ROBOTICS

Application manual

Machine Tending Template

Application manual

Machine Tending Template 1.0.0

RobotWare 7/8

Document ID: Application manual Machine Tending Template 1.0.0

Revision: A

© Copyright 2026 ABB. All rights reserved.
Specifications subject to change without notice.

© Copyright 2026 ABB. All rights reserved.

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.

© Copyright 2026 ABB. All rights reserved.
Specifications subject to change without notice.

© Copyright 2026 ABB. All rights reserved.

Table of contents

1.1 Overview of this manual ... 9
1.2 Product documentation ... 11
1.3 Safety ... 12

1 Introduction ... 13

1.1 Machine Tending Template overview ... 13
1.2 RAPID framework and Wizard Easy Programming ... 13
1.3 Relationship between WebApp, Wizard, and RAPID ... 14
1.4 WebApp and Configuration Wizard ... 15
1.5 RobotWare compatibility ... 16
1.6 Gripper support ... 16
1.7 Machine interface ... 17
1.8 Station variants ... 17
1.8.1 Omnivance ... 18
1.8.2 ProFeederS 5 Trays ... 19
1.9 Production cycle overview ... 19
1.10 Customization with ABB tools ... 20

2 Installation ... 22

2.1 System requirements ... 22
2.2 Hardware installation ... 22
2.2.1 Gripper design ... 22
2.2.2 Machine interface wiring ... 23
2.3 Software installation ... 24
2.3.1 Add-In installation ... 24
2.3.2 Building a robot controller ... 25
2.3.3 HOME folder contents ... 25
2.3.4 System parameters ... 26
2.3.5 RAPID program and system modules ... 26
2.3.6 WebApp ... 27
2.3.7 Uninstallation ... 27

3 System configuration ... 28

3.1 Machine definitions ... 28
3.1.1 Machine HAL module (MTT_Machine) ... 28
3.1.2 Machine door configuration ... 28
3.1.3 Spindle/chuck configuration ... 29
3.1.4 Machine cycle control ... 32
3.1.5 Machine wait position ... 32
3.1.6 Machine enter and exit positions ... 32

© Copyright 2026 ABB. All rights reserved.

3.1.7 Machine variants 33
3.2 Workstation and tray definitions 34
3.2.1 Station HAL module (MTT_User) 34
3.2.2 Workstation workobjects 34
3.2.3 Workstation open and close positions 34
3.2.4 Tray layout configuration 35
3.2.5 Tray increment mode 37
3.2.6 Tray usage mode (load and unload) 38
3.2.7 Workstation approach and depart offsets 38
3.2.8 Workstation state management 38
3.2.9 Station variants 39
3.3 Gripper configuration 41
3.3.1 Gripper HAL module (MTT_EndEffector) 41
3.3.2 Dual gripper support 41
3.3.3 Gripper definitions 41
3.3.4 Tool data calibration 42
3.3.5 Gripper feedback signals 43
3.3.6 Gripper control signals 43
3.3.7 Item missed handling 44
3.3.8 Blow-off configuration 44
3.4 IO signal configuration 45
3.4.1 Signal overview 45
3.4.2 Digital input signals (machine to robot) 45
3.4.3 Digital output signals (robot to machine) 45
3.4.4 Hardware Integration Options 46
3.5 Load data configuration 48
3.5.1 Raw workpiece load data 48
3.5.2 Finished workpiece load data 49
3.5.3 Load management during cycle 49

4 Wizard blocks, RAPID, and WebApp data 50

4.1 Overview 50
4.2 Block palettes 51
4.2.1 MTT Cycle blocks 51
4.2.2 MTT Machine blocks 52
4.2.3 MTT Grippers blocks 53
4.2.4 MTT Tending blocks 54
4.3 How block dropdowns consume WebApp configuration data 56
4.4 Side-by-side example: Wizard blocks and RAPID code 56

5 The Machine Tending Template WebApp 59

5.1 WebApp overview 59
5.1.1 Starting the WebApp 59

© Copyright 2026 ABB. All rights reserved.

5.1.2 Main screen and navigation 59
5.2 The Configuration Wizard 60
5.2.1 Opening the Configuration Wizard 60
5.2.2 Position teaching 60
5.2.3 Process parameter configuration 62
5.2.4 Machine interface configuration 66
5.2.5 Configuration validation 67
5.2.6 Saving configuration 67
5.3 Running production from the WebApp 68
5.3.1 Operation view 68
5.3.2 Production monitoring 69
5.3.3 Drawer management 69
5.4 Wizard Easy Programming 72
5.4.1 Overview 72
5.4.2 MTT Tending blocks 72
5.4.3 MTT Machine blocks 72
5.4.4 MTT Grippers blocks 73
5.4.5 MTT Cycle blocks 73
5.4.6 Modifying the production cycle 73
5.5 Gantry mode (optional) 74

6 RAPID reference 75

6.1 Core module — MTT_Wizard.sysx 76
6.1.1 Pick and Place 76
6.1.2 Drawer operations 79
6.1.3 Tray operations 83
6.1.4 Spindle operations 85
6.1.5 Gripper control 89
6.1.6 Load management 92
6.1.7 Tray position management 96
6.1.8 Offset calculation 99
6.1.9 Drawer helper functions 100
6.1.10 Cycle management 102
6.1.11 Module loading 105
6.1.12 Logging 106
6.1.13 WebApp display 107
6.1.14 Manual drawer operations 108
6.2 Station HAL — MTT_User.sysx 110
6.2.1 Home and cycle management 110
6.2.2 Drawer operations 114
6.3 Machine HAL — MTT_Machine.sysx 121
6.3.1 Machine door 121
6.3.2 Spindle/chuck 127
6.3.3 Machine cycle 130

© Copyright 2026 ABB. All rights reserved.

6.4 Gripper HAL — MTT_EndEffector.sysx 136
6.4.1 Gripper open/close 136
6.4.2 Gripper feedback 138
6.4.3 Item verification 140
6.4.4 Blow-off 143
6.5 Data types 145
6.5.1 MTT_TRAY — Tray layout definition 145
6.5.2 MTT_GRIPPER — Gripper definition 145
6.5.3 MTT_APPROACH_OFFSET — Approach/depart offset 145
6.5.4 MTT_TRAY_POSITION — Tray position index 146
6.5.5 MTT_DRAWER_STATE — Drawer state 146
6.5.6 MTT_TRAY_INFO — Runtime tray information 146
6.6 Constants and enumerations 147
6.6.1 Grip direction 147
6.6.2 Tray increment mode 147
6.6.3 Drawer state 147
6.6.4 Gripper command sets 147
6.7 Error codes 148
6.8 Persistent state variables 149

7 Program examples 150

7.1 Default production cycle 151
7.1.1 RAPID code 151
7.1.2 Step-by-step walkthrough 151
7.1.3 Cycle timing breakdown 152
7.2 Single-spindle cycle 153
7.2.1 RAPID code 153
7.2.2 Key differences from default 153
7.3 Cycle with separate load and unload drawers 155
7.3.1 RAPID code 155
7.3.2 Notes 155
7.4 Cycle with blow-off cleaning station 156
7.4.1 RAPID code 156
7.4.2 Notes 156
7.5 Cycle with standalone tray (no drawers) 157
7.5.1 RAPID code 157
7.5.2 Notes 157
7.6 Cycle with user-defined pick and place positions 158
7.6.1 RAPID code 158
7.6.2 Notes 158
7.7 Customization guidelines 159
7.7.1 Adding a new operation to the cycle 159
7.7.2 Adding custom logic between operations 159
7.7.3 Adding a third gripper 159

© Copyright 2026 ABB. All rights reserved.

7.7.4 Modifying machine behavior 159
7.7.5 Creating a program variant 159

8 AppStudio project shipped with the Add-In 161

8.1 Overview....161
8.1.1 Steps to open the package folder and locate the project....161
8.1.2 Steps to create a local copy and open in AppStudio 162
8.1.3 Steps to redeploy the WebApp to the controller 162

9 Index 165

© Copyright 2026 ABB. All rights reserved.

1.1 Overview of this manual

About this manual

This manual contains information about the application software **Machine Tending Template** (MTT).

Use this manual to:

- Understand what the Machine Tending Template is and how it supports machine tending operations.
- Install and configure the template on an ABB OmniCore controller.
- Commission the robot cell using the WebApp Configuration Wizard.
- Understand the RAPID framework, including procedures, functions, data types, and predefined data.
- Build and modify the production cycle using Wizard Easy Programming blocks or direct RAPID editing.
- Operate and monitor production from the FlexPendant WebApp.

Who should read this manual

This manual is intended for:

- **System integrators** who install, commission, and customize the template for a specific machine tending application.
- **Robot programmers** who modify or extend the RAPID program cycle and machine interface modules.
- **Operators** who use the FlexPendant WebApp to run production, manage drawers, and monitor cycle status.

Prerequisites

The reader should be familiar with:

- Industrial robot terminology and operation.
- The RAPID programming language.
- OmniCore controller system parameters and configuration.
- Basic concepts of web application interfaces.

References

Reference	Document ID
Product manual OmniCore E10	3HAC079399-001
Product manual OmniCore C30	3HAC060860-001
Product manual OmniCore C90XT	3HAC073706-001

© Copyright 2026 ABB. All rights reserved.

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	First release. Machine Tending Template 1.0.0.

© Copyright 2026 ABB. All rights reserved.

1.2 Product documentation

Product documentation for ABB robot products is organized into the following categories:

- **Product manuals** — describe the product hardware (manipulators, controllers, etc.) including installation, maintenance, and repair.
- **Technical reference manuals** — provide detailed technical information about RAPID programming, system parameters, and communication.
- **Application manuals** — describe application-specific software products such as this template. These manuals explain installation, configuration, programming, and operation.
- **Operating manuals** — describe daily operation of the robot system, including the FlexPendant, RobotStudio, and controller interfaces.

The latest versions of all manuals are available on the ABB documentation portal and through RobotStudio.

© Copyright 2026 ABB. All rights reserved.

1.3 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 rapid, hazardous movement. Even if a pattern of movement is predicted, a change in operation can be triggered by an external signal, meaning the robot starts an unpredicted movement. Therefore, it is important that all safety regulations are followed when entering safeguarded areas.

Safety regulations

Before beginning work with the robot, make sure you are familiar with the safety instructions described in:

- *Safety manual for robot — Manipulator*
- *Safety manual for robot — OmniCore controller*

Machine Tending Template — safety note

The Machine Tending Template provides process-level safeguards including:

- **Gripper feedback verification** — the template checks gripper position sensors after every pick operation to confirm the workpiece is secured (see `MTT_checkGrip` in section 5).
- **Machine door status monitoring** — door open/closed signals are verified before the robot enters or exits the machine work envelope.
- **Spindle/chuck confirmation** — chuck feedback signals are monitored during spindle open/close operations, with configurable timeouts.
- **Drawer state management** — the system tracks drawer open/closed, ready, and enabled states to prevent unsafe operations.



Warning

These process-level safeguards are NOT a substitute for: - Certified machine safety systems compliant with applicable standards (ISO 10218, ISO/TS 15066, etc.) - Formal risk assessments performed for the specific installation - Site-specific safety validation and commissioning procedures The integrator is responsible for ensuring the complete installation meets all applicable safety requirements. The Machine Tending Template is a software baseline — it does not replace proper safety engineering.

© Copyright 2026 ABB. All rights reserved.

1 Introduction

1.1 Machine Tending Template overview

The Machine Tending Template (MTT) is a complete, ready-to-deploy software solution for robot-based machine tending operations. It supports common scenarios such as loading and unloading CNC machines, injection molding machines, grinding machines, and similar equipment where a robot transfers workpieces between a storage area and a machining station.

The template is not a turnkey application. It is a **validated software baseline** that system integrators adapt to the specific hardware and process requirements of each installation. By providing tested building blocks for the most common machine tending tasks, MTT eliminates the need to write a machine tending framework from scratch and enables faster, more predictable deployments.

MTT serves two distinct audiences:

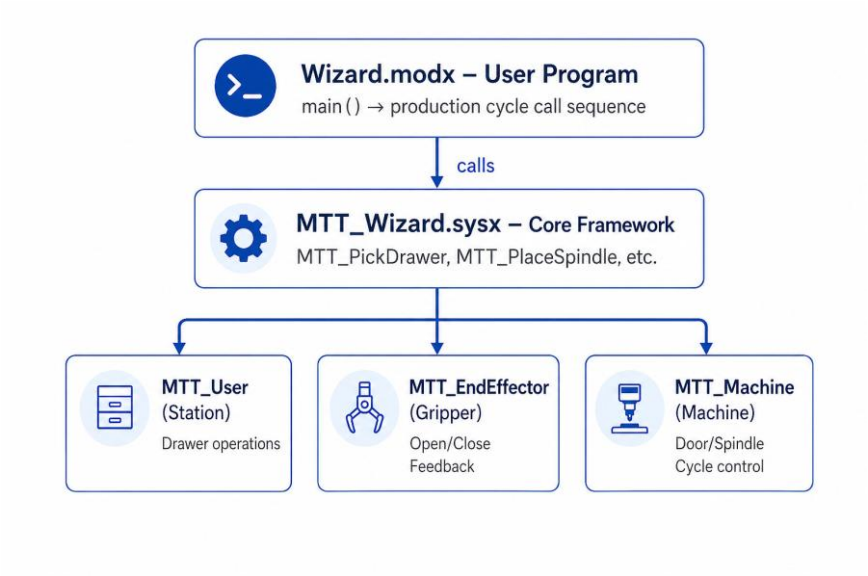
- **System integrators** — install the template, configure the hardware interfaces, teach positions, and customize the production cycle for the specific cell.
- **Operators** — use the FlexPendant WebApp to run production, manage drawers, and monitor cycle status without modifying RAPID code.

1.2 RAPID framework and Wizard Easy Programming

The RAPID framework is organized as a set of **modular system modules** that separate concerns into well-defined layers:

Module	Responsibility	Modifiable
`MTT_Wizard.sysx`	Core framework — pick/place algorithms, tray/drawer management, gripper logic, load tracking, approach offsets	No — core system module
`MTT_WizardData.modx`	Configuration data — targets, offsets, gripper definitions, tray layouts, load data	Indirect — written by WebApp
`Wizard.modx`	User program — the `main()` procedure containing the production cycle	Yes — integrators adapt the cycle
`MTT_User.sysx`	Station HAL — drawer operations, home position, cycle management (station-specific)	Swap — different file per station
`MTT_EndEffector.sysx`	Gripper HAL — tool data, gripper open/close, feedback checking (station-specific)	Swap — different file per station
`MTT_Machine.sysx`	Machine HAL — spindle/chuck, door, machine cycle control (machine-specific)	Swap — different file per machine

The framework follows a **Hardware Abstraction Layer (HAL) pattern**: the core module (`MTT_Wizard.sysx`) calls procedures defined in the HAL modules using a standard interface. Integrators swap the HAL modules to match their hardware without modifying the core logic.



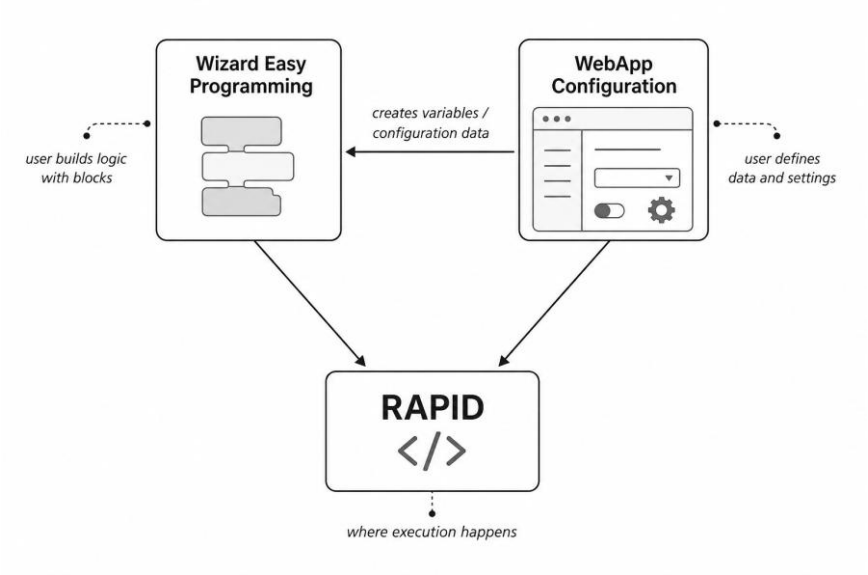
Rapid HAL diagram

Core and HAL procedures are exposed as **Wizard Easy Programming blocks** — visual blocks on the FlexPendant that map higher-level operations to RAPID procedure calls.

The wizard model can be modified by integrators to define the process logic. Operators can also adjust the model via **Wizard Easy Programming**, so they do not need to know RAPID to manipulate the production sequence. The system modules (`MTT_Wizard.sysx`, `MTT_User.sysx`, `MTT_EndEffector.sysx`, `MTT_Machine.sysx`) provide the procedures the wizard model calls to cover the specific process logic related to the particular use-case.

1.3 Relationship between WebApp, Wizard, and RAPID

The Machine Tending Template is built around three connected layers:



Relationship between WebApp, Wizard, and RAPID

- **WebApp Configuration** is used to define application data such as targets, grippers, trays, loads, and offsets.
- **Wizard Easy Programming** is used to define the machine tending logic by arranging blocks into the desired production flow.
- **RAPID** is the execution layer where the final robot behavior runs in the controller.

These layers are connected in a specific way:

- The **WebApp** creates and updates RAPID configuration data, mainly persistent variables and records stored in the template data modules.
- The **Wizard blocks** consume that configured data. The parameters selected in the blocks refer to the data previously created or maintained through the WebApp.
- The resulting block-based process flow is executed as **RAPID** at runtime.

In practice, this means:

- the **WebApp** defines the application data
- **Wizard Easy Programming** defines the application logic
- **RAPID** executes both together

This separation is a key part of the MTT design: configuration, programming, and execution are clearly linked, but each is handled in the ABB tool that best fits its purpose.

1.4 WebApp and Configuration Wizard

The Machine Tending Template includes a **WebApp** built with ABB AppStudio that runs on the FlexPendant. The WebApp provides two main operating modes:

- **Process interface** — a runtime interface with tabs for Production (start/stop, cycle time, counters), Manual Operation (gripper/machine/drawer controls), and Drawers (enable/disable, priority, ready state).
- **Configuration interface** — a guided wizard with tabs for Loads, Trays, Grippers, Approach Offsets, and Targets. The integrator uses this during commissioning to teach positions, define tray layouts, configure gripper parameters, and set approach/depart offsets. All configuration data is stored in `MTT_WizardData.modx`.
- **Gantry mode** apply program-wise offsets and TCP adjustments (tray heights, gripper TCP Z offsets) and save the result as a program variant.

The complete AppStudio project is delivered with the template, enabling integrators to customize the WebApp using standard ABB tools without external toolchains.

1.5 RobotWare compatibility

The Machine Tending Template requires:

- **OmniCore controller** with RobotWare 7.0 or later
- Optimized for collaborative robots such as GoFa and PoWa
- ABB Scalable IO module (DSQC1031) for digital I/O signals

1.6 Gripper support

The template supports **single and dual gripper configurations** — with upto two independent grippers mounted on the same end-effector. This enables simultaneous load/unload operations: pick a finished part from one spindle with Gripper 2 and place a raw part with Gripper 1, all in a single machine visit.

Each gripper is defined by an `MTT_GRIPPER` record with the following properties:

Property	Description
Tool data reference	Name of the `tooldata` variable (e.g., `MTT_tGripper1`)
Gripper command set	Which control command set to use (1 or 2)
Grip direction	Inward (0) — gripper closes to grip; Outward (1) — gripper opens to grip
TCP Z offset	Additional Z-axis offset applied to the tool center point

The gripper HAL module (`MTT_EndEffector.sysx`) implements the hardware-specific open/close procedures, position feedback functions, and item-missed checking. The template includes ready-to-use implementations for pneumatic parallel grippers with position sensors.

Additional gripper features:

- **Blow-off nozzle** — air nozzle for cleaning the spindle or part before/after placement.
- **Gripper feedback validation** — after every pick, the template checks gripper position sensors and raises `ERR_MTT_ITEM_MISSED` if the part was not secured.

1.7 Machine interface

The template uses a **machine-agnostic HAL** (`MTT_Machine.sysx`) that isolates all machine-dependent behavior into a single swappable module. This design supports any machine type — CNC, injection molding, grinding, testing equipment — by implementing a standard set of procedures:

Procedure	Purpose
`MTT_OpenMachineDoor` / `MTT_CloseMachineDoor`	Robot physically opens/closes the machine door using motion + magnet
`MTT_waitMachine`	Moves robot to wait position, waits for machine ready signal (`DI3`)
`MTT_startMachine`	Pulses the machine cycle start signal (`DO5`)
`MTT_EnterMachine` / `MTT_ExitMachine`	Moves robot into/out of the machine work envelope
`MTT_OpenSpindle1` / `MTT_CloseSpindle1`	Opens/closes spindle 1 chuck with feedback verification
`MTT_OpenSpindle2` / `MTT_CloseSpindle2`	Opens/closes spindle 2 chuck with feedback verification
`MTT_WaitForMachineDoorOpen` / `MTT_WaitForMachineDoorClosed`	Waits for door position feedback with timeout

One machine variants is included:

Variant	Description
`MachineX`	Generic real machine variant — standard pneumatic door and spindle control with position sensors

Integrators create new machine variants by implementing the same set of procedures in a new `MTT_Machine.sysx` module.

1.8 Station variants

The station HAL module (`MTT_User.sysx`) handles everything specific to the physical cell layout: drawer operations, home position, cycle management, and cell-specific safety features. Two station variants are included:

1.8.1 Omnivance



Omnivance Station

- 6 drawers with magnetic attach mechanism
- Blow-off cleaning station
- ABB Safe Scalable I/O (DSQC1042 + DSQC1031) with CIP Safety
- Workobjects: `MTT_WobjDrawer1` through `MTT_WobjDrawer6`

1.8.2 ProFeederS 5 Trays



MTT Profeeder S Station

- 5 drawers with mechanical grip mechanism
- Workobjects: `MTT_WobjDrawer1` through `MTT_WobjDrawer5`

Each station variant provides drawer open/close positions, drawer workobjects, and station-specific gripper configurations. The station module is swapped during Add-In installation based on the selected hardware configuration.

1.9 Production cycle overview

The default production cycle in `Wizard.modx` follows this sequence:

```
MTT_Home                                          → Move robot to
home position
WHILE TRUE DO
    MTT_StartCycle                                → Reset timer,
prepare load tracking
    MTT_PickDrawer Gripper1, item1_raw            → Pick raw part
from unload drawer
```

```
    MTT_waitMachine                                 → Wait at wait
position for machine ready
    MTT_OpenMachineDoor                             → Robot opens the
machine door
    MTT_EnterMachine                                → Move into machine
work envelope
    MTT_PlaceSpindle Gripper1, 1, item1_raw,
        \BlowOff \releaseBefore                     → Place raw part
into spindle 1
    MTT_PickSpindle Gripper2, 2, item1_finished,
        \BlowOff                                    → Pick finished
part from spindle 2
    MTT_ExitMachine                                 → Move out of
machine work envelope
    MTT_CloseMachineDoor                            → Robot closes the
machine door
    MTT_startMachine                                → Pulse machine
cycle start signal
    MTT_PlaceDrawer Gripper2, item1_finished        → Place finished
part into load drawer
    MTT_EndCycle                                    → Stop timer,
increment counter, check last cycle
ENDWHILE
```

This default cycle demonstrates a **dual-spindle, dual-gripper** workflow: the robot loads a raw part into spindle 1 while picking a finished part from spindle 2 in a single machine visit. The cycle is fully configurable — integrators rearrange, add, or remove steps using Wizard Easy Programming blocks or by editing the RAPID code directly.

Key cycle features:

- **Automatic drawer selection** — the framework selects the next available drawer based on priority and ready state.
- **Tray position tracking** — the framework automatically calculates the next grid position within the tray after each pick or place.
- **Cycle time measurement** — `MTT_StartCycle` / `MTT_EndCycle` bracket the cycle for timing.
- **Production counter** — incremented at each `MTT_EndCycle`.
- **Last cycle handling** — when `MTT_LastCycle` is set (from the WebApp), the robot completes the current cycle, closes all open drawers, and moves home.

