ROBOTICS

Application manual

Palletizing Template

Application manual

Palletizing Template 1.1.0

RobotWare 7

Document ID: Application manual Palletizing Template 1.1.0

Revision: F

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, damage 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

Overview of this manual

About this manual

This manual contains information about the application software Palletizing Template.

Usage

This manual can be used to find out what Palletizing Template is and how to use it. The manual also provides information about RAPID components and system parameters related to Palletizing Template, and examples of how to use them.

Who should read this manual?

This manual is mainly intended for robot programmers

Prerequisites

The reader should be familiar with:

- Industrial robots and their terminology
- The RAPID programming language
- System parameters and how to configure them

References

Reference	Document ID
Product manual OmniCore C30	*3HAC079399-001*
Product manual OmniCore C30	*3HAC060860-001*
Product manual OmniCore C90XT	*3HAC073706-001*
Product manual OmniCore V250XT	*3HAC087112-001*
Product manual OmniCore V400XT	*3HAC081697-001*
Application manual - Controller software OmniCore	*3HAC066554-001*
Operating manual - OmniCore	*3HAC065036-001*
Operating manual - RobotStudio	*3HAC032104-001*
Technical reference manual - RAPID Overview	*3HAC065040-001*
Technical reference manual - RAPID Instructions, Functions and Data types RW 7	*3HAC065038-001*

Technical reference manual - System parameters RW 7	*3HAC065041-001*

Revisions

Revision	Description
A	Release document for version 1.0.0
B	- Installation procedure updated - Supported feeders updated from 1 to 2 - obSheetMag renamed to obSheetMag1 - Updating the Palletizing Template Pattern Builder
C	Release document for version 1.0.2
D	Release document for version 1.1.0
E	- Updated SetPalletizeCycle instruction - Changed OPT_TOP_SAFE_X and OPT_TOP_SAFE_Y to an array.
F	Support for mechanical grippers added

Product documentation

Categories for user documentation from ABB Robotics

The user documentation from ABB Robotics is divided into a number of categories. This listing is based on the type of information in the documents, regardless of whether the products are standard or optional. All documents can be found via myABB Business Portal, www.myportal.abb.com.

Product manuals

Manipulators, controllers, DressPack/SpotPack, and most other hardware is delivered with a product manual that generally contains:

- Safety information.
- Installation and commissioning (descriptions of mechanical installation or electrical connections).
- Maintenance (descriptions of all required preventive maintenance procedures including intervals and expected lifetime of parts).
- Repair (descriptions of all recommended repair procedures including spare parts).
- Calibration.
- Decommissioning.
- Reference information (safety standards, unit conversions, screw joints, lists of tools).
- Spare parts list with corresponding figures (or references to separate spare parts lists).
- References to circuit diagrams.

Technical reference manuals

The technical reference manuals describe reference information for robotics products, for example lubrication, the RAPID language, and system parameters.

Application manuals

Specific applications (for example software or hardware options) are described in Application manuals. An application manual can describe one or several applications.

An application manual generally contains information about:

- The purpose of the application (what it does and when it is useful).
- What is included (for example cables, I/O boards, RAPID instructions, system parameters, software).
- How to install included or required hardware.
- How to use the application.
- Examples of how to use the application.

Operating manuals

The operating manuals describe hands-on handling of the products. The manuals are aimed at those having first-hand operational contact with the product, that is production cell operators, programmers, and troubleshooters.

Safety

Safety of personnel

A robot is heavy and extremely powerful regardless of its speed. A pause or long stop in movement can be followed by a fast hazardous movement. Even if a pattern of movement is predicted, a change in operation can be triggered by an external signal resulting in an unexpected movement. Therefore, it is important that all safety regulations are followed when entering safeguarded space.

Safety regulations

Before beginning work with the robot, make sure you are familiar with the safety regulations described in the manual Safety manual for robot Manipulator and IRC5 or OmniCore controller.

1 Introduction

Introduction

Palletizing Template is a robot application software package for palletizing applications. Palletizing Template is designed for simple palletizing cells, used in for example end of line palletizers.

End of line palletizers typically require a small footprint, which means that there are usually one or two pallets and one feeder.

Typically the capacity of such a system is relatively low, so handling one box at the time is sufficient. Palletizing Template however supports handling of up to 3 boxes per cycle.

RAPID framework

Palletizing Template is supplied with a framework of RAPID code that is set-up for a minimal palletizing system. Due to the open-source nature of the framework, the application can easily be modified to suit the actual installation.

Modification typically consists of setting up gripper, infeeder and pallet control. This will also imply that system parameters for the IO configuration may need to be added.

Background tasks can also be added to control feeder systems. For simple system the use of an additional PLC can be avoided by controlling for example an infeeder from a background task. This software is dedicated and needs to be developed for the installation. It’s not part of the Palletizing Template package

Stack configurator

Palletizing Template consists of a stack configurator, which runs on the robots FlexPendant. With this configurator pallet pattern configurations can easily be selected.

The stack configurator uses a wizard-like style to configure the patterns. In 6 steps the user is guided through the process of creating the patterns for the individual layers on the pallet. The focus in this wizard is put on the product to be handled by the robot and not on typical robot related issues.

Label positions can be set for the boxes. In this way the pattern can be built with label positions as much as possible on the outside of the stack.

When necessary, slip sheets can be added between layers.

RobotWare

The RAPID framework supplied with Palletizing Template runs on a robot

controller with RobotWare 7.15 or higher and supports any 4 or 6 axis robot. The software is optimized for use on the Gofa cobot.

The RAPID framework contains a basic program for a basic pick and place cycle, with one feeder and up to four pallets. The RAPID framework can be modified to suit the users' requirements for the installation. The recipes created by the stack configurator are used to physically build the stacks.

Gripper support

To handle the boxes two types of grippers are supported. These are vacuum gripper systems that pick boxes from the top as well as mechanical grippers like clamp, fork grippers etc.

Vacuum grippers pick boxes at the top of the box without any obstructions to any of the sides of the boxes. This gripper type allows placement of boxes at any orientation against other boxes already present on the pallet layer.

Mechanical grippers clamp or hold boxes on any of the sides or at the bottom. These types of grippers have a mechanical structure at one side of the box that limits the available orientations when placing a box against boxes already present on the pallet layer.

The used gripper type is configured during installation. The Palletizing Template user interface is identical for vacuum and mechanical grippers.

Palletizing Template generates positions that require at least +/- 180° rotation of the gripper. The gripper should be able to perform this rotation, without any limitation, anywhere in the necessary workrange. The so called “helicoptering” (large rotation from pick to place and vice) is avoided as much as possible but can still occur.

The routing of the cables to the gripper should be such that it can handle these rotations.

Box handling

Palletizing Template can be configured to handle up to 3 boxes in a single pick – place cycle. When handling more than one box at the time the feeder should be capable of supplying the correct number of boxes for each place cycle. The number of boxes for each cycle will be calculated automatically by Palletizing Template.

The boxes can be fed to the system either long side or short side lead, for both single box and multi box handling.

For readability purposes the term box will be used, which indicates either one box or a combination of up to 3 boxes.

Production screen

Palletizing Template provides a production screen from which recipes can be selected, started and stopped.

During palletizing the robot can be halted when required. When halted, the robot can be sent to the home position to provide easy access to the pallet to, for example, remove a damaged or lost box. From the home position the palletizing can either continue or be aborted.

The current stacking status of the pallet is also shown. The status includes a preview of the actual pattern being built and which box is currently being handled by the robot.

The production screen also provides means to preview recipes, optimize the robot as well as stacking performance.

Product and motion tuning

A key feature of Palletizing Template is that tolerances in boxes, in all three dimensions, can be compensated for while running production (on the fly), without the need of recreating the pattern or aborting to current stack.

Motion tuning is also possible where the robot speed and acceleration can be optimized for a specific recipe. For, for example large boxes the acceleration and / or speed can be reduced to handle the box with more stability.

Motion tuning data are stored with the recipe.

Palletizing - depalletizing

Palletizing Template can be configured to both palletize and depalletize pallet stacks.

2 Installation

2.1 System requirements

Standard options

Palletizing Template can run on any OmniCore controller with RW 7.15 or higher. The minimum requirements for the robot system are:

Option	Description
IO unit	IO unit (of any type) for the control of the gripper

Additional options

When the robot controller is used to control the feeder the following option is also required:

Option	Description
IO unit	Additional IO unit(s) for the control of the feeder

The following software option is required for the robot controller

Option	Description
623-1: Multitasking	Multitasking is typically used to control a feeder from a background task.

Limitations

When handling multiple boxes Palletizing Template uses the single pick - single place approach. Multi place (i.e. placing the boxes in gripper in several individual steps on the pallet) is not supported.

2.2 Hardware installation

Description

Palletizing Template is a software package, so no hardware installation is required.

For control of the feeder(s) specific hardware may need to be installed, however this is recipe specific.

Gripper design

Pick and place orientations are calculated by Palletizing Template, based on algorithms. As a result of this axis 6 (6 axis robots) or axis 4 (4 axis robots) may turn over a range of at least +/- 180° (>360° total). So, make sure that the gripper cabling is routed in such way that the gripper can turn freely over at least +/- 180°.

Palletizing Template will try to make axis 6 turn a little as possible. But the fact that the full 360° of the workrange of axis 6 is used can’t be avoided.

2.3 Software installation

Description

The Palletizing Template framework is an open-source framework. This implies that the user may need to adapt Palletizing Template to suit the actual installation. All RAPID routines can be modified to fit the actual installation. However, some of the RAPID is mandatory and should not be removed.

The same applies to the system parameters. These contain a minimum set of parameters required to make the system run.

It’s up to the user to make the required modifications.

Installation

Palletizing Template is supplied as an add-in for RobotWare 7. Before starting software installation make sure all system requirements (as described in chapter 2.1) are installed and functioning properly.

For the installation of the software basic RAPID programming knowledge is required.

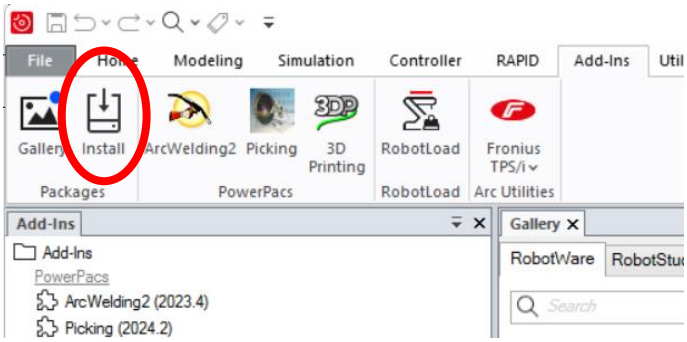
To install Palletizing Template the following actions need to be taken:

Step	Action
1	Install the Palletizing Template add-in in RobotStudio.
2	Installation of the RAPID framework.
3	Configuring the gripper.
4	Calibration of the pallet position(s).
5	Calibration of the feeder(s).
6	Configuring the pallet settings.
7	Configuring the feeder settings.
9	Configuring the gripper dimensions.

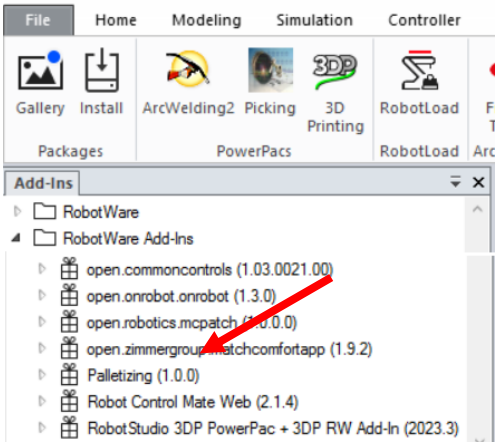
Installation procedure

To install Palletizing Template download the “Palletizing-1.0.0.rspak” file from GitHub. The version number 1.0.0 will change in the future when new versions are made available.

To install the package, open RobotStudio. From the Add-ins tab select the install button.

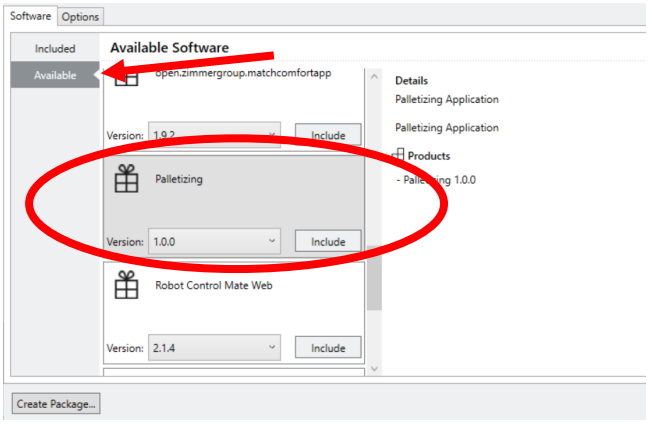


Browse to the “Palletizing-1.0.0.rspak” file and open this file. Palletizing Template will now be installed in RobotStudio. Under the RobotWare Add-ins you should now have the Palletize Template package installed.



Building a robot controller

Once Palletizing Template is installed the Palletizing Template can be added as a RobotWare option to any OmniCore controlled robot. When setting up the RobotWare the Palletizing Template can be chosen from the list of available options.



Include the option and reboot the robot controller. After the restart the following software is installed:

File or folder	Description
HOME: folder	A subfolder called “Palletizing_Files” with a demo recipe, demo pattern and language files is created.
System parameters	Required basic system parameters.
RAPID code	RAPID program and system modules
FlexPendant app	The Palletizing Template FlexPendant app

In the next paragraphs more detailed description is given on the installed items

HOME folder contents

The following content is installed in the HOME:\Palletzing_Files folder:

File or folder	Description
Languages	All Palletize Template FlexPendant texts in different languages. Languages are in JSON format can be added if not yet supported. The file name starts with “lang “ followed by the initials of the country as used in the RobotWare. The extension must be “.json”
Patterns	The patterns that are used to configure the recipes in Palletizing Template are based on libraries. The library files can be found in this folder.
Recipes	Contains the Palletizing Template user created recipes and matching tune data files.

System parameters

The system parameters consist mainly of a standard set of IO definitions required to run the Palletize Template in its basic form. The functionality of the individual IO signals are described in chapter 3.

RAPID program and system modules

The Palletizing Template consist of program and system modules.

The program modules contain the runtime environment for the palletizing cycle. The modules mentioned can be modified to fit the actual installation. Care must be taken not to remove instructions that are required for the correct functionality of Palletizing Template.

Module	Description
Palletizing Template	This is the main module of the Palletizing Template application. This module contains the main and home run routine of Palletizing Template.
RunCycle	This module contains the execution of the palletizing cycle for picking and placing boxes and slip sheets.
Settings	This module contains data with the installation specific settings for Palletizing Template.
Gripper	This module contains basic procedures for vacuum activation, deactivation and vacuum check. The predefined code manages a gripper with three zones for a MultiPick application with up to three boxes.
LiftKit	This module contains routines to run a liftkit.

The system modules contain the core of the Palletizing Template. It is recommended not to make any changes to these modules unless there is a defect or missing functionality. In almost all cases a solution can be implemented in the program modules.

In case a defect or missing functionality is implemented in the system module(s) please notify ABB so the modification can be done in a new official release.

The risk of making changs to the system modules is when a new official version of the Palletizing Template is released, the locally changed modules is no longer compatible with the official one.

The following system modules are automatically installed:

Module	Description
AppDate	This module contains the variables that are used for the internal communication between the RAPID code and the FlexPendant app. Do not change or remove any of the variables.

FileHandling	This module contains routine for loading the recipes.
Init	This module contains initialistaion routines declaration of Palletizing Template internal variables and data types.
PatternCalc	Pattern calculation routines for interpreting the pattern formula's and setting up the cycleto targets.
PickPlaceItem	Pattern calculation routines for converting the pattern formula's to targets.
TXT_ENG	Text file with texts for RAPD generated texts like error messages

Note

Avoid making changes to the system modules for compatibility reasons. If changes are required inform ABB about these changes, other ABB can't guarantee compatibility with future releases and / or bug fixes.

FlexPendant app

All software required to run the FlexPendant app in located in the "WebApps\Palletizing" folder.

The FlexPendant app is also open source, so it can be modified if needed. Knowledge of JavaScript, html and CSS is required. The structure and coding of the FlexPendant app is not part of this document.

3 System configuration

Description

The system configuration defines the way that the palletizing cell is build. To configure the system several Palletizing Template parameters need to be set up. Perform the following steps to set up the system

Step	Action
1	Set up the pallet parameters like workobject calibration, layer building direction, path settings, pallet status and palletizing mode.
2	Set up the feeder parameters like workobject calibration, box alignment and path settings.
3	Set up the slip sheet tray parameters like workobject calibration.
4	Set up the gripper parameters like gripper type and TCP position.

The different parameters are described in the next chapters.

As Palletizing Template can be configured for multi box handling, in the remainder of this manual “box” is used to indicate the handling of either one box or a group of up to 3 boxes.

Note

When in this manual a “box” is mentioned, this should be interpreted as a group of up to 3 boxes.

3.1 Pallet definitions

Introduction

For placing products on the pallet the following pallet related settings need to be configured:

Setting	Description
Pallet workobject	Describes how to use and calibrate the workobjects for each pallet location
Layer building direction	Describes the way the layer is buil by the robot.
Path settings	Defines how the movement towards and from the place position is to be executed.
Palletizing mode	Defines if the pallet is to be palletized or depalletized

3.1.1 Pallet workobject

Pallet workobject

The pallet locations require calibration for each pallet. The RAPID framework supports four pallet positions. In the RAPID framework the four pallets are referenced as pallet 1 to pallet 4. It’s up to the user to decide which physical pallet location is chosen for pallet 1, pallet 2 and so on.

The calibration for the pallet consists of teaching a workobject for each pallet.

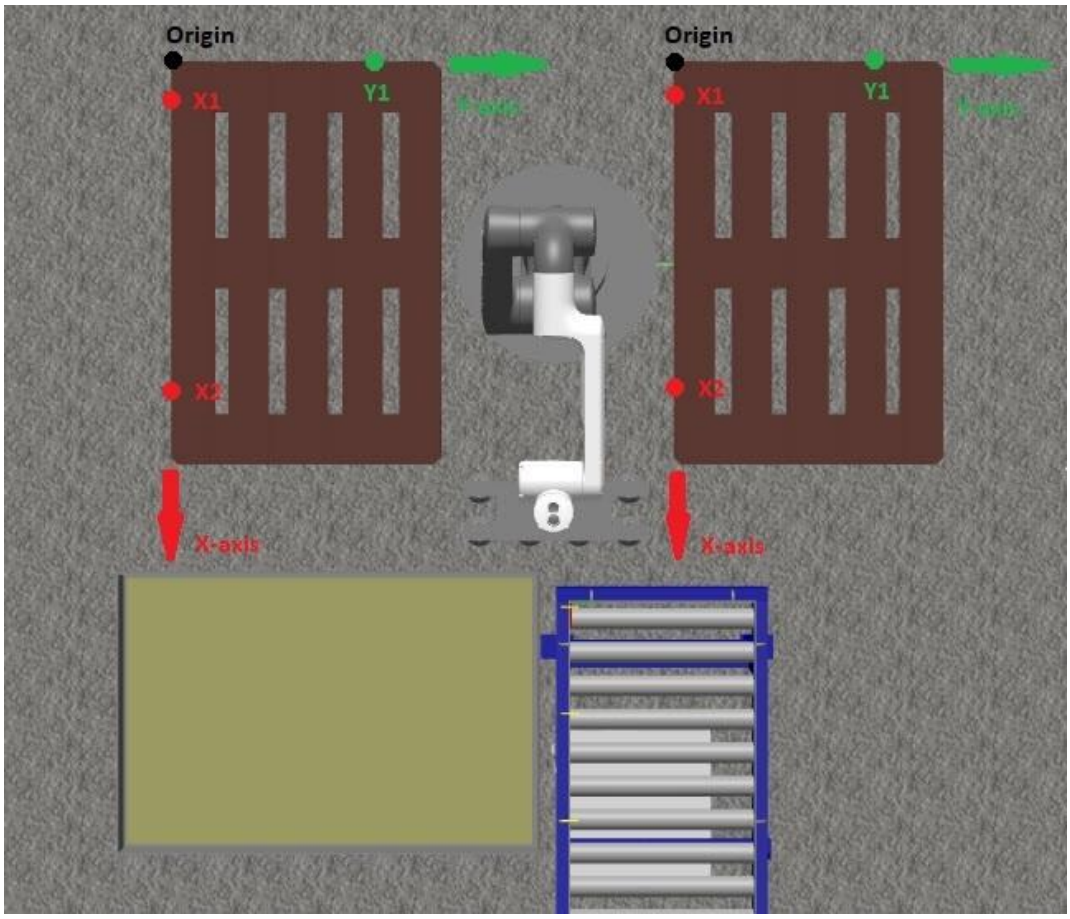
Workobject	Description
obPallet1	Workobject with origin of the first pallet
obPallet2	Workobject with origin of the second pallet
obPallet3	Workobject with origin of the third pallet
obPallet4	Workobject with origin of the fourth pallet

Pallet workobject definition

The position of the origin of each workobject should be defined as shown in the image below. By definition the position and orientation of the workobject must be chosen such that the X-axis runs along the long side of the pallet and the Y-axis runs along the short side of the pallet. As a result the Z axis will point upwards.

The workobject should be defined at the bottom of the pallet, so on floor or conveyor level.

In the image below the direction of the X and Y axis are shown in black.



Note

Make sure that the positive X-axis runs along the long side of the pallet and the positive Y-axis runs along the short side of the pallet as shown in the image above.

Pallet workobject calibration

The workobject can be calibrated from the FlexPendant. Refer to the 3HAC065036-001 Operating manual OmniCore under Defining a workobject and Defining the work object coordinate system for more information about workobject calibration.

Before calibration starts the robot must be equipped with a calibration tool, with its TCP accurately defined and active. Make sure that the calibration tool can reach the three positions as shown in red in the image above. These positions must be reachable at the bottom of the pallet.

To calibrate the workobject the 3 point method for the user frame should be chosen. Move the calibration pin to the three points indicated by X1, X2 and Y1 in

the image above. X1 and X2 must be taken at the fixed long side of the pallet. Y2 must be taken at the short side of the pallet. The three positions must be accurately aligned with the pallet sides. To get the best accuracy the distance between the three positions should be as far apart as possible.

When the calibration process in finished the origin of the pallet workobject should be located in one corner of the pallet as shown in the image above.

Note

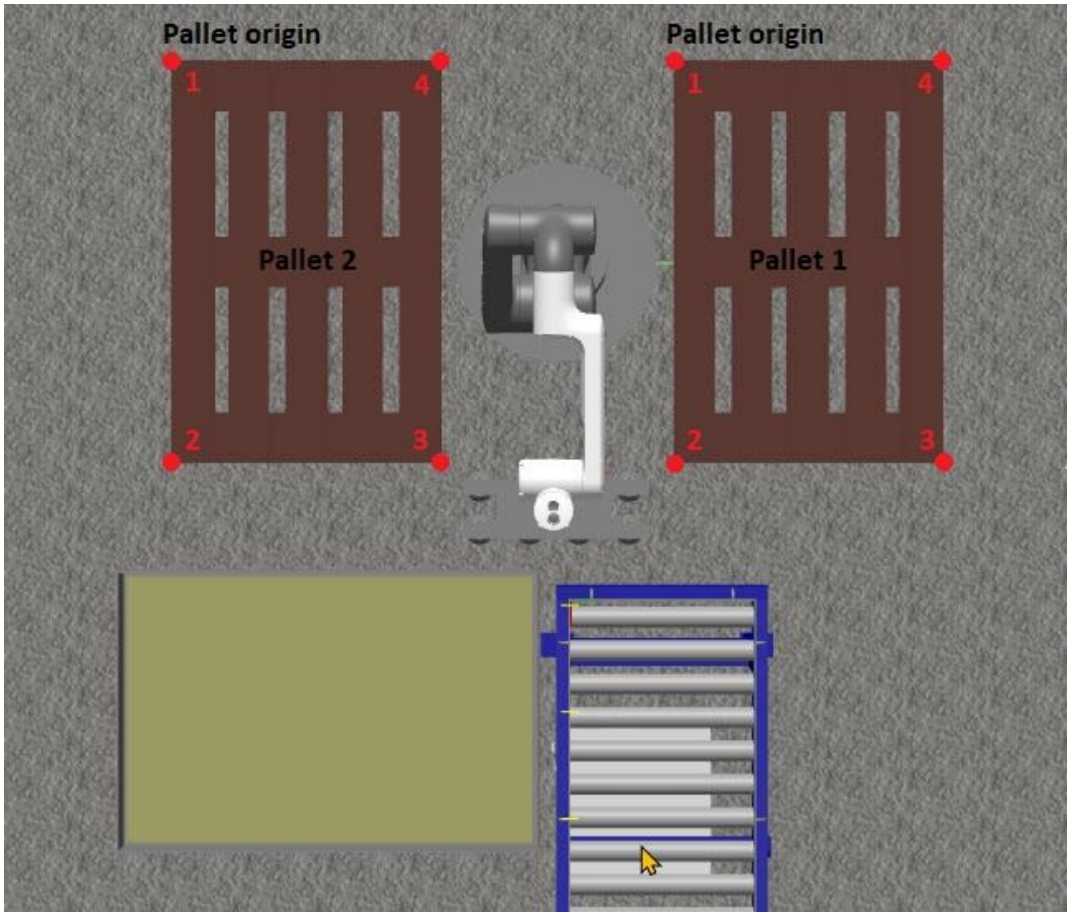
The calibration positions for the pallet must be taught at the bottom of the pallet, so at floor or conveyor level.

3.1.2 Layer building direction

Description

The layer building direction describes from which corner the pallet the robot starts building the pattern The layer is always build up starting from one corner towards the opposite corner of the pallet. In the settings it can be chosen from which corner the robot should start placing boxes for each pallet.

The start corner is indicated by a number between 1 and 4 and is situated as shown in the image below. Corner 1 is always in the origin of the pallet. The remaining corners are numbered in counter clockwise order.



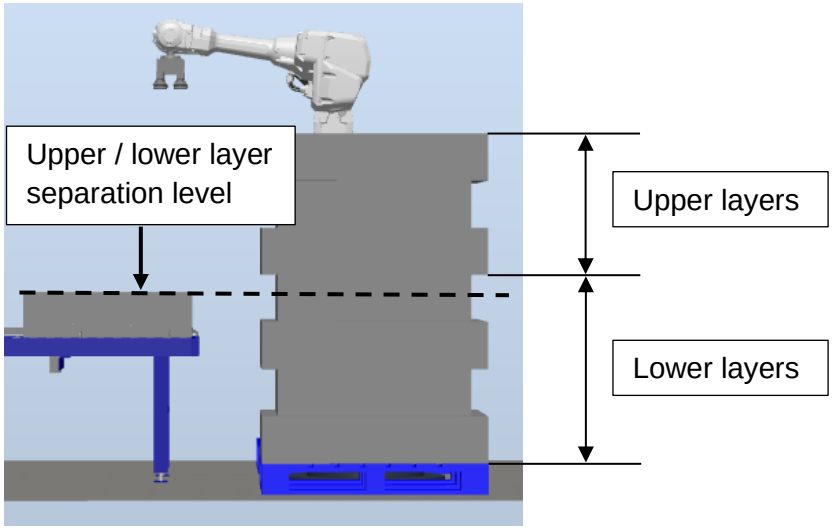
Typically, on the lower layers the robot will build the layers from the inside out. In the cell layout shown in the image above this is start corner 1 for pallet 1 and start corner 4 for pallet 2.

On the higher levels the stack is usually built from outside in. This gives the best reach on the top level. In the example of the image above this is corner 3 for pallet 1 and corner 2 for pallet 2.

Upper and lower layers

To maximize the robots workarea different corners can be chosen for the lower and upper layers. The lower layers are the pallet layers that are located below the level of the feeder plus one box height as shown in the image below. The upper layers are located above this level.

The switch between lower and upper layers is automatically calculated in the RAPID framework.



3.1.2.1 Start corner Lower layers

Description

The variable CORNER_LOWER_LAYERS indicates from which corner the layer is built for the lower layers on the pallet as shown in the image above.

The CORNER_LOWER_LAYERS variable is declared in the Settings module.

Basic example

The following example illustrates the CORNER_LOWER_LAYERS variable

Example 1

```
PERS num CORNER_LOWER_LAYERS {4}:=
[
  1
  4,
  1,
  1
];
```

In this example the start corner for the lower layers is set to 1 for pallets 1, 3 and 4. Start corner 3 is used for pallet 2.

3.1.2.2 Start corner upper layers

Description

The variable CORNER_UPPER_LAYERS indicates from which corner the layer is built for the upper layers on the pallet as shown in the image above

The CORNER_UPPER_LAYERS variable is declared in the Settings module.

Basic example

The following example illustrates the CORNER_UPPER_LAYERS variable

Example 1

```
PERS num CORNER_UPPER_LAYERS {4}:=
[
  3
  2,
  3,
  3
];
```

In this example the start corner for the lower layers is set to 3 for pallets 1, 3 and 4. Start corner 2 is used for pallet 2.

3.1.3 Path settings

Introduction

In order for the robot to execute a collision free path to and from the place position, an approach and depart path must be executed.

The approach path, where the robot holds a box, consists typically of two motion instructions. The position of these motion instructions is based on offsets given from the place position.

The depart path is typically one motion instruction, defined as an offset from place position.

In the next chapters a detailed description is given for both the approach and depart path.

3.1.3.1 Place approach offset

Description

The approach offsets define the location of the approach fly by position that is located above the place position. The location of the approach position is defined as offsets from the place position.

The offsets are declared in the variables PLACE_APPROACH_X, PLACE_APPROACH_Y and PLACE_APPROACH_Z. These offsets can be adapted to the cycle and gripper type.

The PLACE_APPROACH_X and PLACE_APPROACH_Y offset define the clearance distance between the box to place and the boxes already placed on the pallet, in horizontal plane. These offsets are defined in the tool coordinate system. The PLACE_APPROACH_X defines a movement away from the box. The PLACE_APPROACH_Y defines a movement along the side of the box. As the place position can be approached from different directions both offsets should be set.

The approach movement to the place position consists typically of two movements before moving to the place position. In the first position the robot moves at sufficient height above the layer already placed, with the offset in horizontal plane. Then the robot moves to the second position straight down to just above the place position. The same offsets in the horizontal plane are kept making sure the robot moves down besides the boxes already placed on the pallet while keeping the same clearance in the horizontal plane. From here the robot will move to the place position.

Two offsets are defined for the height. PLACE_APPROACH_Z defines the clearance height at which the robot should move over the boxes already placed on the layer. The LOW_APPROACH_OFFS is the height offset from the place position.

The PLACE_APPROACH_Z is overridden when the feeder depart position is higher than the place approach position. This is typically the case at the lower layers of the pallet. In this case the PLACE_APPROACH_Z will be set such that the same height is kept as the feeder depart position.

The offsets are declared in the Settings module.

Basic example

The following example illustrates the PLACE_APPROACH_X, PLACE_APPROACH_Y, PLACE_APPROACH_Z and LOW_APPROACH_OFFS variables

Example 1

```
CONST num PLACE_APPROACH_X:=50;
CONST num PLACE_APPROACH_Y:=50;
CONST num PLACE_APPROACH_Z:=150;
CONST num LOW_APPROACH_OFFS:=75;
```

In this example the robot approaches the place position with a distance of 50 mm in the horizontal plane.

The robot moves with at height clearance of 150 mm over the boxes already placed on the pallet. For there the robot moves down to 75 mm above the pallet.

3.1.3.2 Place depart offset

Description

The depart offset is the position to where the robot moves after placing a box on the pallet. For vacuum grippers this is typically only a Z offset as the robot can move freely upwards.

The offsets should be set in the variables PLACE_DEPART_X, PLACE_DEPART_Y and PLACE_DEPART_Z. The offsets are in mm.

The PLACE_DEPART_X, PLACE_DEPART_Y and PLACE_DEPART_Z offsets are relative to the tool coordinate system (in negative direction).

The PLACE_DEPART_Z is overridden when the feeder approach position is higher than the place depart position. This is typically the case at the lower layers of the pallet. In this case the PLACE_APPROACH_Z will be set such that the same height is kept as the feeder approach position.

The offsets are declared in the Settings module.

Basic example

The following example illustrates the PLACE_ DEPART _X, PLACE_ DEPART _Y and PLACE_ DEPART _Z variables

Example 1

```
CONST num PLACE_DEPART_X:=0;
CONST num PLACE_DEPART_Y:=0;
CONST num PLACE_DEPART_Z:=150;
```

In this example the robot moves straight up over a distance of 150 mm after placing the box on the pallet.

3.1.4 Pallet Status

Introduction

When the robot has finished the stack on a pallet, the user needs to be notified. Once the full pallet is removed from the cell and an empty pallet has been placed back, the user has to notify Palletizing Template that the empty pallet is present.

There are two possible scenarios to report that a new pallet has been placed in the cell. This can either be from an external IO device or from the Production window of the Palletizing Template FlexPendant app.

3.1.4.1 Allow app release

Description

In the Production window there is a button foreseen for each of the pallet position where the status of the pallet is indicated. This functionality of these buttons can be set for each button individually.

The used functionality can be set in the array AllowAppRelease. This array has four elements with one element for each pallet location.

Each element in this array can have three different values:

Variable	Description	Value
APP_NO_BUTTON	The new pallet report is done from an external IO device. No indication of pallet status is shown in the Production window on the FlexPendant. This status is also used if the corresponding pallet is not foreseen.	0
APP_STATUS_ONLY	The new pallet report is done from an external device. However, the status of the pallet is shown in the Production window in the Palletizing Template FlexPendant app (new pallet, pallet running and pallet full). The button in the Production window will not be active.	1
APP_SINGLE_CONFIRM	The new pallet can be reported from the Production window in the Palletizing Template FlexPendant app. The new pallet can also be reported from an external IO device.	2

A pallet can have four different states:

State	Description
None	Pallet state is unknown. This typically applies if the pallet is not used in the installation.
New	A new pallet has been reported, but the pallet is not active i.e. the robot is palletizing on another pallet.
Running	The pallet is active i.e. the robot is palletizing on this pallet.
Full	The pallet is full

The AllowAppRelease array is declared in the Settings module.

FlexPendant new pallet report

When a new pallet is to be reported from the FlexPendant the to the pallet corresponding element in the AllowAppRelease array must be set to APP_SINGLE_CONFIRM. For these pallet locations a button will appear in the production window (see chapter 4.2.1.4). By means of this button a new pallet can be reported.

Basic example

The following example illustrates the AllowAppRelease variable

Example 1

```
CONST num AllowAppRelease{4}:=
[
  APP_SINGLE_CONFIRM,
  APP_SINGLE_CONFIRM,
  APP_NO_BUTTON,
  APP_NO_BUTTON
];
```

In this example the new pallet for location 1 and 2 are to be reported from the FlexPendant app with a single confirmation. The pallets at location 3 and 4 are not used.

3.1.4.2 External new pallet report

Description

The presence of a new pallet can also be reported from external devices such as a PLC or an operator panel as well. The new pallet is then usually reported through the IO system of the robot.

Reporting new pallets over IO is always possible, independent of the Allow app release setting (see chapter 3.1.4.1).

To report a new pallet four digital inputs and four digital outputs are predefined in the system parameters of the robot controller

Four digital outputs are foreseen to report a full pallet to the user. The output goes high when the respective pallet is full. When the pallet is reported as new or when the pallet is running the output will be set low. These outputs can for example be used to control a status light close to the pallet location.

Output	Description
doPalletFull1	Pallet full indicator for pallet 1 0 - Pallet is reported as new or pallet is active 1 - Pallet is full
doPalletFull2	As above for pallet 2
doPalletFull3	As above for pallet 3
doPalletFull4	As above for pallet 4

To report that a new pallet is placed in the cel four inputs are foreseen. With a high pulse signal a full pallet is reported to Palletizing Template. Each input represents one pallet.

Output	Description
diNewPallet1	Report new pallet for pallet 1. A high pulse is required to report the new pallet
diNewPallet2	As above for pallet 2
diNewPallet3	As above for pallet 3
diNewPallet4	As above for pallet 4

A new pallet report will only be accepted if the pallet is full. If the pallet is in running state, the new pallet report is ignored.

There is no confirmation foreseen for the external pallet report. When the signal is given Palletizing Template assumes that the pallet is empty and can thus start palletizing immediately. It’s the user’s responsibility that the full pallet is replaced by an empty one before the new pallet is reported.

If needed additional sensors can be added to verify that the pallet is empty, thus avoiding the accidental new pallet reports. The extra sensors need to be implemented in the RAPID code or system parameters by the user.

The inputs are only active when the robot program is running.

Note

A new pallet can only be reported when the pallet is full. If a new pallet is reported while a pallet is running, it will be ignored.

Note

It’s the users responsibility that an pallet is present when a new pallet is reported.

3.1.5 Palletizing mode

Description

Palletizing Template can be configured for both palletizing and depalletizing functionality. In palletizing mode, the robot will pick boxes from a infeeder and place these on a pallet. In depalletize mode this is done in opposite direction, i.e. the robot picks boxes from the pallet and places these on an outfeeder.

The palletizing mode must be specified in the variable Palletize. Palletize is an array with four elements. Each element represents a pallet position.

To set the respective pallet in palletize mode three values can be used

Mode / CONST	Description	Value
SYS_NOT_USED	The pallet is not used in the current installation.	0
SYS_PALLETIZE	The pallet is used for palletizing.	1
SYS_DEPALLETIZE	The pallet is used for depalletizing.	2

Note

In order to use the depalletizing function the RAPID needs to be reversed and thoroughly tested. The supplied RAPID framework doesn't foresee the depalletizing function.

The Palletize variable is declared in the Settings module.

Basic example

The following example illustrates the Palletize variable

Example 1

```
PERS num Palletize{4}:=
[
  SYS_PALLETIZE
  SYS_PALLETIZE,
  SYS_NOT_USED,
  SYS_NOT_USED
];
```

In this example the pallet 1 and 2 are used for palletizing. Pallet 3 and 4 are not used.

3.2 Feeder definitions

Introduction

For picking products from the feeder the following settings can be modified:

Setting	Description
Feeder workobject	Describes how the to use and calibrate the workobject(s) for the feeder(s)
Box alignment	Defines how the box is mechanically aligned on the feeder
Path settings	Defines how the movement towards and from the place position is to be executed.

3.2.1 Feeder workobject

Feeder workobject

The feeder requires a calibrated workobject. Palletizing Template supports two feeders.

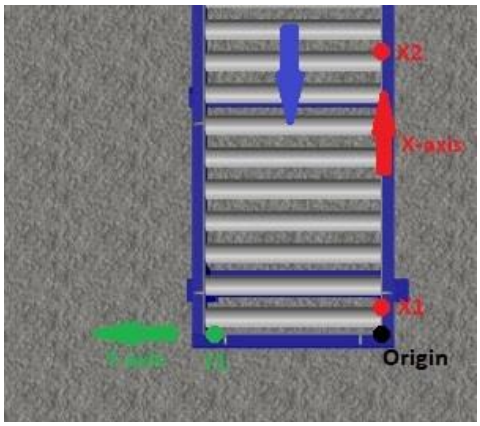
Workobject	Description
obFeeder1	Workobject with origin on the first feeder
obFeeder2	Workobject with origin on the second feeder

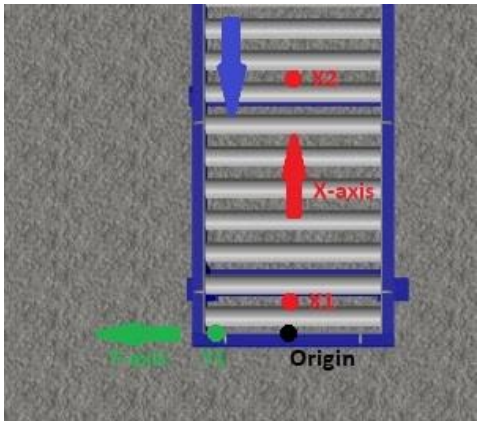
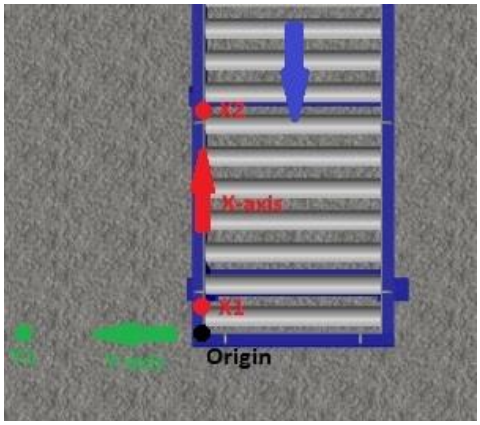
Feeder workobject definition

The origin of the feeder workobject can be defined in three different ways. The origin must be chosen according to how the box is aligned on the feeder. If the box is aligned on the left side (facing the front of the conveyor) the workobject should also be calibrated on the left side of the feeder. The same applies for center and the right-side alignment where the workobject is to be calibrated in the resp. center or the right side of the feeder.

