



Siemens
Industry
Online
Support

APPLICATION EXAMPLE

Automated multi-side object visual inspection with Inspekto and Universal Robots UR5e.

SIEMENS Inspekto VIS / Simatic Robot Library / SRCI / PROFINET / TIA Portal / S7-1500V

Legal information

Use of application examples

Application examples illustrate the solution of automation tasks through an interaction of several components in the form of text, graphics and/or software modules. The application examples are a free service by Siemens AG and/or a subsidiary of Siemens AG ("Siemens"). They are non-binding and make no claim to completeness or functionality regarding configuration and equipment. The application examples merely offer help with typical tasks; they do not constitute customer-specific solutions. **The application examples are not subject to standard tests and quality inspections of a chargeable product and may contain functional and performance defects or other faults and security vulnerabilities. You are responsible for the proper and safe operation of the products in accordance with all applicable regulations, including checking and customizing the application example for your system, and ensuring that only trained personnel use it in a way that prevents property damage or injury to persons. You are solely responsible for any productive use.**

Siemens grants you the non-exclusive, non-sublicensable and non-transferable right to have the application examples used by technically trained personnel. Any change to the application examples is your responsibility. Sharing the application examples with third parties or copying the application examples or excerpts thereof is permitted only in combination with your own products. Any further use of the application examples is explicitly not permitted and further rights are not granted. You are not allowed to use application examples in any other way, including, without limitation, for any direct or indirect training or enhancements of AI models.

Disclaimer of liability

Siemens shall not assume any liability, for any legal reason whatsoever, including, without limitation, liability for the usability, availability, completeness and freedom from defects of the application examples as well as for related information, configuration and performance data and any damage caused thereby. This shall not apply in cases of mandatory liability, for example under the German Product Liability Act, or in cases of intent, gross negligence, or culpable loss of life, bodily injury or damage to health, non-compliance with a guarantee, fraudulent non-disclosure of a defect, or culpable breach of material contractual obligations. Claims for damages arising from a breach of material contractual obligations shall however be limited to the foreseeable damage typical of the type of agreement, unless liability arises from intent or gross negligence or is based on loss of life, bodily injury or damage to health. The foregoing provisions do not imply any change in the burden of proof to your detriment. You shall indemnify Siemens against existing or future claims of third parties in this connection except where Siemens is mandatorily liable.

By using the application examples you acknowledge that Siemens cannot be held liable for any damage beyond the liability provisions described.

Other information

Siemens reserves the right to make changes to the application examples at any time without notice and to terminate your use of the application examples at any time. In case of discrepancies between the suggestions in the application examples and other Siemens publications such as catalogs, the content of the other documentation shall have precedence.

The Siemens terms of use (<https://www.siemens.com/global/en/general/terms-of-use.html>) shall also apply.

Cybersecurity information

Siemens provides products and solutions with industrial cybersecurity functions that support the secure operation of plants, systems, machines and networks.

In order to protect plants, systems, machines and networks against cyber threats, it is necessary to implement – and continuously maintain – a holistic, state-of-the-art industrial cybersecurity concept. Siemens' products and solutions constitute one element of such a concept.

Customers are responsible for preventing unauthorized access to their plants, systems, machines and networks. Such systems, machines and components should only be connected to an enterprise network or the internet if and to the extent such a connection is necessary and only when appropriate security measures (e.g. firewalls and/or network segmentation) are in place.

For additional information on industrial cybersecurity measures that may be implemented, please visit www.siemens.com/cybersecurity-industry.

Siemens' products and solutions undergo continuous development to make them more secure. Siemens strongly recommends that product updates are applied as soon as they are available and that the latest product versions are used. Use of product versions that are no longer supported, and failure to apply the latest updates may increase customer's exposure to cyber threats.

To stay informed about product updates, subscribe to the Siemens Industrial Cybersecurity RSS Feed under <https://www.siemens.com/cert>.

Table of contents

1.	Introduction	5
1.1.	Overview	5
1.1.1.	Vision inspection systems	5
1.1.2.	SIEMENS Inspekto VIS	5
1.1.3.	Simatic Robot Library	6
1.1.4.	SIMATIC S7-1500V	6
1.1.5.	Purpose.....	7
1.1.6.	Scope.....	7
1.1.7.	What is in this manual?	8
1.1.8.	System Architecture.....	8
2.	Engineering	10
2.1.	Components used.....	10
2.2.	Hardware Setup.....	11
2.2.1.	Inspekto hardware connection	11
2.2.2.	Inspekto-UR5e Mounting Flange	12
2.2.3.	Complete setup connection	13
2.3.	Project planning	13
2.4.	Commissioning Configuration	14
2.4.1.	PLC configuration	14
2.4.2.	Robot configuration.....	14
2.4.3.	Inspekto configuration.....	14
2.4.3.1.	Enable PROFINET communication	14
2.4.4.	TIA Portal integration	15
2.4.4.1.	SIMATIC S7-1500V integration.....	15
2.4.4.2.	SIMATIC Robot Library integration	15
2.4.4.3.	Inspekto integration.....	17
2.5.	Robot path programming & Inspekto profile training.....	26
2.5.1.	Robot moving points	27
2.5.1.1.	"Home" position:.....	27
2.5.1.2.	"back0" position:	30
2.5.1.3.	"back1" position:	31
2.5.1.4.	"backInspection" position:	32
2.5.1.5.	"front0" position:	37
2.5.1.6.	"front1" position:	37
2.5.1.7.	"frontInspection" position:.....	38
2.5.2.	Robot's Paths.....	39
2.5.2.1.	backFull path	39
2.5.2.2.	frontFull Path.....	42
2.5.2.3.	multiSide Path	42