1.10 Customization with ABB tools

The Machine Tending Template is customizable using standard ABB tools — no external toolchains, IDEs, or licenses are required:

Tool	What it customizes

Wizard Easy Programming (FlexPendant)	Production cycle — drag and rearrange blocks to change the pick/place sequence, add machine steps, modify gripper operations
WebApp Configuration Wizard (FlexPendant)	Positions, offsets, tray layouts, gripper definitions, load data — all stored in `MTT_WizardData.modx`
ABB AppStudio	WebApp interface — modify screens, add controls, change layout. Full project delivered with template
Skill Creator	Custom Wizard blocks — create new graphical blocks that map to custom RAPID procedures
RobotStudio	Offline programming, signal configuration, system parameter editing, virtual controller testing
RAPID text editor	Direct code editing of `Wizard.modx` and HAL modules for advanced customization

2 Installation

2.1 System requirements

Standard options

The Machine Tending Template requires the following minimum configuration:

- **Controller**: ABB OmniCore controller
- **RobotWare**: Version 7.0 or later
- **Robot**: GoFa, PoWa, or any ABB Omnicore compatible models
- **I/O module**: ABB Scalable IO (DSQC1031) for digital input/output signals

Additional options

Depending on the cell configuration, the following additional options may be required:

- **Additional I/O units** — for machine interface signals that exceed the base 16 DI / 16 DO capacity.
- **Safety module** — DSQC1042 with CIP Safety for Omnivance stations.
- **Gripper-specific I/O** — Schunk MTB DG 80 dual gripper requires EtherNet/IP device configuration.
- **Multitasking option** — if background tasks are needed for auxiliary processes.

Limitations

- The template is a software baseline, not a turnkey solution. Each installation requires position teaching, hardware wiring, and process validation.
- Performance (cycle time, reach, payload) depends on the robot model, gripper design, and machine type.
- The template supports up to 6 drawers (Omnivance) or 5 drawers (ProFeederS). Custom station variants may support different counts.
- The template supports up to 2 spindles. Machines with more spindles require a custom `MTT_Machine.sysx` variant.

2.2 Hardware installation

2.2.1 Gripper design

The template assumes a dual gripper end-effector with the following characteristics:

- **Two independent grippers** (Gripper 1 and Gripper 2), each with its own pneumatic valve and position sensor.

- **Position feedback sensors** — each gripper must provide at least two digital signals: one indicating the closed position and one indicating the open position.
- **Pneumatic control** — each gripper is actuated by a single digital output (one for close/open direction selection) plus a pneumatic release signal shared between both grippers.
- **Tool data** — each gripper has its own TCP (Tool Center Point) definition. A separate tool data definition is used for drawer open/close operations (`MTT_tOpenClose`) and blow-off operations (`MTT_tBlowOff`).

The gripper wiring must connect to the ABB Scalable IO module as follows:

Signal	I/O	Description
Gripper 1 closed	DI9	Position sensor A — gripper 1 in closed position
Gripper 1 open	DI10	Position sensor B — gripper 1 in open position
Gripper 2 closed	DI11	Position sensor A — gripper 2 in closed position
Gripper 2 open	DI12	Position sensor B — gripper 2 in open position
Close Gripper 1	DO16	Pneumatic valve command for gripper 1
Close Gripper 2	DO15	Pneumatic valve command for gripper 2
Enable gripper (pneumatic release)	DO14	Shared pneumatic release signal
Activate magnet	DO12	Drawer magnet for Omnivance drawer operations

2.2.2 Machine interface wiring

The machine interface requires the following signal connections:

Signal	I/O	Description
Machine ready	DI3	Machine has completed its cycle and is ready for loading/unloading
Machine door open	DI4	Machine door position sensor — door is open
Machine door closed	DI5	Machine door position sensor — door is closed
Chuck spindle 1 closed	DI6	Spindle 1 chuck position sensor — closed/clamped
Chuck spindle 2 closed	DI7	Spindle 2 chuck position sensor — closed/clamped
Open chuck spindle 1	DO1	Command to open spindle 1 chuck
Close chuck spindle 1	DO2	Command to close spindle 1 chuck
Open chuck spindle 2	DO3	Command to open spindle 2 chuck
Close chuck spindle 2	DO4	Command to close spindle 2 chuck
Start machine cycle	DO5	Pulse output to trigger the machine cycle

For detailed signal mapping and hardware integration options (cross-connection vs. direct replacement), see section 3.4.

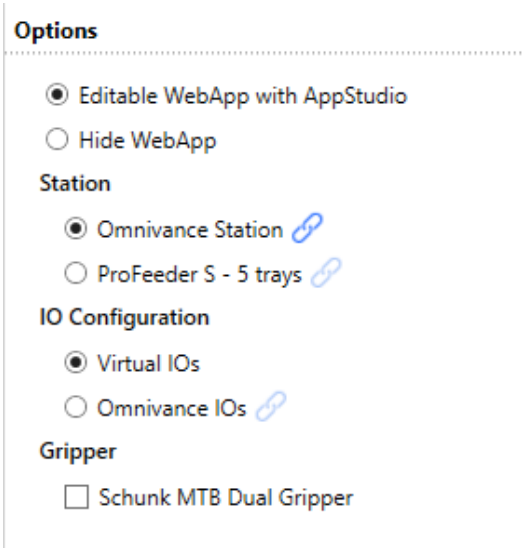
2.3 Software installation

2.3.1 Add-In installation

The Machine Tending Template is distributed as a RobotWare Add-In package (`.rpk`). To install:

1. Open **RobotStudio** and connect to the target controller (or create a new virtual controller).
2. Navigate to the **Add-Ins** tab.
3. Locate the Machine Tending Template Add-In and install it.
4. During installation, the Add-In prompts for the following selections:

Selection	Options	Description
Station	Omnivance, ProFeederS 5 Trays	Determines which `MTT_User.sysx` and `MTT_EndEffector.sysx` modules are installed
I/O configuration	Omnivance, Virtual	Determines which EIO configuration is loaded (physical or virtual signals)
Gripper	Schunk MTB DG 80, None	Optionally loads gripper-specific EIO configuration
WebApp	Expose (editable), Hide (read-only)	Determines whether the AppStudio project is copied to HOME for customization



Add-In Options

5. The Add-In automatically detects the robot type (GoFa, PoWa) and loads the corresponding motion configuration (MOC).
6. **Restart the controller** to complete the installation.

2.3.2 Building a robot controller

When the Add-In is included in a controller build, the installation script performs the following steps:

1. Defines the `MTT` environment variable pointing to the Add-In boot path.
2. Creates the `HOME/MTT/` folder structure for language files and machine backups.
3. Copies the **Wizard block library** files (`.coblox`) to `HOME/BlockLibrary/`.
4. Copies the selected **station modules** (`MTT_User.sysx`, `MTT_EndEffector.sysx`) to the Add-In RAPID folder.
5. Copies the **default RAPID modules** (`Wizard.modx`, `MTT_WizardData.modx`, `MTT_Machine.sysx`) to the Add-In RAPID folder.
6. Loads the **I/O configuration** (`EIO.cfg`) matching the selected I/O option.
7. Loads the **system configuration** (`MTT_SYS.cfg`) which defines module loading order and startup hooks.
8. Copies the **WebApp** to `HOME/WebApps/MachineTending/`.
9. Copies a **demo program** (`demoOmnivance` or `demoProFeederS`) to `HOME/Programs/`.

2.3.3 HOME folder contents

After installation, the HOME directory contains:

```
HOME/
├── BlockLibrary/
│   ├── MTT Cycle.coblox          Wizard block definitions (cycle
control)
│   ├── MTT Grippers.coblox       Wizard block definitions (gripper
operations)
│   ├── MTT Machine.coblox        Wizard block definitions (machine
operations)
│   └── MTT Tending.coblox        Wizard block definitions
(pick/place operations)
├── MTT/
│   ├── Languages/                 Multi-language resource files (de,
en, es, fr, it, ja, nl, zh)
│   ├── Machines/
│   │   └── MachineX/             Backup of default machine module
│   └── AppStudio/                 Full AppStudio project (if
"expose" selected)
├── Programs/
│   └── demoOmnivance/             Demo program for the selected
station
│       └── demoOmnivance.pgf
└── WebApps/
    └── MachineTending/            FlexPendant WebApp files
```

The Add-In RAPID folder (`OPEN.ABB.MTT_HOME/RAPID/`) contains the active RAPID modules:

```
OPEN.ABB.MTT_HOME/RAPID/
├── Wizard.modx                    User program (production cycle)
├── MTT_WizardData.modx            Configuration data module
├── MTT_User.sysx                  Station HAL module (Omnivance or
ProFeederS)
├── MTT_EndEffector.sysx           Gripper HAL module
└── MTT_Machine.sysx               Machine HAL module (MachineX
default)
```

2.3.4 System parameters

The system configuration file `MTT_SYS.cfg` defines:

Startup hooks (executed on controller RESET):

Hook	Purpose
`MTT_loadDataModule`	Loads the `MTT_WizardData.modx` module from the HOME directory into RAPID memory
`MTT_resetTrayStates`	Resets all tray position counters to [1,1,1] for a clean start

Module loading order (Task `T_ROB1`):

Order	Module	Source
1	`MTT_Wizard.sysx`	Add-In boot path (`MTT:/RAPID/`) — core framework
2	`MTT_User.sysx`	Add-In HOME (`OPEN.ABB.MTT_HOME:/RAPID/`) — station HAL
3	`MTT_EndEffector.sysx`	Add-In HOME — gripper HAL
4	`MTT_Machine.sysx`	Add-In HOME — machine HAL
5	`MTT_WizardData.modx`	Add-In HOME — configuration data
6	`Wizard.modx`	Add-In HOME — user program

The core framework module (`MTT_Wizard.sysx`) is loaded from the Add-In boot path and is not modifiable. All other modules are loaded from the Add-In HOME path and can be modified by the integrator.

2.3.5 RAPID program and system modules

The following table summarizes which modules are safe to modify and which should not be changed:

Module	Type	Purpose	Safe to modify
`Wizard.modx`	Program	Production cycle (`main()`)	**Yes** — this is the integrator's program
`MTT_WizardData.modx`	Program	Configuration data (targets, offsets, trays, grippers, loads)	**Indirect** — written by the WebApp Configuration Wizard

`MTT_User.sysx`	System	Station HAL (drawer operations, home, cycle management)	**Swap** — replace with a different station variant
`MTT_EndEffector.sysx`	System	Gripper HAL (tool data, open/close, feedback)	**Swap** — replace with a different gripper implementation
`MTT_Machine.sysx`	System	Machine HAL (door, spindle, cycle control)	**Swap** — replace with a different machine variant
`MTT_Wizard.sysx`	System	Core framework (pick/place logic, tray management)	**No** — do not modify

2.3.6 WebApp

If the **Expose** option was selected during installation, the WebApp project is copied to `HOME/WebApps/MachineTending/`, making it accessible in ABB AppStudio. Integrators can open the project, customize the interface, and redeploy it to the controller. If **Hide** was selected instead, the WebApp is registered on the controller but not exposed to the file system — it will run normally, but cannot be modified or redeployed.

2.3.7 Uninstallation

To uninstall the Machine Tending Template:

1. Remove the Add-In from the controller through RobotStudio or the controller management interface.
2. The uninstall script automatically removes:
3. `HOME/MTT/` — language files, machine backups, and AppStudio project
4. `HOME/BlockLibrary/` — Wizard block definition files
5. `HOME/WebApps/MachineTending/` — FlexPendant WebApp files
6. Restart the controller to complete the uninstallation.



Note

The uninstall script does not remove `HOME/Programs/` to preserve any user-modified programs. Delete demo programs manually if no longer needed.

3 System configuration

This chapter describes how to configure the Machine Tending Template for a specific installation. Configuration involves defining the machine interface, drawer/tray layouts, gripper parameters, I/O signal mappings, safety signals, and load data.

3.1 Machine definitions

3.1.1 Machine HAL module (MTT_Machine)

The `MTT_Machine.sysx` module isolates all machine-dependent behavior — door operations, spindle/chuck control, machine cycle commands, and machine-specific positions — into a single system module. Integrators swap this module to support different machine types without modifying the core framework or the production cycle.

Every machine variant must implement the following standard interface:

Procedure / Function	Purpose
`MTT_OpenMachineDoor`	Open the machine door (robot motion + actuator control)
`MTT_CloseMachineDoor`	Close the machine door
`MTT_WaitMachine`	Move to wait position, wait for `diMTT_MachineReady` signal
`MTT_StartMachine`	Pulse `doMTT_StartMachineCycle` to trigger the machine
`MTT_FeedHoldMachine`	Trigger feed hold (optional, placeholder in default variants)
`MTT_EnterMachine`	Move into the machine work envelope
`MTT_ExitMachine`	Move out of the machine work envelope
`MTT_OpenSpindle1` / `MTT_OpenSpindle2`	Open spindle chuck, wait for feedback
`MTT_CloseSpindle1` / `MTT_CloseSpindle2`	Close spindle chuck, wait for feedback
`MTT_Spindle1Closed` / `MTT_Spindle2Closed`	Return TRUE if spindle chuck is closed
`MTT_MachineDoorOpen` / `MTT_MachineDoorClosed`	Return TRUE if door is in the named position
`MTT_MachineReady`	Return TRUE if the machine is ready for loading/unloading
`MTT_WaitForMachineDoorOpen` / `MTT_WaitForMachineDoorClosed`	Wait for door feedback with timeout

3.1.2 Machine door configuration

Machine door positions

The machine door is controlled by two robot positions:

Variable	Description
`MTT_pDoorOpen`	Position where the robot holds the door in the fully open state
`MTT_pDoorClosed`	Position where the robot holds the door in the fully closed state

In the `MachineX` variant, the robot physically opens and closes the door using:

1. An approach move to 100 mm behind the door closed position.
2. A linear move to the door closed position.
3. Magnet activation (`doMTT_ActivateMagnet`) to grip the door.
4. A linear move from the closed position to the open position (or vice versa).
5. Magnet deactivation to release the door.
6. A retract move to clear the door area.

Motion supervision is temporarily raised to 300% and acceleration is reduced to 8% during door operations to handle the mechanical resistance of the door mechanism.

To teach the door positions:

1. Jog the robot to the position where the door is fully closed, with the gripper tool aligned to the door handle/grip surface. Record this as `MTT_pDoorClosed`.
2. Jog the robot (while gripping the door) to the position where the door is fully open. Record this as `MTT_pDoorOpen`.
3. Test the full open/close motion at reduced speed using the Manual Operation tab in the WebApp.

Machine door signals

Signal	Type	Description
`diMTT_MachineDoorOpen` (DI4)	Digital input	Machine door position sensor — door is open
`diMTT_MachineDoorClosed` (DI5)	Digital input	Machine door position sensor — door is closed

The `MTT_OpenMachineDoor` and `MTT_CloseMachineDoor` procedures wait for the corresponding feedback signal after completing the motion, with a 3-second timeout. If the feedback is not received within the timeout, the system logs an error message and stops the program (operator intervention required).

3.1.3 Spindle/chuck configuration

Spindle positions

Each spindle has a pick position and a place position:

Variable	Description
`pPickSpindle1`	Position to pick a finished part from spindle 1
`pPlaceSpindle1`	Position to place a raw part into spindle 1
`pPickSpindle2`	Position to pick a finished part from spindle 2
`pPlaceSpindle2`	Position to place a raw part into spindle 2

These positions are defined in `MTT_WizardData.modx` and are configurable through the WebApp Targets tab.

These targets define where the robot picks finished parts from the machine and where it places raw parts into the machine. For the Targets tab workflow and the meaning of **Teach** / **Go To**, see the WebApp chapter section on the Targets tab.

Spindle approach and depart offsets

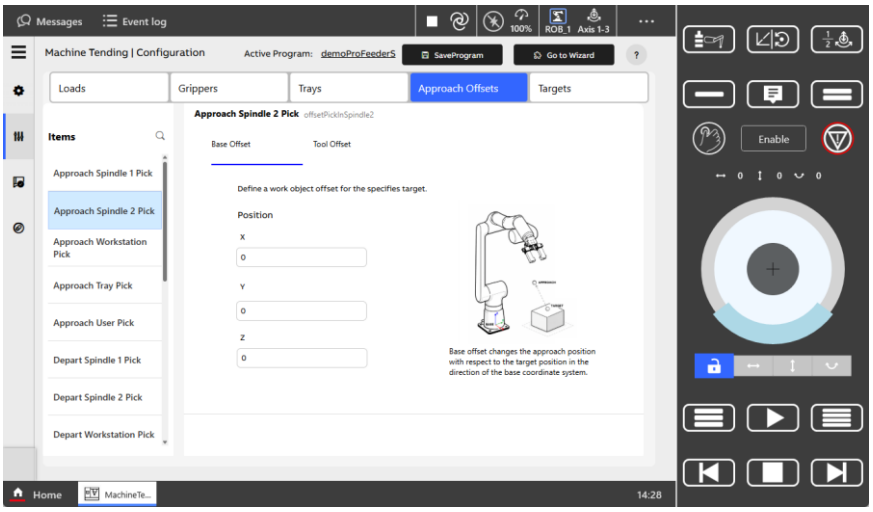
Each spindle operation (pick/place, in/out) has a configurable approach offset defined as an `MTT_APPROACH_OFFSET` record:

Offset variable	Default value	Description
`offsetPickInSpindle1`	[0,0,0,0,0,−50]	Approach offset when picking from spindle 1
`offsetPickOutSpindle1`	[0,0,0,0,0,−50]	Depart offset after picking from spindle 1
`offsetPlaceInSpindle1`	[0,0,0,0,0,−50]	Approach offset when placing to spindle 1
`offsetPlaceOutSpindle1`	[0,0,0,0,0,−50]	Depart offset after placing to spindle 1
`offsetPickInSpindle2`	[0,0,0,0,0,−50]	Approach offset when picking from spindle 2
`offsetPickOutSpindle2`	[0,0,0,0,0,−50]	Depart offset after picking from spindle 2
`offsetPlaceInSpindle2`	[0,0,0,0,0,−50]	Approach offset when placing to spindle 2
`offsetPlaceOutSpindle2`	[0,0,0,0,0,−50]	Depart offset after placing to spindle 2

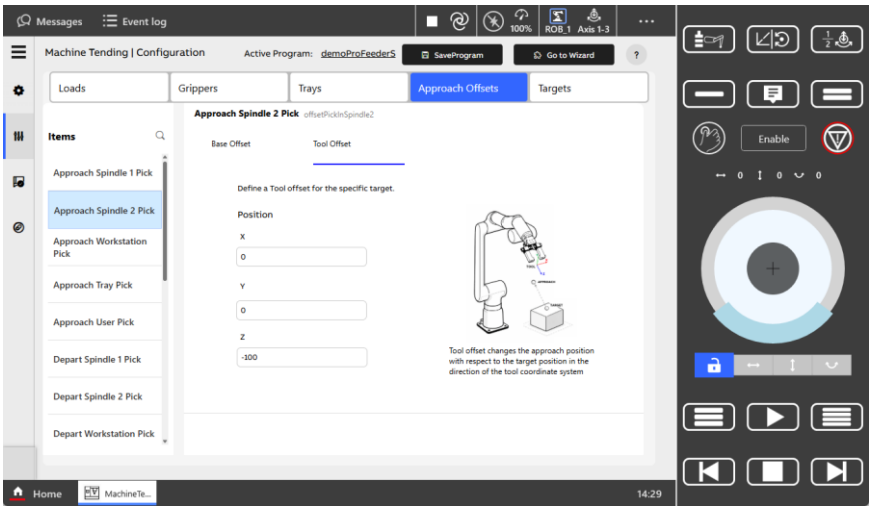
The `MTT_APPROACH_OFFSET` record has two groups of three values:

Field	Description
`wobjOffsx`, `wobjOffsy`, `wobjOffsz`	Offset in the work object coordinate system (base offset)
`toolOffsx`, `toolOffsy`, `toolOffsz`	Offset in the tool coordinate system (tool offset)

The default spindle offsets use a tool Z offset of −50 mm, meaning the robot approaches and departs 50 mm along the tool Z-axis (typically straight back from the spindle face).



Approach Offsets — Base Offset



Approach Offsets — Tool Offset

Spindle signals

Signal	Type	Description
`doMTT_OpenChuckSpindle1` (DO1)	Digital output	Command to open spindle 1 chuck
`doMTT_CloseChuckSpindle1` (DO2)	Digital output	Command to close spindle 1 chuck
`doMTT_OpenChuckSpindle2` (DO3)	Digital output	Command to open spindle 2 chuck
`doMTT_CloseChuckSpindle2` (DO4)	Digital output	Command to close spindle 2 chuck
`diMTT_ChuckSpindle1Closed` (DI6)	Digital input	Spindle 1 chuck position sensor — closed
`diMTT_ChuckSpindle2Closed` (DI7)	Digital input	Spindle 2 chuck position sensor — closed

Spindle operations use a `PulseDO` for the open/close command, then wait for the feedback signal with a 3-second timeout (`WaitUntil` with `\MaxTime:=3`). If the

spindle does not reach the expected position within the timeout, the system logs a message and stops.

3.1.4 Machine cycle control

Machine ready signal

Signal	Type	Description
diMTT_MachineReady (DI3)	Digital input	Machine has completed its cycle and is ready for loading/unloading

The MTT_WaitMachine procedure moves the robot to the MTT_pWaitMachine position (a safe location near the machine) and waits until diMTT_MachineReady becomes active. In the MachineX variant, the procedure also monitors diMTT_OptionalGoHome (DI15), allowing the operator to send the robot home during the machine wait.

Machine cycle start signal

Signal	Type	Description
doMTT_StartMachineCycle (DO5)	Digital output	Pulse output to trigger the machine cycle

The MTT_StartMachine procedure uses PulseDO to send a single pulse to the machine.

Feed hold signal

Signal	Type	Description
doMTT_FeedHold (DO6)	Digital output	Pause machine operation

The MTT_FeedHoldMachine procedure is a placeholder in the current machine variants. Integrators implement this for machines that support mid-cycle pausing.

3.1.5 Machine wait position

Variable	Description
MTT_pWaitMachine	Safe position near the machine where the robot waits for the machine ready signal

This position should be outside the machine work envelope but close enough to minimize travel time when the machine signals ready.

3.1.6 Machine enter and exit positions

Variable	Description
MTT_pInMachine	Position inside the machine work area, used as an intermediate waypoint before moving to spindle positions

The `MTT_EnterMachine` procedure first calls `MTT_WaitMachine` (ensuring the machine is ready), then moves to `MTT_pInMachine`. The `MTT_ExitMachine` procedure moves to `MTT_pInMachine` and then calls `MTT_WaitMachine`.

3.1.7 Machine variants

The Machine Tending Template provides one example machine, but additional machine variants can be added as required.

MachineX

The generic real machine variant provides:

- Physical door open/close with magnet attach/detach mechanism
- Pneumatic spindle/chuck control with position sensors
- Machine cycle start via pulse output
- Machine ready signal monitoring with optional "go home" during wait
- 3-second timeouts on all spindle operations
- 5-second timeouts on door operations

This is the default variant installed by the Add-In.

3.2 Workstation and tray definitions

3.2.1 Station HAL module (MTT_User)

The `MTT_User.sysx` module contains all station-specific behavior: worstation open/close operations, home position, cycle start/end management, and station-specific features like blow-off cleaning. This module is swapped when changing the station variant.

The module implements the following standard interface:

Procedure	Purpose
`MTT_Home`	Move robot to home joint position (`MTT_jHome`)
`MTT_InitCycle`	Initialize production, reset load tracking
`MTT_StartCycle`	Start cycle timer, reset load to `load0`
`MTT_EndCycle`	Stop timer, increment production counter, handle last cycle
`MTT_OpenDrawer(worstation)`	Open the specified worstation number
`MTT_CloseDrawer(worstation)`	Close the specified worstation number
`MTT_CloseAnyOpenDrawer`	Close the first open worstation found
`MTT_NoDrawerForLoad`	Handle no-worstation-available for loading (log warning, stop)
`MTT_NoDrawerForUnload`	Handle no-worstation-available for unloading (log warning, stop)

3.2.2 Workstation workobjects

Each worstation has a dedicated workobject that defines the coordinate system origin for tray grid calculations:

Variable	Station
`MTT_WobjDrawer1` through `MTT_WobjDrawer6`	Omnivance (6 drawers)
`MTT_WobjDrawer1` through `MTT_WobjDrawer5`	ProFeederS (5 drawers)

The workobject defines where the tray grid origin is located when the worstation is in the open position. All tray positions within the worstation are calculated relative to this workobject.

To calibrate worstation workobjects:

1. Open the worstation fully.
2. Jog the robot to the corner of the tray that represents position [1,1,1].
3. Record the workobject frame using 3-point calibration or direct coordinate entry.

