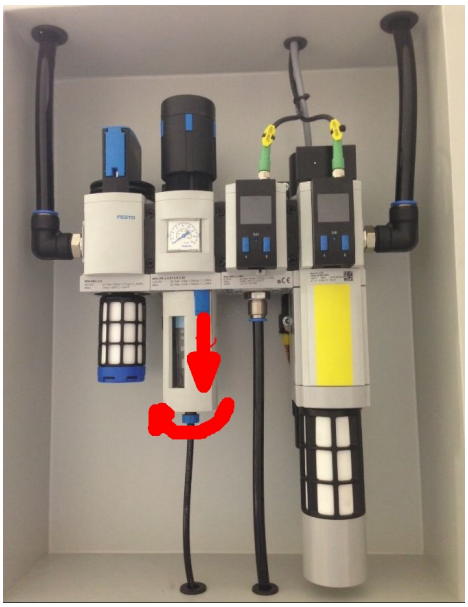
Replace the MS6-LFP-E filter cartridge

What: Replace the MS6-LFP-E filter cartridge.

When: Depending on the quality of the compressed air, generally 1–2 times/year, or when the air flow becomes to low.

How:

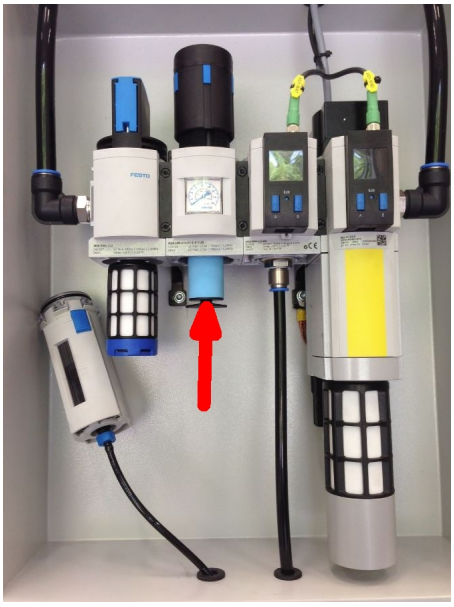
1. Exhaust the unit.
2. Push and hold the blue slider bar in the direction of the arrow.
3. Turn the filter bowl in an anti-clockwise direction as seen from below.
4. Pull the filter bowl away from the unit.
5. Turn the filter plate in an anti-clockwise direction.
6. Replace the filter cartridge if the pores are dirty. Grasp the filter cartridge only at the lower end when it is new.
7. Clean the bowl with water, soap solution (max. +60 °C), or petroleum ether (free of aromatic compounds).
8. Fit the individual parts again in the reverse order. Position the slider bar on the filter bowl facing the large recess in the housing and ensure that it clicks into place when the end stop is reached.



xx1900000383

Figure 9.1: Opening the filter housing

Continues on next page



xx1900000384

Figure 9.2: Replacing the filter cartridge

Check the oil level of the compressed air lubricator

What: Check the oil level of the lubricator (if a deburring/grinding unit is used). Refill when necessary.

When: Depending on usage of the deburring/grinding unit, generally every week.

How: The lubricator is placed in front of the air cleaning box. It has an inspection window that shows the oil level. If the oil level is below the minimum level mark, refill it by pulling the blue slider down and turn the bowl clockwise until it is loose. Refill with a suitable oil until the oil level is above minimum level.

The recommended oil is Festo special oil as per ISO VG 32 (type Festo OFSW-32).

9.5 Backup

Backup the vision system

What: Do a backup of the details stored in the vision system.

When: When you have made a change to the system.

How: Create a backup on the **Service** tab on the **Settings** menu and save it onto a secure media.

Backup the robot program

What: Do a backup of the robot program.

When: When you have made a program change.

How: Follow the robot manual and save the copy onto a secure media.

Recovery image of the vision system

What: Do a complete recovery image of the vision system.

When: After major system changes, upgrades or updates.

How: Follow instructions in the FlexLoader Vision product manual.

9.6 Other maintenance

Clean the camera optics

What: Clean the camera optics.

When: When the image quality is reduced by dirt.

How: Wipe the front of the object using a clean cloth. If this is not sufficient, use pure alcohol or similar. Ensure not to leave any streaks on the lens.

Note

If you happen to change the camera position, perform a new calibration.

Clean the computer screen

What: Carefully wipe the screen off using a damp cloth.

When: As necessary.

Marking unit

What: Clean the stylus pin guide and the stylus assembly regularly.

When: Depends on amount of dust and dirt in the cell. Every month.

How: Unplug the marking machine. Unscrew the stylus pin guide (see marking unit manual). Remove the stylus, the spring and the core. Clean all parts and remove the grease. Lubricate the stylus and the stylus pin guide using the oil supplied with the marking unit. Reassemble the machine and manually fasten the stylus pin guide. Note : please pay attention to the direction when reassembling the core (see marking unit manual). Avoid dust and abrasive particles on the guiding and driving elements.

Windows updates

What: Download and install of Windows updates.

When: Windows updates are downloaded and installed by standard Windows functionality as long as the PC has internet connectivity.

How: Normally no user interaction is necessary. If the system needs to restart this is performed by standard windows functionality.

Note

Feature updates shall not be installed on the PC. This is defined by deselecting the "Defer feature updates" switch.

10 Repair

10.1 Introduction

This section describes how to repair FlexLoader SC 6000 and options that are not included in all installations. If the installation contains further parts, repair instructions may also be found in other documentation.

Most repairs of FlexLoader SC 6000 are relatively straightforward to carry out. The most important procedures can be found in this section. Minor repairs that can be performed with common industry methods are not described in this manual.

WARNING

Always follow the general safety instructions, see *Safety on page 13*.

Note

How to repair the robot is described in the robot product manual.

10.2 Conveyors

DANGER

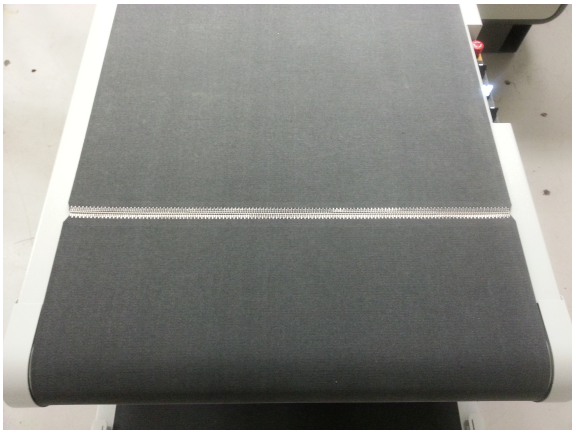
Prior to all work with the conveyors:

After positioning the belts in suitable positions, safely shut the power to the FlexLoader SC 6000 down and lock against accidental turn-on.

Exchange of conveyor belt

Exchange a conveyor belt by the following procedure

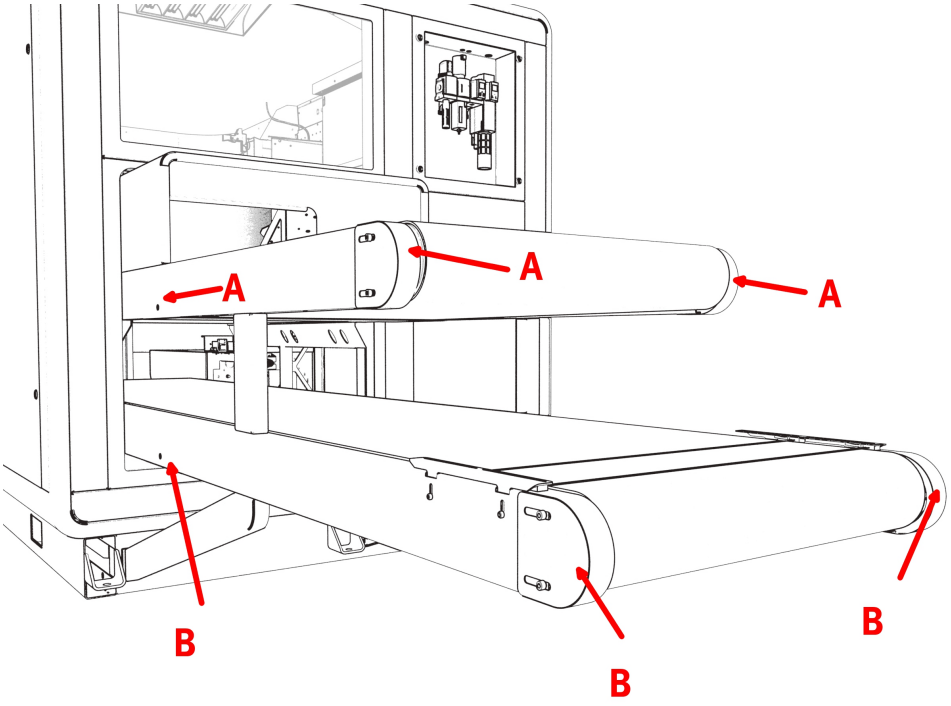
- Run the belt to a suitable exchange position. The belt splice should be close to the end of the conveyor.



xx1900000385

- Remove the relevant protection cover at the end of the belt (see illustrations below).
- Reduce belt tension by releasing the tensioning screws (see illustrations below).
- Remove the idler drum below the middle of the conveyor (see illustrations below).

Continues on next page

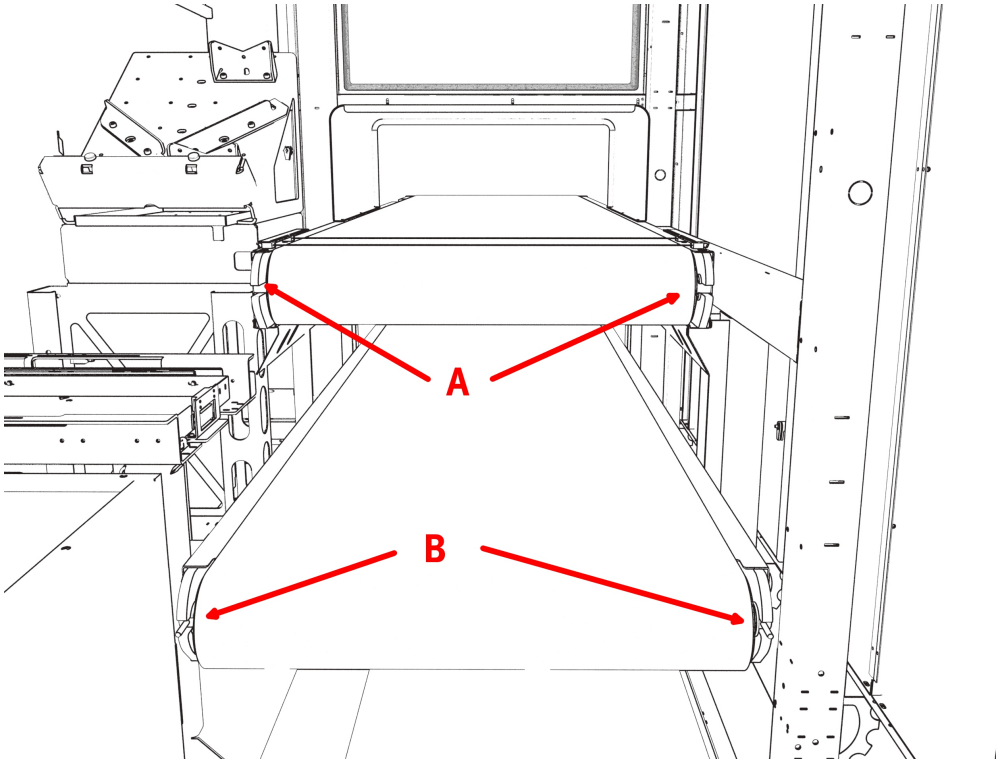


xx1900000386

Protective caps and release points idler drum in middle of conveyor

Pos.	Description
A	In-conveyor points of interest
B	Out-conveyor points of interest

Continues on next page

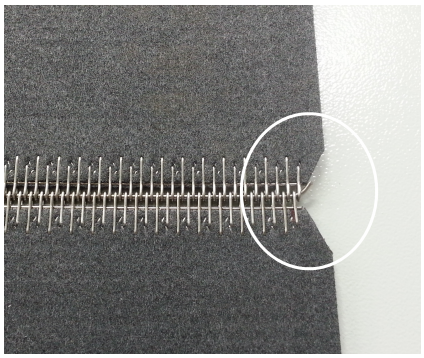


xx1900000387

Location of belt tensioning screws

Pos.	Description
A	In-conveyor points of interest
B	Out-conveyor points of interest

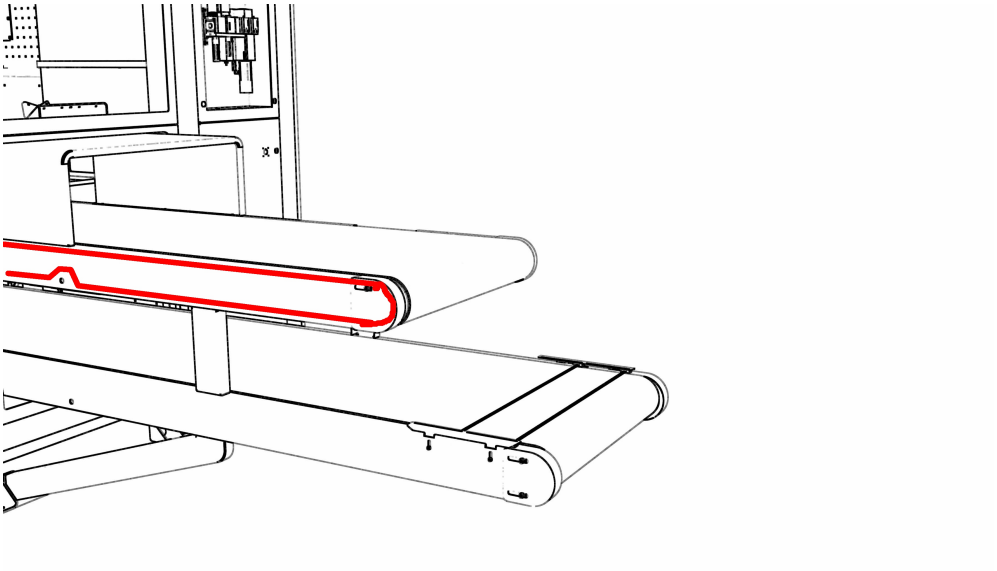
- Open the belt splice by straightening the lock wire.



xx1900000388

- Connect the new belt to the old belt splice.
- Drag the new belt with the help of the old belt through the FlexLoader SC 6000.
- Remove old belt, then connect and lock the new belt splice.
- Refit the idler drum below the middle of the conveyor.
- Take special care to the belt path below the conveyor.

Continues on next page



xx1900000389

The belt path should be as indicated in picture.

- Adjust belt tension according to instructions in the maintenance section.

DANGER

Power to the FlexLoader SC 6000 has to be restored prior to mounting the protection caps, in order to perform the belt adjustment. Be careful when adjusting the belt. Risk for finger squeeze.

- Adjust belt side position and stability according to instructions in the maintenance section.
- Mount the protection cover.

Exchange of conveyor motor

- Disconnect the motor cable in the electrical cabinet.
- Follow the instructions for belt exchange.
- When the belt is opened, replace the conveyor motor.
- Refit the belt and check belt adjustment and stability according to instructions in the maintenance section.

10.3 Illumination

DANGER

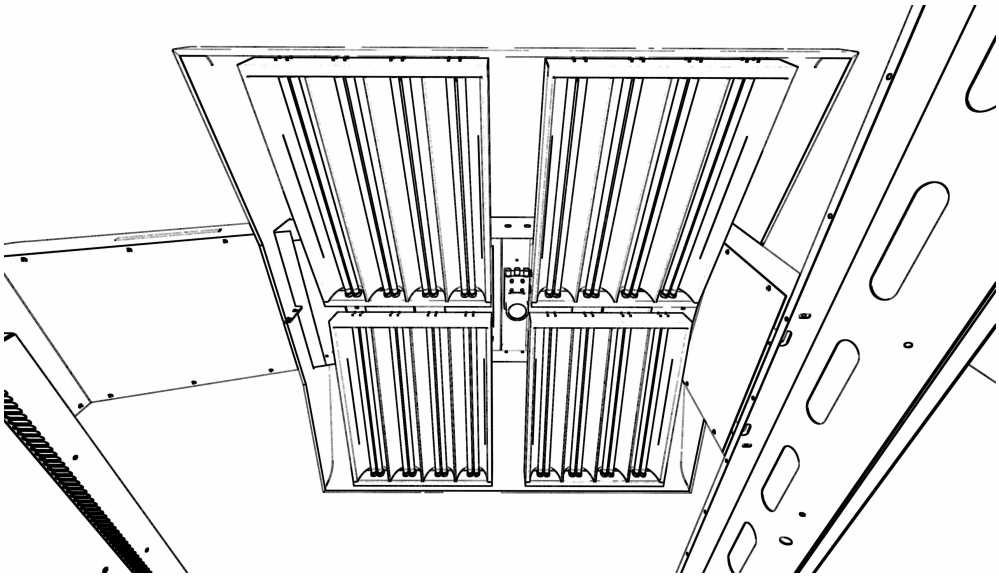
Prior to work with the illumination:

Safely shut the power to the FlexLoader SC 6000 down and lock against accidental turn-on.

Exchange of illumination units is described in the maintenance section.

In rare cases, the complete luminaires have to be exchanged. Use the following procedure:

- Identify the defect luminaires.
- Remove the fluorescent tube and the reflector.
- Remove the inner protection housing.
- Loosen the power cables.
- Remove the luminaires.
- Replace by the new luminaires and assemble in reverse order.



xx1900000390

Layout of luminaires and camera

10.4 FlexLoader Vision

Replacing the camera lens

To change the camera lens with the least possible impact on existing detail teachin, use the following procedure:

1. Unscrew the old lens.
2. Fit the new lens on the camera.
3. Adjust the focus of the new lens.
4. Use an existing image, e.g. one of the existing teachin images, and compare the image brightness. Adjust the aperture to match the appearance with the old lens.
5. Calibrate the vision system with the robot according to the FlexLoader Vision manual.
6. Run the system at a slow speed for the first few picks.

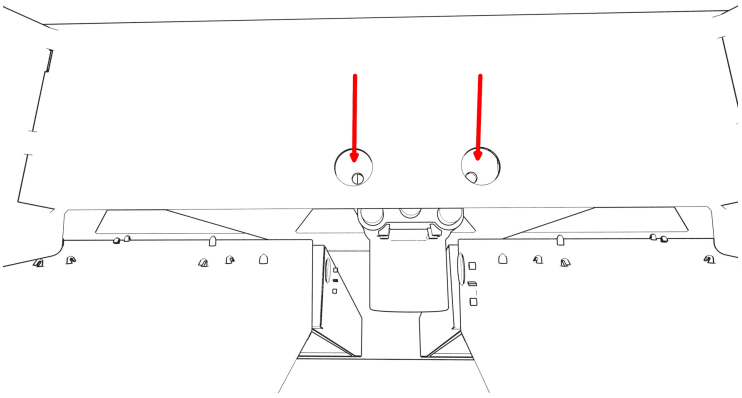
Replacing the camera

To replace the camera with the least possible impact on existing detail teachin, use the following procedure:

WARNING

Before replacing the camera: Switch off the current to FlexLoader SC 6000 and lock the switch to prevent unintentional switching on.

1. Disconnect the camera cable.
2. Remove the screws holding the camera mounting plate, preferably using a magnetic tool.



xx1900000391

Figure 10.1: Mounting plate screws

Continues on next page

3 Remove the camera and replace it by a new camera.
4 Move the lens from the old camera to the new camera. See *Replacing the camera lens on page 149*.
5 Reinstall the camera mounting plate on the FlexLoader SC 6000 frame. Do not tighten the screws firmly yet.
6 Attach the camera cable.
7 Turn on power to the FlexLoader SC 6000.
8 Adjust the position of the camera mounting plate. Use an existing teachin as a reference. The camera image (field of view) should look the same as in the existing teachin.
9 Tighten the screws holding the camera mounting plate.
10 Calibrate the vision system with robot according to the FlexLoader Vision manual.
11 Run the system at a slow speed for the first few picks.

10.5 Standard gripper

Safety

DANGER

Before carrying out work on the gripper: Safely shut down the pneumatic supply to FlexLoader SC 6000 and lock the switch to prevent unintentional switching on.

Replacing the gripping module on the gripper

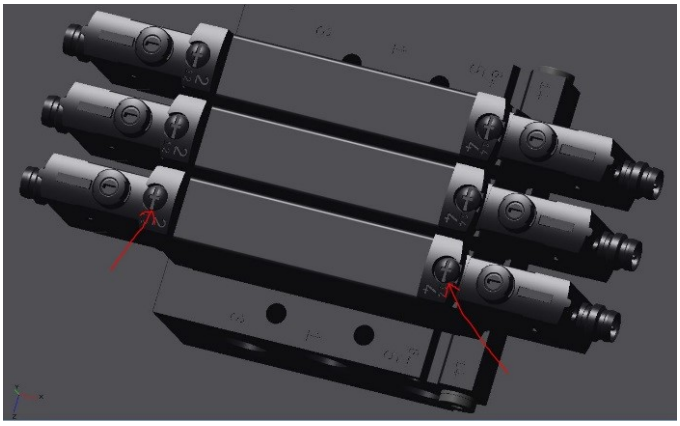
To replace one or more gripping modules (Schunk):

1. Move the robot to a suitable service position.
2. Unscrew the gripper fingers.
3. Mark the pneumatic tubes if necessary.
4. Loosen the pneumatic tubes.
5. Remove the pneumatic connections.
6. Unscrew the gripper module.
7. Fit the new gripper, pneumatic connections and pneumatic tubing.
8. Fit the gripper fingers.
9. Restore the pneumatic supply to FlexLoader SC 6000.
10. Test the new gripper.

Replacing the gripper valves

To replace one or more valve modules (Festo):

1. Mark the signal cables if necessary.
2. Loosen the signal connectors if necessary.
3. Unscrew the two screws holding the control valve. See the figure below.
4. Pull the control valve straight up.
5. Assemble in reverse order.



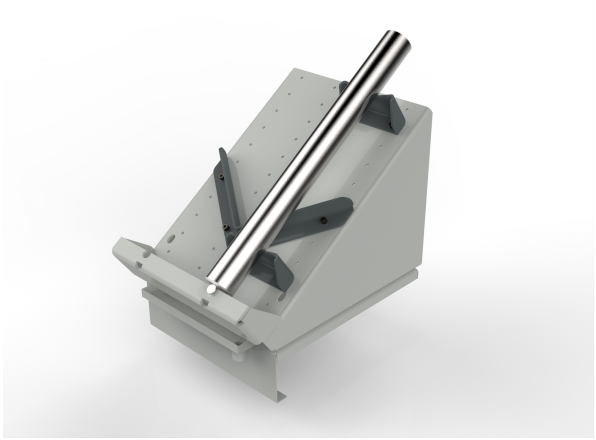
xx1900000392

Figure 10.2: Control valve screws

10.6 Re-grip table

Mechanical parts on the re-grip table can - if worn out - be replaced by spare parts. No special procedure has to be followed.