- 2.6. Controlling Inspekto via PLC 44
 - 2.6.1. Working sequence 44
 - 2.6.2. Operating modes..... 44
 - 2.6.3. Programming Inspekto function blocks..... 44
- 2.7. HMI programming 48
- 3. Additional information 49**
 - 3.1. Inspekto functions 49
 - 3.2. Inspekto system installation 49
- 4. Appendix 50**
 - 4.1. Service and support 50
 - 4.2. Application support 51
 - 4.3. Links and literature 51
 - 4.4. Documentation change 51

1. Introduction

1.1. Overview

1.1.1. Vision inspection systems

In today's automated production lines, vision inspection processes are indispensable for ensuring product quality and consistency. Vision inspection processes allow defects detection, verify assembly accuracy, and maintain high standards of quality control.

Historically, vision inspection was mainly performed manually by human operators, nevertheless it is still very frequently present in nowadays highly automated production lines. This method relies heavily on the skill and attention of workers to identify defects and ensure the product quality. Manual inspection is time-consuming, prone to human error, and limited by the inspector's ability to consistently detect subtle issues. As manufacturing processes evolve and the demands towards production are constantly growing, the need for more reliable and efficient inspection methods becomes apparent.

By automating the inspection process, manufacturers can significantly reduce human error, enhance production efficiency, and ensure that each product meets stringent quality requirements. The integration of vision inspection systems into automation lines not only improves product reliability but also adds substantial value by streamlining operations and reducing costs.

Early automated vision systems utilized basic image processing techniques to assist human inspectors. Over time, advancements in technology led to the development of sophisticated AI-based vision inspection systems, capable of performing complex inspections with high accuracy and speed. Autonomous AI-based vision inspection systems, such as SIEMENS Inspekto VIS, play a critical role in modern manufacturing environments. These modern systems have revolutionized quality control in manufacturing, providing real-time feedback and data analysis to quickly identify and correct issues. This proactive approach to quality control helps manufacturers maintain a competitive edge by delivering superior products to the market. The ability to automate inspection tasks also frees up human resources for more complex and value-added activities, further enhancing overall productivity and operational efficiency.

1.1.2. SIEMENS Inspekto VIS

Powered by the Autonomous Machine Vision Artificial Intelligence (AMV-AI) technology, the SIEMENS Inspekto VIS system, shortly, Inspekto, allows easy fit into dynamic and versatile production environments for a wide range of use cases, while maintaining high visual inspection quality. It is the only self-contained vision inspection product on the market and represents the most user-friendly solution for industrial quality inspection currently available. It can be set up and deployed by the final user in about 45 minutes and can be trained to inspect new products using an average of only 20 good samples of the item. No defective samples are needed for the training process.

The Inspekto system can be installed on several locations on the production line, in-line, end-of-line, offline or ad-hoc, and after a simple and short setup process, it captures images of the manufactured items using the connected camera.

The captured image of the inspected item is compared with the good references, using AI capabilities and if there are any defects, Inspekto will present them within an user-friendly easy to use user interface. The inspection searches for physical abnormalities, which exceed the shape tolerances and surface variations visible in the reference items. Inspekto will only present true defects, not those that can be explained by other phenomena, such as changes to lighting or movement. The AMV-AI technology is highly sensitive to details and identifies even minor differences between defects and permissible variations.

The Inspekto system can also communicate the results of the inspection to the PLC or upper-level control systems over different well-known industrial-grade communication protocols, such as Profinet or Ethernet/IP but also as simple TCP/IP telegrams. What's more, inspection images can be transferred forward to external systems for further processing via FTP.

1.1.3. Simatic Robot Library

The SIMATIC Robot Library concept transfers the command sets of a robot controller to the Totally Integrated Automation Portal (TIA Portal) and thus enables, for the first time, manufacturer-independent and uniform programming of industrial robots (incl. cobots) in the TIA Portal, executed on a SIMATIC S7-1500 controller. Additional programming on the robot controller is not required.

The SIMATIC Robot Library concept offers a solution for the growing need for uniform and consistent integration of industrial robots (incl. cobots) into the higher-level machine and plant control system. It does this by permitting robot programming for PLC specialists on a uniform automation platform. Small and medium-sized companies in particular benefit from this, as dependence on external robot specialists is reduced.

The basis for the SIMATIC Robot Library is the "Standard Robot Command Interface" specification, which is offered as a profile within the PROFIBUS user organization. This specification defines the basic communication mechanism between robot controller and PLC. It also standardizes definitions and robot commands from different manufacturers. Implementation takes place in each case by the respective robot manufacturer as "Interpreter" (server program on the robot controller), as well as by Siemens as SIMATIC Robot Library (client with command set library).