3.2.3 Workstation open and close positions

Each worstation has a pair of robot positions for physical open/close operations:

Variables	Description
`MTT_pOpendrawer1` through `MTT_pOpendrawer6`	Positions where the robot holds each worstation in the fully open state
`MTT_pClosedrawer1` through `MTT_pClosedrawer6`	Positions where the robot holds each worstation in the fully closed state

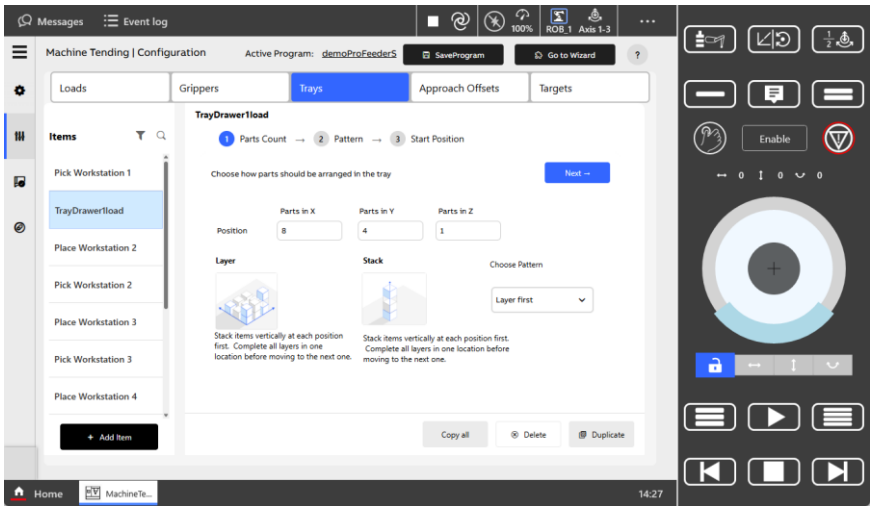
In the Omnivance variant, worstation operations use a magnetic attach mechanism:

1. Robot approaches the worstation closed position.
2. Magnet activates to grip the worstation handle.
3. Robot moves linearly from closed to open position (or vice versa).
4. Magnet releases the worstation.
5. Robot retracts to a safe position.

Motion supervision is temporarily raised and acceleration is reduced during worstation operations to handle mechanical resistance.

3.2.4 Tray layout configuration

Each worstation has separate tray definitions for load (placing finished parts) and unload (picking raw parts) operations. Tray layouts are defined in `MTT_WizardData.modx` and configured through the WebApp Trays tab.



Trays Configuration — Parts Count

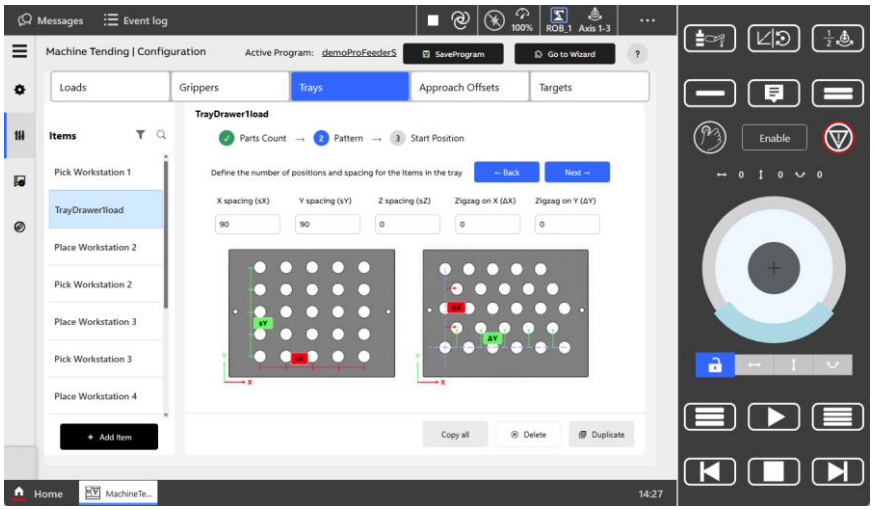
Tray grid dimensions (rows, columns, layers)

Field	Description
`numRowsX`	Number of positions in the X direction (rows)
`numColumnsY`	Number of positions in the Y direction (columns)
`numLayersZ`	Number of positions in the Z direction (layers/stacks)

Example: a 2×4×1 tray has 2 rows, 4 columns, and 1 layer = 8 total positions.

Tray spacing

Field	Description
`spacingX`	Distance in mm between row positions (X direction)
`spacingY`	Distance in mm between column positions (Y direction)
`spacingZ`	Distance in mm between layer positions (Z direction)



Trays Configuration — Pattern and Spacing

Zigzag offsets

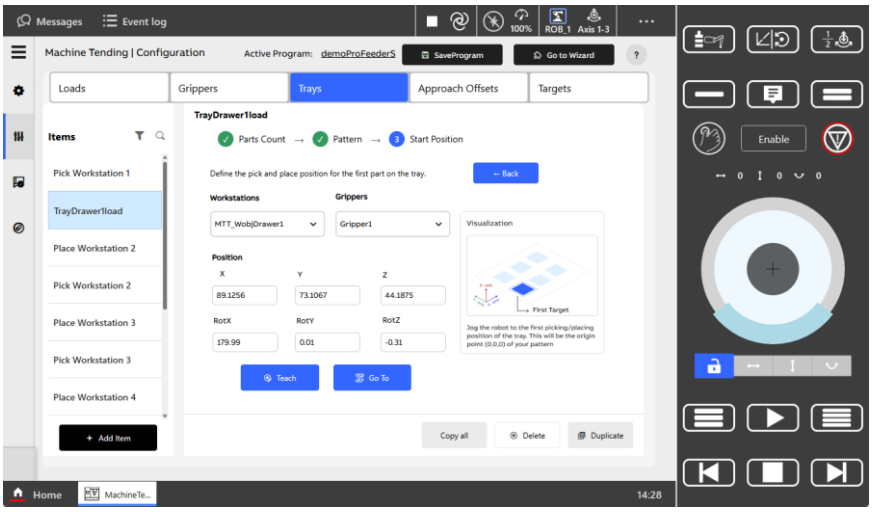
Field	Description
`offsetXEven`	Additional X offset applied to even-numbered rows
`offsetYEven`	Additional Y offset applied to even-numbered columns

These offsets support staggered tray layouts where even rows or columns are shifted relative to odd rows or columns.

Tray first target (origin position)

Field	Description
`firstTarget`	A `robtarget` defining the position and orientation of the first tray slot [1,1,1]

This position is the origin of the tray grid. All other positions are calculated by adding the grid spacing to this origin. The first target includes full position (X, Y, Z) and orientation (quaternion) data.



Trays Configuration — Start Position

To teach the first target:

1. Select the tray in the WebApp Trays tab (Step 3 — Start Position).
2. Select the appropriate work object and gripper.
3. Jog the robot to the first pick/place position in the tray.
4. Press the **Teach** button.
5. Use the **Go To** button to verify.

3.2.5 Tray increment mode

The increment mode determines the order in which positions within the tray grid are visited:

Layer-first mode

Constant: `MTT_TRAY_LAYER_FIRST` = 0

Iterates across the grid layer by layer: completes all positions in a row (X), then moves to the next column (Y), then moves to the next layer (Z) after all X×Y positions are filled.

Iteration order: X → Y → Z

This mode is typical for flat trays where parts are arranged in a single layer.

Stack-first mode

Constant: `MTT_TRAY_STACK_FIRST` = 1

Stacks parts vertically at each position before moving to the next position: completes all layers (Z) at position [1,1], then moves to [2,1], and so on.

Iteration order: Z → X → Y

This mode is used when parts are stacked on top of each other at each grid position.

3.2.6 Tray usage mode (load and unload)

Each worstation has separate tray definitions for two usage modes:

Tray variable pattern	Usage	Description
`TrayDrawer<N>Unload`	Unload (pick raw parts)	Robot picks raw parts from this tray
`TrayDrawer<N>Load`	Load (place finished parts)	Robot places finished parts into this tray

The load and unload trays for the same worstation can have different grid dimensions, spacing, and first target positions. This allows the same worstation to have one tray layout for raw parts and a different layout for finished parts.

3.2.7 Workstation approach and depart offsets

Offset variable	Default value	Description
`offsetPickInUnloadDrawer`	[0,0,200,0,0,0]	Approach offset when picking from an unload worstation
`offsetPickOutUnloadDrawer`	[0,0,200,0,0,0]	Depart offset after picking from an unload worstation
`offsetPlaceInLoadDrawer`	[0,0,200,0,0,0]	Approach offset when placing to a load worstation
`offsetPlaceOutLoadDrawer`	[0,0,200,0,0,0]	Depart offset after placing to a load worstation

The default worstation offsets use a work object Z offset of 200 mm, meaning the robot approaches and departs 200 mm above the tray surface in the work object Z direction.

3.2.8 Workstation state management

The framework tracks the state of each worstation in the `MTT_DrawersState{6}` persistent array. Each element is an `MTT_DRAWER_STATE` record.

Workstation open/closed state

Constant	Value	Description
`MTT_DRAWER_OPEN`	1	Workstation is in the open position
`MTT_DRAWER_CLOSED`	0	Workstation is in the closed position

Updated by `MTT_OpenDrawer` and `MTT_CloseDrawer` after successful motion.

Workstation ready state

Constant	Value	Description
`MTT_DRAWER_READY`	1	Workstation is ready (tray present and available)

MTT_DRAWER_NOT_READY	0	Workstation is not ready

The ready state can be toggled from the WebApp Drawers tab.

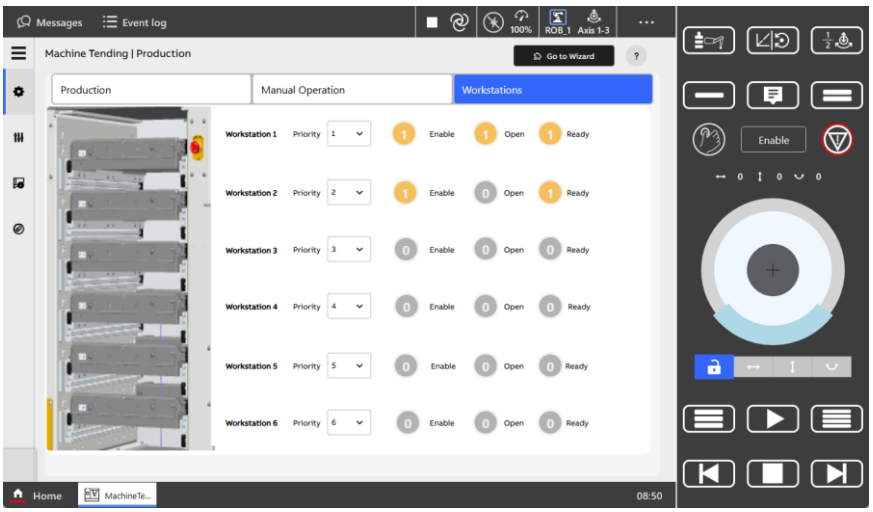
Workstation enabled/disabled state

Constant	Value	Description
MTT_DRAWER_ENABLED	0	Workstation is active and can be selected for operations
MTT_DRAWER_DISABLED	1	Workstation is disabled and will be skipped

Disabled drawers are excluded from the automatic worstation selection algorithm.

Workstation priority

Each worstation has a numeric priority value. Lower numbers indicate higher priority. When the framework selects the next worstation for a load or unload operation, it chooses the enabled, ready worstation with the lowest priority number.



Workstation Management

3.2.9 Station variants

Omnivance

- 6 drawers with magnetic attach mechanism
- Blow-off cleaning station with dedicated positions (`BlowOffApproach`, `BlowOffPosition`)
- ABB Safe Scalable I/O (DSQC1042 + DSQC1031) with CIP Safety integration
- Workobjects: `MTT_WobjDrawer1` through `MTT_WobjDrawer6`
- Drawer sensor signal: `DI8` (sensor drawer active)
- Magnet control: `DO12` (activate magnet)

- Nozzle control: `DO13` (switch off nozzle)

ProFeederS 5 Trays

- 5 drawers with mechanical grip mechanism
- Workobjects: `MTT_WobjDrawer1` through `MTT_WobjDrawer5`
- Adapted drawer open/close motion sequences for mechanical grippers

3.3 Gripper configuration

3.3.1 Gripper HAL module (MTT_EndEffector)

The `MTT_EndEffector.sysx` module provides all gripper-specific tooldata definitions and control procedures. This module is swapped together with `MTT_User.sysx` when changing the station variant.

3.3.2 Dual gripper support

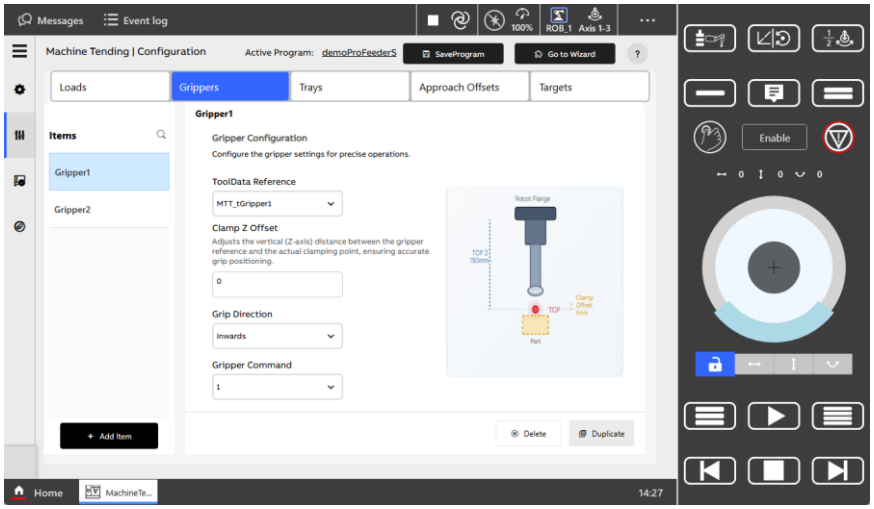
The template supports two independent grippers (Gripper 1 and Gripper 2) mounted on the same end-effector. Each gripper has independent:

- Tool data (TCP position and orientation)
- Control signals (separate digital output for each gripper)
- Position feedback (separate open/closed sensors for each gripper)
- Grip direction configuration (inward or outward)

This enables simultaneous load/unload: pick a finished part with Gripper 2 while placing a raw part with Gripper 1, all in a single machine visit.

3.3.3 Gripper definitions

Gripper definitions are stored in `MTT_WizardData.modx` as `MTT_GRIPPER` records and configured through the WebApp Grippers tab.



Grippers Configuration

Tool data reference

The `tooldataReference` field contains the name of the `tooldata` variable used by this gripper (e.g., `"MTT_tGripper1"`). The framework resolves this string reference to the actual tooldata at runtime.

Gripper command set

The `gripperCmdSet` field selects which set of hardware control procedures to use:

Value	Constant	Procedures called
1	`MTT_GRIP_CMD_SET_1`	`MTT_OpenGripper1`, `MTT_CloseGripper1`, `MTT_CheckItemInwardGripper1`, `MTT_CheckItemOutwardGripper1`
2	`MTT_GRIP_CMD_SET_2`	`MTT_OpenGripper2`, `MTT_CloseGripper2`, `MTT_CheckItemInwardGripper2`, `MTT_CheckItemOutwardGripper2`

The framework uses `CallByVar` to call the appropriate procedure based on the command set number.

Grip direction (inward / outward)

Value	Constant	Description
0	`MTT_GRIP_INWARD`	Gripper closes to grip the part (parallel jaw gripper)
1	`MTT_GRIP_OUTWARD`	Gripper opens to grip the part (expanding mandrel gripper)

The grip direction determines how `MTT_Grip` and `MTT_Release` translate to open/close commands:

- **Inward grip**: `MTT_Grip` → close, `MTT_Release` → open
- **Outward grip**: `MTT_Grip` → open, `MTT_Release` → close

TCP Z offset

The `offsetZTCP` field applies an additional Z-axis offset to the tool center point. This allows compensating for part height differences or gripper jaw geometry without modifying the tool data.

3.3.4 Tool data calibration

The following tool data variables are defined in `MTT_EndEffector.sysx` and should be calibrated for the specific gripper hardware:

Gripper 1 tool data (MTT_tGripper1)

```
PERS tooldata
MTT_tGripper1:=[TRUE,[[0,79.93,116.61],[0,0,0.382685,0.923879]],
    [4.8,[-9.4,-8.7,86.1],[1,0,0,0],0.106,0.048,0.037]];
```

Defines the TCP position and orientation for Gripper 1, plus the tool load data (mass, center of gravity, moments of inertia).

Gripper 2 tool data (MTT_tGripper2)

```
PERS tooldata MTT_tGripper2:=[TRUE,[[0,-
79.93,116.61],[0.923880,0.382683,0,0]],
    [4.8,[-9.4,-8.7,86.1],[1,0,0,0],0.106,0.048,0.037]];
```

Defines the TCP for Gripper 2. Note the different Y offset and orientation compared to Gripper 1, reflecting the physical mounting position on a dual gripper assembly.

Open/close tool data (MTT_tOpenClose)

```
PERS tooldata MTT_tOpenClose:=[TRUE,[[111.75,9,94],[0.5,-0.5,0.5,-
0.5]],
    [4.3,[-1.4,-15.9,79.4],[1,0,0,0],0.096,0.089,0.036]];
```

Dedicated tool data for drawer open/close operations. Typically calibrated to a point on the gripper assembly that aligns with the drawer handle.

Blow-off tool data (MTT_tBlowOff)

```
PERS tooldata MTT_tBlowOff:=[TRUE,[[111.75,9,94],[0.5,-0.5,0.5,-
0.5]],
    [4.3,[-1.4,-15.9,79.4],[1,0,0,0],0.096,0.089,0.036]];
```

Dedicated tool data for blow-off nozzle operations.

3.3.5 Gripper feedback signals

Signal	I/O	Description
`diMTT_isGripper01_Closed` (DI9)	Digital input	Gripper 1 position sensor A — closed position
`diMTT_isGripper01_Opened` (DI10)	Digital input	Gripper 1 position sensor B — open position
`diMTT_isGripper02_Closed` (DI11)	Digital input	Gripper 2 position sensor A — closed position
`diMTT_isGripper02_Opened` (DI12)	Digital input	Gripper 2 position sensor B — open position

3.3.6 Gripper control signals

Signal	I/O	Description
`doMTT_CloseGripper01` (DO16)	Digital output	Pneumatic valve command — 1 = close Gripper 1, 0 = open Gripper 1
`doMTT_CloseGripper02` (DO15)	Digital output	Pneumatic valve command — 1 = close Gripper 2, 0 = open Gripper 2
`doMTT_EnableGripper` (DO14)	Digital output	Pneumatic release signal — must be 1 for gripper actuation
`doMTT_ActivateMagnet` (DO12)	Digital output	Magnet activation for drawer grip operations

3.3.7 Item missed handling

After every pick operation, the framework calls `MTT_checkGrip` which invokes the appropriate `MTT_CheckItemInward/Outward` procedure. If the gripper feedback indicates the part was not secured, the procedure raises the `ERR_MTT_ITEM_MISSED` error.

The error is caught by the `MTT_ItemMissedHandling` procedure in `MTT_EndEffector.sysx`. The default implementation logs a warning message. Integrators should customize this procedure to define the appropriate response (retry, stop, alarm).

3.3.8 Blow-off configuration

Signal	I/O	Description
`doMTT_NozzleOn` (DO13)	Digital output	Air nozzle control — 1 = air on, 0 = air off

The blow-off function is used to clean the spindle face or part before/after placement. Blow-off is enabled as an optional parameter on `MTT_PlaceSpindle` and `MTT_PickSpindle` blocks (the `\BlowOff` switch).

The `MTT_EnableBlowOff` and `MTT_DisableBlowOff` procedures in `MTT_EndEffector.sysx` control the nozzle signal.

3.4 IO signal configuration

3.4.1 Signal overview

The Machine Tending Template uses 16 digital inputs (DI1–DI16) and 16 digital outputs (DO1–DO16) on the ABB Scalable IO module.

3.4.2 Digital input signals (machine to robot)

DI	Signal name	Category	Description
DI1	diMTT_SafetyStandstill	System	Safety scanner — request standstill
DI2	diMTT_SafetyLowSpeed	System	Safety scanner — request low speed
DI3	diMTT_MachineReady	Custom	Machine ready for loading/unloading
DI4	diMTT_MachineDoorOpen	Custom	Machine door is open
DI5	diMTT_MachineDoorClosed	Custom	Machine door is closed
DI6	diMTT_ChuckSpindle1Closed	Custom	Spindle 1 chuck is closed
DI7	diMTT_ChuckSpindle2Closed	Custom	Spindle 2 chuck is closed
DI8	diMTT_DrawerSensorActive	System	Drawer presence sensor
DI9	diMTT_isGripper01_Closed	System	Gripper 1 position sensor A (closed)
DI10	diMTT_isGripper01_Opened	System	Gripper 1 position sensor B (open)
DI11	diMTT_isGripper02_Closed	System	Gripper 2 position sensor A (closed)
DI12	diMTT_isGripper02_Opened	System	Gripper 2 position sensor B (open)
DI13	diMTT_TeachGripper1	System	Teach mode — Gripper 1
DI14	diMTT_TeachGripper2	System	Teach mode — Gripper 2
DI15	diMTT_OptionalGoHome	Custom	Optional home position request
DI16	—	Available	Spare

3.4.3 Digital output signals (robot to machine)

DO	Signal name	Category	Description
DO1	doMTT_OpenChuckSpindle1	Custom	Open spindle 1 chuck
DO2	doMTT_CloseChuckSpindle1	Custom	Close spindle 1 chuck
DO3	doMTT_OpenChuckSpindle2	Custom	Open spindle 2 chuck
DO4	doMTT_CloseChuckSpindle2	Custom	Close spindle 2 chuck
DO5	doMTT_StartMachineCycle	Custom	Start machine cycle (pulse)
DO6	doMTT_FeedHold	Custom	Feed hold (placeholder)
DO7	—	Available	Spare
DO8	doMTT_LeadThroughActive	System	Lead-through task active
DO9	doMTT_ResetSafetyScanner	System	Reset safety scanner
DO10	—	Available	Spare
DO11	—	Available	Spare
DO12	doMTT_ActivateMagnet	System	Activate magnet (drawer operations)

DO13	doMTT_NozzleOn	System	Air nozzle on/off (blow-off)
DO14	doMTT_EnableGripper	System	Pneumatic release signal
DO15	doMTT_CloseGripper02	System	Gripper 2 pneumatic valve
DO16	doMTT_CloseGripper01	System	Gripper 1 pneumatic valve

3.4.4 Hardware Integration Options

The Machine Tending Template uses virtual signals for development and testing. For production deployment with real machine hardware, you have two integration options:

Option 1: Signal Cross-Connection (Recommended)

Create cross-connections between template signals and your actual machine signals using RobotStudio's IO Engineering Tool:

1. **Keep template signals unchanged** in `MTT_EIO.cfg` and `MTT_User.sysx`
2. **Add your machine-specific signals** to the EIO configuration
3. **Create signal cross-connections** using RobotStudio's IO Engineering Tool

Steps to Create Cross-Connections in RobotStudio:

1. Open IO Engineering Tool: In RobotStudio, go to **Controller** tab → **IO Engineering**
2. Configure your machine signals: Add your actual ABB Scalable IO device signals (e.g., `ABB_Scalable_IO_1_DI4`, `ABB_Scalable_IO_1_DO1`), configure the physical device mappings
3. Create cross-connections: In the IO Engineering Tool, locate the **Cross Connections** section, create connections between your machine signals and template signals using the visual interface

Cross-Connections in EIO Configuration: When created through RobotStudio, cross-connections are automatically added to the EIO configuration file using the `EIO_CROSS` syntax:

```
EIO_CROSS:
    # Example: Connect ABB Scalable IO chuck sensor to template
gripper signal
    -Name "CrossCnx1" -Res "diMTT_isGripper01_Closed"\
    -Act1 "ABB_Scalable_IO_1_DI9"
    # Example: Connect ABB Scalable IO door signals to template
signals
    -Name "CrossCnx2" -Res "diMTT_MachineDoorOpen"\
    -Act1 "ABB_Scalable_IO_1_DI4"
    # Example: Connect template commands to ABB Scalable IO outputs
    -Name "CrossCnx3" -Res "ABB_Scalable_IO_1_DO1"\
    -Act1 "doMTT_CloseChuckSpindle1"
```

```
# Example: Connect template gripper command to ABB Scalable IO
-Name "CrossCnx4" -Res "ABB_Scalable_IO_1_DO16"\
-Act1 "doMTT_CloseGripper01"
```

Cross-connection parameters:

Parameter	Description
`-Name`	Unique identifier for the cross-connection
`-Res`	Result signal (the signal that will receive the value)
`-Act1`	Action signal (the signal that provides the value)

Example Cross-Connection Setup:

ABB Scalable IO Signal	Direction	Template Signal
`ABB_Scalable_IO_1_DI4`	→	`diMTT_MachineDoorOpen`
`ABB_Scalable_IO_1_DI5`	→	`diMTT_MachineDoorClosed`
`ABB_Scalable_IO_1_DI6`	→	`diMTT_ChuckSpindle1Closed`
`ABB_Scalable_IO_1_DI7`	→	`diMTT_ChuckSpindle2Closed`
`ABB_Scalable_IO_1_DI9`	→	`diMTT_isGripper01_Closed`
`ABB_Scalable_IO_1_DI10`	→	`diMTT_isGripper01_Opened`
`doMTT_CloseChuckSpindle1`	→	`ABB_Scalable_IO_1_DO1`
`doMTT_OpenChuckSpindle1`	→	`ABB_Scalable_IO_1_DO2`
`doMTT_StartMachineCycle`	→	`ABB_Scalable_IO_1_DO5`
`doMTT_CloseGripper01`	→	`ABB_Scalable_IO_1_DO16`

Visual Cross-Connection Interface: The IO Engineering Tool provides a graphical interface with **FWD** (Forward) blocks. Simply drag connections between input and output signals. The interface shows signal status (Valid/Invalid) in real-time. Cross-connections are automatically saved to the controller configuration.