The position of the origin of each workobject should be defined as shown in the image below. By definition the position and orientation of the workobject must be chosen such that the X-axis runs in opposite of the travel direction of the feeder and the Y-axis runs parallel to the front of the feeder. As a result, the Z axis will point upwards.

In the image below the direction of the X and Y axis are shown in black.

Alignment	Image
Right	

Center	
Left	

Note

For the left hand alignment the Y1 position needs to be located outside the feeder.

Note

Make sure that the X-axis points in the opposite direction of the feeder direction and that the Y-axis is always points to the left (when facing the feeder) as shown in the image above.

Feeder workobject calibration

The workobject can be calibrated from the FlexPendant. Refer to the Operating manual OmniCore under Defining a workobject and Defining the work object coordinate system for more information about workobject calibration.

Before calibration starts the robot must be equipped with a calibration tool, with its TCP accurately defined and active. Make sure that the calibration tool can reach the three positions as shown in red in the image above. These positions must be reachable at feeder level.

To calibrate the workobject the 3 point method for the user frame should be chosen. Move the calibration pin to the three points indicated by X1, X2 and Y1 in the image above. X1 and X2 must be taken in the opposite direction of the feeder. Y2 must be taken at the front of the feeder. Depending on the location of the workobject the Y1 position should be taken outside the feeder. The three positions must be accurately aligned as shown in the image above. To get the best accuracy the distance between the three positions should be as far apart as possible.

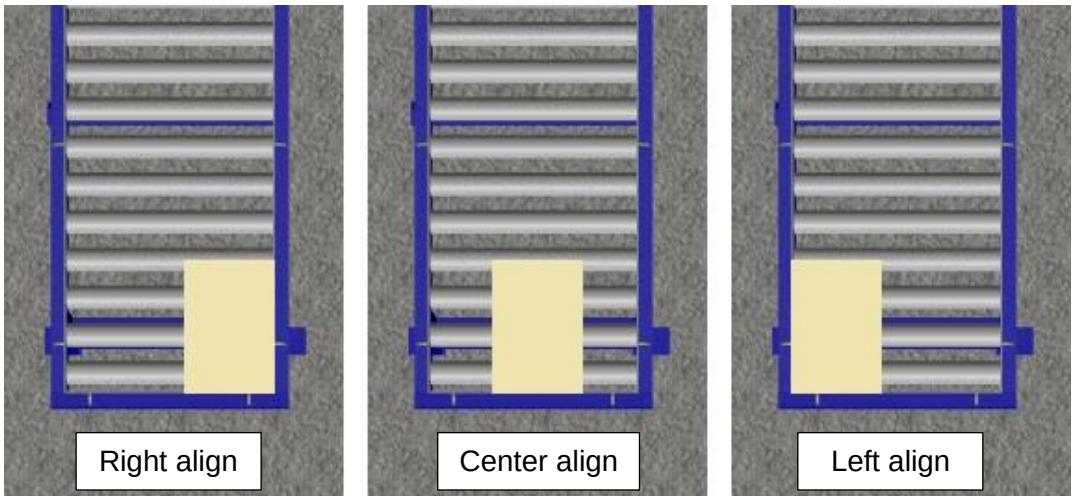
When the calibration process in finished the origin of the pallet workobject should be located in one corner of the pallet as shown in green in the image above.

3.2.2 Box alignment

Description

The pick workobject location defines where the pick workobject is located on the feeder. The Pick location should be defined in the RAPID variable PickObjLocation. This variable is declared in the AppData module.

This selection depends on where the fixed corner of the box is located on the feeder. The feeder will typically align the boxes on one of the three sides as shown in the image below.



The feeder workobject must be defined according to this setting. See chapter 3.2.1 for a detailed description of the workobject definition.

The PickObjLocation variable should be set to the following three values.

Side / CONST	Description	Value
PICK_OBJ_RIGHTSIDE	The boxes are aligned at the right-hand side of the feeder. The workobject will also need to be located on the right-hand side of the feeder, as shown in the right picture of the image above.	1
PICK_OBJ_CENTER	The workobject is located in the center of the feeder. As shown in the center picture of the image above.	2
PICK_OBJ_LEFTSIDE	The workobject is located on the left hand side of the feeder. As shown in the left picture of the image above.	3

The PickObjLocation is declared in the Settings module.

Basic example

The following example illustrates the PickObjLocation variable.

Example 1

```
CONST num PickObjLocation:= PICK_OBJ_RIGHTSIDE;
```

The feeder aligns the box to the right side (seen from the front of the feeder).

3.2.3 Path settings

Introduction

In order for the robot to execute a collision free path to and from the pick position, an approach and depart path must be executed.

Both approach and depart path consist of one motion instruction, defined as an offset from pick position.

In the next chapters a detailed description is given for both the approach and depart path.

3.2.3.1 Pick approach offset

Description

The approach offsets define the location of the approach fly by position that is located above the pick position. The location of the approach position is defined as offsets from the pick position.

The offsets should be set in the variables PICK_APPROACH_X, PICK_APPROACH_Y and PICK_APPROACH_Z. The offsets are in mm.

The PICK_APPROACH_X, PICK_APPROACH_Y and PICK_APPROACH_Z offsets are relative to the tool coordinate system (in negative direction).

The PICK_APPROACH_Z is overridden when the pallet depart position is higher than the feeder approach position. This is typically the case at the upper layers of the pallet. In this case the PICK_APPROACH_Z will be set such that the same height is kept as the pallet depart position.

The offsets are declared in the Settings module.

Basic example

The following example illustrates the PICK_APPROACH_X, PICK_APPROACH_Y and PICK_APPROACH_Z variables

Example 1

```
CONST num PICK_APPROACH_X:=0;
CONST num PICK_APPROACH_Y:=0;
CONST num PICK_APPROACH_Z:=150;
```

In this example the robot moves to a position 150 mm straight above the box pick position. The settings in this example are typically used for vacuum grippers.

Example 2

```
CONST num PICK_APPROACH_X:=-50;
CONST num PICK_APPROACH_Y:=0;
CONST num PICK_APPROACH_Z:=150;
```

In this example the robot moves to a position 150 mm above the pick position and with a 50 mm clearance from the side of the box. The settings in this example are typically used for mechanical grippers.

3.2.3.2 Pick depart offset

Description

The approach offset defines the location of the approach fly by position that is located above the pick position. The location of the approach position is defined as offsets from the pick position.

The offsets should be set in the variables PICK_DEPART_X, PICK_DEPART_Y and PICK_DEPART_Z. The offsets are in mm.

The PICK_DEPART_X, PICK_DEPART_Y and PICK_DEPART_Z offsets are relative to the tool coordinate system (in negative direction).

The PICK_DEPART_Z is overridden when the pallet approach position is higher than the feeder depart position. This is typically the case at the upper layers of the pallet. In this case the PICK_DEPART_Z will be set such that the same height is kept as the pallet approach position.

The offsets are declared in the Settings module.

Basic example

The following example illustrates the PICK_DEPART_X, PICK_DEPART_Y and PICK_DEPART_Z variables

Example 1

```
CONST num PICK_DEPART_X:=0;
CONST num PICK_DEPART_Y:=0;
CONST num PICK_DEPART_Z:=150;
```

In this example the robot moves straight up over a distance of 150 mm after picking the box from the feeder.

3.3 Slip sheet tray definitions

Introduction

For picking slip sheets from the slip sheet tray the following settings can be modified:

Setting	Description
Slip sheet tray workobject	Describes how the to use and calibrate the workobject for the slip sheet tray

3.3.1 Slip sheet tray workobject

Slip sheet tray workobject

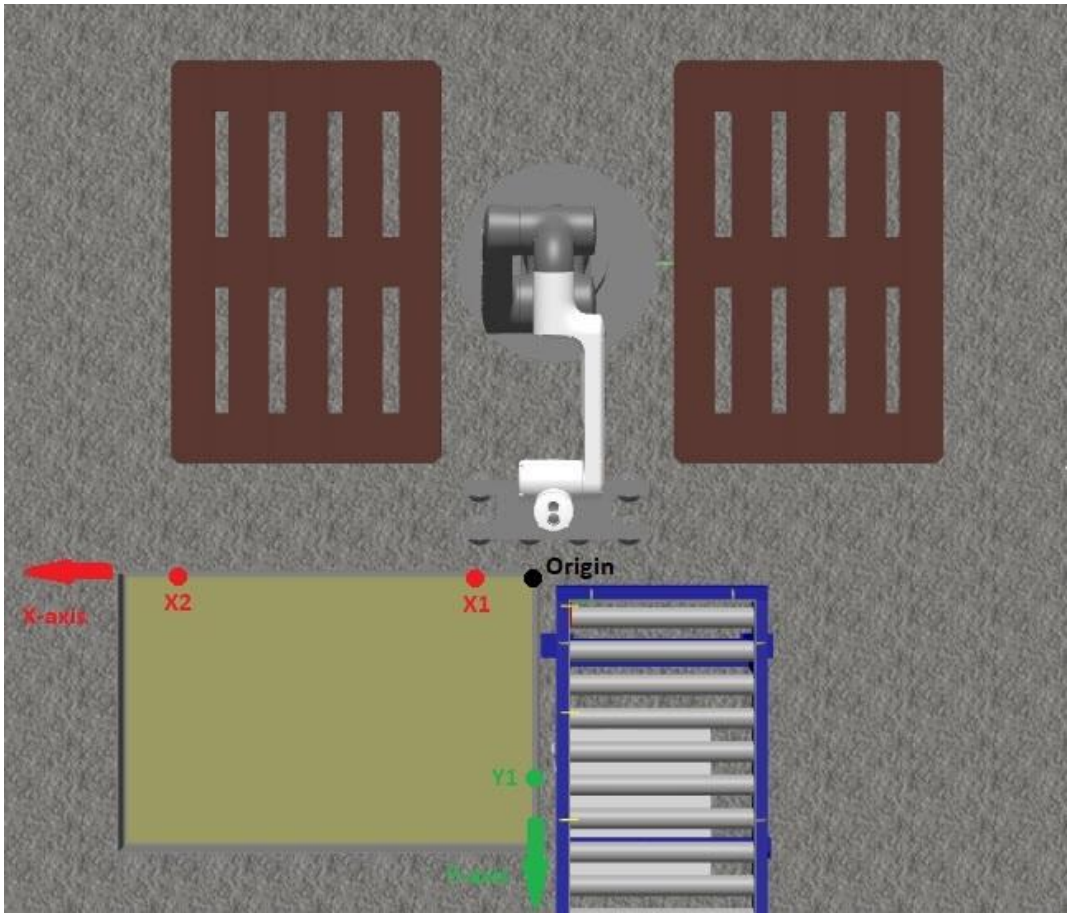
The slip sheet tray requires a calibrated workobject. By default, one slip sheet tray is foreseen in the RAPID framework.

Workobject	Description
obSheetMag1	Workobject with origin of the slip sheet tray

Slip sheet tray workobject definition

The position of the origin of each workobject should be defined as shown in the image below. By definition the position and orientation of the workobject must be chosen such that the X-axis runs along the long side of the pallet and the Y-axis runs along the short side of the pallet. As a result the Z axis will point upwards.

The workobject should be defined at the bottom of the slip sheet tray.



Slip sheet tray workobject calibration

The workobject can be calibrated from the FlexPendant. Refer to the Operating manual OmniCore under Defining a workobject and Defining the work object coordinate system for more information about workobject calibration.

Before calibration starts the robot must be equipped with a calibration tool, with its TCP accurately defined and active. Make sure that the calibration tool can reach the three positions as shown in red in the image above. These positions must be reachable at the bottom of the slip sheet tray.

To calibrate the workobject the 3 point method for the user frame should be chosen. Move the calibration pin to the three points indicated by X1, X2 and Y1 in the image above. X1 and X2 must be taken at the fixed long side of the slip sheet tray. Y2 must be taken at the short side of the slip sheet tray. The three positions must be accurately aligned with the slip sheet tray sides. To get the best accuracy the distance between the three positions should be as far apart as possible.

When the calibration process in finished the origin of the slip sheet tray workobject should be located in one corner of the slip sheet tray as shown in the image above.

Note

Make sure that the positive X-axis runs along the long side of the slip sheet and the positive Y-axis runs along the short side of the slip sheet.

Note

The calibration positions for the slip sheet tray must be taught at the bottom of the slip sheet tray.

3.4 Gripper configuration

Introduction

Palletizing Template only supports both vacuum and mechanical grippers. Vacuum grippers pick the boxes from the top and don't have any mechanical parts at the side or bottom of the boxes. The mechanical part is only present above the box.

Mechanical grippers, however, clamp the boxes on the side or hold or clamp boxes at the bottom. This means that there are always mechanical parts present at one side of the box. This limits the orientations at which a box can be placed on a layer, with respect to the boxes already present on the layer.

In applications where crates are palletized mechanical grippers are typically used. When these mechanical grippers only reside above the crates Palletizing Template can support these as well. They should however be defined as vacuum grippers.

Although the algorithms for calculating the pick and place position are designed to prevent rotation of axis 6 as much as possible, it can't be avoided that axis 6 will turn over angles of + to – 180°. The cabling to the gripper must be designed in such way that it can cope with these rotations. The work range of axis 6 should however not be reduced in the robot system parameters as may result in unreachable positions.

Note

Grippers should be able to rotate freely by axis 6 (6 axis robots) or axis 4 (4 axis robots) over an angle of at least +/- 180°.

Note

The workrange of axis 6 should not be reduced in the robot system parameters.

Eleven different types of grippers are supported. In the next chapters a description is given on how the define the used gripper type and how the set-up the correct tooldata for the used gripper.

For multi box handling the gripper should be capable of picking and releasing up to 3 boxes at once.

For mechanical grippers two methods of gripping are supported:

- Fork type grippers. For this type of gripper the mechanics are located at one side of the box. The box can’t be placed against already present boxes with the side of the mechanics as the mechanics will then collide with the already placed boxes. The tcp of the gripper is located at the side of the mechanics.
- Clamp type grippers. This type of gripper usually consists of two plates of which one is moveable and the other is fixed. When placing boxes the fixed plate should be located between the box in the gripper and the already placed boxes, so that the moveable plate can open without hitting any of the boxes already placed. The tcp for this gripper type is at the fixed plate

So for the fork type grippers the tcp / mechanics location must be away from already placed boxes where with the clamp method the tcp / fixed plate is to be positioned against the already placed boxes, which is a fundamental difference between the two gripping methods

3.4.1 Gripper type selection

Introduction

Eleven types of vacuum grippers are supported by Palletizing Template.

Gripper type / CONST	Description	Value
GRP_VACUUM_CENTER	Vacuum gripper with the tcp at the center of the gripper.	1
GRP_VACUUM_CORNER	Vacuum gripper with the tcp at the edge of the vacuum surface.	2
GRP_MECH_TCP_SIDE_CENTER_BOT	Mechanical gripper with the tcp at the bottom center of the gripper (fork type gripper).	3
GRP_MECH_TCP_SIDE_CORNER_BOT	Mechanical gripper with the tcp at the bottom corner of the gripper (fork type gripper).	4
GRP_MECH_OPP_SIDE_CENTER_BOT	Mechanical gripper with the tcp at the bottom center of the gripper (clamp type gripper).	5
GRP_MECH_OPP_SIDE_CORNER_BOT	Mechanical gripper with the tcp at the bottom corner of the gripper (clamp type gripper).	6
GRP_MECH_TCP_SIDE_CENTER_TOP	Mechanical gripper with the tcp at the top center of the gripper (fork type gripper).	7
GRP_MECH_TCP_SIDE_CORNER_TOP	Mechanical gripper with the tcp at the top corner of the gripper (fork type gripper).	8
GRP_MECH_OPP_SIDE_CENTER_TOP	Mechanical gripper with the tcp at the top center of the gripper (clamp type gripper).	9
GRP_MECH_OPP_SIDE_CORNER_TOP	Mechanical gripper with the tcp at the top corner of the gripper (clamp type gripper).	10

GRP_MECH_CENTER_TOP	Mechanical gripper with the tcp at the top center of the gripper.	11

The different gripper types will need different strategies for picking and placing. This is automatically handled by the Palletizing Template. The user interface when it comes to defining project and recipes is identical despite the installation gripper type.

However it can't guarantee that when using mechanical gripper the label orientation of the boxes on the pallet is as chosen when creating the recipe. The label position depends on the available pick and place orientations.

Gripper type definition

The used gripper type is to be defined in the variable GripperType.

The GripperType variable is declared in the module: Settings.

Basic example

The following example illustrates the GripperType variable

Example 1

```
CONST num GripperType:= GRP_VACUUM_CENTER;
```

In this example the gripper has the TCP at the center of the vacuum surface and picks the box(es) at the center of box(es).

Tool and loaddata

It is important to not only set the tcp location of the gripper but to also specify the correct loaddata. The correct loaddata will enhance the robots performance in lifetime. On 6 axis robots the loaddata can be defined using the Load identification functionality that is present in the robot controller. For 4 axis robots the loaddata must be estimate as good possible, or if possible, retrieved from a CAD system

Note

Care should be taken so that the correct loaddata of the tool are defined.

Palletizing Template provides a standard tooldata called tGripper for the definition of the tooldata. tGripper is declared in the module Settings. Common methods, like the four-point method, can be used to define the tooldata.

Tooldata available from for example a CAD model can entered directly with the aid of RobotStudio or from the FlexPendant.

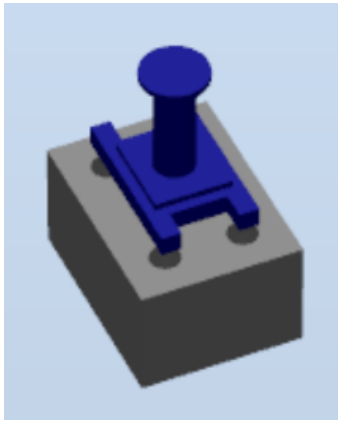
Refer to the Operating manual OmniCore under Defining the tool frame.

The different gripper types and their properties are described in the following chapters.

3.4.1.1 Vacuum gripper with the TCP at the center

Definition

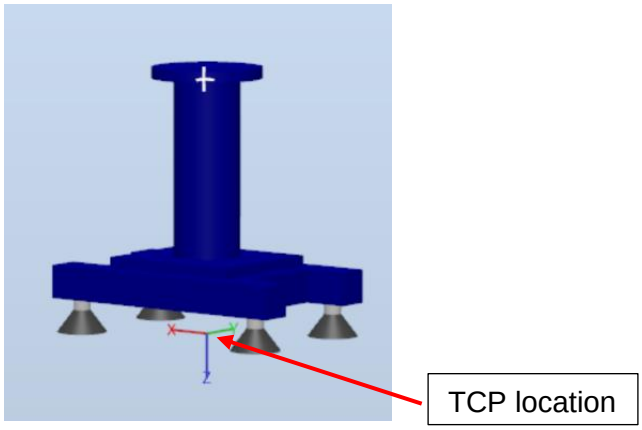
This type of gripper picks the box or a cluster of boxes at the center of the top side of the box as illustrated in the picture below.



The TCP of the gripper should be calibrated at the center of the vacuum surface.

TCP calibration

Following guidelines apply for the calibration of the tooldata for grippers with the TCP at the center of the vacuum surface.



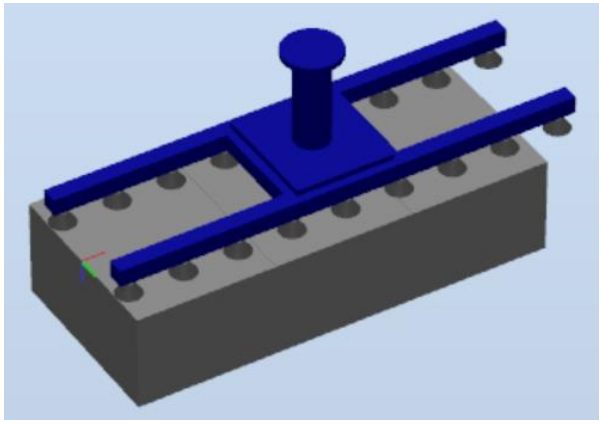
The tools X axis should point towards the center of the robot flange. If the tool is located in the center of the robot flange the X axis should run parallel to the X axis of tool0.

The tools Z axis should always point away from the vacuum surface.

3.4.1.2 Vacuum gripper with the TCP at the center

Definition

This type of gripper picks the box(es) with one side of the gripper aligned with a side of the box(es) as shown in the image below.

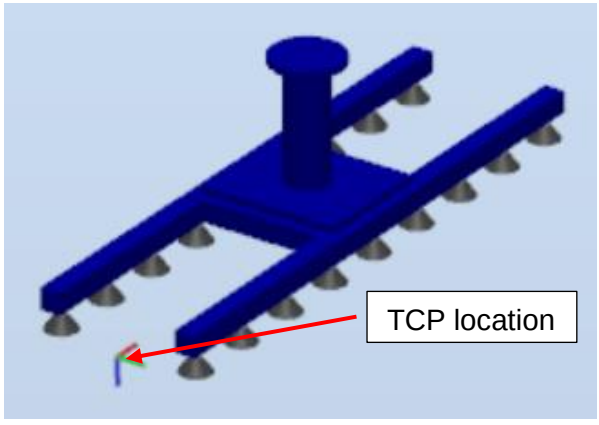


The TCP of the gripper should be calibrated at one side of the vacuum surface. This is usually the middle of the short side of the vacuum surface.

To avoid “helicoptering” the box(es) can be picked in two different ways. This is with the TCP at the front or at the rear of the box(es) (when facing the feeder). This implies that the gripper will also be rotated 180° depending on what way the box(es) are picked.

TCP calibration

Following guidelines apply for the calibration of the tooldata for grippers with the TCP at the corner of the vacuum surface.



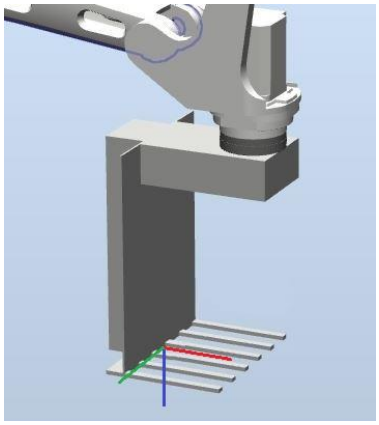
The tools X axis should point towards the center of the robot flange. If the tool is located in the center of the robot flange the X axis should run parallel to the X axis of tool0.

The tools Z axis should always point away from (out of) the vacuum surface.

3.4.1.3 Mechanical gripper with the TCP at the bottom center of the gripper (fork type)

Definition

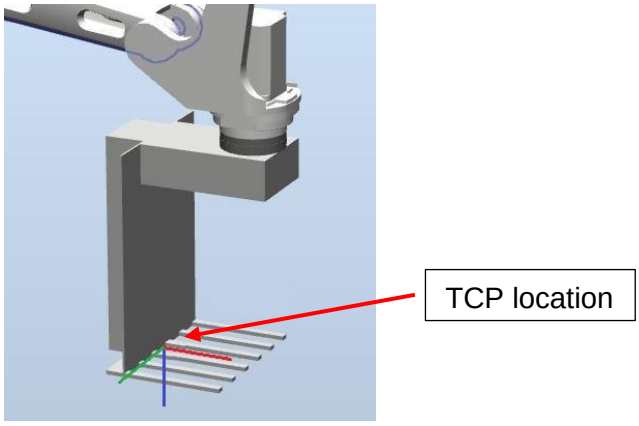
This type of gripper picks the box(es) with the center of the gripper aligned with the center of the box(es) as shown in the image below. The box(es) are supported by forks.



The TCP of the gripper should be calibrated at the center of the gripper, against the plate on top of the forks.

TCP calibration

Following guidelines apply for the calibration of the tooldata for grippers with the TCP at the bottom center of the grippers back plate.



The tools X axis should point towards the center of the robot flange. If the tool is located in the center of the robot flange the X axis should run parallel to the X axis of tool0.

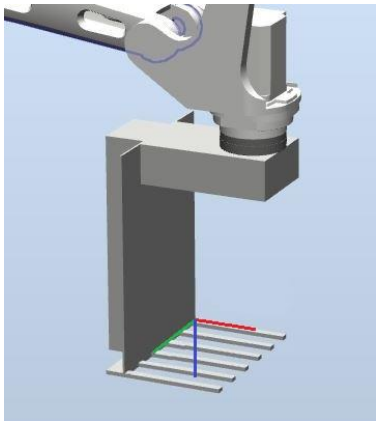
The Y axis should run parallel to the plate of the gripper

The tools Z axis should always point downwards from the forks.

3.4.1.4 Mechanical gripper with the TCP at the bottom corner of the gripper (fork type)

Definition

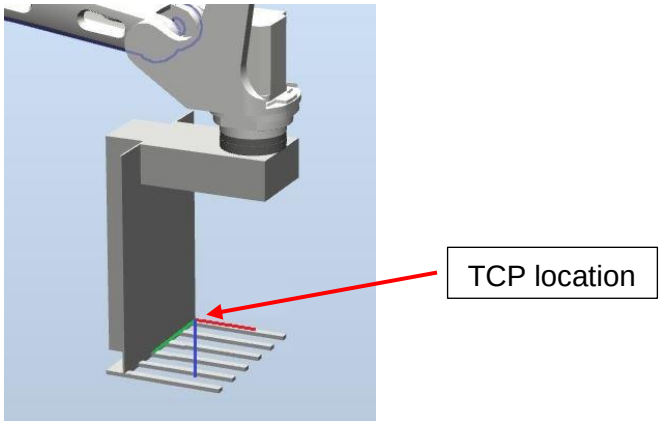
This type of gripper picks the box(es) with one side of the gripper aligned with a corner of the box(es) as shown in the image below. The box(es) are supported by forks.



The TCP of the gripper should be calibrated at the corner of the gripper, against the plate on top of the forks.

TCP calibration

Following guidelines apply for the calibration of the tooldata for grippers with the TCP at the bottom corner of the grippers back plate.



The tools X axis should point towards the center of the robot flange. If the tool is located in the center of the robot flange the X axis should run parallel to the X axis of tool0.

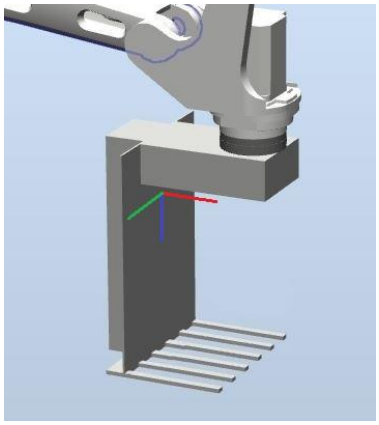
The Y axis should run parallel to the plate of the gripper

The tools Z axis should always point downwards from the forks.

3.4.1.5 Mechanical gripper with the TCP at the top center of the gripper (fork type)

Definition

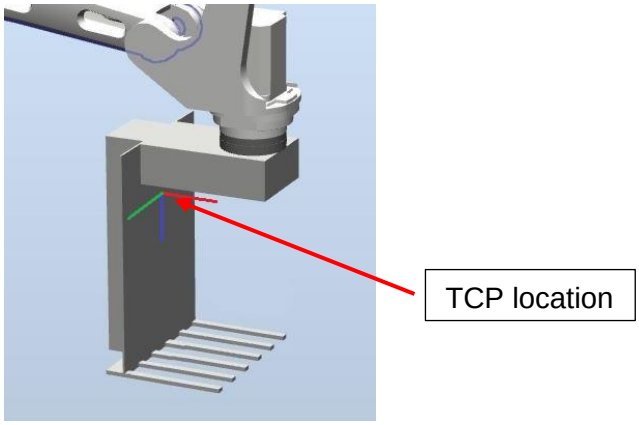
This type of gripper picks the box(es) with the center of the gripper aligned with the center of the box(es) as shown in the image below. The box(es) are supported by forks.



The TCP of the gripper should be calibrated at the center of the gripper, against the plate at the top of the clamping mechanism.

TCP calibration

Following guidelines apply for the calibration of the tooldata for grippers with the TCP at the top center of the grippers back plate.



The tools X axis should point towards the center of the robot flange. If the tool is located in the center of the robot flange the X axis should run parallel to the X axis of tool0.

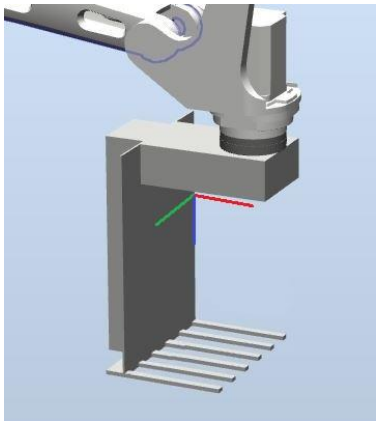
The Y axis should run parallel to the plate of the gripper

The tools Z axis should always point downwards from the forks.

3.4.1.6 Mechanical gripper with the TCP at the top corner of the gripper (fork type)

Definition

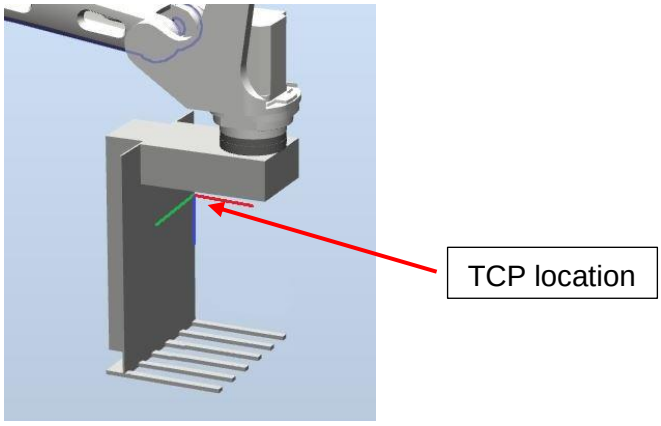
This type of gripper picks the box(es) with one side of the gripper aligned with a corner of the box(es) as shown in the image below. The box(es) are supported by forks.



The TCP of the gripper should be calibrated at the corner of the gripper, against the plate at the top of the clamping mechanism.

TCP calibration

Following guidelines apply for the calibration of the tooldata for grippers with the TCP at the top corner of the grippers back plate.



The tools X axis should point towards the center of the robot flange. If the tool is located in the center of the robot flange the X axis should run parallel to the X axis of tool0.

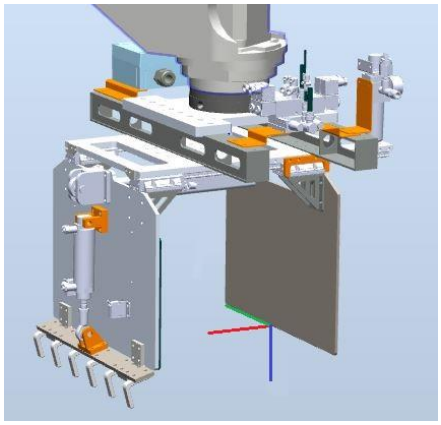
The Y axis should run parallel to the plate of the gripper

The tools Z axis should always point downwards from the forks.

3.4.1.7 Mechanical gripper with the TCP at the bottom center of the gripper (clamp type)

Definition

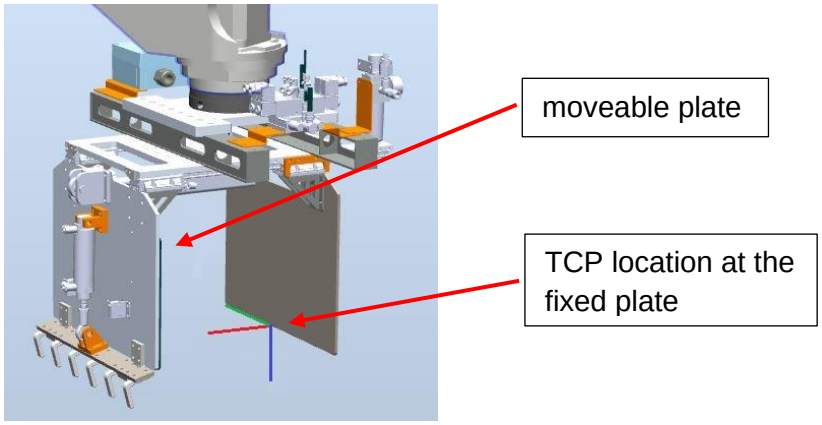
This type of gripper picks the box(es) with the center of the fixed gripper plate aligned with the center of the box(es) as shown in the image below. The boxes are clamped between a movable and a fixed plate.



The TCP of the gripper should be calibrated at the center of the gripper, against the fixed plate, at the bottom of the plate.

TCP calibration

Following guidelines apply for the calibration of the tooldata for grippers with the TCP at the bottom center of the grippers fixed plate.



The tools X axis should point towards the center of the robot flange. If the tool is located in the center of the robot flange the X axis should run parallel to the X axis of tool0.

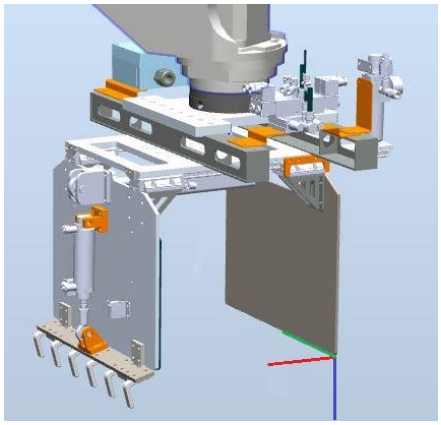
The Y axis should run parallel to the plate of the gripper

The tools Z axis should always point downwards from the fixed plate.

3.4.1.8 Mechanical gripper with the TCP at the bottom corner of the gripper (clamp type)

Definition

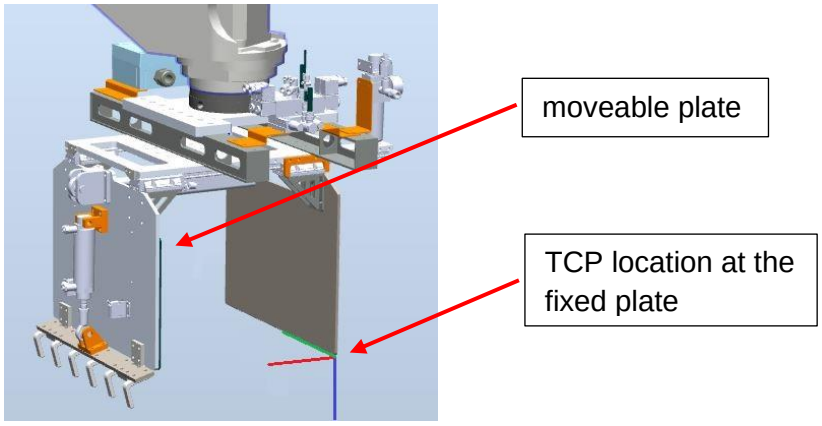
This type of gripper picks the box(es) with the corner of the fixed gripper plate aligned with the corner of the box(es) as shown in the image below. The boxes are clamped between a movable and a fixed plate.



The TCP of the gripper should be calibrated at the corner of the gripper, against the fixed plate, at the bottom of the plate.

TCP calibration

Following guidelines apply for the calibration of the tooldata for grippers with the TCP at the bottom corner of the grippers fixed plate.



The tools X axis should point towards the center of the robot flange. If the tool is located in the center of the robot flange the X axis should run parallel to the X axis of tool0.

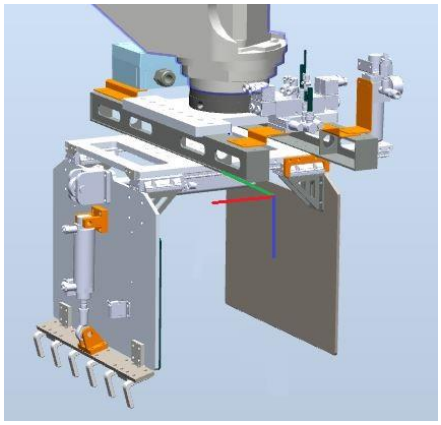
The Y axis should run parallel to the plate of the gripper

The tools Z axis should always point downwards from the fixed plate.

3.4.1.9 Mechanical gripper with the TCP at the top center of the gripper (clamp type)

Definition

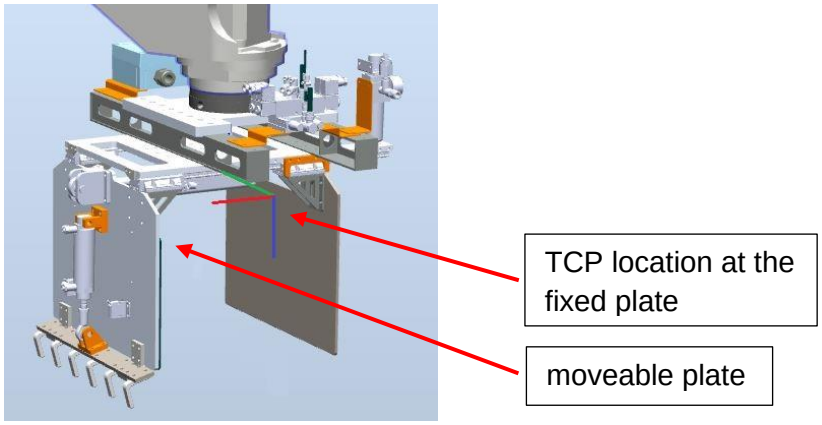
This type of gripper picks the box(es) with the center of the fixed gripper plate aligned with the center of the box(es) as shown in the image below. The boxes are clamped between a movable and a fixed plate.



The TCP of the gripper should be calibrated at the center of the gripper, against the fixed plate, at the top of the plate.

TCP calibration

Following guidelines apply for the calibration of the tooldata for grippers with the TCP at the top center of the grippers fixed plate.



The tools X axis should point towards the center of the robot flange. If the tool is located in the center of the robot flange the X axis should run parallel to the X axis of tool0.

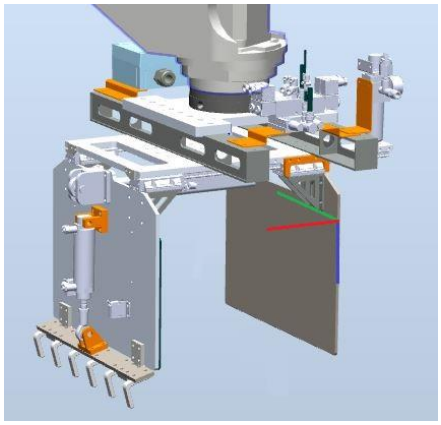
The Y axis should run parallel to the plate of the gripper

The tools Z axis should always point downwards from the fixed plate.

3.4.1.10 Mechanical gripper with the TCP at the top corner of the gripper (clamp type)

Definition

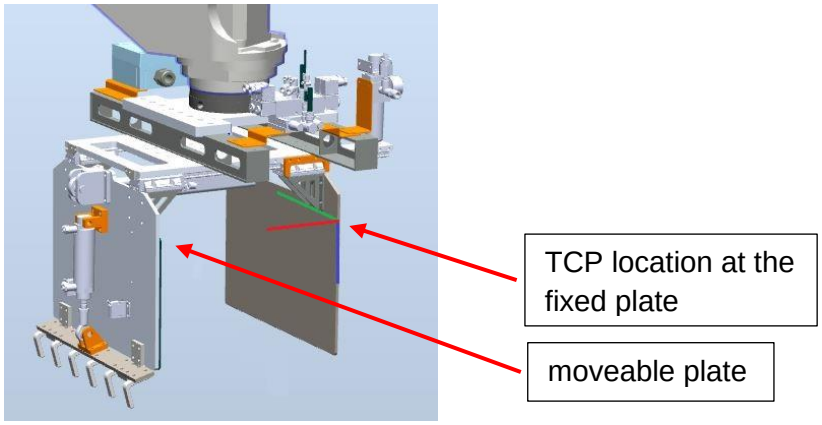
This type of gripper picks the box(es) with the corner of the fixed gripper plate aligned with the corner of the box(es) as shown in the image below. The boxes are clamped between a movable and a fixed plate.



The TCP of the gripper should be calibrated at the corner of the gripper, against the fixed plate, at the top of the plate.

TCP calibration

Following guidelines apply for the calibration of the tooldata for grippers with the TCP at the top corner of the grippers fixed plate.



The tools X axis should point towards the center of the robot flange. If the tool is located in the center of the robot flange the X axis should run parallel to the X axis of tool0.

The Y axis should run parallel to the plate of the gripper

The tools Z axis should always point downwards from the fixed plate.

3.5 Amount of boxes to handle

Introduction

Palletizing Template can be configured to handle more than one box at the time. The palletizing cycle however remains to be based on a single pick – single place cycle. Per cycle Palletizing Template will pick the amount of boxes that can be placed in one place action. Palletizing Template will try to optimize the pattern stacking in such way that the maximum amount of boxes per cycle can be handled. This will result in the least number of cycles to fill the layer, thus resulting in the highest throughput.