The robot controller is connected to the SIMATIC S7 controller via a GSDML file as PROFINET I/O device. Communication between the SIMATIC S7 and the respective robot controller is implemented via PROFINET (isochronous operation is not required), with the robot controller acting as the IO Device and the SIMATIC S7 as the IO Controller.

The SIMATIC S7 (client) calls server functions by sending command telegrams and the corresponding arguments to the robot controller (server) and its Interpreter. The robot controller executes the function and sends back command-specific response messages.

The SIMATIC Robot Library provides uniform robot control command sets for the user through its TIA Portal function blocks for:

- General commands
- Movement commands
- Read and write commands
- Trigger commands
- Calculation commands

1.1.4. SIMATIC S7-1500V

The S7-1500V virtual control software (CPU 1517V-1 PN), part of the S7-1500 PLC product family, is the innovated SIMATIC controller that uses Industrial Edge technology to be executable on any hardware platform with real-time behavior.

This makes it possible to set up an integrated automation system consisting of hardware-based and software-based components and this can be used holistically in many industrial sectors. For example, the S7-1500V converts any hardware, from a small industrial PC to a powerful IT server farm, into an Edge controller and opens up completely new possibilities for software / IT-defined automation that would not be possible with a pure hardware-based controller.

This product evolution is the first step into the future Industrial Operations X automation landscape, i.e. this virtual control runtime offers solutions that are possible in a software world, i.e. no delivery times of hardware and cost optimization through flexible pricing models depending on the service life.

IT integration for installation and updates, scalability through any Edge Device type and flexibility through multi-instantiation result in completely new possibilities under the further use of customer know-how and the already existing TIA Portal projects, as it is a compatible virtual controller of the SIMATIC S7-1500 product family that can be configured with TIA Portal.

1.1.5. Purpose

This application example focuses on the integration of the AI-based vision inspection system, Siemens Inspekto VIS or shortly, Inspekto, with an industrial collaborative robot (cobot). The integration is demonstrated using a Universal Robots UR5e cobot for implementing a simulated end-of-line multi-side object inspection station. Specifically, as an example of a possible real scenario, a SIMATIC S7-1500 controller is being inspected for defects that might have been caused during the manufacturing process, both on the front and the back side. Precisely, two types of inspection are performed over the PLC, which represents the main inspection mode the Inspekto system currently allows:

- **Defect inspection:** On the back side of the PLC the system checks whether there are any bent pins on the electrical connector.
- **Presence detection:** On the front side of the PLC the system inspects whether the PLC display is in place or not.

The goal of this application example is to demonstrate the feasibility and benefits of Integrating the Inspekto system into a robot arm to perform multi-angle visual inspections in an efficient and automated way. Additionally, it aims to serve as a reference implementation for users seeking to deploy the Inspekto system for similar inspection scenarios in their own production environments. Therefore, in addition to this manual, a reference TIA Portal project is available for downloading in the same entry of this document.

1.1.6. Scope

The application example is built around a SIMATIC S7-1500V virtual controller, the software-based counterpart of the classic SIMATIC S7-1500 hardware PLC. It showcases how to configure the UR5e robot controller and how to train the Inspekto system to be able to detect possible defects or missing components on different sides of a SIMATIC S7-1500 PLC.

In addition, it highlights how to control both the UR5e cobot and the Inspekto system using TIA Portal, leveraging the Simatic Robot Library for robot path programming and integration as well as the ready-to-use TIA Portal function blocks for the control of the Inspekto system using the PROFINET protocol. This setup enables users to configure and program robot arms from various vendors through a unified engineering interface, eliminating the need for multiple software tools. The Inspekto TIA Portal function blocks allows seamless integration and control of the system within the automation process.

The application example also includes an HMI interface designed for the KTP900F Mobile Panel, as an implementation example that allows operators to monitor the inspection results as well as switch between different operating modes of the system. For demo purposes, within the application example 2 operating modes are presented:

- **Auto:** In this mode the system performs a complete front-back inspection of the S7-1500 PLC. Additionally, in this mode the user can enable the "Continuous" mode in which case the system will perform consecutive front-back inspections until the user stops the loop using the "Stop" button.
- **Manual:** In this mode, the user can select which side, back or front, of the S7-1500 PLC must be inspected and the system will perform an one-time inspection. The operation can be repeated by pressing any of the 2 dedicated buttons again.

NOTE

The use case presented in this application example is intended solely for demonstration purposes and does not reflect a real production scenario.

The accompanying reference project is offered as a guideline for implementing similar use cases. It should not be used as-is in production environments.

Users are responsible for adapting and modifying all components of the example to meet the specific requirements of their individual use cases and inspection scenarios.

The S7-1500 PLC that is being inspected in this use case is located in a testing environment and it is manually placed and secured in the inspection point. As a result, the inspection cycle is initiated manually by the user via the HMI interface and the inspection trigger logic is programmed accordingly. Additionally, the movement paths of the UR5e cobot are programmed based on

- the sides of the object that is inspected, and
- the environment where it is located.