Benefits of Cross-Connection:

- Template code remains unchanged and maintainable
- Easy to switch between simulation and production
- Clear separation between template logic and machine-specific hardware
- Supports multiple machine configurations with the same template

Option 2: Direct Signal Replacement

Replace template signals directly with your machine-specific signals:

1. **Update `MTT_EIO.cfg`**: Replace virtual device signals with your actual hardware signals
2. **Update RAPID modules**: Replace template signal names with your machine signal names in the relevant modules (`MTT_Machine.sysx` for machine signals, `MTT_EndEffector.sysx` for gripper signals, `MTT_User.sysx` for station signals)

Example Direct Replacement:

```
! Replace in MTT_User.sysx:
! Original template signal:
IF diMTT_MachineDoorOpen=1 THEN

! Replace with your machine signal:
IF ABB_Scalable_IO_1_DI4=1 THEN
```

Benefits of Direct Replacement:

- Simpler configuration with fewer signal layers
- Direct hardware connection without intermediate signals
- Reduced signal count in the controller

Integration Guidelines

For Gripper Integration:

- Connect gripper position sensors to `diMTT_isGripper01_Closed/Opened`, `diMTT_isGripper02_Closed/Opened`
- Connect gripper control outputs to `doMTT_CloseGripper01/02`, `doMTT_EnableGripper`
- Connect drawer magnet control to `doMTT_ActivateMagnet`

For Machine Interface:

- Map machine ready signals to `diMTT_MachineReady`
- Map door status to `diMTT_MachineDoorOpen/Closed`
- Map chuck feedback to `diMTT_ChuckSpindle1/2Closed`
- Map machine commands to `doMTT_StartMachineCycle`

For Safety Integration:

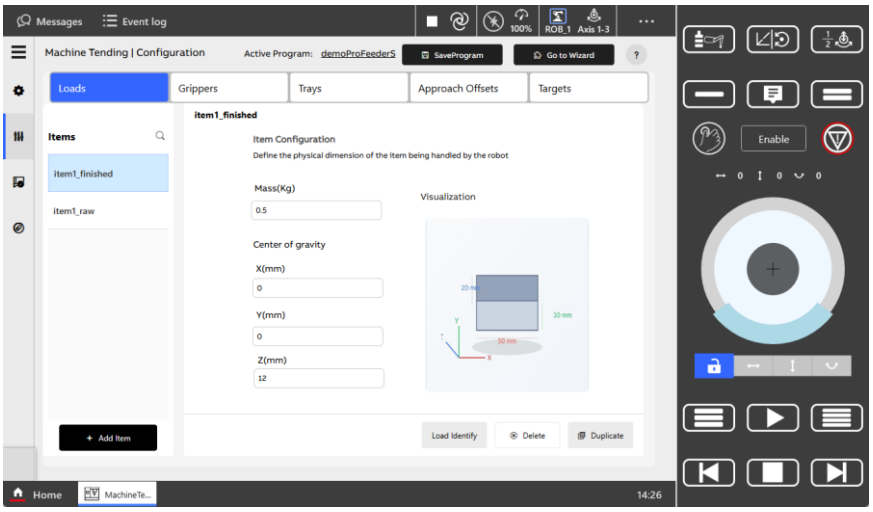
- Reserve system signals (`diMTT_SafetyStandstill`, `diMTT_SafetyLowSpeed`) for safety functions

3.5 Load data configuration

3.5.1 Raw workpiece load data

```
PERS loaddata item1_raw:=[0.7,[0,0,12],[1,0,0,0],0,0.1,0];
```

Defines the mass (0.7 kg), center of gravity offset ([0, 0, 12] mm from TCP), orientation, and moments of inertia for the raw workpiece.



Loads Configuration

3.5.2 Finished workpiece load data

```
PERS loaddata item1_finished:=[0.5,[0,0,12],[1,0,0,0],0,0.1,0];
```

Defines the load data for the finished workpiece. The mass may differ from the raw part (e.g., material removed during machining).

3.5.3 Load management during cycle

The framework automatically manages robot payload through the production cycle:

Procedure	Action
`MTT_addLoad`	Adds the specified load to the current robot payload (called after pick)
`MTT_removeLoad`	Removes the specified load from the current robot payload (called after place)
`MTT_setLoad`	Sets the robot payload to the exact specified load
`MTT_StartCycle`	Resets load to `load0` (empty) at the start of each cycle

Proper load data is critical for accurate dynamic path planning, especially at higher speeds and with heavier workpieces. The load data fields (mass, center of gravity, moments of inertia) should be measured or calculated for each workpiece type.

4 Wizard blocks, RAPID, and WebApp data

This chapter explains how the three core systems of the Machine Tending Template work together: the **Wizard Easy Programming blocks** on the FlexPendant, the **RAPID procedures** they invoke, and the **WebApp configuration data** they consume.

4.1 Overview

The Machine Tending Template uses three interconnected systems to define, configure, and execute the production cycle:

1. **WebApp Configuration Wizard** — the integrator uses the FlexPendant WebApp to define grippers, loads, tray layouts, targets, and approach offsets. All configuration data is written to the RAPID module `MTT_WizardData.modx`.
2. **Wizard Easy Programming blocks** — visual blocks on the FlexPendant that represent high-level operations (pick from drawer, place to spindle, open machine door, etc.). Each block is a **visual wrapper** around an existing RAPID procedure. Blocks do not generate or create procedures — they compose the sequence of procedure calls that becomes the production cycle.
3. **RAPID procedures** — the actual logic that executes robot motion, gripper control, tray management, and machine interface operations. These procedures are defined in the system modules (`MTT_Wizard.sysx`, `MTT_Machine.sysx`, `MTT_User.sysx`, `MTT_EndEffector.sysx`) and are called by the blocks.

The data flow is:

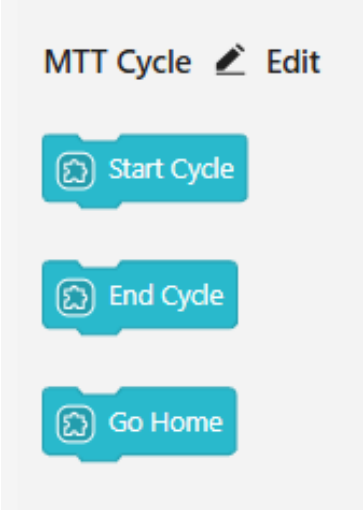
```
WebApp Configuration Wizard
        │
        │ writes gripper defs, load data, tray layouts,
        │ targets, and approach offsets
        ▼
MTT_WizardData.modx (RAPID data module)
        │
        │ data instances appear as dropdown selections
        ▼
Wizard Easy Programming blocks (FlexPendant)
        │
        │ user arranges blocks and selects parameters
        ▼
Wizard.modx — main() procedure (RAPID call sequence)
        │
        │ each line calls a pre-existing RAPID procedure
        ▼
System modules (MTT_Wizard.sysx, MTT_Machine.sysx, etc.)
        │
        │ procedures use data from MTT_WizardData.modx
```

```
          | and HAL modules for hardware control
          ▼
Robot motion, gripper actuation, machine interface
```

4.2 Block palettes

The template provides four block palettes, each grouping related operations. The palettes are available in the Wizard Easy Programming editor on the FlexPendant.

4.2.1 MTT Cycle blocks

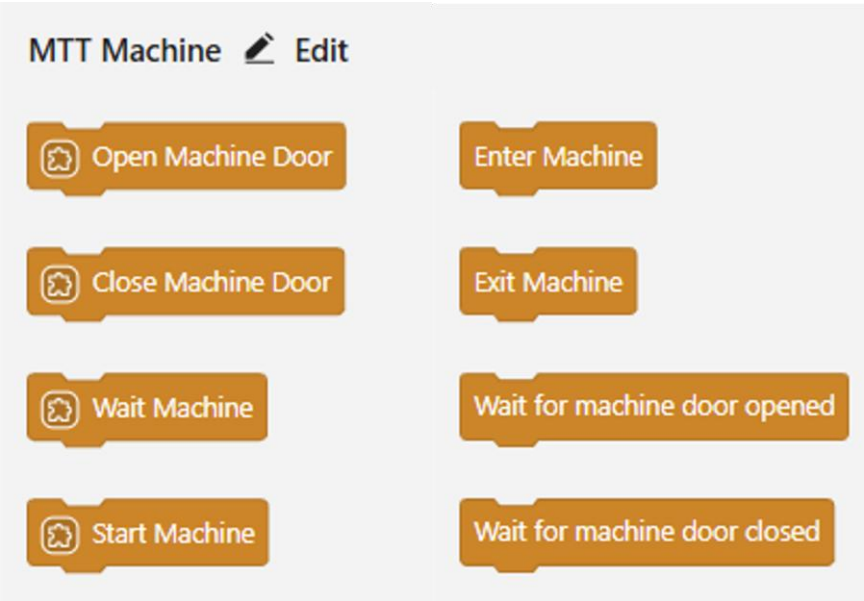


MTT Cycle blocks

These blocks manage the production cycle boundaries and home positioning.

Block	RAPID procedure	Description
Start Cycle	`MTT_StartCycle`	Starts the cycle timer and resets load tracking to empty (`load0`). Place this at the beginning of each cycle iteration.
End Cycle	`MTT_EndCycle`	Stops the cycle timer, increments the production counter, and checks the `MTT_LastCycle` flag. If last cycle is requested, the robot closes all open drawers and moves home.
Go Home	`MTT_Home`	Moves the robot to the home joint position (`MTT_jHome`). Typically placed at the start of the program, before the main loop.

4.2.2 MTT Machine blocks



MTT Machine blocks

These blocks control the machine interface — door, spindle, and cycle operations.

Block	RAPID procedure	Description
Open Machine Door	`MTT_OpenMachineDoor`	Robot physically opens the machine door using motion and magnet, then verifies the door open feedback signal.
Close Machine Door	`MTT_CloseMachineDoor`	Robot physically closes the machine door, then verifies the door closed feedback signal.
Wait Machine	`MTT_waitMachine`	Moves the robot to the wait position (`MTT_pWaitMachine`) and waits until the machine signals ready (`diMTT_MachineReady`).
Start Machine	`MTT_startMachine`	Sends a pulse to `doMTT_StartMachineCycle` to trigger the machine cycle.
Enter Machine	`MTT_EnterMachine`	Waits for machine ready, then moves the robot into the machine work envelope (`MTT_pInMachine`).
Exit Machine	`MTT_ExitMachine`	Moves the robot out of the machine work envelope and back to the wait position.
Wait for machine door opened	`MTT_WaitForMachineDoorOpen`	Waits for the door open feedback signal with a 5-second timeout.
Wait for machine door closed	`MTT_WaitForMachineDoorClosed`	Waits for the door closed feedback signal with a 5-second timeout.

4.2.3 MTT Grippers blocks



MTT Grippers blocks

These blocks control gripper operations and part verification.

Block	Parameters	RAPID procedure	Description
Grip with	Gripper (dropdown)	`MTT_Grip <gripper>`	Activates the selected gripper to grip a part. For inward grippers, this closes the jaws; for outward grippers, this opens them.
Release	Gripper (dropdown)	`MTT_Release <gripper>`	Releases the part from the selected gripper. Inverse of Grip.
Start Blow Off	—	`MTT_StartBlowOff`	Activates the air nozzle for cleaning.
Stop Blow Off	—	`MTT_StopBlowOff`	Deactivates the air nozzle.
Check Item in Gripper	Gripper number (1 or 2)	`MTT_checkItemInGripper <n>`	Checks the gripper position feedback to verify that a part is held. Raises `ERR_MTT_ITEM_MISSED` if the check fails.

The **Gripper dropdown** shows all `MTT_GRIPPER` instances defined in `MTT_WizardData.modx` — for example, `Gripper1` and `Gripper2`. These instances are created and configured through the WebApp Grippers tab.

4.2.4 MTT Tending blocks



MTT Tending blocks

These blocks perform the high-level pick and place operations that form the core of the production cycle.

Drawer operations

Block	Parameters	RAPID procedure	Description
Pick from Drawer	Load (dropdown), Gripper (dropdown)	`MTT_PickDrawer <gripper>, <load>`	Automatically selects the next available unload drawer (based on priority and ready state), opens it if closed, calculates the next tray position, moves the robot to pick, activates the gripper, and increments the tray counter.
Place to Drawer	Load (dropdown), Gripper (dropdown)	`MTT_PlaceDrawer <gripper>, <load>`	Automatically selects the next available load drawer, opens it if closed, calculates the

			next tray position, moves the robot to place, releases the gripper, increments the tray counter, and closes the drawer when the tray is full.

Optional parameters (accessible via the **>** expand button):

- **Close when finished** (Pick from Drawer only) — close the drawer after the tray is emptied.
- **Don't skip first stack** — disables the default behavior of skipping position [1,1,1] on the first pick.

Spindle operations

Block	Parameters	RAPID procedure	Description
Place to Spindle	Load (dropdown), Spindle (1 or 2), Gripper (dropdown)	`MTT_PlaceSpindle <gripper>, <spindle>, <load>`	Opens the spindle chuck, moves to the place position with approach offsets, releases the part, closes the spindle chuck.
Pick from Spindle	Load (dropdown), Spindle (1 or 2), Gripper (dropdown)	`MTT_PickSpindle <gripper>, <spindle>, <load>`	Opens the spindle chuck, moves to the pick position with approach offsets, grips the part, verifies the grip feedback.

Optional parameters:

- **Blow Off** — activates the air nozzle before/after the spindle operation to clean the spindle face.
- **Release Before** (Place to Spindle only) — releases the part before closing the spindle chuck, rather than after. Useful when the part must be free-standing in the chuck before clamping.

User location operations

Block	Parameters	RAPID procedure	Description
Pick at	Load (dropdown), Location (robtarget), Gripper (dropdown)	`MTT_Pick <location>, <gripper>, <load>`	Generic pick at a user-defined position. Uses approach offsets from `offsetPickInUser` / `offsetPickOutUser`.
Place at	Load (dropdown), Location (robtarget), Gripper (dropdown)	`MTT_Place <location>, <gripper>, <load>`	Generic place at a user-defined position.

Optional parameter:

- **WObj** — specify a work object for the operation (default: `wobj0`).

Tray operations (standalone)

Block	Parameters	RAPID procedure	Description
Pick from Tray	Load (dropdown), Tray (dropdown), Gripper (dropdown)	`MTT_PickTray <gripper>, <load>, <tray>`	Pick from a standalone tray (not associated with a drawer). The tray dropdown shows all `MTT_TRAY` instances.
Place to Tray	Load (dropdown), Tray (dropdown), Gripper (dropdown)	`MTT_PlaceTray <gripper>, <load>, <tray>`	Place to a standalone tray.

4.3 How block dropdowns consume WebApp configuration data

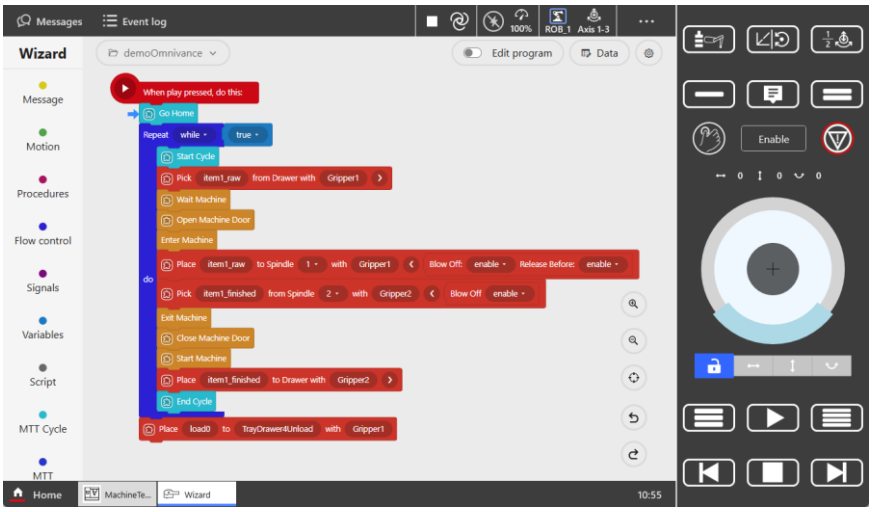
The dropdown selections in the Wizard blocks are populated from data instances defined in `MTT_WizardData.modx` — the same module that the WebApp writes to during configuration. This creates a seamless connection between the configuration and programming phases.

Dropdown	Data type	Populated from	Configured in WebApp tab
Gripper	`MTT_GRIPPER`	`Gripper1`, `Gripper2`, etc.	**Grippers** tab
Load	`loaddata`	`item1_raw`, `item1_finished`, `load0`, etc.	**Loads** tab
Tray	`MTT_TRAY`	`TrayDrawer1Load`, `TrayDrawer1Unload`, etc.	**Trays** tab
Spindle	Fixed list	[1, 2]	Targets configured in **Targets** tab
Location	`robtarget`	User-defined robtarget variables	**Targets** tab
WObj	`wobjdata`	Work object instances in loaded modules	**Targets** tab

When the integrator adds a new gripper definition in the WebApp Grippers tab (e.g., `Gripper3`), that instance immediately becomes available in all Wizard block gripper dropdowns. Similarly, when new load items or tray definitions are created in the WebApp, they automatically appear in the corresponding block dropdowns.

4.4 Side-by-side example: Wizard blocks and RAPID code

The following table shows the default production cycle as it appears in the Wizard editor (visual blocks) alongside the corresponding RAPID code in `Wizard.modx`:



Default cycle in Wizard editor

Wizard block	RAPID code
Go Home	`MTT_Home;`
Repeat while **true**	`WHILE TRUE DO`
Start Cycle	`MTT_StartCycle;`
Pick `item1_raw` **from Drawer with** `Gripper1`	`MTT_PickDrawer Gripper1, item1_raw;`
Wait Machine	`MTT_waitMachine;`
Open Machine Door	`MTT_OpenMachineDoor;`
Enter Machine	`MTT_EnterMachine;`
Place `item1_raw` **to Spindle** `1` **with** `Gripper1` [Blow Off: enable, Release Before: enable]	`MTT_PlaceSpindle Gripper1, 1, item1_raw, \BlowOff \releaseBefore;`
Pick `item1_finished` **from Spindle** `2` **with** `Gripper2` [Blow Off: enable]	`MTT_PickSpindle Gripper2, 2, item1_finished, \BlowOff;`
Exit Machine	`MTT_ExitMachine;`
Close Machine Door	`MTT_CloseMachineDoor;`
Start Machine	`MTT_startMachine;`
Place `item1_finished` **to Drawer with** `Gripper2`	`MTT_PlaceDrawer Gripper2, item1_finished;`
End Cycle	`MTT_EndCycle;`

Each visual block maps exactly to one RAPID procedure call. The block parameters (gripper, load, spindle number) become the procedure arguments. Optional parameters (`\BlowOff`, `\releaseBefore`) appear as switches accessible through the expand button on the block.

The production cycle can be viewed and edited in either form:

- **Wizard Easy Programming** on the FlexPendant — drag, reorder, add, or remove blocks without writing code.
- **RAPID text editor** — edit `Wizard.modx` directly for advanced customization.

Both views represent the same program. Changes made in the Wizard editor update the RAPID code, and changes made in the RAPID code are reflected in the Wizard view.

Wizard blocks, RAPID, and WebApp data

5 The Machine Tending Template WebApp

5.1 WebApp overview

The Machine Tending Template WebApp is a FlexPendant application built with ABB AppStudio. It provides the primary user interface for both commissioning (configuration) and daily operation (production).

5.1.1 Starting the WebApp

The WebApp is accessible from the FlexPendant home screen under the name **MachineTending**. It can also be accessed from a web browser at:

```
http://<controller-ip>/fileservice/$home/WebApps/MachineTending/
```

5.1.2 Main screen and navigation

The WebApp uses a hamburger menu (☰) in the top-left corner for navigation between the main views:

View	Purpose
Production	Runtime monitoring and control — start/stop cycle, counters, drawer management
Configuration	Commissioning interface — teach positions, define parameters, set up tray layouts
Gantry	Optional gantry-specific configuration (hidden by default)

The Configuration view also provides a **Go to Wizard** button that opens the Wizard Easy Programming editor on the FlexPendant for block-based cycle editing.

5.2 The Configuration Wizard

The Configuration view provides a tabbed interface with five configuration domains. The integrator works through these tabs during commissioning to define all parameters needed for production.

5.2.1 Opening the Configuration Wizard

From the Production view, press the **Go to Wizard** button in the header area. Alternatively, use the hamburger menu to navigate directly to the Configuration view.

The Configuration view header shows:

- The **active program name** (e.g., `demoOmnivance`)
- A **SaveProgram** button to persist changes
- A **Go to Wizard** button to open the Wizard Easy Programming editor

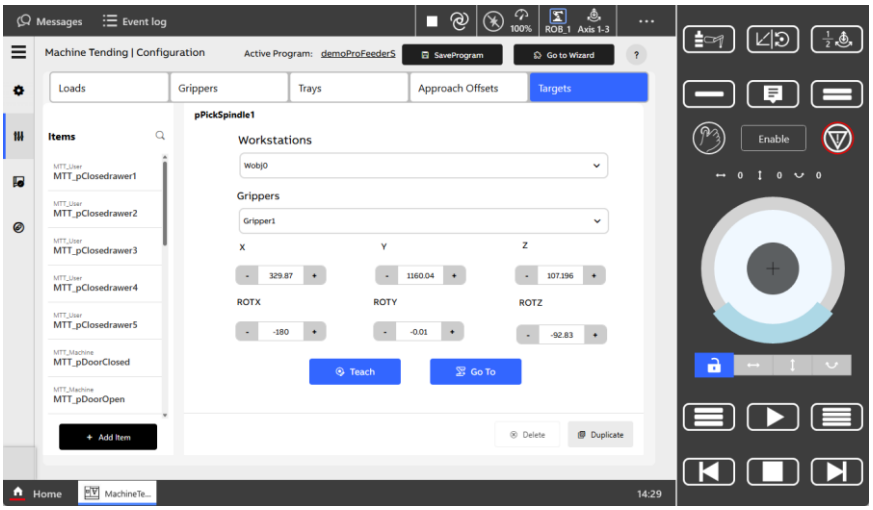
5.2.2 Position teaching

The WebApp supports teaching robot positions through two methods:

- **Lead-through teaching** — physically guide the robot to the desired position (collaborative robots).
- **FlexPendant jogging** — use the FlexPendant joystick to move the robot, then press the Teach button.

Targets tab

The **Targets** tab is the main configuration window for teaching robot positions used by the application. It is the place where the integrator selects an existing target, chooses the active work object and gripper/tool context, and then stores the current robot pose with **Teach**.



Targets Configuration

The Targets tab is used for positions such as:

- spindle pick and place targets (`pPickSpindle1`, `pPlaceSpindle1`, `pPickSpindle2`, `pPlaceSpindle2`)
- machine door targets (`MTT_pDoorOpen`, `MTT_pDoorClosed`)
- machine wait and in-machine targets (`MTT_pWaitMachine`, `MTT_pInMachine`)
- drawer open and close targets (`MTT_pOpendrawer1`–`MTT_pOpendrawer6`, `MTT_pClosedrawer1`–`MTT_pClosedrawer6`)
- user-defined pick and place targets (`pPickUser`, `pPlaceUser`)

Typical workflow in the Targets tab:

1. Select the target to edit from the target list (for example, `pPickSpindle1`).
2. Select the appropriate work object and gripper/tool data for that target.
3. Move the robot to the desired position using lead-through or FlexPendant jogging.
4. Press **Teach** to record the current position.
5. Press **Go To** to verify the stored target.