Palletizing Template can be configured to handle up to 3 boxes. When configured to handle 3 boxes Palletizing Template will pick either 1, 2 or 3 boxes depending on the requirement for the next cycle.

Depending on the design of the gripper and the requirements of the installation can hold a certain maximum of boxes. In the Palletizing Template recipe is possible to choose a reduced maximum amount boxes if a certain box doesn't allow the gripper to hold the maximum theoretical amount due to for example the size of the box.

On the feeder boxes can be fed in two ways, long side or short side leading. Palletizing Template supports both formations. If the feeder system is capable of feeding boxes both ways, the user has to configure which formation is used in the recipe. Only one formation can be chosen for the recipe.

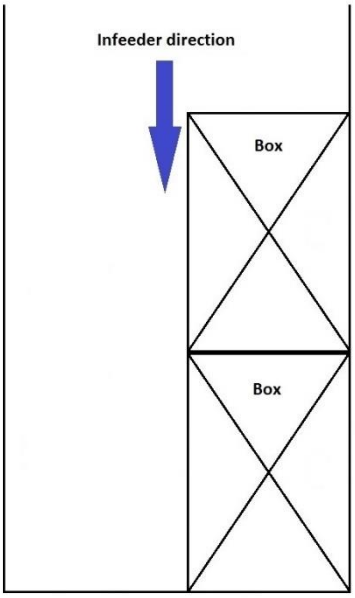
Note

Palletizing Template can be configured to handles both long side or short side lead boxes. The user needs to select which of the two formations is used in the recipe. A combination of the two is not supported.

3.5.1 Short side lead

Description

The image below shows the feeder with short side lead boxes.



The amount of boxes that can be handled by the gripper in short side lead formation has to be defined in the variable MaxShortSideLead.

The amount of boxes in short side lead is limited to 3.

Note

A combination of long and short side lead boxes is not supported.

The MaxShortSideLead variable is declared in the module: Settings.

Basic example

The following example illustrates the MaxShortSideLead variable

Example 1

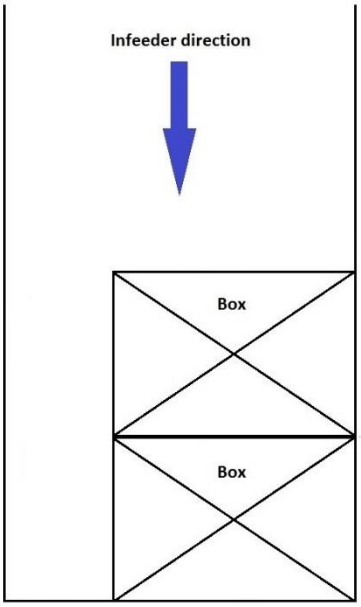
```
CONST num MaxShortSideLead:=2
```

In this example the maximum amount of boxes, short side lead, is 2.

3.5.2 Long side lead

Description

The image below shows the feeder with long side leading boxes.



The amount of boxes that can be handled by the gripper in long side lead formation has to be defined in the variable MaxShortSideLead.

The amount of boxes in long side lead is limited to 3.

Note

A combination of long and short side lead boxes is not supported.

The MaxShortSideLead variable is declared in the module: Settings.

Basic example

The following example illustrates the MaxLongSideLead variable

Example 1

```
CONST num MaxLongSideLead:=3;
```

In this example the maximum amount of boxes, long side lead, is 3.

3.6 Available pick positions

Mechanical grippers

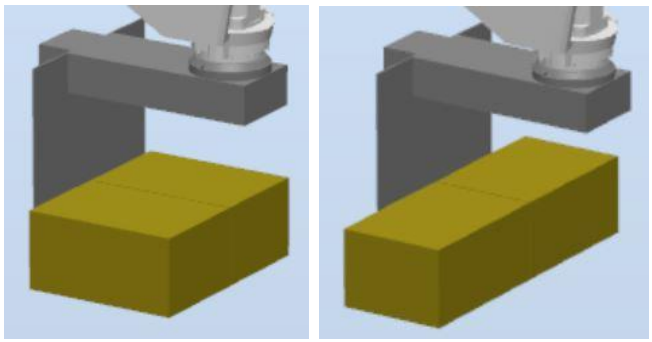
Mechanical grippers can pick the box from the feeder in different ways. Depending on the location of the tcp as described in the previous two chapters, the box will either be picked in the center or aligned at one corner of the gripper.

LeanPalletize will generate patterns based on the available pick positions. If too few pick positions are available LeanPalletize may not be able to create the pattern. An error message is generated when the pattern is saved.

Grippers with the tcp at the center

To pick the box from the feeder the center of the gripper will be placed at the center of the box. The box can both be picked from the long or from the short side. The images below show the two orientations of the boxes in the gripper.

The same principle applies for the mechanical gripper with one fixed and one moveable plate.

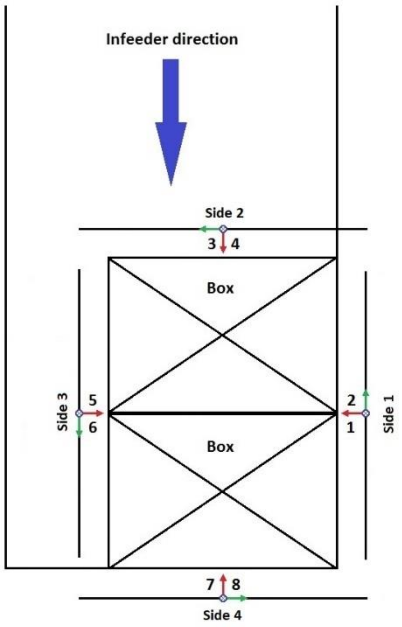


For this particular gripper type the width of the fixed gripper plate should not exceed the width of the box to be handled.

The different orientations of the box in the gripper result in four possible pick positions. In the image below the gripper is schematically drawn in each of the four possible positions. In most installations the access of the gripper on the feeder is limited, so that only one or two orientations are possible.

In the image below side 2 and side 4 can’t be used as the gripper has only access to one box. However when handling one box at the time, side 2 and side 4 can be used.

Each pick position is represented by a coordinate system (red = X axis, green = Y axis), which is the location of the TCP. The black line represents the fixed plate of the gripper.



In LeanPalletize the possible gripper positions need to be configured in the variable AvailablePickPos. AvailablePickPos is an array with 8 elements. This array contains two elements for each side. For side 1 this is element 1 and 2 for side 2 this is element 3 and 4 etc. If side 1 is available for the gripper to pick boxes, array element 1 should be set to 1. The second array element doesn't need to be set. Elements matching a side that is not accessible should be set to 0.

Note

Make sure the boxes to handle are wider than the gripper, otherwise it's not possible to build all types of patterns.

AvailablePickPos is declared in the Settings module.

Basic example

The following example illustrates the AvailablePickPos variable

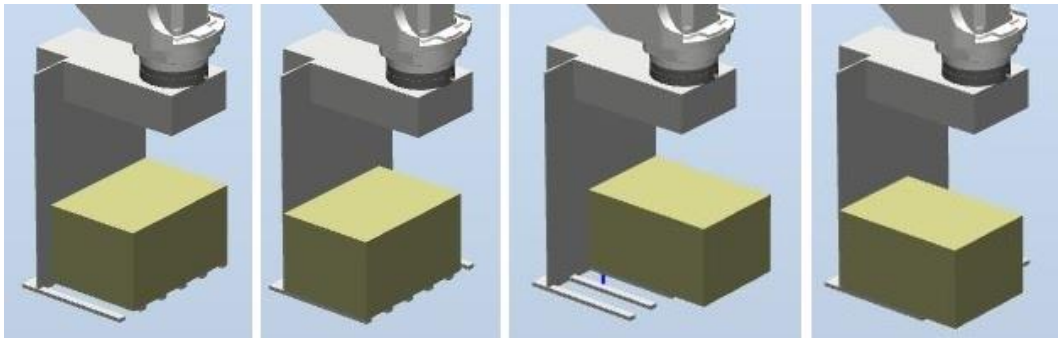
Example 1

```
CONST num AvailablePickPos{8}:=[1,0,0,0,0,0,0,0];
```

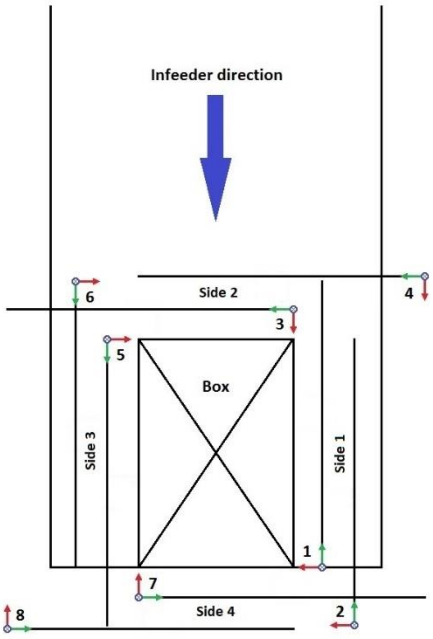
In this example side 1 is available for picking boxes. Only array element 1 is set to 1.

Grippers with the tcp at the corner

When picking boxes from the feeder, with this type of gripper, LeanPalletize will align a corner of the box with the side of the gripper. The means that any box can be held in the gripper in 4 different ways. Two configurations when holding the box with the short side against the fixed plate and 2 configurations when holding the box with the long side against the fixed plate. The same principle applies for the mechanical gripper with one fixed and one moveable plate.



The four different positions the box can be held in the gripper results in 8 different positions at which the box can picked from the feeder. In the schematic overview in the image below the different gripper positions at the pick position are shown. In most installations the access of the gripper on the feeder is limited, so that only a limited number of orientations are possible.



Each pick position is represented by a coordinate system (red = X axis, green = Y axis), which is the location of the TCP. The black line represents the fixed plate of the gripper.

In LeanPalletize the possible gripper positions need to be configured in the variable AvailablePickPosX. AvailablePickPosX is an array with 8 elements. This array contains two elements for each side. For side 1 this is element 1 and 2 for side 2 this is element 3 and 4 etc.

On side 1 position 1 the gripper will always be placed with the tcp corner at front right corner of the box, seen from the top view of the image above. The gripper will always be placed in the same position independent of the size and amount of boxes. This gripper position is typically used for fork grippers as the forks will hold the box at the bottom. This means that the forks will need to placed between the rollers.

In position 2 on side 1, the opposite side of the tcp side will be aligned with the top right side of the box, seen from the top view of the image above. This implicates that the positions of the gripper relative to the feeder will change as the box size or amount of boxes to handle changes. For fork grippers this location is not desired as position of the forks varies and thus may not always fit between the rollers of the conveyor.

If the gripper is wider than the box to pick at least two pick orientations will need to be selected, where the box corner is aligned with both sides of the gripper. If this is not the case a may not be possible to create the pattern as the gripper will collide with the boxes already placed on the pallet.

Note

If the gripper is wider than the boxes to handle make sure there are at least two pick configurations, otherwise it’s not possible to build all types of patterns.

Since it is possible to define four different gripper types a separate array is foreseen for each gripper type (see chapter 3.3). For the first gripper this is AvailablePickPos1. For the second gripper this is AvailablePickPos2 etc.

The same rules apply for the other gripper positions.

AvailablePickPos is declared in the Settings module.

Basic example

The following example illustrates the AvailablePickPos variable

Example 1

```
CONST num AvailablePickPos{8}:=[1,0,0,0,0,1,0,0];
```

In this example position 1 and 6 are available for picking the box. In both position 1 and 6 the gripper forks will always fit between the rollers, but the box is aligned at different sides of the gripper as the gripper is rotated 180) between these two configurations.

3.7 Gripper width

Description

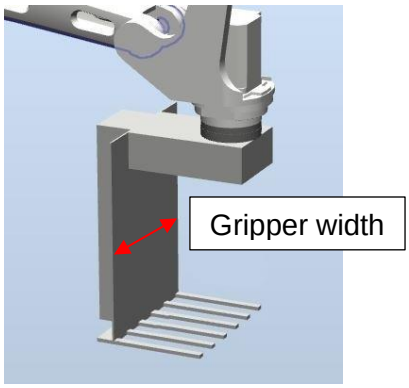
To be able to align the side of the grippers fixed plate with the corner of the box, the width of the fixed plate (as shown in the image below) needs to be configured. When the gripper is wider than the box the gripper will stick out on one side. LeanPalletize will take this in account and will make sure that this part of the gripper doesn't collide with boxes already on the pallet.

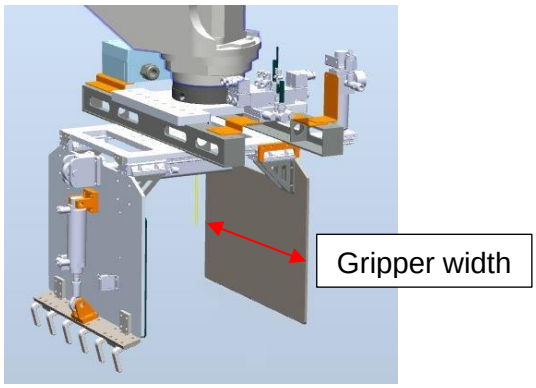
Since it is possible to define four different gripper types (see chapter 3.3), the GripperWidth array has four elements, one element for each gripper type.

The gripper width must be defined for mechanical grippers only.

The width of the gripper should be less than the length or width of the box (depending on the feed direction, long or short side lead), when the tcp is located in the center of the fixed plate. If this is not the case the pattern can't be build as the gripper will collide with the boxes already placed on the pallet.

This configuration is done from the RAPID variable GripperWidth. GripperWidth is declared in the module: Settings.





The gripper width is specified in mm.

Note

For grippers with the tcp at the center, make sure the boxes to handle are wider than the gripper, otherwise it’s not possible to build all types of patterns.

Note

If the gripper, with the tcp at the corner, is wider than the boxes to handle make sure there are at least two pick configurations, otherwise it’s not possible to build all types of patterns.

Basic example

The following example illustrates the AvailablePickPos1 variable

Example 1

```
CONST num GripperWidth:=350;
```

In this example the gripper has a width of 350 mm.

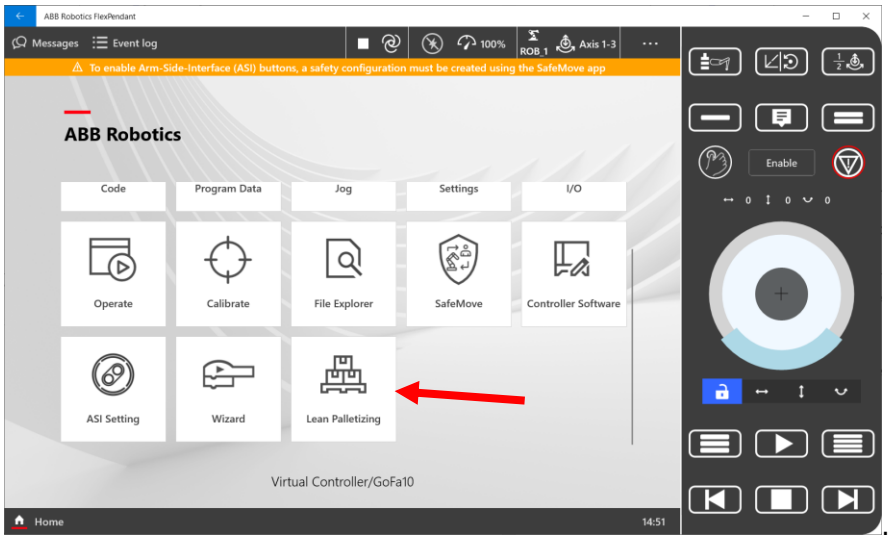
4 The Palletizing Template FlexPendant app

Introduction

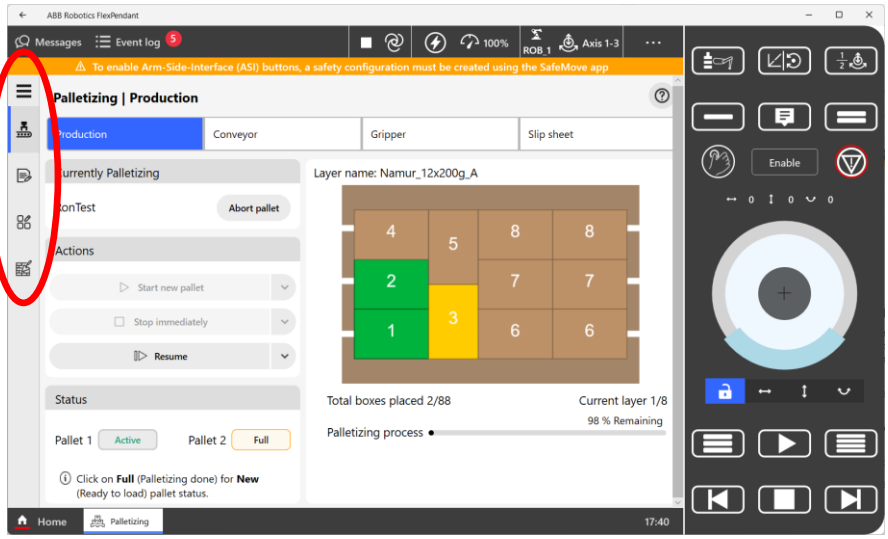
This chapter describes the functionality of the FlexPendant application for Palletizing Template.

The user interface is identical for all types of robots (4 axis, 6 axis or cobots) as well as for the used gripper type (vacuum or mechanical). The used gripper type is to configurated in the Settings module. The robot type is automatically selected.

The Palletizing Template FlexPendant app is started by selecting Palletizing Template from the home page



The different functions are chosen from the Palletizing Template main screen, from either the hamburger menu or the function buttons at the left-hand side of the screen.



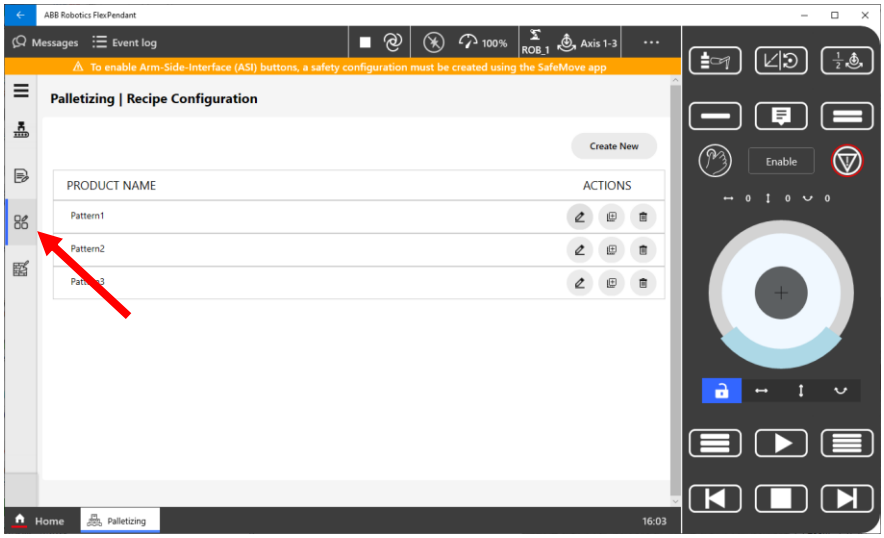
The FlexPendant app has four functions with corresponding buttons as shown in the table below:

Icon	Description
	Production running.
	Box, motion tuning
	The recipe configurator for the creation of new and modification of existing palletizing recipes.
	Creating new pattern libraries.

4.1 The recipe configurator

Introduction

The creation of a new palletizing recipe is done with the recipe configurator. A palletizing recipe contains all the information for the robot to build a pallet stack.



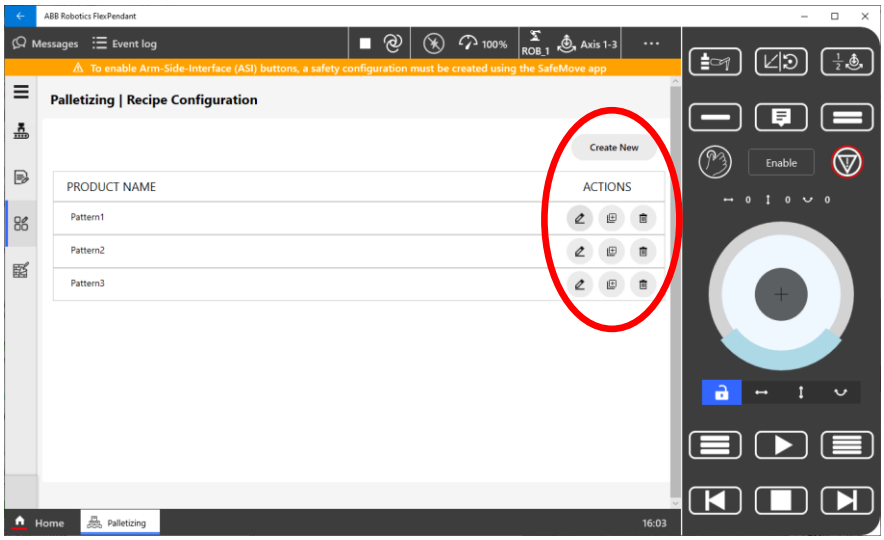
Creating a palletizing recipe is solely based on the product data which needs to be entered by the user. The recipe configurator is wizard-based tool, with the focus on product data and not on robot related issues. This means that the creation of a palletizing recipe is straight forward and intuitive.

The recipe configurator is used for both creating new palletizing recipes as for modifying existing recipes.

The next chapters explain how to create a new recipe and how to modify an existing one.

To start the recipe configurator, press the recipe configurator button.

The recipe screen appears as shown in the image below:



The following buttons are available for each recipe:

Tab	Description
Create New	Create a new recipe. When pressed the user will be prompted to enter the name of the new recipe. Then the recipe configuration wizard will be started.
	Modify an existing recipe. When pressed the recipe configuration wizard will be started.
	Make a copy of the recipe. When pressed the user will be prompted to enter the name for the copy of the recipe. To enter the recipe configurator the modify an existing recipe will need to be selected
	Delete the recipe. When pressed the user is asked to confirm the deletion of the recipe.

The buttons to modify, copy and delete a recipe are shown for each recipe.
Pressing such a button will activate the functionality for the related recipe.

The product configuration wizard

The product configuration wizard has 6 steps with which the user will be guided through the process of defining or modifying the recipe.

When modifying an existing recipe all data fields show the property values of the existing recipe. The property values can then be changed.

When creating a new recipe, the data fields are set to default values.

The steps are identical for both modifying an existing recipe as for creating a new recipe.

The wizard has six steps with for each the following functionality

Tab	Description
Box	Define the box dimensions and weight, number of boxes to handle and label positions.
Pallet	Define the pallet dimensions.
Slip sheets	Define the slip sheet dimensions and the maximum stack height
Pattern	Select a pattern for the odd, even and top layer(s).
Stack	Set the number of layers or maximum pallet height. Define and modify the predefined layer order. Add slip sheets.
Summary	Save the recipe and exit the recipe configurator.

The functionality for each wizard step will be explained in more detail in the following chapters.

Note

The Recipe Configurator can be disabled by setting the “disable_recipe_config” property equals “true” inside the setup.json file located at “HOME/Palletizing_Files” directory.

4.1.1 The Box wizard step

Introduction

In the Box wizard step the box parameters have to be defined. The box definition consists of the following parameters:

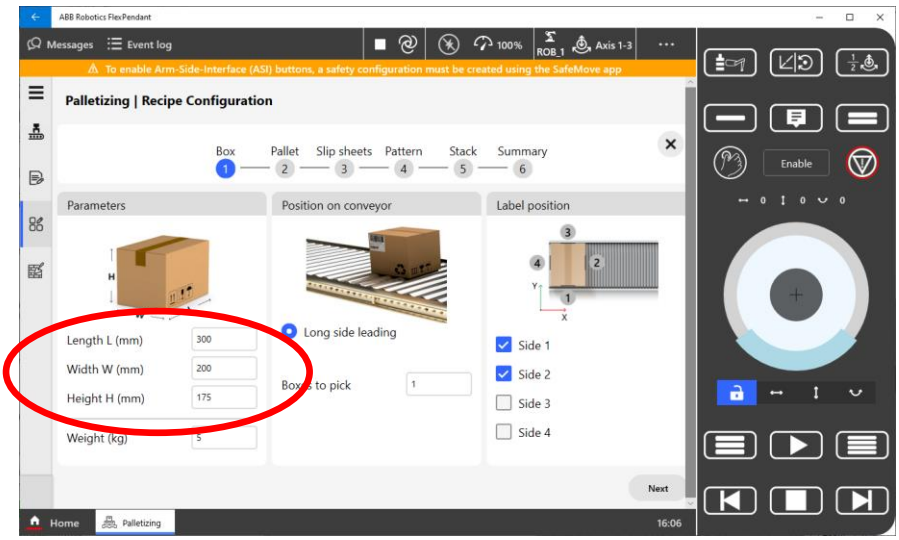
Parameter	Description
Size	Set the dimensions of the box.
Weight	Specify the weight of the box
Label position	Specify the label positions on the box.
Boxes to handle	Specify the amount of boxes to handle in one pick - place

Changing the labels positions will result in a reset of the label positions for the selected patterns, if already selected in the Pattern wizard step.

Changing the box height may result in a different amount of layers or a different max pallet height (Stack wizard step).

Box size

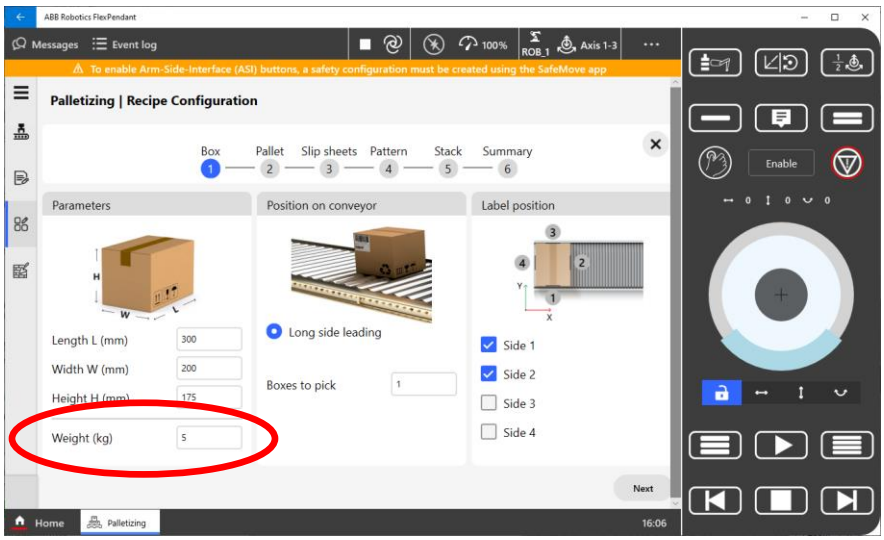
The dimensions of the box are the length, width and height of the box. These are specified in the corresponding fields as shown in the image below



The size range for the box length and width is between 150 and 2000 mm. For the height this between 10 and 1000 mm.

Weight

Specify the weight of the box.



The weight of the box needs to be specified in the Weight field. The weight is specified in Kg. in a multi box handling system this is still the weight of 1 box.

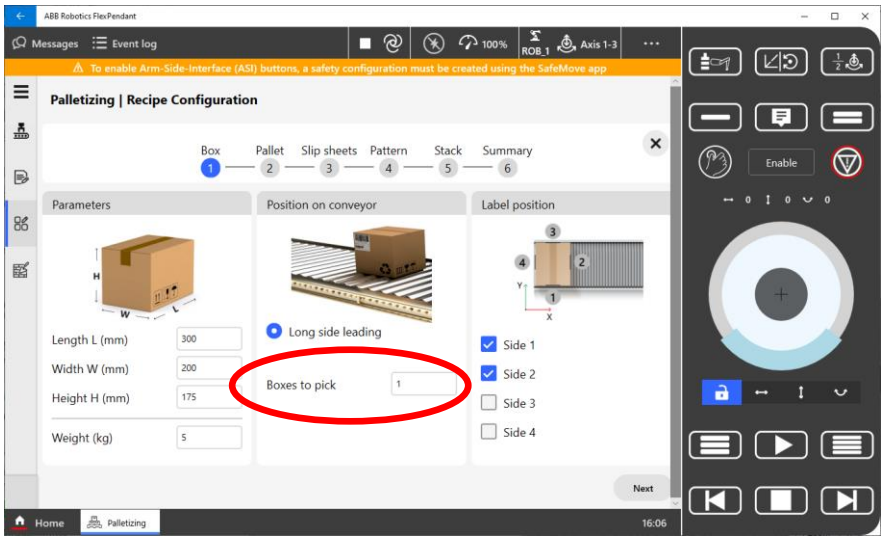
Note

It's very important for the robots performance to enter the correct weight of the individual box.

Boxes to pick

When the system is foreseen to handle multiple boxes the maximum amount of boxes to handle for this particular recipe needs to be specified. This maximum value is limited by the setting MaxLongSideLead or MaxShortSideLead as described in chapter 3.4.1.2.

The image below shows the selection of the amount of boxes to pick for both long and short side lead.

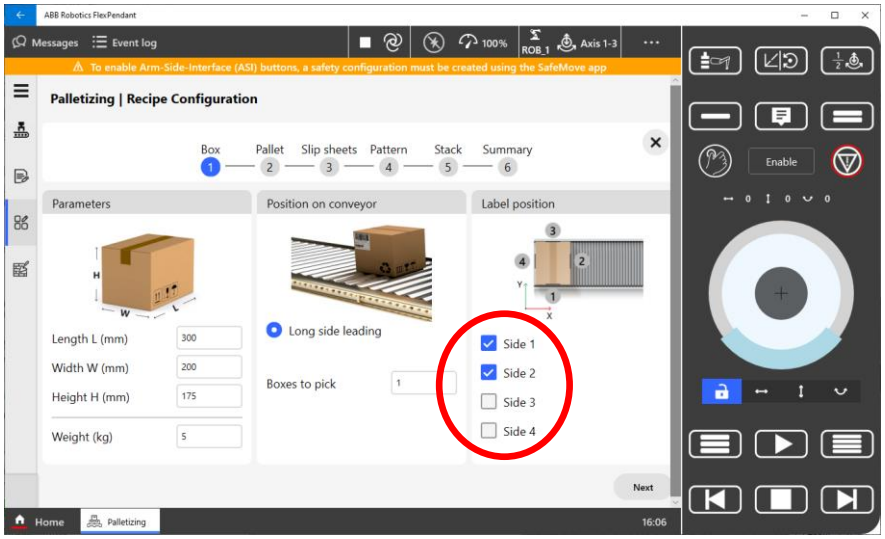


Label position

The side(s) of the box that contain a label need to be checked in the checkboxes which are drawn around the box shown in its position at the end of the feeder. When a box is checked that side is highlighted with a yellow line in the box. When a when pattern is chosen, Palletizing Template will calculate the pattern in such way that the label position(s) are as much as possible on the outside of the pallet. When there are for example two solutions, i.e. either a long side label or a short side label on the outside of the pallet, the long side label on the outside of the pallet has preference and will be chosen.

Labels can be present on more than one side of the box.

The image below illustrates the labels position for long side leading boxes.



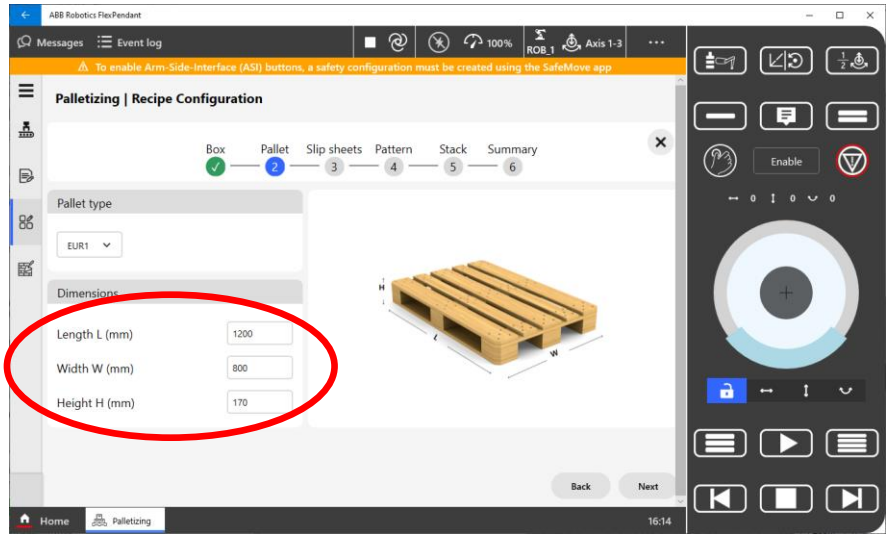
Note

Label positions can be changed after the patterns have been chosen. This will result in a recalculation of the label positions for the chosen pallet patterns in the Pattern wizard step.

4.1.2 The Pallet wizard step

Pallet dimensions

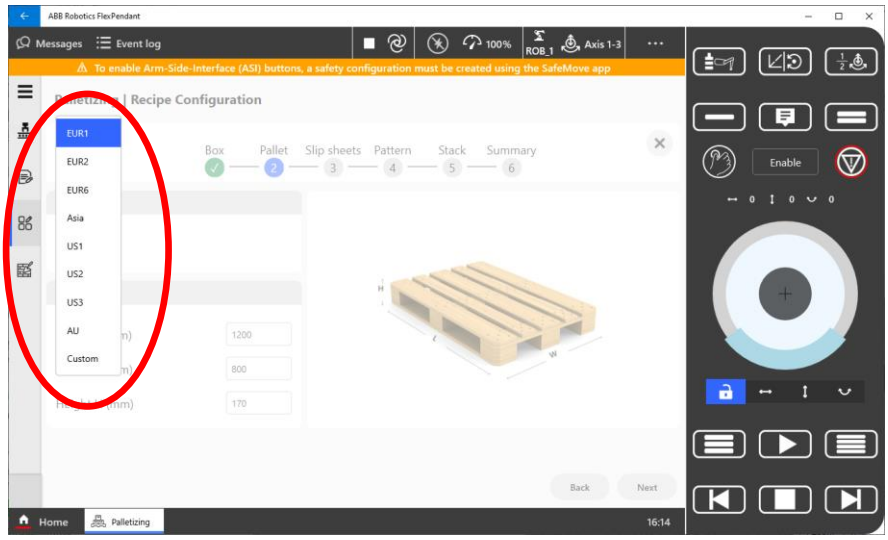
In the Pallet wizard step the dimensions of the pallet have to be entered.



The size range for the pallet length and width is between 400 and 2000 mm. For the height this between 10 and 400 mm.

Pallet type

With the Pallet type drop down box a choice can be made of a list of standardized pallet types with predefined dimensions.

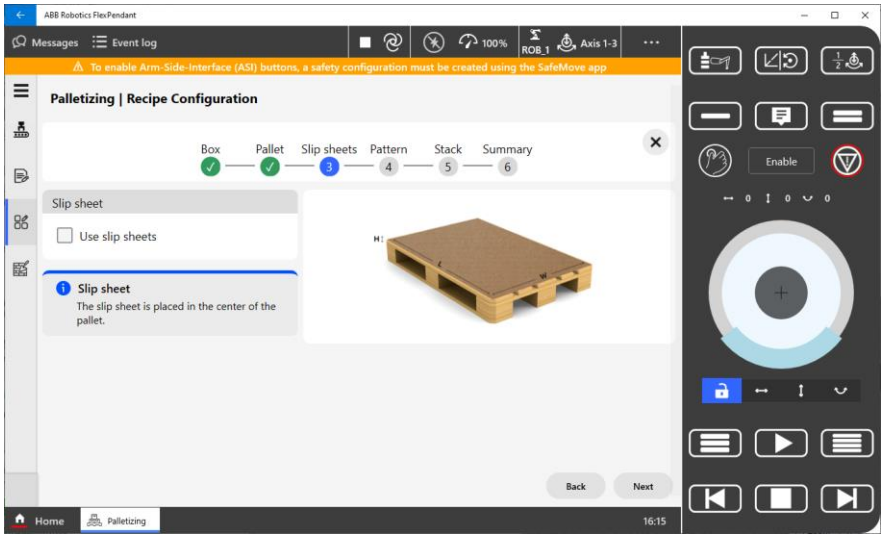


The values from the selected pallet type will be copied to the Length, Width and Height fields. The individual values can be modified when needed.

4.1.3 The Slip sheet wizard step

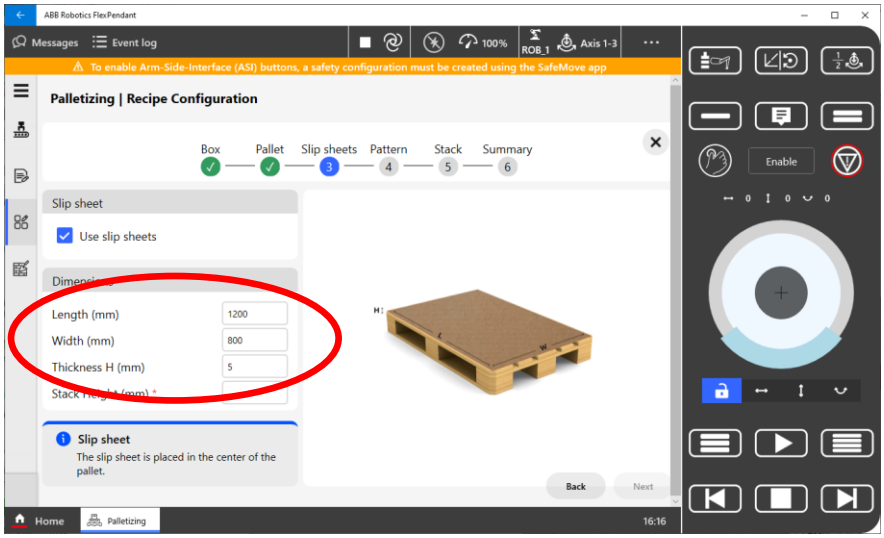
Description

The Slip sheets wizard step lets the user choose whether or not slip sheets are used for the recipe. If checked the slip sheet dimensions and the maximum height of the slip sheet stack have to be entered.



Slip sheet dimensions

The default length and width of the slip sheet are the same as the pallet length and width. The default thickness is 5 mm. These value can be adapted to the physical size of the slip sheet when needed.



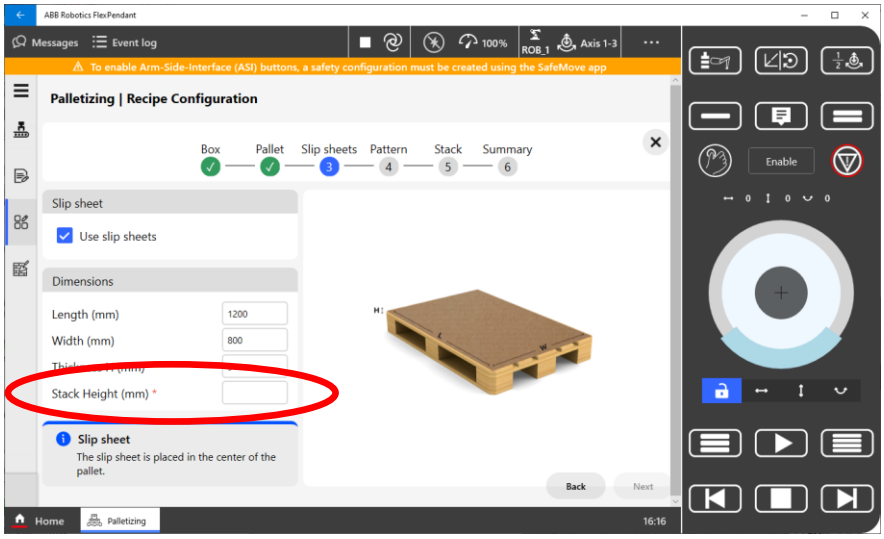
The size range for the slip sheet length and width is between 400 and 2000 mm. For the thickness this is between 0.1 and 100 mm.

Slip sheet stack height

Slip sheets are typically fed into the system on a stack. When a slip sheet is required the robot will move to the slip sheet magazine and search the stack with the vacuum gripper. A standard search routine is implemented in the RAPID code to perform this search.

After the robot program is started the robot will start the search from above the maximum height of the slip sheet stack. Once the first slip sheet is found the search for succeeding slip sheets can start from just above the previously located stack. Once the robot program has stopped and started again the search will start for the maximum stack height as an operator can have added slip sheets to the stack.

The maximum height form where the robot can start the search must be defined under the Stack height parameter.



The height range for the Search height is between 10 and 2000 mm.