NOTE

In production-ready scenarios, users must adapt both the inspection trigger logic in the TIA Portal project and the robot's movement paths to suit the specific application requirements.

1.1.7. What is in this manual?

This manual walks you through the different steps of the configuration process, including:

- **Robot controller configuration:** Setting up the UR5e cobot and connecting it to the SIMATIC S7-1500V virtual controller.
- **SIMATIC Robot Library integration:** Importing and configuring the library within TIA Portal to enable robot path programming.
- **Inspekto system integration:** Connecting and configuring the Inspekto system in TIA Portal, including communication setup and TIA Portal function block usage.
- **Robot path programming:** Creating movement paths and inspection points using the Simatic Robot Library to ensure that the sides of interest for inspection are accurately captured by the Inspekto camera.
- **Inspekto system configuration:** Creating and training the inspection profiles to detect defects or missing components on the inspected object as well as enabling the PROFINET connectivity interface and testing communication with the PLC.
- **Inspection logic:** Usage of dedicated TIA Portal function blocks to load inspection profiles and trigger inspections.
- **HMI Design:** Implementing an HMI interface on the KTP900F Mobile Panel to visualize inspection results and manage operating modes.

1.1.8. System Architecture

This chapter introduces the overall system architecture of the application example, offering a high-level view of the key components used and how they interact.

- **Inspekto System:** The autonomous machine vision system mounted on the robot arm. It performs visual inspection by capturing images after the robot has reached and stabilized at each inspection point. It includes a camera and processing unit, and it is integrated within the whole system via PROFINET.
- **Universal Robot UR5e Collaborative Robot Arm:** A lightweight cobot used to move the Inspekto camera to predefined inspection positions around a stationary PLC S7-1500 which is the object to be inspected. It stops at inspection point to allow the camera to capture image and inspect it.
- **SIMATIC S7-1500V Virtual PLC:** The software-based PLC that replicates the functionality of the SIMATIC S7-1500 hardware controller. It manages the control logic for both the robot and the inspection system. Since Virtual PLC runs as an Industrial Edge app, a SIMATIC Industrial Edge IPC BX-39A is used as a runtime environment.
- **TIA Portal Engineering Workstation:** The central engineering environment used to configure the virtual PLC, program the robot paths, and integrate the Inspekto system. It also hosts the Simatic Robot Library.
- **SIMATIC Robot Library:** A software library within TIA Portal that provides a unified interface for path programming and controlling robots from different vendors, simplifying engineering and reducing tool complexity.
- **KTP900F Mobile Panel:** A mobile operator panel used for robot path programming directly from the interface offered by the SIMATIC Robot Library, visualize inspection results and switch between different operating modes of the system.
- **PROFINET network:** The communication backbone that connects all components, ensuring seamless data exchange and synchronization between the PLC, UR5e robot, Inspekto, and the HMI panel.

The following visual diagram that maps out the hardware and software elements involved aims to provide you with a quick and clear understanding of how the elements work together to enable the automated multi-side inspection of the S7-1500 PLC.

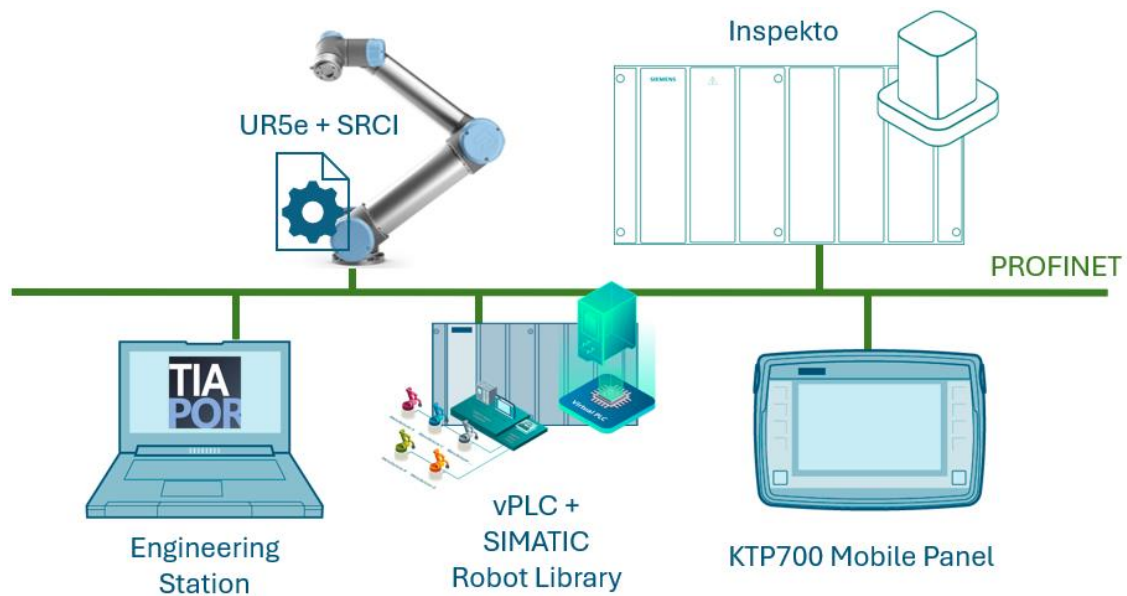


Figure 1-1: Application example components and network architecture

2. Engineering

2.1. Components used

This application example has been created with the following hard- and software components:

Component	Qty	Article Number	Note
SIEMENS Inspekto VIS Basic V6	1	6ET2901-0AA00-1BA0	The software SIEMENS Inspekto VIS Professional V6 can also be used for this application example.
STEP 7 Professional V20 (TIA Portal)	1	6ES7822-1AE24-0YA5	The screenshots of this application example were taken from TIA Portal V20. However, the reference project can be open with any higher version of STEP7 Professional (TIA Portal) can be used. The application example can be also implemented with lower TIA Portal versions if a hardware-based PLC is used.
WinCC Unified PC (max.) ES V20 (TIA Portal)		6AV2153-0XB02-4LA5	The screenshots of this application example were taken from TIA Portal V20. However, for this application example any higher version of WinCC Unified ES (TIA Portal) or any WinCC package that allows the programming of Mobile Panels higher than V20 can be used.
CPU S7-1517V	1	-	vPLC is available via the Industrial Edge Marketplace . For this application a hardware-based S7-1500 PLC can also be used. Since the vPLC runs as an Industrial Edge app, additional requirements , such as IEHub access, an IEM and an IED are needed to run the vPLC.
Mobile Panel KTP900F	1	6AV2125-2JB23-0AX0	-
SIEMENS Inspekto VIS GSDML	1	-	The SIEMENS Inspekto GSDML represents the .XML file required to add the SIEMENS Inspekto VIS system in the hardware of TIA Portal. It is provided within the reference project available in the Download section of this SIOS entry of this document.
Universal Robot UR5e	1	-	-
SIMATIC Robot Library	1	6ES7825-0AE00-3AA0	For this application example the version v1.2 of SIMATIC Robot library was used.
SCALANCE XC108	1	6GK5108-0BA00-2AC2	-

Table 2-1: Components used for the creation of this application example

2.2. Hardware Setup

2.2.1. Inspekto hardware connection

To start using the Inspekto system, all its hardware components must be connected as shown below