The Targets tab explains **how** the position is taught. The individual chapters in this manual explain **what each target is used for** in the machine, station, or cycle logic.

Home position

The home position (`MTT_jHome`) is a joint target defined in `MTT_User.sysx`. It is the safe starting and ending position for all production cycles. The home position is typically taught once during initial commissioning and does not appear in the WebApp Targets tab (it is a constant in the station module).

Spindle pick and place positions

Target	Description
`pPickSpindle1`	Position to pick a finished part from spindle 1
`pPlaceSpindle1`	Position to place a raw part into spindle 1
`pPickSpindle2`	Position to pick a finished part from spindle 2
`pPlaceSpindle2`	Position to place a raw part into spindle 2

These targets are configured in the **Targets** tab. Select the target, choose the work object and gripper, jog the robot, and press **Teach**.

Drawer positions

Drawer open and close positions (`MTT_pOpendrawer1`–`MTT_pOpendrawer6`, `MTT_pClosedrawer1`–`MTT_pClosedrawer6`) are defined in `MTT_User.sysx` and are typically taught during initial station commissioning using the Targets tab.

Machine door positions

The machine door positions (`MTT_pDoorOpen`, `MTT_pDoorClosed`) are defined in `MTT_Machine.sysx`. These are taught using the Targets tab, selecting the `MTT_tOpenClose` tool data.

Machine wait and enter positions

Target	Description
`MTT_pWaitMachine`	Safe position near the machine for waiting
`MTT_pInMachine`	Position inside the machine work envelope

User-defined pick and place positions

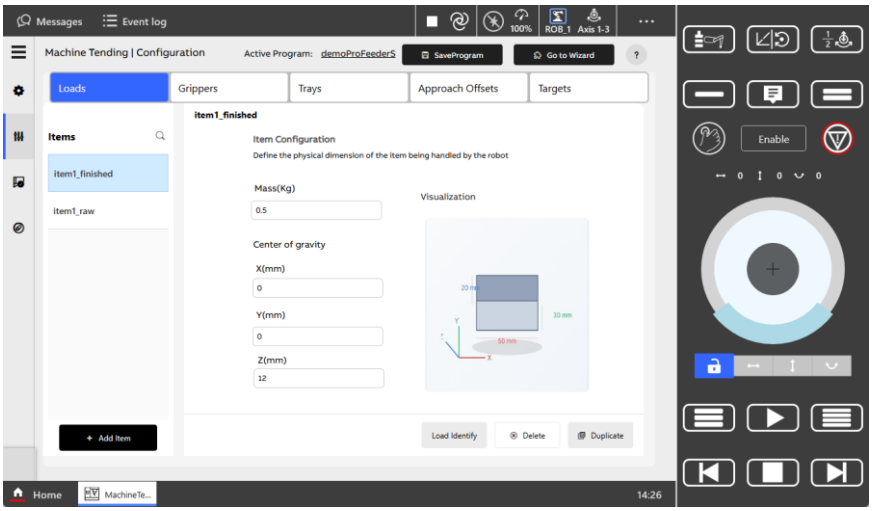
Target	Description
`pPickUser`	Generic user-defined pick position
`pPlaceUser`	Generic user-defined place position

These are available for custom cycle steps using the "Pick at" and "Place at" blocks.

5.2.3 Process parameter configuration

Load data

The **Loads** tab defines payload properties for items being handled:



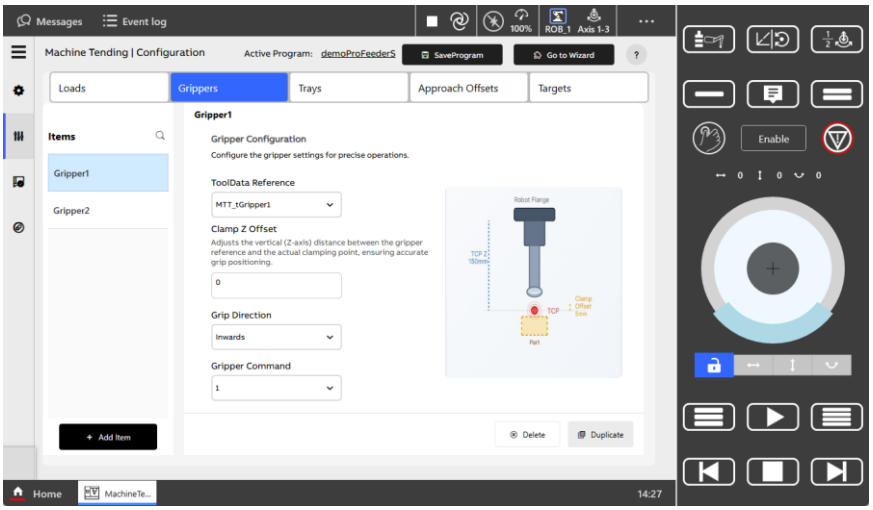
Loads Configuration

Field	Description
Mass (Kg)	Weight of the item
X (mm), Y (mm), Z (mm)	Center of gravity offsets (relative to TCP)

Items are identified by name (e.g., `item1_finished`, `item1_raw`). New items can be added with the **+ Add Item** button.

Gripper definitions

The **Grippers** tab allows configuration of each gripper's properties:



Grippers Configuration

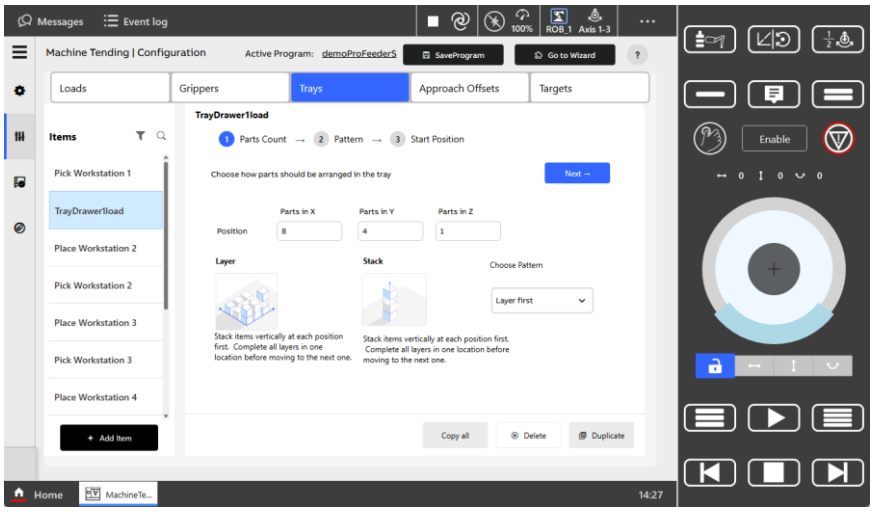
Field	Description
ToolData Reference	Dropdown selecting the tool data variable (e.g., `MTT_tGripper1`)
Clamp Z Offset	Additional Z-axis offset for the gripper clamping point
Grip Direction	Inwards (close to grip) or Outwards (open to grip)
Gripper Command	Command set number (1 or 2) — determines which hardware procedures are called

New grippers can be added with the **+ Add Item** button.

Tray layout parameters

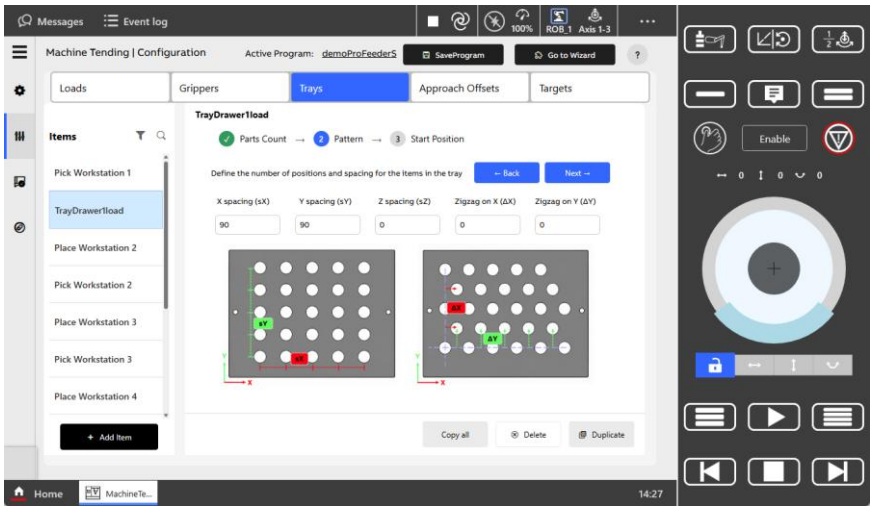
The **Trays** tab uses a three-step wizard for each tray definition:

Step 1 — Parts Count: Define the grid dimensions (Parts in X, Parts in Y, Parts in Z) and select the increment pattern (Layers or Stack).



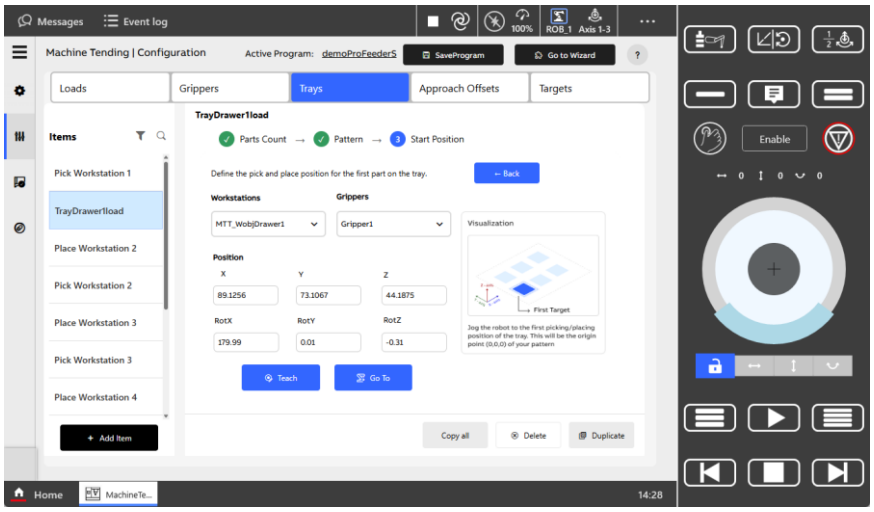
Trays — Parts Count

Step 2 — Pattern: Define spacing (X spacing, Y spacing, Z spacing) and even row/column offsets. A visualization shows the resulting grid layout.



Trays — Pattern

Step 3 — Start Position: Select the work object and gripper, define the first target position (X, Y, Z, RotX, RotY, RotZ), and use **Teach** / **Go To** buttons.



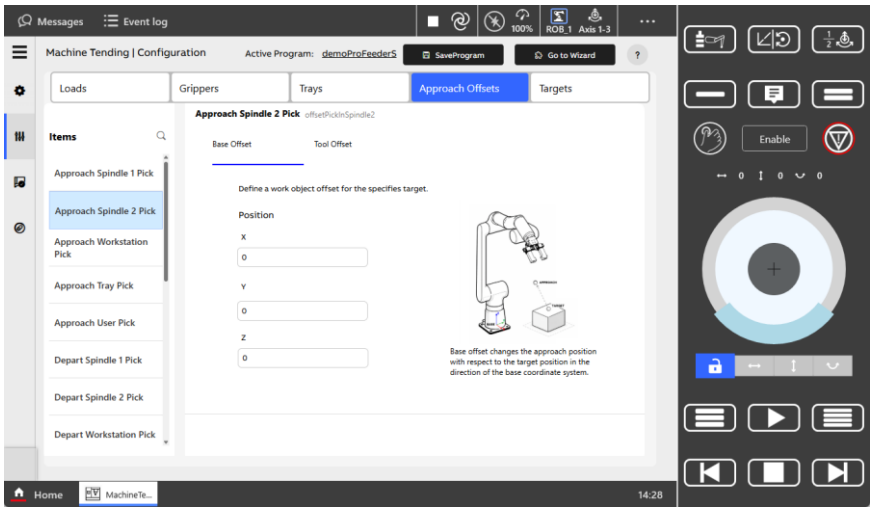
Trays — Start Position

The **Copy all** button copies the current tray configuration to all other trays of the same type (load or unload).

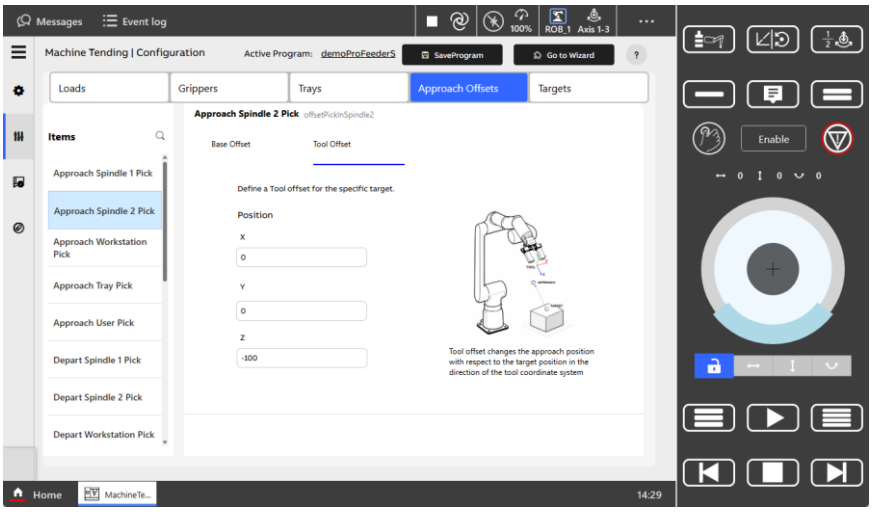
Approach and depart offsets

The **Approach Offsets** tab shows all defined offset records. Each offset has two sub-tabs:

- **Base Offset** — offset in the work object coordinate system (X, Y, Z position).
- **Tool Offset** — offset in the tool coordinate system (X, Y, Z position).



Approach Offsets — Base Offset



Approach Offsets — Tool Offset

The approach offset determines the path the robot follows when approaching and departing a target. For example, a tool Z offset of −50 means the robot approaches from 50 mm behind the target along the tool Z-axis.

Predefined approach offsets include:

- Spindle offsets (`offsetPickInSpindle1`, `offsetPickOutSpindle1`, etc.)
- Drawer offsets (`offsetPickInUnloadDrawer`, `offsetPlaceInLoadDrawer`, etc.)
- Tray offsets (`offsetPickInUnloadTray`, `offsetPlaceInLoadTray`, etc.)
- User position offsets (`offsetPickInUser`, `offsetPickOutUser`)

5.2.4 Machine interface configuration

Machine door behavior

The machine door behavior is defined by the `MTT_OpenMachineDoor` and `MTT_CloseMachineDoor` procedures in `MTT_Machine.sysx`. The door positions (`MTT_pDoorOpen`, `MTT_pDoorClosed`) are taught through the Targets tab.

Chuck/spindle sequences

Spindle open/close sequences are implemented in the machine module. The integrator can test individual spindle operations from the Manual Operation tab (see section 4.3).

Gripper sequences

Gripper open/close behavior is tested from the Manual Operation tab. The Cell section shows separate controls for Gripper 1 and Gripper 2 with open/close buttons and feedback indicators.

5.2.5 Configuration validation

After completing the configuration:

1. Load the program using the **Load Program** button on the Production tab.
2. Switch to **Manual mode** on the FlexPendant.
3. Run the program at reduced speed (use the speed override on the FlexPendant).
4. Observe the full cycle, verifying clearances at each position.
5. Test drawer open/close, gripper pick/place, and machine door operations individually using the Manual Operation tab.

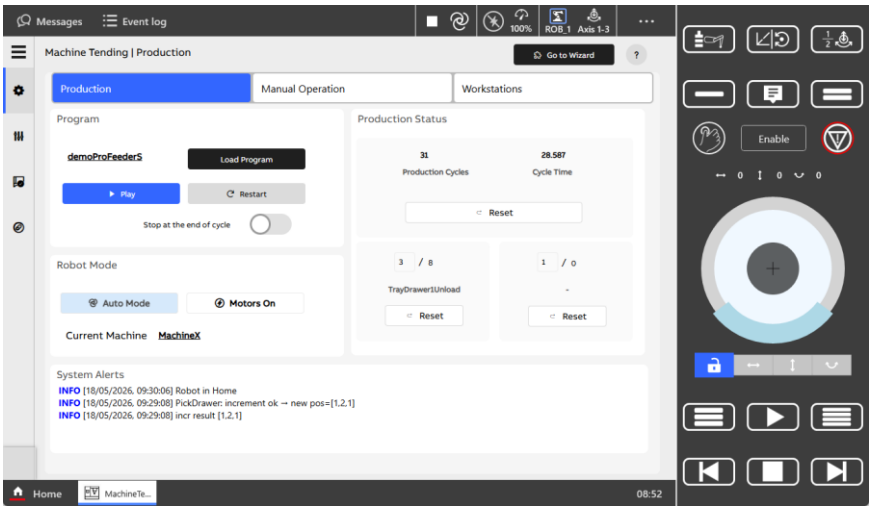
5.2.6 Saving configuration

Press the **SaveProgram** button in the Configuration view header to persist all changes. The configuration is stored in `MTT_WizardData.modx` in the Add-In HOME RAPID folder.

5.3 Running production from the WebApp

5.3.1 Operation view

The **Production** tab is the primary runtime interface.



Production View

Production start actions

Control	Action
Load Program	Loads the selected program file (`.pgf`) into the controller
Play	Starts program execution
Restart	Restarts the program from the beginning
Auto Mode	Switches the controller to automatic mode
Motors On	Enables the robot motors

The **Current Machine** indicator shows which machine module is active (e.g., `MachineX`).

Production stop actions (pause, resume, stop)

Control	Action
Stop at the end of cycle	Toggle — when enabled, sets `MTT_LastCycle` flag; the robot completes the current cycle, closes drawers, and moves home

Home run

Press the **Go Home** button on the Manual Operation tab to send the robot to the home position at any time.

Last cycle handling

When "Stop at the end of cycle" is enabled:

1. The `MTT_LastCycle` flag is set to `TRUE`.
2. The current cycle completes normally.
3. At `MTT_EndCycle`, the robot detects the flag.
4. The robot sets load to `load0`, closes all open drawers, moves home, and exits the cycle loop.

5.3.2 Production monitoring

Cycle time

The **Cycle Time** indicator on the Production tab shows the elapsed time for the last completed cycle. Timing is measured between `MTT_StartCycle` and `MTT_EndCycle`.

Production counter

The **Production Cycles** counter shows the total number of completed cycles since the last reset. Press the **Reset** button to zero the counter.

Drawer and tray status

The tray position counters (displayed below the production counter) show the current position within each active tray. Press the individual **Reset** buttons to reset tray positions to [1,1,1].

Gripper status

Gripper feedback is monitored in the Manual Operation tab, showing open/closed indicators for both grippers.

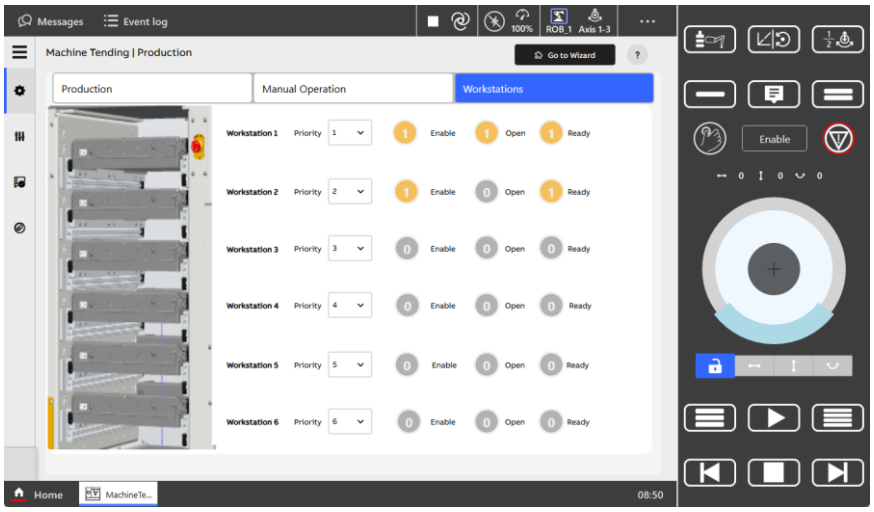
System alerts

The **System Alerts** section at the bottom of the Production tab shows real-time log messages from the RAPID program, including:

- Informational messages (tray position changes, drawer state)
- Warning messages (no drawers available)
- Error messages (timeouts, item missed)

5.3.3 Drawer management

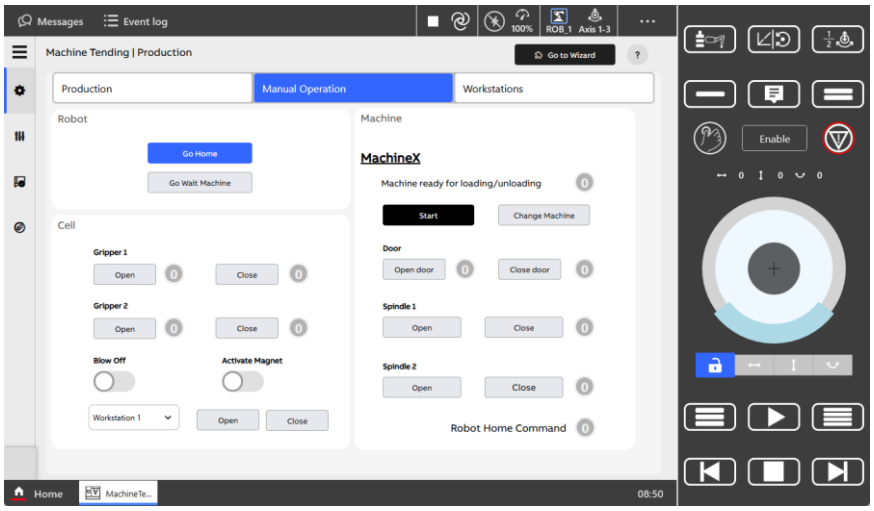
The **Drawers** tab provides real-time drawer state monitoring and control.



Drawers Management

Manual drawer open/close

The Manual Operation tab provides dropdown selection of a drawer number and **Open** / **Close** buttons for manual drawer operations during setup.



Manual Operation

Drawer ready/disabled state

Each drawer on the Drawers tab shows:

- **Priority** dropdown (1–6) — lower number = higher priority for automatic selection
- **Enable** indicator — toggle to enable/disable the drawer
- **Open** indicator — current open/closed state
- **Ready** indicator — toggle to mark the drawer as ready (tray present)

Tray position reset

Tray position counters can be reset from the Production tab using the individual Reset buttons. This resets the tray to position [1,1,1], which is useful after replacing a tray or recovering from an interruption.

5.4 Wizard Easy Programming

5.4.1 Overview

Wizard Easy Programming is a graphical block-based programming environment on the FlexPendant. It allows operators and integrators to build and modify the production cycle by dragging and arranging visual blocks without writing RAPID code.

Access the Wizard editor by pressing the **Go to Wizard** button in the Configuration view header. The Wizard editor opens as a separate view on the FlexPendant, showing:

- The **block palette** on the left side (Message, Motion, Procedures, Flow control, Signals, Variables, Script, MTT Cycle, MTT Machine, MTT Grippers, MTT Tending)
- The **program canvas** in the center, where blocks are arranged
- **Zoom** and **undo/redo** controls on the right side

The **Edit program** toggle switches between view mode and edit mode. In edit mode, blocks can be dragged from the palette to the canvas, rearranged, or deleted.

5.4.2 MTT Tending blocks

The MTT Tending palette provides 8 blocks for pick and place operations. See section 3B for the complete block reference with parameters, RAPID mapping, and optional parameters.

Key blocks:

- **Pick from Drawer** / **Place to Drawer** — automatic drawer selection and tray position management
- **Pick from Spindle** / **Place to Spindle** — machine spindle operations with optional blow-off
- **Pick at** / **Place at** — generic position-based operations
- **Pick from Tray** / **Place to Tray** — standalone tray operations

5.4.3 MTT Machine blocks

The MTT Machine palette provides 8 blocks for machine interface control. These blocks take no parameters — all machine positions and signals are defined in `MTT_Machine.sysx`.

5.4.4 MTT Grippers blocks

The MTT Grippers palette provides 5 blocks for gripper control and part verification. The Grip/Release blocks take a gripper parameter (dropdown); the Check Item block takes a gripper number (1 or 2).