However, check if the robots work object has to be redefined or if picking positions have to be adjusted.



xx1900000393

xx1900000394

FlexLoader SC 6000 re-grip table in two different configurations

10.7 Turn station

DANGER

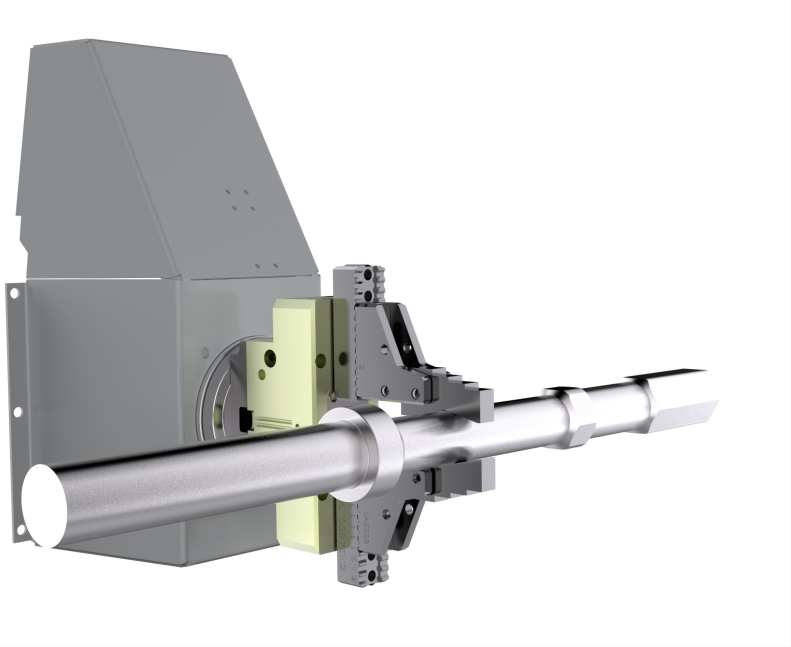
Prior to work with the turn station:

Safely shut the pneumatic supply to the FlexLoader SC 6000 down and lock against accidental turn-on.

Exchange of gripping module on turn station

The gripping module (Schunk) can be exchanged by the following procedure

- Move the robot to a suitable service position.
- Unscrew gripper fingers.
- Remove pneumatic connections.
- Unscrew gripper module.
- Mount new gripper.
- Mount gripper fingers.
- Restore pneumatic supply to FlexLoader SC 6000.
- Test the new gripper.



xx1900000395

FlexLoader SC 6000 turn station with axis in gripper

Exchange of rotation module

The rotation module can be exchanged by the following procedure

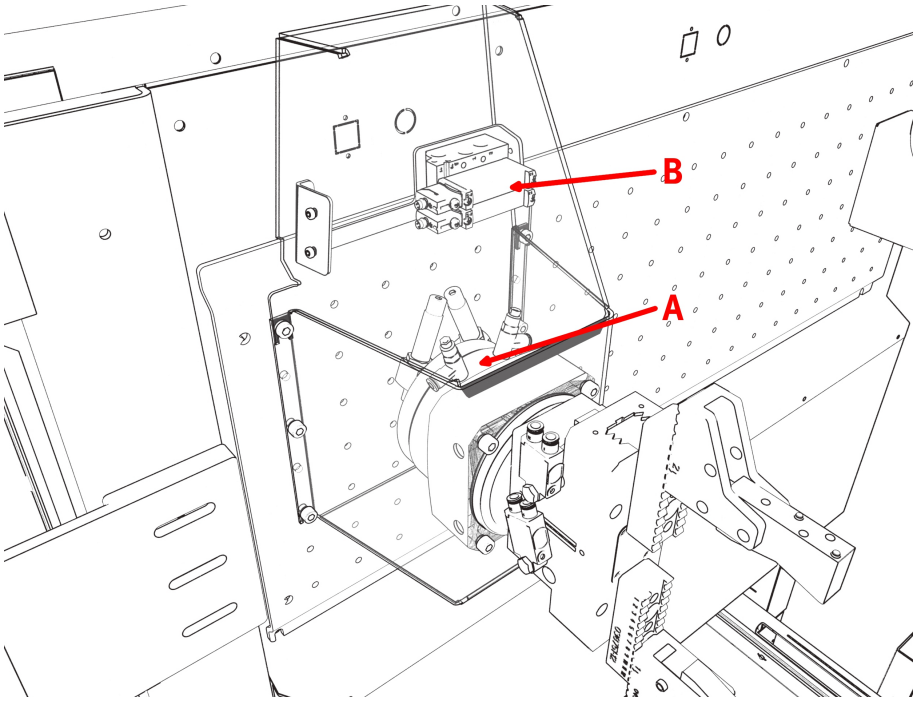
- Remove electrical and pneumatic connections to the turn station. Remove turn station from holding panel by unscrewing the six holding screws.
- Follow the instructions above form removal of gripping module.

Continues on next page

- Remove internal pneumatic connections to the rotation module.
- Dismount and exchange the rotation module.
- Assemble in reverse order. Be sure to adjust the angular stop positions to the desired range.
- Test the turn station gripper.

CAUTION

If there are any changes in position/design between the new and old unit the robot positions need to be updated.



xx1900000397

FlexLoader SC 6000 turn station, showing the rotation module and the valve positioning

Exchange of valves

For exchange of valve modules in the turn station follow the first steps of instructions above (exchange of rotation module) in order to obtain access to the valve package.

Then follow the procedure for changing valves on the gripper.

The valves are positioned in the turn station housing, as seen in the figure above.

10.8 Air cleaning box (option)

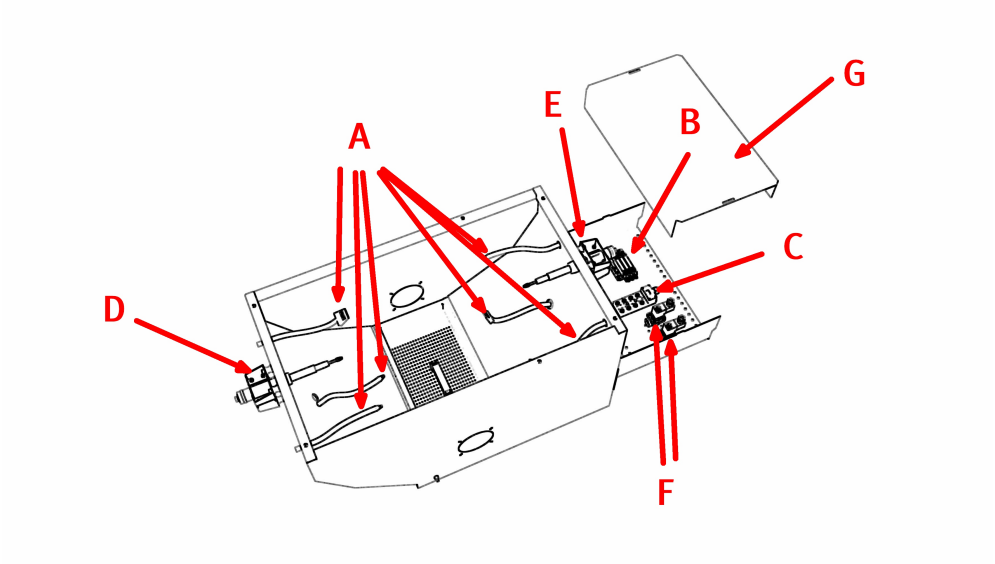
Safety

DANGER

Before carrying out work on the air cleaning box:

Safely shut down the pneumatic supply to the FlexLoader SC 6000 and lock the switch to prevent unintentional switching on.

Components of the air cleaning box and deburring tool



xx1900000398

Pos.	Description
A	Flexblow hoses
B	Valves controlling the air cleaning nozzles
C	Signal splitter box
D	Deburring tool 1
E	Deburring tool 2
F	Valves controlling the deburring tools

Replacing the flexblow hoses

If a flexblow hose holding a blowing nozzle gets unstable, it can be replaced the following way:

1. Use a wrench to hold the pneumatic connection. Turn the flexblow hose counterclockwise until it is loose.
2. Turn the blowing nozzle counterclockwise to release it from the hose.
3. Replace the flexblow hose.
4. Assemble in reverse order.

Continues on next page

CAUTION

If there are any changes in position or design between the new and old unit the robot positions must be updated.

Replacing the valves controlling the air cleaning nozzles

To replace one or more valve modules (Festo), use the same procedure as for the gripper valves, see *Standard gripper on page 151*.

10.9 Grinding/Deburring units (option)

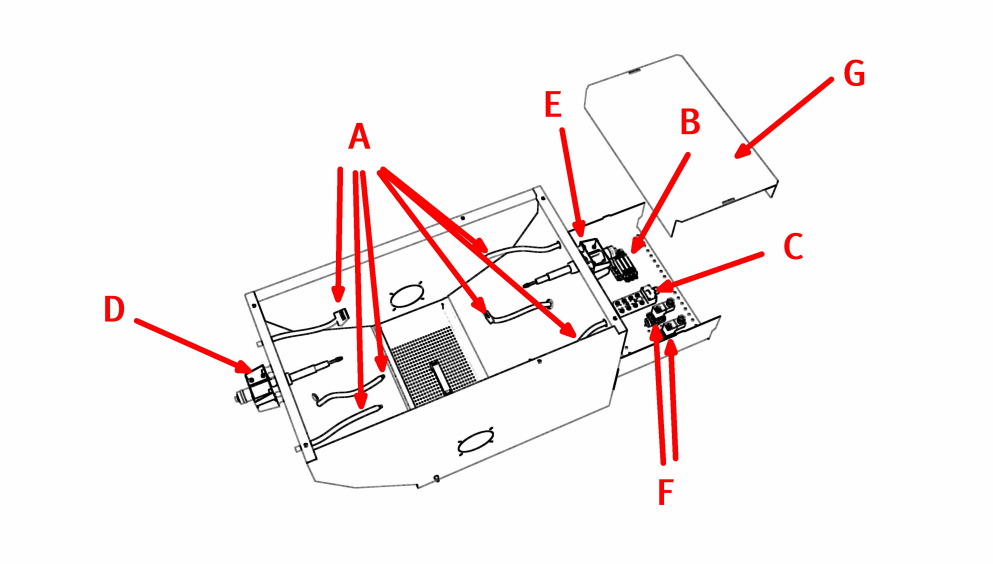
Safety

DANGER

Before carrying out work on the grinding or deburring units:

Safely shut down the pneumatic supply to the FlexLoader SC 6000 and lock the switch to prevent unintentional switching on.

Components of the air cleaning box and deburring tool



xx1900000398

Pos.	Description
A	Flexblow hoses
B	Valves controlling the air cleaning nozzles
C	Signal splitter box
D	Deburring tool 1
E	Deburring tool 2
F	Valves controlling the deburring tools

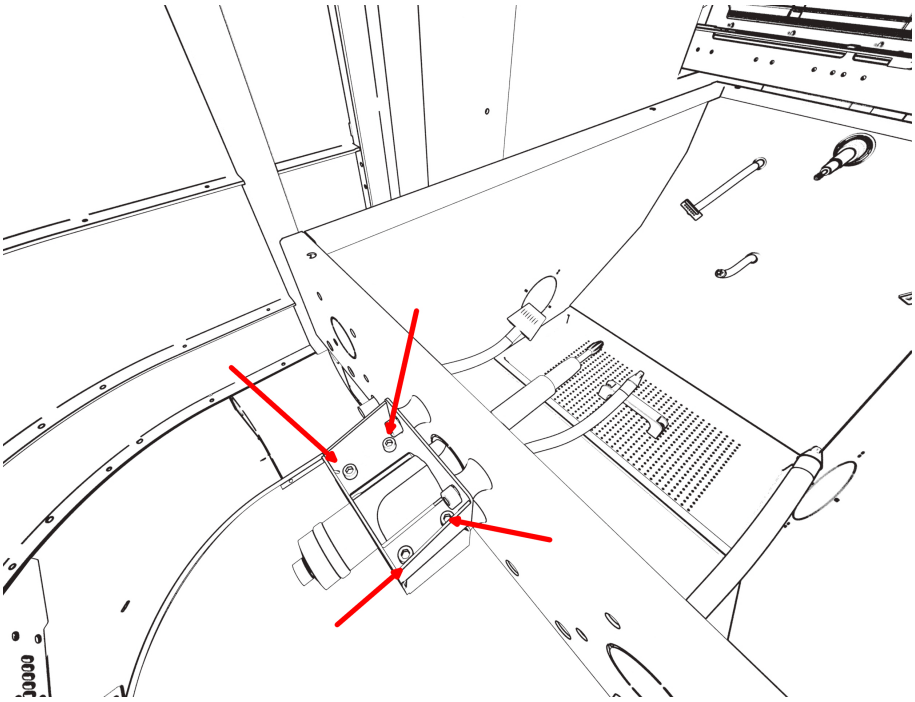
Replacing the grinding/deburring unit

To replace the grinding/deburring unit:

1 Disconnect the pneumatic connection to the grinding/deburring unit.

Continues on next page

2 Remove the four bolts and nuts holding the bracket. See figure below.



xx1900000399

Figure 10.3: Grinding/deburring unit fastening system.

3 Pull the unit backwards.
4 Install the new unit.
5 Connect the pneumatic connection to the grinding/deburring unit.
6 Restore the pneumatic supply to FlexLoader SC 6000.

CAUTION

If there are any changes in position or design between the new and old unit the robot positions must be updated.

Replacing the valves controlling the deburring/grinding units

One or more valve modules (Festo) can be replaced by the following procedure:

1 Disconnect the pneumatic connections that connects the valves to FlexLoader SC 6000.
2 Disconnect the electrical connection to the distributor box.
3 Disconnect the pneumatic connections and the two M4 screws, placed underneath, holding the valve.
4 Replace the valve with a new one.
5 Reinstall the air cleaning box in reverse order. Make sure the air cleaning box is correctly connected to the pins placed in the FlexLoader SC 6000 frame.

Continues on next page

CAUTION

If there are any changes in position or design between the new and old unit the robot positions must be updated.

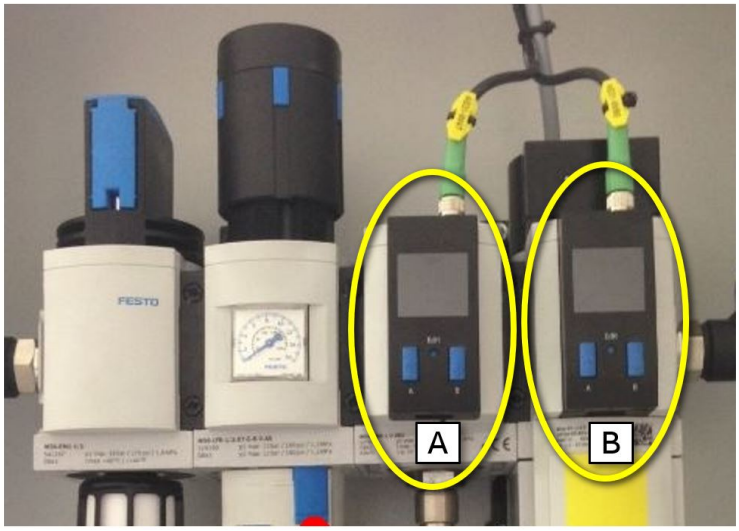
CAUTION

Run the system at a reduced speed during the first cycles, to ensure that the air cleaning box is in the correct position.

10.10 Air preparation units

Replacing the air preparation pressure switch (if available)

Follow the instructions from the manufacturer on how to perform the replacement.
Adjust the set point and hysteresis after replacing the pressure switch.



xx1900001059

Pos.	Component
A	Incoming air pressure switch
B	Safety valve pressure switch

The values are pre-configured as follows:

- Incoming pressure switch (left):
  Switching pressure set point **2** bar/**29** psi.
  Hysteresis **0.5** bar/**7** psi.
- Pressure switch on safety valve (right)
  Switching pressure set point **4** bar/**58** psi.
  Hysteresis **0.0** bar/**0** psi.

Make sure that the earthing strap is electrically connected to the outer housing.

To change the values, do as follows:

1. Press **EDIT** on the pressure switch.
2. Press **EDIT** button until the display says **[SP]**.
3. Use the UP/DOWN button to set the the correct value.
4. Press **EDIT** button once and the display says **[Hy]**.
5. Use the UP/DOWN button to set the correct value.
6. Press **EDIT** twice.
7. Done.

Continues on next page

Replacing the safety valve (if available)

Follow the instructions from the manufacturer on how to perform the replacement. Adjust the flow control valve of the safety valves after replacing the safety valve according to the following instructions:

1. Open the flow control valve fully (situated on top of the safety valve).
2. From this position, close the valve by turning the screw five turns clockwise.
3. If the valve opens too fast or too slow, adjust the valve by half a turn at the time, until the system operates correctly.
4. Make sure the pressure switch is adjusted as described above.

10.11 Marking unit

Exchange of marking unit

The marking unit can be exchanged by the following procedure:

- Disconnect the electrical contact by turning it anticlockwise
- Unscrew the four M6 screws holding the marking unit
- Replace with a new marking unit in reverse order.

CAUTION

If there are any changes in position/design between the new and old unit the robot positions need to be updated.

CAUTION

Run system at slow speed for the first cycles to ensure the air cleaning box is in correct position.

10.12 Robot mounting/dismounting

Robot weight

The weight for some example robots is shown below. The weight of other robots can be found in the robot manual.

- 4600, 60kg, 2.05m: 425 kg
- 2600, 20kg, 1.65m: 272 kg

Specification of bolts and washers

The required bolts for some example robots are shown below. Data for other robots can be found in the robot manual.

- ABB 4600: 6 bolts M16 x 60, 6 washers 17 x 30 x 3, quality 8.8
- ABB 2600: 3 bolts M16 x 60, 3 washers 17 x 30 x 3, quality 8.8

Tightening torque 200 Nm

How to mount or dismount a robot to FlexLoader SC 6000

DANGER

Avoid dangerous robot swing movements by carefully taking into account the center of mass of the hanging robot.

Mount the robot to FlexLoader SC 6000 by the following procedure

- Use a fork lift and place the pallet with the robot as close as possible to the FlexLoader SC 6000 robot pedestal.
- Attach straps to robot.
- Attach the straps to a forklift with enough handling weight to handle the robots weight.
- Place the robot at the FlexLoader SC 6000 robot pedestal.
- Secure the robot by mounting the M16 bolts, tightening torque 200 Nm.
- Mount electric and pneumatic connections to the robot.

For dismounting the robot, follow the steps above in reverse order.

Robot fine calibration

When fine-calibrating the robot, it must be on flat level surface. Thus it must be dismounted from the FlexLoader SC 6000.

10.13 Software

General

When delivered, a backup of the robot and FlexLoader Vision is included. Both backups are saved on the FlexLoader Vision PC.

These backups are normally located in the folder D:\Backup.

Robot backup and recovery

A robot backup that restores the robot system to its factory settings is saved on the FlexLoader Vision PC hard disk.

Copy this backup to a USB memory stick and restore it with standard robot backup restore procedures (see the robot product manual).

FlexLoader Vision backup and recover

A FlexLoader Vision backup that restores the system to its factory settings is saved on a separate USB drive.

See the backup and recovery information in the FlexLoader Vision manual for more information.

11 Troubleshooting

11.1 Alarms, warnings and informations and troubleshooting

Introduction

A number of alarms, warnings and informations are generated from the system. All messages are accompanied by corresponding light tower signalling.

Colors are blue (B) for information, yellow (Y) for warning, and red (R) for alarms.

General messages and options

Pos.	Symbol	Description	Color
91 202	NOT_VALID_ROBOT_ZONE	No valid zone detected in CheckPos. Jog robot to known zone.	R
91 203	LOW_AIR_PRESSURE	Low air pressure detected in CheckSystem. Check air preparation.	R
91 204	GATE_NOT_LOCKED	Gate not locked detected during system start. Lock gate with reset switch and restart.	R
91 205	WRONG_TARGET_POSITION	Internal error in MoveTo-routine. No valid target zone. Check user RAPID code.	R
91 206	NO_DETAIL_IN_POSITION	No parts on inconveyor. Load more parts.	B
91 207	PRESENT_POSITION_UNIDENTIFIED	Current position/zone of the robot cannot be determined. Jog robot to suitable position.	R
91 208	CANT_FIND_SIGNAL_NAME	Wrong signal specification when calling configuration routines. Check I/O configuration and signal names.	R
91 300	IMAGE_GRABBING_PROBLEM	No image coordinate received from FlexLoader Vision. Check FlexLoader Vision status.	R
91 301	NO_DETAIL_FOUND	No detail found by FlexLoader Vision. Check teachIn, illumination and part appearance.	B
91 302	BAD_PICKMT_COORD	Bad pick coordinates or pick offset from FlexLoader Vision. Check pick offset value in Z and camera calibration.	R
91 360	MACHINE_DOOR_NOT_OPENED	Machine door tool was not opened within time limit. Check machine tool.	R
91 361	MAX_HEIGHT_EXCEEDED_UNLOAD	Maximum part height for unloading exceeded. Reduce height or check value of FEEDER_MAX_HEIGHT.	R

Continues on next page

11.1 Alarms, warnings and informations and troubleshooting
Continued

Pos.	Symbol	Description	Color
91 362	TOO_SMALL_DETAIL_DIAMETER	Minimum part diameter reached. Increase diameter or check value of MIN_RAW_DETAIL_DIAMETER.	R
91 363	WRONG_CONFIGURATION	Wrong detail configuration from FlexLoader Vision. Make sure to configure detail in FlexLoader Vision to match machine operation type.	R
91 365	SAFEMOVE_OVERRIDE_ACTIVE	SafeMove override active during robot start. Switch to normal mode.	R
91 366	SOFTWARE_SYNC_NEEDED	Software synchronization needed. Turn robot to manual, move to sync position and call SoftwareSync.	R
91 400	SAMPLE_OUTLET_NOT_IN_POSITION	Statistical outlet is not pushed in. Push outlet into position and press **Play**.	Y
91 401	DETAIL_IN_SAMPLE_OUTLET	Part is still present in statistical outlet. Remove part and press **Play**.	Y
91 402	STATISTICAL_DETAIL_ORDERED	Control detail ordered. To abort detail press button 4 on the FlexPendant again.	B
91 403	STATISTICAL_DETAIL_ABORTED	Control detail aborted. To order detail press button 4 on the FlexPendant again.	B
91 404	STATISTICAL_OUTLET_ACCESS	Operator requested access to statistical outlet. Wait until robot is in SafeStandStill.	B
91 405	STATISTICAL_OUTLET_REMOVE	Detail in statistical outlet. Open statistical outlet and remove detail. Restart robot with button 4 on the FlexPendant.	B
91 406	SAMPLE_OUTLET_SENSOR_ERROR	No detail in the statistical outlet when expected. Check the statistical outlet sensor.	R
91 407	STATISTICAL_OUTLET_READY	Detail ready in statistical outlet. Request robot standstill with button 4 on the FlexPendant.	B
91 420	MARK_TEXT_CHANGE_PROBLEM	Marker unit text could not be changed. Check communication status. Reset marker unit.	B
91 421	MARK_TEXT_TOO_LONG	Marker unit text to long. Shorten text to be written.	R
91 422	MARK_FILE_LOAD_PROBLEM	Marker unit file could not be loaded. Check file integrity and reset marker unit.	B
91 423	MARKER_COM_TIMEOUT	Marker communication timeout. Check communication status and reset marker unit.	B
91 424	MARKER_UNEXPECTED_ANSWER	Marker unit unexpected response. Check communication status and reset marker unit.	R

Continues on next page

Conveyor subsystem

Pos.	Symbol	Description	Color
91 310	TIMEOUT_IN_POSITION	Timeout inconveyor. Place parts on belt and reset inconveyor.	R
91 311	ALARM_PARTS_LOW_BELT1	No details on inconveyor. Load new details on inconveyor.	B
91 312	ALARM_TOO_FAR_BELT1	Part went to far on inconveyor. Remove part and reset inconveyor.	R
91 313	ALARM_EMERGENCY_STOP_BELT1	Emergency stop. Reset cell to normal operation state.	R
91 314	ALARM_FC_BELT1	Alarm frequency inverter inconveyor. Check frequency inverter and inconveyor.	R
91 315	ALARM_FC_OUTBELT1	Alarm frequency inverter outconveyor. Check frequency inverter and outconveyor.	R
91 316	TIMEOUT_MANUAL_FORWARD_BELT1	Timeout manual run forward inconveyor. Restart belt with run forward knob.	R
91 317	TIMEOUT_MANUAL_BACKWARD_BELT1	Timeout manual run backward inconveyor. Restart belt with run backward knob.	R
91 318	TIMEOUT_EMPTYING_OUTBELT1	Timeout emptying outconveyor. Restart outconveyor with outconveyor auto/manual switch.	R
91 319	ALARM_TOO_FAR_OUTBELT1	Part reached end of outconveyor. Remove part. Restart outconveyor with outconveyor auto/manual switch.	R
91 320	ALARM_TOO_FAR_FAULTY_OUTBELT1	Faulty signal from TooFar sensor outconveyor. Check sensor function.	R
91 321	ALARM_IN_BELT_MANUAL_MODE	Inconveyor is in manual mode and cannot be controlled by robot when necessary.	R
91 322	ALARM_ROBOT_BLOCK_EMPTYING	Robot is blocking possibility for emptying mode. Wait for robot block signal to reset or reset it manually.	R
91 323	OUT_BELT_IN_EMPTYING	Outconveyor in emptying mode.	B
91 324	ROBOT_CANT_LEAVE_OUTBELT	Robot is not allowed to leave on outconveyor. Empty outconveyor and make sure that is is in auto mode.	R

FlexLoader Vision

Alarms for the FlexLoader Vision system are described in the FlexLoader Vision manual.