Note

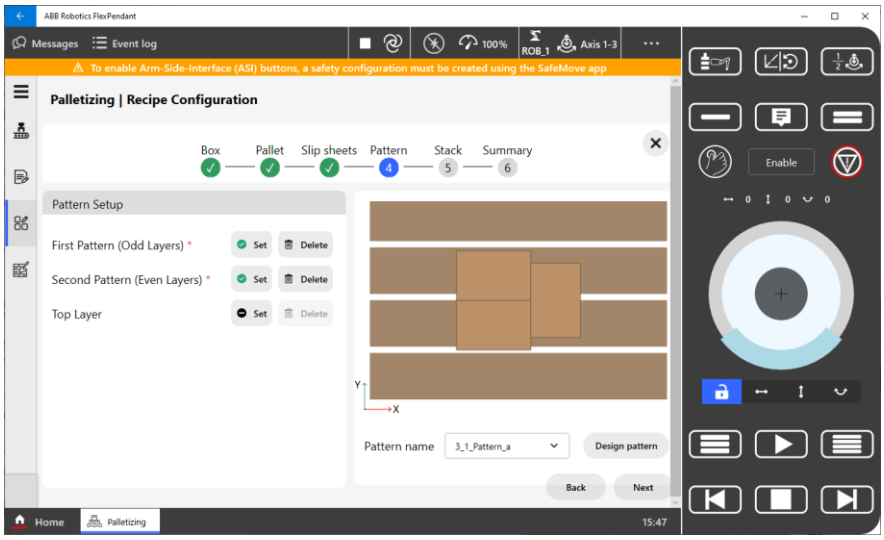
When a slip sheets are used the height of the slip sheet stack must be given in order to continue to the next wizard step.

4.1.4 The Pattern wizard step

Description

In the Pattern wizard step the patterns for the odd and even layers and if required a different pattern for the top layer. Different patterns can be chosen from the Pattern name drop down menu. When the desired pattern is selected a preview of the pattern is shown on the pallet.

The pattern is always shown centered on the pallet. Both pattern and pallet are drawn on scale.



The available patterns are loaded form the pattern library. Library patterns can be created or modified with the aid of the Pattern builder (see chapter 4.4).

Pattern color indication

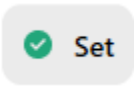
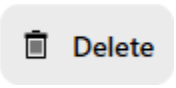
Different colors are used to show the pattern or individual boxes in the pattern. The pallet can have two colors:

Color	Description
Brown	Correct box position.
Blue	Box is selected for rotation of the label position
Red	Indicates that either the boxes are overlapping or that the pattern is larger than the pallet. In these cases the patterns can't shouldn't be build by the robot However it's still possible to choose this pattern, but care should be taken when this pattern stacked by the robot.

Pattern selection

When the desired pattern is shown as a preview on the pallet, this patterns can be added to the to or removed from the stack.

Following buttons with status indication are available:

Tab	Description
	No pattern has been selected for the pallet stack. Pressing this button will add the pattern shown as preview to the corresponding layer(s). Once pressed the button image changes to the state as shown in the below.
	A pattern is selected for the pallet stack. Pressing this button will cause a preview of the selected pattern to be shown. When a wrong pattern is shown it should be deleted first after which a different pattern can be selected again.
	Delete the selected pattern. This button is only available when a pattern is selected for the corresponding layer (odd, even or top).

To continue in the wizard to the next step a pattern for at least the odd and the even layers have to be selected. If the same pattern is used for both odd and even layers, this pattern should then be selected for the both.

When a wrong pattern is selected it should be deleted first after which a different pattern can be selected again.

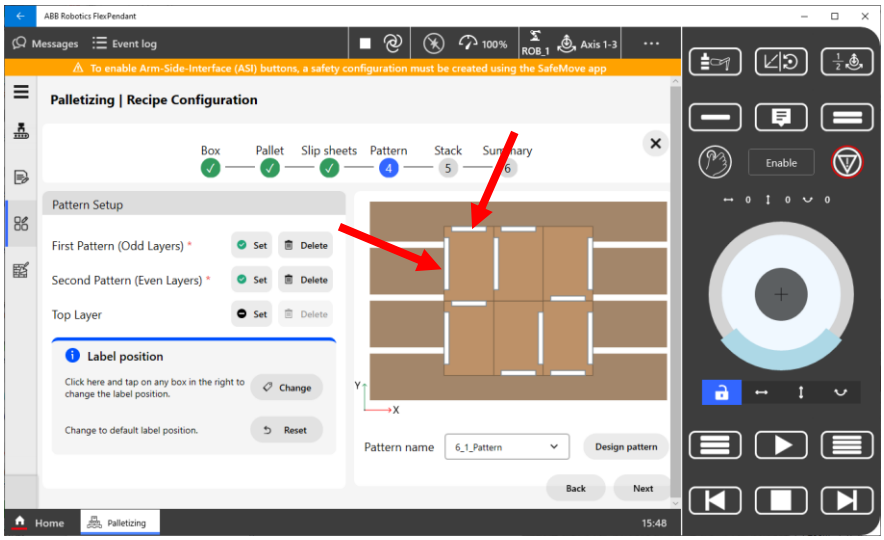
A different pattern can be chosen for the top layer is required, but this selection is optional.

Note

A pattern has to be selected for both the odd and even layers, to be able to continue to the next step in the wizard. Selecting a pattern for the top layer is optional.

Label orientation

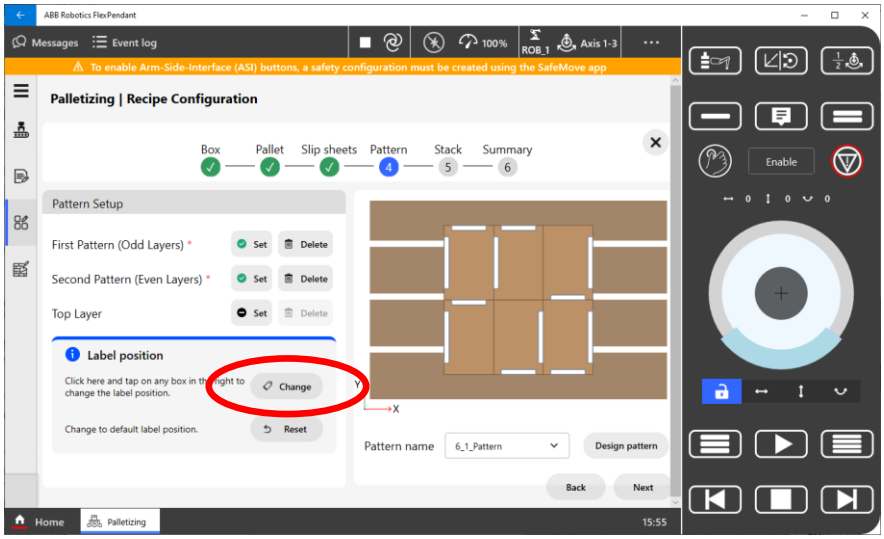
Upon selection of a pattern the label positions for each of the boxes is shown in the pattern preview. The labels as shown in white. The position of the labels when the box is in the pick position on the feeder can be selected in the Box wizard step (see chapter 4.1.1).



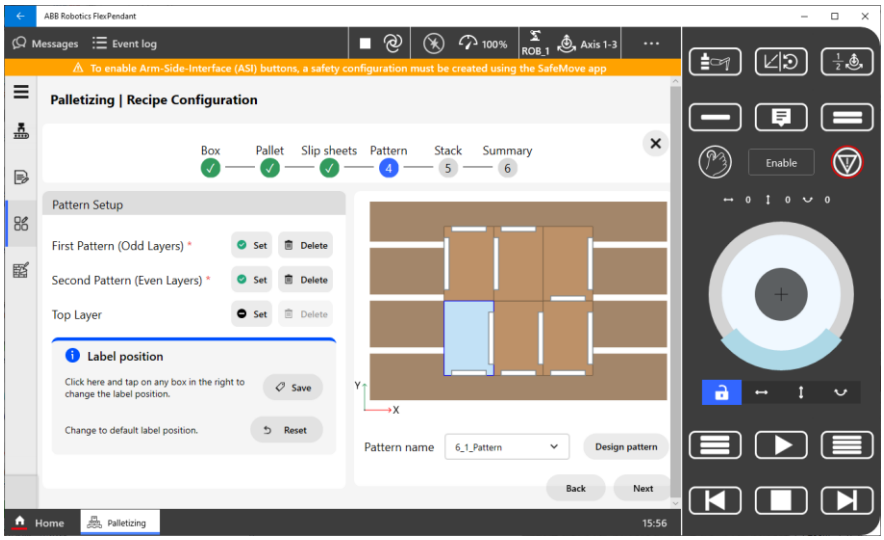
The Palletizing template automatically calculates the label orientation based on the following two rules:

1. Boxes are oriented such that the labels face a much as possible to the outside of the pallet.
2. If it's not possible to orient the box so that all labels face to the outside of the pattern, the box will be oriented with the label on the long side facing to the outside of the pattern.

If the presented box / label orientation is not desired it's possible to change this. First the Change button needs to be pressed.



The box for which the label orientation needs to changed, can be selected in the pattern preview, by pressing on this box. The selected box will now light up blue. Pressing again on the box will rotate the box 180°.



With the change function active the orientation of different boxes can be changed. When finished press Save.

To return to the automatically calculated label orientation, press Reset.

Pattern designer

The design of the pattern lay-out can be changed, or a new one can be created, form the pattern designer. The pattern designer can be opened from the Recipe configurator wizard step as well as from the hamburger menu.

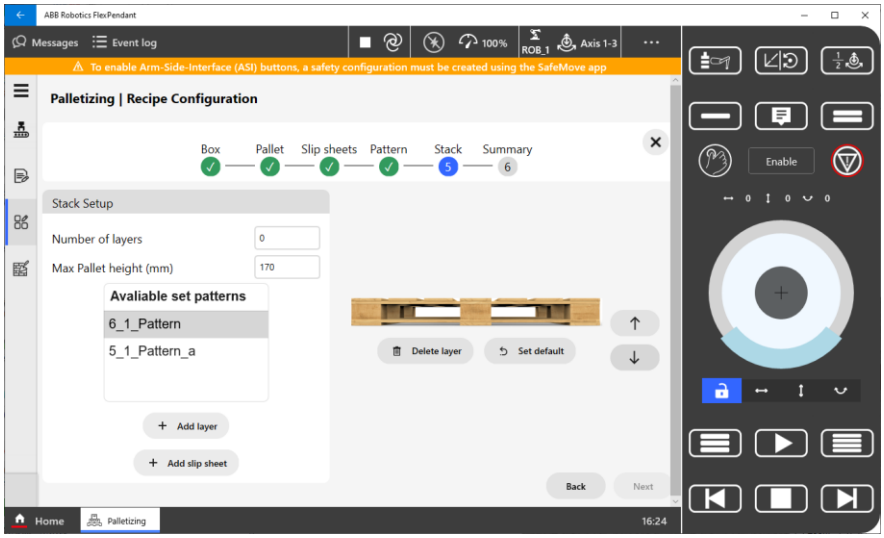
A detailed description of the pattern designer is given in chapter 4.4. Once the pattern design is finished and the pattern designer is closed the Recipe configurator wizard is returned to.

4.1.5 The Stack wizard step

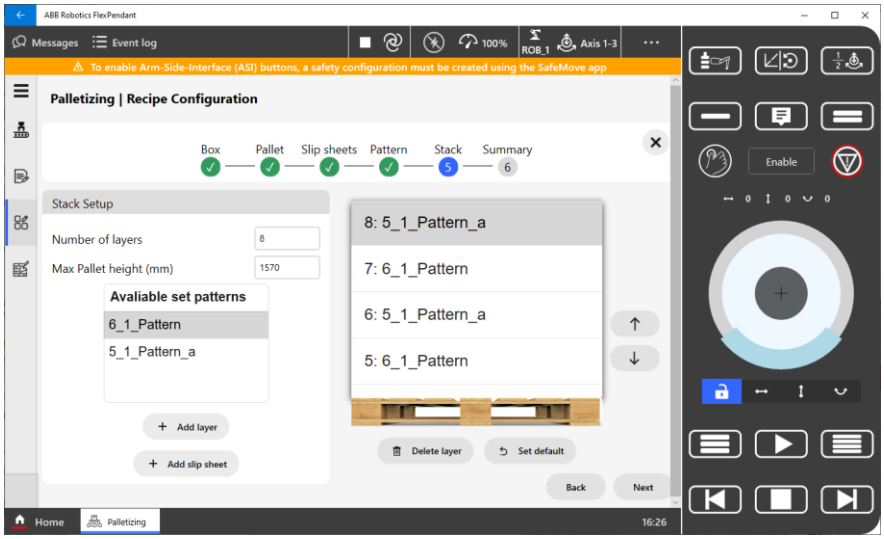
Description

In the Stack wizard step the stack configuration, which is the order in which the layers will be built on the pallet, is shown.

When a new recipe was started the stack is empty. To build a stack either the Number of layers or the required Max Pallet height need to specified in the respective fields.



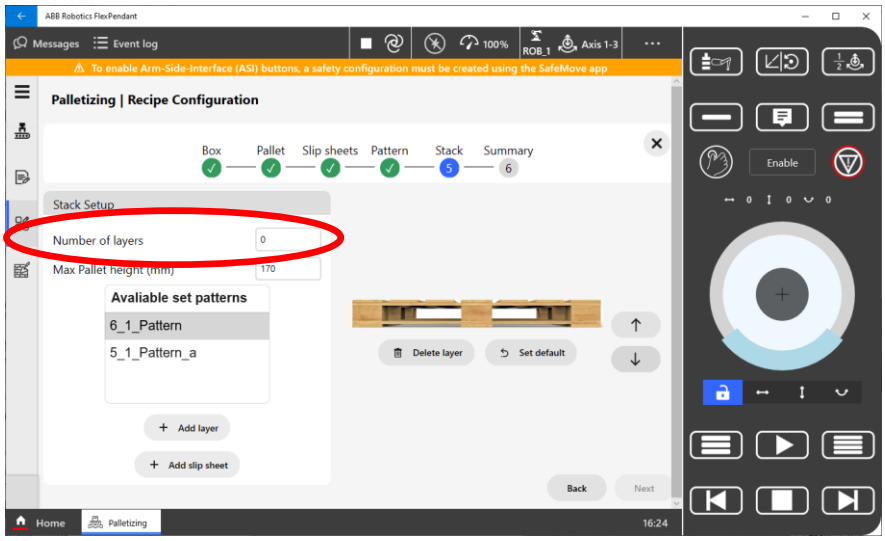
When an existing recipe is modified the stack configuration will be shown on top of the image of the pallet. The number of layers and stack height are also shown.



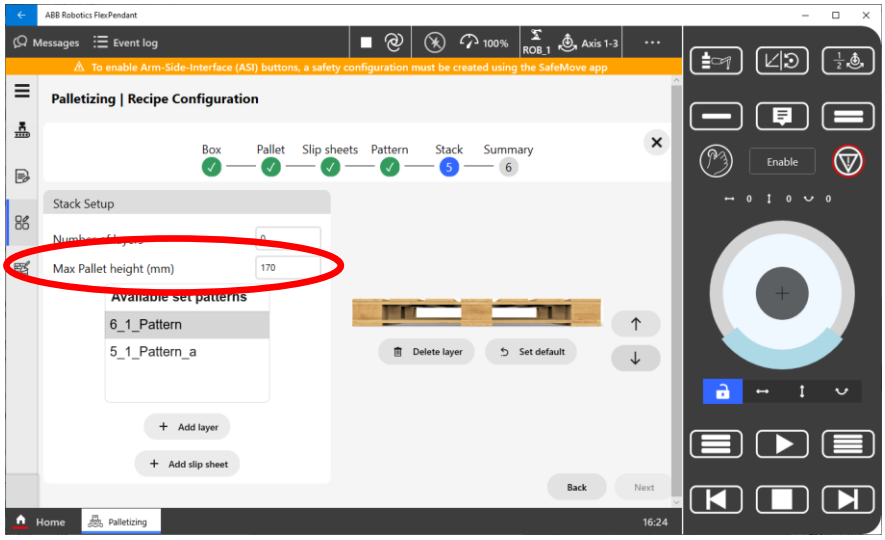
Number of layers and max pallet height

The number of layers can be added in three ways:

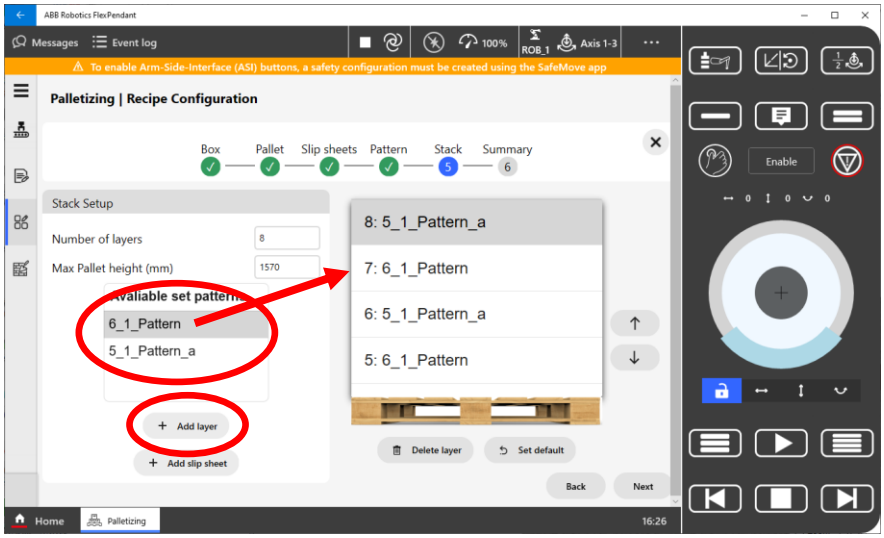
- Enter the required number of layers in the Number of layers field. Upon entry of a value the layer order is calculated, based upon the patterns selected in the Pattern wizard step. When the number of layers is changed the layer order is recalculated. Added slip sheets will remain at their position, except for the slipsheet that were selected to layers above the newly chosen number of layers. The Max Pallet height field will automatically be calculated upon entry of a number of layers.



- Changing the Max Pallet height field value will also result in a recalculation of the number of layers and will and will update the stack configuration in an identical way as described above. Setting the Max Pallet height will cause the Number of layers field to be updated.



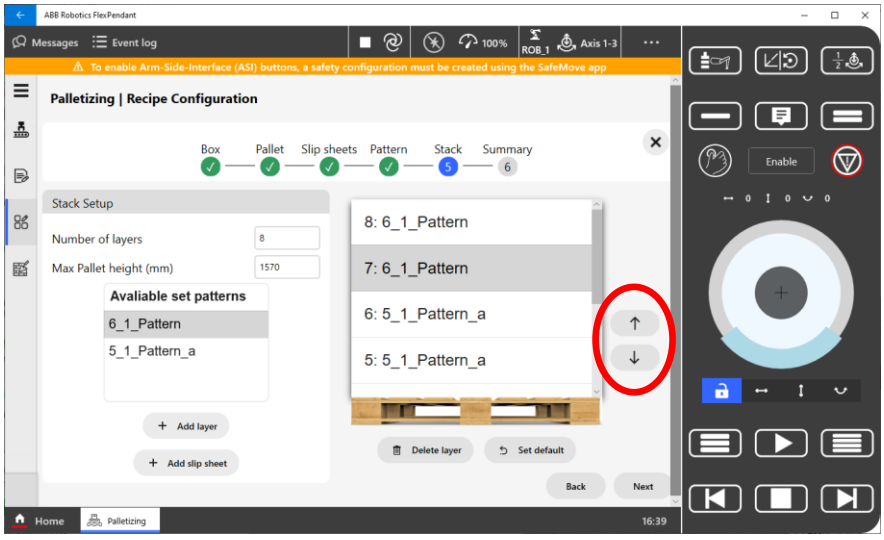
- Pressing the Delete layer button will remove the selected layer from the stack list. A pattern selected in the left hand list can be added to the stack by pressing the Add layer button.



Adding and deleting layers will also update the Number of layers and the Max Pallet height field.

Change stacking order

A different stacking order can be obtained by selecting one of the layers. Using the arrow buttons the selected pattern can be moved up and down through the stack.

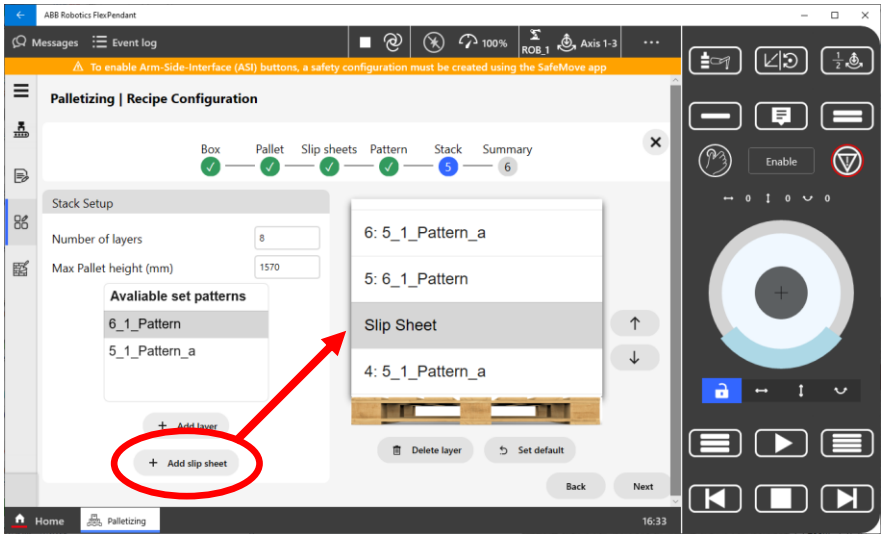


Pressing the Set default button will set the stack configuration back to its default status. This will also remove the slip sheets, if present.

The stack configuration is automatically set to default, when patterns are added or removed from the Pattern wizard step.

Slip sheets

Slip sheets can be added by pressing the Add slip sheet button. The slip sheet is then added above the selected layer. Only one slip sheet can be added per pattern layer. The presence of a slip sheet is indicated by Slip Sheet.



With the arrow keys slip sheets can also be moved up and down through the stack.

A slip sheet will remain on the layer it was added, even when a pattern or slip sheet at a lower layer is deleted from the stack. However, only one slipsheet can be a present on top of the pallet or a pattern layer.

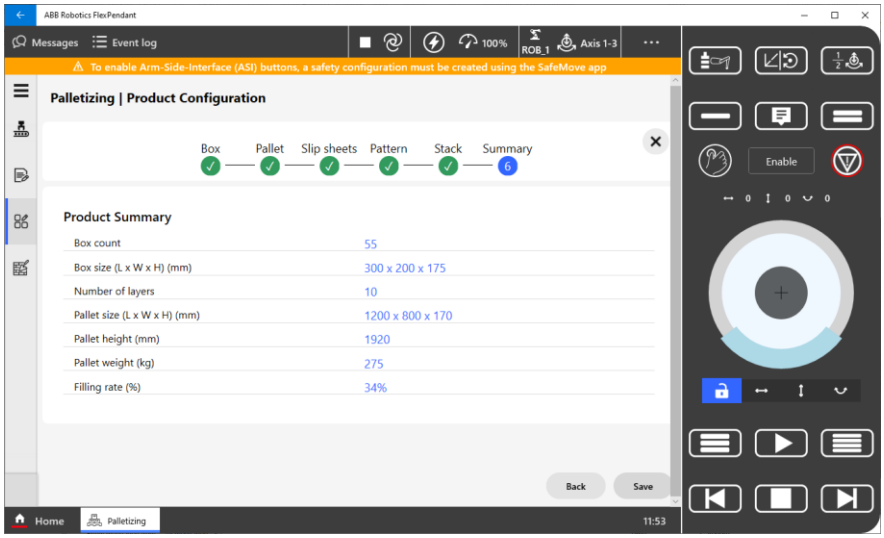
4.1.6 The Save wizard step

Description

Once all configurations as described in the previous chapters are finished the recipe can be saved.

Statistic information about the recipe is available.

Information	Description
Box count	Total amount of boxes that will be placed by the robot on the pallet.
Box size	Box dimensions
Number of layers	Number of layers on the pallet. Value doesn't include slip sheets
Pallet size	Pallet dimensions
Pallet height	Total height of the pallet. This includes the box layers, slip sheets and the pallet.
Pallet weight	Total weight of the boxes on the pallet. The weight of the pallet is not included.
Filling rate	The percentage of pallet space that is occupied by boxes on the entire pallet.



Saving the recipe will end the product configuration process. If the recipe is not to be saved the recipe configurator can only be exited using the X button.

When saving the recipe, the recipe will be stored on the flash drive of the robot controller. The recipes will be stored in a fixed location. The recipes are stored in the HOME:\Palletizing_files\Recipes folder.

By pressing the Save button several algorithms will be executed to prepare the recipe for stacking by the robot.

Below a description of the different algorithms is given. These calculations are done automatically by Palletizing Template and do not require any user interaction.

Once the recipe is saved it is ready for production.

Box matching

Box matching is an algorithm used for multiple box handling. With box matching Palletizing Template will automatically look for the optimal amount of boxes to handle for each cycle in order to build the layer with the least amount of cycles, thus increasing productivity. Additionally the groups of boxes handled per cycle is chosen is such way that the outer edges of the gripper stay within the pallet as much as possible.

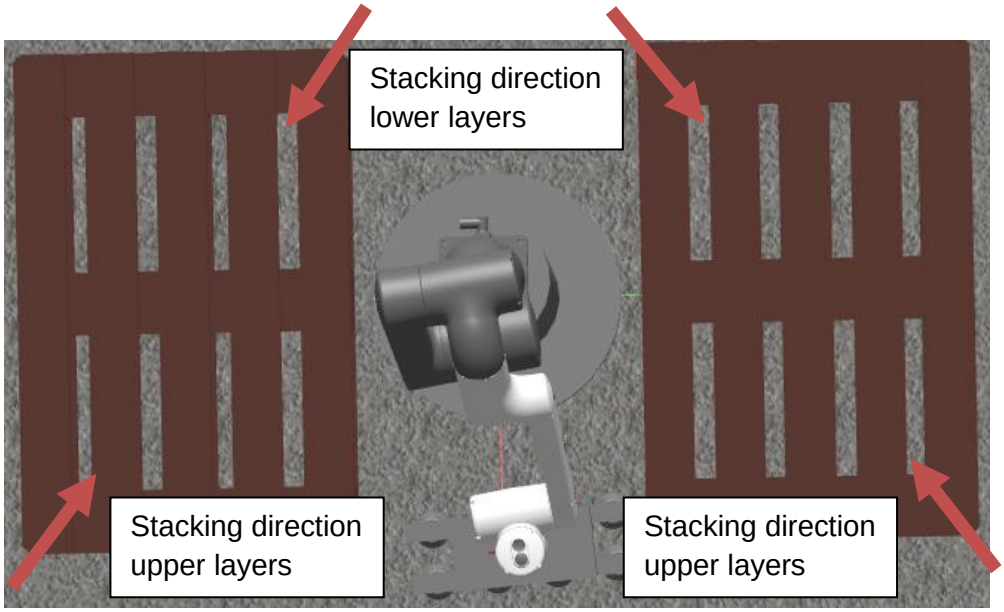
All patterns can be used for single or multi box handling as the Palletizing Template calculates the optimum stacking cycles based on the specified maximum amount of boxes.

Box place order

The order in which the boxes are placed on the pallet is calculated. The order is based on starting to place boxes in one corner and then build the pattern towards the opposite corner of the pallet. Since building the pattern can start in any of the four corners the optimization is thus performed four times, once for each corner.

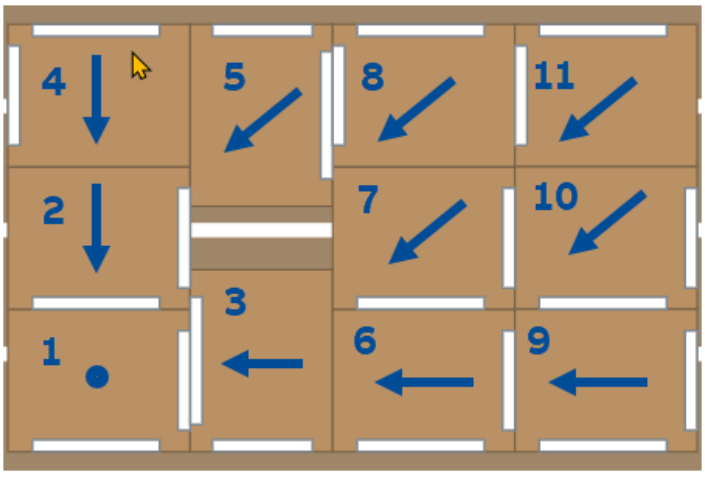
In the RAPID program the choice is made from which corner the robot should start building the pattern.

Typically the lower layers are build starting at the corner closest to the center of the robot. This will improve accessibility, and will make the best use of the robots workarea. Once the robot reaches about halfway the stack it is recommended to start from the opposite corner. This will improve reachability for the top layers.



Approach direction

Approach direction is calculated next for all the boxes in the pattern. In the example below the calculated order in which the boxes are placed (number indicated in red) is shown as well as the approach direction (blue arrow). The first box is always placed straight down. The boxes on the outside of the pattern are placed in a direction perpendicular to the boxes already on the pallet. The other boxes approach at an angle of 45°.



The actual offset values for the approach motion are set in the RAPID program in the Settings module.

Pick and place orientation

Finally the pick and place orientation are calculated. The pick and place orientation define the way the boxes are to be picked and placed to get the labels of the boxes in the desired position and orientation.

The orientation is calculated such that when an off-center gripper is used, the center of the robot flange is kept is close as possible to the center of the pallet. In this way the workrange of the robot will be optimized as much as possible.

4.2 Running production from the FlexPendant app.

Introduction

Production running consists of four topics:

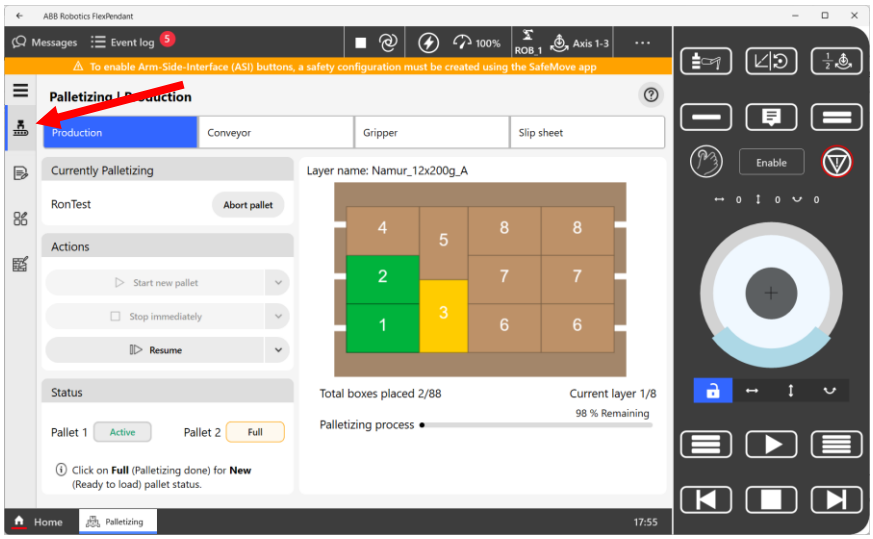
Topic	Description
Production	Select, run and stop a recipe.
Conveyor	Conveyor status view.
Gripper	Gripper status view and manual functions.
Slip sheet	Slip sheet status view

The production screen is activated from the Palletizing Template main screen, by selecting the Production screen.

4.2.1 Production screen

Description

When the production window is selected from the Palletizing Template main screen, the Palletizing | Production screen is shown.



The production screen is divided in 4 areas:

Name	Description
Currently palletizing	Abort the current recipe and select a different recipe for production
Actions	Start, stop, home run etc. functions.
Status	Pallet status indication and report empty pallets
Progress indication	Pattern preview with indication of the boxes already placed, still to place and placing.

The different screen areas are described in more detail in the following chapters.

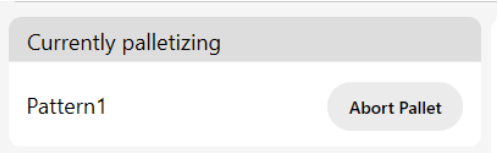
4.2.1.1 Change recipe

Description

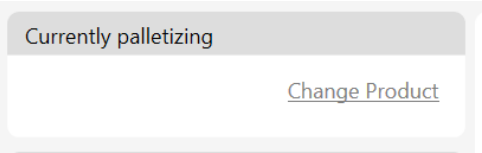
The palletizing template is designed to easily change recipes when a product changeover is required. A choice a be made from a number of existing recipes. When needed new recipes can be created from the Palletizing Template or existing recipes can be modified.

Currently palletizing

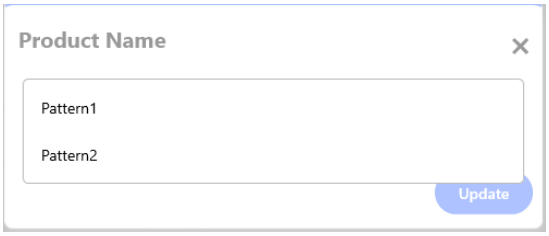
Shows the recipe that is currently in production.



When a different recipe should be taken in production the current recipe can be aborted by pressing the Abort Pallet button. It's then possible to select a new recipe by pressing the Change product button.



From the drop down list the required recipe can be selected. When the required recipe is selected press the Update button to activate the recipe.



Production can be started as described in the next chapter.

4.2.1.2 Production start and stop actions

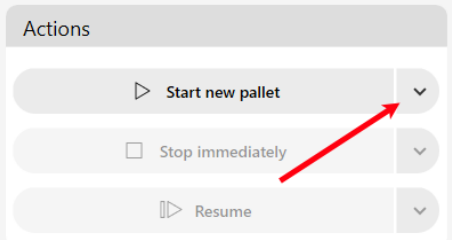
Production start actions

Production start can only be performed when a recipe is selected. If there is no recipe is selected all buttons in the Actions dialog will be grayed out.

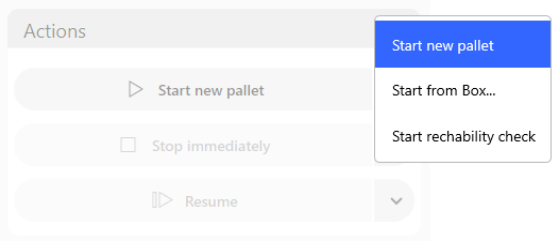
Starting the production can be done in two ways.

Action	Description
Start new pallet	Start the production with a new pallet. An empty pallet must be present in the cell.
Start from box	A partially finished pallet can be completed by starting the palletizing from a specific box at a specific layer.
Start reachability check	Starts a test to verify if all programmed positions on the pallet are reachable by the robot. This check helps identify potential reachability issues before starting the actual production.

To select between the alternative start actions, press the arrow point on the right-hand side of the Start new pallet button. Start new pallet is selected by default.



This will open a list with the above-mentioned start actions.



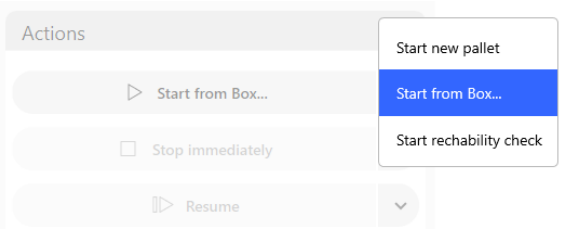
Start new pallet

The start new pallet action will always cause the robot to start palletizing on an empty pallet. So, an empty pallet must be present at the pallet position.

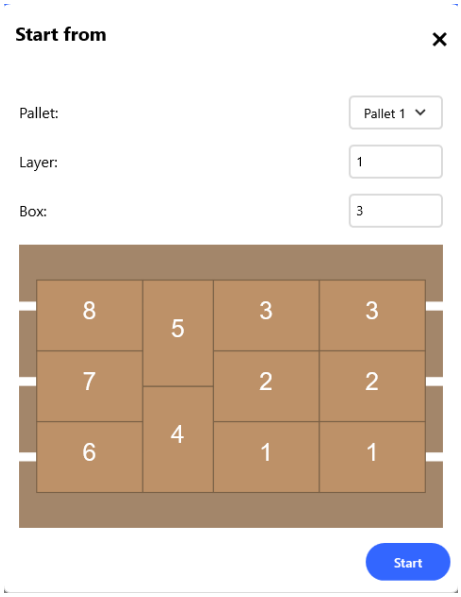
To start a new pallet a new pallet must be reported present by the user. See chapter 4.2.1.4 for more information on how to report an empty pallet.

Start from box

An unfinished pallet can be completed which means that palletizing can be started from a particular box at a particular layer. From the Actions menu select the Start from box action.



By pressing the Start from box action a dialog will appear, where the pallet, layer and box to start palletizing from need to be selected. The pattern shown once the pallet is selected.



By pressing Start palletizing will start on the selected pallet from the selected layer and box.

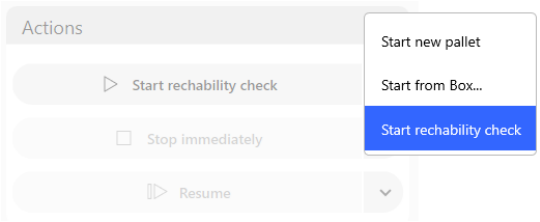
Note

Make sure that the pallet, layer and box are properly selected and are in accordance with the state of the physical pallet.
Collisions may occur if incorrect settings are used!

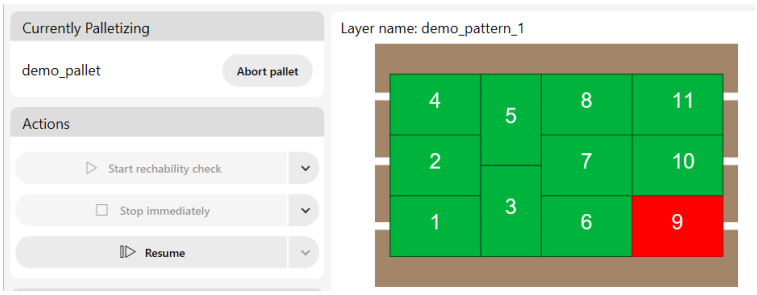
Reachability check.

The reachability check allows for verifying if all target positions on the pallet are reachable by the robot. This helps to identify any positioning issues before production starts.

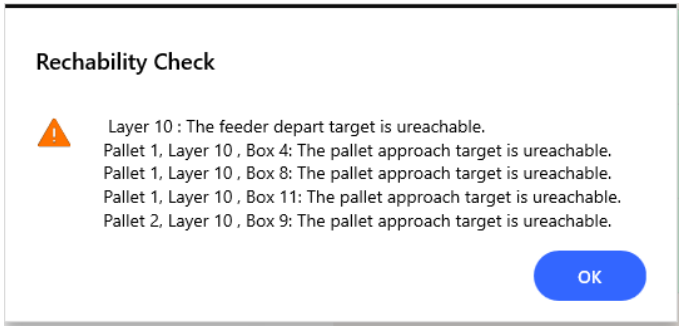
From the Actions menu, select the Start reachability check action.



By pressing the Start reachability check action, the system will begin validating each box position on every pallet included in the active recipe. During the check, a visual indicator shows the current progress directly on the pallet pattern view. Each box is highlighted briefly to reflect the current box under inspection.



At the end of the process, a Reachability Check summary will be shown as a popup. This contains a list of all unreachable targets that were detected.



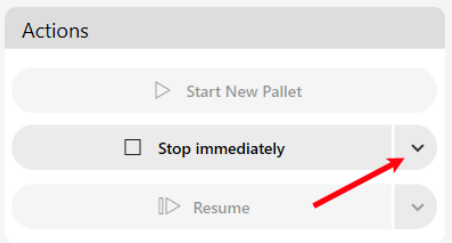
This allows the user to make corrections to the pattern, robot configuration, or pallet placement before proceeding with actual palletizing.

Production stop actions

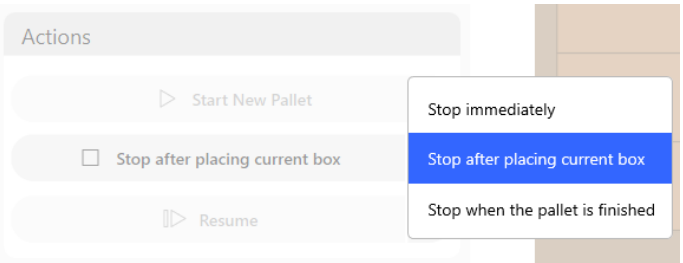
Once the production is running the production can be stopped again if needed. To stop production the following actions are available:

Action	Description
Stop immediately	Same functionality as the Stop button
Stop after placing current box	The robot finished the current cycle and once the box.is placed on the pallet the program execution is stopped.
Stop when the pallet is finished	The robot will continue to execute the program until the active pallet is full. Program execution is then stopped.

During production running the Stop immediately is selected by default. To select the other alternative stop actions, press the arrow point on the right-hand side of the Stop immediately button.



This will open a list with the above-mentioned stop actions.



The chosen stop action will not have any impact on the fixed stop button on the FlexPendant. This button will always stop the robot immediately.