5.4.5 MTT Cycle blocks

The MTT Cycle palette provides 3 blocks for cycle management (Start Cycle, End Cycle, Go Home). These blocks take no parameters.

5.4.6 Modifying the production cycle

To modify the production cycle:

1. Open the Wizard editor (Go to Wizard button).
2. Enable **Edit program** toggle.
3. Make changes:
4. **Add a step**: drag a block from the palette to the desired position in the canvas.
5. **Remove a step**: drag the block off the canvas or right-click and delete.
6. **Reorder steps**: drag blocks to new positions within the sequence.
7. **Change parameters**: click on a block's dropdown to change the gripper, load, spindle, or other parameter.
8. **Enable optional parameters**: click the **>** button on blocks that support optional parameters (e.g., BlowOff, releaseBefore).
9. Save the program.

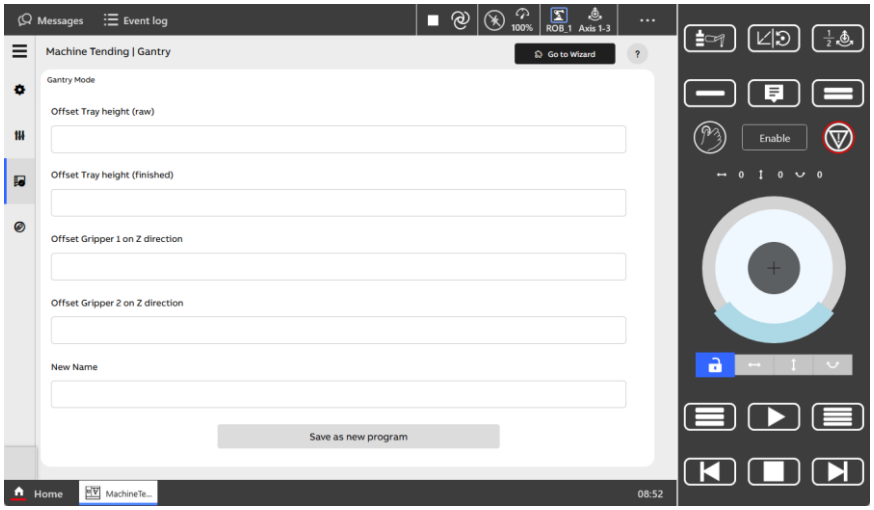
The Wizard editor can also be switched to direct RAPID editing:

- Toggle **Edit program** to switch between block view and code view.
- Standard Wizard blocks (Flow control, Variables, etc.) can be combined with MTT blocks.
- Any valid RAPID statement can be added using the Script block.

For advanced customization beyond what blocks support, edit `Wizard.modx` directly in a RAPID text editor.

5.5 Gantry mode (optional)

The Gantry view provides configuration for gantry-specific installations.



Gantry Mode

Field	Description
Offset Tray height (raw)	Height offset applied to raw part tray positions
Offset Tray height (finished)	Height offset applied to finished part tray positions
Offset Gripper 1 on Z direction	TCP Z offset compensation for Gripper 1
Offset Gripper 2 on Z direction	TCP Z offset compensation for Gripper 2
New Name	Name for saving the configuration as a new program

Press **Save as new program** to create a program variant with the gantry-specific offsets applied.

6 RAPID reference

This chapter provides a complete reference for every public procedure and function in the Machine Tending Template. Entries are grouped by module and follow a consistent format:

- **Description** — what the routine does.
- **Syntax** — RAPID signature.
- **Parameters** — argument table (if applicable).
- **Program execution** — step-by-step behavior.
- **Example** — usage in a production cycle.
- **Related** — cross-references to related routines.

Convention: Procedures marked **HAL** are implemented differently per station or machine variant. The behavior described here refers to the default variants (Omnivance station, MachineX machine). Integrators may need to adapt these procedures for custom hardware.

6.1 Core module — MTT_Wizard.sysx

6.1.1 Pick and Place

MTT_Pick — Pick a part at a position

Description

Picks a part at a specified robot position. The robot approaches with an offset, opens the gripper, moves to the pick position, grips the part, checks the grip feedback, adds the load, and departs with an offset.

Syntax

```
MTT_Pick pPick, gripper, load [\wobj] [\offsetIN] [\offsetOUT]
```

Parameters

Parameter	Type	Description
`pPick`	`robtarget`	The pick position
`gripper`	`MTT_GRIPPER`	The gripper definition to use
`load`	`loaddata`	The load data to add after picking
`\wobj`	`wobjdata`	Optional. Work object for the motion. Default: `wobj0`
`\offsetIN`	`MTT_APPROACH_OFFSET`	Optional. Approach offset. Default: `MTT_ApprOffs_Default` (200 mm Z)
`\offsetOUT`	`MTT_APPROACH_OFFSET`	Optional. Depart offset. Default: `MTT_ApprOffs_Default`

Program execution

1. Calculate approach position by applying `offsetIN` to `pPick`.
2. Calculate depart position by applying `offsetOUT` to `pPick`.
3. Fetch the tool data from `gripper.tooldataReference`.
4. MoveJ to approach position at `vmax`, `z10`.
5. Open the gripper (prepare grip).
6. MoveL to `pPick` at `v100`, `fine`.
7. Close the gripper (grip).
8. Check grip feedback. If `ERR_MTT_ITEM_MISSED` is raised, call `MTT_ItemMissedHandling` and continue.
9. Add load to robot payload.
10. MoveL to depart position at `vmax`, `z10`.

Example

```
MTT_Pick pPickUser, Gripper1, item1_raw \wobj:=wobj0
    \offsetIN:=offsetPickInUser \offsetOUT:=offsetPickOutUser;
```

Related: `MTT_Place`, `MTT_PickDrawer`, `MTT_PickSpindle`, `MTT_PickTray`

MTT_Place — Place a part at a position

Description

Places a part at a specified robot position. The robot approaches with an offset, moves to the place position, releases the gripper, removes the load, and departs with an offset.

Syntax

```
MTT_Place pPlace, gripper, load [\wobj] [\offsetIN] [\offsetOUT]
```

Parameters

Parameter	Type	Description
`pPlace`	`robtarget`	The place position
`gripper`	`MTT_GRIPPER`	The gripper definition to use
`load`	`loaddata`	The load data to remove after placing
`\wobj`	`wobjdata`	Optional. Work object for the motion. Default: `wobj0`
`\offsetIN`	`MTT_APPROACH_OFFSET`	Optional. Approach offset. Default: `MTT_ApprOffs_Default`
`\offsetOUT`	`MTT_APPROACH_OFFSET`	Optional. Depart offset. Default: `MTT_ApprOffs_Default`

Program execution

1. Calculate approach and depart positions from offsets.
2. Fetch tool data from `gripper.tooldataReference`.
3. MoveJ to approach position at `vmax`, `z10`.
4. MoveL to `pPlace` at `v100`, `fine`.
5. Release the gripper.
6. Remove load from robot payload.
7. MoveL to depart position at `vmax`, `z10`.

Example

```
MTT_Place pPlaceUser, Gripper2, item1_finished \wobj:=wobj0
    \offsetIN:=offsetPlaceInUser \offsetOUT:=offsetPlaceOutUser;
```

Related: `MTT_Pick`, `MTT_PlaceDrawer`, `MTT_PlaceSpindle`, `MTT_PlaceTray`

6.1.2 Drawer operations

MTT_PickDrawer — Pick a part from the next available unload drawer

Description

Automatically selects the best available unload drawer (based on priority, ready state, and enabled state), opens it if closed, loads the tray configuration, calculates the next tray position, picks the part, and increments the tray counter.

Syntax

```
MTT_PickDrawer gripper, load [\closeWhenFinished] [\noStackSkip]
```

Parameters

Parameter	Type	Description
`gripper`	`MTT_GRIPPER`	The gripper definition to use
`load`	`loaddata`	The load data to add after picking
`\closeWhenFinished`	`switch`	Optional. Close the drawer when the tray is emptied
`\noStackSkip`	`switch`	Optional. Disable the first-stack skip behavior (stack-first mode)

Program execution

1. Call `MTT_selectCurrentDrawer\UnLoad` to find the best drawer.
2. If no drawer is available, call `MTT_NoDrawerForUnload` (stops the program).
3. If the selected drawer is closed, call `MTT_OpenDrawer`.
4. Fetch the tray definition from `TrayDrawer<N>Unload`.
5. Fetch the drawer work object.
6. **First-stack skip** (default): if in stack-first mode with multiple layers and position is [1,1,1], skip to [2,1,1]. This avoids picking from a position that may interfere with the tray structure.
7. Calculate the tray target position.
8. Fetch tool data, open gripper (prepare).
9. MoveJ to approach, MoveL to pick position.
10. Grip, check grip, add load.
11. MoveL to depart.
12. Increment the tray counter. If the tray is finished:
13. Mark the drawer as NOT_READY.
14. If `\closeWhenFinished`, close the drawer.

Example

```
MTT_PickDrawer Gripper1, item1_raw;
MTT_PickDrawer Gripper1, item1_raw \closeWhenFinished;
MTT_PickDrawer Gripper1, item1_raw \noStackSkip;
```

Related: `MTT_PlaceDrawer`, `MTT_selectActiveDrawer`, `MTT_OpenDrawer`

MTT_PlaceDrawer — Place a part into the next available load drawer

Description

Automatically selects the best available load drawer, opens it if closed, loads the tray configuration, calculates the next tray position, places the part, and increments the tray counter. Closes the drawer when the tray is full.

Syntax

```
MTT_PlaceDrawer gripper, load [\noStackSkip]
```

Parameters

Parameter	Type	Description
`gripper`	`MTT_GRIPPER`	The gripper definition to use
`load`	`loaddata`	The load data to remove after placing
`\noStackSkip`	`switch`	Optional. Disable the last-stack skip behavior

Program execution

1. Call `MTT_selectCurrentDrawer\Load` to find the best drawer.
2. If no drawer is available, call `MTT_NoDrawerForLoad` (stops the program).
3. If the selected drawer is closed, call `MTT_OpenDrawer`.
4. Fetch the tray definition from `TrayDrawer<N>Load`.
5. Fetch the drawer work object, calculate tray target.
6. MoveJ to approach, MoveL to place position.
7. Release gripper, remove load.
8. MoveL to depart.
9. Evaluate if the tray is finished or in the last stack (stack-first mode).
10. If finished or last-stack reached:
11. Mark drawer as NOT_READY.
12. Close the drawer.

Example

```
MTT_PlaceDrawer Gripper2, item1_finished;
```

Related: `MTT_PickDrawer`, `MTT_selectActiveDrawer`, `MTT_CloseDrawer`

MTT_selectActiveDrawer — Find the best available drawer

Description

Returns the index (1–6) of the first available drawer using a two-pass selection algorithm:

1. First pass: find the first open, ready, and enabled drawer (lowest index wins).
2. Second pass: if no open drawer, find the ready+enabled drawer with the lowest priority number.

Returns 0 if no drawer is available.

Syntax

```
num := MTT_selectActiveDrawer()
```

Return value: Drawer index (1–6), or 0 if none available.

Related: `MTT_PickDrawer`, `MTT_PlaceDrawer`

6.1.3 Tray operations

MTT_PickTray — Pick from a standalone tray

Description

Picks a part from a standalone tray (not associated with a drawer). Checks tray readiness, fetches the work object, calculates the target, performs the pick, and increments the counter. If the tray is finished, prompts the operator to reset or exit.

Syntax

```
MTT_PickTray gripper, load, tray
```

Parameters

Parameter	Type	Description
`gripper`	`MTT_GRIPPER`	The gripper to use
`load`	`loaddata`	The load data to add after picking
`tray`	`MTT_TRAY` (INOUT)	The tray definition (position is updated in place)

Program execution

1. Set the current tray to the specified tray for unload mode.
2. Check if the tray is ready and the position is valid. If not:
3. Prompt the operator with a dialog: **Reset** or **Cancel**.
4. If Reset: set position to [1,1,1] and continue.
5. If Cancel: go home and exit cycle.
6. Fetch approach offsets (`offsetPickInUnloadTray`, `offsetPickOutUnloadTray`).
7. Fetch work object from the tray definition.
8. Calculate target and call `MTT_Pick`.
9. Increment counter. If finished, mark tray as NOT_READY.

Example

```
MTT_PickTray Gripper1, item1_raw, TrayDrawer1Unload;
```

Related: `MTT_PlaceTray`, `MTT_PickDrawer`

MTT_PlaceTray — Place into a standalone tray

Description

Places a part into a standalone tray. Same flow as `MTT_PickTray` but for load mode.

Syntax

```
MTT_PlaceTray gripper, load, tray
```

Parameters

Parameter	Type	Description
`gripper`	`MTT_GRIPPER`	The gripper to use
`load`	`loaddata`	The load data to remove after placing
`tray`	`MTT_TRAY`	The tray definition

Program execution

1. Set the current tray for load mode.
2. Check readiness. Prompt operator if finished.
3. Fetch approach offsets (`offsetPlaceInLoadTray`, `offsetPlaceOutLoadTray`).
4. Fetch work object, calculate target, call `MTT_Place`.
5. Increment counter. If finished, mark tray as NOT_READY.

Example

```
MTT_PlaceTray Gripper2, item1_finished, TrayDrawer2Load;
```

Related: `MTT_PickTray`, `MTT_PlaceDrawer`

6.1.4 Spindle operations

MTT_PickSpindle — Pick a part from a machine spindle

Description

Picks a part from the specified machine spindle. Waits for the machine door to be open, approaches the spindle, grips the part, opens the spindle chuck, verifies the grip, and departs.

Syntax

```
MTT_PickSpindle gripper, spindle, load [\wobj] [\blowoff]
```

Parameters

Parameter	Type	Description
`gripper`	`MTT_GRIPPER`	The gripper to use
`spindle`	`num`	Spindle number (1 or 2)
`load`	`loaddata`	Load data to add after picking
`\wobj`	`wobjdata`	Optional. Work object. Default: `wobj0`
`\blowoff`	`switch`	Optional. Enable blow-off during operation

Program execution

1. Fetch pick position from `pPickSpindle<N>`.
2. Fetch approach/depart offsets from `offsetPickInSpindle<N>`, `offsetPickOutSpindle<N>`.
3. Calculate approach and depart positions.
4. Fetch tool data.
5. Wait for machine door open.
6. MoveJ to approach at `vmax`, `z10`.
7. If `\blowoff`, enable blow-off.
8. Open gripper (prepare grip).
9. MoveL to pick position at `v100`, `fine`.
10. Grip and check grip feedback.
11. Open spindle chuck (call `MTT_OpenSpindle<N>` via `CallByVar`).
12. Add load.
13. MoveL to depart at `vmax`, `z10`.
14. If `\blowoff`, disable blow-off.

Example

```
MTT_PickSpindle Gripper2, 2, item1_finished \blowoff;
```

Related: `MTT_PlaceSpindle`, `MTT_Pick`, `MTT_OpenSpindle1`, `MTT_OpenSpindle2`

MTT_PlaceSpindle — Place a part into a machine spindle

Description

Places a part into the specified machine spindle. Waits for the machine door to be open, opens the spindle chuck, approaches, places the part, closes the spindle chuck, and departs.

Syntax

```
MTT_PlaceSpindle gripper, spindle, load [\wobj] [\blowoff]
[\releaseBefore]
```

Parameters

Parameter	Type	Description
`gripper`	`MTT_GRIPPER`	The gripper to use
`spindle`	`num`	Spindle number (1 or 2)
`load`	`loaddata`	Load data to remove after placing
`\wobj`	`wobjdata`	Optional. Work object. Default: `wobj0`
`\blowoff`	`switch`	Optional. Enable blow-off during operation
`\releaseBefore`	`switch`	Optional. Release the part before closing the spindle chuck

Program execution

1. Fetch place position from `pPlaceSpindle<N>`.
2. Fetch offsets from `offsetPlaceInSpindle<N>`, `offsetPlaceOutSpindle<N>`.
3. Calculate approach/depart.
4. Fetch tool data.
5. Wait for machine door open.
6. MoveJ to approach at `vmax`, `z10`.
7. If `\blowoff`, enable blow-off.
8. Open spindle chuck.
9. MoveL to place at `v100`, `fine`.
10. If `\releaseBefore`:
11. Release gripper first, then close spindle.
12. Else:
13. Close spindle first, then release gripper.
14. Remove load.
15. MoveL to depart at `vmax`, `z10`.
16. If `\blowoff`, disable blow-off.

Example

```
MTT_PlaceSpindle Gripper1, 1, item1_raw \blowoff \releaseBefore;
```

Related: `MTT_PickSpindle`, `MTT_Place`, `MTT_CloseSpindle1`, `MTT_CloseSpindle2`

6.1.5 Gripper control

MTT_Grip — Grip a part

Description

Activates the gripper to secure a part. The behavior depends on the grip direction defined in the `MTT_GRIPPER` record:

- **Inward** (`MTT_GRIP_INWARD`): closes the gripper jaws.
- **Outward** (`MTT_GRIP_OUTWARD`): opens the gripper (expanding mandrel).

A 200 ms delay is applied before and after the gripper command to allow the pneumatics to stabilize.

Syntax

```
MTT_Grip gripper
```

Parameters

Parameter	Type	Description
`gripper`	`MTT_GRIPPER`	The gripper to activate

Program execution

1. Wait 200 ms.
2. If inward: call `MTT_CloseGripper<cmdSet>` via `CallByVar`.
3. If outward: call `MTT_OpenGripper<cmdSet>` via `CallByVar`.
4. Wait 200 ms.

Related: `MTT_Release`, `MTT_PrepareGrip`, `MTT_checkGrip`

MTT_Release — Release a part

Description

Releases a part from the gripper. Inverse operation of `MTT_Grip`:

- **Inward**: opens the gripper jaws.
- **Outward**: closes the gripper.

A 200 ms pre-delay is applied.

Syntax

```
MTT_Release gripper
```

Parameters

Parameter	Type	Description
`gripper`	`MTT_GRIPPER`	The gripper to release

Related: `MTT_Grip`

MTT_checkGrip — Verify part is secured in gripper

Description

Checks the gripper feedback sensors to verify that a part is held. Calls `MTT_CheckItemInwardGripper<N>` or `MTT_CheckItemOutwardGripper<N>` based on the grip direction. If the check fails, the HAL procedure raises `ERR_MTT_ITEM_MISSED`.

Syntax

```
MTT_checkGrip gripper
```

Parameters

Parameter	Type	Description
`gripper`	`MTT_GRIPPER`	The gripper to verify

Related: `MTT_Grip`, `MTT_ItemMissedHandling`

6.1.6 Load management

MTT_addLoad — Add load to robot payload

Description

Adds the specified load to the current robot payload using the parallel-axis theorem for combined center of gravity and inertia calculation.

Syntax

```
MTT_addLoad loadToAdd
```

Parameters

Parameter	Type	Description
loadToAdd	loaddata	Load data to add

Related: MTT_removeLoad, MTT_setLoad, MTT_getLoad

MTT_removeLoad — Remove load from robot payload

Description

Removes the specified load from the current robot payload. If the remaining mass would be zero or negative, resets to `load0` (empty).

Syntax

```
MTT_removeLoad loadToRemove
```

Parameters

Parameter	Type	Description
`loadToRemove`	`loaddata`	Load data to remove

Related: `MTT_addLoad`, `MTT_setLoad`

MTT_setLoad — Set robot payload

Description

Sets the robot payload to the exact specified load, replacing the current accumulated load.

Syntax

```
MTT_setLoad loadToSet
```

Parameters

Parameter	Type	Description
loadToSet	loaddata	The new load data

Related: MTT_addLoad, MTT_removeLoad

MTT_getLoad — Read current robot payload

Description

Returns the current accumulated robot payload.

Syntax

```
loaddata := MTT_getLoad()
```

Return value: Current `loaddata`.

Related: `MTT_addLoad`, `MTT_setLoad`

6.1.7 Tray position management

MTT_resetTrayState — Reset tray position

Description

Resets the working tray position to [1,1,1] and sets the tray as READY. Can reset load tray, unload tray, or both.

Syntax

```
MTT_resetTrayState [\load | \unload]
```

Parameters

Parameter	Type	Description
\load	switch	Reset the load tray only
\unload	switch	Reset the unload tray only

If neither switch is provided, both load and unload trays are reset.

Related: MTT_resetTrayStates

MTT_resetTrayStates — Reset both tray positions

Description

Convenience procedure that resets both load and unload trays. Called automatically on controller RESET via the `CAB_EXEC_HOOKS` configuration.

Syntax

```
MTT_resetTrayStates
```

Related: `MTT_resetTrayState`

MTT_getCurrTrayPosNum — Get current tray position number

Description

Calculates and returns the sequential position number (1-based) for the given tray position, accounting for the increment mode (layer-first or stack-first).

Syntax

```
num := MTT_getCurrTrayPosNum(pos, tray)
```

Parameters

Parameter	Type	Description
pos	MTT_TRAY_POSITION	Current position [row, column, layer]
tray	MTT_TRAY	Tray definition (for dimensions and mode)

Return value: Sequential position number (1 to total positions).

6.1.8 Offset calculation

MTT_applyApproachOffset — Apply approach offset to a robtarget

Description

Applies both work-object and tool offsets to a robtarget. The work-object offset is applied first using `Offs`, then the tool offset using `RelTool`.

Syntax

```
robtarget := MTT_applyApproachOffset(pBase, offset)
```

Parameters

Parameter	Type	Description
`pBase`	`robtarget`	The base position to offset
`offset`	`MTT_APPROACH_OFFSET`	The offset to apply (contains both wobj and tool offsets)

Return value: The offset `robtarget`.

Example

```
VAR robtarget pApproach;
pApproach := MTT_applyApproachOffset(pPickSpindle1,
offsetPickInSpindle1);
```

6.1.9 Drawer helper functions

MTT_getOpenDrawerTarget — Get drawer open position

Description

Returns the `robtarget` for the open position of the specified drawer. Fetches by name `MTT_pOpenDrawer<N>` from the RAPID data.

Syntax

```
robtarget := MTT_getOpenDrawerTarget(drawer)
```

Parameters

Parameter	Type	Description
`drawer`	`num`	Drawer index (1–6)

Return value: `robtarget` for the open position.

MTT_getCloseDrawerTarget — Get drawer close position

Description

Returns the `robtarget` for the close position of the specified drawer.

Syntax

```
robtarget := MTT_getCloseDrawerTarget(drawer)
```

Parameters

Parameter	Type	Description
`drawer`	`num`	Drawer index (1–6)

Return value: `robtarget` for the close position.

6.1.10 Cycle management

MTT_ClockStart — Start cycle timer

Description

Resets and starts the cycle timer (`MTT_clock`). Called by `MTT_StartCycle`.

Syntax

```
MTT_ClockStart
```

MTT_ClockEnd — Stop cycle timer

Description

Stops the cycle timer and stores the elapsed time (in seconds) in `MTT_cycleTime`.

Syntax

```
MTT_ClockEnd
```

MTT_IncrProdCounter — Increment production counter

Description

Increments the persistent production counter (`MTT_prodCounter`) by 1.

Syntax

```
MTT_IncrProdCounter
```

6.1.11 Module loading

MTT_loadDataModule — Load configuration data module

Description

Loads `MTT_WizardData.modx` from `HOME:/MTT/` if it is not already loaded. Called automatically on controller RESET via `CAB_EXEC_HOOKS`.

Syntax

```
MTT_loadDataModule
```

6.1.12 Logging

MTT_logMessage — Log a message

Description

Logs a message to the system error log and WebApp. Supports three severity levels: Error (default), Warning (`\W`), and Info (`\I`). Supports localized text via text table resource IDs.

Syntax

```
MTT_logMessage msg [\W | \I] [\instTextNum] [\Arg1] [\Arg2] [\Arg3]
    [\Arg4] [\Arg5] [\Arg6] [\debug]
```

Parameters

Parameter	Type	Description
`msg`	`string`	Message text (ignored if `\instTextNum` is provided)
`\W`	`switch`	Warning severity
`\I`	`switch`	Info severity
`\instTextNum`	`num`	Text table resource number for localized text
`\Arg1–\Arg6`	`string`	Substitution arguments for localized text
`\debug`	`switch`	Only log if `MTT_Debug` is TRUE

Example

```
MTT_logMessage "Custom integration step completed" \I;
MTT_logMessage "" \W \instTextNum:=MTT_LOG_ITEM_MISSED;
```

6.1.13 WebApp display

MTT_webappDisplay — Display the WebApp on FlexPendant

Description

Opens the Machine Tending WebApp on the FlexPendant screen using `UIDisplay`.

Syntax

```
MTT_webappDisplay
```

6.1.14 Manual drawer operations

MTT_openManualDrawer — Open drawer from WebApp manual control

Description

Opens the drawer specified by the `MTT_manualOpDrawer` variable (set from the WebApp Manual Operation tab).