Troubleshooting information concerning FlexLoader Vision, e.g. teachin or accuracy problems, can be found in the FlexLoader Vision manual.

Other FlexLoader Vision related problems can often be solved by checking the FlexLoader Vision generated log-file. Refer to the FlexLoader Vision manual for information on extended logging.

Continues on next page

Robot

Troubleshooting information concerning the robot can be found in the robot manuals.

Marking unit

The marking unit can - if in rare cases necessary - be reset by shortly pressing the red button in the control cabinet. It is located directly on the side of the marking unit control unit.

12 Decommissioning

12.1 General

Introduction

This section contains information to consider when taking a product, robot or controller, out of operation.

It deals with how to handle potentially dangerous components and potentially hazardous materials.

When decommissioning the FlexLoader SC 6000, start with decommissioning of the robot according to the robot product manual.

Disposal of storage media

Before disposal of any storage equipment (anything from an SD card to a complete controller), make sure that all sensitive information has been deleted.

Tip

To remove all data from the SD card, use the **Clean Disk** function (part of **Recovery Disk** function) in RobotStudio. See *Operating manual - RobotStudio*.

12.2 Environmental information

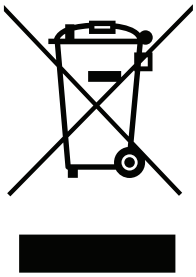
General

All used grease/oils and dead batteries **must** be disposed of in accordance with the current legislation of the country in which the robot and the control unit are installed.

If the robot or the control unit is partially or completely disposed of, the various parts **must** be grouped together according to their nature (which is all iron together and all plastic together), and disposed of accordingly. These parts **must** also be disposed of in accordance with the current legislation of the country in which the robot and control unit are installed.

Symbol

The following symbol indicates that the product must not be disposed of as common garbage. Handle each product according to local regulations for the respective content.



xx1800000058

For professional users in the European Union

The crossed –out wheeled bin symbol on the product(s) and / or accompanying documents means that used electrical and electronic equipment (WEEE) should not be mixed with general household waste.

If you wish to discard electrical and electronic equipment (EEE), please contact your dealer or supplier for further information.

Disposing of this product correctly will help save valuable resources and prevent any potential negative effects on human health and the environment, which could otherwise arise from inappropriate waste handling.

For disposal in countries outside of the European Union

The crossed –out wheeled bin symbol is only valid in the European Union (EU) and means that used electrical and electronic equipment (WEEE) should not be mixed with general household waste.

If you wish to discard this product please contact your local authorities or dealer and ask for the correct method of disposal.

Disposing of this product correctly will help save valuable resources and prevent any potential negative effects on human health and the environment, which could otherwise arise from inappropriate waste handling.

Continues on next page

Oil and grease

Where possible, arrange for oil and grease to be recycled. Dispose of via an authorized person/contractor in accordance with local regulations. Do not dispose of oil and grease near lakes, ponds, ditches, down drains, or onto soil. Incineration must be carried out under controlled conditions in accordance with local regulations.

Also note that:

- Spills can form a film on water surfaces causing damage to organisms. Oxygen transfer could also be impaired.
- Spillage can penetrate the soil causing ground water contamination.

Hazardous material

The table specifies some of the materials in the FlexLoader function package and their respective use throughout the product.

Dispose of the components properly to prevent health or environmental hazards.

Material	Example application
Batteries, Lithium	Main computer Serial measurement board
Copper	Cables Motors
Steel	Cabinet structure, plates, screws, etc.
Plastic/rubber (PVC)	Cables, connectors, etc.
Aluminium	Heat sinks on power supplies and drive units
Lead	Electronics
Brominated flame retardants	Electronics
Oil, grease	Gearboxes
Neodymium brakes	Motors

This page is intentionally left blank

13 Spare parts

Spare part list is not included in the manual but is delivered as a separate document.

This page is intentionally left blank

14 Diagrams

Circuit diagram

The circuit diagram is not included in this manual, but delivered as separate document.

Pneumatic diagram

The pneumatic diagram is not included in this manual, but delivered as separate document.

This page is intentionally left blank

A Project specific parameters that can be adapted

Some parameters that control the flow of details and function of in- and outconveyors can be adjusted specifically during the teachin in FlexLoader Vision.

Inconveyor parameters	Description
nTimeDelayGrabBelt1	Delay between inconveyor stop and the **InPositionBelt1** signal.
nTimePartAtStartRunBelt1	The maximum time the belt may run when the sensor **diFeeder1InPosition** is affected at belt start.
nTimePartsLowBelt1	The time within which the sensor **diFeeder1PartsLow** must be affected again so that the alarm for filling requirement is not triggered.
nTimeoutBelt1	Maximum continuous time for running a belt with no details on, after which the belt is stopped.
nTimeDelayStopBelt1	Delay between the sensor **diFeeder1InPosition** and the belt stop.
nDisableTooFarFeeder1	If this flag is set to 1, the sensor **diFeeder1TooFar** is ignored.

Outconveyor parameters	Description
nTimeoutEmptyingOutbelt1	Maximum running time for outconveyor 1 during emptying.
nDisableTooFarOutbelt1	If this flag is set to 1, the sensor **diOutFeeder1TooFar** is ignored.
nRunTimeFeederOut	The time which the outconveyor must run in order to allow new details to be put on the outconveyor.

General parameters	Description
nTimeDelayAllowRobotOutbelt1	Delay time for the signal **DOF_AllowRobotLeaveFeederOut1** which takes into account ramp-down time of the belt. Must be lower than ramp-down time.
nTimeDelayMode	Delay time for switching between automatic and manual operation (inconveyor).
nTimeMaxManualForwardBelt1	The longest forward belt running time during manual operation of the inconveyor.
nTimeMaxManualBackwardBelt1	The longest backward belt running time during manual operation of the inconveyor.

These parameters are defined as a user-defined PLC in FlexLoader Vision. Parameters can be changed, renamed, and added. See the FlexLoader Vision manual for more detailed information about user-defined PLC.

This page is intentionally left blank

B Robot I/O

Conveyor system

Name	Description
diFeeder1InPosition	Sensor that indicates that parts are on the cameras field of view
diFeeder1TooFar	Sensor that indicates that parts are at the end of inconveyor
diFeeder1PartsLow	Optional sensor on inconveyor. Used for indication that parts on inconveyor are nearly finished.
diEmergencyStopOk	Emergency stop is not active
diFeeder1ManualForward	Knob for forward operation of inconveyor
diFeeder1ManualBackward	Knob for backward operation of inconveyor
diFeeder1SelectAutomatic	Knob for selecting method of operation (automatic – manual) of inconveyor
diOutFeeder1TooFar	Sensor that indicates that parts are at the end of outconveyor
diFeeder1SelectEmptying	Knob for selecting method of operation (automatic – emptying) of outconveyor
diAlarmU11FeederIN	Alarm from frequency inverter inconveyor
diAlarmU12FeederOUT	Alarm from frequency inverter outconveyor
doRunForwardBelt1	Runs inconveyor forward
doRunBackwardBelt1	Runs inconveyor backward
doManualLampFeeder1	Indication for operator when running inconveyor manually is permitted
doRunFeederOut1	Run outconveyor

Entry control and safety

The following signals are defined for entry and safety related tasks, e.g. operator panels located at gates.

Name	Description
diEntryRequest	Entry request signal from door operator panel
diResetAutoStop	Reset autostop signal from door operator panel
diStartRobot	Start robot signal from door operator panel
doLockGate	Lock door with magnetic holder
doEntryRequestControlLamp	Turn on lamp in entry request button on door operator panel
diAirPressureOK	Signal from air preparation unit that air pressure is ok.

Spare signals

The following signals are defined for general use.

Name	Description
diSpareInterface1	Spare input signal
...	

Continues on next page

Continued

Name	Description
diSpareInterface16	Spare input signal
doSpareInterface1	Spare output signal
...	
doSpareInterface16	Spare output signal

Gripper

Gripper signals are always defined. The definition corresponds to selection of FlexLoader SC 6000 option FLG-111.

Name	Description
diGripperSensor1	Spare signal on robot arm that can be used by integrator
diGripperSensor2	Spare signal on robot arm that can be used by integrator
diGripperSensor3	Spare signal on robot arm that can be used by integrator
diGripperSensor4	Spare signal on robot arm that can be used by integrator
doValve1Gripper	Open gripper 1
doValve2Gripper	Open gripper 2
doValve3Gripper	Spare signal on robot arm connected to valve. Can be used by integrator
doValve4Gripper	Spare signal on robot arm connected to valve. Can be used by integrator
doValve5Gripper	Spare signal on robot arm connected to valve. Can be used by integrator
doValve6Gripper	Spare signal on robot arm connected to valve. Can be used by integrator

Turn station

Name	Description
doTurnStationGripperOpen	Open gripper
doTurnStationRotate	Rotate turn station

Air cleaning box

Name	Description
doValve1AirClean	Activate valve 1 on air cleaning box
doValve2AirClean	Activate valve 2 on air cleaning box
doValve3AirClean	Activate valve 3 on air cleaning box
doValve4AirClean	Activate valve 4 on air cleaning box
doValve5AirClean	Activate valve 5 on air cleaning box
doValve6AirClean	Activate valve 6 on air cleaning box

Continues on next page

Deburring

Name	Description
doValve1Deburr	Activate grinding/deburring tool 1
doValve2Deburr	Activate grinding/deburring tool 1

Marking unit

No digital I/O is used for the marking unit.

Statistical outlet

Name	Description
diDetailInSampleOutletSensor	Sensor that indicates that parts are placed in the statistical outlet.
diSampleOutletInPosition-Sensor	Sensor that indicates that the box is in position and ready to receive parts.

Light signals

The following signals are defined for signaling operation status to the operator.

Name	Description
doRedRGB	Controls the red component of RGB signal light.
doGreenRGB	Controls the green component of RGB signal light.
doBlueRGB	Controls the blue component of RGB signal light.
doVisionLightsOn	Controls the lightning for the vision camera.

This page is intentionally left blank

C Internal communication

FlexLoader Vision <> Robot

For details on communication between FlexLoader Vision and robot refer to the FlexLoader Vision manual.

Marking unit <> Robot

A number of commands are used to control the SIC controller. Note that the example commands below are not representing complete code, neither do they contain necessary handshaking communication with the controller. Consult RAPID code for in-depth information.

```
WriteStrBin ioSicMarker,"LOADFILE " + "MEDIUM" +"\0D\0A";
Response: "LOADFILE OK\0D\0A"
WriteStrBin ioSicMarker,"SETVAR STRING ” + "example text” +"\0D\0A";
Response: "SETVAR OK\0D\0A"
WriteStrBin ioSicMarker,"RUN\0D\0A";
Response: "RUN OK\0D\0A"
WriteStrBin ioSicMarker,"RUN SIMULATION"+"\0D\0A" -> "RUN OK\0D\0A";
Response: "RUN OK\0D\0A"
```

A number of advanced commands can be issued to the marking unit controller. Please refer to the marking units manual for in-depth information.

This page is intentionally left blank

D Pre-configuration settings

Introduction

The FlexLoader SC 6000 is pre-configured when delivered. This section describes some pre-configuration data that might be needed for in-depth technical use of the FlexLoader SC 6000.

Robot pre-configuration

The robot is pre-configured with default code according to the specification.

Serial port setting for communication with the marking unit (optional) are as follows:

Name: COM1, Connector: COM1, Baud rate 9600, Parity: None, Number of bits: 8, Number of stop bits: 1, Flow control: None, Duplex: Full

FlexLoader Vision pre-configuration

FlexLoader Vision is pre-configured with the options that are present in the delivered FlexLoader SC 6000. If needed, this pre-configuration can be changed. See the FlexLoader Vision manual for detailed information on configuration.

Air preparation pre-configuration

The pressure switch and the safety valve are pre-configured. For detailed information, see *Air preparation units on page 160*.

Marking unit (optional) pre-configuration

The marking unit has been pre-configured for use with FlexLoader SC 6000.

There are three pre-configured marking files that are used with FlexLoader Vision (SMALL, MEDIUM, LARGE). These files can - if needed - be changed by the use of the marking units configuration software (se reference section on auxiliary software).

When changing marking files, keep the following information in mind: The part specific marking string is written to the text variable PICKMT, with a constant length that is defined in the FlexLoader Vision setup files.

The serial communication of the SIC controller is set to match the standard setting for COM1 on the robot: Baud rate 9600, Parity: None, Number of bits: 8, Number of stop bits: 1, Only RC: NO, Type: RS232

No further configuration of the SIC controller is necessary. If desired, most advanced options of the marking unit can be used. In certain cases, this would need minor modifications of the RAPID code.

Frequency inverter pre-configuration

The frequency inverter units are pre-configured for use with FlexLoader SC 6000.

See *Frequency inverter on page 187* for parameter handling of these devices.

This page is intentionally left blank

E Frequency inverter

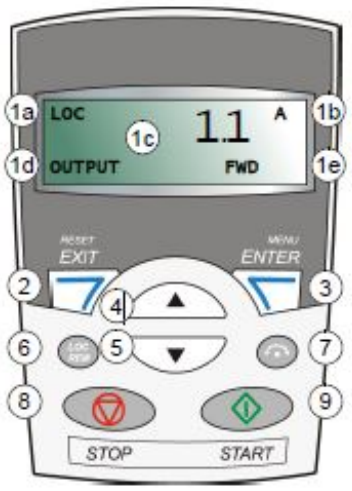
General

This document describes the parameter settings of the frequency inverter ABB ACS355 which is used in the FlexLoader SC 6000.

The document specifies the settings of the essential parameters that are required for tuning the FlexLoader SC 6000. Further optimization and settings can be done, for this refer to User's manual ACS355 user's manual (3AUA0000066143), which can be found on the ABB website.

This manual refers only to the English user interface of the frequency inverter. For other languages, refer to the applicable language editions of the manual on the ABB website.

Overview of the frequency inverter ACS355



xx1800001095

Figure E.1: Overview of the frequency inverter ACS355

1. LCD display – divided into five areas:
   - a. Upper left – Control location:
     - LOC: drive control is local, that is, from the control panel.
     - REM: drive control is remote, such as the drive I/O or fieldbus.
   - b. Upper right – the unit of the displayed value.
   - c. Center – variable. Generally shows parameter and signal values, menus or lists, but can also show error and alarm codes.
   - d. Lower left and center – panel operation state:
     - OUTPUT: Output mode.
     - PAR: Parameter mode.
     - MENU: Main menu.
     - FAULT: Fault mode.

Continues on next page

Continued

- e. Lower right – indicators:
  - FWD (forward) / REV (reverse): direction of the motor rotation.
  - Flashing slowly: stopped.
  - Flashing rapidly: running, not at setpoint.
  - Steady: running, at setpoint.
  - SET: the displayed value can be modified (in the Parameter and Reference modes).

2. RESET/EXIT – Exits to the next higher menu level without saving changed values. Resets faults in the Output and Fault modes.

3. MENU/ENTER – Enters deeper into the menu level. In the Parameter mode, it saves the displayed value as the new setting.

4. Up
   - Scrolls up through a menu or list.
   - Increases a value if a parameter is selected.
   - Increases the reference value in the **Reference** mode.
   - Holding the key down changes the value faster.

5. Down
   - Scrolls down through a menu or list.
   - Decreases a value if a parameter is selected.
   - Decreases the reference value in the **Reference** mode.
   - Holding the key down changes the value faster.

6. LOC/REM – Changes between local and remote control of the drive.

7. DIR – Changes the direction of the motor rotation.

8. STOP – Stops the drive in local control.

9. START – Starts the drive in local control.

Operating the frequency inverter

You operate the frequency inverter with the help of menus and keys. Select an option, e.g. operation mode or parameter, by scrolling the  or  keys until the desired option is visible in the display and confirm your choice by pressing the **Enter**  key. With the **Exit**  key, you return to the previous operation level without saving the changes you have made. The basic control panel has five panel modes: **Output** mode, **Reference** mode, **Parameter** mode, **Copy** mode and **Fault** mode.

By default, the inverter starts in **Output** mode, where you can start, stop, change the direction, switch between local and remote control and monitor up to three actual values (one at a time). To do other tasks, first return to the main menu and select the appropriate mode.

Continues on next page

Parameters

View and change parameters

Step	Action	Display
1	Go to the main menu by pressing if you are in the output mode, otherwise by pressing repeatedly until the display shows **MENU** at the bottom.	LOC rEF MENU FWD
2	If the panel is not in the parameter mode (**PAr** not visible), press or until you see **PAr** and then press . The display shows the number of one of the parameter groups.	LOC PAr MENU FWD
3	Use keys and to find the desired parameter group.	LOC -01- PAR FWD
4	Press . The display shows one of the parameters in the selected group.	LOC 1101 PAR FWD
5	Use and to find the desired parameter.	LOC 1103 PAR FWD
6	Press and hold for about two seconds until the display shows the value of the parameter with **SET** underneath. This indicates that you can now change the value of the parameter. Note: When **SET** is visible, pressing and simultaneously changes the displayed value to the default value of the parameter.	LOC 1 PAR SET FWD
7	Use and to modify the parameter value. When you have changed the parameter value, **SET** starts flashing. To save the displayed parameter value, press . To cancel the new value and keep the original, press .	LOC 2 PAR SET FWD LOC 1103 PAR FWD

Default parameter settings

The following default parameter settings are used for frequency converters (e.g. U11 and U12).

Parameter number	Description	Setting	Value
9901	LANGUAGE	ENGLISH	0
9902	APPLIC MACRO	ALTERNATE	3
9903	MOTOR TYPE	AM	1

Continues on next page

Continued

Parameter number	Description	Setting	Value
9904	MOTOR CTRL MODE	SCALAR:FREQ	3
9905	MOTOR NOM VOLT	EU: 230 V	230
9906	MOTOR NOM CURR		Current according to motor data sign
9907	MOTOR NOM FREQ	EU: 50 Hz USA: 60 Hz	50 60
9908	MOTOR NOM SPEED	1370	1370
9909	MOTOR NOM POWER	0.37 kW	0.37
9910	ID RUN	OFF	0
9914	PHASE INVERSION	NO	0
1401	RELAY OUT 1	FAULT/WARNING	16
1202	CONST SPEED 1	20 Hz	20
1203	CONST SPEED 2	20 Hz	20
2007	MINIMUM SPEED	20 Hz	20
2008	MAXIMUM SPEED	50 Hz	50
2102	STOP FUNCTION	RAMP	2
2103	DC MAGN TIME	0 s	0
2104	DC HOLD CTL	DC BRAKING	2
2104	DC BRAKE TIME	3 s	3
2201	ACC/DEC 1/2 SEL	Not Selected	0
2202	ACCELER TIME 1	0.5 s	0.5
2203	DECELER TIME 1	0.1 s	0.1
3401	SIGNAL 1 PARAM	OUTPUT FREQ	103
3501	SENSOR TYPE	THERM(0)	5
3502	INPUT SELECTION	DI5	7

User adjustable parameter settings

The following parameter settings are typically adjusted by the user.

Parameter number	Description	Default	User value U11	User value U12
1202	CONST SPEED 1	20 Hz		
1203	CONST SPEED 2	20 Hz		
2202	ACCEL TIME 1	0.5 s		
2203	DECELER TIME 1	0.1 s		
9906	MOTOR NOM CURR	Current according to motor data sign		

Continues on next page

Alarms

Error codes are shown on the display in the event of malfunctions. See the table below for the most common error codes. A complete description of alarm and error codes are found in the frequency inverter manual.

The drive can be reset either by pressing , or by turning the supply voltage off for a while. When the problem has been resolved, the motor can be restarted.

Alarm codes

2001 Current limit

2002 Overvoltage

2003 Undervoltage

2009 High temp frequency inverter

Error codes

0001 Current limit exceeded

0002 Overvoltage

0003 High temperature frequency inverter

0004 Short-circuit in motor cables or motor

0006 Undervoltage

0009 High motor temperature

This page is intentionally left blank

F FlexLoader RAPID reference

F.1 FlexLoader data type prefix

Variable prefixes and naming

The following prefixes are used for data types.

- b (bool), e.g. bRobotStandsStill
- btn (btnres), e.g. btnResponse
- c (clock), e.g. clkInPosition
- DOF (signaldo on virtual I/O), e.g. DOF_RunFeeder1
- i (intnum), e.g. iOutletInPosition
- io (iodev), e.g. ioSicMarker
- jt (jointtarget), e.g. jtSoftwareSyncPos
- n (num), e.g. nDetailInGripper1
- p (robtarget), e.g. pViaMachine1
- s (string), e.g. sModuleCam1
- di (signaldi), e.g. diSampleOutletInPositionSensor
- do (signaldo), e.g. doRunFeederOut
- t (tooldata), e.g. tCalibTool1
- v (speeddata), e.g. vLowSpeed
- w (wobjdata), e.g. wCamera1

Several seldom used data types do not obtain their own prefix, e.g. shapedata, errnum, wzstationary. Variables inside records do not use prefixes either.

Variables are named with prefix and a descriptive name, where start of new words is indicated with capital letters (see examples above).

Constant prefixes and naming

Constants do not have any prefix. They are named with capital letters, with words divided by “_”. Examples are LOW_AIR_PRESSURE and STATE_LOAD_MAIN.

F.2 FlexLoader Vision interface

Overview

The FlexLoader Vision interface consists of the following modules.

Module	Description
VisionCom.mod	Background task for base communication with FlexLoader Vision.
Vision.sys	Base communication with FlexLoader Vision and interface towards the application RAPID code.
Calibration3D.sys	Support module used when calibrating FlexLoader Vision with a 3D camera.
VisionSlave.mod	Background task used for external control of FlexLoader Vision by means of a master system.
VisionSlaveControl.sys	Help module loaded in all tasks if this option selected. Makes it much easier to handle the case where robot it self is the master of FlexLoader Vision.
VisionControlImage.sys	Functions for using the original camera to grab a second control image and then return settings to normal again. Loaded in all motion tasks.