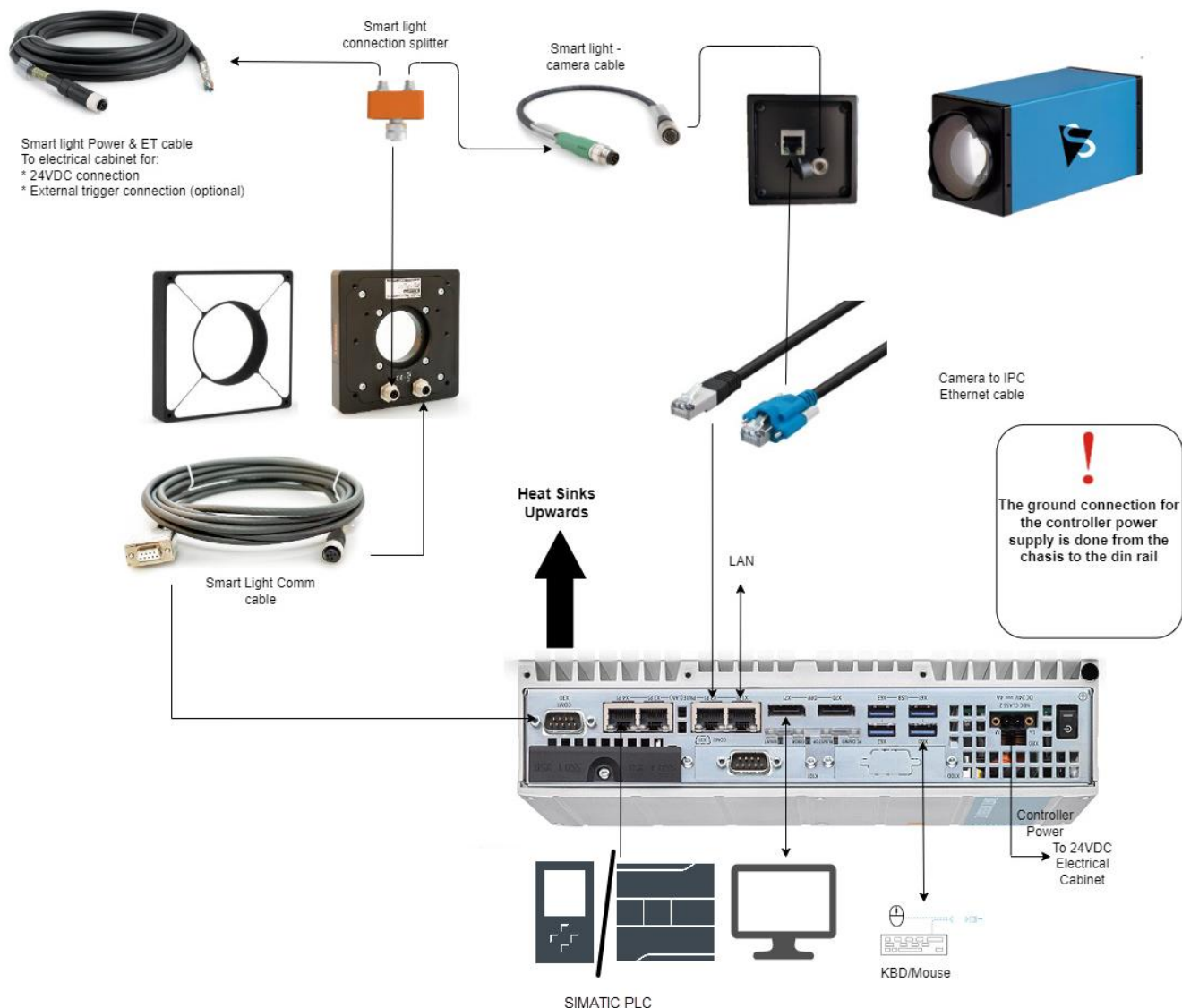


Figure 2-1: Hardware connection diagram for Inspekto

NOTE

To allow PROFINET communication, the controller must be connected to the X4 Ethernet port of the Inspekto IPC.

2.2.2. Inspekto-UR5e Mounting Flange

The Inspekto camera is mounted to the tool flange of the UR5e cobot using a custom 3D printed 2-piece mounting bracket designed following the specifications of the UR5e tool flange available in the following [link](#).

NOTE

The mounting flange shown in this documentation was designed specifically for the described use case.
For other applications, the design should be adapted to ensure that the robot arm can move freely and reach all required inspection points.

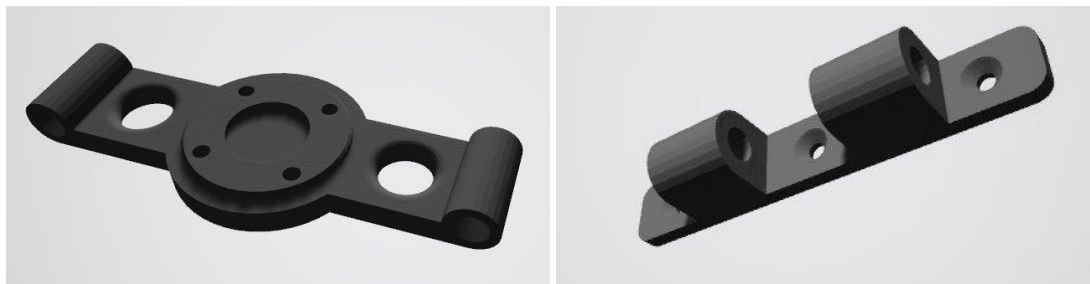


Figure 2-2: Mounting flange for Inspekto camera on UR5e's tool flange

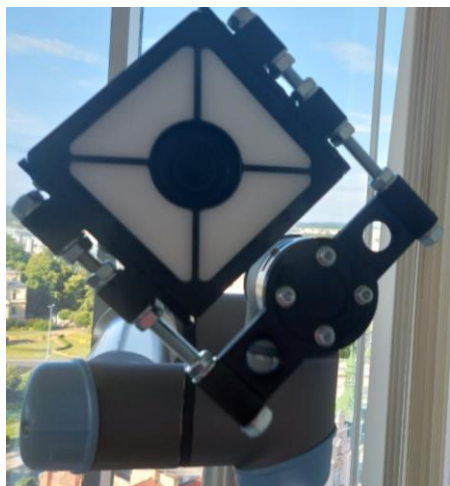


Figure 2-3: Inspekto camera mounted on the UR5e cobot

2.2.3. Complete setup connection

The connection between Inspekto system and the other hardware components used, was done following the next diagram.