Syntax

```
MTT_openManualDrawer
```

MTT_closeManualDrawer — Close drawer from WebApp manual control

Description

Closes the drawer specified by `MTT_manualOpDrawer`.

Syntax

```
MTT_closeManualDrawer
```

6.2 Station HAL — MTT_User.sysx

These procedures are implemented differently per station variant. The following descriptions reference the **Omnivance** variant.

6.2.1 Home and cycle management

MTT_Home — Move robot to home position (HAL)

Description

Moves the robot to the predefined home joint position (`MTT_jHome`). Uses MoveAbsJ for safe joint motion regardless of current configuration.

Syntax

```
MTT_Home
```

Program execution

1. MoveAbsJ to `MTT_jHome` at `vmax`, `fine`.
2. Log home position reached.

Related: `MTT_StartCycle`, `MTT_EndCycle`

MTT_InitCycle — Initialize production cycle (HAL)

Description

Initializes the production cycle. Resets load tracking, prepares signals, and initializes the tray states.

Syntax

```
MTT_InitCycle
```

Related: `MTT_StartCycle`

MTT_StartCycle — Start a production cycle (HAL)

Description

Starts a new production cycle. Resets the robot payload to `load0`, starts the cycle clock.

Syntax

```
MTT_StartCycle
```

Program execution

1. Set load to `load0` (empty payload).
2. Call `MTT_ClockStart` to begin timing.

Related: `MTT_EndCycle`, `MTT_ClockStart`

MTT_EndCycle — End a production cycle (HAL)

Description

Ends the current production cycle. Stops the clock, increments the production counter, and checks if this is the last cycle.

Syntax

```
MTT_EndCycle
```

Program execution

1. Call `MTT_ClockEnd` to stop timing.
2. Call `MTT_IncrProdCounter`.
3. If `MTT_LastCycle` is TRUE:

a. Set load to `load0`. b. Close all open drawers (`MTT_CloseAnyOpenDrawer`). c. Move home (`MTT_Home`). d. Exit the cycle loop.

Related: `MTT_StartCycle`, `MTT_ClockEnd`, `MTT_IncrProdCounter`

6.2.2 Drawer operations

MTT_OpenDrawer — Open a drawer (HAL)

Description

Opens the specified drawer using the robot with a magnetic attach mechanism.

Syntax

```
MTT_OpenDrawer drawer
```

Parameters

Parameter	Type	Description
`drawer`	`num`	Drawer index (1–6)

Program execution (Omnivance variant)

1. Fetch open and close positions for the specified drawer.
2. Calculate approach position (100 mm offset behind closed position).
3. MoveJ to approach at `vmax`, `z10`.
4. Temporarily raise MotionSup to 300%, reduce acceleration to 8%.
5. MoveL to closed position at `v100`, `fine`.
6. Activate magnet (`doMTT_ActivateMagnet`).
7. MoveL to open position at `v100`, `fine`.
8. Release magnet.
9. Restore MotionSup and acceleration.
10. MoveJ to retract at `vmax`, `z10`.
11. Update `MTT_DrawersState{drawer}.open := MTT_DRAWER_OPEN`.
12. Log drawer opened.

Related: `MTT_CloseDrawer`, `MTT_GripDrawer`, `MTT_ReleaseDrawer`

MTT_CloseDrawer — Close a drawer (HAL)

Description

Closes the specified drawer using the robot with a magnetic attach mechanism.

Syntax

```
MTT_CloseDrawer drawer
```

Parameters

Parameter	Type	Description
`drawer`	`num`	Drawer index (1–6)

Program execution (Omnivance variant)

1. Fetch open and close positions.
2. Calculate approach (100 mm offset behind open position).
3. MoveJ to approach at `vmax`, `z10`.
4. MotionSup to 300%, acceleration to 8%.
5. MoveL to open position, activate magnet.
6. MoveL to closed position.
7. Release magnet.
8. Restore MotionSup and acceleration.
9. MoveJ to retract.
10. Update `MTT_DrawersState{drawer}.open := MTT_DRAWER_CLOSED`.
11. Log drawer closed.

Related: `MTT_OpenDrawer`

MTT_CloseAnyOpenDrawer — Close the first open drawer found (HAL)

Description

Iterates through all drawers and closes the first one that has state `MTT_DRAWER_OPEN`. Used during `MTT_EndCycle` with `MTT_LastCycle` to ensure all drawers are closed when production stops.

Syntax

```
MTT_CloseAnyOpenDrawer
```

Related: `MTT_CloseDrawer`, `MTT_EndCycle`

MTT_GripDrawer — Activate magnet for drawer (HAL)

Description

Activates the drawer magnet to grip the drawer handle. Low-level procedure used by `MTT_OpenDrawer` and `MTT_CloseDrawer`.

Syntax

```
MTT_GripDrawer
```

Related: `MTT_ReleaseDrawer`

MTT_ReleaseDrawer — Deactivate magnet for drawer (HAL)

Description

Deactivates the drawer magnet to release the drawer handle.

Syntax

```
MTT_ReleaseDrawer
```

Related: `MTT_GripDrawer`

MTT_NoDrawerForLoad — Handle no drawer available for loading (HAL)

Description

Called when `MTT_selectActiveDrawer` returns 0 during a place-to-drawer operation. Logs a warning message and stops program execution, requiring operator intervention.

Syntax

```
MTT_NoDrawerForLoad
```

Related: `MTT_NoDrawerForUnload`, `MTT_PlaceDrawer`

MTT_NoDrawerForUnload — Handle no drawer available for unloading (HAL)

Description

Called when `MTT_selectActiveDrawer` returns 0 during a pick-from-drawer operation. Logs a warning and stops.

Syntax

```
MTT_NoDrawerForUnload
```

Related: `MTT_NoDrawerForLoad`, `MTT_PickDrawer`

6.3 Machine HAL — MTT_Machine.sysx

These procedures are implemented differently per machine variant. The following descriptions reference the **MachineX** variant.

6.3.1 Machine door

MTT_OpenMachineDoor — Open the machine door (HAL)

Description

Robot physically opens the machine door using motion, magnet attach/detach, and feedback verification.

Syntax

```
MTT_OpenMachineDoor
```

Program execution (MachineX variant)

1. Switch tool to `MTT_tOpenClose`.
2. Calculate approach (100 mm behind door closed position).
3. MoveJ to approach at `vmax`, `z10`.
4. MotionSup to 300%, acceleration to 8%.
5. MoveL to door closed position at `v100`, `fine`.
6. Activate magnet.
7. MoveL to door open position at `v100`, `fine`.
8. Release magnet.
9. Restore MotionSup and acceleration.
10. MoveJ to retract.
11. Wait for `diMTT_MachineDoorOpen` feedback with 5-second timeout.

Related: `MTT_CloseMachineDoor`, `MTT_WaitForMachineDoorOpen`

MTT_CloseMachineDoor — Close the machine door (HAL)

Description

Robot physically closes the machine door using motion and magnet mechanism.

Syntax

```
MTT_CloseMachineDoor
```

Program execution (MachineX variant)

1. Switch tool to `MTT_tOpenClose`.
2. Approach door open position.
3. Activate magnet, move to closed position.
4. Release magnet.
5. Wait for `diMTT_MachineDoorClosed` feedback.

Related: `MTT_OpenMachineDoor`, `MTT_WaitForMachineDoorClosed`

MTT_WaitForMachineDoorOpen — Wait for door open feedback (HAL)

Description

Waits for the `diMTT_MachineDoorOpen` signal to become active, with a 5-second timeout. If the timeout expires, logs an error and stops.

Syntax

```
MTT_WaitForMachineDoorOpen
```

Related: `MTT_OpenMachineDoor`

MTT_WaitForMachineDoorClosed — Wait for door closed feedback (HAL)

Description

Waits for the `diMTT_MachineDoorClosed` signal to become active, with a 5-second timeout.

Syntax

```
MTT_WaitForMachineDoorClosed
```

Related: `MTT_CloseMachineDoor`

MTT_MachineDoorOpen — Check if door is open (HAL)

Description

Returns TRUE if `diMTT_MachineDoorOpen` is active.

Syntax

```
bool := MTT_MachineDoorOpen()
```

MTT_MachineDoorClosed — Check if door is closed (HAL)

Description

Returns TRUE if `diMTT_MachineDoorClosed` is active.

Syntax

```
bool := MTT_MachineDoorClosed()
```

6.3.2 Spindle/chuck

MTT_OpenSpindle1 / MTT_OpenSpindle2 — Open spindle chuck (HAL)

Description

Opens the specified spindle chuck. Pulses the open command signal and waits for the closed feedback signal to become inactive (confirming the chuck has opened). Timeout: 3 seconds.

Syntax

```
MTT_OpenSpindle1
MTT_OpenSpindle2
```

Program execution

1. `PulseDO` on `doMTT_OpenChuckSpindle<N>`.
2. `WaitUntil diMTT_ChuckSpindle<N>Closed = 0` with `\MaxTime:=3`.
3. If timeout: log error, stop.

Related: `MTT_CloseSpindle1`, `MTT_CloseSpindle2`

MTT_CloseSpindle1 / MTT_CloseSpindle2 — Close spindle chuck (HAL)

Description

Closes the specified spindle chuck. Pulses the close command signal and waits for the closed feedback signal. Timeout: 3 seconds.

Syntax

```
MTT_CloseSpindle1
MTT_CloseSpindle2
```

Program execution

1. `PulseDO` on `doMTT_CloseChuckSpindle<N>`.
2. `WaitUntil diMTT_ChuckSpindle<N>Closed = 1` with `\MaxTime:=3`.
3. If timeout: log error, stop.

Related: `MTT_OpenSpindle1`, `MTT_OpenSpindle2`

MTT_Spindle1Closed / MTT_Spindle2Closed — Check spindle state (HAL)

Description

Returns TRUE if the spindle chuck is in the closed position.

Syntax

```
bool := MTT_Spindle1Closed()
bool := MTT_Spindle2Closed()
```

6.3.3 Machine cycle

MTT_WaitMachine — Wait for machine ready (HAL)

Description

Moves the robot to the wait position (`MTT_pWaitMachine`) and waits until the machine signals ready.

Syntax

```
MTT_WaitMachine
```

Program execution (MachineX variant)

1. Switch tool to `MTT_tGripper1`.
2. MoveJ to `MTT_pWaitMachine` at `vmax`, `fine`.
3. Wait until `diMTT_MachineReady = 1`.
4. Also monitor `diMTT_OptionalGoHome` (DI15): if active, move home and wait there.

Related: `MTT_StartMachine`, `MTT_EnterMachine`, `MTT_MachineReady`

MTT_StartMachine — Start machine cycle (HAL)

Description

Sends a pulse to start the machine cycle.

Syntax

```
MTT_StartMachine
```

Program execution

1. `PulseDO` on `doMTT_StartMachineCycle`.

Related: `MTT_WaitMachine`

MTT_FeedHoldMachine — Feed hold (HAL)

Description

Placeholder procedure for feed hold control. Integrators implement this for machines that support mid-cycle pausing.

Syntax

```
MTT_FeedHoldMachine
```

MTT_EnterMachine — Enter machine work envelope (HAL)

Description

Waits for the machine to be ready, then moves the robot into the machine work area.

Syntax

```
MTT_EnterMachine
```

Program execution

1. Call `MTT_WaitMachine`.
2. MoveL to `MTT_pInMachine` at `vmax`, `z10`.

Related: `MTT_ExitMachine`, `MTT_WaitMachine`

MTT_ExitMachine — Exit machine work envelope (HAL)

Description

Moves the robot out of the machine work area and back to the wait position.

Syntax

```
MTT_ExitMachine
```

Program execution

1. MoveJ to `MTT_pInMachine` at `vmax`, `z10`.
2. Call `MTT_WaitMachine`.

Related: `MTT_EnterMachine`

MTT_MachineReady — Check machine ready state (HAL)

Description

Returns TRUE if `diMTT_MachineReady` is active.

Syntax

```
bool := MTT_MachineReady()
```

6.4 Gripper HAL — MTT_EndEffector.sysx

These procedures are implemented per station/gripper variant. The following descriptions reference the **Omnivance** variant.

6.4.1 Gripper open/close

MTT_OpenGripper1 / MTT_OpenGripper2 — Open gripper (HAL)

Description

Opens the specified gripper by setting the control signal low and enabling the pneumatic valve.

Syntax

```
MTT_OpenGripper1
MTT_OpenGripper2
```

Program execution

1. Set `doMTT_EnableGripper` high.
2. Set `doMTT_CloseGripper01` (or `02`) low.

Related: `MTT_CloseGripper1`, `MTT_CloseGripper2`

MTT_CloseGripper1 / MTT_CloseGripper2 — Close gripper (HAL)

Description

Closes the specified gripper by setting the control signal high.

Syntax

```
MTT_CloseGripper1
MTT_CloseGripper2
```

Program execution

1. Set `doMTT_EnableGripper` high.
2. Set `doMTT_CloseGripper01` (or `02`) high.

Related: `MTT_OpenGripper1`, `MTT_OpenGripper2`

6.4.2 Gripper feedback

MTT_Gripper1Closed / MTT_Gripper2Closed — Check closed state (HAL)

Description

Returns TRUE if the gripper position sensor indicates the closed state.

Syntax

```
bool := MTT_Gripper1Closed()
bool := MTT_Gripper2Closed()
```

MTT_Gripper1Open / MTT_Gripper2Open — Check open state (HAL)

Description

Returns TRUE if the gripper position sensor indicates the open state.

Syntax

```
bool := MTT_Gripper1Open()
bool := MTT_Gripper2Open()
```

6.4.3 Item verification

MTT_CheckItemInwardGripper1 / MTT_CheckItemInwardGripper2 — Check inward grip (HAL)

Description

Checks that the gripper (closing inward) has successfully secured a part. Reads the closed-position sensor. If the gripper did not reach the closed position (part is present between jaws), the check passes. If the gripper is fully closed (no part), raises `ERR_MTT_ITEM_MISSED`.

Syntax

```
MTT_CheckItemInwardGripper1
MTT_CheckItemInwardGripper2
```

Related: `MTT_CheckItemOutwardGripper1`, `MTT_ItemMissedHandling`

MTT_CheckItemOutwardGripper1 / MTT_CheckItemOutwardGripper2 — Check outward grip (HAL)

Description

Checks that the gripper (expanding outward) has secured a part. Logic is inverted from inward check.

Syntax

```
MTT_CheckItemOutwardGripper1
MTT_CheckItemOutwardGripper2
```

MTT_ItemMissedHandling — Handle missed item (HAL)

Description

Called when `ERR_MTT_ITEM_MISSED` is raised during a pick operation. Default implementation logs a warning. Integrators should customize this to define recovery behavior (retry, alarm, stop).

Syntax

```
MTT_ItemMissedHandling
```

Related: `MTT_checkGrip`

6.4.4 Blow-off

MTT_EnableBlowOff — Enable air nozzle (HAL)

Description

Activates the blow-off nozzle by setting `doMTT_NozzleOn` low (active low in Omnivance).

Syntax

```
MTT_EnableBlowOff
```

Related: `MTT_DisableBlowOff`

MTT_DisableBlowOff — Disable air nozzle (HAL)

Description

Deactivates the blow-off nozzle by setting `doMTT_NozzleOn` high.

Syntax

```
MTT_DisableBlowOff
```

Related: `MTT_EnableBlowOff`

6.5 Data types

6.5.1 MTT_TRAY — Tray layout definition

Description

Defines the physical layout and parameters of a tray grid.

```
RECORD MTT_TRAY
    num numRowsX;          ! Number of positions in X direction
    num numColumnsY;       ! Number of positions in Y direction
    num numLayersZ;        ! Number of layers in Z direction
    num spacingX;          ! Distance between X positions (mm)
    num spacingY;          ! Distance between Y positions (mm)
    num spacingZ;          ! Distance between Z positions (mm)
    num offsetXEven;       ! X offset for even-numbered Y rows
    num offsetYEven;       ! Y offset for even-numbered X rows
    num incMode;           ! 0 = layer-first, 1 = stack-first
    robtarget firstTarget; ! Position and orientation of slot [1,1,1]
    string wobjdataReference; ! Name of the work object variable
ENDRECORD
```

6.5.2 MTT_GRIPPER — Gripper definition

Description

Defines a gripper's properties and hardware mapping.

```
RECORD MTT_GRIPPER
    string tooldataReference; ! Name of the tooldata variable
    num gripperCmdSet;        ! Command set (1 or 2)
    num gripDirection;        ! 0 = inward, 1 = outward
    num offsetZTCP;           ! Additional Z offset on TCP (mm)
ENDRECORD
```

6.5.3 MTT_APPROACH_OFFSET — Approach/depart offset

Description

Defines a 6-DOF offset split into work-object and tool coordinate systems.

```
RECORD MTT_APPROACH_OFFSET
    num wobjOffsx;   ! Work object X offset (mm)
    num wobjOffsy;   ! Work object Y offset (mm)
    num wobjOffsz;   ! Work object Z offset (mm)
    num toolOffsx;   ! Tool X offset (mm)
    num toolOffsy;   ! Tool Y offset (mm)
    num toolOffsz;   ! Tool Z offset (mm)
ENDRECORD
```

6.5.4 MTT_TRAY_POSITION — Tray position index

Description

Tracks the current position within a tray grid.

```
RECORD MTT_TRAY_POSITION
    num posRowX;     ! Current row (X), 1-based
    num posColumnY;  ! Current column (Y), 1-based
    num posLayerZ;   ! Current layer (Z), 1-based
ENDRECORD
```

6.5.5 MTT_DRAWER_STATE — Drawer state

Description

Tracks the runtime state of a single drawer.

```
RECORD MTT_DRAWER_STATE
    num open;                          ! MTT_DRAWER_OPEN (1) or
MTT_DRAWER_CLOSED (0)
    num ready;                         ! MTT_DRAWER_READY (1) or
MTT_TRAY_NOT_READY (0)
    num disabled;                      ! MTT_DRAWER_ENABLED (0) or
MTT_DRAWER_DISABLED (1)
    num priority;                      ! Selection priority (lower =
higher priority)
    MTT_TRAY_POSITION trayPosLoad;     ! Current load tray position
    MTT_TRAY_POSITION trayPosUnload;   ! Current unload tray position
ENDRECORD
```

6.5.6 MTT_TRAY_INFO — Runtime tray information

Description

Combines a tray definition with its runtime state for the currently active tray.

```
RECORD MTT_TRAY_INFO
    string name;             ! Name of the tray variable
    MTT_TRAY value;          ! Tray layout definition
    MTT_TRAY_POSITION pos;   ! Current position in the tray
    num ready;               ! MTT_TRAY_READY (1) or
MTT_TRAY_NOT_READY (0)
    bool isDrawer;           ! TRUE if this tray belongs to a drawer
ENDRECORD
```

6.6 Constants and enumerations

6.6.1 Grip direction

Constant	Value	Description
MTT_GRIP_INWARD	0	Close to grip (parallel jaw)
MTT_GRIP_OUTWARD	1	Open to grip (expanding mandrel)

6.6.2 Tray increment mode

Constant	Value	Description
MTT_TRAY_LAYER_FIRST	0	Iterate X → Y → Z
MTT_TRAY_STACK_FIRST	1	Iterate Z → X → Y

6.6.3 Drawer state

Constant	Value	Description
MTT_DRAWER_OPEN	1	Drawer is open
MTT_DRAWER_CLOSED	0	Drawer is closed
MTT_DRAWER_READY	1	Tray is present and available
MTT_TRAY_NOT_READY	0	Tray not ready
MTT_DRAWER_ENABLED	0	Drawer active
MTT_DRAWER_DISABLED	1	Drawer skipped

6.6.4 Gripper command sets

Constant	Value	Procedures
MTT_GRIP_CMD_SET_1	1	MTT_OpenGripper1, MTT_CloseGripper1
MTT_GRIP_CMD_SET_2	2	MTT_OpenGripper2, MTT_CloseGripper2

6.7 Error codes

Error variable	Description
ERR_MTT_ITEM_MISSED	Raised by MTT_CheckItemInward/OutwardGripper<N> when gripper feedback indicates the part was not secured. Caught in MTT_Pick and MTT_PickDrawer error handlers.

6.8 Persistent state variables

Variable	Type	Description
MTT_DrawersState{6}	MTT_DRAWER_STATE	Runtime state of all 6 drawers
MTT_TrayLoad	MTT_TRAY_INFO	Currently active load tray
MTT_TrayUnload	MTT_TRAY_INFO	Currently active unload tray
MTT_currDrawerLoad	num	Index of the current load drawer
MTT_currDrawerUnLoad	num	Index of the current unload drawer
MTT_cycleTime	num	Last cycle time in seconds
MTT_prodCounter	num	Total completed production cycles
MTT_LastCycle	bool	Flag — stop after current cycle completes
MTT_Debug	bool	Flag — enable debug logging
MTT_Load	loaddata	Current accumulated robot payload
MTT_manualOpDrawer	num	Drawer number for manual WebApp operations

7 Program examples

This chapter provides annotated production cycle examples, from the default template cycle to customized variants, demonstrating how to compose Wizard blocks and RAPID procedures for different machine tending scenarios.

7.1 Default production cycle

This is the cycle shipped with the template in `Wizard.modx`. It demonstrates a dual-spindle, dual-gripper workflow.

7.1.1 RAPID code

```
MODULE Wizard

    PROC main()
        MTT_Home;
        WHILE TRUE DO
            MTT_StartCycle;
            MTT_PickDrawer Gripper1, item1_raw;
            MTT_waitMachine;
            MTT_OpenMachineDoor;
            MTT_EnterMachine;
            MTT_PlaceSpindle Gripper1, 1, item1_raw, \BlowOff
\releaseBefore;
            MTT_PickSpindle Gripper2, 2, item1_finished, \BlowOff;
            MTT_ExitMachine;
            MTT_CloseMachineDoor;
            MTT_startMachine;
            MTT_PlaceDrawer Gripper2, item1_finished;
            MTT_EndCycle;
        ENDWHILE
    ENDPROC

ENDMODULE
```

7.1.2 Step-by-step walkthrough

Step	RAPID call	Description
1	`MTT_Home`	Move robot to home joint position. Executed once at startup.
2	`MTT_StartCycle`	Reset payload to `load0`, start cycle timer.
3	`MTT_PickDrawer Gripper1, item1_raw`	Select the best unload drawer (auto), open if closed, pick a raw part using Gripper 1. Load `item1_raw` (0.7 kg) is added to the robot payload.
4	`MTT_waitMachine`	Move to the wait position near the machine. Wait until `diMTT_MachineReady` signal is active (machine has finished its previous cycle).
5	`MTT_OpenMachineDoor`	Robot grips the machine door with the magnet, moves it from closed to open position, releases. Waits for door open feedback.
6	`MTT_EnterMachine`	Move robot into the machine work envelope (`MTT_pInMachine`).
7	`MTT_PlaceSpindle Gripper1, 1, item1_raw, \BlowOff \releaseBefore`	Open spindle 1 chuck, blow air on the spindle face, move to place position, release Gripper 1 *before* closing the

		chuck (raw part sits in the chuck unsupported briefly), close spindle 1 chuck, remove `item1_raw` from payload, retract, stop blow-off.
8	`MTT_PickSpindle Gripper2, 2, item1_finished, \BlowOff`	Blow air, open spindle 2 chuck, move to pick position, grip with Gripper 2, verify grip feedback, open spindle 2, add `item1_finished` (0.5 kg) to payload, retract, stop blow-off.
9	`MTT_ExitMachine`	Move robot out of the machine work area back to the wait position.
10	`MTT_CloseMachineDoor`	Robot grips the door, moves it from open to closed, releases. Waits for door closed feedback.
11	`MTT_startMachine`	Pulse `doMTT_StartMachineCycle` to trigger the machine cycle for the newly loaded raw part on spindle 1.
12	`MTT_PlaceDrawer Gripper2, item1_finished`	Select the best load drawer (auto), open if closed, place the finished part using Gripper 2, remove `item1_finished` from payload. If the tray is full, close the drawer.
13	`MTT_EndCycle`	Stop timer, increment production counter. If `MTT_LastCycle` is set, close drawers, go home, and exit loop.

7.1.3 Cycle timing breakdown

Typical cycle time breakdown for the default Omnivance setup:

Phase	**Approximate time**
Pick from drawer	3–5 s
Wait for machine	Variable (depends on machine cycle)
Open door + enter	4–6 s
Place spindle 1 + pick spindle 2	6–10 s
Exit + close door	4–6 s
Start machine	< 1 s
Place to drawer	3–5 s
Total robot time	**~20–32 s** (excluding machine wait)

7.2 Single-spindle cycle

A simplified cycle for machines with only one spindle. The robot picks a raw part, waits for the machine, swaps parts at a single spindle, and places the finished part back.