When the Stop after placing current box or Stop when the pallet is finished action is active it is still possible to stop the robot immediately while the robot is finishing the cycle or pallet.

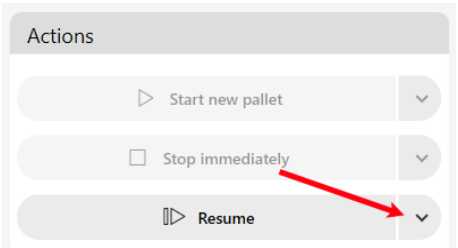
> **Note**
>
> **The fixed button on the FlexPendant will always stop the robot immediately.**

Production resume actions

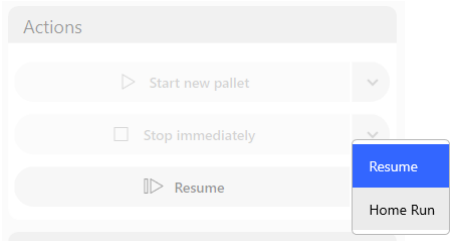
Once the robot program is stopped following actions are available.

Action	Description
Resume	Same functionality as the Start button
Home run	The robot will move to the home position. Production can be continued from there, without having to remove partially build pallet stacks. When the home run was started, after a Stop immediately, between pick and place, this box will be considered as not being placed. So the same cycle will be repeated if the production is continued. If the box was already placed on the pallet, the cycle will not be repeated.

At this time the Resume function will be selected by default. To select the alternative functions, press the arrow point on the right hand side of the Resume button



This will open a list with the above mentioned functions.



The chosen function will now be active for the Stop button in the Palletizing Template app.

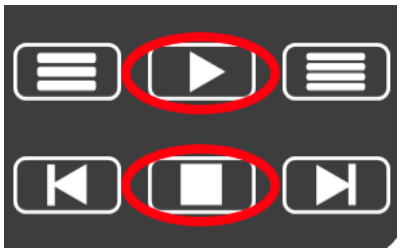
The fixed start button on the FlexPendant will not be affected by the selected stop mode. This button will always continue program execution.

Note

The Home run function will abort the current cycle and move the robot to the home position. When the program is resumed the same cycle is repeated in case the box wasn't placed on the pallet yet.
If a box is still present in the gripper it must be disposed of. It should also be checked that the place position on the pallet is free.

Fixed FlexPendant buttons

The FlexPendant has several fixed buttons to start and stop the robot program. This concerns mainly the start and the stop button



The Play button will always continue program execution

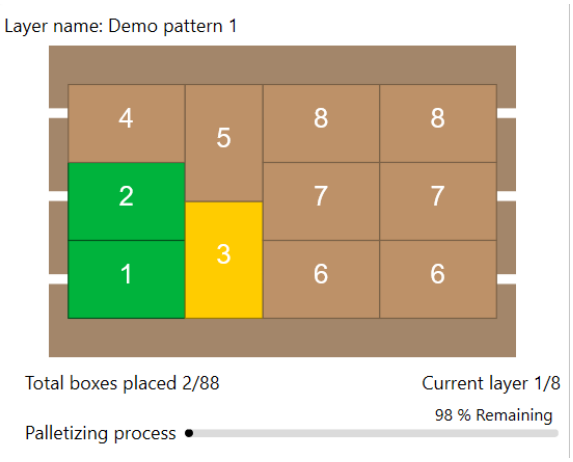
The Stop button will always stop program execution immediately.

Any selected action in the action area of the Palletizing Template will not be considered by the fixed FlexPendant buttons.

4.2.1.3 Stacking progress indicator

Description

In the center area in the production menu of the Palletizing Template App, the stacking progress indicator is shown.



The graphical view shows the pattern for the current layer. The boxes in this pattern can have different colors based upon the status of the box. An overview of colors used for the different box statuses is given below.

Status	Description
Green	This box is present on the pallet.
Yellow	This is the active cycle. I.e. the robot busy placing this box on the pallet.
Brown	Box is part of the pattern but not yet placed on the pallet.
Red	The box is lost.

Each box also has a number. The number indicates the order in which the boxes are placed on the pallet. In a number of cases two or three boxes ca have the same number. Boxes with the same number will be handled at once in a Multi Pick application.

Additional statistical info is given for the layer.

Info	Description
Layer name	The name of the layer the robot is currently palletizing or depalletizing

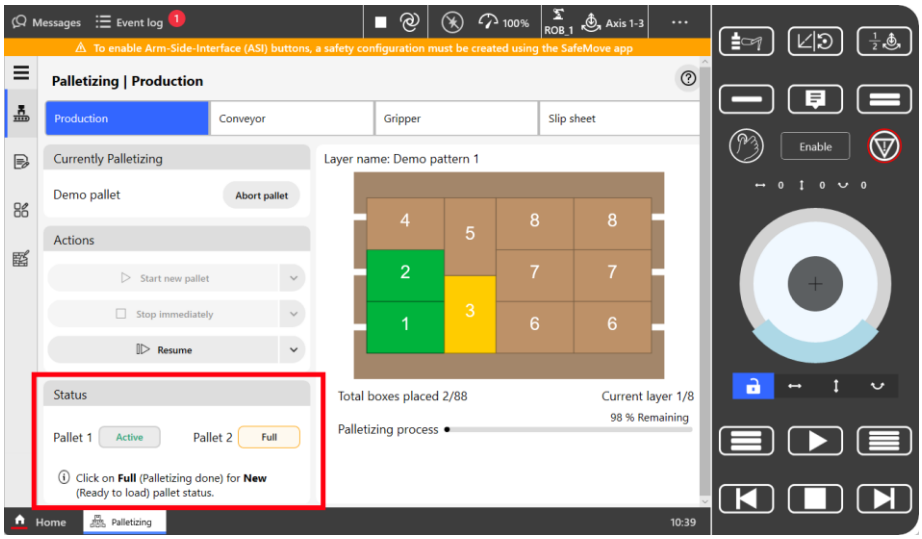
4.2.1 Production screen

Total Boxes placed	This shows the current running box as well as the total amount of boxes on the pallet.
Palletizing progress	Bar indicator showing the progress of the current stack, including the percentage of boxes that still need to be placed on the pallet.
Current layer	Shows the current running layer as well as the total amount of layers.

4.2.1.4 Pallet status

Description

The status of each pallet in the cell is displayed in the Status area of the production screen.



The buttons show the following status for each pallet:

Status	Description
None	Status not available.
New	A new empty pallet is reported.
Active	The robot is palletizing.
Full	The pallet is full.

Only one pallet will can have the Active status, which is the pallet the robot currently palletizing. The other pallets can either have the status New or Full. Pallets reported as New will be handled as soon as the active pallet is full and thus receives the Full status.

One button is available per active pallet position. As a maximum of 4 pallets are supported A maximum of 4 buttons can be displayed. Whether or not a pallet / button is used is to be configured in the AppAllowRelease variable. See chapter 3.1.4.1 on how to configure the buttons.

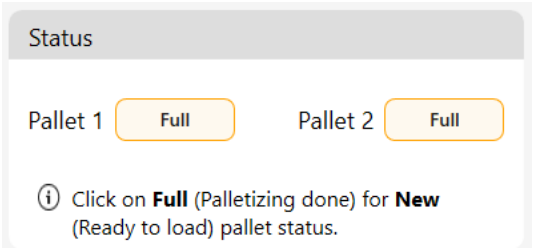
Reporting a new pallet can be done in two ways:

Action	Description
From FP App	Use of the status buttons in the Palletizing template to report the presence of a new pallet
IO based	Push buttons can be connected to dedicated predefined IO to report a new pallet.

A combination where a new pallet can reported both through the Palletizing template or IO is supported

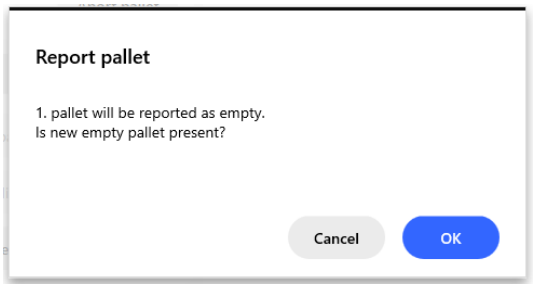
Report new pallet from the Palletizing template

To report the presence of a new pallet in the robot (the full pallet has be replaced by an empty one) press the button corresponding to the pallet to start at.



Buttons can only be pressed when the button shows Full.

Pressing the Full button for a particular pallet requires a confirmation.



Once confirmed the robot will start palletizing.

Note

Make sure that the pallet at the selected position is empty before confirming the new pallet. The robot may immediately start palletizing at this location.

Report new pallet using IO

New pallets can also be reported IO based. This means that for example a push button can be foreseen close to the location where the pallet is physically removed. The push button should be connected to the diNewPalletX input for the corresponding pallet.

See chapter 3.1.4.2 for more information on the external new pallet report.

Note

Make sure that the pallet at the selected position is empty before confirming the new pallet. The robot may immediately start palletizing at this location.

4.3 Tuning

Description

During production several parameters can be tuned to improve the performance of the system and / or the improve the quality of the stack. Common parameters that require tuning are robot speeds and acceleration, waiting time in pick and / or place position and the dimensions of the boxes. These parameters can be altered on the fly, while robot is in production. A stop of the robot program execution is not required.

4.3.1 Product dimension tuning

Description

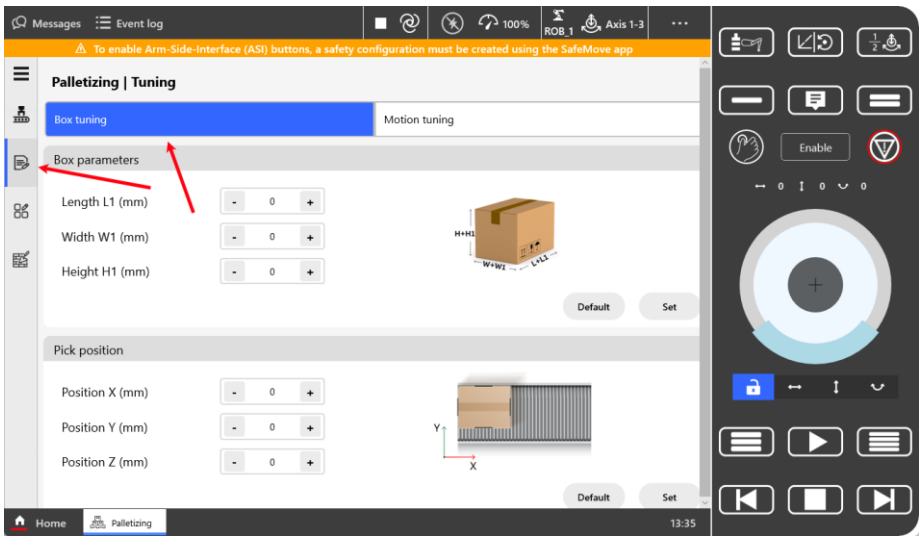
In palletizing applications, tolerances on the boxes are a common issue. Product tolerances typically have a major influence one the quality of the stack. In Palletizing Template it is possible to compensate for these product tolerances during production.

If the boxes are bigger then specified this results in the robot pushing the boxes against each other on the pallet. This results typically in a layer that looks uneven on the outside.

If the boxes are smaller than specified gaps between the boxes will appear, making the stack less stable.

Tolerances to the height of the box can have a big impact especially on the top layers, since these tolerances accumulate as the pallet gets higher. This will result in the robot either pushing the boxes onto the previous layer, with increasing force, or dropping the boxes from an increasing height on the previous layer.

To optimize the product tolerances, the Palletizing | Tunning screen and Box tuning tab should be selected as shown in the image below.



The box dimensions as well as the pick position can be tuned independently.

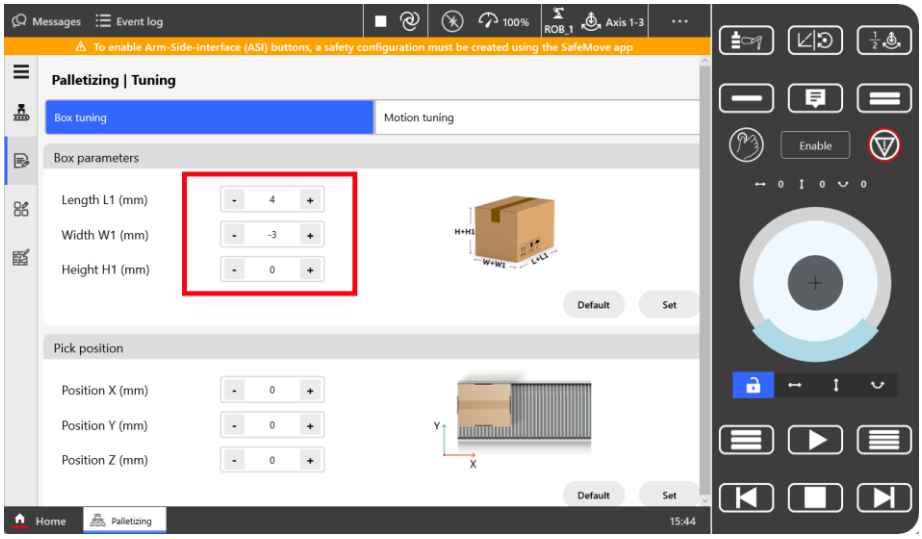
Tune box dimensions

In the tune box dimension dialog the actual box size can be optimized for the length, width and height under Box parameters.

By tapping on the numerical field behind for the box dimension to tune, the tolerance of the physical box dimensions can be entered with the aid of a

numerical key pad. With the “+” and “-“ buttons the tolerance can be increased or decreased in steps of 1 mm.

The maximum allowed tolerance is limited to +/- 20 mm.



Once the desired box tolerances are set the tune values need to be written to the robot controller by pressing the Set button. The Default button will update the fields with the actual values from the robot controller.

The tune values for the length and the width will be applied when the robot starts with a new layer. The height correction will be applied immediately.

Note

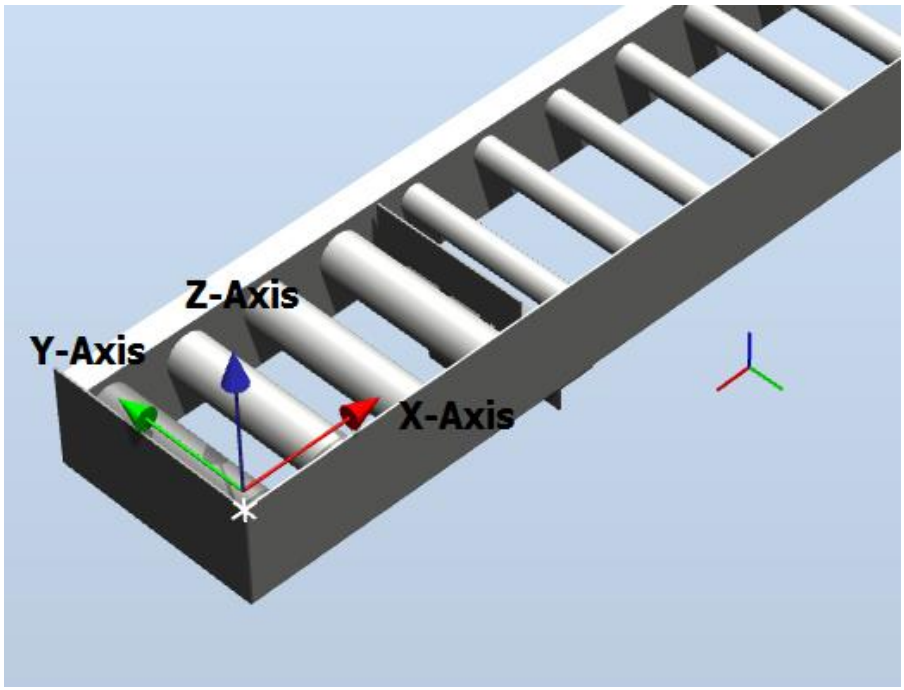
The new settings will only be written to the controller when the *Set* button is pressed. The settings will become active at the next cycle.

Tune pick position

In order to achieve pallet stack with even surfaces on the outside, the robot should pick the boxes with the gripper well aligned with the box. If this is not the case the pick position can be optimized by tuning the pick position. Tuning the pick position can be done under Pick position.

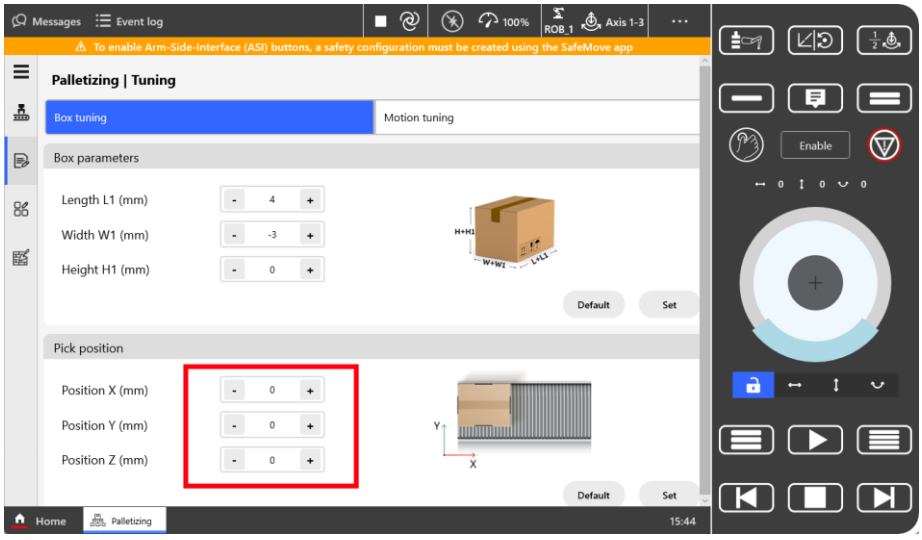
The pick position can be tuned in three directions, X, Y and Z as shown in the image below. The X-axis is indicated by the red arrow, the Y-axis by the green arrow and the Z-axis by the blue arrow.

4.3.1 Product dimension tuning



By tapping on the numerical field behind for the box dimension to tune, the tolerance of the physical box dimensions can be entered with the aid of a numerical key pad. With the “+” and “-“ buttons the tolerance can be increased or decreased in steps of 1 mm.

The maximum allowed tune value is limited to +/- 20 mm.



Once the desired feeder offsets are set the tune values need to be written to the robot controller by pressing the Set button. The Default button will update the fields with the actual values from the robot controller.

The tune values are applied from the next cycle.

4.3.1 Product dimension tuning

Note

The new settings will only be written to the controller when the *Set* button is pressed. The settings will become active at the next cycle.

4.3.2 Motion tuning

Description

The OmniCore controller has a high level motion control, consisting of two motion principles, QuickMove and TrueMove. These two principles make that the robot is accurately following the programmed path, with the highest possible speed.

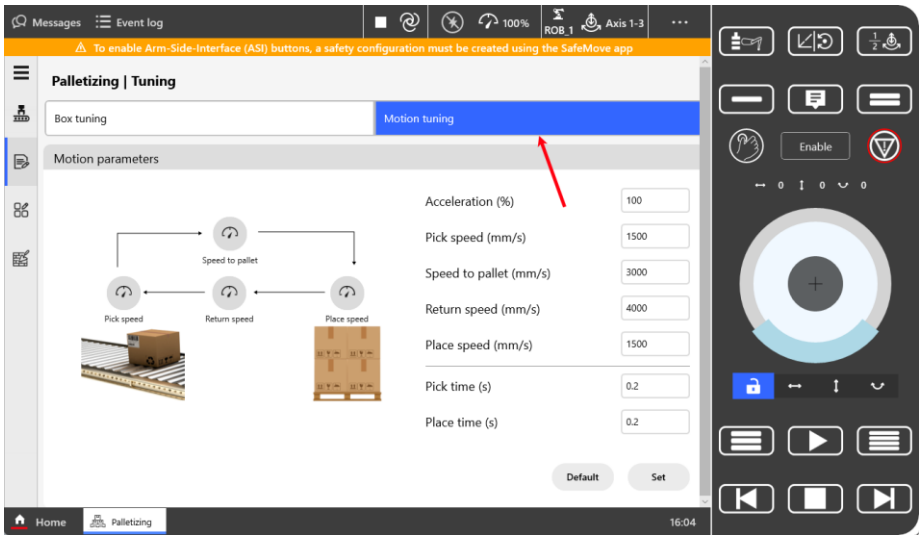
In some cases, mainly depending on the box size and weight, the motion needs to be optimized.

This is for example the case when the robot is handling heavy and / or high boxes. In these cases speed or acceleration can be reduced to prevent the robot from losing box keep the boxes in a more stable way.

When handling for example smaller and lighter boxes, speed and acceleration may be set to a higher or to the maximum value.

If the performance of the robot is higher than the required production volume, speed and acceleration can also be reduced. Reducing speed and acceleration typically improve reliability.

The motion parameters can be tuned from the Palletizing | Tuning screen in the Motion tuning tab.



The following parameters can be tuned:

Parameter	Description
Acceleration	The acceleration can be reduced for the movement with a box from the pick position to the place position.
Motion speed	Speed used for the motion cycle.

Pick- place time time	Time the robot waits in the pick or place position for the gripper to properly pick or release the box

Once the desired motion parameters are set the parameters need to be written to the robot controller by pressing the Set button. The Default button will update the fields with the actual values from the robot controller.

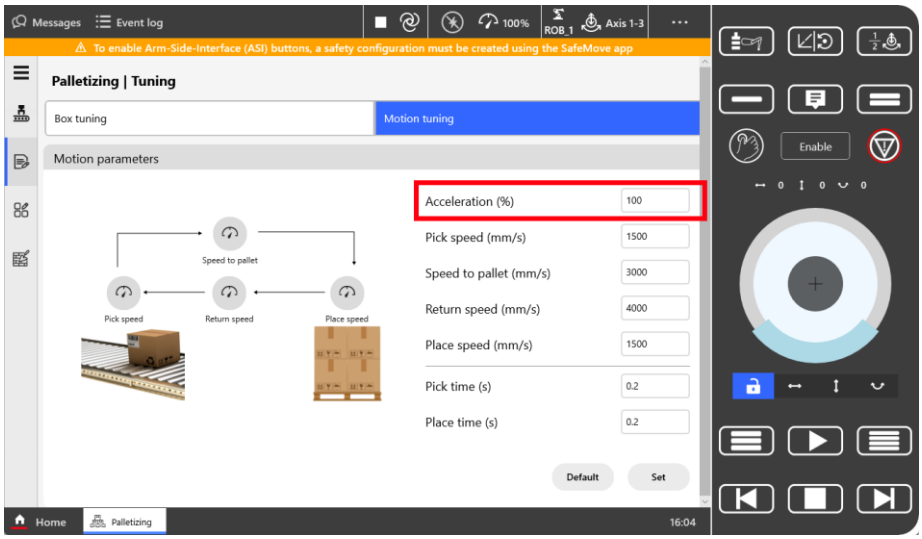
The motion tuning parameters are automatically saved with the recipe so the next time the recipe is started again the same motion parameters will be applied.

Acceleration

When the robot needs to handle for example heavy boxes with a vacuum gripper the default acceleration may too high for this type of boxes. E.g. the robot could lose the box while moving to the pallet. Reducing the acceleration is usually the preferred way to resolve this problem and is preferred over reducing the speed.

It's only possible to change the acceleration of the movement with the box. The return motion will always be at maximum acceleration, since the robot doesn't hold any boxes.

Change the acceleration in the Acceleration (%) field.



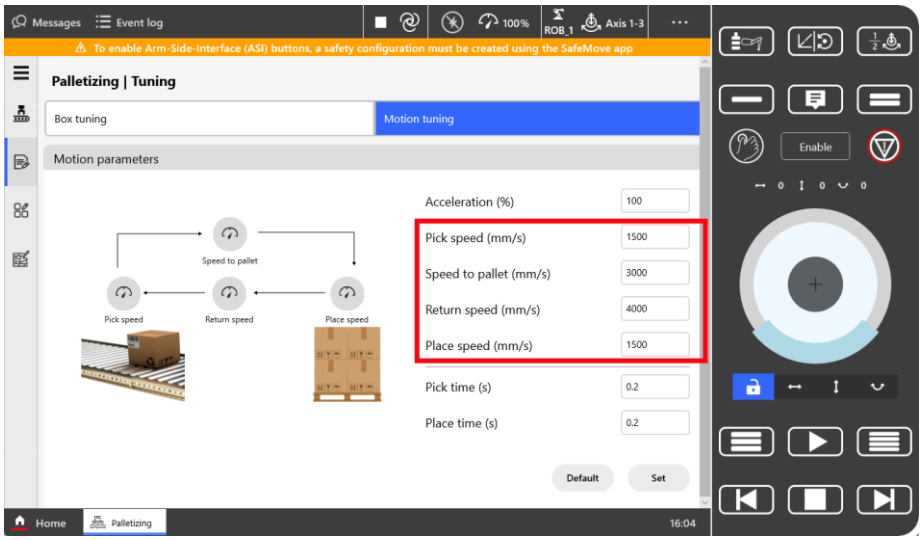
The acceleration is a parameter in %. The maximum acceleration is 100% and can only be reduced.

Press the Set button to write the acceleration parameter to the robot controller. Press Default to return to the original value. The robot will apply the changed acceleration the next cycle.

> **Note**
>
> **The new settings will only be written to the controller when the *Set* button is pressed. The settings will become active at the next cycle.**

Motion speed

The speed can be individually changed for all movements from the pick to the place position. For the return movement (without box only an overall speed can be changed. To change the speed press the button indicated in the image below.



The speed is a parameter in mm/s. The maximum speed depends on the robot type for a GoFa type robot this is 2200 mm/s for a large industrial robot this can be up to 7000 mm/s. The maximum speed is required this value should be chosen according to the robot type.

The speed can be changed for 4 individual movements

Parameter	Description
Pick speed	the speed at which the robot moves upwards after picking the box.
Speed to pallet	Speed used for the movement with the box to the pallet
Place speed	Speed used for the movement down to the place position.
Return speed	Speed at which the robot makes the entire movement back from the place to pick position. This is typically the maximum robot speed, as the robot doesn't hold any boxes.

Press the Set button to write the speed parameters to the robot controller. Press Default to return to the original value. The robot will apply the changed speed the next cycle.

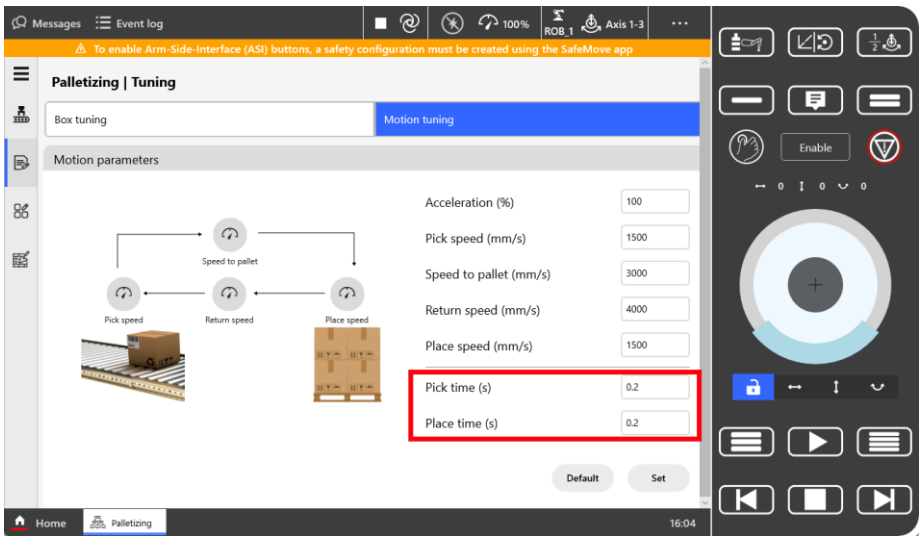
> **Note**
>
> **Pressing Set will update the values in the controller. If Set isn't pressed the entered motion speed will not be applied.**

Pick and place time

In the actual pick position the robot will need to wait some time for a vacuum gripper to reach sufficient vacuum to properly hold the box.

In the place position the robot will also need to wait some time to release the vacuum.

To change the pick or place time press one of the buttons indicated in the image below.



The pick or place time is given is seconds. The minimum time is 0.001 sec (1 msec).

Press the Set button to write the pick – place time parameters to the robot controller. Press Default to return to the original value. The robot will apply the changed times the next cycle.

Note

The new settings will only be written to the controller when the *Set* button is pressed. The settings will become active at the next cycle.

4.4 Pattern designer

Description

The recipes created by the Palletizing Template are based upon standard patterns which can be used to configure odd and even layers and if applicable a different top layer.

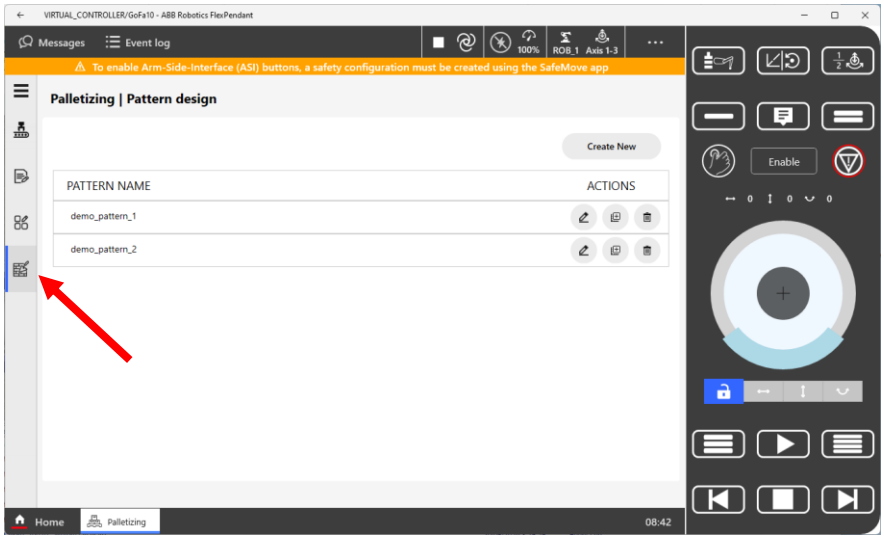
A pattern is a box size independent design of a layer. This means that the pattern is based on formula's instead of coordinates. Due to the nature of the formula's patterns automatically scale up or down when the box size changes. This happens both when creating or changing a recipe as well as when running production (tunning the box dimensions).

To create new patterns a pattern designer is available in the Palletizing Template app. This pattern designer uses a graphical interface to build the pattern as desired. The formula's defining the location of each box are automatically created by the pattern designer.

The pattern designer can be opened in two different ways.

Open pattern designer from the main view

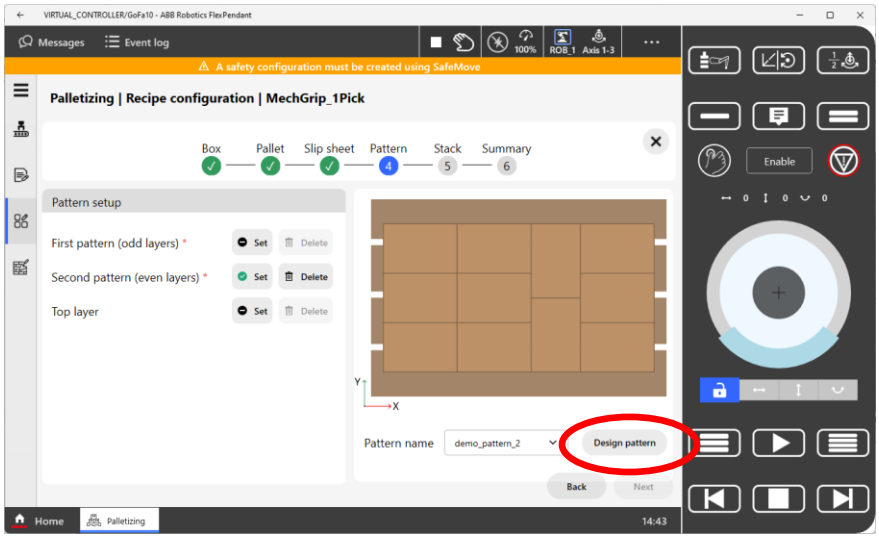
Tap the Palletizing | Pattern design button to open the pattern designer.



The pattern designer is used for both creating new patterns as well as for modifying existing patterns.

Open pattern designer from the recipe

The pattern designer can also be opened from the recipe from the Pattern tab.

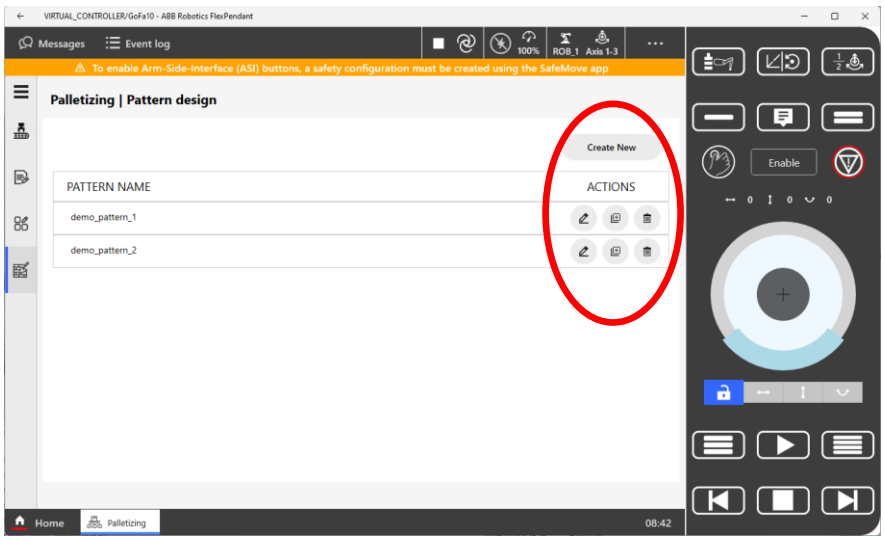


Press the Design pattern button to go straight to the pattern designer. Once a new patter is added or an existing one is modified the pattern can be used straight away in the recipe.

Pattern design start screen

The pattern design start screen appears as shown in the image below:

> **Note**
>
> **The patterns are formula based and are thus independent of box dimensions.**



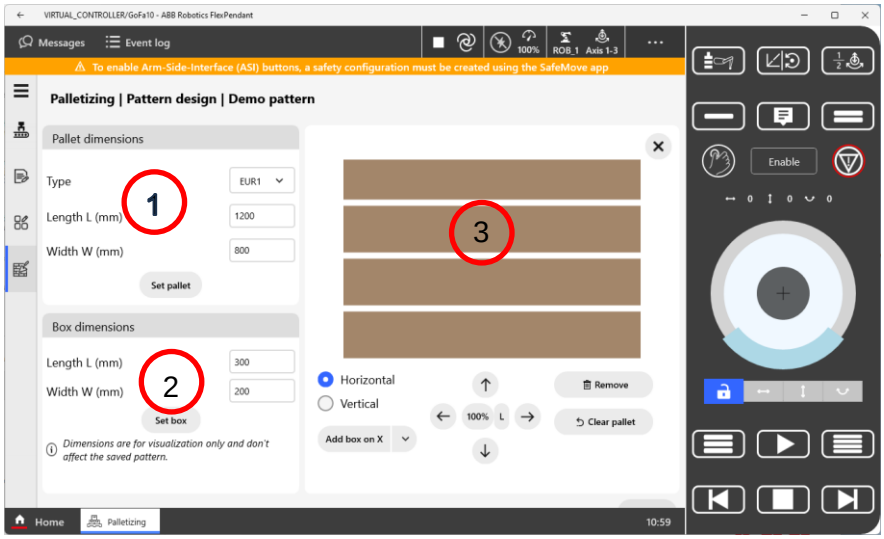
Following buttons are available for each pattern:

Button	Description
Create New	Create a new pattern. When pressed the user will be prompted to enter the name of the new pattern. Then the pattern designer will be started.
	Modify an existing pattern. When pressed the pattern designer will started.
	Make a copy of the pattern. When pressed the user will be prompted to enter the name for the copy of the pattern. To enter the pattern designer the modify an existing pattern will need to be selected
	Delete the pattern. When pressed the user is asked to confirm the deletion of the pattern. Patterns can be deleted without any impact on existing recipe's. The recipe, once created, contains all information needed to build the pattern. There is no dependency on the pattern files.

The buttons to modify, copy and delete a pattern are shown for each pattern.
Pressing such a button will activate the functionality for the related pattern.

Pattern design view

The pattern design view consists of 3 area's:



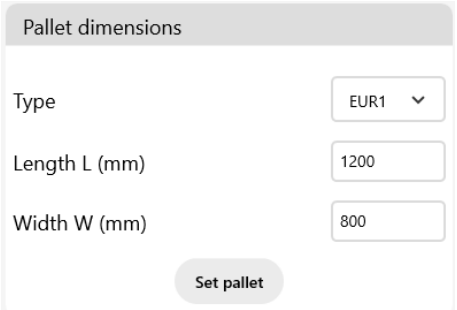
Index	Description
1	Define the pallet dimensions. The pallet dimensions are only used to represent the a pallet in the graphic view. The pallet dimensions can be changed while creating the pattern to check if the pattern fits on different pallet sizes. The pallet dimensions will not be stored in the pattern.
2	Define the box dimensions. The box dimensions are only used to represent the a box in the graphical view. The box dimensions can be changed while creating the pattern to check if the pattern fits on different pallet sizes. The box dimensions will not be stored in the pattern.
3	Graphic representation of the pallet and the pattern.

Note

Pallet and box dimensions will only be used for the graphical representation. They will not be stored in the pattern.

Pallet dimensions

For the pallet dimensions a selection can be made from the Type list of standard pallet types, with respective pallet dimensions. If the required pallet dimension is not available as a standard pallet, the length and width can be specified as well in the Length and Width field.

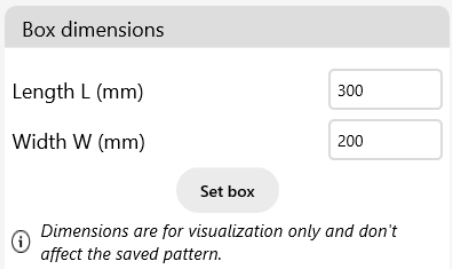


When the pallet dimensions are set to the desired values press the Set pallet button to activate the new values. The pallet size will immediately be scaled to fit the design area. The pattern will be scaled accordingly.

A pallet height can’t be specified as the pattern representation is in 2D only. The pallet dimensions will not be stored in the pattern. By changing the pallet dimensions a check can be made if the pattern fits different sizes of pallets. Pallet dimensions can be changed at any time during the editing of a pattern.

Box dimensions

The box dimensions can be specified in the Length and Width field.

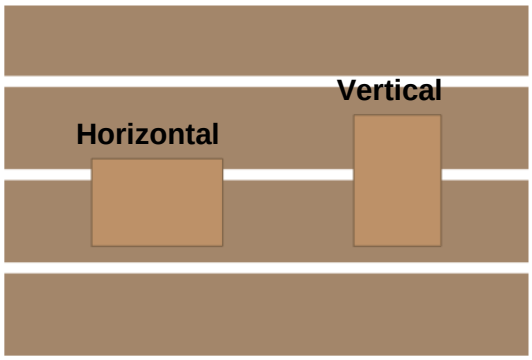


When the box dimensions are set to the desired values press the Set box button to activate the new values. The box size (pattern) will immediately be scaled to fit the design area.

A box height can’t be specified as the pattern representation is in 2D only. The box dimensions will not be stored in the pattern. By changing the box dimensions a check can be made if the pattern fits different sizes of pallets. Box dimensions can be changed at any time during the editing of a pattern.

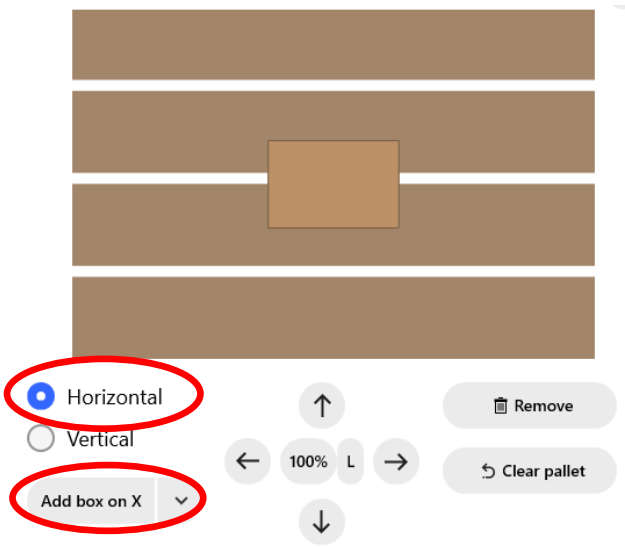
Designing the pattern

To build the pattern, a box can be placed either horizontal or vertical on the pallet.