VisionCom.mod

This module contains routines needed for background communication between the robot and FlexLoader Vision and must not be changed.

VisionCom synchronizes the vision system and conveyor events with robot program execution. It relieves the foreground task from handling feeding devices and coordinate reception from FlexLoader Vision.

Important routines

Routine	Description
BeltReadyTrap1..4	Handles notifications from the feeder system.
CoordTrap	Receives all coordinate and result information from FlexLoader Vision.

Continues on next page

Vision.sys

This module contains some routines needed for communication between the robot and FlexLoader Vision and must not be changed. It is loaded to all motion tasks and to VisionCom task.

Normal use

Important data

Routine	Description
BELT_ACTION{4}	Must be set according to the desired system behavior for each camera. Normally set from **InitializeCamX** in **ModCamX**. • **RUN_ONE_DETAIL** (default): Run the conveyor belt directly after the last part has been picked. • **RUN_NO_DETAIL**: Run the conveyor belt only if no parts have been found in the image. • **RUN_NEVER**: Never run the conveyor belt. • **RUN_ALWAYS**: Run the conveyor belt after each pick. **Note** When running with the setting **RUN_NEVER**, e.g. when picking from pallets: If FlexLoader Vision does not find anything after taking a new image, the robot must have some type of management for this. The user must check the action sent in the vision result to decide what to do with the result. If action value is equal to 1 (**NO_DETAIL**) the appropriate action must be taken, for example by taking a new image several times before the cell is stopped. If **SetNextTarget** is called up when the action value is 1, the robot program stops.
ALLOW_AUTO_GRAB{4}	Must be set according to the desired system behavior for each camera. Normally set from **InitializeCamX** in **ModCamX**. • **TRUE**: When the belt has been run and the **InPosition** signal set a new image is taken by the automatic device. • **FALSE**: Imaging must be explicitly initiated by the robot program by calling up **GrabImage**.

Important routines

Routine	Description
PROC ConfirmPick1..4()	If **SetNextTarget** returns valid grip information, **ConfirmPick** must be called by the user to run the conveyor and complete the started grab-pick sequence. This usually is done by calling **RefPosOut**. **Note** One **GrabImage** must always be confirmed with **SetNextTarget** and **ConfirmPick** before the next **GrabImage** is started.
PROC DiscardAndTakeNewImage(num grabCamera)	Called up by the user if a new image is to be taken in FlexLoader Vision before all sent coordinates are used. **Note** Routines such as **DiscardAndTakeNewImage()** are to be used with care. If the normal **GrabImage-SetNextTarget-ConfirmPick** chain is deviated from, it is extremely important to have full control of all image taking, to prevent that double images are taken (a new image is taken before the previous image has been processed).

Continues on next page

Routine	Description
PROC DiscardAndStartBelt(num grabCamera)	Called up by the user in order to start the feeder before all sent coordinates are used. **Note** Routines such as **DiscardAndStartBelt()** are to be used with care. If the normal **GrabImage-SetNextTarget-ConfirmPick** chain is deviated from, it is extremely important to have full control of all image taking, to prevent that double images are taken (a new image is taken before the previous image has been processed).
PROC DiscardCoordinates(num grabCamera)	Discards current coordinates which have not been used yet. **Note** Routines such as **DiscardCoordinates()** are to be used with care. If the normal **GrabImage-SetNextTarget-ConfirmPick** chain is deviated from, it is extremely important to have full control of all image taking, to prevent that double images are taken (a new image is taken before the previous image has been processed).
FUNC visionres GetNewVisionResult(num Camera,\num DesiredPosition,\num MaxImageRetries,\num TimeoutTimeVision)	This function waits for coordinates from vision and retrieve the result. Then the result is returned within a visionres data. If nothing found this function will retry a new image or run belt again if there is a belt. **Note** This routine is handling grabbing (but still possible to do it in advance too). It also calls SetNextTarget to receive results and do a first evaluation of what to do. If user wants, this routine could do image retires too.
PROC GrabImage(num grabCamera)	Called up by the user to take a new image with the selected camera. The routine waits until the accompanying signal **InPosition** is 1 and sends a command to FlexLoader Vision to take a new image with **nCamera**. **Note** Do not use **GrabImage** if the camera is set to **ALLOW_AUTO_GRAB = TRUE** (set in **IntitializeMain** in **MainModule**). New images will be requested automatically after **ConfirmPick1..4()**.
PROC InitVision()	Called at the beginning of the program to initialize the communication with FlexLoader Vision.
FUNC num SendPvStatus()	Returns current operating status in FlexLoader Vision (**PV_IDLE**, **PV_OPERATION**, **PV_STARTING**, **PV_STOPPING**).
PROC SetNextTarget(nCamera,\nDesiredPosition)	Called up to retrieve the next valid grip position from FlexLoader Vision for camera **nCamera**. The function returns when the coordinates for a valid part that can be gripped are available. It is possible to specify that only one selected teachin position in FlexLoader Vision is to be returned.
PROC StopVisionSystem()	Stops FlexLoader Vision at the robot's request.

Continues on next page

Routine	Description
FUNC string VisionInputBox()	Can be called up to write an input query on the FlexLoader Vision user interface. The function returns the reply that the operator has entered.
FUNC num VisionMessageBox()	Can be called up to write a message on the FlexLoader Vision user interface and read off the operator's reply (**Ok**, **Yes**, **No**).
PROC WriteLog()	Writes a message directly to the FlexLoader Vision log file.

Further routines

There are several other routines that can be useful. Most of them are found below. Some routines are only help routines and are only called up by other routines in **Vision.sys**. These are not mentioned here.

Routine	Description
PROC CameraContrast(nCamera, nValue)	Called up to change the camera contrast.
PROC CameraExposureTime(nCamera, nValue)	Called up to change the exposure time in the camera.
PROC CameraGain(nCamera, nValue)	Called up to change camera brightness.
FUNC bool checkCoordinates(num pickAreaWidth,num maxMovementHeightUnderCamera,\num pickAreaLength,\robtarget CheckTarget)	This routine checks that received FlexLoader Vision coordinates is within maximum picking area. Also controls that given pick offset is ok.
PROC ClearCoordinates()	For system internal usage.
PROC DisablePosition(nCamera, nPosition)	Called up to deactivate a certain teachin position in FlexLoader Vision. All positions can be locked at the same time by specifying a value **nPosition** > 999.
PROC EnablePosition(nCamera, nPosition)	Called up to activate a certain teachin position in FlexLoader Vision. All positions can be enabled at the same time by specifying a value **nPosition** > 999.
PROC ForceInPosition()	For system internal usage.
FUNC bool isPositionPresent(num cam,num desiredPosition)	Called up to check whether a certain position is among the results from FlexLoader Vision.
FUNC num numPositionPresent(num cam,num desiredPosition)	Called up to check how many times a certain position occurs in the results sent by FlexLoader Vision.
PROC ResetAlarm(sMessage)	Called up to reset a certain alarm that was set in FlexLoader Vision from the robot.
PROC ResetInformation(sMessage)	Called up to reset a specific information message that was set in FlexLoader Vision from the robot.
PROC ResetWarning(sMessage)	Called up to reset a certain warning that was set in FlexLoader Vision from the robot.

Continues on next page

Routine	Description
PROC SendBlackRegion(sBlackRegion)	Sends a reconfiguration to FlexLoader Vision to concentrate the camera's field of view on the selected areas. The user can either make a rectangular area invisible (=black image), or make the area outside a selected rectangle invisible. The areas are specified in the format A-BBB-CCC-DDD-EEE-F, as follows: A indicates camera number (1–4). BBB indicates in percent where the rectangle starts in the x-axis (000–100). CCC indicates in percent where the rectangle ends in the x-axis (000–100). DDD indicates in percent where the rectangle starts in the Y-axis (000–100). EEE indicates in percent where the rectangle ends in the Y-axis (000–100). F indicates whether the actual rectangle becomes black (1) or if the surrounding area becomes black (0).
PROC SendClearSend()	For system internal usage.
PROC SendConfirmPickCam1...4()	For system internal usage.
FUNC bool SelectDetailOnTheFly(nCamera, sGroupName, sDetailName)	For advanced integration only. Used to change part during operation. **Note** Use this functionality with care. Ensure that all other processes are finished before calling SelectDetailOnTheFly, else unpredictable behaviour or system crashes might occur.
PROC SendEdgeHeight(nCamera, nEdgeHeight)	Called up to set a new edge height in the selected camera. Usually used when running stacks on the belt.
PROC SendGrab(nCamera)	For system internal usage.
PROC SendStop()	For system internal usage.
PROC SetAlarm(sMessage)	Called up to set an alarm in FlexLoader Vision from the robot.
PROC SetInformation(sMessage)	Called up to set an information message in FlexLoader Vision from the robot.
PROC SetWarning(sMessage)	Called up to set a warning in FlexLoader Vision from the robot.
PROC StartBelt(nCamera)	Starts the belt under the selected camera. Usually only used from **VisionCom**.
FUNC bool WriteFileVisionPCFlexLoader Vision(sFileName, sMessageData)	Called up to write a text to the selected file name on the FlexLoader Vision computer.

Calibration3D.sys

This module contains routines needed for the calibration of 3D cameras.

Important routines

Routine	Description
PROC Calibration3DRoutine()	This routines performs the actual calibration in cooperation with FlexLoader Vision.
PROC Calibration3DUpdatePositions()	Support routine for convenient update and control of all calibration positions.

Continues on next page

VisionSlave.mod

This module contains routines and data that is used to control FlexLoader Visionthrough robot I/O signals.
Important routines

Routine	Description
TRAP AutoStartTrap	Handles Start requests from external master and starts FlexLoader Vision.
TRAP CycleStopTrap	Handles Stop requests from external master and stops FlexLoader Vision.
PROC main()	Execution start of the task, initiate and then just wait for the traps.
TRAP SelectIdTrap-Cam1..4	Handles part selection requests from external master and changes part in FlexLoader Vision.
PROC Init()	Initializes the various request handlers.

VisionSlaveControl.sys

This module provides some routines that makes it easier for robot to act as master directly to FlexLoader Vision. The routines in this modules connects to VisionSlave functionality to handle this.
Important routines

Routine	Description
PROC VisionControl-Start()	Starts FlexLoader Vision.
PROC VisionControl-Stop()	Stops FlexLoader Vision.
PROC VisionControl-PartSelect(\dnum Part-Camera1|dnum Part-Camera2|dnum Part-Camera3|dnum Part-Camera4)	Changes part selection for a selected camera in FlexLoader Vision. This routine only changes part in one selected camera.
PROC VisionControl-ChangeParts(\dnum PartCamera1,\dnum PartCamera2,\dnum PartCamera3,\dnum PartCamera4)	Changes parts in more than one camera. Internally uses Vision-ControlPartSelect as many times as needed.

Continues on next page

VisionControlImage.sys

This module provides some routines to prepare the standard camera for a second control image. After the control image is grabbed and analyzed, camera is prepared for normal image again. Normal operation is that robot picks from camera area after getting the normal results of where to find part. Then robot send command to prepare camera for second image, moves closer to camera and while holding the part asks to grab the second control image. After the vision result robot evaluates and do necessary actions depending on the result and then prepare the camera for normal operation again.

Important routines

Routine	Description
FUNC visionres ControlImage(num CameraNumber,\num ControlImagePositions{*},\num NormalImagePositions{*},\camerasetting CameraSettingsControl,\camerasetting CameraSettingsNormal,\string BlackRegionValuesControl,\string BlackRegionValuesNormal,\num MaxImageRetries)	This function is the only thing needed to be called from normal user program. It prepares camera for second image, call routine to get robot in position, grabs image, call routine to get the robot out of camera area, prepare for normal image and then returns the result of the second image. **Note** This routine will be depending on two movement routines that must be created inside the part module file. Make sure there is a MoveToControlImageX routine that moves robot to camera X second image position and a MoveFromControlImageX that moves away out of camera X area after image. These instructions should be purely movements. X is the number of the camera which should be used for the second image.
PROC PrepareNormalImage(num CameraNumber,\num PositionsToUse{*},\num InitBeltAction,\bool InitAllowAutoGrab,\camerasetting CameraSettingsNormal,\string BlackRegionValues)	Prepares camera for a normal image. This routine is called from within ControlImage, but could also be used from user program like in the initialization phase.
PROC PrepareControlImage(num CameraNumber,\num PositionsToUse{*},\camerasetting CameraSettingsControl,\string BlackRegionValues)	Prepares camera for the second control image. This routine is called from within ControlImage, but could also be used from user program like in the initialization phase.

F.3 FlexLoader application functionality

Overview

The FlexLoader SC 6000 application functionality handles application and part specific aspects of the robot cell.

The integrator is responsible for adapting this functionality to the actual application.

Module	Description
Main.pgf	Main program automatically loaded by FlexLoader Vision.
MainModule.mod	This module contains the overall and part independent logic for the robot cell. Loaded to all motion tasks.
PartCam1..4.mod	This module provides part-specific application code. This module is further specified for parametrized applications (e.g. FlexLoader Vision Lite) and generic applications. In the case of FlexLoader Vision Lite only one camera is possible and the module for this is named LitePartCam1.mod. Used in all motion tasks.
Common.sys	This module contains general tool data, event routines, world zones, configuration routine and more which is general for all kinds of cell types. Could be used in all motion tasks but might need modifications for robot two and up.
MoveRobotVia.mod	This module handle robot movement between via positions (general movements in the cell). Could be used in all motion tasks but might need modifications for robot two and up.

MainModule.mod

MainModule.mod is always supplied with FlexLoader Vision. It is the main robot program of the system. MainModule.mod provides the following routines:

Routine	Description
PROC CheckSystem()	Checks whether anyone has requested entry to the cell or has been inside the cell. The routine also ensures that new images are taken when the cell is started after someone has been inside. This happens as it can no longer be guaranteed that the part remains under the camera.
PROC ControlImageHandling(num CameraNumber)	Example routine of how to use the ControlImage functionality. This routine should only be used when second control image is needed.
PROC InitializeMain()	Contains the necessary initialization for the main program.
PROC LoadCameraModules()	Loads relevant module files for each camera. The name of each module that is to be loaded is provided by FlexLoader Vision.
PROC Main()	Start of program, a base structure is prepared. Integrator could add or modify if needed.
PROC PickPartAtCamera(num CameraNumber\num DesiredPosition,\num MaxImageRetries,\num TimeoutTimeVision)	Called when the robot is to retrieve a new part by the camera. This routine calls up several sub-routines to request new coordinates and perform movements to and from the pick position.

Continues on next page

Routine	Description
FUNC num PickMultiPartsAtCamera(num CameraNumber,\num NumberOfPartsToPick,\num DesiredPositions{*},\num MaxImageRetries,\num TimeoutTimeVision)	Does the same as PickPartAtCamera with the addition that it also can handle to pick multiple parts from one image.

PartCam1.mod

PartCam1.mod is always supplied with FlexLoader Vision. It handles the part specific programming, e.g. loading, unloading, marking, air cleaning.

Two example modules are delivered, one for operation with simplified TeachIn (FlexLoader Vision Lite) which is named LitePartCam1.mod, the other for standard operation with standard TeachIn. Refer to chapter on robot program.

Standard functionality

Routine	Description
PROC AirCleanDetail()	Template routine for the air cleaning option.
PROC DeburrDetail()	Template routine for the deburring unit option.
PROC Cam1Position_n()	Part specific handling routine for camera 1 depending in which position FlexLoader Vision identified the part. Note, this routine is not always used in the example routines.
PROC InitializeCam1()	Initializes the camera and part specific data.
PROC LeaveFeeder_OUT()	Leaves a part on an out belt.
PROC LeaveSampleOutlet()	Template routine for the statistical outlet option.
PROC LoadMachine()	Loads a part into the machine, multiply routine if several machines.
PROC MainRoutine1()	Handles the main flow of the cell together with main(). This include to check status of machines and so on to figure out what to do next. All these kind of selections and checks that are part specific is placed in this routine. The rest is placed in either of the routines depending on how the integrator wants the structure.
PROC MarkDetail()	Template routine for the marking unit option.
PROC MoveToControlImage()	Moves robot to control image position. Only used if need of second control image.
PROC MoveFromControlImage()	Moves robot from control image position. Only used if need of second control image.
PROC PickCam_1()	This routine performs the actual picking of a part from the conveyor at camera position.
PROC PickMultiPartsCam_1(num PartNo,num Position)	This routine does the same as PickCam_1 but for the case of picking multiple parts from one image.
PROC RefPosInCam_1()	This routine is called to approach an intermediate position towards the conveyor before picking. Update this position to suite the current application.
PROC RefPosOutCam_1()	This is an intermediate position used when leaving the picking area. Update this position to suite the current application.

Continues on next page

Routine	Description
PROC ReGripDetail()	Template routine for the regrip station option.
PROC StopRoutine-Cam1()	This procedure is executed when FlexLoader Vision is stopped.
PROC TurnDetail()	Template routine for the turn station option.
PROC UnloadMachine()	Unloads a part from the machine, multiply routine if several machines.
PROC WashDetail()	Template routine for the washer option.

Additional functionality for FlexLoader Vision Lite use

LitePartCam1.mod for FlexLoader Vision Lite applications contains extended templates and lathe specific RAPID code than allows for a fully parametric teachin.

Specifically, the MainRoutine1 calls getState() to understand how the machine should be handle in the specific case and what to do at the moment. Local routines then cover all allowed combinations for loading and unloading to main and sub spindel, e.g. LoadMachine_MAIN_UnLoad_SUB, UnloadLeftOverPartMAIN, LoadMachine_SUB, UnloadMachine_MAIN_Load_MAIN.

A couple of routines is handling calling actions for selected options. Those might need changes or additions.

Routine	Description
PROC PerformFinishin-gOptions()	Operations that shall be done with the part after unloading from machine tool. Change according application.
PROC PerformPrepare-Options()	Operations that shall be done with the part prior to loading into the machine tool. Change according application.

Machine tool handling procedures and functions

Note that the GLOBAL procedures (like LoadMachine_MAIN) are called from the MainRoutine and perform logical work, whereas the LOCAL routines (like LoadMAIN) perform the actual movement work.

Name	Description
LOCAL PROC EnterMa-chine(\switch Grip-per1FaceMAIN|switch Gripper1FaceSUB)	This procedure is for moving the robot into the machine. Add positions here if necessary. In this example code, pViaMachine1/pViaMachine2 is a position inside the machine where both chucks are reachable and the gripper could turn around without crashing. pViaMachine1 is placed with gripper 1 facing the main chuck. pViaMachine2 is placed with gripper 1 facing the sub chuck.
LOCAL PROC ExitMa-chine(\switch Grip-per1FaceMAIN|switch Gripper1FaceSUB)	This procedure is for moving the robot out of the machine. Add positions here if necessary. pViaMachine1/pViaMachine1 see above. In this example code, pViaMachine is a position just outside the machine tool where the robot can request the machine to close its door.
PROC LoadMa-chine_MAIN()	This procedure is for loading the main chuck in the machine tool.

Continues on next page

Name	Description
LOCAL PROC LoadMAIN()	This procedure loads the part in the main chuck. Make sure to use EnterMachine before all loading/unloading and ExitMachine after all loading/unloading. The load position is always calculated. X and Y is set to 0 and Z is calculated with the help from FlexLoader Vision values. To change the leave position, adjust wMainChuckM1 by running the procedure CalibMainChuck.
PROC SynchroneousMode()	This procedure unloads the sub chuck and then waits for the machine to be ready before loading the main chuck and leaving on the out belt.
LOCAL PROC UnloadLeftOverPart()	This procedure unloads left over part in main chuck. Make sure to use EnterMachine before all loading/unloading and ExitMachine after all loading/unloading. The unload position is always calculated. X and Y is set to 0 and Z is calculated with help from FlexLoader Vision values. To change the unload position, the work object has to be adjusted until the position is correct. In this case adjust wMainChuckM1 by running the procedure CalibMainChuck.
PROC UnloadMachine_LeftOverPart()	This procedure is for unloading the LeftOverPart from the machine tool.
PROC UnLoadMachine_MAIN()	This procedure is for unloading the main chuck in the machine tool.
PROC UnloadMachine_MAIN_Load_MAIN()	This procedure is for first unloading and then loading the main chuck in machine tool.
PROC UnLoadMachine_SUB()	This procedure is for unloading the sub chuck in the machine tool.
LOCAL PROC UnloadMAIN()	This procedure unloads the part in the main chuck. Make sure to use EnterMachine before all loading/unloading and ExitMachine after all loading/unloading. The unload position is always calculated. X and Y is set to 0 and Z is calculated with the help from FlexLoader Vision values. To change the unload position, the work object has to be adjusted until the position is correct. In this case adjust wMainChuckM1 by running the procedure CalibMainChuck.
LOCAL PROC UnloadSUB()	This procedure unloads the part in the sub chuck. Make sure to use EnterMachine before all loading/unloading and ExitMachine after all loading/unloading. The unload position is always calculated. X and Y is set to 0 and Z is calculated with the help from FlexLoader Vision values. To change the unload position, the work object has to be adjusted until the position is correct. In this case adjust wSubChuckM1 by running the procedure CalibSubChuck.
PROC UnLoadMachine_SUB_Load_MAIN()	This procedure is for first unloading the sub spindle and loading the main spindle in machine tool.

Continues on next page

Common.sys

Data definitions, world zones, event routines and other stuff general to all cell types is stored in Common.sys. This system module is accessible by all programs, and by storing the data in this module all programs always have access to the right data. Normally there is no additions to this module. When an integrator has general definitions or other general data, that is preferably stored in CommonCell, CommonSC3000 or CommonSC6000 depending on current cell type.

Routine	Description
PROC AutoConfiguration()	Reads configured IO:s and loaded modules to set cell specific configuration switches.
FUNC bool CheckInPosition(num CameraNumber)	This routine checks if inconveyor is in position, i.e. ready to take a new image. It is only to be used if the cell use a in and doesn't use AUTO_GRAB of images.
PROC ClearToolsPartInfo()	Clears parta data for all tools for current calling robot.
PROC DefForbiddenZone()	Activates a user-defined area where the robot may never be. This must normally be positioned directly above the robot so that it cannot make backward bending movements.
PROC DefSafeZone()	Activates a user-defined area where it is safe for the robot to start its program from the start.
PROC DefWorldZones()	Called up automatically when the robot is switched on. From this routine **DefForbiddenZone**, **DefSafeZone** and **DefMachineZone** are called up.
FUNC mttargetdata getClosestZone()	Finds the closest defined zone to where the robot is located right now.
PROC PowerOnEvent()	This routine is called on power on event. Fill with suitable code if needed in application.
PROC QStopEvent()	This routine is called on quick stop event. Fill with suitable code if needed in application.
PROC RestartEvent()	This routine is called on restart event. Fill with suitable code if needed in application.
PROC StartEvent()	This routine is called on start event. Fill with suitable code if needed in application.
PROC StopEvent()	This routine is called on stop event. Fill with suitable code if needed in application.

CommonSC6000.sys

CommonSC6000.sys is supplied with FlexLoader. It handles data and routines general to this machine type.

Standard functionality

Routine	Description
PROC CellSetup()	Setup of cell specific stuff.
PROC DefMachineZone()	Defines a zone inside machine. When the robot is inside this zone the machine shall not be able to run because of output signal DOF_LoaderOut.
PROC SetSpeed()	Calculate and set correct robot speed depending on weight of part/parts in gripper.

Continues on next page

Routine	Description
PROC SetupTools()	Set up all used tools in the cell.