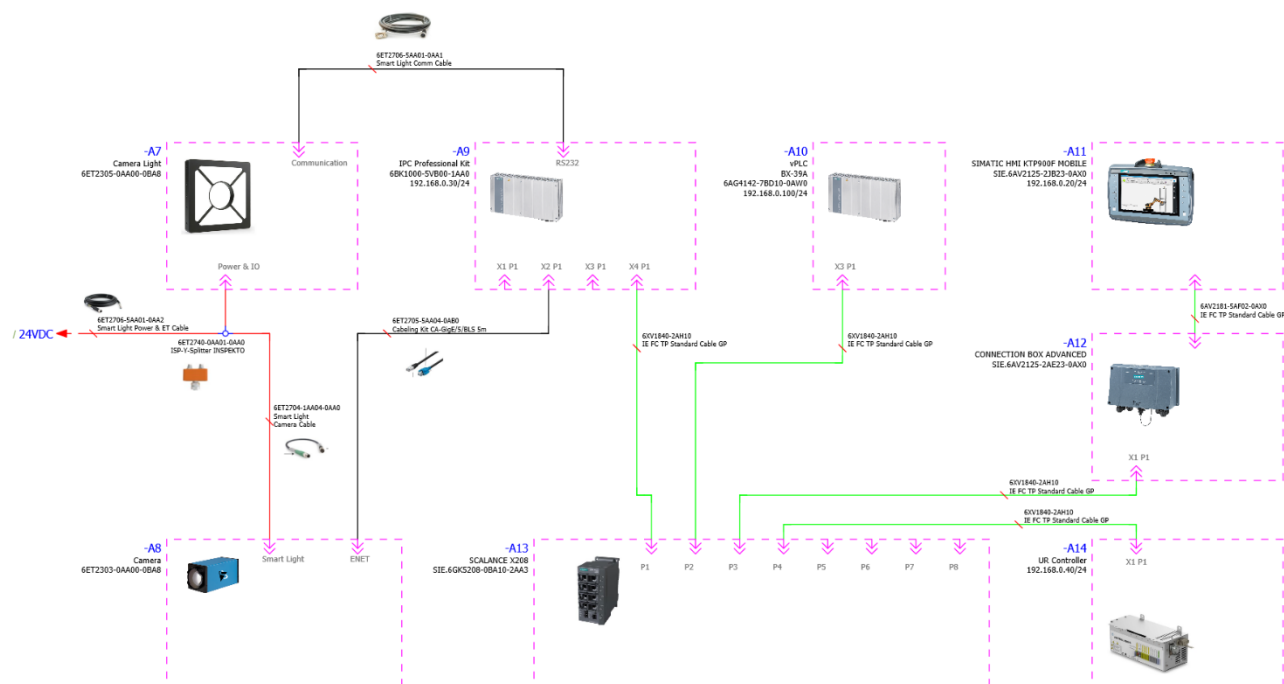


Figure 2-4: Complete system hardware connection diagram

2.3. Project planning

To successfully establish a PROFINET communication between the Inspekto system, the UR5e cobot and the controller, there are different steps that must be performed. By doing this, you can integrate Inspekto and the robot arm within the automation environment and implement more complex quality inspection processes such as object multi-side inspection.

For this application example the following steps were followed:

- **System architecture design** – planning the overall layout of the hardware and software components including the communication interfaces.
- **Hardware setup** – physical installation of the UR5e robot, the Inspekto system, the HMI panel, the virtual controller and the Scalance switch.
- **PLC configuration** – deployment and configuration of the SIMATIC S7-1500V virtual controller within the IED.
- **Robot configuration** – initial setup of the UR5e's SRCI configuration to allow the communication with the PLC.
- **Inspekto configuration**– activating PROFINET communication for enabling the system control over the PLC.
- **TIA Portal configuration** – integration of the SIMATIC Robot Library and the GSDML of the Inspekto system together with the TIA Portal function blocks to enable the robot path programming and the control of Inspekto system.
- **Robot path programming and Inspekto profile training** – Definition and programming of the robot movement paths and the inspection points for the SIMATIC Robot Library with the support of the KTP900F Mobile Panel and creating the profiles for defect detection.
- **Inspekto function blocks programming**– implementation of control logic using the dedicated TIA Portal function blocks to control the inspection process, including the inspection trigger and switching between inspection profiles.
- **HMI development** – implementation of the operator interface on the KTP900F Mobile Panel to visualize the inspection results and switch the operating modes.
- **Testing and validation** – Execution of functional test to verify robot movement and inspection accuracy.

2.4. Commissioning Configuration

2.4.1. PLC configuration

In this setup, we're using a vPLC (CPU 1517V) running on an IED **BX-39A**. The IED is onboarded to an **IEM** instance.

In the **IEM**, the **"CPU 1517V"** app is already available in our catalog (previously pushed from **IE Hub**).

After installing the **"CPU 1517V"** app on the BX-39A, complete the following steps:

- Configure **NIC isolation** on the IED.
- Set the vPLC IP address.

For detailed instructions on those steps, follow the official vPLC example on GitHub from here:

<https://github.com/industrial-edge/S7-1500V-getting-started/blob/main/docs/Installation.md#configure-1517v-nic-isolation-on-iedv>

2.4.2. Robot configuration

To establish communication between the UR5e robot controller and the SIMATIC PLC using the SIMATIC Robot Library, it is necessary to install the SRCI interpreter software (URCap) on the UR5e controller. This software enables the robot to be fully integrated and controlled from the PLC environment.

With SRCI, robot motion commands and programming interfaces are handled directly within the PLC, alongside standard automation functions such as automation process logic and safety control. This unified approach simplifies integration by allowing engineers to manage diverse robotic systems through a consistent and centralized platform.

SRCI operates as a command-based protocol over PROFINET and is maintained as an open standard by PROFINET International. The UR5e implementation supports the CORE PROFILE defined in version 1.3 of the SRCI specification, covering essential commands for both basic and advanced robotic operations.

NOTE

The SRCI is not a standard functionality of the UR robots. To be able to use the URCap SRCI interpreter you need to activate it using the corresponding license. Please contact your local UR sales representative to order the activation license.

Once you have a valid activation license, please follow the instructions below to install the URCap SRCI interpreter on your UR5e robot controller.