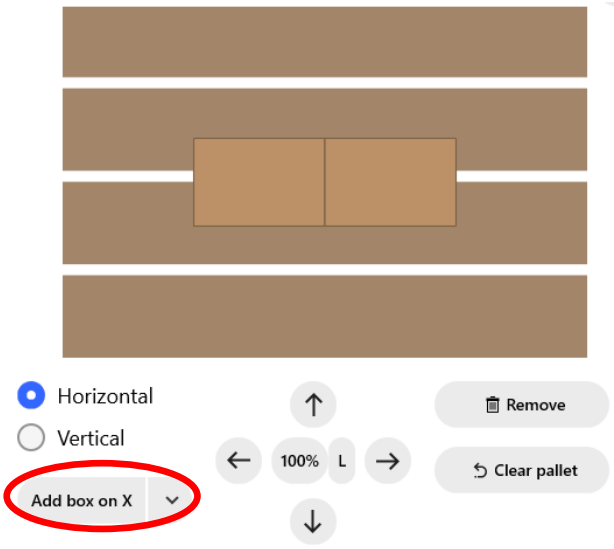
To place the first box on the pallet select either Horizontal or Vertical. Then click Add box on X. The box will now appear centered on the pallet. In the image below the box is placed horizontal on the pallet. Boxes are drawn in the default size of 300 x 200 mm.

The way to create a pattern is explained based on an example on the following pages.

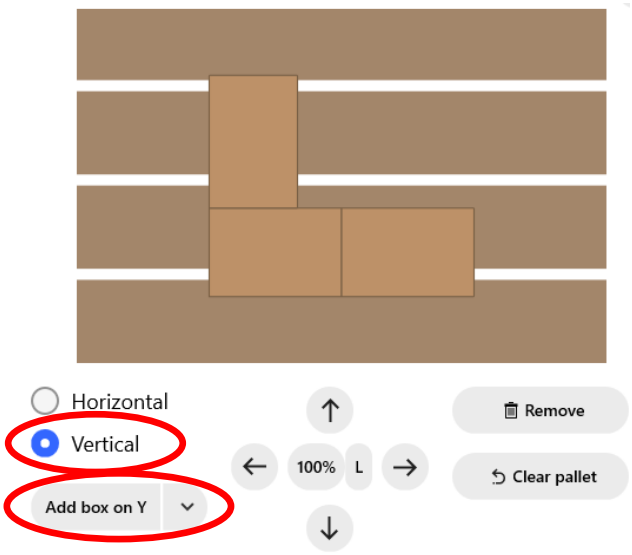


To place the second box next to the first box on the horizontal axis, press Add box on X again. The new box will be placed next to the existing box on the horizontal axis. The X axis represents the horizontal axis. The Y axis represents the vertical axis.

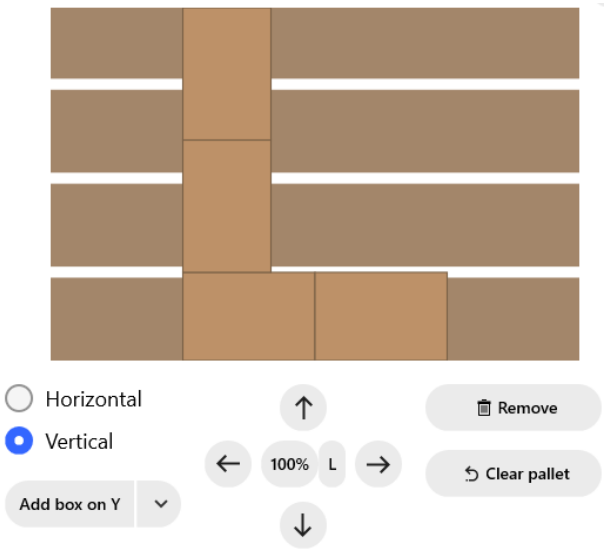
The so far build pattern is always drawn centered on the pallet.



To place a box vertically on top of the first box, select Vertical and set the Add box on X to Add box on Y, using the pull down menu on the right hand side of the button.



If a forth box has to be placed vertically against the third box and on top of the first box, the box can be added in Y direction. The box will appear on top of the third box.



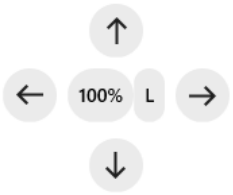
The box needs to be moved to the final position using the arrow buttons

Moving the box

Any box can be moved independently within the pattern, using the arrow buttons. Moving the box is done in steps. The size of the steps is a percentage of either the box length or the box width. By default the step size is 100%, meaning one box length or one box width.

It is important the box are placed as much as possible against another. This is particularly important for multi pick applications. In order to determine which boxes to pick at the same time these boxes must be placed strictly against each other. For this reason it’s advisable to move the boxes only with step sizes of 100%.

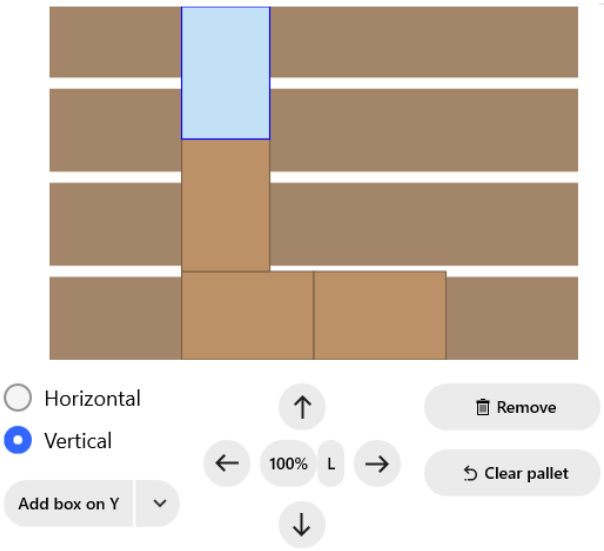
Below an overview is given on how to maneuver the boxes.



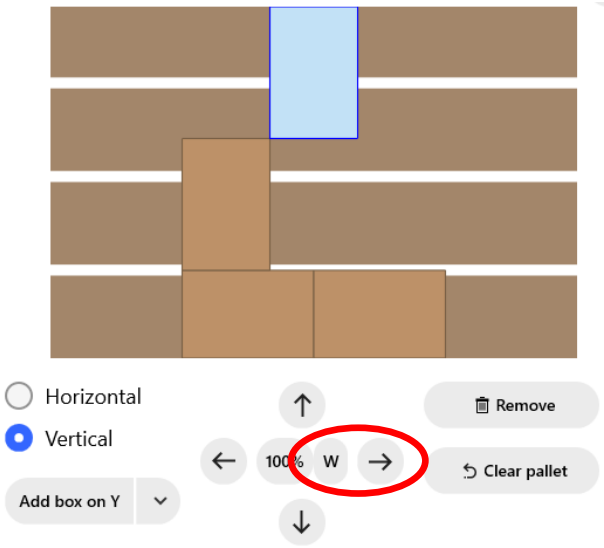
button	Description
100%	Toggle button to set the step size. The set size is defined as a percentage of either the length or the width of the box. Tapping on this button will change the percentage from 100% to 50% to 10% and finally to 1%. Try to avoid using the 1% setting as this may result in gaps between the boxes, meaning that they can’t be handled as one unit of up to 3 individual boxes.
L	Toggle button to select whether the step size is based on the length (L) or width (W) of the box. Tap the button to change between L or W.
↑	Move the selected box the selected percentage * box length or width up.
↓	Move the selected box the selected percentage * box length or width down.
→	Move the selected box the selected percentage * box length or width right.
←	Move the selected box the selected percentage * box length or width left.

For the pattern in the example in this chapter box 4 needs to be moved 100% * the box width to the right and 100% * the box length down.

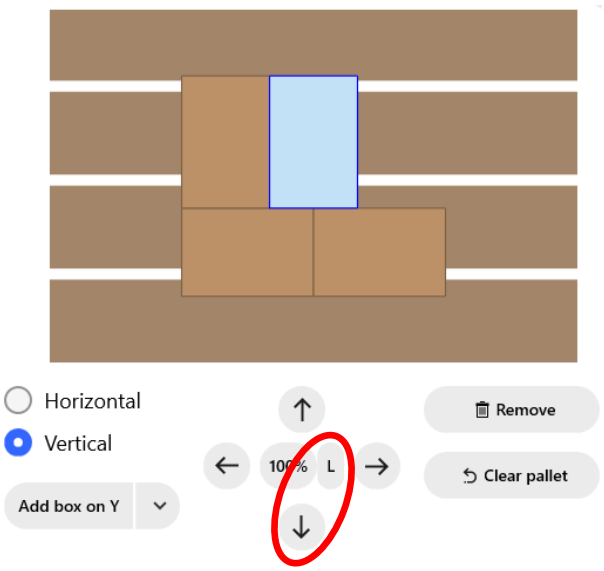
To select the box tap once on the box. The box will be highlighted as shown in the image below.



Now tap L (changes to W) and tab the right button. The box will now move one box width to the right.



The final step is to move the box one box length down. Select W (changes to L) then tap the down arrow



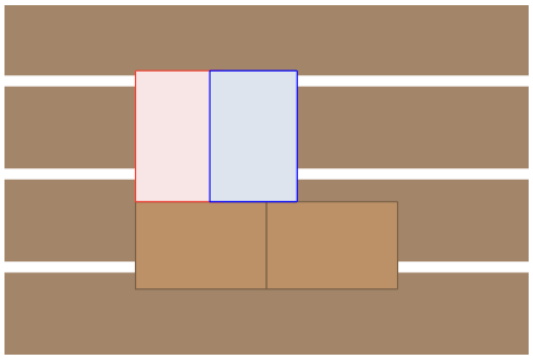
Using this method ensures that boxes are placed against each other without any gaps.

Note

Avoid (small) gaps between boxes, especially for multi pick applications. Gaps will cause Palletizing template not to recognize multiple boxes as one group of boxes.

Boxes can be moved in such way that they can overlap. Overlapping boxes are visualized by coloring these boxes red, as can be seen in the picture below.

While creating the pattern make sure that there are no overlapping boxes. During production boxes will be placed on the pallet as shown in the pattern designer, which could cause collision with boxes already placed on the pallet.



> **Note**
>
> **Avoid overlapping boxes as this will create collisions during palletizing in production.**

On the other hand gaps can be purposely added if this is needed for the stability of the stack.

> **Note**
>
> **Gaps can purposely be created for stability reasons.**

Removing boxes

Boxes can be removed from the pattern, or the entire pattern can be cleared.

To remove one box from the pattern, select the box, so it's highlighted. Press the Remove button to remove the box.

To clear the entire pattern press the Clear pallet button.

Saving the pattern

When the pattern is finished it must be saved using the Save button. The pattern will saved under the HOME:\Palletizing_files\Patterns folder in the flash disk of the robot controller.

Once saved the pattern can be used in a recipe.

External Pattern designer

A separate windows app is also available for creating patterns. With this app patterns are created by entering the formula's. This external app gives more possibilities to define the pattern.

See chapter 7 for more information on the external pattern designer app.

5 RAPID reference

5.1 Instructions

5.1.1 GetFeederItem

Usage

GetFeederItem returns a robtarget containing the pick or place position for the feeder.

Basic example

The following example illustrates the GetFeederItem routine.

Example 1

```
GetFeederItem PickTarget,tGripper;
```

The feeder position based on tooldata tGripper is returned.

Arguments

```
GetFeederItem  Tool [\WObj] [\NrOfItems] [\PickOrient];
```

ToPoint

Data type: robtarget
Returns the feeder pick or place position.

Tool

Data type: tooldata
The persistent variable for the tool used to calculate the feeder target.

[\NrOfItems]

Data type: num
Returns the number of boxes to be picked from the feeder for the current cycle.

[\PickOrient]

Data type: num
Returns the orientation of the pick position. This is a numerical value from 1 to 4 where 1 = 0°, 2 = 90°, 3 = 180° and 4 = 270°.

Program execution

The feeder position of the item is returned. The location of the feeder position is depending on the gripper type installed on the robot.

The feeder for which the pick or place position is to be calculated is set by the SetPalletizeCycle instruction.

5.1.1 GetFeederItem

The \NrOfItems optional argument will return the number of boxes that will be to be picked from the feeder. This value can be used to activate the correct amount of vacuum groups for the current pick action.

The \PickOrient optional argument will return the orientation of gripper, given as a segment. The four segments are 1 = 0°, 2 = 90°, 3 = 180° and 4 = 270°.

Syntax

```
GetFeederItem ‘ ( ‘
    [ ToPoint ' := ' ] <expression (INOUT) of robtarget>
    [ Tool ' := ' ] < persistent (PERS) of tooldata >
    [ '\' NrOfItems ‘ := ‘ <expression (INOUT) of num> ]
    [ '\' PickOrient ‘ := ‘ <expression (INOUT) of num> ] ‘ ) ’
```

Related information

For information about	See
Definition of position	See RAPID reference manual – robtarget data
Definition of tools	See RAPID reference manual – tooldata
Set the palletizing cycle	See SetPalletizeCycle in chapter 5.1.9
Palletizing Template coordinate systems	Calibration of the pallet and feeder position, and the configuration of the tooldata (see chapter 3.1.13.1.5, 3.2.1 and 3.4).

5.1.2 GetPalletItem

Usage

GetPalletItem returns a robtarget containing the pick or place position on the pallet.

Basic example

The following example illustrates the GetPalletItem routine.

Example 1

```
GetPalletItem PlaceTarget,tGripper;
```

The pallet position based on tooldata tGripper is returned.

Arguments

```
GetPalletItem  ToPoint,Tool;
```

ToPoint

Data type: robtarget
Returns the pallet pick or place position.

Tool

Data type: tooldata
The persistent variable for the tool used to calculate the pallet target.

Program execution

The pallet pick / place position for the box is returned.

Syntax

```
GetPalletItem ' ( '
    [ ToPoint ' := ' ] <expression (INOUT) of robtarget>
    [ Tool ' := ' ] < persistent (PERS) of tooldata > ' ; '
```

Related information

For information about	See
Definition of positions	See RAPID reference manual – robtarget data
Definition of tools	See RAPID reference manual – tooldata
Set the palletize cycle	See SetPalletizeCycle in chapter 5.1.9
Palletizing Template coordinate systems	Calibration of the pallet and feeder position, and the configuration of the tooldata (see chapter 3.1.5, 3.1.4 and 3.3).

5.1.3 GetSlipsheetPickItem

Usage

GetSlipsheetPickItem returns a robtarget containing the pick position for a slipsheet.

Basic example

The following example illustrates the GetSlipsheetPickItem routine.

Example 1

```
GetSlipsheetPickItem
SlipSheetPickTarget,SafeApproachHeight,CurrentStackHeight;
```

The slipsheet magazine pick position is returned including the safe approach height and the current height of the slipsheet stack.

Arguments

```
GetSlipsheetPickItem ToPoint,ApproachHeight,SearchStartHeight;
```

ToPoint

Data type: robtarget
Returns the position located at the bottom of the slipsheet stack.

ApproachHeight

Data type: num
Returns the Z offset to be used when the search for a slipsheet needs to be started from the maximum height.

SearchStartHeight

Data type: num
Returns the actual height of the slipsheet stack determined by searching the stack.

Program execution

Slipsheets are typically fed to the robot cell on stacks. GetSlipsheetPickItem returns the position of the slipsheet stack, in the center of the stack at the bottom.

In most cases the height of the slipsheet stack is unknown. So, picking a slipsheet from the stack requires the robot to search for the stack. In most cases this is done by means of the vacuum gripper.

The robot is moved to position at the center of the stack at safe height above the maximum level of the slipsheet stack. Vacuum is switched on and the robot moves down until vacuum is reached, meaning that a slipsheet is found. The

height found at this previous search can be used as start position, to reduce search height and thus cycle time.

ApproachHeight returns the safe height to start a search for a new stack.

SearchStartHeight returns the actual slipsheet stack height i.e. the stack height found at the previous search.

When the SearchStartHeight returns the value -1 the stack height is uncertain, meaning that the search must be started from the safe height (ApproachHeight). The SearchStartHeight will always be set to -1 when the robot program execution is stopped.

Syntax

GetPalletItem ‘ (‘

[ToPoint ‘ := ‘] <expression (INOUT) of robtarget>

[ApproachHeight ‘ := ‘] < persistent (INOUT) of num >’

[SearchStartHeight ‘ := ‘] < persistent (INOUT) of num > ‘ ; ’

Related information

For information about	See
Definition of positions	See RAPID reference manual – robtarget data
Set the slipsheet search height	See SetSlipsheetSearchHeight in chapter 5.1.10

5.1.4 GetSlipsheetPlaceItem

Usage

GetSlipsheetPlaceItem returns a robtarget containing the place position for a slipsheet on the pallet.

Basic example

The following example illustrates the GetSlipsheetPlaceItem routine.

Example 1

```
GetSlipsheetPlaceItem SlipSheetPlaceTarget,obPallet;
```

The slipsheet place position is returned.

Arguments

```
GetSlipsheetPlaceItem ToPoint,ApproachHeight,SearchStartHeight;
```

ToPoint

Data type: robtarget
Returns the position located at the center and on top of the stack.

RotateP1

Data type: swicth
Rotate the gripper 180° for pallet 1.

RotateP2

Data type: swicth
Rotate the gripper 180° for pallet 2.

RotateP3

Data type: swicth
Rotate the gripper 180° for pallet 3.

RotateP4

Data type: swicth
Rotate the gripper 180° for pallet 4.

Program execution

GetSlipsheetPlaceItem returns the place position for a slipsheet on the pallet. This position is located in the center of the pallet on top of the finished layer.

In a number of cases the slip sheet place position may need to be rotated 180° for a particular pallet. The main reason is that the gripper might turn the wrong direction, resulting in stress on the gripper cable. By adding the optional

argument RotatePX the gripper can be rotated 180° for a particular pallet thus reducing the stress on the gripper cable.

Syntax

```
GetPalletItem ‘ ( ‘
    [ ToPoint ‘ := ‘ ] <expression (INOUT) of robtarget>
    [ ‘ \ ‘ RotateP1 ]
    [ ‘ \ ‘ RotateP2 ]
    [ ‘ \ ‘ RotateP3 ]
    [ ‘ \ ‘ RotateP4 ] ‘ ; ’
```

Related information

For information about	See
Definition of positions	See RAPID reference manual – robtarget data

5.1.5 InitAll

Usage

Main initialization.

Basic example

The following example illustrates the InitAll routine.

Example 1

```
InitAll;
```

Initializes Palletizing Template.

Program execution

The instruction must be called at the beginning of the program, from the main procedure.

Syntax

InitAll ' ; '

5.1.6 OpenRecipe

Usage

Opens a Palletizing Template recipe. The OpenRecipe instruction loads the selected recipe file and if present the matching tune data file. When the recipe is loaded the recipe will be initialized

Basic example

The following example illustrates the OpenRecipe routine.

Example 1

```
OpenRecipe RecipeName;
```

Within the Palletizing template framework the predefined variable RecipeName variable contains the name of the recipe selected in the Production screen of the Palletizing template FlexPendant app.

Example 2

```
OpenRecipe \Path:=”HOME:/My recipes”,Product123;
```

The recipe Product123 is loaded from the Home:\Myrecipes folder on the flash drive of the robot controller.

Arguments

```
ReadDataFile [\Path], File;
```

[\Path]

Data type: string
Alternative path where the Palletizing Template data files are stored. If this argument is omitted the default location, HOME:/Palletizing_Files/Recipes is used.

FileName

Data type: string
The name of the Palletizing Template recipe file to be loaded. The file name should be given without extension.

Program execution

The specified recipe and, if present, tune data file are opened.

In the Production screen, at the start of a new recipe, the recipe is chosen from a dropdown menu. This recipe name is stored in the internal Palletizing Template variable RecipeName. In the Palletizing Template framework RecipeName is by default used as FileName argument for the OpenRecipe instruction.

5.1.6 OpenRecipe

The Palletizing Template FP app only creates or modifies Palletizing Template recipe files. The Palletizing Template tune data file is created or modified from the LeanPallize RAPID framework.

When a new recipe is created using the Palletizing Template FP app only a recipe file is created. Since the tune data file doesn't exists at this time the tune data should be set to initial values and a new tune data file can be created.

Error handling

The following recoverable errors can be generated. The errors can be handled in an ERROR handler. The system variable ERRNO will be set to:

Name	Cause of error
ERR_FILEOPEN	A file cannot be opened.
ERR_FILEACC	An error occurs during reading.
ERR_MISSING_ARGUMENT	At least one of the two arguments \Recipe or \Tune must be specified.

Syntax

```
OpenRecipe
    [ '\' Path ' := ' <expression (IN) of string>] ‘ , ‘
    [ FileName ' := ' ] <expression (IN) of string> ‘ ; ’
```

5.1.7 PlaceCycleDone

Usage

Reports the current palletizing cycle as completed.

Basic example

The following example illustrates the PlaceCycleDone routine.

Example 1

```
PlaceCycleDone TRUE;
```

Reports the current palletizing cycle as completed.

Arguments

```
PlaceCycleDone BoxPlaced [\BoxLost];
```

BoxPlaced

Data type: bool
Status indication of the place cycle. When set to TRUE the palletizing cycle has finished successfully. When set to FALSE the cycle has finished unsuccessfully.

[\BoxLost]

Data type: switch
Box lost status indication. The box will be shown in red in the stacking progress indicator.

Program execution

PlaceCycleDone should be called and the end of the palletizing cycle, to indicate that the cycle is finished. By setting the BoxPlaced argument to FALSE, the box will be considered as not placed. The same cycle will be repeated. Setting the BoxPlaced flag to TRUE the cycle is considered to be finished successfully.

Adding the \BoxLost optional argument will color the box red in the Stacking progress indicator in the production screen. The BoxPlaced argument will not be considered.

Syntax

```
PlaceCycleDone
    [ BoxPlaced ' := ' ] <expression (IN) of bool>
    [ '\' BoxLost ] ' ; '
```

5.1.7 PlaceCycleDone

Related information

For information about	See
Set the palletizing cycle	SetPalletizeCycle in chapter 5.1.9.

5.1.8 ProductionStatusCheck

Usage

Checks if the robot has finished the cycle or the pallet. This is used when selecting the corresponding cycle stop mode. See chapter 4.2.1.2 for more information.

Basic example

The following example illustrates the ProductionStatusCheck routine.

Example 1

```
ProductionStatusCheck;
```

Checks the production stop action.

Program execution

The instruction must be called after every pick – place cycle to check if the condition for the alternative stop mode, Stop after placing current box or Stop when the pallet is finished, is met. When the condition for the production stop is met a predefined routine is called.

In these routine RAPID code can be added when needed.

Stop mode	Handler
Stop after placing current box	Once the box is placed on the pallet the routine StopEndOfCycle is called. By default the robot program is stopped.
Stop when the pallet is finished	Once the pallet is full the routine StopEndOfPallet is called. By default the robot program is stopped.

Syntax

```
ProductionStatusCheck ' ; '
```

5.1.9 SetPalletizeCycle

Usage

Set the source and destination for the next palletizing cycle.

Basic example

The following example illustrates the SetPalletizeCycle routine.

Example 1

```
SetPalletizeCycle 1,1;
```

For the next palletizing cycle the robot will pick a box from feeder 1 and place it on pallet 1.

Arguments

```
SetPalletizeCycle FeederNr, PalletNr [\OptimizeTopLayer] [\StartLayer];
```

PalletNr

Data type: num
Number of the pallet on which the robot should pick or place a box in the next cycle.

FeederNr

Data type: num
Number of the feeder from which the robot should pick or place a box for the next cycle.

[\OptimizeTopLayer]

Data type: switch
In order to optimize the height of the pallet the \OptimizeTopLayer argument can be added. When this argument is added the robot will perform a low approach for all the cycles on the top layer.

Program execution

To create a collision free palletizing cycle the source (feeder) and destination (pallet) need to set for the next palletizing cycle. Based on this information Palletizing Template can determine the approach and depart heights above the feeder and the pallet.

The instruction must be executed before each actual palletizing cycle is started.

The optional argument \OptimizeTopLayer can be used to improve the total pallet height that can be stacked by the robot. During normal palletizing the robot moves with the box

over the layer it's actually forming. When a high stack is to be built with a relatively small robot one layer is lost due to this fact. With the optimization of the top layer the robot will rotate the box in the correct orientation while moving upwards from the feeder. The box can it can be placed on the pallet moving at a low height over the previous layer, thus gaining an additional layer on the pallet. OptimizeTopLayer uses the offset OPT_TOP_SAFE_X and OPT_TOP_SAFE_Y to define clearance when approaching the pallet (see chapter 5.4.23 and 5.4.24).

Both palletizing and depalletizing are supported

Error handling

The following recoverable errors can be generated. The errors can be handled in an ERROR handler. The system variable ERRNO will be set to:

Name	Cause of error
ERR_INVALID_FEEDER	The feeder number is incorrect. The value must be between 1 and the supported number of feeders.
ERR_INVALID_PALLET	The pallet number is incorrect. The value must be between 1 and the supported number of pallets.

Syntax

```
SetPalletizeCycle
    [ FeederNr ' := ' ] <expression (IN) of num> ' , '
    [ PalletNr ' := ' ] <expression (IN) of num>
    [ '\' OptimizeTopLayer ] ' ; '
```

5.1.10 SetSlipsheetSearchHeight

Usage

Sets the height of the slipsheet stack.

Basic example

The following example illustrates the SetSlipsheetSearchHeight routine.

Example 1

```
SetSlipsheetSearchHeight pFound.trans.z;
```

Set the height of the slipsheet stack to the z coordinate of the last search action.

Arguments

```
SetSlipsheetSearchHeight SlipsheetPlaced;
```

SearchHeight

Data type: num

Sets the current height of the slipsheet stack.

Program execution

When handling slipsheets the robot will typically search the slipsheet stack to find the top slipsheet on the stack.

For the first cycle after the robot program was stopped, at this stage an operator could have entered the cell a add slipsheets to the stack, the robot needs to start searching from a sufficient safe height in order the prevent collisions. Once the first slipsheet is found and picked the robot can use the actual height of the slipsheet stack to start searching for the next time is slipsheet is to be placed.

Searching the stack from a safe height will typically take a longer time. Searching from a known slipsheet stack height will reduce this time significantly.

After stopping the robot program execution the slipsheet stack height will be set to the safe height, set in the Palletzing Template app (see chapter 4.1.3)

Syntax

```
SetSlipsheetSearchHeight

    [ SearchHeight ' := ' ] <expression (IN) of num> ‘ ; ‘
```

5.1.10 SetSlipsheetSearchHeight

Related information

For information about	See
Get the slipsheet pick position	See GetSlipsheetPickItem in chapter 5.1.3

5.1.11 SlipsheetCycleDone

Usage

Reports the current slipsheet pick – place cycle as completed.

Basic example

The following example illustrates the SlipsheetCycleDone routine.

Example 1

```
SlipsheetCycleDone TRUE;
```

Reports the current slipsheet pick - place cycle as completed.

Arguments

```
SlipsheetCycleDone SlipsheetPlaced;
```

SlipsheetPlaced

Data type: bool
Status indication of the slipsheet pick - place cycle. When set to TRUE the slipsheet is successfully placed. When set to FALSE the slipsheet is not placed (lost). This will result in the cycle being repeated.

Program execution

SlipsheetCycleDone should be called and the end of the slipsheet pick - place cycle, to indicate that the cycle is finished. By setting the SlipsheetPlaced argument to FALSE, the slipsheet will be considered as not placed. The same cycle will be repeated. Setting the SlipsheetPlaced flag to TRUE the cycle is considered to be finished successfully.

Syntax

```
SlipsheetCycleDone
    [ SlipsheetPlaced ' := ' ] <expression (IN) of bool> ' ; '
```

5.1.12 UpdateMotionParams

Usage

Activates the customer defined motion settings.

Basic example

The following example illustrates the UpdateMotionParams routine.

Example 1

```
UpdateMotionParams;
```

Updates the motion parameters set in the Palletizing template app.

Program execution

UpdateMotionParams activates the motion parameters like, speed, acceleration and pick, place wait times. See chapter 4.3.2 for more information on the motion parameters.

Syntax

```
UpdateMotionParams ‘,‘
```

5.2 Functions

5.2.1 GetActivePallet

Usage

GetActivePallet returns the pallet where the robot is currently building the stack.

Basic example

The following example illustrates the GetActivePallet routine.

Example 1

```
SetPalletizeCycle 1,GetActivePallet();
```

Sets the source and destination for the next palletizing cycle. The robot will pick the boxes from feeder 1 and will place these on the active pallet.

Return value

Data type: num
The active pallet position.

Program execution

GetActivePallet return the number of the active pallet. The active pallet is the pallet where the robot is currently building the stack.

Syntax

```
GetActivePallet '('')'
```

Related information

For information about	See
Set the palletizing cycle	See SetPalletizeCycle in chapter 5.1.9

5.2.2 GetFeederApproach

Usage

GetFeederApproach returns a feeder approach position.

Basic example

The following example illustrates the GetFeederApproach routine.

Example 1

```
pApproach:= GetFeederApproach (pPick);
```

Get the feeder approach position based on pick position pPick.

Return value

Data type: robtarget
The feeder approach position.

Arguments

```
GetFeederApproach ( ToPoint );
```

ToPoint

Data type: robtarget
The position on which the feeder approach position will be based.

Program execution

Before the robot can move to the feeder position it must first move to an approach position, above the feeder position, in order to obtain a collision free path to the feeder. The approach position is usually placed directly above the feeder position.

Offsets will be used to calculate the feeder approach position. These offsets are specified in the variables FEEDER_APPROACH_X GetFeederApproach , FEEDER_APPROACH_Y GetFeederApproach and FEEDER_APPROACH_Z GetFeederApproach . These offsets are declared in the Settings system module and can be modified by the user.

The Z offset however will be adapted to the height of the last pallet depart position. When the pallet depart position is higher than the feeder approach position, the feeder approach position will be brought to the same level as the pallet depart position.

The orientation of the feeder approach position is the same as in the feeder position.

5.2.2 GetFeederApproach

The feeder for which the approach position is calculated is set by the SetPalletizeCycle instruction.

The above mentioned rules are identical when the system is set to depalletize mode.

Syntax

```
GetFeederApproach ' ( '
    [ ToPoint ' := ' ] < expression (IN) of robtarget > ' ) '
```

Related information

For information about	See
Definition of positions	See RAPID reference manual – robtarget data
Get the feeder position	See GetFeederItem in chapter 5.1.1

5.2.3 GetFeederDepart

Usage

GetFeederDepart returns a feeder depart position.

Basic example

The following example illustrates the GetFeederDepart routine.

Example 1

```
pDepart:= GetFeederDepart (pPick);
```

Get the feeder depart position based on pick position pPick.

Return value

Data type: robtarget
The feeder depart position.

Arguments

```
GetFeederDepart ( ToPoint );
```

ToPoint

Data type: robtarget
The position on which the feeder depart position will be based.

Program execution

Before the robot can move to the pallet position it must first move to a feeder depart position, which is typically located sufficiently high above the feeder position.

Offsets will be used to calculate the feeder depart position. These offsets are specified in the variables FEEDER_DEPART_X, FEEDER_DEPART_Y and FEEDER_DEPART_Z. The height of the box is automatically added to the FEEDER_DEPART_Z offset to ensure that when in the depart position new boxes can be fed with colliding against the box held by the robot.

These offsets are declared in the Settings module and can be modified by the user.

The Z offset however will be adapted to the height of the pallet approach position, when the pallet approach position is higher than the feeder depart position, the feeder depart position will be brought to the same level as the pallet approach position. In this way the robot moves at the same height over the pallet to the pallet approach position.

The orientation in the feeder depart position is the same as in the feeder position. However if the OptimizeTopLayer optional argument is used in the SetPalletizeCycle instruction the orientation in the feeder depart position will be the same as in the pallet position. This implicates that the box is reoriented while moving up from the feeder position to the feeder depart position.

The feeder for which the depart position is calculated is set by the SetPalletizeCycle instruction.

Syntax

```
GetFeederDepart ' ( '
    [ ToPoint ' := ' ] < expression (IN) of robtarget > ' ) '
```

Related information

For information about	See
Definition of positions	See RAPID reference manual – robtarget data
Get the feeder position	See GetFeederItem on page XXX

5.2.4 GetNextPickAmount

Usage

GetNextPickAmount returns the number of item to pick for the next cycle.

Basic example

The following example illustrates the GetNextPickAmount routine.

Example 1

```
NextBoxesToPick:=GetNextPickAmount (1);
```

The amount of boxes to pick for the next palletize cycle on pallet 1 is returned.

Return value

Data type: num
The amount of items to pick for the next cycle.

Arguments

```
GetNextPickAmount ( PalletNr [\StartLayer]);
```

PalletNr

Data type: num
The variable representing the pallet number for the amount of items to pick for the next cycle.

Program execution

The amount of boxes for the next cycle is returned. This is typically used when the feeder system needs to know in advance how many items have to be presented for the next cycle. These data are typically communicated to the feeder controller (PLC) just after the items for the current cycle have been picked.

The amount of items required for the current cycle can be retrieved from the GetFeederItem function.

Syntax

```
GetNextPickAmount ' ( '
    [ PalletNr ' := ' ] < expression (IN) of num ' ) '
```

5.2.5 GetPalletApproach

Usage

GetPalletApproach returns the pallet approach position.

Basic example

The following example illustrates the GetPalletApproach routine.

Example 1

```
pApproach:= GetPalletApproach (pPlace);
```

Get the pallet approach position based on the pallet position pPlace.

Return value

Data type: robtarget
The pallet approach position.

Arguments

```
GetPalletApproach ( ToPoint [\LowApproach] [\SlipSheet]);
```

ToPoint

Data type: robtarget
The position on which the pallet approach position will be based.

[\LowApproach]

Data type switch
The position will be generated with a low height above the place position.

[\SlipSheet]

Data type switch
When present the pallet approach position for a slip sheet is calculated.

Program execution

Before the robot can move to the pallet position it must first move to an approach position, located above the current layer, in order to obtain a collision free path.

Offsets will be used to calculate the pallet approach position. These offsets are specified in the variables PALLET_APPROACH_X, PALLET_APPROACH_Y and PALLET_APPROACH_Z. These offsets are declared in the Settings module and can be modified by the user.

The Z offset however will be adapted to the height of the feeder depart when this feeder depart position is higher than the pallet approach position. In this case the pallet approach position will be brought to the same level as the feeder depart position.

5.2.5 GetPalletApproach

The pallet for which the approach position is calculated is set by the SetPalletizeCycle instruction.

If the \LowApproach optional argument is present the pallet approach position will not be placed above the current layer, but above the previous layer. This offset is specified in the variable LOW_APPROACH_OFFS, which is declared in the Settings module and can be modified by the user.

If the \SlipSheet argument is present only the approach height will be calculated. The PALLET_APPROACH_X and PALLET_APPROACH_Y offsets will not be used.

The orientation in the pallet approach position is the same as in the pallet position.

Syntax

```
GetPalletApproach ' ( '
    [ ToPoint ' := ' ] < expression (IN) of robtarget >
    [ '\' LowApproach]
    [ '\' SlipSheet ] ' ) '
```

Related information

For information about	See
Definition of positions	See RAPID reference manual – robtarget data
Set the palletizing cycle	See SetPalletizeCycle on page XXX
Get the pallet position	See GetPalletItem on page XXX

5.2.6 GetPalletDepart

Usage

GetPalletDepart returns a pallet depart position.

Basic example

The following example illustrates the GetPalletDepart routine.

Example 1

```
pDepart:= GetPalletDepart (pPlace);
```

Get the pallet depart position based on place position pPlace.

Return value

Data type: robtarget
The pallet depart position.

Arguments

```
GetPalletDepart ( ToPoint);
```

ToPoint

Data type: robtarget
The position on which the pallet depart position will be based.

Program execution

In order to obtain a collision free path when moving to the feeder the robot must first move to a pallet depart position. The pallet depart position is usually located directly above the current layer.

Offsets will be used to calculate the pallet depart position. These offsets are specified in the variables PALLET_DEPART_X, PALLET_DEPART_Y and PALLET_DEPART_Z. These offsets are declared in the Settings module and can be modified by the user.

The Z offset however will be adapted to the height of the feeder approach position. When the feeder approach position is higher than the pallet depart position, the pallet depart position will be brought to the same level as the feeder approach position. In this way the robot moves at a constant height over the pallet to the feeder approach position.

The orientation in the pallet depart position is the same as in the pallet position.

The pallet for which the depart position is calculated is set by the SetPalletizeCycle instruction.

Syntax

```
GetPalletDepart ' ( '
    [ ToPoint ' := ' ] < expression (IN) of robtarget > ' ) '
```

Related information

For information about	See
Definition of positions	See RAPID reference manual – robtarget data
Set the palletizing cycle	See SetPalletizeCycle on page XXX
Get the pallet position	See GetPalletItem on page XXX

5.2.7 GetSlipSheet

Usage

GetSlipSheet is used to indicate if a slip sheet needs to be placed on the pallet before the next palletizing cycle.

Basic example

The following example illustrates the GetSlipSheet routine.

Example 1

```
IF GetSlipsheet()THEN
    PickSlipSheet;
    PlaceSlipSheet SlipSheetPos;
ENDIF
```

If a slip sheet is required, i.e. GetSlipSheet returns TRUE, the slip sheet will be picked from a slip sheet feeder and is placed on the pallet using the position returned through the variable SlipSheetPos.

Return value

Data type: bool
Returns TRUE if a slipsheet is required for the current finished layer.

Program execution

The function returns TRUE if a slip sheet is required for the current finished layer.

Syntax

```
GetSlipSheet '(' ')'
```

5.2.8 GetSlipsheetApproach

Usage

GetSlipsheetApproach returns the slipsheet magazine approach position.

Basic example

The following example illustrates the GetSlipsheetApproach routine.

Example 1

```
pApproach:= GetSlipsheetApproach (pSlipsheetPick);
```

Get the slipsheet magazine approach position based on the slipsheet pick position pSlipsheetPick.

Return value

Data type: robtarget
The slipsheet magazine approach position.

Arguments

GetPalletApproach (ToPoint [\LowApproach] [\SlipSheet]);

ToPoint

Data type: robtarget
The position on which the slipsheet magazine approach position will be based.

Program execution

Before the robot can search for slipsheet and can pick the slipsheet from the slipsheet stack it first needs to approach the slipsheet stack at a safe distance - height. The GetSlipsheetApproach routine returns the approach position based on the actual slipsheet stack height and the SLIPSHEET _APPROACH_Z setting. Once the slipsheet approach position is reached the robot can to search for a slipsheet.

Offsets will be used to calculate the slipsheet approach position. These offsets are specified in the variables SLIPSHEET_APPROACH_X, SLIPSHEET _APPROACH_Y and SLIPSHEET _APPROACH_Z. These offsets are declared in the Settings module and can be modified by the user.

5.2.8 GetSlipsheetApproach

Syntax

```
GetSlipsheetApproach ‘ ( ‘
    [ ToPoint ‘ := ‘ ] < expression (IN) of robtarget > ‘ ) ‘
```

Related information

For information about	See
Get the slipsheet pick position	See GetSlipsheetPickItem on page XXX
Definition of positions	See RAPID reference manual – robtarget data

5.2.9 GetSlipsheetDepart

Usage

GetSlipsheetDepart returns a slipsheet magazine depart position.

Basic example

The following example illustrates the GetSlipsheetDepart routine.

Example 1

```
pDepart:= GetSlipsheetDepart (pSlipsheetPick);
```

Get the slipsheet magazine depart position based on pick position pSlipsheetPick.

Return value

Data type: robtarget
The slipsheet magazine depart position.

Arguments

GetSlipsheetDepart (ToPoint);

ToPoint

Data type: robtarget
The position on which the slipsheet magazine depart position will be based.

Program execution

In order to obtain a collision free path when moving away from the slipsheet magazine the robot must first move to a slipsheet magazine depart position

Offsets will be used to calculate the slipsheet magazine depart position. These offsets are specified in the variables SLIPSHEET _DEPART_X, SLIPSHEET _DEPART_Y and SLIPSHEET _DEPART_Z. These offsets are declared in the Settings module and can be modified by the user.

Syntax

```
GetPalletDepart ' ( '
    [ ToPoint ' := ' ] < expression (IN) of robtarget > ' ) '
```

5.2.9 GetSlipsheetDepart

Related information

For information about	See
Get the slipsheet pick position	See GetSlipsheetPickItem on page XXX
Definition of positions	See RAPID reference manual – robtarget data

5.2.10 LiftKitCorrect

Usage

LiftKitCorrect returns the corrected pick – place position when a lift kit is used.

Basic example

The following example illustrates the LiftKitCorrect routine.

Example 1

```
pApproach:= LiftKitCorrect (pPlace,obPallet);
```

Get the place that is corrected for the movement of a lift kit.

Return value

Data type: robtarget
The liftkit corrected position.

Arguments

LiftKitCorrect (ToPoint,WObj);

ToPoint

Data type: robtarget
The position that will be corrected for the lift kit movement.