MoveRobotVia.mod

The principal motion in the FlexLoader SC 6000 is controlled by the MT_MoveRobotTo routine, which moves the robot from known zone positions to a target zone position. From these zone positions (via-positions), movement within certain areas of the robot cell can be initiated. MT_MoveRobotTo is defined within the FlexLoader Library Add-in but need some help from the MoveRobotVia module to manage it's tasks. In this module there are definitions of all zones and the pure movements between them. All other logic around this when to use which movement is within the FlexLoader Library Add-in.

CAUTION

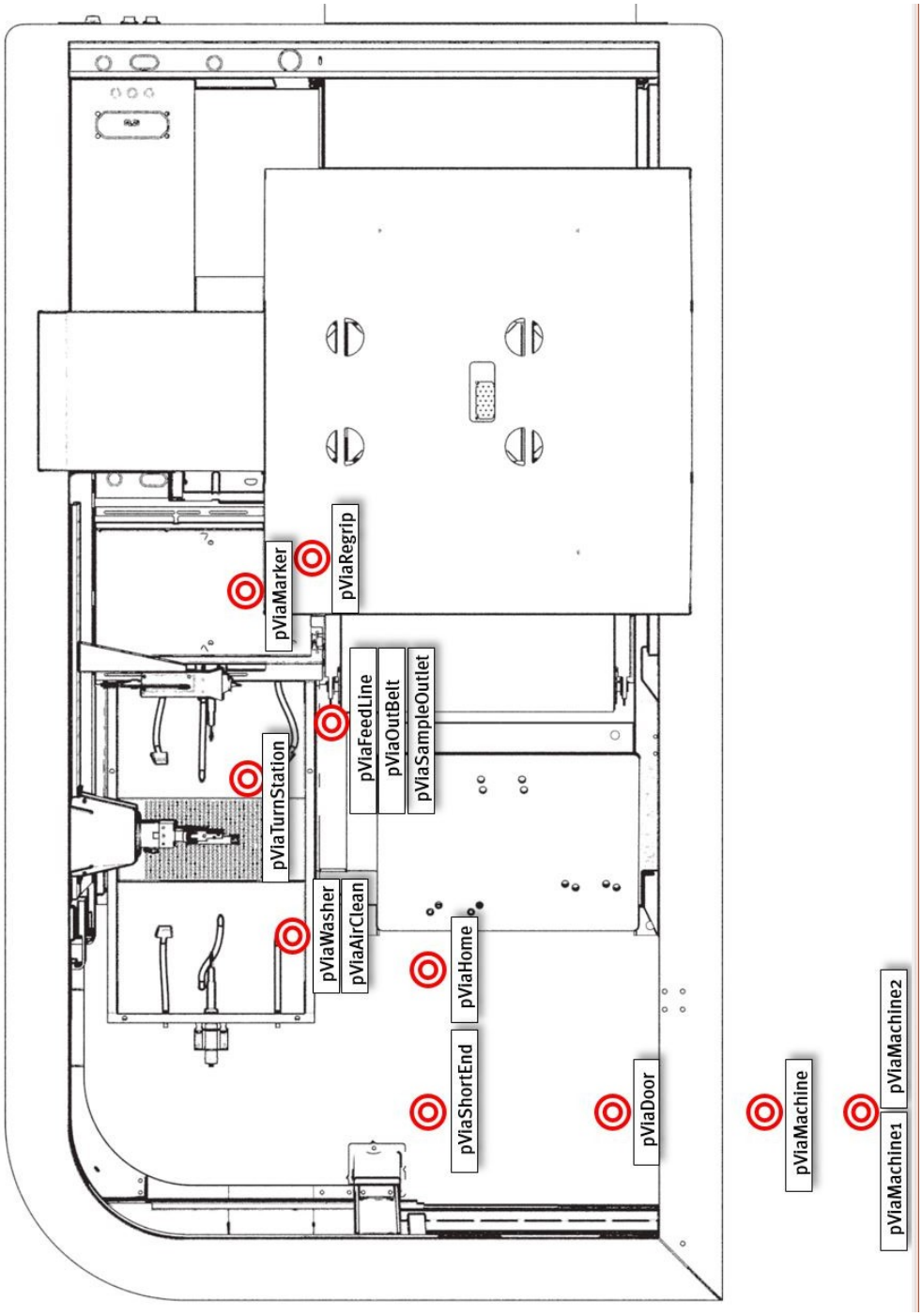
When starting the robot from unknown positions, collisions can occur.

Routine	Description
PROC MoveToZoneMenu()	This will show a menu of all zones and the user has the possibility to select where robot should move.
PROC MT_MoveToVia(mttargetdata Target)	Help routine that moves directly to the via position of the zone passed as argument. User could add own code for special cases, for example if robot is inside a machine, a couple of extra positions might be needed first before moving to via position. Also called from built in FlexLoader Library Add-in routines.
PROC MT_UndefinedZoneHandler()	Help user solve problem if robot is standing in an undefined zone. Also called from built in FlexLoader Library Add-in routines.
PROC mvXXXXXX()	For each defined zone in the cell there must be a mv routine that moves the robot to that zone. Routine name must be exactly mvXXXXX where XXXXX is the name of the defined zone. For example; if mttargetdata ZONE_HOME is defined in the system then there must be a mvZONE_HOME routine in the MoveRobotVia module.
PROC Update_ViaPosition()	Make it easier to teach all via-positions. Only called in manual mode.

A schematic view of the principle motion within the FlexLoader SC 6000 and main positions are shown in the next figure.

Continues on next page

The pre-configured setup of FlexLoader SC 6000 zones is shown in the figure thereafter.

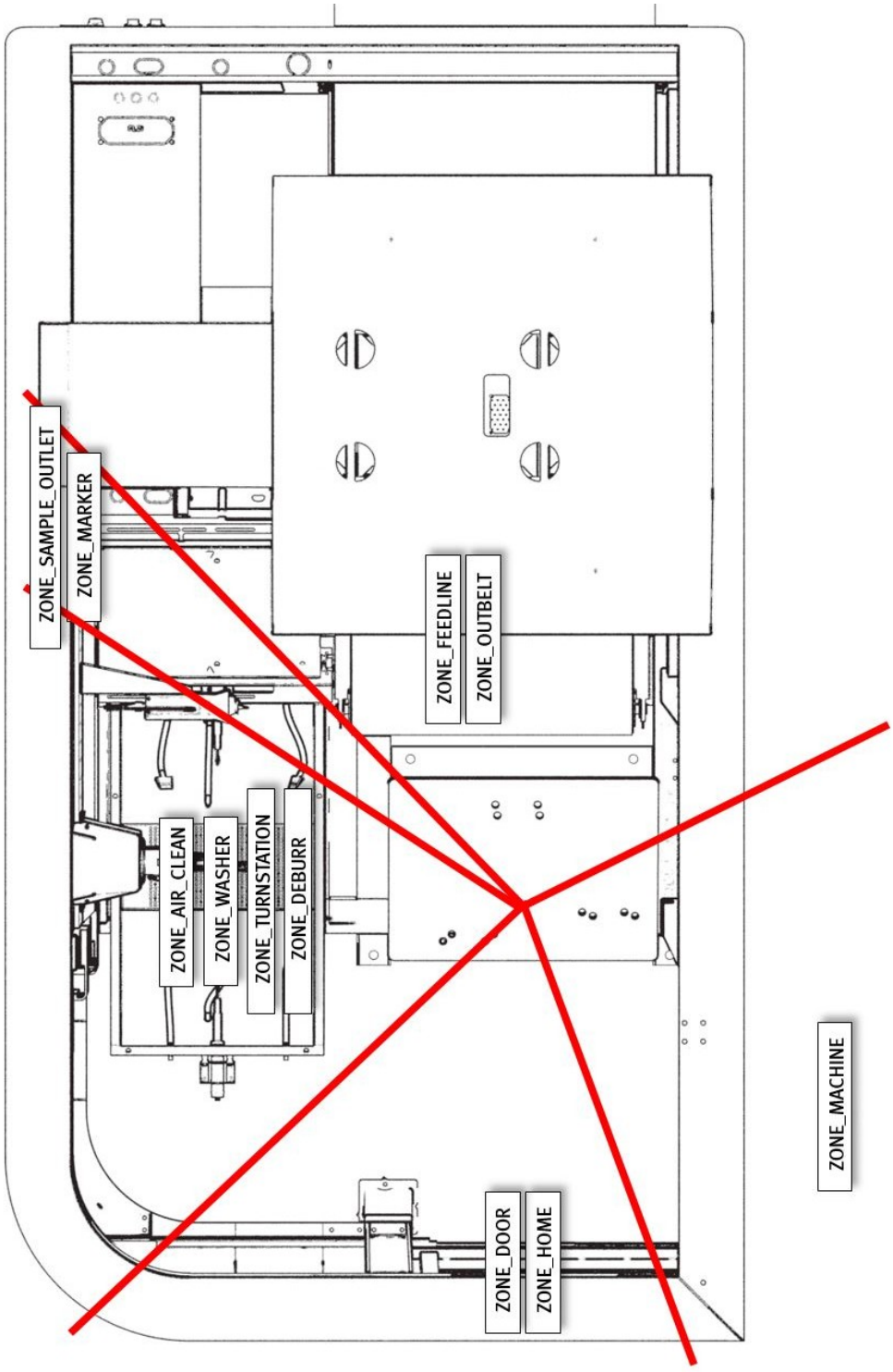


xx1900000400

Schematic motion overview

Continues on next page

F.3 FlexLoader application functionality
Continued



xx1900000409

Schematic zone overview

F.4 FlexLoader Vision Lite functionality

Overview

The FlexLoader Vision Lite uses the part module LitePartCam1 as described earlier. In addition to that there is also an important system module which handles all communication variables that is needed to get correct information from FlexLoader Vision.

Module	Description
FlexLoaderVision-Lite.sys	Contains FlexLoader Vision Lite variables and calculation routines.

FlexLoaderVisionLite.sys

The module FlexLoaderVisionLite.sys contains a number of predefined procedures/functions that enables the parametrized loading and unloading of the machine tool.

The variable names in this module must not be changed, as FlexLoader Vision transfers data to these variables.

Routine	Description
FUNC num calcLoad-Compensation()	Calculate load compensation when leaving and picking from chuck.
PROC CalibLoadCom-pensation()	Calibrates load compensation constants for later use in calcLoad-Compensation.
FUNC pos calcPosLoad-Machine(\num nSpindleOffset)	Calculate load position inside machine.
FUNC pos calcPosUn-loadLeftOverPart(\num nSpindleOffset)	Calculate grip position for left over part.
FUNC pos calcPosUn-loadMachine(\num nSpindleOffset)	Calculate unload position inside machine.
FUNC bool Check-IfLiteBranchValidated()	Checks if current light branch has been tested and validated.
PROC CheckState()	Checks if current state is validated. If not, ask user how to proceed.
FUNC num getAllowed-NumberToStack()	Returns allowed number of details to stack on outbelt.
FUNC num getState()	Checks current machine and cell state. Also checks settings used in part teach in to calculate what to do next. Returns a state which describes next action for robot. This action is then called from the MainRoutine1 to handle the main flow.
PROC ValidateL-iteBranch()	Add current branch to validated array.

F.5 FlexLoader conveyor system control

Overview

The FlexLoader SC 6000 conveyor system control is structured according to the following overviews.

These modules are part of the internal control system and must not be changed.

Module	Description
FeedLine.mod	Overall handling of conveyors and alarms connected to conveyor system. This module is loaded into a separate background task.
FeedLineInPosition.mod	Specialized task for handling start and stop of inconveyor with short response time, regular operation with InPosition-sensor. This module is loaded into a separate background task.
FeedLineToFar.mod	Specialized task for handling start and stop of inconveyor with short response time, regular operation with ToFar-sensor. This module is loaded into a separate background task.
FeedersSuperVision.mod	Specialized task for checking if operator has manually operated any conveyor. This module is loaded into a separate background task.
TpsView files	Optional files for control of conveyor system through the FlexPendant.

F.6 FlexLoader assistance and utility functionality

Overview

The FlexLoader assistance and utility functionality is structured according to the following overview.

Module	Description
AlarmsToVision.sys	Listen to events create by alarm system and sets and resets alarms to FlexLoader Vision when needed. Never touch this code! Loaded as hidden in T_ROB1.
EntryControl.mod	Module for handling entry request, entry indication lamp and door lock signals. Loaded into own background task.
FeederOut.sys	Functionality for control of outconveyor and leave pattern calculation. Could be used in all motion tasks.
GlobalCodeAndConfig.sys	Shared module containing global code for FlexLoader SC 6000 RAPID code.
IndicationLights.mod	Module for handling indicator lights according to operational status. Loaded into own background task.
Indication-Lights_shared.sys	Shared module for some task shared data and routines for indication light handling.
ToolControlMenu.mod	Module for improved and safe handling of gripping functions by the operator. Loaded into own background task.
Tools_base.sys	Module for convenient and common tool handling. Loaded into all motion tasks.
Tools_shared.sys	Shared module for some task shared data and routines for tool handling.

EntryControl.mod

This background task monitors the entry request to the cell. As soon as an entry is requested, a signal is sent to the main program.

This task also ensures that the indication lamp for the entry request lights up/flashes correctly depending on the status of the machine. The lamp starts to flash when entry is requested and stays lit when entry is permitted. When the robot runs in normal operation, the lamp is turned off.

Routine	Description
PROC main()	Initiates entry control handling.

Continues on next page

FeederOut.sys

The module FeederOut.sys provides routines for control of outconveyor and leave pattern calculation.

Routine	Description
FUNC pos CalcFeederOutLeaveOffset(\inout num presentLayer,\num feederNo,\bool leftOverPart)	This procedure is called from user program whenever a part should be placed on outconveyor. User should use the returned offsets for the leave position when leaving part and then call ConfirmLeaveFeederOut when finished. If no confirm, then robot will get the same leave offsets next time this function is called. If using SetUpFeederOutBelt the leave position pLeaveFeederOut is forced to x,y,z = 0,0,0 in this function! Never change that position name.
PROC CalibSpeedFeederOut()	Help procedure for calibrating out belt speed.
PROC ConfirmLeaveFeederOut(\num feederNo)	Call this to confirm that last part has been placed at the last coordinates that was given from CalcFeederOutLeaveOffset. Number of parts left at feeder out is updated and belt is started if current row is full.
FUNC bool hasPermissionToLeaveFeederOut(\num feederNo)	This function checks if robot has permission to leave on feeder out. If selected, it will also check if feeder out has moved and in that case re-initialize the feeder.
PROC RunFeederOut(\num feederNo,\num runLength,\num runTime)	This procedure runs feeder one part length + desired space space between parts. Also resets and initializes variables for feeder.
PROC SetUpFeederOutBasic(\num feederNo,num detailsPerRow,num leaveOffset,num feederRunTime,\bool leaveFromLeft)	Setup restrictions for out feeder belt and part. This procedure should be called once before start using RunFeederOut or CalcFeederOutLeaveOffset. This function is used for basic setup. For advanced setup, use SetUpFeederOutBelt and SetUpFeederOutDetail.
PROC SetUpFeederOutBelt(\num feederNo,num feederWidth,num feederSpeed,num spaceBetweenParts,num spaceToEdge,num yDistanceToLeave,\bool leaveFromLeft,\string RunFeederDOSignalName,\string AllowLeaveDOSignalName,\string FeederHasMovedDOSignalName,\string BlockEmptyingModeSignalName)	Setup restrictions for out feeder belt. This procedure should be called once before start using RunFeederOut or CalcFeederOutLeaveOffset.

Continues on next page

Routine	Description
PROC SetUpFeederOutDetail(\num feederNo,num detailWidth,num detailLength,num heightToTCP,\num allowedDetailsInHeight,\num detailHeight, \num xDistanceToTCP,\bool hasLeftOverPart,\num leftOverPartWidth,\num leftOverPartLength,\num leftOverPartHeightToTCP)	Setup restrictions for out feeder part. This procedure should be called once before start using RunFeederOut or CalcFeederOutLeaveOffset. The routine also calculates how many parts fit in each row and if a left over part space should be reserved.

GlobalCodeAndConfig.sys

This is a module containing global support functions that could be used in any machine. It also includes configuration switches and global standard constant declarations.

CAUTION

Always make sure configuration switches are set to values matching the current cell. But also note that most of those switches is automatically set by AutoConfiguration().

Routine	Description
FUNC string AddTrailingBlanks(string Text,num WantedTextLength)	Returns the string Text but with added blanks to get to WantedTextLength length.
FUNC num calcInchToMM()	Calculates mm value from Inch.
FUNC num calcLbToKg()	Calculates Kg value from Lb.
FUNC tooldata calcToolData(robtarget pickPos,wobjdata pickWorkObject,\num RobotNumber)	This function returns a new tooldata updated with a TCP matching position of currently found object in FlexLoader Vision. Use this procedure first before teaching or using positions that need a TCP that is attached to the FlexLoader Vision position. Use with caution.
FUNC num getCycleTime(num nArrayIndex,\num nNumberOfCycleTimes,\num nPartsPerCycle)	Returns cycle time in seconds as a num. If no valid cycle time calculated, return value will be -1. It also adds cycle time of the last cycle to the array.
FUNC dionum getDISignalState(string SignalName)	Reads value of a digital input signal passed to function as string, returns 1 if signal is high, else 0 will returned.
FUNC dionum getDOSignalState(string SignalName)	Reads value of a digital output signal, returns 1 if signal is high, else 0 will returned.
FUNC bool getFlashingHz(num Hz)	This function returns true or false depending on a timer and a wanted frequency.

Continues on next page

Routine	Description
FUNC string getModuleName(string sModulePath\switch WithFileExtension)	This function returns the module name from a search path string.
FUNC num getPartCount(num nArrayIndex\num nPartsPerCount)	Help to count parts in cell.
FUNC num getRobotNumber()	Returns robot number of the robot where calling module is executing.
FUNC bool isDistanceOK(\num RobotNumber,\tooldata checkTool,robtarget checkPosition,wobjdata workObject,num maxDistance\switch PrintDistance)	Function that checks if a robot with its current tool is close to the specified robtarget in the specified workobject.
FUNC num maxNumValue(num Value1,num Value2)	Returns highest value of two num variables.
FUNC num minNumValue(num Value1,num Value2)	Returns lowest value of two num variables.
PROC ResetCycleTimes(num nArrayIndex)	Reset the cycle time calculation.
PROC ResetPartCount(num nArrayIndex)	Reset part counter.
PROC SetConfigurationVariable(INOUT bool ConfiguratioValue,string SignalName,\switch OutputSignal|switch InputSignal)	Sets a configuration value depending on if passed IO exists in system or not.
PROC SetDOSignal(string SignalName,dionum Value)	Set a digital output signal with signal name passed as a string.
FUNC tooldata ToolOffset(PERS tooldata ToolToAdjust,\num tOffsX,\num tOffsY,\num tOffsZ)	Returns tooldata value with offsets according to optional arguments. If no optional arguments is used Return value is the same as input value. Example: `tTempDeburr:=ToolOffset(tDeburrOriginal\tOffsX:=0\tOffsY:=0\tOffsZ:=2);`
PROC WriteActiveAlarmsToFile()	Writes all active alarms in system to a file.

IndicationLights.mod

The background task with module IndicationLights.mod, handles the indicator lights according to operational status.

Routine	Description
PROC main()	Main loop for task.

Continues on next page

Routine	Description
PROC BlueLight()	Sets blue light in RGB.
PROC FlashFunction()	Toggle values for flash memories.
PROC GreenLight()	Sets green light in RGB.
PROC HandleStandardTower()	Handle main logic for a standard light tower. A standard tower has 1-5 separate lamps with single color.
PROC HandleRGBTower()	Handle main logic for a RGB light tower. RGB tower has one lamp possible to handle red, green and blue and also to mix those.
PROC NoLight()	Sets no light in RGB.
PROC RedLight()	Sets red light in RGB.
PROC UpdateLocalAlarmStatus()	Update local bits for sum alarm, warning, information and questions
PROC VisionLightsOnControl()	Controls and checks if lights would be switched on or off.
PROC WhiteLight()	Sets white light in RGB.
PROC YellowLight()	Sets yellow light in RGB.

IndicationLights_shared.sys

The shared module IndicationLights_shared.sys handles task general data and routines for indication lights.

Routine	Description
PROC IndicationLightsSetupLightTower()	Assigns correct color to each message type.
PROC IndicationLightsSetupRGBLights()	Assign correct color to each message type.

ToolControlMenu.mod

The module ToolControlMenu.mod handles the FlexPendant menu for manual control of tool signals.

This module depends on the use of Tools_shared.sys to declare all tools in the system. As long as this is done, no further input is needed in ToolControlMenu.mod.

The tool menu is activated by the signal DOF_ToolMenuButton. For convenience, this signal can be connected to a function key.

Routine	Description
TRAP ToolControlMenuTrap	This trap displays menu for control of Tools. Normally connected to function key 1 on TeachPendant.

Tools_base.sys

The module Tools_base.sys handles tool actions and supports with some tool state functions to check state of parts held in grippers etc.

Routine	Description
FUNC bool checkAllToolsEmpty()	Returns TRUE if all robot mounted tools are empty.

Continues on next page

Routine	Description
FUNC mtpartstate GetToolPartState(dnum Tool)	Help function to slim down arguments when finding out part state of the part in robot tool.
PROC GripLoadUpdate(dnum ActiveTool)	Search for all robot tools and what they are holding. Calculates a total loaddata sum for all tools that holds something. Calls GripLoad with the calculated total load.
PROC PickPartAt(mtstationdata FromStation,\num TrackerIndex,dnum PickTool,\bool InsideGrip)	Modifies tool signal to pick a part from FromStation. Then move part tracker data from FromStation to PickTool.
PROC PlacePartAt(mtstationdata ToStation,\num TrackerIndex,dnum PlaceTool,\bool InsideGrip)	Modify tool signal to release a part from tool PlaceTool. Then move part tracker data from PlaceTool to ToStation.

Tools_shared.sys

The module Tools_shared.sys handles tool definitions and a lot of tool action and control routines. A lot of the routines is used by the Tools_base.sys module.

Routine	Description
PROC AddToSumGripLoad(INOUT loaddata pSumLoad,tooldata ActiveTool,loaddata ThisLoad,tooldata ThisTool)	Help function to calculate a total load from all loads currently held by robot.
FUNC bool CheckValidTool(\num RobotNumber,dnum ToolNumber)	Checks if a specific tool is defined in system.
PROC ClearToolData(\bool RobHold,\num RobotNumber,\dnum ToolNumber)	Clears tooldefinition data. If using optional argument, only remove that specific data, else remove all.
FUNC num getCurrentTotalLoadMass(\num RobotNumber)	Returns total load mass carried in specific robot tools.
FUNC tooldefinition getToolDefinitionData(\bool RobHold,\num RobotNumber,dnum ToolNumber)	Returns the tooldefinition data for a given tool number.
FUNC tooldata getToolData(\bool RobHold,\num RobotNumber,dnum ToolNumber)	Returns the tooldata for a given tool number.
FUNC string getToolStatus(\bool RobHold,\num RobotNumber,dnum Tool)	Returns current tool status label. Depending on tool current status return corresponding text label from the tool declaration.