7.2.1 RAPID code

```
PROC main()
    MTT_Home;
    WHILE TRUE DO
        MTT_StartCycle;

        ! Pick raw part from drawer
        MTT_PickDrawer Gripper1, item1_raw;

        ! Wait for machine to finish
        MTT_waitMachine;
        MTT_OpenMachineDoor;
        MTT_EnterMachine;

        ! Pick finished part from spindle 1 with Gripper 2
        MTT_PickSpindle Gripper2, 1, item1_finished, \BlowOff;

        ! Place raw part to spindle 1 with Gripper 1
        MTT_PlaceSpindle Gripper1, 1, item1_raw, \BlowOff
\releaseBefore;

        ! Exit and start machine
        MTT_ExitMachine;
        MTT_CloseMachineDoor;
        MTT_startMachine;

        ! Place finished part to drawer
        MTT_PlaceDrawer Gripper2, item1_finished;

        MTT_EndCycle;
    ENDWHILE
ENDPROC
```

7.2.2 Key differences from default

- Uses only **spindle 1** for both pick and place.
- **Pick before place**: the finished part is removed first, then the raw part is placed. This requires the dual gripper — one gripper holds the raw part while the other picks the finished part.
- Machine cycle processes only one part at a time (no pipeline).

Program examples

7.3 Cycle with separate load and unload drawers

A cycle where raw parts and finished parts use different drawers, with explicit drawer closing after the unload tray is emptied.

7.3.1 RAPID code

```
PROC main()
    MTT_Home;
    WHILE TRUE DO
        MTT_StartCycle;

        ! Pick raw part — close drawer when tray is empty
        MTT_PickDrawer Gripper1, item1_raw \closeWhenFinished;

        MTT_waitMachine;
        MTT_OpenMachineDoor;
        MTT_EnterMachine;
        MTT_PlaceSpindle Gripper1, 1, item1_raw, \BlowOff
\releaseBefore;
        MTT_PickSpindle Gripper2, 2, item1_finished, \BlowOff;
        MTT_ExitMachine;
        MTT_CloseMachineDoor;
        MTT_startMachine;

        ! Place finished part
        MTT_PlaceDrawer Gripper2, item1_finished;

        MTT_EndCycle;
    ENDWHILE
ENDPROC
```

7.3.2 Notes

- The `\closeWhenFinished` switch on `MTT_PickDrawer` closes the unload drawer when the last raw part is picked from it. This signals the operator that the tray is empty and can be replaced.
- The load drawer (for finished parts) is automatically closed by `MTT_PlaceDrawer` when its tray is full (default behavior).
- Drawer priority settings in the WebApp determine which drawer is selected first. If drawer 1 and drawer 2 are both set to priority 1, the algorithm picks the first open drawer, or the first ready+enabled drawer.

7.4 Cycle with blow-off cleaning station

A cycle that includes a blow-off cleaning step after picking from the spindle, using the dedicated blow-off position on the Omnivance station.

7.4.1 RAPID code

```
PROC main()
    MTT_Home;
    WHILE TRUE DO
        MTT_StartCycle;
        MTT_PickDrawer Gripper1, item1_raw;
        MTT_waitMachine;
        MTT_OpenMachineDoor;
        MTT_EnterMachine;
        MTT_PlaceSpindle Gripper1, 1, item1_raw, \releaseBefore;
        MTT_PickSpindle Gripper2, 2, item1_finished;
        MTT_ExitMachine;
        MTT_CloseMachineDoor;
        MTT_startMachine;

        ! Move to blow-off station and clean the finished part
        MTT_StartBlowOff;
        ! (Robot moves to blow-off position — custom motion)
        MoveJ BlowOffApproach, vmax, z10, MTT_tBlowOff;
        MoveL BlowOffPosition, v100, fine, MTT_tBlowOff;
        WaitTime 2;  ! Hold at blow-off position for 2 seconds
        MoveJ BlowOffApproach, vmax, z10, MTT_tBlowOff;
        MTT_StopBlowOff;

        MTT_PlaceDrawer Gripper2, item1_finished;
        MTT_EndCycle;
    ENDWHILE
ENDPROC
```

7.4.2 Notes

- `MTT_StartBlowOff` enables the air nozzle (`doMTT_NozzleOn`).
- The blow-off positions (`BlowOffApproach`, `BlowOffPosition`) are defined in `MTT_User.sysx` for the Omnivance station.
- The hold time at the blow-off position depends on the part geometry and contamination level.
- `MTT_StopBlowOff` disables the nozzle.
- This example uses the blow-off *outside* the spindle operations (no `\BlowOff` switch on `MTT_PickSpindle`), which is appropriate when a dedicated cleaning step is needed rather than a quick spindle-face blow.

7.5 Cycle with standalone tray (no drawers)

A cycle that uses standalone trays instead of drawers. Useful for fixed pallet positions where the operator loads/unloads trays while the robot is working on the machine.

7.5.1 RAPID code

```
PROC main()
    MTT_Home;
    WHILE TRUE DO
        MTT_StartCycle;

        ! Pick from a standalone unload tray
        MTT_PickTray Gripper1, item1_raw, TrayDrawer1Unload;

        MTT_waitMachine;
        MTT_OpenMachineDoor;
        MTT_EnterMachine;
        MTT_PlaceSpindle Gripper1, 1, item1_raw, \BlowOff
\releaseBefore;
        MTT_PickSpindle Gripper2, 2, item1_finished, \BlowOff;
        MTT_ExitMachine;
        MTT_CloseMachineDoor;
        MTT_startMachine;

        ! Place to a standalone load tray
        MTT_PlaceTray Gripper2, item1_finished, TrayDrawer2Load;

        MTT_EndCycle;
    ENDWHILE
ENDPROC
```

7.5.2 Notes

- `MTT_PickTray` and `MTT_PlaceTray` use the same tray definitions as the drawer operations but without the automatic drawer selection and open/close mechanism.
- When the tray is finished, the operator is prompted with a dialog: **Reset** (return to position [1,1,1]) or **Cancel** (go home and exit cycle).
- The tray work object must be calibrated for the fixed tray position (not a drawer workobject).
- Approach offsets are read from `offsetPickInUnloadTray` / `offsetPlaceInLoadTray` (separate from the drawer offsets).

7.6 Cycle with user-defined pick and place positions

A cycle that uses generic pick/place positions for custom integration scenarios (e.g., picking from a conveyor, placing onto an inspection station).

7.6.1 RAPID code

```
PROC main()
    MTT_Home;
    WHILE TRUE DO
        MTT_StartCycle;

        ! Pick from a user-defined position
        MTT_Pick pPickUser, Gripper1, item1_raw
            \offsetIN:=offsetPickInUser
            \offsetOUT:=offsetPickOutUser;

        MTT_waitMachine;
        MTT_OpenMachineDoor;
        MTT_EnterMachine;
        MTT_PlaceSpindle Gripper1, 1, item1_raw, \BlowOff
\releaseBefore;
        MTT_PickSpindle Gripper2, 2, item1_finished, \BlowOff;
        MTT_ExitMachine;
        MTT_CloseMachineDoor;
        MTT_startMachine;

        ! Place at a user-defined position
        MTT_Place pPlaceUser, Gripper2, item1_finished
            \offsetIN:=offsetPlaceInUser
            \offsetOUT:=offsetPlaceOutUser;

        MTT_EndCycle;
    ENDWHILE
ENDPROC
```

7.6.2 Notes

- `MTT_Pick` and `MTT_Place` are the lowest-level pick/place procedures. All higher-level operations (`MTT_PickDrawer`, `MTT_PickSpindle`, `MTT_PickTray`) ultimately use these.
- `pPickUser` and `pPlaceUser` are taught through the WebApp Targets tab.
- Approach offsets are configured through the WebApp Approach Offsets tab.
- The `\wobj` parameter can be added to specify a work object other than `wobj0`.

7.7 Customization guidelines

7.7.1 Adding a new operation to the cycle

1. **Wizard blocks**: Open the Wizard editor, enable Edit mode, drag the appropriate block from the palette to the desired position in the sequence.
2. **RAPID code**: Add the procedure call to the desired position in the `main()` procedure of `Wizard.modx`.

7.7.2 Adding custom logic between operations

Use the availables blocks in the Wizard editor to insert arbitrary RAPID statements between MTT blocks. For example:

```
! Wait for external signal before proceeding
WaitDI diCustomReady, 1;
```

7.7.3 Adding a third gripper

1. Define a new `MTT_GRIPPER` record in the WebApp Grippers tab (e.g., `Gripper3`).
2. Implement `MTT_OpenGripper3`, `MTT_CloseGripper3`, and `MTT_CheckItem*Gripper3` in `MTT_EndEffector.sysx`.
3. Set `Gripper Command` to 1 or 2 to select which of the two available hardware control procedures to use.
4. The new gripper appears automatically in all Wizard block dropdowns.

7.7.4 Modifying machine behavior

To adapt the template for a different machine type:

1. Create a new copy of `MTT_Machine.sysx`.
2. Implement all procedures in the machine HAL interface (see section 5.3).
3. Update the installation script or manually replace the module in `HOME:/MTT/Machines`.
4. Test all machine operations using the WebApp Manual Operation tab.

7.7.5 Creating a program variant

The WebApp Gantry tab provides a **Save as new program** feature that creates a copy of the current program with applied offsets. This is useful for:

- Gantry installations with height offsets.
- Multiple part variants with different TCP offsets.
- Backup configurations for different product runs.

Program examples

8 AppStudio project shipped with the Add-In

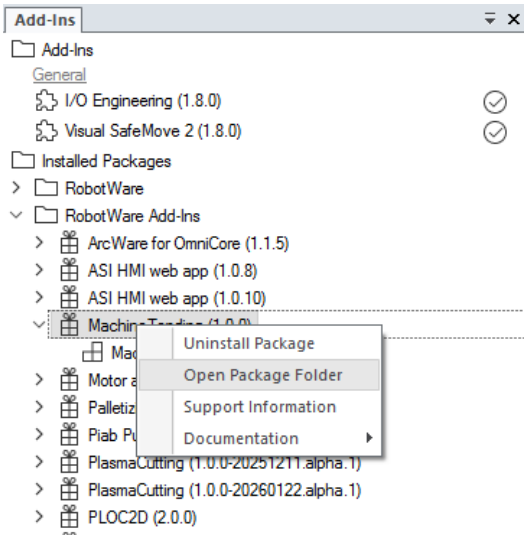
This page explains how to locate the AppStudio project that ships with the Machine Tending Template Add-In, make a local copy, edit it in AppStudio, and redeploy the WebApp.

8.1 Overview

The Add-In delivers the AppStudio project inside the distribution/package installation folder. That folder is managed by the Add-In and may be overwritten by updates or reinstalls, so it is not intended for direct editing. Create a local working copy of the AppStudio project, perform and validate your edits locally, and only deploy the tested WebApp from your local copy.

8.1.1 Steps to open the package folder and locate the project

1. Open RobotStudio and click the **Add-Ins** tab in the ribbon. The Add-Ins panel opens on the left, showing installed packages.
2. Locate the Machine Tending Template package in the installed packages list.
3. Right-click the package and choose **Open Package Folder** (this opens the package installation folder on disk).

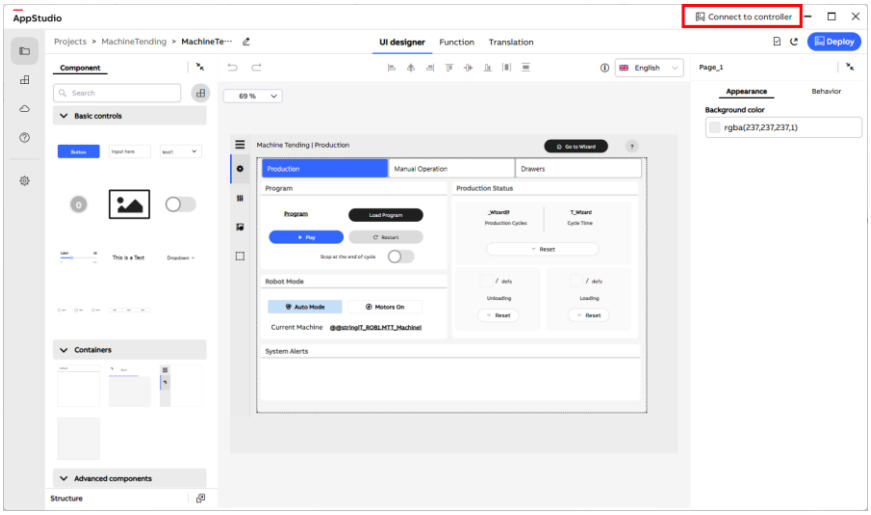


Open Package Folder context menu

4. In the opened folder you will find an `AppStudio` subfolder. Inside it is the `MachineTending.asppag` file — this is the AppStudio project for the Machine Tending Template.

8.1.2 Steps to create a local copy and open in AppStudio

1. Copy the AppStudio project folder or `MachineTending.asppag` file from the package folder to a local working folder on your PC (for example `C:\Work\MTT-AppStudio-Local\`).
2. Open AppStudio and choose **Open project** and point to your local copy.
3. Once opened you will see the `MachineTending` project in the Projects list — select it to enter the UI designer.



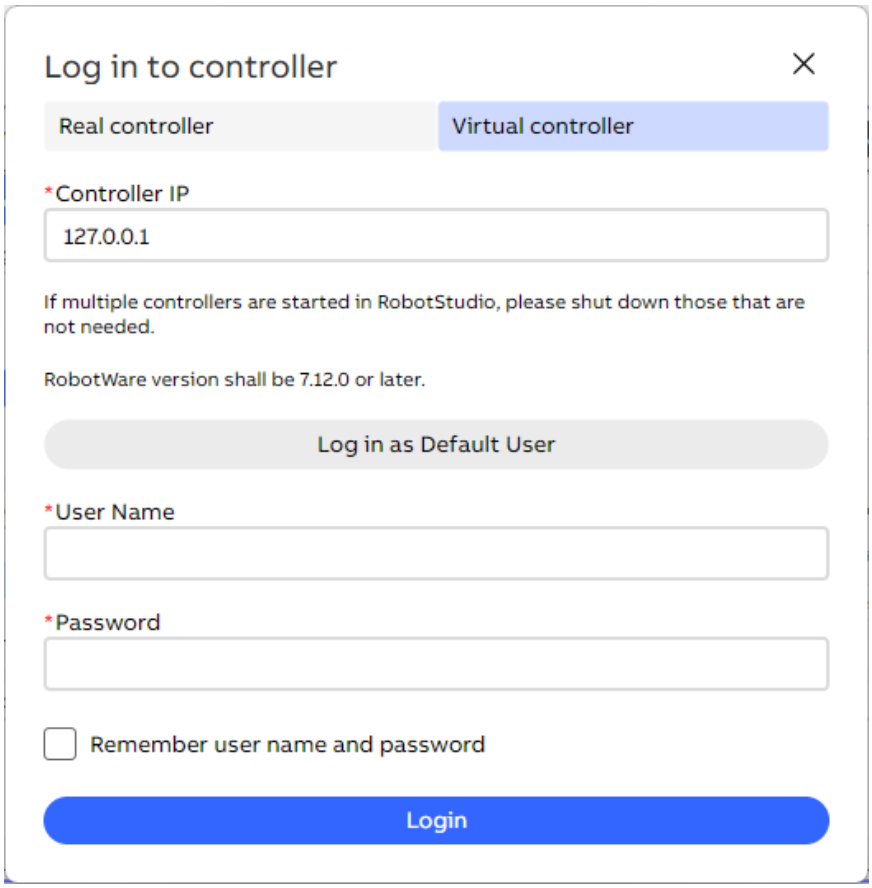
AppStudio projects list

4. Use the UI Designer, Function and Translation tabs to edit the WebApp content.

8.1.3 Steps to redeploy the WebApp to the controller

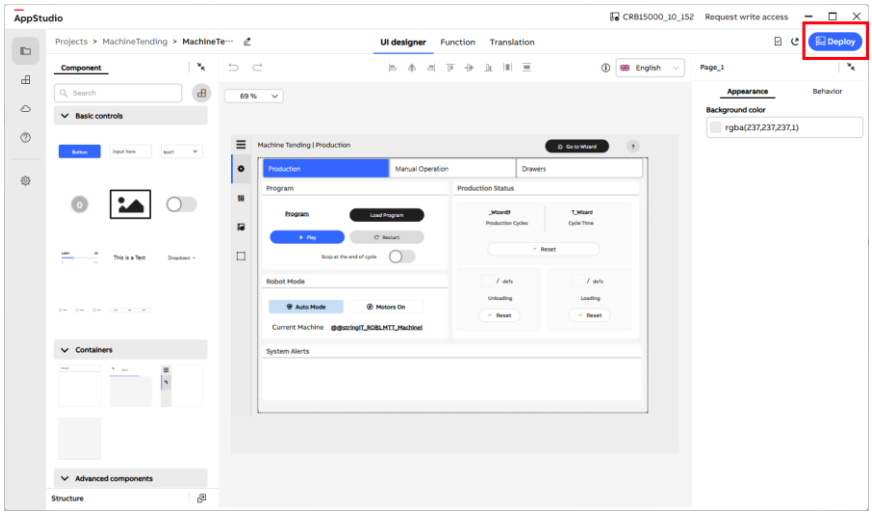
1. After validating your changes locally, connect to the target controller (click **Connect to controller** in the top-right). You will be prompted to log in.

AppStudio project shipped with the Add-In



AppStudio controller login dialog

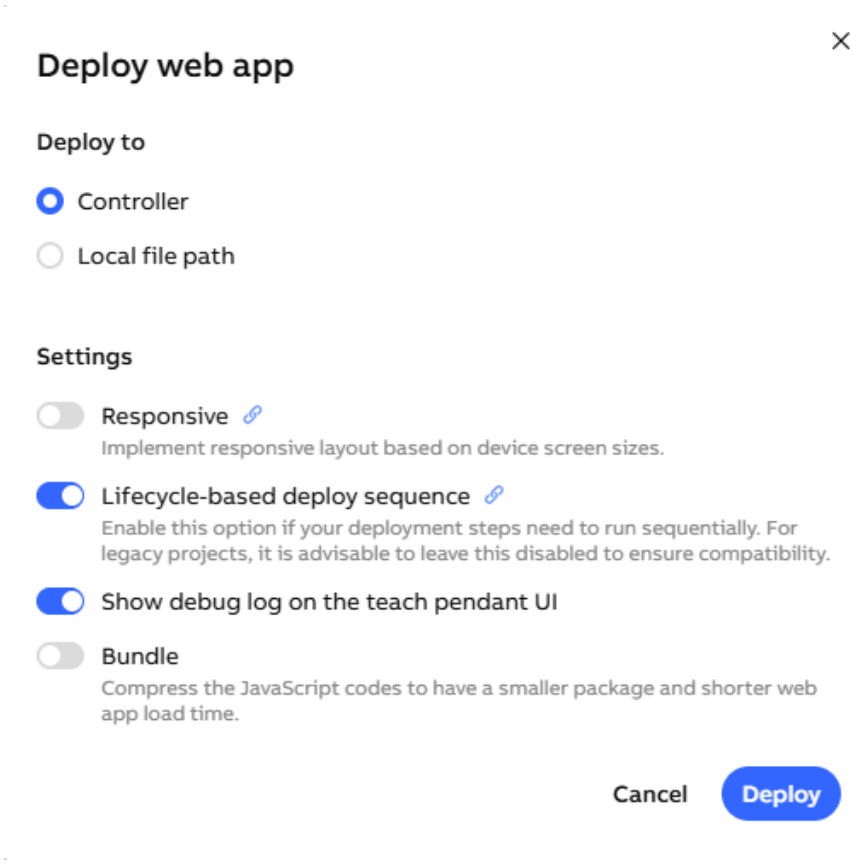
2. Click the **Deploy** button in the top-right to open the Deploy dialog.



AppStudio deploy button

3. In the Deploy dialog choose deployment target and options. Example settings:
4. `Deploy to`: Controller

5. `Responsive`: OFF
6. `Lifecycle-based deploy sequence`: ON
7. `Show debug log on the teach pendant UI`: ON
8. `Bundle`: optional



AppStudio deploy options

4. Click **Deploy** to push the built WebApp to the connected controller.

9 Index

A

ABB AppStudio — WebApp development platform. See 4.1. **Approach offset** — 6-DOF offset applied before/after pick/place. See 3.1.3.2, 5.1.8.

B

Blow-off — Air nozzle cleaning. See 3.3.8, 5.4.4, 6.4.

C

CallByVar — RAPID instruction for dynamic procedure dispatch. Used by gripper and spindle operations. See 5.1.5. **CAB_EXEC_HOOKS** — System configuration for procedures called on controller events. See 2.3.4. **Cycle time** — Measured between `MTT_StartCycle` and `MTT_EndCycle`. See 4.3.2.1, 5.1.10. **Cross-connection** — Recommended IO integration method. See 3.4.4.1.

D

Drawer — Sliding tray holder on the station. See 3.2. **Drawer state** — Open/closed, ready, enabled, priority. See 3.2.8, 5.5.5. **DSQC1031** — ABB Scalable IO module (16 DI + 16 DO). See 3.4.

E

ERR_MTT_ITEM_MISSED — Error raised on failed grip verification. See 3.3.7, 5.7.

F

Feed hold — Machine pause signal. See 3.1.4.3, 5.3.3. **First-stack skip** — Default behavior in `MTT_PickDrawer` (stack-first mode). See 5.1.2.

G

Gripper — End effector for part handling. See 3.3. **Grip direction** — Inward (close) vs outward (open). See 3.3.3.3, 5.6.1.

H

HAL (Hardware Abstraction Layer) — Module pattern isolating hardware-specific code. See 1.2. **Home position** — `MTT_jHome` joint target. See 5.2.1.

I

Increment mode — Layer-first vs stack-first tray iteration. See 3.2.5, 5.6.2. **Install script** — `install.cmd` for Add-In deployment. See 2.3.

L

Last cycle — `MTT_LastCycle` flag for graceful stop. See 4.3.1.4, 5.2.1. **Load data** — Robot payload definition. See 3.6, 5.1.6.

M

Machine HAL — `MTT_Machine.sysx` module. See 3.1, 5.3. **MachineX** — Generic real machine variant. See 3.1.7.2. **Magnet** — Electromagnetic attachment for drawers and doors. See 3.2.3, 3.1.2.1. **MotionSup** — Motion supervision override during door/drawer operations. See 3.1.2.1. **MTT_APPROACH_OFFSET** — Record type for approach/depart offsets. See 5.5.3. **MTT_DRAWER_STATE** — Record type for drawer runtime state. See 5.5.5. **MTT_GRIPPER** — Record type for gripper definition. See 5.5.2. **MTT_TRAY** — Record type for tray layout. See 5.5.1. **MTT_TRAY_INFO** — Record type for runtime tray state. See 5.5.6. **MTT_TRAY_POSITION** — Record type for tray position index. See 5.5.4. **MTT_WizardData.modx** — Configuration data module. See 2.3, 3B.

O

Omnivance — Station variant with 6 drawers and magnetic attachment. See 3.2.9.1.

P

Priority — Drawer selection priority (lower = higher). See 3.2.8.4. **Production counter** — `MTT_prodCounter`. See 4.3.2.2, 5.1.10. **ProFeederS** — Station variant with 5 drawers and mechanical grip. See 3.2.9.2.

S

Safety signals — DI1 (standstill), DI2 (low speed), DO9 (reset). See 3.5. **Spindle** — Machine chuck/clamping mechanism. See 3.1.3, 5.3.2. **Station HAL** — `MTT_User.sysx` module. See 3.2.1, 5.2.

T

Tool data — TCP calibration. See 3.3.4. **Tray** — Grid of part positions within a drawer. See 3.2.4.

W

WebApp — FlexPendant application for configuration and production. See 4. **Wizard blocks** — Visual programming blocks on the FlexPendant. See 3B, 4.4. **Wizard.modx** — User program module containing the production cycle. See 6.1. **Work object** — Coordinate system for drawer tray calculations. See 3.2.2.

Index

ABB Robotics Sweden AB
S-721 71 VÄSTERÅS, Sweden
Telephone +46 10-732 50 00

ABB Robotics Norway AS
Nordlysvegen 7, N-4340 BRYNE, Norway
Box 265, N-4349 BRYNE, Norway
Telephone: +47 22 87 2000

ABB Robotics (Shanghai) Limited
No. 39, 99 Miaoqiao Road,
Pudong New Area,
Shanghai, China
Telephone: +86 21 6105 6666

ABB Robotics United States LLC
1250 Brown Road Auburn Hills,
MI 48326 USA
Telephone: +1 248 391 9000

abb.com/robotics