WObj

Data type wobjdata
The workobject used for the target.

Program execution

LiftKitCorrect returns a robtarget that is corrected for the movement of the robot using a lift kit.

A lift kit is typically used to increase the stack height of a collaborative robot. As a lift kit typically has an external controller the movement of the lift kit is not automatically included in the robtarget by the robot controller. As the bas frame of the robot is thus also moved by the lift kit, the pick and place targets need to be corrected for, which is done by the LiftKitCorrect instruction.

5.2.10 LiftKitCorrect

Syntax

LiftKitCorrect ‘ (‘

[ToPoint ‘ := ‘] < expression (IN) of robtarget >

[WObj ‘ := ‘] < expression (PERS) of wobjdata > ‘) ‘

Related information

For information about	See
Definition of positions	See RAPID reference manual – robtarget data
Definition of tools	See RAPID reference manual – tooldata
Get the feeder position	See GetFeederItem on page XXX
Get the pallet position	See GetPalletItem on page XXX

5.3.1 boxdata

5.3 Data types

5.3.1 boxdata

Usage

The boxdata is an internal Palletizing Template data type for box data.

Description

Data type used to store information about the boxes to be handled for the current recipe.

Components

The data type boxdata has the following components:

Length

Data type: num
Predefined length of the box.

Width

Data type: num
Predefined width of the box.

Height

Data type: num
Predefined height of the box.

CorrLength

Data type: num
The corrected length of box. Result of the predefined box length with the box length tune value added.

CorrWidth

Data type: num
The corrected width of box. Result of the predefined box width with the box width tune value added.

CorrHeight

Data type: num
The corrected heigth of box. Result of the predefined box heigth with the box heigth tune value added.

Weight

Data type: num
Weight of the box.

5.3.1 boxdata

LabelPos0

Data type: bool
Label present on the first side of the box.

LabelPos1

Data type: bool
Label present on the second side of the box.

LabelPos2

Data type: bool
Label present on the third side of the box.

LabelPos3

Data type: bool
Label present on the fourth side of the box.

LongSideLead

Data type: num
maximum amount of boxes in long side lead configuration.

ShortSideLead

Data type: num
maximum amount of boxes in long short lead configuration.

Structure

< dataobject of layerrefdata >

< Length of num >

< Width of num >

< Heigth of num >

< CorrLength of num >

< CorrWidth of num >

< CorrHeigth of num >

< Weight of num >

< LabelPos0 of bool >

< LabelPos1 of bool >

< LabelPos2 of bool >

5.3.1 boxdata

< LabelPos3 of bool >

< LondSideLead of num >

< ShortSideLead of num >

5.3.2 childdata

Usage

The childdata is an internal Palletizing Template data type used for storing combinations of boxes handled in each cycle (multi pick).

Description

Data type used to store the combination of boxes that are handled in each cycle. Palletizing Template can handle up to 5 boxes in each cycle.

Components

The data type childdata has the following components:

Child1

Data type: num
Reference to the first child.

Child2

Data type: num
Reference to the second child.

Child3

Data type: num
Reference to the third child.

Child4

Data type: num
Reference to the fourth child.

Child5

Data type: num
Reference to the fifth child.

Structure

```
< dataobject of childdata >
    < Child1 of num >
    < Child2 of num >
    < Child3 of num >
    < Child4 of num >
    < Child5 of num >
```

5.3.3 fileparameterdata

Usage

The fileparameterdata is an internal Palletizing Template data type used for the conversion of file objects.

Description

Data type used for the conversion of the Palletizing Template file objects from the recipe or tune data files. The individual objects in the data files consist of a variable name and value separated by a semicolon. The variable name is store in the Name element. The value is stored on the element matching the type of variable (string, num or bool).

Components

The data type fileparameterdata has the following components:

Param

Data type: string
The variable object.

Name

Data type: string
Name of the variable.

ValueStr

Data type: string
Value of the variable for string type.

ValueNum

Data type: num
Value of the variable for num type.

ValueDnum

Data type: dnum
Value of the variable for dnum type.

ValueBool

Data type: bool
Value of the variable for bool type.

5.3.3 fileparameterdata

Structure

< dataobject of fileparameterdata >

< Param of string >

< Name of string >

< ValueStr of string >

< ValueNum of num >

< ValueDnum of dnum >

< ValueBool of bool >

5.3.4 formuladata

Usage

The formuladata is an internal Palletizing Template data type used for storing the pattern formula and its individual components.

Description

Data type used for the conversion of the Palletizing Template pattern formulas to the internal components. The pattern formulas are stored string based and contain an orientation, a formula for the X coordinate, a formula for the Y coordinate and a grouping number. After the conversion of the pattern formula the data are stored in the respective variables of the proper data type.

Components

The data type formuladata has the following components:

Formula

Data type: string
The pattern formula as read from the recipe file.

BoxOrient

Data type: string
Orientation of the box (either H or V).

XCalcL

Data type: num
Amount of box length offsets for the X coordinate.

XCalcW

Data type: num
Amount of box width offsets for the X coordinate.

YCalcL

Data type: num
Amount of box length offsets for the Y coordinate.

YCalcW

Data type: num
Amount of box width offsets for the Y coordinate.

Group

Data type: num
Number of the group of which the box is part.

5.3.4 formuladata

Structure

< dataobject of childdata >

< Formula of string >

< BoxOrient of string >

< XCalcL of num >

< XCalcW of num >

< YCalcL of num >

< YCalcW of num >

< Group of num >

5.3.5 layerrefdata

Usage

The layerrefdata is an internal Palletizing Template data type with layer reference data.

Description

Data type used to store layer refence information on the pallet layer.

Components

The data type layerrefdata has the following components:

Name

Data type: string
The name of the layer.

PatternRef

Data type: num
Reference to the patterndata used for this layer.

NrOfBoxes

Data type: num
Total amount of boxes on the layer.

NrOfCycles

Data type: num
Number of cycles required to build the layer.

SlipSheet

Data type: bool
Set to TRUE when a slip sheet is to be placed before boxes are placed.

Structure

```
< dataobject of layerrefdata >
    < Name of string >
    < PatternRef of num >
    < NrOfBoxes of num >
    < NrOfCycles of num >
    < SlipSheet of bool >
```

5.3.6 paldata

Usage

The paldata is an internal Palletizing Template data type for pallet data.

Description

Data type used to store information about the pallet in the current recipe.

Components

The data type paldata has the following components:

Length

Data type: num
Predefined length of the pallet.

Width

Data type: num
Predefined width of the pallet.

Height

Data type: num
Predefined height of the pallet.

NrOfLayers

Data type: num
The total number of layers on the pallet.

MaxHeight

Data type: num
Maximum total height of the pallet.

Structure

```
< dataobject of paldata >
    < Length of num >
    < Width of num >
    < Heigth of num >
    < NrOfLayers of num >
    < MaxHeight of num >
```

5.3.7 patterndata

Usage

The patterndata is an internal Palletizing Template data type used for storing the individual pattern elements.

Description

Stores all the information required to perform 1 cycle.

Components

The data type patterndata has the following components:

Formula

Data type: formuladata
Formula in basic format and split in individual components

Position

Data type: pos
Pallet position.

ApproachDir

Data type: num
Approach direction

VacuumGroup

Data type: num
Vacuum groups required to pick the boxes

PickOrient

Data type: num
Pick orientation. Given in as segments of 90°.

PlaceOrient

Data type: num
Place orientation. Given in as segments of 90°.

LabelOrient

Data type: num
Label orientation.

ID

Data type: num
Identification code of the box.

5.3.7 patterndata

Parent

Data type: num
Identification code of the parent of the box. If the box is the parent this variable will be set to -1.

LinkedType

Data type: num
Long side lead or short side lead.

Child

Data type: childdata
List of children for this parent.

BoxCount

Data type: num
Amount of boxes for the cycle. Only applicable when the box s the parent.

Structure

< dataobject of patterndata >

< Formula of formuladata >

< Position of num >

< ApproachDir of num >

< VacuumGroup of num >

< PickOrient of num >

< PickOrient of num >

< PlaceOrient of num >

< LabelOrient of num >

< ID of num >

< Parent of num >

< LinkedType of num >

< Child of childdata >

< BoxCount of num >

5.3.8 sizedata

Usage

The sizedata is a common Palletizing Template data type used to describe a size.

Description

Data type used to store size and offset information for any item.

Components

The data type sizedata has the following components:

Length

Data type: num
Predefined length of the object.

Width

Data type: num
Predefined width of the object.

XOffset

Data type: num
XOffset for the object.

YOffset

Data type: num
YOffset for the object.

Structure

```
< dataobject of layerrefdata >
    < Length of num >
    < Width of num >
    < XOffset of num >
    < YOffset of num >
```

5.3.9 slipsheetdata

Usage

The slipsheetdata is an internal Palletizing Template data type with slip sheet data.

Description

Data type used to store refence information for slip sheets per layer.

Components

The data type slipsheetdata has the following components:

Used

Data type: bool
If TRUE the layer has a slip sheet.

Position

Data type: pos
Position of the slip sheet on the layer.

Structure

```
< dataobject of slipsheetdata >
    < Used of bool >
    < Position of pos >
```

5.3.10 slipsheetsizedata

Usage

The slipsheetsizedata is a internal Palletizing Template data type used to describe the size of a slipsheet.

Description

Data type used to store size information for slipsheets.

Components

The data type slipsheetsizedata has the following components:

Length

Data type: num
Predefined length of the object.

Width

Data type: num
Predefined width of the object.

Thickness

Data type: num
Thickness of the slipsheet.

StackHeight

Data type: num
Maximum slipsheet stackheight.

SearchHeight

Data type: num
Actual slipsheet stack height.

Structure

```
< dataobject of layerrefdata >
    < Length of num >
    < Width of num >
    < Thickness of num >
    < StackHeight of num >
    < SearchHeight of num >
```

5.4 Predefined Palletizing Template data

Description

Palletizing Template uses a number of predefined data that can be configured by the user. Several of these data are also described in chapter 3. The internal data are not described.

As most of the described variables are used internally by Palletizing Template the variables must not be renamed or moved to a different module.

Note

Do not rename or move predefined Palletizing Template data.

5.4.1 Acceleration

Usage

Acceleration defines the acceleration the robot uses in the palletizing cycle.

Description

Acceleration is used to reduce the robots acceleration when moving with boxes. Reducing the acceleration can prevent the robot from losing boxes. A decrease in acceleration will result in an increase of cycle time. The acceleration can be set / changed from the tune tab of the Motion tuning tab in the Palletizing Template FlexPendant app.

The acceleration may require reduction when large and / or heavy boxes are handled.

Acceleration is of type num and declared in the module appdata.

5.4.2 ActLayer

Usage

ActLayer contains the actual layer the robot is currently forming.

Description

ActLayer can be used to read the actual layer number the robot is forming on the pallet

ActLayer is of type num and declared in the module appdata.

Note

This variable is intended to be read only. Do not change this variable.

5.4.3 BoxLength

Usage

BoxLength contains the length of the box used in the current recipe

Description

BoxLength can be used to read the length of the box in mm.

BoxLength is of type num and declared in the module appdata.

Note

This variable is intended to be read only. Do not change this variable.

5.4.4 BoxHeigth

Usage

BoxHeight contains the height of the box used in the current recipe

Description

BoxHeight can be used to read the height of the box in mm.

BoxHeight is of type num and declared in the module appdata.

Note

This variable is intended to be read only. Do not change this variable.

5.4.5 BoxesTotal

Usage

BoxesTotal contains the total amount of boxes for the current pallet stack.

Description

BoxesTotal can be used to read the total amount of boxes of the pallet stack. Once the robot has reached this amount of boxes the pallet is full.

BoxesTotal is of type num and declared in the module appdata.

Note

This variable is intended to be read only. Do not change this variable.

5.4.6 BoxesOnPallet

Usage

BoxesOnPallet contains the number of boxes the robot has already place on the pallet stack.

Description

BoxesOnPallet can be used to get the number of boxes the robot has already place on the pallet stack.

BoxesOnPallet is of type num and declared in the module appdata.

Note

This variable is intended to be read only. Do not change this variable.

5.4.7 BoxWidth

Usage

BoxWidth contains the width of the box used in the current recipe

Description

BoxWidth can be used to read the width of the box in mm.

BoxWidth is of type num and declared in the module appdata.

Note

This variable is intended to be read only. Do not change this variable.

5.4.8 CORNER_LOWER_LAYERS

Usage

CORNER_LOWER_LAYERS defines the start corner for the lower layers.

Description

CORNER_LOWER_LAYERS define the start corner from which the robot will start the build the layer. The lower layers are the pallet layers that are situated below the feeder level. See chapter 3.1.2 for more information on the start corner.

CORNER_LOWER_LAYERS is an array of type num and declared in the module Settings.

5.4.9 CORNER_UPPER_LAYERS

Usage

CORNER_UPPER_LAYERS defines the start corner for the lower layers.

Description

CORNER_UPPER_LAYERS define the start corner from which the robot will start the build the layer. The upper layers are the pallet layers that are situated above the feeder level. See chapter 3.1.2. for more information on the start corner.

CORNER_UPPER_LAYERS is an array of type num and declared in the module Settings.

5.4.10 FEEDER_APPROACH_X

Usage

FEEDER_APPROACH_X is used as offset for the feeder approach position.

Description

In order to move collision free to the feeder the robot moves by an approach position. The approach position is defined as an offset from the feeder position. This offset is executed according to the feeder workobject.

FEEDER_APPROACH_X is of type num and declared in the module Settings.

5.4.11 FEEDER_APPROACH_Y

Usage

FEEDER_APPROACH_Y is used as offset for the feeder approach position.

Description

In order to move collision free to the feeder the robot moves by an approach position. The approach position is defined as an offset from the feeder position. This offset is executed according to the feeder workobject.

FEEDER_APPROACH_Y is of type num and declared in the module Settings.

5.4.12 FEEDER_APPROACH_Z

Usage

FEEDER_APPROACH_Z is used as offset for the feeder approach position.

Description

In order to move collision free to the feeder the robot moves by an approach position. The approach position is defined as an offset from the feeder position. This offset is executed according to the feeder workobject.

FEEDER_APPROACH_Z defines minimum height at which the robot can approach the feeder. A positive FEEDER_APPROACH_Z results in a higher approach position.

In order to achieve a collision free cycle the FEEDER_APPROACH_Z will be overruled by the pallet height. If the pallet is higher than the feeder approach position the actual pallet height will be used.

FEEDER_APPROACH_Z is of type num and declared in the module Settings.

5.4.13 FEEDER_DEPART_X

Usage

FEEDER_DEPART_X is used as offset for the feeder depart position.

Description

In order to move collision free from the feeder the robot moves by a depart position. The depart position is defined as an offset from the feeder position. This offset is executed according to the feeder workobject.

In most cases this offset will be 0 as the robot should move straight up from the feeder position.

FEEDER_DEPART_X is of type num and declared in the module Settings.

5.4.14 FEEDER_DEPART_Y

Usage

FEEDER_DEPART_Y is used as offset for the feeder depart position.

Description

In order to move collision free from the feeder the robot moves by a depart position. The depart position is defined as an offset from the feeder position. This offset is executed according to the feeder workobject.

In most cases this offset will be 0 as the robot should move straight up from the feeder position.

FEEDER_DEPART_Y is of type num and declared in the module Settings.

5.4.15 FEEDER_DEPART_Z

Usage

FEEDER_DEPART_Z is used as offset for the feeder depart position.

Description

In order to move collision free from the feeder the robot moves by a depart position. The depart position is defined as an offset from the feeder position. This offset is executed according to the feeder workobject.

FEEDER_ DEPART _Z defines minimum height at which the robot can depart from the feeder. A positive FEEDER_ DEPART _Z results in a higher depart position.

In order to achieve a collision free cycle the FEEDER_ DEPART _Z will be overruled by the pallet height. If the pallet is higher than the feeder depart position the actual pallet height will be used.

FEEDER_ DEPART _Z is of type num and declared in the module Settings.

5.4.16 LOW_APPROACH_OFFSET

Usage

LOW_APPROACH_OFFSET is used as an alternative offset for the pallet approach position on the top layer.

Description

In order to move collision free to the pallet the robot moves by an approach position. The approach position is defined as an offset from the pallet position. This offset is executed in to the pallet workobject.

LOW_APPROACH_OFFSET _Z defines minimum height at which the robot can approach the pallet. A positive LOW_APPROACH_OFFSET results in a higher approach position.

Typically the approach position is located above the layer it's currently building to be able to orientate the gripper collision free. The approach position is located away, in X and Y direction, from the boxes already placed on the pallet to make a collision free path possible down to the place position. The low approach position is then located, just above the place position however it maintains the X and Y approach offsets.

To gain an extra layer on the pallet the, it is possible to let the robot orient the gripper while moving up from the feeder, instead of when it's moving over the pallet. In this way it's not needed to approach above the layer currently being build but the robot can move at a low height over the previous layer i.e. just above the place position.

LOW_APPROACH_OFFSET is of type num and declared in the module Settings.

5.4.17 obFeeder

Usage

obFeeder contains the workobject of the active feeder.

Description

obFeeder contains the workobject data for the active feeder. It will automatically be set by Palletizing Template

Do not define this this workobject when setting up the palletizing cell, as this workobject will be overwritten by the value of either obFeeder1 or obFeeder2, depending on which feeder is active.

obFeeder should be used in the motion instructions to handle the box on the feeder.

obFeeder is of type wobjdata and declared in the module PatternCalc.

Note

Do not define this workobject when setting up the palletizing cell. This workobject should be used in the motion instruction for handling boxes on the feeder.

Related information

For information about	See
Definition of workobjects	See RAPID reference manual – wobjdata

5.4.18 obFeeder1 and obFeeder2

Usage

obFeeder1 and obFeeder2 are used as calibration workobjects for a feeder.

Description

Palletizing Template supports two feeders. For each feeder a separate workobject is foreseen. One or both of these workobjects, depending on the amount of feeders used, need to be calibrated when setting up the palletizing cell see chapter 3.2.1 for more information.

obFeeder1 and obFeeder2 are of type wobjdata and declared in the module Settings.

Note

These workobjects should be defined when setting up the palletizing cell.

Related information

For information about	See
Definition of workobjects	See RAPID reference manual – wobjdata

5.4.19 obPallet

Usage

obPallet contains the workobject for the active pallet.

Description

obPallet contains the workobject data for the active pallet. It will automatically be set by Palletizing Template

Do not define this this workobject when setting up the palletizing cell, as this workobject will be overwritten by the value of either obPallet1, obPallet2, obPallet3 or obPallet4 , depending on which pallet is active.

obPallet should be used in the motion instructions to handle the box on the pallet.

obPallet is of type wobjdata and declared in the module PatternCalc.

Note

Do not define this workobject when setting up the palletizing cell. This workobject should be used in the motion instruction for handling boxes on the pallet.

Related information

For information about	See
Definition of workobjects	See RAPID reference manual – wobjdata

5.4.20 obPallet1, obPallet2, obPallet3 and obPallet4

Usage

obPallet1, obPallet2, obPallet3 and obPallet4 are used as calibration workobjects for a feeder.

Description

Palletizing Template supports four pallets. For each pallet a separate workobject is foreseen. One or several of these workobjects, depending on the amount of pallets used, need to be calibrated when setting up the palletizing cell see chapter 3.1.1 for more information.

obFeeder1 and obFeeder2 are of type wobjdata and declared in the module Settings.

Note

These workobjects should be defined when setting up the palletizing cell.

Related information

For information about	See
Definition of workobjects	See RAPID reference manual – wobjdata

5.4.21 obSheetMag

Usage

obSheetMag contains the workobject for the slipsheet magazine.

Description

obSheetMag contains the workobject data for the slipsheet magazine. It will automatically be set by Palletizing Template

Do not define this this workobject when setting up the palletizing cell, as this workobject will be overwritten by the value obSheetMag1.

obSheetMag should be used in the motion instructions to handle slipsheet.

obSheetMag is of type wobjdata and declared in the module PatternCalc.

Note

Do not define this workobject when setting up the palletizing cell. This workobject should be used in the motion instruction for slipsheets.

Related information

For information about	See
Definition of workobjects	See RAPID reference manual – wobjdata

5.4.22 obSheetMag1

Usage

obSheetMag1 is used as calibration workobjects for the slipsheet magazine.

Description

Palletizing Template supports one slipsheet magazine, for a workobject is foreseen. This workobject need to be calibrated when setting up the palletizing cell see chapter 3.3.1 for more information.

obSheetMag1 is of type wobjdata and declared in the module Settings.

Note

This workobject should be defined when setting up the palletizing cell.

Related information

For information about	See
Definition of workobjects	See RAPID reference manual – wobjdata

5.4.23 OPT_TOP_SAFE_X

Usage

OPT_TOP_SAFE_X{4} is used as offset for the pallet approach position when the optimize top layer functionality is used. OPT_TOP_SAFE_X{4} is an array with an offset for each of the 4 pallet locations.

Description

With the optimize top layer functionality the robot will gain an extra layer on the pallet. To achieve this the robot will use the low offset to approach the place position on the pallet. In this case the robot will not rotate the box to the correct orientation while moving over the pallet, but will orientate the box will moving up from the feeder. This implicates that the approach position will need to placed outside the pallet. The variable OPT_TOP_SAFE_X will define how far the robot has to stay outside the pallet. OPT_TOP_SAFE_X is defines the offset from the pallet edge along the X axis of the pallet workobject

OPT_TOP_SAFE_X{4} is an array of type num and declared in the module Settings.

5.4.24 OPT_TOP_SAFE_Y

Usage

OPT_TOP_SAFE_Y{4} is used as offset for the pallet approach position when the optimize top layer functionality is used. OPT_TOP_SAFE_Y{4} is an array with an offset for each of the 4 pallet locations.

Description

With the optimize top layer functionality the robot will gain an extra layer on the pallet. To achieve this the robot will use the low offset to approach the place position on the pallet. In this case the robot will not rotate the box to the correct orientation while moving over the pallet, but will orientate the box will moving up from the feeder. This implicates that the approach position will need to placed outside the pallet. The variable OPT_TOP_SAFE_Y will define how far the robot has to stay outside the pallet. OPT_TOP_SAFE_Y is defines the offset from the pallet edge along the X axis of the pallet workobject

OPT_TOP_SAFE_Y{4} is an array of type num and declared in the module Settings.

5.4.25 PALLET_APPROACH_X

Usage

PALLET_APPROACH_X is used as offset for the pallet approach position.

Description

In order to move collision free to the pallet the robot moves by an approach position. The approach position is defined as an offset from the pallet position. This offset is executed according to the pallet workobject.

A positive value of PALLET_APPROACH_X defines a distance away from the boxes already placed on the pallet.

PALLET_APPROACH_X is used for default approach height (PALLET_APPROACH_Z) as well as for the low approach height (LOW_APPROACH_OFFSET)

PALLET_APPROACH_X is of type num and declared in the module Settings.

5.4.26 PALLET_APPROACH_Y

Usage

PALLET_APPROACH_Y is used as offset for the pallet approach position.

Description

In order to move collision free to the pallet the robot moves by an approach position. The approach position is defined as an offset from the pallet position. This offset is executed according to the pallet workobject.

A positive value of PALLET_APPROACH_Y defines a distance away from the boxes already placed on the pallet.

PALLET_APPROACH_Y is used for default approach height (PALLET_APPROACH_Z) as well as for the low approach height (LOW_APPROACH_OFFSET)

PALLET_APPROACH_Y is of type num and declared in the module Settings.

5.4.27 PALLET_APPROACH_Z

Usage

PALLET_APPROACH_Z is used as offset for the pallet approach position.

Description

In order to move collision free to the pallet the robot moves by an approach position. The approach position is defined as an offset from the pallet position. This offset is executed according to the pallet workobject.

PALLET_APPROACH_Z defines minimum height at which the robot can approach the pallet. A positive PALLET_APPROACH_Z results in a higher approach position.

In order to achieve a collision free cycle the PALLET_APPROACH_Z will be overruled by the feeder height. If the feeder is higher than the pallet approach position the actual feeder height will be used.

PALLET_APPROACH_Z is of type num and declared in the module Settings.

5.4.28 PALLET_DEPART_X

Usage

PALLET_DEPART_X is used as offset for the pallet depart position.

Description

In order to move collision free from the pallet the robot moves by a depart position. The depart position is defined as an offset from the pallet position This offset is executed according to the pallet workobject.

In most cases this offset will be 0 as the robot should move straight up from the pallet position.

PALLET_DEPART_X is of type num and declared in the module Settings.

5.4.29 PALLET_DEPART_Y

Usage

PALLET_DEPART_Y is used as offset for the pallet depart position.

Description

In order to move collision free from the pallet the robot moves by a depart position. The depart position is defined as an offset from the pallet position. This offset is executed according to the pallet workobject.

In most cases this offset will be 0 as the robot should move straight up from the pallet position.

PALLET_DEPART_Y is of type num and declared in the module Settings.

5.4.30 PALLET_DEPART_Z

Usage

PALLET_DEPART_Z is used as offset for the feeder depart position.

Description

In order to move collision free from the pallet the robot moves by a depart position. The depart position is defined as an offset from the pallet position. This offset is executed according to the pallet workobject.

PALLET_ DEPART _Z defines minimum height at which the robot can depart from the pallet. A positive PALLET_ DEPART _Z results in a higher depart position.

In order to achieve a collision free cycle the PALLET_ DEPART _Z will be overruled by the feeder height. If the feeder is higher than the pallet depart position the actual feeder height will be used.

PALLET_ DEPART _Z is of type num and declared in the module Settings.

5.4.31 PickSpeed

Usage

PickSpeed defines the speed the robot moves at from the feeder position to the feeder depart position.

Description

PickSpeed defines the speed the robot moves to the feeder depart position. This speed can be set / changed from the Motion tuning tab of the Palletizing Template FlexPendant app.

The feeder depart speed is usually set to a lower value in order not to lose box when picking it from the feeder.

PickSpeed is of type speeddata and declared in the module Settings.

5.4.32 PickTime

Usage

PickTime defines the time the robot wait in the pick position.

Description

PickTime is used by the robot to wait in the pick position for the gripper to close or for the vacuum circuit to reach sufficient vacuum. The pick time can be set / changed from the tune tab of the Production menu in the Palletizing Template FlexPendant app.

The required pick time depends on the gripper type and boxes to be handled.

PickTime is of type num and declared in the module appdata.

5.4.33 PlaceSpeed

Usage

PlaceSpeed defines the speed the robot moves at from the pallet approach position to the pallet position.

Description

PlaceSpeed define the speed the robot moves to the pallet position. This speed can be set / changed from the Motion tuning tab of the Palletizing Template FlexPendant app.

The place speed is usually set to a lower value in order to move gradually to the pallet place position.

PlaceSpeed is of type speeddata and declared in the module Settings.

5.4.34 PlaceTime

Usage

PlaceTime defines the time the robot wait in the place position.

Description

PlaceTime is used by the robot to wait in the place position for the gripper to open or for the vacuum circuit to switch off. The place time can be set / changed from the tune tab of the Production menu in the Palletizing Template FlexPendant app.

The required place time depends on the gripper type and boxes to be handled.

PlaceTime is of type num and declared in the module appdata.

5.4.35 TargetFeeder

Usage

TargetFeeder contains the number of the active feeder.

Description

The feeder handled in the active pick-place is set by the instruction SetPalletizeCycle. This value is stored on the TargetFeeder variable for use during the palletizing cycle.

TargetFeeder is of type num and declared in the module PickPlaceItem.

TargetFeeder must not be changed by the user.

Related information

For information about	See
Set the palletizing cycle	See SetPalletizeCycle on page XXX

5.4.36 TargetPallet

Usage

TargetPallet contains the number of the active feeder.

Description

The pallet handled in the active pick-place is set by the instruction SetPalletizeCycle. This value is stored on the TargetPallet variable for use during the palletizing cycle.

TargetPallet is of type num and declared in the module PickPlaceItem.

TargetPallet must not be changed by the user.

5.4.37 ReturnSpeed

Usage

ReturnSpeed defines the speed the robot moves at from the pallet to the feeder position.

Description

ReturnSpeed defines the speed the robot moves to the feeder position. This speed can be set / changed from the Motion tuning tab of the Palletizing Template FlexPendant app.

This speed is usually set as high as possible as the robot doesn't carry any boxes.

ReturnSpeed is of type speeddata and declared in the module Settings.

5.4.38 PalletSpeed

Usage

PalletSpeed defines the speed the robot moves at from the feeder depart position to the pallet approach position.

Description

PalletSpeed define the speed the robot moves to the pallet approach position. This speed can be set / changed from the Motion tuning tab of the Palletizing Template FlexPendant app.

The speed to the pallet approach is usually set to a higher value in order to move as quickly as possible to the pallet.

PalletSpeed is of type speeddata and declared in the module Settings.

5.4.39 PickZone

Usage

zInfeed defines the zone the robot uses in the feeder depart position.

Description

zInfeed defines the zone the robot uses in the feeder depart position to cut the corners when moving from the feeder to the pallet. This zone can be set / changed from the tune tab of the Production menu in the Palletizing Template FlexPendant app.

This zone must be chosen such that that the robot can move collision free to the pallet with boxes in the gripper.

zInfeed is of type zonedata and declared in the module Settings.

5.4.40 zPallet

Usage

zPallet defines the zone the robot uses in the pallet approach position.

Description

zPallet defines the zone the robot uses in the pallet approach position to cut the corners when moving from the feeder to the pallet. This zone can be set / changed from the tune tab of the Production menu in the Palletizing Template FlexPendant app.

This zone must be chosen such that that the robot can move collision free to the pallet with boxes in the gripper.

zPallet is of type zonedata and declared in the module Settings.

5.4.41 zPalletize

Usage

zPalletize defines the zone the robot uses in the pallet depart and feeder approach position.

Description

zPallet defines the zone the robot uses I to cut the corners when moving back from the pallet to the feeder. This zone can be set / changed from the tune tab of the Production menu in the Palletizing Template FlexPendant app.

This zone is typically set to a high value so the robot can move as quickly as possible back from the pallet to the feeder.

zPalletize is of type zonedata and declared in the module Settings.

6 Program examples

Introduction

In this chapter program examples are given on how to implement specific functionality.

6.1 Start palletizing at a specific layer and box

Description

In this example the robot will continue with an unfinished pallet. The box and the layer from which the robot should continue are in this example set by a PLC. This is typically useful when an incomplete pallet needs to be completed by the robot.

The PLC should select the recipe, which is in this case based based on a group input, the layer and box to start from.

Care must be taken that the pallet is properly, in the correct orientation, placed in the robot cell.

A manual function in the Palletize Template FlexPendant app also exists for this functionality (see chapter 4.2.1.2)

Basic example

The following example illustrates the PickFeeder procedure.

Example 1:

```
PROC PickFeeder()
    VAR robtarget PickTarget;
    !
    VAR num NrOfItems;

    ! Get the feeder pick position
    GetFeederItem PickTarget,tVacuum\NrOfItems:=NrOfItems;
    MoveLiftKit;

    ! Lock axis 4 to avoid singularities
    SingArea\Off;
    ConfL\Off;

    ! Move to the feeder approach position
    MoveJ GetFeederApproach(PickTarget),vToInfeed,z500,tVacuum
        \WObj:=obFeeder;

    ! Wait for a box to be present at the end of the feeder
```

```
    WaitDI diBoxAtFeeder,1;
    WaitLiftKit;

    ! Move to the pick position
    MoveL PickTarget,vToInfeed,fine,tVacuum\WObj:=obFeeder;

    ! Close the gripper and check for vacuum
    VacuumOn NrOfItems;

    ! Reduce acceleration if needed
    AccSet Acceleration,Acceleration;

    ! Move above the conveyor
    MoveL GetFeederDepart(PickTarget),vInfeedDepart,z200,tVacuum
       \WObj:=obFeeder;
ENDPROC
```

First the pick position (PickTarget) is retrieved. The lift kit (if applicable) is moved to the correct height. As the lift kit is not an external axis which means that it's not included in the robot controller, the height of the lift kit will be compensated in the generated pick target. This is followed by a motion to the pick approach position. In the pick approach position the robot will wait for a signal from the feeder indicating that a box is available to be picked up.

Then the robot will move to the pick position switch the vacuum on and will check if the vacuum is present. The robot will wait for vacuum for a configurable time (see chapter 4.3.2).

Vacuum control is done by a call to the VacuumOn procedure located in the Gripper module.

When vacuum is present the robot will pick the box and normal program execution continues (i.e. the box will be placed on the pallet)

6.2 Place on pallet example

Description

In this example the robot moves to the selected pallet and places the box on the pallet. The pallet is selected using the SetPalletizeCycle instruction (see chapter 5.1.9). The SetPalletizeCycle instruction also sets the obPallet workobject (it copies it from either obPallet1 obPallet1, obPallet2, obPallet3 or obPallet4)

Basic example

The following example illustrates the PlacePallet routine.

Example 1:

```
PROC PlacePallet()
    VAR robtarget PlaceTarget;

    ! Get the pallet place position and the associated workobject
    GetPalletItem PlaceTarget,tVacuum;

    ! Lock axis 4 to avoid singularities
    SingArea\LockAxis4;

    ! Move to the pallet place position
    MoveJ GetPalletApproach(PlaceTarget),vToPallet,z200,tVacuum
        \WObj:=obPallet;

    MoveL GetPalletApproach(PlaceTarget\LowApproach),vPalletApproach,
        z20,tVacuum\WObj:=obPallet;
    MoveL PlaceTarget,vPalletApproach,fine,tVacuum\WObj:=obPallet;

    ! Open the gripper
    VacuumOff;

    ! Confirm the place cycle
    PlaceCycleDone TRUE;

    MoveL\Conc,GetPalletDepart(PlaceTarget),vToInfeed,z500,tVacuum
        \WObj:=obPallet;
ENDPROC
```

First the place workobject (obPlace) and the and the place position (ToPoint) are retrieved. This is followed by a motion to the pallet place position

Once the robot has reached the place position the status of the sensor diVacuumOn is checked.

If this input is high the box is present at the gripper and can thus be placed. The ELSE part of the IF statement is then executed which means that the box will be reported as placed. PlaceCycleDone is called with the argument TRUE.

If the input is low the THEN part of the IF statement is executed. After switching the vacuum off the robot moves away from the box (moves 100 mm up). PlaceCycleDone is called with status FALSE, indicating the box is lost. The optional argument \BoxLost is added. By adding this argument the missing box will be colored in red in the pallet status indicator (see chapter 4.2.1). The program is now stopped to allow the operator to enter the cell and remove the box from the cell. For this case the box must be removed from the cell. Once the box is removed the operator can continue program execution. The box is again reported lost but without the optional argument \BoxLost. The box will turn yellow in the pallet status indicator (see chapter 4.2.1) and the robot will place the box at the same spot.

6.3 Start at main without losing the pallet status

Description

In the example below the robot can be started from the main routine without losing the pallet status. I.e. the robot will continue palletizing even after a start at main, thus the (partially filled) pallets need not to be removed.

Basic example

The following example illustrates the main routine.

Example 1

```
PERS string ActualRecipeName:="Demo2";

PROC main()
  VAR bool FileExists;

  IF ActualRecipeName<>RecipeName THEN
    InitAll;
  ELSE
    InitAll\BasicInit;
  ENDIF
  !
  SingArea\LockAxis4;
  !
  MoveHome;
  IF AppCommand=CMD_HOME_RUN Stop;
  !
  IF ActualRecipeName<>RecipeName THEN
    ReadDataFile RecipeName\Recipe;
    ReadDataFile RecipeName\Tune\FileExists:=FileExists;
    IF NOT FileExists THEN
      SetDefaultParams;
    ELSE
      UpdateMotionParams\UpdateOnly;
    ENDIF

    InitRecipe;
  ENDIF
  !
  ActualRecipeName:=RecipeName;
  !
  UpdateLPParams;
  !
  ProductionStatus:=STATUS_PRODUCTION;
  !
```

```
    ! ************************************************************************
    ! Release the gripper (example code. To be replaced by installation specific
          gripper control)
    Reset doGrip1;
    Reset doGrip2;
    Reset doGrip3;
    ! ************************************************************************
    !
    WHILE TRUE DO
      PickPlaceSeq;
      !
      UpdateMotionParams;
      !
      IF ProductionStatus=STATUS_END_CYCLE THEN
        Stop;
        ProductionStatus:=STATUS_PRODUCTION;
      ENDIF
      !
      IF ProductionStatus=STATUS_END_PALLET AND
          PalletStatus{TargetPallet}=PALLET_FULL THEN
        Stop;
        ProductionStatus:=STATUS_PRODUCTION;
      ENDIF
    ENDWHILE
  ERROR (ERR_HOMERUN)
    !
    ! Move to the home position
    MoveHome;
    !
    Stop;
    !
    ProductionStatus:=STATUS_PRODUCTION;
    !
    ResetRetryCount;
    RETRY;
  ENDPROC
```

In the example above the string variable ActualRecipeName stores the name of the running recipe. When the program is restarted from main the running recipe name is compared to the recipe name that can be selected in the Palletizing Template FlexPendant app.

If the running recipe name differs the recipe the operator has selected, a new recipe which will need to loaded and started from scratch, with empty pallets. In this case the full initialization will need to take place.

If the running recipe is the same as the selected recipe, Palletizing Template will continue with the running recipe without the need of removing the pallets. In this case only basic initialization is required. Basic initialization is achieved by adding the \BasicInit optional argument to the InitAll instruction. Reading the recipe data file and initializing the recipe data is then no longer required.

7 Palletizing Template pattern builder

Introduction

Palletizing Template uses predefined patterns in its configurator to build recipes from. These patterns are scalable, which means that a particular pattern lay-out can be used for different recipes with different box dimensions. So a particular pattern lay-out will need to be created once.

A second advantage is that patterns can be scaled while the robot is running production. This means that box length, width and height can be changed for the active pattern while the robot is palletizing. This functionality is available from the Production menu under the Tune box tab (see chapter 4.3).

To achieve scaling of patterns, the patterns need to be built based on formulas. These formulas describe based on lengths and widths where the box is located relative to the origin of the pattern.

These patterns can then be stored in Palletizing Template pattern library files which must be copied to the flash drive on the robot controller. These libraries can then be used in the Stack configurator (see chapter 4.1) for use in new or existing recipes.

Palletizing Template pattern library files are JSON based. As these are difficult to create without any visual interface, a Palletizing Template pattern builder is available to build these libraries.

The Palletizing Template does not require extra installation. It is included as part of the delivered WebApp. It is enabled by default but can be disabled by setting the **“**disable_pattern_builder” property equals “true” inside the setup.json file located at “HOME/Palletizing_Files” directory.

Note

The following documentation corresponds to the LEGACY pattern designer used for IRC5. It is independent software that runs on the PC. However, the generated formulas, which are here explained, remains valid.

7.1 Installing the Palletizing Template pattern builder

Description

To install the Palletizing Template pattern builder unpack the Palletizing Template pattern builder.zip file to a convenient location on the PC. Run the setup.exe file and follow the instructions in the installation wizard.

The Palletizing Template pattern builder icon will appear on the desktop. Double click the icon the start the Palletizing Template pattern builder. The Palletizing Template pattern builder can also be started from the Windows start menu where it’s located under the ABB folder.

The Palletizing Template pattern builder is developed for Windows 10 or higher.