Continues on next page

Routine	Description
PROC ToolAction(\bool RobHold,\num RobotNumber,\switch Open|switch Release|switch On|switch Close|switch Grip|switch Off,dnum Tool,\bool InsideGrip,\num WaitTimeOverride,\switch NoManualConfirmation)	Perform action with a robot tool. Actions could be Open, On, Close or Off. Uses signal data and wait times declared through ToolSetup.
PROC ToolSetup(\bool RobHold,\num RobotNumber,dnum ToolNumber,string Label,tooldata Tool_Data,mtstationdata Station,\num StnIndex,\string OpenOrOnSignal,\num WaitTimeOpenOrOn,\string LabelOpenOrOn,\string CloseOrOffSignal,\num WaitTimeCloseOrOff,\string LabelCloseOrOff,\string BlockManualControl)	Setup a new tool in system. Stores all values needed for the tool.
PROC ToolUpdate(\bool RobHold,\num RobotNumber,dnum ToolNumber,\string Label,\tooldata Tool_Data,\mtstationdata Station,\num StnIndex,\string OpenOrOnSignal,\num WaitTimeOpenOrOn,\string LabelOpenOrOn,\string CloseOrOffSignal,\num WaitTimeCloseOrOff,\string LabelCloseOrOff,\string BlockManualControl)	Updates tooldefinition data. Finds the robot tool (through it's ToolNumber) and updates all data for that tool which is passed in the arguments. Data which is not passed as argument will not be touched.

F.7 FlexLoader machine tool interface functionality

Overview

The FlexLoader SC 6000 application functionality handles machine tool and part specific aspects of the FlexLoader SC 6000. The integrator is fully responsible for adopting this functionality to the actual application. Described below is two different template modules. Those describes which routines and data that is needed to communicate with a machine. All real machine signals is only handled within this machine module. This makes it easier to change machine or reuse an earlier made machine module. This means that in a real project, the template module should be switched out to a module that follow the same structure but are prepared for the correct machine used in the project. This could be done by rewriting the template module or replacing it by a pre-made module.

Module	Description
TemplateMachine_CNC.sys	Provides bridge between logical functions within the FlexLoader SC 6000 application and physical interface for up to two CNC machine tools. It also includes code to handle indexing table within both machines.
TemplateMachine_Lathe.sys	Provides bridge between logical functions within the FlexLoader SC 6000 application and physical interface to a lathe machine tool.

Note

If more machine tools is used than what is handled by the template, then the functionality can be enhanced by the integrator.

TemplateMachine_CNC.sys

This module contains a number of predefined procedures or functions for communication with the machine tool:

Name	Description
FUNC mtpartstate getMachine1PartType()	Intended use is for index table machines to keep track of parts in machine 1.
FUNC mtpartstate getMachine2PartType()	Intended use is for index table machines to keep track of parts in machine 2.
PROC InitMachine1()	Initiates what is necessary for machine 1 handling. NEEDED: Set up correct signals for interrupt handling.
PROC InitMachine2()	Initiates what is necessary for machine 2 handling. NEEDED: Set up correct signals for interrupt handling.
PROC Machine1Action(machineaction Action)	Performs desired machine 1 tool action. NEEDED: Define signals and communicate with machine tool.
FUNC bool Machine1Check(machinecheck Check,\INOUT bool Result)	Checks machine 1 tool and conditions and returns correct state for use in main flow of program. NEEDED: Define signals and communicate with machine tool.

Continues on next page

Name	Description
FUNC bool Machine1Wait(machinecheck Wait,num MaxTime)	Waits for machine 1 condition to be ok within **MaxTime**. Returns false if **MaxTime** elapses. NEEDED: Define signals and communicate with machine tool.
PROC Machine1Action(machineaction Action)	Performs desired machine 2 tool action. NEEDED: Define signals and communicate with machine tool.
FUNC bool Machine1Check(machinecheck Check,\INOUT bool Result)	Checks machine 2 tool and conditions and returns correct state for use in main flow of program. NEEDED: Define signals and communicate with machine tool.
FUNC bool Machine1Wait(machinecheck Wait,num MaxTime)	Waits for machine 2 condition to be ok within **MaxTime**. Returns false if **MaxTime** elapses. NEEDED: Define signals and communicate with machine tool.

DANGER

Ensure that the standard features in **MachineXAction**, **MachineXCheck** and **MachineXWait** are correctly implemented (where X is machine number). Malfunction and collisions might otherwise occur.

Note

We recommend the use of an additional background task if the robot cell requires time critical machine tool communication.

Note

Operation with more than two machines requires the creation of these routines with equivalent names, e.g. Machine3Action and Machine4Wait.

TemplateMachine_Lathe.sys

This module contains a number of predefined traps for information transfer with machine tool.

Name	Description
TRAP DoorClosedTrap	Used on machine tools with slow doors. Signal is caught by trap instead of foreground program so that the robot can continue working. Confirms to the machine tool that a door closed signal has been received and the whole loading sequence is finished.
TRAP DoorNotOpenedTrap	Can be used for special conditions or checks on certain machine tools.
TRAP PrePickTrap	Receives information from machine tool that a part soon needs to be loaded. If needed, add code to confirm to the machine tool that a signal has been received.

Continues on next page

This module contains a number of predefined procedures or functions for communication with the machine tool:

Name	Description
PROC CalibMainChuck()	Help procedure for calibrating chuck at main spindle.
PROC CalibSubChuck()	Help procedure for calibrating chuck at sub spindle.
PROC InitMachine1()	Initiates what is necessary for machine handling. Sets up traps. NEEDED: Set up correct signals for interrupt handling.
PROC Machine1Action(machineaction Action)	Performs desired machine tool action. NEEDED: Define signals and communicate with machine tool.
FUNC bool Machine1Check(machinecheck Check,\INOUT bool Result)	Checks machine tool and conditions and returns correct state for use in main flow of program. NEEDED: Define signals and communicate with machine tool.
FUNC bool Machine1Wait(machinecheck sWait,num nMaxTime)	Waits for machine condition to be ok within **nMaxTime**. Returns false if **nMaxTime** elapses. NEEDED: Define signals and communicate with machine tool.

DANGER

Ensure that the standard features in **Machine1Action**, **Machine1Check** and **Machine1Wait** are correctly implemented. Malfunction and collisions might otherwise occur.

Note

We recommend the use of an additional background task if the robot cell requires time critical machine tool communication.

Note

Operation with more than one machine requires the creation of these routines with equivalent names, e.g. Machine2Action and Machine3Wait.

States and actions

The machine modules uses a number of predefined states and actions that are consistently used. If needed, new values can be added in GlobalCodeAndConfig.sys. However, do not remove the existing values, as these are used in the FlexLoader core functionality.

Examples for predefined values are:

- Examples for predefined machine checks/waits:
  DOOR_OPENED, PREPICK_OK, LOAD_SUB_OK, LOAD_MAIN_OK, UNLOAD_SUB_OK, UNLOAD_MAIN_OK, HOME_POSITION_OK, MAIN_CHUCK_OPENED, MAIN_CHUCK_CLOSED, SUB_CHUCK_OPENED, SUB_CHUCK_CLOSED
- Examples for predefined machine actions:

Continues on next page

CLOSE_CHUCK_MAIN, CLOSE_CHUCK_SUB, CLOSE_DOOR, OPEN_CHUCK_MAIN, OPEN_CHUCK_SUB, OPEN_DOOR, PREPARE_LOAD_MAIN, PREPARE_LOAD_SUB, PREPARE_UNLOAD_MAIN, PREPARE_UNLOAD_SUB, CYCLE_START, INIT_MACHINE, STOP_MACHINE

There are further predefined values in GlobalCodeAndConfig.sys that can be implemented in machine communication modules.

F.8 FlexLoader options functionality

Overview

The FlexLoader options functionality is structured according to the following overview.

Module	Description
AirClean.sys	Contains helper functions for setting up and controlling the air cleaning unit. Could be loaded in all motion tasks but currently only prepared for T_ROB1.
MarkingUnit.sys	Contains helper functions for setting up and controlling the marking unit. Could be loaded in all motion tasks but currently only prepared for T_ROB1.
MarkingUnit_shared.sys	Contains some task general data definitions. Could be loaded in all motion tasks but currently only prepared for T_ROB1.
StatisticalOutlet.mod	Handles the user interaction for statistical outlet, for example requesting sample part, aborting sample. Loaded into own background task.
StatisticalOutlet.sys	Contains helper functions for setting up and controlling the statistical outlet. Loaded into T_ROB1 task.
StatisticalOutlet_shared.sys	Shared module containing some task general data definitions.
TurnStation.sys	Contains a couple of data definitions used for turn station. Could be loaded in all motion tasks but currently only prepared for T_ROB1.
Washer.sys	Contains helper functions for setting up and controlling the washer unit. Could be loaded in all motion tasks but currently only prepared for T_ROB1.
SafeMove.sys	Contains helper function for cyclic break check. Could be loaded in all motion tasks but currently only prepared for T_ROB1.

AirClean.sys

The module AirClean.sys handles functionality related to the air cleaning unit.

Routine	Description
PROC CalibAirClean()	Help procedure for calibrating air cleaning unit.
PROC StartAirClean(num nNozzle)	This procedure starts air through selected nozzle.
PROC StopAirClean(num nNozzle)	This procedure stops air through selected nozzle.

MarkingUnit.sys

The module MarkingUnit.sys handles functionality related to the marking unit.

Routine	Description
PROC CalibMarker()	Help procedure for calibrating marking unit.
PROC SetMarker()	Initiates marking unit, loads marking file and sets correct marking string.
PROC StartMarker()	Start marking of a part.

Continues on next page

StatisticalOutlet.mod

The module StatisticalOutlet.mod handles user interaction functionality related to the statistical outlet function.

Routine	Description
PROC main()	Handles functions for user interaction, creating sample part orders, setting user information and so on.
TRAP OutletInPosition-Trap	Tests if detail is in statistical outlet when closed. Resets warning in this case.

StatisticalOutlet.sys

The module StatisticalOutlet.sys handles functionality related to the statistical outlet function.

Routine	Description
PROC CalibSampleOut-let()	Help procedure for calibrating statistical outlet.
FUNC bool is-ThisASamplePart()	This function checks current part is a sample part.
PROC LeaveSamplePar-tIfNeeded()	Check if needed to leave sample part and call routine to do so if true.
PROC PrepareLeaveIn-StatisticalOutlet()	Prepares everything for robot to leave in statistical outlet.
PROC PrepareSample-PartRemoval()	Make sure robot stops and then tells operator to fetch part from statistical outlet.

Washer.sys

The module Washer.sys handles functionality related to the washing unit.

Routine	Description
PROC CalibWasher()	Help procedure for calibrating washer unit.

SafeMove.sys

The module SafeMove.sys handles functionality related to SafeMove.

Routine	Description
PROC CheckCBC()	This procedure is executed to check if cyclic brake check is needed. Brake check is performed automatically.

This page is intentionally left blank

G FlexLoader Library Add-in reference

G.1 Instructions

G.1.1 MT_ClearAllPartInfo - Clears all part tracker information

Usage

`MT_ClearAllPartInfo` is used to clear all part tracker information.

Basic examples

The following example illustrates the instruction `MT_ClearAllPartInfo`.

Example 1

```
MT_ClearAllPartInfo;
```

This will clear all part info in the all trackers defined in the program.

G.1.2 MT_ClearPartInfo - Resets part information of a part tracker

Usage

`MT_ClearPartInfo` is used to reset the complete part information in a specific part tracker of a station.

Basic examples

The following example illustrates the instruction .

Example 1

```
PERS mtstationdata sdRobot;
CONST num GRIPPER_1:=1;
PROC TestProc()
  MT_ClearPartInfo\Station:=sdRobot,\Index:=GRIPPER_1;
ENDPROC
```

This will clear all part info in the first part tracker of the robot station. For a better readability the GRIPPER_1 was assigned value 1.

Arguments

```
MT_ClearPartInfo [\Station] | [\StationName] [\Index]
```

`\Station`

Data type: `mtstationdata`

The station data which part tracker data should be cleared in.

`\StationName`

Data type: `string`

Name of the station data declaration as a string. Through this name the correct mtstationdata definition will be found and the corresponding part tracker where part data should be cleared.

`\Index`

Data type: `num`

Index number of the part tracker to clear. Note, each station can have up to 20 trackers. If argument not passsed, index number 1 is used.

G.1.3 MT_CopyPartInfo - Copies part info between trackers

Usage

`MT_CopyPartInfo` is used to copy part tracker information from one part tracker to another.

Basic examples

The following example illustrates the instruction `MT_CopyPartInfo`.

Example 1

```
PERS mtstationdata sdRobot;
PERS mtstationdata sdMachine;
PERS num GRIPPER_1:=1;
PROC TestProc()
MT_CopyPartInfo
     \FromStation:=sdRobot,\FromIndex:=GRIPPER_1,\ToStation:=sdMachine;
ENDPROC
```

This call will copy part information from the part tracker connected to the robot station with index 1 (corresponds to gripper 1 in this case). The data will be copied (Note, not moved!) to the part tracker of the machine, sdMachine. Since this machine only have one tracker the index was left out (means that index 1 will be used internally).

Arguments

```
MT_CopyPartInfo [\FromStation] | [\FromStationName] [\FromIndex]
     [\ToStation] | [\ToStationName] [\ToIndex]
```

`\FromStation`

Data type: `mtstationdata`

The station to copy part tracker data from.

`\FromStationName`

Data type: `string`

The station to copy part tracker data from. In this argument, the name of the mtstationdata definition is passed as a string.

`\FromIndex`

Data type: `num`

Specifies which part tracker index in FromStation to fetch part info from.

`\ToStation`

Data type: `mtstationdata`

The station to copy part tracker data to.

`\ToStationName`

Data type: `string`

The station to copy part tracker data to. In this argument, the name of the mtstationdata definition is passed as a string.

Continues on next page

G.1.3 MT_CopyPartInfo - Copies part info between trackers
RobotWare - OS
Continued

`\ToIndex`

Data type: `num`

Specifies which part tracker index in ToStation to copy part info to.

G.1.4 MT_CreateAlarm - Creates new alarm or message

Usage

Normally an alarm is defined by adding a new row in a csv file, see general description of Alarms&Messages. But when integrator is adding stuff later this procedure could be used instead. This instruction adds a new alarm/message definition to the system.

Basic examples

The following example illustrates the instruction `MT_CreateAlarm`. This will define a new alarm with identifier “myAlarm” and category Error.

Example 1

```
MT_CreateAlarm "myAlarm","Header text",["body text 1","body text
    2","body text 3"]\Category:=iconError,\ErrorNumber:=123;
```

Arguments

```
MT_CreateAlarm Identifier, Header, Text{*}, [\Category],
    [\ErrorNumber], [\Domain]
```

`Identifier`

Data type: `string`

System unique string that gives the possibility to point to a specific message.

`Header`

Data type: `string`

Header text for the message.

`Text{*}`

Data type: `string array`

Up to 8 strings could be passed in this message body text.

`\Category`

Data type: `icondata`

Defines the category of an alarm. If not used this will be normal message text.

`\ErrorNumber`

Data type: `num`

Defines error number for the message if needed.

`\Domain`

Data type: `num`

Defines domain number for the message if needed.

G.1.5 MT_ErrWrite - Creates error log message in local language

Usage

`MT_ErrWrite` is used to write a message to the robot event log.

Basic examples

The following example illustrates the instruction `MT_ErrWrite`.

This will write the message which match to identifier “myAlarm” to the robot event log. Depending on defined category it will show in event log with matching category. If category is not used for the message it will appear in the event log as an error type. The text will appear with wildcard {1} replaced with “AddText”.

Example 1

```
PROC ErrWriteExample()
  VAR mtmsgdata Alarm;
  Alarm:=MT_GetMessage("myAlarm");
  MT_ErrWrite Alarm,\WildCardTexts:=["AddText"];
ENDPROC
```

Arguments

```
MT_ErrWrite Message, [\WildCardTexts{*}]
```

`Message`

Data type: `mtmsgdata`

The message data that should be used to write message to error log.

`WildCardTexts{*}`

Data type: `string array`

String array of wild card text to replace wild cards in message texts.

G.1.6 MT_GetPartInfo - Gets any kind of part tracker data from a part tracker

Usage

`MT_GetPartInfo` is used to read any type of part data information from a specific part tracker.

Basic examples

The following example illustrates the instruction `MT_GetPartInfo`.

Example 1

```
VAR mtpartstate tempState;
VAR string tempString;
MT_GetPartInfo
     \Station:=sdRobot,\Index:=GRIPPER_1,\State:=tempState,\UserDef:=tempString;
```

This code will get the current state and user defined string for the part currently in robot gripper 1 (Robot station index 1). The data will be stored in tempState and tempString.

Arguments

```
MT_GetPartInfo [\Station] | [\StationName] [\Index] [\ProgCode]
     [\TypeCode] [\AuxCode] [\ToolCode] [\MachineCode] [\Routine]
     [\Part] [\Tool] [\PLoad] [\State] [\UserDef]
```

`\Station`

Data type: `mtstationdata`

The station to fetch part tracker data from.

`\StationName`

Data type: `string`

The station to fetch part tracker data from. In this argument name of the mtstationdata definition is passed as a string.

`\Index`

Data type: `num`

Specifies which part tracker index to fetch part info from.

`\ProgCode`

Data type: `dnum`

Stores ProgCode into this INOUT argument. See mtpartdata.

`\TypeCode`

Data type: `dnum`

Stores TypeCode into this INOUT argument. See mtpartdata.

`\AuxCode`

Data type: `dnum`

Stores AuxCode into this INOUT argument. See mtpartdata.

Continues on next page

G.1.6 MT_GetPartInfo - Gets any kind of part tracker data from a part tracker

RobotWare - OS
Continued

`\ToolCode`

Data type: `num`

Stores ToolCode into this INOUT argument. See mtpartdata.

`\MachineCode`

Data type: `dnum`

Stores MachineCode into this INOUT argument. See mtpartdata.

`\Routine`

Data type: `string`

Stores Routine into this INOUT argument. See mtpartdata.

`\Part`

Data type: `mtpartdata`

Stores Part into this INOUT argument.

`\Tool`

Data type: `tooldata`

Stores Tool into this INOUT argument.

`\PLoad`

Data type: `loaddata`

Stores PLoad into this INOUT argument.

`\State`

Data type: `mtpartstate`

Stores State into this INOUT argument. See mtparttracker.

`\UserDef`

Data type: `string`

Stores UserDef into this INOUT argument. See mtparttracker.

G.1.7 MT_GoHome - Moves robot to home zone

Usage

`MT_GoHome` is used to move robot to home zone.

Basic examples

The following example illustrates the instruction `MT_GoHome`.

Example 1

```
MT_GoHome;
```

Moves robot to position in the mttargetdata ZONE_HOME.

G.1.8 MT_InitializeMessageHandling - Initiates the alarms and messaging module

Usage

`MT_InitializeMessageHandline` is used to initiate the alarms and messaging modules. The core task is to check currently used language and make sure corresponding user texts is loaded in the system.

Basic examples

The following example illustrates the instruction `MT_InitializeMessageHandling`. This call will force new loading of all texts from csv files even if the language selection didn't change since last time. All current messages will be removed first.

Example 1

```
MT_InitializeMessageHandling\ForceLoading;
```

Arguments

```
MT_InitializeMessageHandling [\ForceLoading]
```

`\ForceLoading`

Data type: `switch`

Forces loading of all messages even if there hasn’t been a language change or text file changes.

G.1.9 MT_Log - Creates a log message in log file

Usage

`MT_Log` is used to create a log message in log file.

Basic examples

The following example illustrates the instruction `MT_Log`.

Example 1

```
MT_Log MT_LVL_WARNING,"main",["Log message row 1","Log message row
    2","Log message row 3"];
```

Creates a new log in log file if log level warning is active. Depending on date and time the output in the csv file could look like below.

Arguments

```
MT_Log [Level] , [Source] , [Msg{*}]
```

`Level`

Data type: `mtloglevel`

Specifies the log level of this log message.

`Source`

Data type: `string`

Specifies the source of this log message, for normally routine name. Free for user to use it as something else than routine name.

`msg{*}`

Data type: `string array`

An array of log message lines.

G.1.10 MT_LogEnable - Enables or disables logging

Usage

`MT_LogEnable` is used to activate or deactivate logging in general or for a specific log level.

Basic examples

The following example illustrates the instruction .

Example 1

```
MT_LogEnable TRUE,\Level:=MT_LVL_WARNING;
```

Activates all logging to file which has the log level warning. See data type mtloglevel for more information about the different log levels.

Arguments

```
MT_LogEnable [Enabled] , [\Level]
```

`Enabled`

Data type: `bool`

Set to TRUE to activate logging. Set to deactivate logging.

`\Level`

Data type: `mtloglevel`

Specifies which kind of log level which should be turned on or off. If argument not used, turn on or off all kind of log levels.

G.1.11 MT_MovePartInfo - Moves part info between part trackers

Usage

`MT_MovePartInfo` is used to move part data information from one part tracker to another part tracker.

Basic examples

The following example illustrates the instruction `MT_MovePartInfo`.

Example 1

```
PERS mtstationdata sdRobot;
PERS mtstationdata sdMachine;
PERS num GRIPPER_1:=1;
PROC TestProc()
MT_MovePartInfo
     \FromStation:=sdRobot,\FromIndex:=GRIPPER_1,\ToStation:=sdMachine;
ENDPROC
```

This call will move part information from the part tracker connected to the robot station with index 1 (corresponds to gripper 1 in this case). The data will be moved to the part tracker of the machine, sdMachine. Since this machine only have one tracker the index was left out (means that index 1 will be used internally). Afterwards the FromStation tracker will have empty part data information.

Arguments

```
MT_MovePartInfo [\FromStation] | [\FromStationName] [\FromIndex]
     [\ToStation] | [\ToStationName] [\ToIndex]
```

`\FromStation`

Data type: `mtstationdata`

The station to move part tracker data from.

`\FromStationName`

Data type: `string`

The station to move part tracker data from. In this argument, the name of the mtstationdata definition is passed as a string.

`\FromIndex`

Data type: `num`

Specifies which part tracker index in FromStation to fetch part info from.

`\ToStation`

Data type: `mtstationdata`

The station to move part tracker data to.

`\ToStationName`

Data type: `string`

The station to move part tracker data to. In this argument, the name of the mtstationdata definition is passed as a string.

Continues on next page

G.1.11 MT_MovePartInfo - Moves part info between part trackers

RobotWare - OS
Continued

`\ToIndex`

Data type: `num`

Specifies which part tracker index in ToStation to move part info to.

G.1.12 MT_MoveRobotTo - Moves robot to a specific zone

Usage

`MT_MoveRobotTo` is used to move robot to a new zone/station. Call this when robot is standing in the via position of the current zone. If robot is not in the current zones via position, add argument \CheckStart.

Basic examples

The following example illustrates the instruction `MT_MoveRobotTo`.

Example 1

```
MT_MoveRobotTo ZONE_MACHINE;
```

This code moves robot to the position that is defined within the ZONE_MACHINE definition. But it's not a direct movement to this point. The movement will be done by calling the mvZONE_MACHINE.

Arguments

```
MT_MoveRobotTo [\CheckStart] , [TargetZone] , [\TargetZoneName]
```

`\CheckStart`

Data type: `switch`

Use switch if the robot should first move to the current zone via position. Normally the robot is in the current zone via position when calling this, and in that case don't use this switch. If using this argument the user will first get a question if robot should start moving straight for the via position.

`TargetZone`

Data type: `mttargetdata`

The mttargetdata of the zone that robot should move to.

`TargetZoneName`

Data type: `string`

If TargetZone is passed through a variable and not by it's real variable definition, then use this variable to pass the real name of the zone to get this working.

G.1.13 MT_MoveToStart - Moves robot to closest start position

Usage

`MT_MoveToStart` is used to move robot to closest via position, i.e. the closest safe start position.

Basic examples

The following example illustrates the instruction `MT_MoveToStart`.

Example 1

```
MT_MoveToStart;
```

Arguments

```
MT_MoveToStart [\RealMode]
```

`\RealMode`

Data type: `switch`

Only usable in virtual controller. If not used in virtual mode, robot will always go directly to closest without any checks. If used, robot will act as in auto and check distance to closest position and if too far away ask the operator if movement to that position is ok or not.

G.1.14 MT_MoveToZoneMenu - Shows manual menu to move between zones

Usage

`MT_MoveToZoneMenu` is used to move robot to any desired zone. Routine will show a list of all possible zones and ask user to select where to move. It's a good idea to connect this to a function key on Flexpendant and allow it only in manual mode.

Basic examples

The following example illustrates the instruction `MT_MoveToZoneMenu`.

Example 1

```
MT_MoveToZoneMenu;
```

If robot is in manual mode, the selection list of possible zones to move to will show on Flexpendant.

G.1.15 MT_ResetAlarms - Resets all or a specific alarm

Usage

`MT_ResetAlarms` is used to reset one or several alarms.

Basic examples

The following examples illustrates the instruction `MT_ResetAlarms`.

Example 1

```
MT_ResetAlarms \Identifier:="myAlarm";
```

Resets all active alarms with identifier myAlarm, note it could be more than one if wild card texts differ.

Example 2

```
MT_ResetAlarms;
```

Resets all active alarms.

Example 3

```
MT_ResetAlarms
     \Identifier:="myAlarm",\WildCardTexts:=["myWildCard"];
```

Resets the alarm that have the identifier "myAlarm" and is used with the wild card number 1: "myWildCard". Note that if more than one wild card was used in the alarm creation or if more was passed to the reset above, it is not an exact wild card match and the reset would not be done.

Arguments

```
MT_ResetAlarms [\Identifier] , [\WildCardTexts{*}]
```

`\Identifier`

Data type: `string`

System unique string that gives the possibility to point to a specific message to reset.

`\WildCardTexts{*}`

Data type: `string array`

String array of wild cards text to look for. All text must match with the active alarms wild card text to reset the message.

G.1.16 MT_SetAlarm - Sets an alarm to active

Usage

`MT_SetAlarm` is used to set an alarm to active state and also writes the message to robot e-log. At the same time a new event MT_EV_MSG_WRITTEN is created.

Basic examples

The following example illustrates the instruction `MT_SetAlarm`.

Example 1

```
MT_SetAlarm "myAlarm"\WildCardTexts:=["text 1", "text 2"];
```

myAlarm is set to active with the wild card texts “text 1” and “text 2”.

Arguments

```
MT_SetAlarm Identifier , [\WildCardTexts{*}]
```

`Identifier`

Data type: `string`

System unique string that gives the possibility to point to a specific message.

`\WildCardText{*}`

Data type: `string array`

String array of wild cards text to use for this message.

G.1.17 MT_SetPartInfo - Sets part information of a part in a specific tracker

Usage

`MT_SetPartInfo` is used to set any type of part data information to a specific part tracker.

Basic examples

The following example illustrates the instruction `MT_SetPartInfo`.

Example 1

```
PERS mtstationdata sdRobot;
PERS num GRIPPER_1:=1;
MT_SetPartInfo \Station:=sdRobot,\Index:=GRIPPER_1,\State:=psRAW;
```

This code will set part state raw to robot gripper 1 (Robot station index 1).

Arguments

```
MT_SetPartInfo [\Station] | [\StationName] [\Index] [\ProgCode]
     [\State] [\UserDef]
```

`\Station`

Data type: `mtstationdata`

The station to set part tracker data to.

`\StationName`

Data type: `string`

The station to set part tracker data to. In this argument name of the mtstationdata definition is passed as a string.

`\Index`

Data type: `num`

Specifies which part tracker index to set part info to.

`\ProgCode`

Data type: `dnum`

ProgCode to set to the specified part tracker. See mtpartdata.

`\State`

Data type: `mtpartstate`

State to store in the specified part tracker. See mtparttracker.

`\UserDef`

Data type: `string`

User defined data to store in the specified part tracker. See mtparttracker.

G.1.18 MT_UpdateMessageTextsIfNeeded - Reads messages into message array if needed

Usage

`MT_UpdateMessageTextsIfNeeded` is used to read message texts of used language into message array if file was changed since last read or if a read is asked anyway.

Basic examples

The following example illustrates the instruction `MT_UpdateMessageTextsIfNeeded`.

Example 1

```
MT_UpdateMessageTextsIfNeeded\ForceLoading;
```

This will always update message texts from files to RAPID system.

Arguments

```
MT_UpdateMessageTextsIfNeeded(\ForceLoading)
```

`ForceLoading)`

Data type: `switch`

Use if texts always should be updated independent of changes to language settings or file modifications.

G.2 Functions

G.2.1 MT_GetActiveAlarm - Get an active alarm data

Usage

Returns a complete active alarm data from the array of active alarms. If there are no alarms or nothing on the index number asked, then an empty active alarm data is returned.

Basic examples

The following example illustrates the function `MT_GetActiveAlarm`.

Example 1

```
VAR mtactivealarmdata Alarm;
Alarm:=MT_GetActiveAlarm(\Index:=1);
```

This will fetch the complete active alarm data for message with index 1 (latest alarm created) and store it into Alarm.

Return value

Data type: `mtactivealarmdata`

Arguments

```
MT_GetMessage ([\Index])
```

`\Index`

Data type: `num`

Index of the alarm that should be returned from the active alarm list. If not used, index 1 (latest created alarm) will be returned.

G.2.2 MT_GetCurrentZone - Get zone where robot is positioned now

Usage

`MT_GetCurrentZone` is used to find out in which zone robot is positioned in right now.

Basic examples

The following example illustrates the function `MT_GetCurrentZone`.

Example 1

```
VAR mttargetdata Current;
Current:=MT_GetCurrentZone();
```

This code checks where the robot is positioned right now and returns matching mttargetdata which is stored in Current.

Return value

Data type: `mttargetdata`

Target data matching the current position of the robot.

Arguments

```
MT_GetCurrentZone(\INOUT ZoneName)
```

`ZoneName)`

Data type: `string`

Name as a string of the returned zone.

G.2.3 MT_GetMessage - Get complete message data

Usage

`MT_GetMessage` is used to get the complete mtmsgdata by passing corresponding identifier.

Basic examples

The following example illustrates the function `MT_GetMessage`. This will fetch the complete message data for message with identifier “myAlarm” and store it into Alarm.

Example 1

```
VAR mtmsgdata Alarm;
Alarm:=MT_GetMessage("myAlarm");
```

Return value

Data type: `mtmsgdata`

Arguments

```
MT_GetMessage (Identifier)
```

`Identifier`

Data type: `string`

System unique string that gives the possibility to point to a specific message.

G.2.4 MT_GetNumberOfActiveAlarms - Gets number of active alarms

Usage

`MT_GetNumberOfActiveAlarms` is used to check how many alarms are currently active.

Basic examples

The following example illustrates the function `MT_GetNumberOfActiveAlarms`. This will fetch the number of active alarms in the system and store it into NumberOfAlarms.

Example 1

```
VAR num NumberOfAlarms;
NumberOfAlarms:=MT_GetNumberOfActiveAlarms();
```

Return value

Data type: `num`

G.2.5 MT_GetPartState - Gets current part state of a part in specific tracker

Usage

`MT_GetPartState` is used to get the part state of a part in a specific part tracker. This is normally a faster way to fetch the state than using MT_GetPartInfo.

Basic examples

The following example illustrates the function `MT_GetPartState`.

Example 1

```
PERS mtstationdata sdRobot;
PERS num GRIPPER_1:=1;
IF MT_GetPartState(\Station:=sdRobot,\Index:=GRIPPER_1)=psRAW THEN
!Do actions if raw part
ENDIF
```

Take some actions if the part in robot gripper 1 is a raw part.

Return value

Data type: `mtpartstate`

Arguments

```
MT_GetPartState ([\Station] | [\StationName] [\Index])
```

`\Station`

Data type: `mtstationdata`

The station data which part state data should be fetched in.

`\StationName`

Data type: `string`

Name of the station data declaration as a string. Through this name the correct mtstationdata definition will be found and the corresponding part tracker where part state data should be fetched.

`\Index`

Data type: `num`

Index number of the part tracker fetch state from. Note, each station can have up to 20 trackers.

G.2.6 MT_GetText - Gets a selected text line from a message

Usage

`MT_GetText` is used to get a selected text line from a specific message.

Basic examples

The following example illustrates the function `MT_GetText`. This will store the header text from the message corresponding to identifier “myAlarm” to the variable header. if no optional argument is used, the header text is returned.

Example 1

```
VAR string header;
header:=MT_GetText("myAlarm",\Header);
```

Return value

Data type: `string`

Arguments

```
MT_GetText (Identifier, [\Header] | [\Text1] | [\Text2] | [\Text3]
     | [\Text4] | [\Text5] | [\Text6] | [\Text7] | [\Text8])
```

`Identifier`

Data type: `string`

System unique string that gives the possibility to point to a specific message.

`\Header`

Data type: `switch`

Gets the header text line.

`\Text1`

Data type: `switch`

Gets the body text number 1.

`\Text2`

Data type: `switch`

Gets the body text number 2.

`\Text3`

Data type: `switch`

Gets the body text number 3.

`\Text4`

Data type: `switch`

Gets the body text number 4.

`\Text5`

Data type: `switch`

Gets the body text number 5.

Continues on next page

G.2.6 MT_GetText - Gets a selected text line from a message

RobotWare - OS
Continued

`\Text6`

Data type: `switch`

Gets the body text number 6.

`\Text7`

Data type: `switch`

Gets the body text number 7.

`\Text8`

Data type: `switch`

Gets the body text number 8.

G.2.7 MT_GetViaPosition - Gets corresponding via position for a zone

Usage

`MT_GetViaPosition` is used to fetch the robtarget data from a specific mttargetdata.

Basic examples

The following example illustrates the function `MT_GetViaPosition`.

Example 1

```
MoveJ MT_GetViaPosition(ZONE_MACHINE),v500,fine,tTool0;
```

Moves robot to the via position connected to ZONE_MACHINE.

Return value

Data type: `robtarget`

Arguments

```
MT_GetViaPosition([Target])
```

`Target`

Data type: `mttargetdata`

The mttargetdata for the target for which the robtarget should be fetched.

G.2.8 MT_IsAlarmActive - Check if an specific alarm is active

Usage

`MT_IsAlarmActive` is used to check if an specific alarm is currently active. Combine category and station if needed to check currently active alarm on a specific station of a specific category.

Basic examples

The following examples illustrates the function `MT_IsAlarmActive`.

Example 1

```
VAR dionum AlarmActive;
AlarmActive:=MT_IsAlarmActive();
```

AlarmActive will be 1 if any alarm is active in the system.

Example 2

```
AlarmActive:=MT_IsAlarmActive(\Identifier:="myAlarm");
```

AlarmActive will be 1 if the alarm with identifier “myAlarm” is active.

Return value

Data type: `dionum`

Arguments

```
MT_IsAlarmActive ([\Identifier] , [\WildCardTexts{*}] , [\Category]
     , [\StationName])
```

`\Identifier`

Data type: `string`

System unique string that gives the possibility to point to a specific message.

`\WildCardTexts{*}`

Data type: `string array`

String array of wild cards text to look for. All text must match to get a result 1 back. This argument only has a meaning if used at the same time as the identifier argument.

`\Category`

Data type: `icondata`

Only return result 1 if an alarm matches this category. This argument can’t be used together with Identifier argument.

`\StationName`

Data type: `string`

Only return true if there is an active alarm which is connected to this station. This argument can’t be used together with Identifier argument.

G.2.9 MT_UIBoolEntry - Show an yes no question in localized language

Usage

`MT_UIBoolEntry` is used to show a UI message box on Flexpendant. Texts are fetched from correct message, use also wild card texts if wanted. Only two buttons are possible to show, one that will give the result TRUE and one that will give result FALSE. Except for the texts, all other arguments available in UIMessageBox is available here to.

Basic examples

The following example illustrates the function `MT_UIBoolEntry`.

Example 1

```
VAR bool answer;
answer:=MT_UIBoolEntry("myMessage",\WildCardTexts:=["text 1","text
     2"],\TrueButtonText:="Yes",\FalseButtonText:="No");
```

This will show a UI message box on the Flexpendant with header and text fetched from the message corresponding to identifier “myMessage”. Wild cards {1} and {2} will be replaced with “text 1” and “text 2” correspondingly. Two buttons will show, one with text “Yes” and one with text “No”. If the "True button" is pressed, function will return TRUE.

Return value

Data type: `bool`

Arguments

```
MT_UIBoolEntry (Identifier , [\WildCardTexts{*}] , [\Wrap] ,
     [\TrueButtonText] , [\FalseButtonText] , [\DefaultBtn] ,
     [\Icon] , [\Image] , [\MaxTime] , [\DIBreak] , [\DIPassive]
     , [\DOBreak] , [\DOPassive] , [\PersBoolBreak] ,
     [\PersBoolPassive] , [\BreakFlag] , [\UIActiveSignal])
```

`Identifier`

Data type: `string`

System unique string that gives the possibility to point to a specific message.

`\WildCardTexts{*}`

Data type: `string array`

String array of wild cards text to use for this message.

`[\Wrap]`

Data type: `switch`

If selected, all the specified strings in the argument `\MsgArray` will be concatenated to one string with single spaces between each individual string and spread out on as few lines as possible.

Default, each string in the argument `\MsgArray` will be on separate line on the display.

Continues on next page

G.2.9 MT_UIBoolEntry - Show an yes no question in localized language
RobotWare - OS
Continued

`\TrueButtonText`

Data type: `string`

Text to show on button resulting true value.

`\FalseButtonText`

Data type: `string`

Text to show on button resulting false value.

`[\DefaultBtn]`

Default Button

Data type: `btnres`

Allows to specify a value that should be returned if the message box is interrupted by `\MaxTime`, `\DIBreak`, or `\DOBreak`. It's possible to specify the predefined symbolic constant of type `btnres` or any user defined value. See *Predefined data on page 266*.

`[\Icon]`

Data type: `icondata`

Defines the icon to be displayed. Only one of the predefined icons of type `icondata` can be used. See *Predefined data on page 266*.

Default, no icon.

`[\Image]`

Data type: `string`

The name of the image that should be used. To launch own images, the images has to be placed in the `HOME:` directory in the active system or directly in the active system.

The recommendation is to place the files in the `HOME:` directory so that they are saved if a Backup and Restore is done.

A **Restart** is required and then the FlexPendant loads the images.

The image that will be shown can have the width of 185 pixels and the height of 300 pixels. If the image is bigger, only 185 * 300 pixels of the image will be shown starting at the top left of the image.

No exact value can be specified on the size that an image can have or the amount of images that can be loaded to the FlexPendant. It depends on the size of other files loaded to the FlexPendant. The program execution will just continue if an image is used that has not been loaded to the FlexPendant.

`[\MaxTime]`

Data type: `num`

The maximum amount of time in seconds that program execution waits. If no button is selected within this time, the program continues to execute in the error handler unless the `BreakFlag` is used (see below). The constant `ERR_TP_MAXTIME` can be used to test whether or not the maximum time has elapsed.

`[\DIBreak]`

Digital Input Break

Continues on next page

Data type: `signaldi`

The digital input signal that may interrupt the operator dialog. If no button is selected when the signal is set to 1 (or is already 1) then the program continues to execute in the error handler, unless the `BreakFlag` is used (see below). The constant `ERR_TP_DIBREAK` can be used to test whether or not this has occurred.

`[\DIPassive]`

Digital Input Passive

Data type: `switch`

This switch overrides the default behavior when using `DIBreak` optional argument. Instead of reacting when signal is set to 1 (or already 1), the instruction should continue in the error handler (if no `BreakFlag` is used) when the signal `DIBreak` is set to 0 (or already is 0). The constant `ERR_TP_DIBREAK` can be used to test whether or not this has occurred.

`[\DOBreak]`

Digital Output Break

Data type: `signaldo`

The digital output signal that may interrupt the operator dialog. If no button is selected when the signal is set to 1 (or is already 1) then the program continues to execute in the error handler, unless the `BreakFlag` is used (see below). The constant `ERR_TP_DOBREAK` can be used to test whether or not this has occurred.

`[\DOPassive]`

Digital Output Passive

Data type: `switch`

This switch overrides the default behavior when using `DOBreak` optional argument. Instead of reacting when signal is set to 1 (or already 1), the instruction should continue in the error handler (if no `BreakFlag` is used) when the signal `DOBreak` is set to 0 (or already is 0). The constant `ERR_TP_DOBREAK` can be used to test whether or not this has occurred.

`[\PersBoolBreak]`

Persistent Boolean Break

Data type: `bool`

The persistent boolean that may interrupt the operator dialog. If no button is selected when the persistent boolean is set to TRUE (or is already TRUE) then the program continues to execute in the error handler unless the `BreakFlag` is used (see below). The constant `ERR_TP_PERSBOOLBREAK` can be used to test whether or not this has occurred.

`[\PersBoolPassive]`

Persistent Boolean Passive

Data type: `switch`

This switch overrides the default behavior when using `PersBoolBreak` optional argument. Instead of reacting when persistent boolean is set to TRUE (or already TRUE), the instruction should continue in the error handler (if no `BreakFlag` is

Continues on next page

G.2.9 MT_UIBoolEntry - Show an yes no question in localized language
RobotWare - OS
Continued

used) when the persistent boolean `PersBoolBreak` is set to FALSE (or already is FALSE). The constant `ERR_TP_PERSBOOLBREAK` can be used to test whether or not this has occurred.

`[\BreakFlag]`

Data type: `errnum`

A variable that will hold the error code if `MaxTime`, `DIBreak`, `DOBreak`, or `PersBoolBreak` is used. If this optional variable is omitted then the error handler will be executed. The constants `ERR_TP_MAXTIME, ERR_TP_DIBREAK`, `ERR_TP_DOBREAK`, and `ERR_TP_PERSBOOLBREAK` can be used to select the reason.

`[\UIActiveSignal]`

Data type: `signaldo`

The digital output signal used in optional argument `UIActiveSignal` is set to 1 when the message box is activated on the FlexPendant. When the user selection has been done and the execution continue, the signal is set to 0 again.

No supervision of stop or restart exist. The signal is set to 0 when the function is ready, or when PP is moved.

G.2.10 MT_UIDNumEntry - Shows a localized dnum entry

Usage

`MT_UIDNumEntry` is used to show a UI dnum entry box on Flexpendant. Texts are fetched from correct message, use also wild card texts if wanted. All other arguments have the exact same functionality as the corresponding standard function UIDnumEntry.

Basic examples

The following example illustrates the function `MT_UIDNumEntry`.

Example 1

```
VAR dnum answer;
answer:=MT_UIDNumEntry("myMessage",\WildCardTexts:=["text 1","text
    2"]);
```

This shows a UI dnum entry box with header and text fetched from the message that corresponds to the Identifier myMessage. The wild cards {1} and {2} in the texts will be replaced by “text 1” and “text 2” respectively.

Return value

Data type: `dnum`

Arguments

```
MT_UIDnumEntry (Identifier , [\WildCardTexts{*}] , [\Wrap] , [\Icon]
    , [\InitValue] ,[\MinValue] , [\MaxValue] , [\AsInteger] ,
    [\MaxTime] , [\DIBreak] , [\DIPassive] , [\DOBreak] ,
    [\DOPassive] , [\PersBoolBreak] , [\PersBoolPassive] ,
    [\BreakFlag] , [\UIActiveSignal])
```

`Identifier`

Data type: `string`

System unique string that gives the possibility to point to a specific message.

`\WildCardTexts{*}`

Data type: `string array`

String array of wild cards text to use for this message.

`[\Wrap]`

Data type: `switch`

If selected, all the specified strings in the argument `\MsgArray` will be concatenated to one string with a single space between each individual string, and spread out on as few lines as possible.

Default, each string in the argument `\MsgArray` will be on a separate line on the display.

`[\Icon]`

Data type: `icondata`

Defines the icon to be displayed. Only one of the predefined icons of type `icondata` can be used. See *Predefined data on page 262*.

Continues on next page

G.2.10 MT_UIDNumEntry - Shows a localized dnum entry
RobotWare - OS
Continued

Default no icon.

`[\InitValue]`

Data type: `dnum`

Initial value that is displayed in the entry box.

`[\MinValue]`

Data type: `dnum`

The minimum value for the return value.

`[\MaxValue]`

Data type: `dnum`

The maximum value for the return value.

`[\AsInteger]`

Data type: `switch`

Eliminates the decimal point from the number pad to ensure that the return value is an integer.

`[\MaxTime]`

Data type: `num`

The maximum amount of time in seconds that program execution waits. If the OK button is not pressed within this time, the program continues to execute in the error handler unless the `BreakFlag` is used (see below). The constant `ERR_TP_MAXTIME` can be used to test whether or not the maximum time has elapsed.

`[\DIBreak]`

Digital Input Break

Data type: `signaldi`

The digital input signal that may interrupt the operator dialog. If the OK button is not pressed before the signal is set to 1 (or is already 1) then the program continues to execute in the error handler unless the `BreakFlag` is used (see below). The constant `ERR_TP_DIBREAK` can be used to test whether or not this has occurred.

`[\DIPassive]`

Digital Input Passive

Data type: `switch`

This switch overrides the default behavior when using `DIBreak` optional argument. Instead of reacting when signal is set to 1 (or already 1), the instruction should continue in the error handler (if no `BreakFlag` is used) when the signal `DIBreak` is set to 0 (or already is 0). The constant `ERR_TP_DIBREAK` can be used to test whether or not this has occurred.

`[\DOBreak]`

Digital Output Break

Data type: `signaldo`

Continues on next page

The digital output signal that may interrupt the operator dialog. If the OK button is not pressed before the signal is set to 1 (or is already 1) then the program continues to execute in the error handler unless the `BreakFlag` is used (see below). The constant `ERR_TP_DOBREAK` can be used to test whether or not this has occurred.

`[\DOPassive]`

Digital Output Passive

Data type: `switch`

This switch overrides the default behavior when using `DOBreak` optional argument. Instead of reacting when signal is set to 1 (or already 1), the instruction should continue in the error handler (if no `BreakFlag` is used) when the signal `DOBreak` is set to 0 (or already is 0). The constant `ERR_TP_DOBREAK` can be used to test whether or not this has occurred.

`[\PersBoolBreak]`

Persistent Boolean Break

Data type: `bool`

The persistent boolean that may interrupt the operator dialog. If no button is selected when the persistent boolean is set to TRUE (or is already TRUE) then the program continues to execute in the error handler unless the `BreakFlag` is used (see below). The constant `ERR_TP_PERSBOOLBREAK` can be used to test whether or not this has occurred.

`[\PersBoolPassive]`

Persistent Boolean Passive

Data type: `switch`

This switch overrides the default behavior when using `PersBoolBreak` optional argument. Instead of reacting when persistent boolean is set to TRUE (or already TRUE), the instruction should continue in the error handler (if no `BreakFlag` is used) when the persistent boolean `PersBoolBreak` is set to FALSE (or already is FALSE). The constant `ERR_TP_PERSBOOLBREAK` can be used to test whether or not this has occurred.

`[\BreakFlag]`

Data type: `errnum`

A variable that will hold the error code if `MaxTime`, `DIBreak`, `DOBreak`, or `PersBoolBreak` is used. If this optional variable is omitted then the error handler will be executed. The constants `ERR_TP_MAXTIME`, `ERR_TP_DIBREAK`, `ERR_TP_DOBREAK`, and `ERR_TP_PERSBOOLBREAK` can be used to select the reason.

`[\UIActiveSignal]`

Data type: `signaldo`

The digital output signal used in optional argument `UIActiveSignal` is set to 1 when the message box is activated on the FlexPendant. When the user selection has been done and the execution continue, the signal is set to 0 again.

No supervision of stop or restart exist. The signal is set to 0 when the function is ready, or when PP is moved.