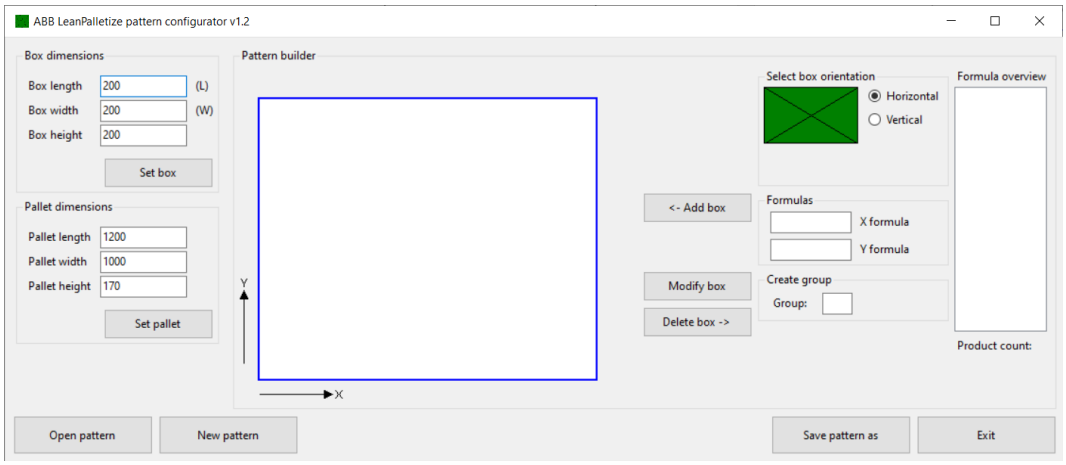
7.2 The Palletizing Template pattern builder

Description

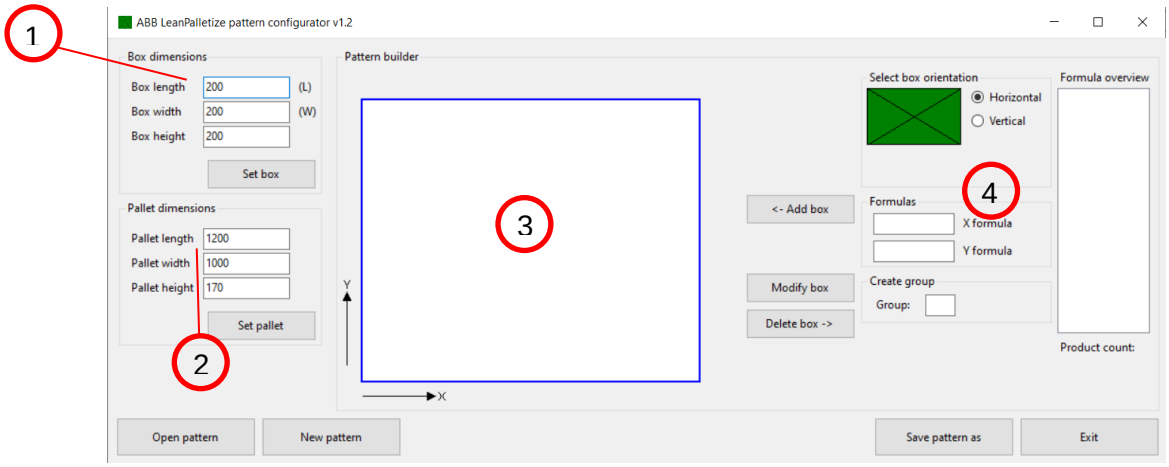
The Palletizing Template pattern builder can be used to create new pattern lay-outs or can be used to modify existing ones. The patterns can be saved to a Palletizing Template library file which can be downloaded to the robot controller for use in the stack configurator.

A Palletizing Template library file contains of the pattern for one layer. Mirroring or rotation functions are not available. So mirrored or rotated pattern libraries should be created separately.

When the Palletizing Template pattern builder is started the following screen will appear:



The pattern builder has four main area's:



Area	Description

1	Box dimension area. In this area box dimensions can be given. These dimensions will only be used for the pattern preview.
2	Pallet dimension area. In this area pallet dimensions can be given. These dimensions will only be used for the pattern preview.
3	Pattern preview area. This area shows the pattern while it’s created.
4	Area where the formulas of the box positions should be entered.

In the following chapters a more detailed description is given on the different area’s

To save the pattern as a Palletizing Template library some file naming conventions are in place and should be followed. Please find more information is chapter 7.2.7

Note

Naming conventions are in place when saving the pattern.

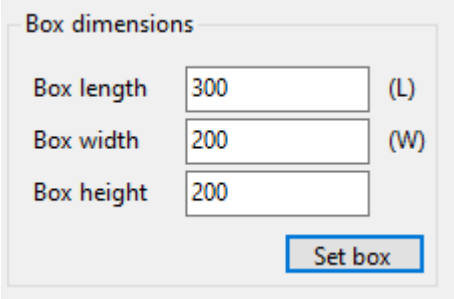
7.2.1 Box dimensions

Description

In the box dimensions area the size of the box can be specified. This size will only be used for preview purposes in the preview area.

The box dimensions can be changed at any time while creating the new pattern. New patterns should always be tested against different box sizes. Due to the nature of the formulas used to build some patterns may create overlapping boxes and will thus not be possible to stack with the robot. How to handle overlapping boxes be described later in this chapter.

Enter the length and width of an example box the three fields in this area. A height can also be entered but has no functionality.



When the dimensions are entered press the Set box button to activate the entered dimensions.

When a (partial) pattern is already created this pattern is immediately adapted to the new dimensions.

The box dimensions will not be saved in the Palletizing Template pattern library file.

Note

Box dimensions will NOT be stored in the Palletizing Template pattern library file.

Note

Always check a pattern with different box sizes!.

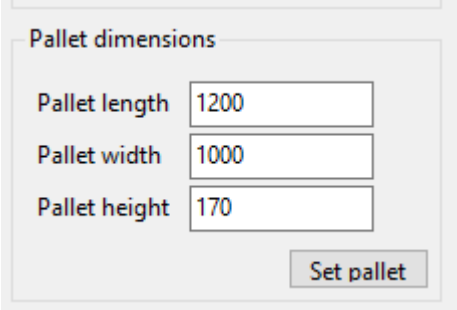
7.2.2 Pallet dimensions

Description

In the pallet dimensions area the size of the pallet can be specified. This size will only be used for preview purposes in the preview area.

The pallet dimensions can be changed at any time while creating the new pattern.

Enter the length and width of an example pallet the three fields in this area. A height can also be entered but has no functionality.



When the dimensions are entered press the Set pallet button to activate the entered dimensions.

The pallet size is immediately adapted to the new dimensions and the pattern is scaled and centered to match the new pallet size.

The pallet dimensions will not be saved in the Palletizing Template pattern library file.

Note

Pallet dimensions will NOT be stored in the Palletizing Template pattern library file.

7.2.3 Pattern preview

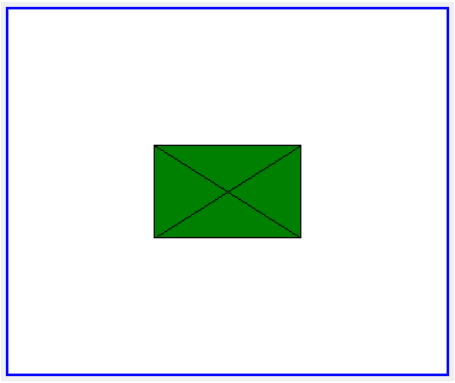
Description

In the pattern preview area a preview of the pattern while it’s created is given. The pattern will always be drawn centered on the pallet.

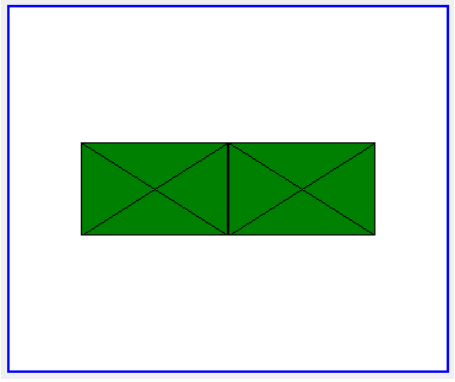
The pallet is shown by means of a blue outline. The pallet is drawn according to scale based on the length and width of the pallet dimensions (see chapter 7.2.2). Boxes are drawn to scale based on box and pallet dimensions.

Pattern centering

The pattern is always drawn centered to the pallet. This means that while building the pattern it will constantly be updated to the center position as more boxes are added to the pattern. When the first box is added to the pattern the preview will look like the example below.

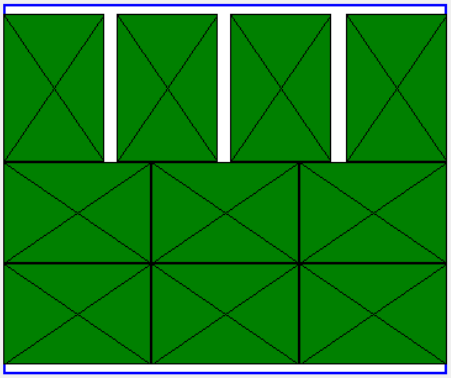


When a second box is added the pattern looks like the example below:



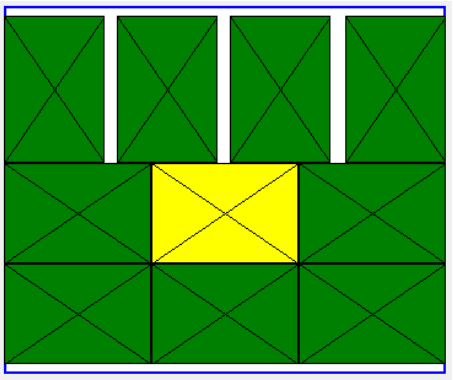
7.2.3 Pattern preview

The final pattern may look like the example below:



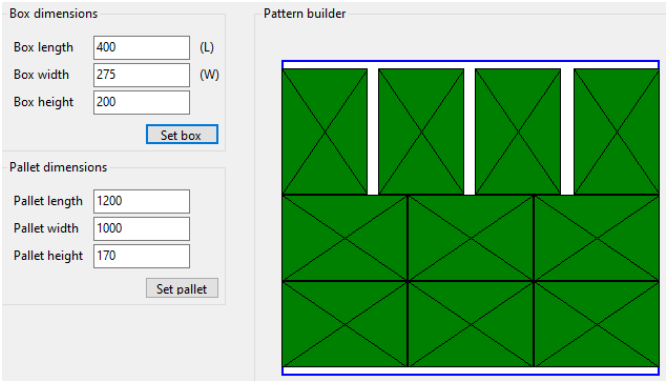
Box selection

When a box is selected the box will turn yellow. The formulas of this box can then be modified. To select the box click required box. To deselect click the selected box again.



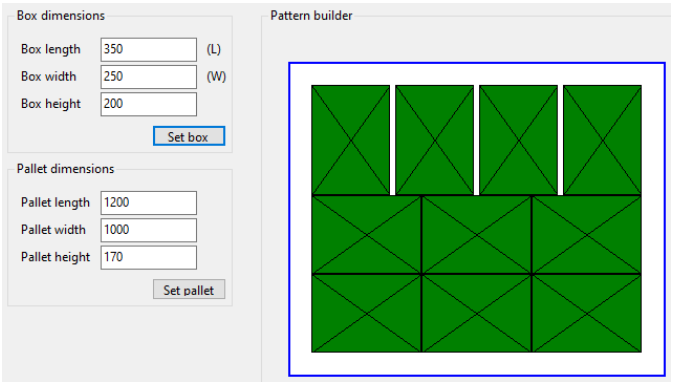
Size change

In the example below the box dimensions are set to 400 x 275 mm with pallet dimensions of 1200 x 1000 mm.

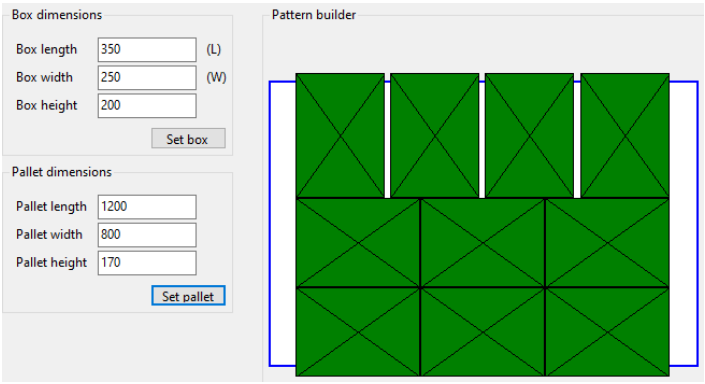


If the box dimensions are then changed to for example 350 x 250 mm, the pattern is scaled down as soon as the Set box button is pressed .

7.2.3 Pattern preview



Same applies for the pallet dimension. When the pallet dimensions are changed, the pallet is drawn on scale and the pattern is fitted to the same scale, as shown in the example below where the pallet dimension is changed to 1200 x 800 mm.



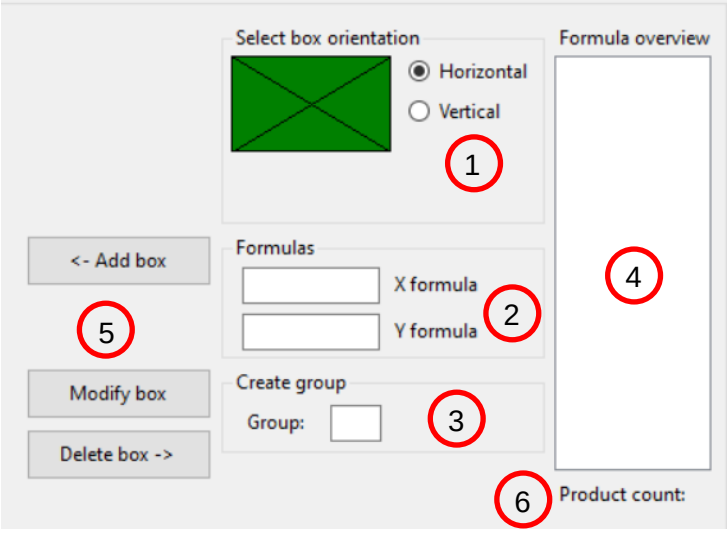
In this case the pattern is overlapping the pallet. The overlap of the pattern is not indicated as this may be acceptable and the pattern can be used for all different dimensions where the pattern does fit onto the pallet.

7.2.4 Pattern creation

Description

Patterns are created based on formulas where the location of a box on the pallet is defined by a number of box lengths and / or box widths away from the start corner of the pattern. Boxes can be placed in either horizontal or vertical direction. Boxes can be combined in a group which results in equal spacing between the boxes. Formulas are entered in the formula area

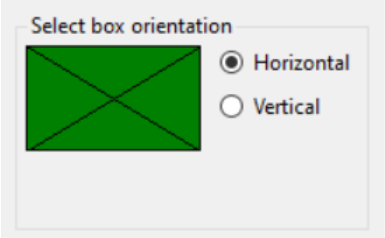
The formula area consist of the following fields:

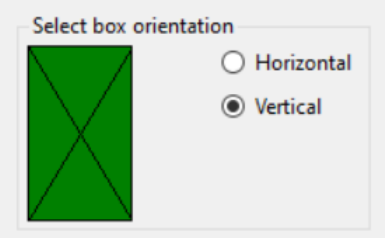


Nr	Description
1	Box orientation selection. Boxes can placed horizontal or vertical in the pattern
2	Formula entry. Formula's need to be given for each box displacement, for both the long and short side of the pallet
3	Group definition takes care of even distributing the distance between boxes in the same group
4	Formula overview. List with formulas of the configured boxes
5	Add to, remove from and modify boxes in the configured pattern
6	Product count. Indication of the amount of boxes configured in the pattern

Box orientation

Boxes can be placed in the pattern in two ways, horizontal and vertical.

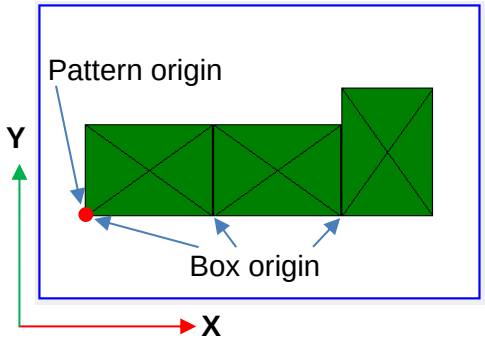




Before adding a box to the pattern the orientation must be selected.

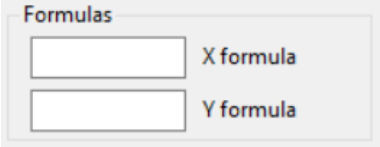
Box formula

Formulas are used to describe the location of the box in the pattern. The location is the box is based on the lower left corner of the box relative to the lower left corner of the pattern, as shown in the example below.



Per box two formulas are required. One formula describes the location of the lower left corner in X direction (along the long side of the pallet) and the second for the location of the lower left corner in Y direction (along the short side of the pallet).

A formula is based on the amount of lengths and widths a box is located from the pattern origin in both X and Y direction. A box length is indicated by L and a width is indicated by W. The L and W can be preceded by a value indicating the amount of lengths or widths. These values may also be fractional. Lengths and widths may be added and subtracted. If no offset is required the field can be left empty. Two fields are available for formulas in X and in Y direction.



When the formulas are entered in the X Formula and Y formula box, the box can be added to the pattern by pressing the <- Add box button.

The order in which the formulas are entered is of no importance. Palletizing Template will automatically reorder the stacking a make sure stacking is order is optimal and collision free.

Note

The pattern can be built in any order by the user. Palletizing Template will optimize wen running on the robot.

Patterns can be used for single or multi box handling. Palletizing Template will automatically optimize the pattern for multi box handling.

Note

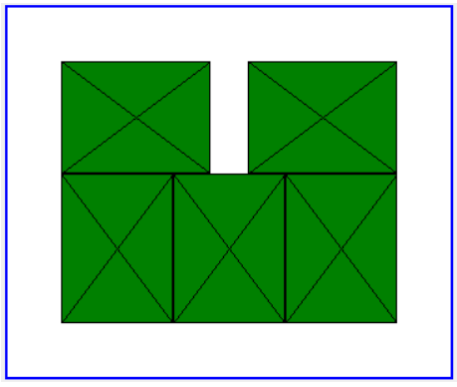
Palletizing Template will automatically optimize the pattern for multi box handling.

Basic example

The following example illustrate the box formulas.

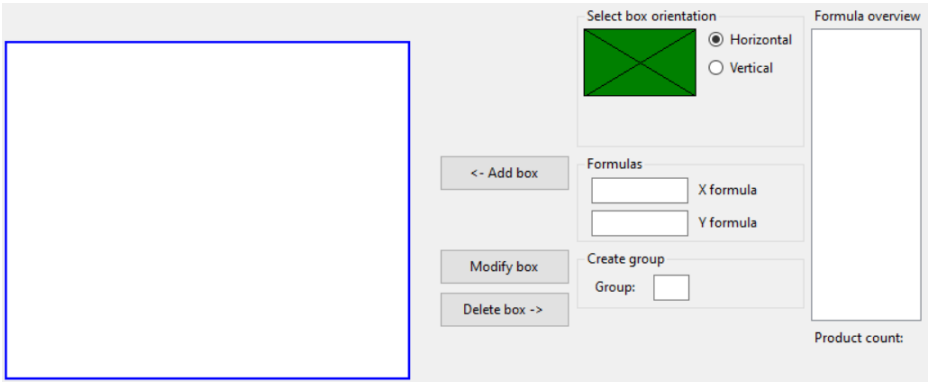
Example 1

In this example the pattern as shown below, consisting of 5 boxes, is created. Box dimensions are set to 400 x 300 mm

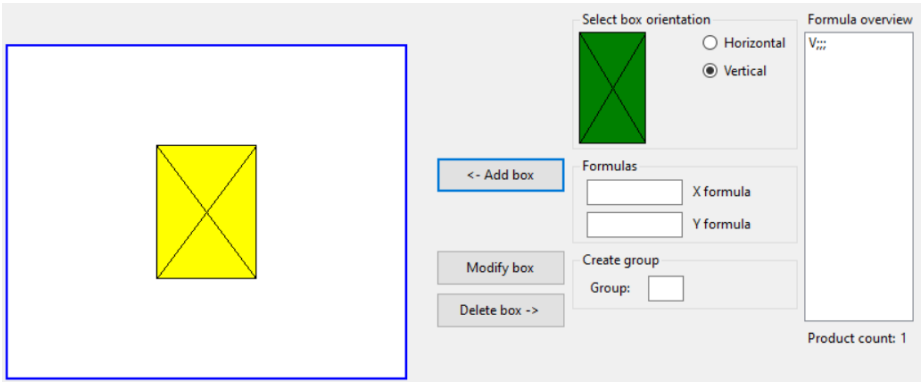


Start with a new pattern by pressing the New pattern button. The pattern preview field is empty.

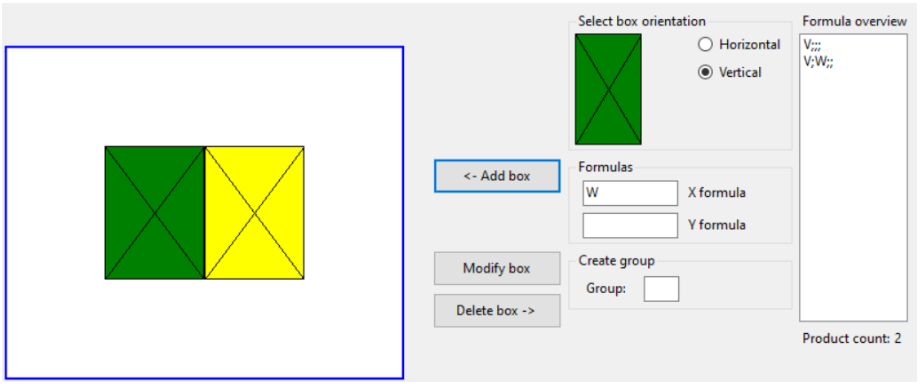
7.2.4 Pattern creation



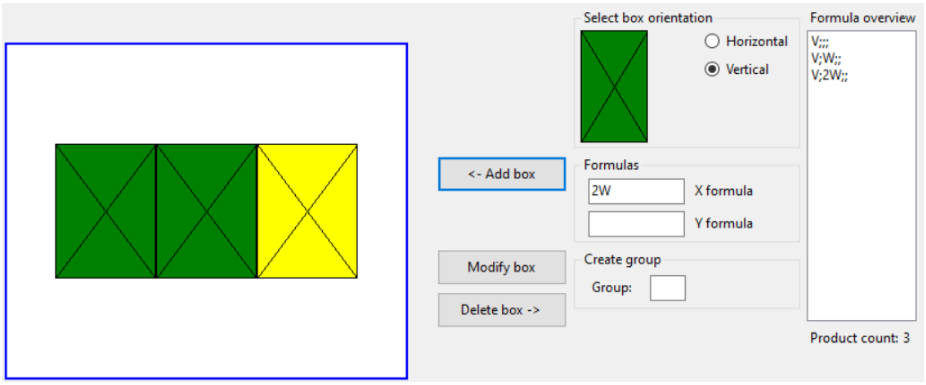
The first box is added in the lower left corner doesn't have any formula's, so both fields can be left empty. The orientation should be set to Vertical. When the <- Add box button is pressed the box will be added as shown below, centered on the pallet. The Formula overview field shows the formula for the particular box



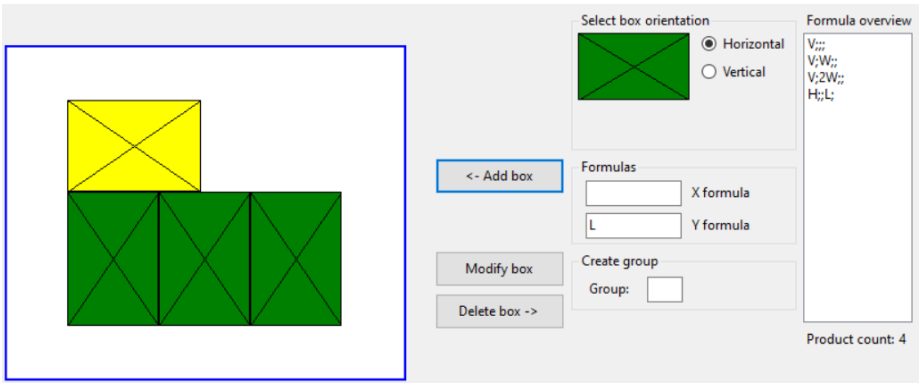
The next box should be placed vertically next to the first box. In this case the box is shifted one box width in X direction. The box orientation remains Vertical and W (1 x the box width) should be entered in the X Formula box. As the lower corner of the box remains at the same level as the first box, the Y Formula field should be left empty. Pressing <- Add box will then give the following result:



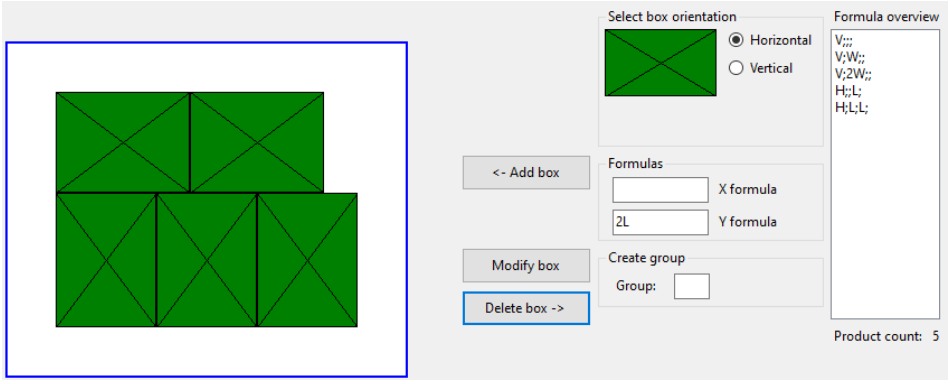
The third box requires the X formula of 2W (2 x the box width). The Y formula is again left empty. Adding the box will give following result:



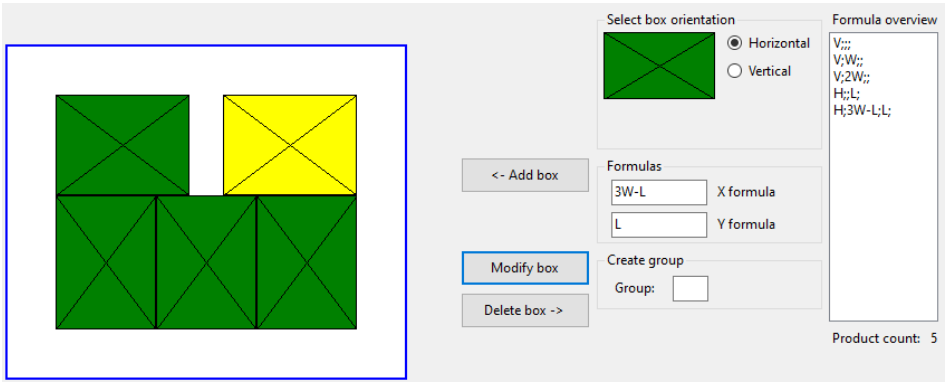
The fourth box will be placed on top of the first and second box with the lower left corner aligned with the lower left corner of the first box.
The box orientation should be set to Horizontal. The X formula should be left empty while the Y formula contains L (1 x the box length). This gives the following result:



If the last box is placed directly against the previous box, i.e. the X Formula is set to L, giving the following result:

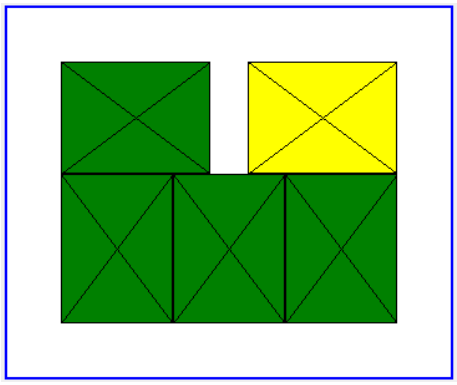


In this case the pattern isn't square as a gap is present in the upper right corner. For this particular case the gap should be placed between the top two boxes. To achieve this the X formula for the last (upper right) box needs to be changed to 3W-L. 3W place the lower left box corner at the end of the pattern, deducting one length places the box such that the pattern is square.



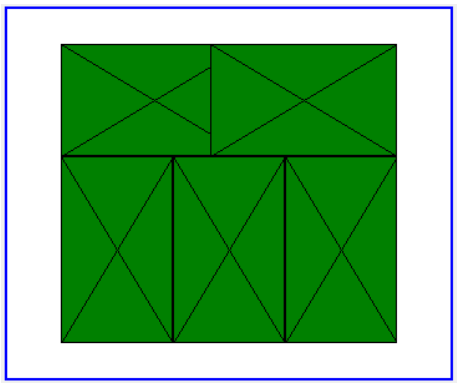
Overlapping boxes

Due to the nature of the formulas boxes can become overlapping. In the example below the X formula of the top right box is created in a way that the pattern is square but the position of this box depends on relationship between the length and width of the box. In this case the X formula is 3W-L



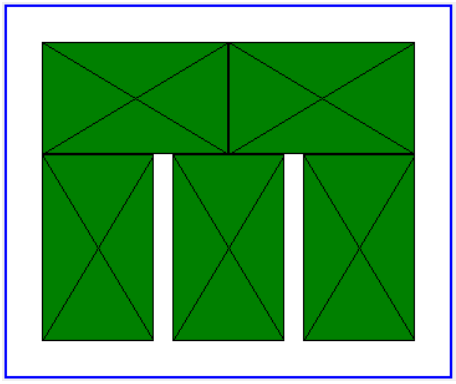
If 3 x the width is more than 2 x the length of a box the gap will appear between the two top boxes as shown above.

When 3 x the width is less than 2 x the length of a box there isn't enough room for the gap between the top two boxes, causing the two top boxes to overlap.



No error indication is given about this occurrence as patterns are not linked to physical box dimensions.

In some cases it might be necessary to create different patterns for different box length width ratios. In the above shown case a second pattern may be required. The gap may then be spaced between the lower boxes as shown in the example below.



For this example following formulas can be used, starting from the lower left corner:

1. Vertical, X formula: <empty>, Y formula: <empty>
2. Vertical, X formula: L-0.5W, Y formula: <empty>
3. Vertical, X formula: 2L-W, Y formula: <empty>
4. Horizontal, X formula: <empty>, Y formula: L
5. Horizontal, X formula: L, Y formula: L

Note

Due to the nature of the formulas boxes may become overlapping as box positions aren't based on physical dimensions.

Note

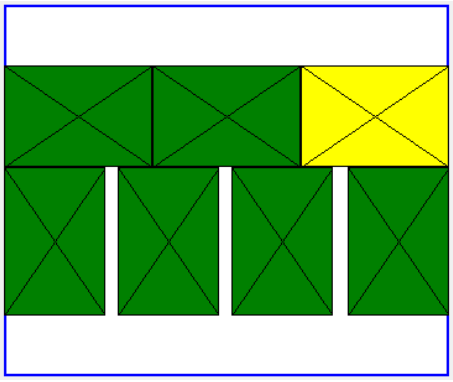
More than one solution may be required for a pattern to accommodate for different box length and width ratios.

In the pattern configurator on the FlexPendant the pattern will be indicated in red when box dimensions cause overlapping boxes (see chapter 4.1.4). In this case the same pattern with different formula set-up needs to be selected.

> **Note**
>
> **The pattern editor on the FlexPendant will indicate patterns in red when box dimensions used for the recipe cause overlapping boxes.**

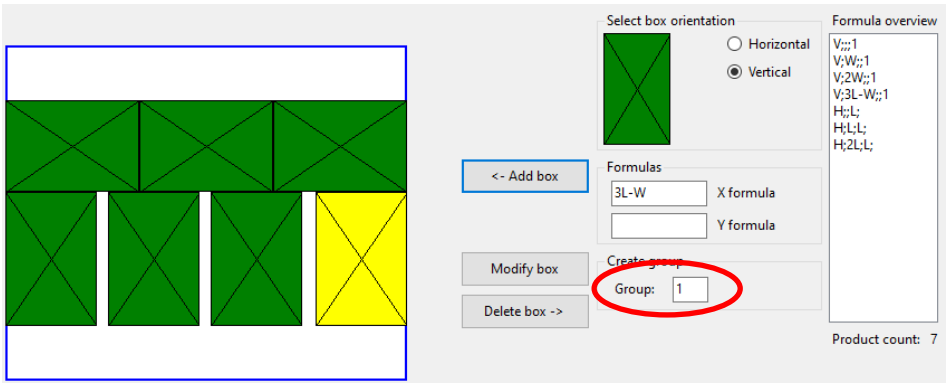
Grouping boxes

When the spacing between boxes has to be divided equally, finding proper formulas may be difficult. This typically happens when spacing is to be spread equally between for example 4 boxes as shown in the example below.



Palletizing Template provides a solution where boxes can grouped. Palletizing Template will then locate the two outer boxes and will distribute the spacing equally between the inner boxes. For the example above the lower row of boxes are grouped. All boxes between the outer boxes are spaced equally independent of their formulas.

To group a number of boxes a unique group number has to be given for the boxes in this group in the Group field



The formulas for the boxes in the lower row of the example above are as follows, starting from the lower left corner:

1. Vertical, X formula: <empty>, Y formula: <empty>, Group: 1
2. Vertical, X formula: W, Y formula: <empty>, Group: 1
3. Vertical, X formula: 2W, Y formula: <empty>, Group: 1
4. Vertical, X formula: 3L-W, Y formula: <empty>, Group: 1
5. Horizontal, X formula: <empty>, Y formula: L, Group: <empty>
6. Horizontal, X formula: L, Y formula: L, Group: <empty>
7. Horizontal, X formula: 2L, Y formula: L, Group: <empty>

By placing the boxes in such way that the pattern is square, last box is placed at 3 x the box length – 1x the box width, the space is divided equally. The boxes on the top row do not require grouping. Up to 10 different groups can be made.

If the ratio of box length and width changes boxes in a group may become overlapping. In this case a second pattern should to be made where the top row uses groups.

> **Note**
>
> **Box overlap can depending on the length / width ratio occur for grouped boxes.**

Modify box formulas

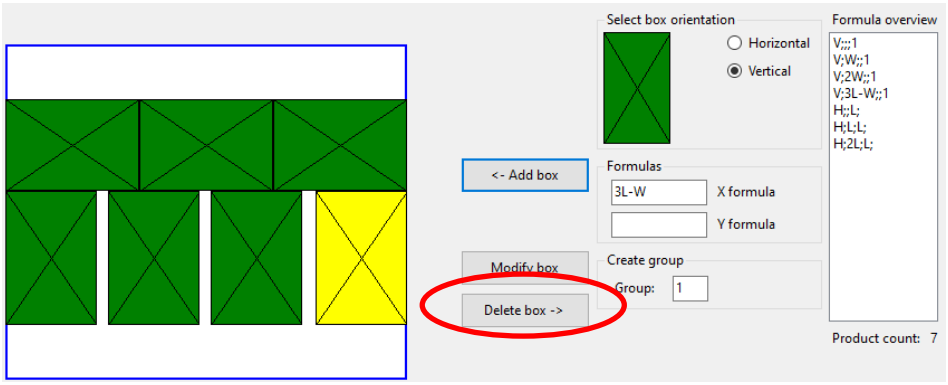
To modify the formula of a particular box, select the box is by clicking on the box in the pattern. The selected box will be highlighted by a yellow color.

The selected orientation, formulas and group will be shown in the respective fields. The existing box settings can be modified. Once the settings are modified the Modify box button can be pressed to update the selected box.

Delete boxes

To delete a box from the pattern select the box by clicking on the box in the pattern. The selected box will be highlighted by a yellow color.

7.2.4 Pattern creation

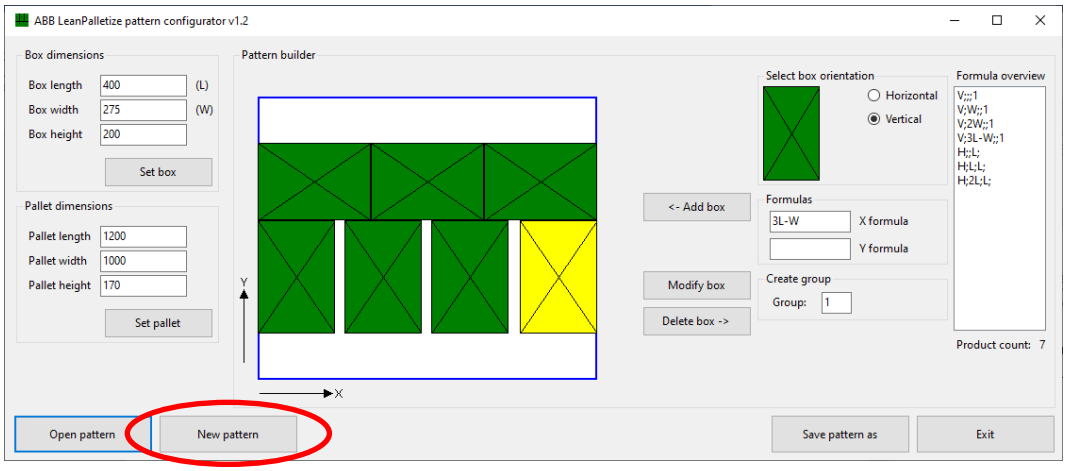


Press the Delete box-> button to remove the box from the pattern

7.2.5 New pattern

Description

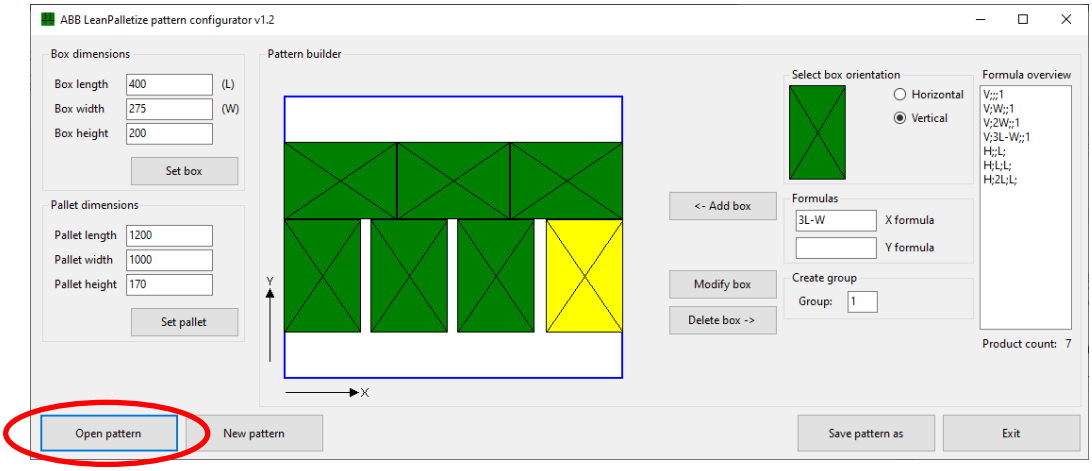
To start with a new pattern click the New pattern button. After acknowledgement the existing pattern will be deleted and a new pattern can be build.



7.2.6 Open an existing library

Description

Existing patterns can be modified by opening the existing pattern. To open an existing pattern click the Open pattern button. Select the desired pattern from the file open dialog window. The pattern will be loaded and can be modified.



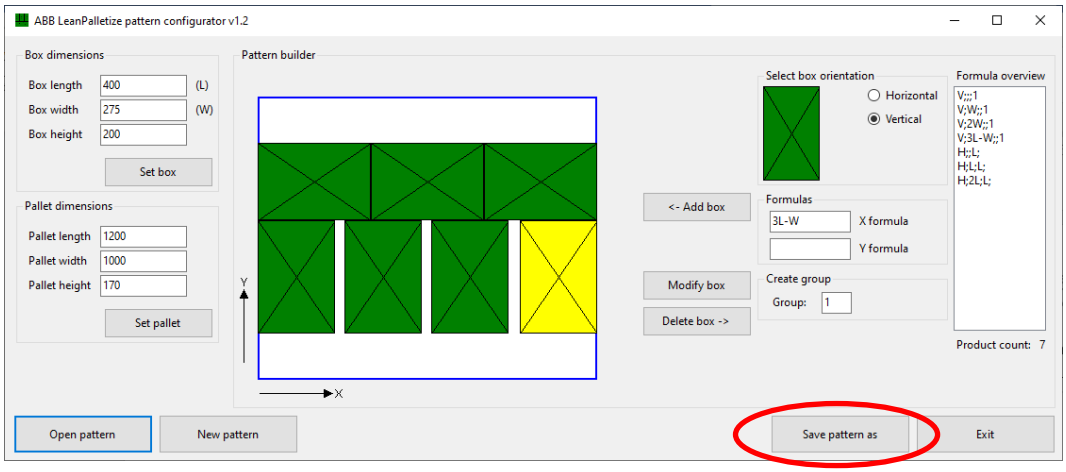
As the box and pallet dimensions used at the time the pattern was created aren't saved in the pattern library file, the box and pallet dimensions should be set (see chapter 7.2). The pattern can be tested against different box and pallet dimensions and be modified if needed.

On a system with a virtual controller the library files can be loaded from the HOME:/Palletizing_Files/Patterns folder of the virtual controller.

7.2.7 Saving the pattern as a library

Description

Patterns can be saved to library files by clicking the Save pattern as button. The Save as dialog will appear.



On a system with a virtual controller the library files can be saved directly into the HOME:/Palletizing_Files/Patterns folder of the virtual controller.

7.2.8 Installing the pattern libraries on the robot controller

Description

To install the pattern libraries on the robot controller the library files need to be downloaded to the robot controller to the Palletizing_Files/Patterns folder, located under the HOME: drive.

The File Transfer tool in RobotStudio can be used to download the library files.

In order to use the newly added libraries for an existing or new recipe the Palletizing Template application on the FlexPendant need to be closed and opened again.

When a library is used in a Palletizing Template recipe, the contents of the library file is copied into this recipe file. This will ensure that the recipe can be used for production even if the library doesn't exist anymore. This also implicates that the recipe doesn't contain a link to the library file. When an existing library file is modified, the recipes that were created using the original library will not be changed. If these recipes require the updated pattern, these recipes need to be modified in the pattern configurator on the FlexPendant.

Note

Updating library files doesn't impact existing recipes. Existing recipes will remain unchanged.

Index

G

H

I

L

M

N

O

P

Q

ABB N.V.
Robotics
Hoge Wei 27
1930 Zaventem
Belgium
Telephone +32 (0)2 718 63 11

abb.com/robotics