Continues on next page

G.2.10 MT_UIDNumEntry - Shows a localized dnum entry
RobotWare - OS
Continued

Predefined data

```
!Icons:
  CONST icondata iconNone := 0;
  CONST icondata iconInfo := 1;
  CONST icondata iconWarning := 2;
  CONST icondata iconError := 3;
```

G.2.11 MT_UIMessageBox - Show a message box in localized language

Usage

`MT_UIMessageBox` is used to show a UI message box on Flexpendant. Texts are fetched from correct message, use also wild card texts if wanted. Except for the header and body texts, all other arguments available in UIMessageBox is available here to.

Basic examples

The following example illustrates the function `MT_UIMessageBox`.

Example 1

```
VAR btnres answer;
answer:=MT_UIMessageBox("myMessage",\WildCardTexts:=["text 1","text
     2"]);
```

This will show a UI message box on the Flexpendant with header and text fetched from the message corresponding to identifier “myMessage”. Wild cards {1} and {2} will be replaced with “text 1” and “text 2” correspondingly. The rest will be according to the standard UIMessageBox function without additional arguments.

Return value

Data type: `btnres`

Arguments

```
UIMessageBox ( Identifier , [\WildCardTexts{*}] [\Wrap][\Buttons]
     | [\BtnArray] [\DefaultBtn] [\Icon][\Image] [\MaxTime]
     [\DIBreak] [\DIPassive] [\DOBreak] [\DOPassive]
     [\PersBoolBreak] [\PersBoolPassive] [\BreakFlag]
     [\UIActiveSignal])
```

`Identifier`

Data type: `string`

System unique string that gives the possibility to point to a specific message.

`\WildCardTexts{*}`

Data type: `string array`

String array of wild cards text to use for this message.

`[\Wrap]`

Data type: `switch`

If selected, all the specified strings in the argument `\MsgArray` will be concatenated to one string with single spaces between each individual string and spread out on as few lines as possible.

Default, each string in the argument `\MsgArray` will be on separate line on the display.

`[\Buttons]`

Data type: `buttondata`

Continues on next page

G.2.11 MT_UIMessageBox - Show a message box in localized language
RobotWare - OS
Continued

Defines the push buttons to be displayed. Only one of the predefined buttons combination of type buttondata can be used. See *Predefined data on page 266*.

Default, the system displays the OK button.

`[\BtnArray]`

Button Array

Data type: `string`

Own definition of push buttons stored in an array of strings. This function returns the array index when corresponding string is selected.

Only one of parameter `\Buttons` or `\BtnArray` can be used at the same time.

Max. 5 buttons with 42 characters each.

`[\DefaultBtn]`

Default Button

Data type: `btnres`

Allows to specify a value that should be returned if the message box is interrupted by `\MaxTime`, `\DIBreak`, or `\DOBreak`. It's possible to specify the predefined symbolic constant of type `btnres` or any user defined value. See *Predefined data on page 266*.

`[\Icon]`

Data type: `icondata`

Defines the icon to be displayed. Only one of the predefined icons of type `icondata` can be used. See *Predefined data on page 266*.

Default, no icon.

`[\Image]`

Data type: `string`

The name of the image that should be used. To launch own images, the images has to be placed in the `HOME:` directory in the active system or directly in the active system.

The recommendation is to place the files in the `HOME:` directory so that they are saved if a Backup and Restore is done.

A **Restart** is required and then the FlexPendant loads the images.

The image that will be shown can have the width of 185 pixels and the height of 300 pixels. If the image is bigger, only 185 * 300 pixels of the image will be shown starting at the top left of the image.

No exact value can be specified on the size that an image can have or the amount of images that can be loaded to the FlexPendant. It depends on the size of other files loaded to the FlexPendant. The program execution will just continue if an image is used that has not been loaded to the FlexPendant.

`[\MaxTime]`

Data type: `num`

The maximum amount of time in seconds that program execution waits. If no button is selected within this time, the program continues to execute in the error handler

Continues on next page

unless the BreakFlag is used (see below). The constant ERR_TP_MAXTIME can be used to test whether or not the maximum time has elapsed.

[\DIBreak]

Digital Input Break

Data type: signaldi

The digital input signal that may interrupt the operator dialog. If no button is selected when the signal is set to 1 (or is already 1) then the program continues to execute in the error handler, unless the BreakFlag is used (see below). The constant ERR_TP_DIBREAK can be used to test whether or not this has occurred.

[\DIPassive]

Digital Input Passive

Data type: switch

This switch overrides the default behavior when using DIBreak optional argument. Instead of reacting when signal is set to 1 (or already 1), the instruction should continue in the error handler (if no BreakFlag is used) when the signal DIBreak is set to 0 (or already is 0). The constant ERR_TP_DIBREAK can be used to test whether or not this has occurred.

[\DOBreak]

Digital Output Break

Data type: signaldo

The digital output signal that may interrupt the operator dialog. If no button is selected when the signal is set to 1 (or is already 1) then the program continues to execute in the error handler, unless the BreakFlag is used (see below). The constant ERR_TP_DOBREAK can be used to test whether or not this has occurred.

[\DOPassive]

Digital Output Passive

Data type: switch

This switch overrides the default behavior when using DOBreak optional argument. Instead of reacting when signal is set to 1 (or already 1), the instruction should continue in the error handler (if no BreakFlag is used) when the signal DOBreak is set to 0 (or already is 0). The constant ERR_TP_DOBREAK can be used to test whether or not this has occurred.

[\PersBoolBreak]

Persistent Boolean Break

Data type: bool

The persistent boolean that may interrupt the operator dialog. If no button is selected when the persistent boolean is set to TRUE (or is already TRUE) then the program continues to execute in the error handler unless the BreakFlag is used (see below). The constant ERR_TP_PERSBOOLBREAK can be used to test whether or not this has occurred.

Continues on next page

G.2.11 MT_UIMessageBox - Show a message box in localized language

RobotWare - OS
Continued

`[\PersBoolPassive]`

Persistent Boolean Passive

Data type: `switch`

This switch overrides the default behavior when using `PersBoolBreak` optional argument. Instead of reacting when persistent boolean is set to TRUE (or already TRUE), the instruction should continue in the error handler (if no `BreakFlag` is used) when the persistent boolean `PersBoolBreak` is set to FALSE (or already is FALSE). The constant `ERR_TP_PERSBOOLBREAK` can be used to test whether or not this has occurred.

`[\BreakFlag]`

Data type: `errnum`

A variable that will hold the error code if `MaxTime`, `DIBreak`, `DOBreak`, or `PersBoolBreak` is used. If this optional variable is omitted then the error handler will be executed. The constants `ERR_TP_MAXTIME`, `ERR_TP_DIBREAK`, `ERR_TP_DOBREAK`, and `ERR_TP_PERSBOOLBREAK` can be used to select the reason.

`[\UIActiveSignal]`

Data type: `signaldo`

The digital output signal used in optional argument `UIActiveSignal` is set to 1 when the message box is activated on the FlexPendant. When the user selection has been done and the execution continue, the signal is set to 0 again.

No supervision of stop or restart exist. The signal is set to 0 when the function is ready, or when PP is moved.

Predefined data

```
!Icons:
  CONST icondata iconNone := 0;
  CONST icondata iconInfo := 1;
  CONST icondata iconWarning := 2;
  CONST icondata iconError := 3;
!Buttons:
  CONST buttondata btnNone := -1;
  CONST buttondata btnOK := 0;
  CONST buttondata btnAbrtRtryIgn := 1;
  CONST buttondata btnOKCancel := 2;
  CONST buttondata btnRetryCancel := 3;
  CONST buttondata btnYesNo := 4;
  CONST buttondata btnYesNoCancel := 5;
!Results:
  CONST btnres resUnkwn := 0;
  CONST btnres resOK := 1;
  CONST btnres resAbort := 2;
  CONST btnres resRetry := 3;
  CONST btnres resIgnore := 4;
  CONST btnres resCancel := 5;
  CONST btnres resYes := 6;
  CONST btnres resNo := 7;
```

G.2.12 MT_UINumEntry - Shows a num entry box in localized language

Usage

`MT_UINumEntry` is used to show a UI num entry box on Flexpendant. Texts are fetched from correct message, use also wild card texts if wanted. All other arguments have the exact same functionality as the corresponding standard function UINumEntry.

Basic examples

The following example illustrates the function `MT_UINumEntry`.

Example 1

```
VAR num answer;
answer:=MT_UINumEntry("myMessage",\WildCardTexts:=["text 1","text
    2"]);
```

This shows a UI num entry box with header and text fetched from the message that corresponds to the Identifier myMessage. The wild cards {1} and {2} in the texts will be replaced by “text 1” and “text 2” respectively.

Return value

Data type: `num`

Arguments

```
MT_UINumEntry (Identifier , [\WildCardTexts{*}] , [\Wrap] , [\Icon]
    , [\InitValue] ,[\MinValue] , [\MaxValue] , [\AsInteger] ,
    [\MaxTime] , [\DIBreak] , [\DIPassive] , [\DOBreak] ,
    [\DOPassive] , [\PersBoolBreak] , [\PersBoolPassive] ,
    [\BreakFlag] , [\UIActiveSignal])
```

`Identifier`

Data type: `string`

System unique string that gives the possibility to point to a specific message.

`\WildCardTexts{*}`

Data type: `string array`

String array of wild cards text to use for this message.

`[\Wrap]`

Data type: `switch`

If selected, all the specified strings in the argument `\MsgArray` will be concatenated to one string with a single space between each individual string, and spread out on as few lines as possible.

Default, each string in the argument `\MsgArray` will be on a separate line on the display.

`[\Icon]`

Data type: `icondata`

Defines the icon to be displayed. Only one of the predefined icons of type `icondata` can be used. See *Predefined data on page 270*.

Continues on next page

G.2.12 MT_UINumEntry - Shows a num entry box in localized language
RobotWare - OS
Continued

Default no icon.

`[\InitValue]`

Data type: `num`

Initial value that is displayed in the entry box.

`[\MinValue]`

Data type: `num`

The minimum value for the return value.

`[\MaxValue]`

Data type: `num`

The maximum value for the return value.

`[\AsInteger]`

Data type: `switch`

Eliminates the decimal point from the number pad to ensure that the return value is an integer.

`[\MaxTime]`

Data type: `num`

The maximum amount of time in seconds that program execution waits. If the OK button is not pressed within this time, the program continues to execute in the error handler unless the `BreakFlag` is used (see below). The constant `ERR_TP_MAXTIME` can be used to test whether or not the maximum time has elapsed.

`[\DIBreak]`

Digital Input Break

Data type: `signaldi`

The digital input signal that may interrupt the operator dialog. If the OK button is not pressed before the signal is set to 1 (or is already 1) then the program continues to execute in the error handler, unless the `BreakFlag` is used (see below). The constant `ERR_TP_DIBREAK` can be used to test whether or not this has occurred.

`[\DIPassive]`

Digital Input Passive

Data type: `switch`

This switch overrides the default behavior when using `DIBreak` optional argument. Instead of reacting when signal is set to 1 (or already 1), the instruction should continue in the error handler (if no `BreakFlag` is used) when the signal `DIBreak` is set to 0 (or already is 0). The constant `ERR_TP_DIBREAK` can be used to test whether or not this has occurred.

`[\DOBreak]`

Digital Output Break

Data type: `signaldo`

Continues on next page

The digital output signal that may interrupt the operator dialog. If the OK button is not pressed before the signal is set to 1 (or is already 1) then the program continues to execute in the error handler, unless the `BreakFlag` is used (see below). The constant `ERR_TP_DOBREAK` can be used to test whether or not this has occurred.

`[\DOPassive]`

Digital Output Passive

Data type: `switch`

This switch overrides the default behavior when using `DOBreak` optional argument. Instead of reacting when signal is set to 1 (or already 1), the instruction should continue in the error handler (if no `BreakFlag` is used) when the signal `DOBreak` is set to 0 (or already is 0). The constant `ERR_TP_DOBREAK` can be used to test whether or not this has occurred.

`[\PersBoolBreak]`

Persistent Boolean Break

Data type: `bool`

The persistent boolean that may interrupt the operator dialog. If no button is selected when the persistent boolean is set to TRUE (or is already TRUE) then the program continues to execute in the error handler unless the `BreakFlag` is used (see below). The constant `ERR_TP_PERSBOOLBREAK` can be used to test whether or not this has occurred.

`[\PersBoolPassive]`

Persistent Boolean Passive

Data type: `switch`

This switch overrides the default behavior when using `PersBoolBreak` optional argument. Instead of reacting when persistent boolean is set to TRUE (or already TRUE), the instruction should continue in the error handler (if no `BreakFlag` is used) when the persistent boolean `PersBoolBreak` is set to FALSE (or already is FALSE). The constant `ERR_TP_PERSBOOLBREAK` can be used to test whether or not this has occurred.

`[\BreakFlag]`

Data type: `errnum`

A variable that will hold the error code if `MaxTime`, `DIBreak`, `DOBreak`, or `PersBoolBreak` is used. If this optional variable is omitted then the error handler will be executed. The constants `ERR_TP_MAXTIME`, `ERR_TP_DIBREAK`, `ERR_TP_DOBREAK`, and `ERR_TP_PERSBOOLBREAK` can be used to select the reason.

`[\UIActiveSignal]`

Data type: `signaldo`

The digital output signal used in optional argument `UIActiveSignal` is set to 1 when the message box is activated on the FlexPendant. When the user selection has been done and the execution continue, the signal is set to 0 again.

No supervision of stop or restart exist. The signal is set to 0 when the function is ready, or when PP is moved.

Continues on next page

G.2.12 MT_UINumEntry - Shows a num entry box in localized language
RobotWare - OS
Continued

Predefined data

```
!Icons:
  CONST icondata iconNone := 0;
  CONST icondata iconInfo := 1;
  CONST icondata iconWarning := 2;
  CONST icondata iconError := 3;
```

G.3 Data types

G.3.1 mtaxisrange - specifies an axis with an axis range

Usage

`mtaxisrange` is used to specify an axis and a range for the axis. Used to define for which axis values the robot is within a zone.

Components

The data type `mtaxisrange` has the following components:

`Axis`

Data type: `num`

Number of the robot axis to use. 1-6 normal robot axis, 7-12 external axis.

`Range`

Data type: `mtrange`

The range for this axis.

G.3.2 mtloglevel - define a log level

Usage

`mtloglevel` is used to define a log level.

Predefined data

The following symbolic constants of the data type `mtloglevel` are predefined and can be used when specifying a condition for the instructions `MT_Log` and `MT_LogEnable`.

Value	Symbolic constant	Comment
1	`MT_LVL_ERROR`	
2	`MT_LVL_WARNING`	
3	`MT_LVL_INFO`	
4	`MT_LVL_DEBUG`	
5	`MT_LVL_USER`	

Characteristics

`mtloglevel` is an alias data type for `num` and consequently inherits its characteristics.

G.3.3 mtmsgdata - alarm and message definition data

Usage

`mtmsgdata` is used to describe all messages and alarms in the system.

Components

The data type `mtmsgdata` has the following components:

`Identifier`

Data type: `string`

System unique string that gives the possibility to point to a specific message.

`Category`

Data type: `icondata`

Describes which alarm category the alarm has. Messages that's not of any alarm type has an empty category data.

`StationName`

Data type: `string`

Name of the station to which the alarm belongs to. This could be used to sort or group alarms better, for example in an HMI.

`Number`

Data type: `num`

Message number will be shown as part of the alarm number in alarm lists. Can also be a way to keep structure of the messages, but not mandatory to use this.

`Domain`

Data type: `num`

Message domain number. This is not mandatory, could be used to group alarms in a different way (maybe a bigger type of group than station level).

`Header`

Data type: `string`

Header text for the message.

`Text1`

Data type: `string`

First text line of body text for the message.

`Text2`

Data type: `string`

Second text line of body text for the message.

`Text3`

Data type: `string`

Third text line of body text for the message.

Continues on next page

G.3.3 mtmsgdata - alarm and message definition data
RobotWare - OS
Continued

`Text4`

Data type: `string`

Forth text line of body text for the message.

`Text5`

Data type: `string`

Fifth text line of body text for the message.

`Text6`

Data type: `string`

Sixth text line of body text for the message.

`Text7`

Data type: `string`

Seventh text line of body text for the message.

`Text8`

Data type: `string`

Eighth text line of body text for the message.

G.3.4 mtpartdata - Describes a part

Usage

`mtpartdata` is used to describe a part.

Components

The data type `mtpartdata` has the following components:

`ProgCode`

Data type: `dnum`

Unique number for this part to identify it.

`TypeCode`

Data type: `dnum`

Type coding that can be used in the robot program if needed.

`AuxCode`

Data type: `dnum`

Alternative program number to address for example, a vision system which uses program numbers that are different from the robot's program numbers.

`ToolCode`

Data type: `num`

Currently not used.

`MachineCode`

Data type: `dnum`

Code for a machine, served by robot (for example, form code of an Injection Moulding Machine). Only use if it makes sense in the project.

`RoutineName`

Data type: `string`

Currently not used.

`RunInTasks`

Data type: `string`

Currently not used.

`Tool`

Data type: `string`

Currently not used.

`PartLoad`

Data type: `string`

Name of the predefined loaddata associated with this part. This can remain empty, if not needed or if the loaddata information shall be handled flexible in part trackers.

G.3.5 mtpartstate - Describes a part state

Usage

`mtpartstate` is used to describe current state of a part.

Predefined data

The following symbolic constants of the data type `mtpartstate` are predefined and can be used when specifying a condition for instructions like `MT_SetPartState` and `MT_GetPartState`.

Value	Symbolic constant	Comment
0	`psEMPTY`	
500	`psRAW`	
501	`psMACHINED`	
502	`psFINISHED`	
503	`psCOOLED`	

Characteristics

`mtpartstate` is an alias data type for `num` and consequently inherits its characteristics.

G.3.6 mtparttracker - describes all data for a part tracker

Usage

`mtparttracker` is used to hold all data handled in a part tracker. Each defined station must have a corresponding mtparttracker definition to make it work. The definition should be an array of wanted size if there is more than one tracker in the station. Note, maximum trackers in a station is 20..

Basic examples

The following example illustrates the data type `mtparttracker`.

Example 1

```
PERS mtstationdata sdRobot;
PERS mtparttracker sdRobot_Tracker{2};
```

First a station definition for the robot and the corresponding tracker definition if part trackers should be used.

Note! The tracker definition must start with the station name and end with "_Tracker". In the example the robot av two grippers and therefore uses an array with 2 places which gives two trackers for the station. Also note that when the definition is made, the components of the defined data could be blank, those are filled later on when using the part tracker instructions.

Components

The data type `mtparttracker` has the following components:

`Station`

Data type: `string`

Name of the mtstationdata declaration, where the part tracker belongs to.

`Part`

Data type: `string`

Name of the mtpartdata declaration, which is currently connected to the part tracker.

`State`

Data type: `string`

Name of the mtpartstate declaration, which holds the state info of the part.

`Tool`

Data type: `string`

Name of the tool to be used for this part in a specific tracker of the station. This can remain empty, if not needed or if the tool information shall be handled by the part data in Part.

`PartLoad`

Data type: `string`

Name of the loaddata to be used for this part in a specific tracker of the station. This can remain empty, if not needed or if the partload information shall be handled by the part data in Part.

Continues on next page

G.3.6 mtparttracker - describes all data for a part tracker
RobotWare - OS
Continued

`UserDef`

Data type: `string`

String for user defined content, e.g. information, parameters, name of routines to be executed etcetera. The content is, as the name says, user-defined and so must be parsed by the application program.

G.3.7 mtrange - describes a robot axis range

Usage

`mtrange` is used to describe start (Min) and stop (Max) angles for a robot axis.

Components

The data type `mtrange` has the following components:

`Min`

Data type: `num`

Start value of the range.

`Max`

Data type: `num`

Stop value of the range.

G.3.8 mtstationdata - describes a station

Usage

`mtstationdata` is used to describe a station in the cell. The main key is to just have a station definition. The system will search for all definitions and handle those as stations. The components will not have any real usage in this release of the add-in.

Components

The data type `mtstationdata` has the following components:

`IOEnabled`

Data type: `string`

Name of the digital input or output through which the station is

selected or deselected. Here, the "high" state of the signal means "Station selected" and the "low" state means "Station deselected".

`IOInvert`

Data type: `bool`

Inversion flag, which will invert the meaning of the value in arguments IOEnabled.

`HMIEnabled`

Data type: `bool`

Station has been selected (TRUE) or deselected (FALSE) by the HMI.

Only if no signal name is defined, then the station can be selected or deselected through the HMI.

G.3.9 mttargetdata - describes a robot zone

Usage

`mttargetdata` is used to define the different zones in the cell. Normally one zone is defined for each station but it doesn't have to be like that. To define when the robot is within a specific zone a mtrange for axis 1 is always used, i.e. a start and stop value for axis 1. In addition there is up to four additional mtaxisrange that could be optionally used where user specify which axis and the range for that axis to also consider for the zone. Note that robot must be within all ranges (between start and stop for all the specified axis) to be in that zone. If an optional axisrange shouldn't be used, just state "0" for the axis number.

Basic examples

The following example illustrates the data type `mttargetdata`.

Example 1

```
CONST mttargetdata
    ZONE_HOME:=[1,"pHome",[-80,70],[3,[-30,30]],[4,[10,40]],[0,[0,0]],[0,[0,0]]];
```

In this example ZONE_HOME is given unique index number 1 and connects to via position "pHome". To be considered in the zone, the robot must have axis 1 between -80 and 70 degrees. And axis 3 between -30 to 30 degrees and axis 4 between 10 to 40 degrees.

Components

The data type `mttargetdata` has the following components:

`Index`

Data type: `num`

Unique index number for this mttargetdata, the number doesn't matter as long as it's unique.

`PosName`

Data type: `string`

The via position connected to this target zone. This gives an indirect connection between the target zone and the actual via position. Note that spelling is very important here!

`RangeAxis1`

Data type: `mtrange`

The range used for axis 1.

`Range2`

Data type: `mtaxisrange`

The axis and range for optional axis check. Set axis number to 0 if not using this.

`Range3`

Data type: `mtaxisrange`

The axis and range for optional axis check. Set axis number to 0 if not using this.

Continues on next page

G.3.9 mttargetdata - describes a robot zone
RobotWare - OS
Continued

`Range4`

Data type: `mtaxisrange`

The axis and range for optional axis check. Set axis number to 0 if not using this.

`Range5`

Data type: `mtaxisrange`

The axis and range for optional axis check. Set axis number to 0 if not using this.

Predefined data

The following symbolic constants of the data type `mttargetdata` are predefined and can be used when specifying a condition for instructions like `MT_GetCurrentZone`.

Value	Symbolic constant	Comment
[9999,"", [-9E9,9E9], [0,[0,0]], [0,[0,0]], [0,[0,0]], [0,[0,0]]]	ZONE_UNDEFINED	

ABB AB, Robotics
Robotics and Motion
S-721 68 VÄSTERÅS, Sweden
Telephone +46 (0) 21 344 400

ABB AS, Robotics
Robotics and Motion
Nordlysvegen 7, N-4340 BRYNE, Norway
Box 265, N-4349 BRYNE, Norway
Telephone: +47 22 87 2000

ABB Engineering (Shanghai) Ltd.
Robotics and Motion
No. 4528 Kangxin Highway
PuDong District
SHANGHAI 201319, China
Telephone: +86 21 6105 6666

ABB Inc.
Robotics and Motion
1250 Brown Road
Auburn Hills, MI 48326
USA
Telephone: +1 248 391 9000

abb.com/robotics

3HAC051768-001, Rev D, en