<https://docs.universal-robots.com/tutorials/communication-protocol-tutorials/srci-tutorials.html#robot-setup>

2.4.3. Inspekto configuration

2.4.3.1. Enable PROFINET communication

To be able to communicate the Inspekto system with a SIMATIC PLC, the dedicated network card interface for connectivity of the SIMATIC IPC BX-39A where the Inspekto software is running must be enabled. Since communication is done via the PROFINET protocol, select the *Profinet* connectivity option.

To do this, navigate to your Inspekto user interface, hover over the left side of the screen to open the navigation menu and then click on "Settings".

Under the General tab, scroll down to the Connectivity section and choose *"Profinet"* option from the drop-down menu.

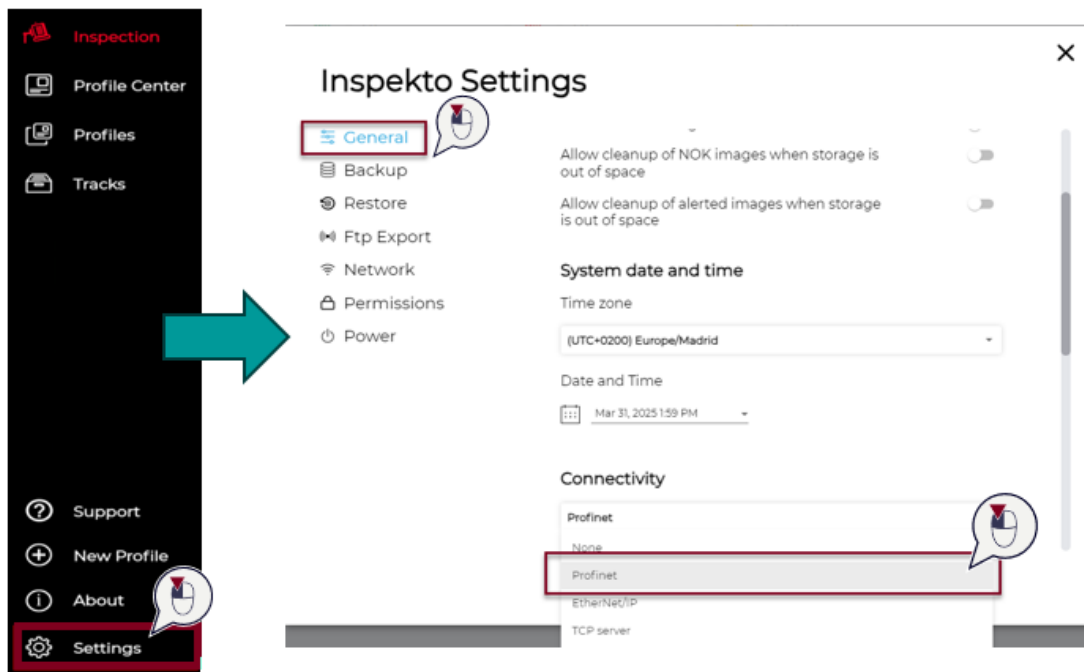


Figure 2-5: Enable Profinet communication within Inspekto.

At this point, no IP needs to be defined for the Connectivity interface. Once connected to the PLC, this will assign the IP that was configured in the TIA Portal project to the Inspekto system automatically.

2.4.4. TIA Portal integration

Once the initial individual configuration of each component of the system's architecture is done, it's time to connect all the components together within TIA Portal to establish the PROFINET communication and allow the virtual controller control both UR5e robot and Inspekto.

The integration steps described in this section were done using TIA Portal V20. Nevertheless, the same steps apply for any higher TIA Portal version.

The application example was set up with the following IP configuration:

- S7-1500V: 192.168.0.100/24
- Inspekto: 192.168.0.30/24
- UR5e robot: 192.168.0.40/24
- HMI KTP900F: 192.168.0.20/24

If you need a different configuration, you will have to adjust the respective addresses in TIA Portal. The addresses of the above-mentioned components can be defined in the respective device configuration in TIA Portal; they are assigned to the devices when the hardware configuration is downloaded.

You should also adjust the IP address of your engineering PC in order to reach the system components over the network.

2.4.4.1. SIMATIC S7-1500V integration

To integrate the 1517V vPLC in TIA Portal:

- Add the CPU 1517V (vPLC) as a device in your TIA Portal project.
- In the device's Ethernet properties, set the IP address to exactly match the IP you configured for the vPLC on the IED.

For step-by-step guidance, follow the TIA Portal section of the official getting-started guide:

<https://github.com/industrial-edge/S7-1500V-getting-started/blob/main/docs/Installation.md#configure-tia-portal-project>

2.4.4.2. SIMATIC Robot Library integration

You have already configured the UR controller with SRCL as described in [2.4.2 Robot configuration](#) chapter.

The next step is to obtain the **SIMATIC Robot Library (SRL)** and the **SIMATIC Robot Integrator (SRI)**. Both are provided within a TIA Portal reference project that you can copy into your own project to streamline SRL setup and programming.

Download and licensing details, including how to purchase an SRL license and how to obtain a 2-hour trial license, are available here:

<https://support.industry.siemens.com/cs/ww/en/view/109989677>

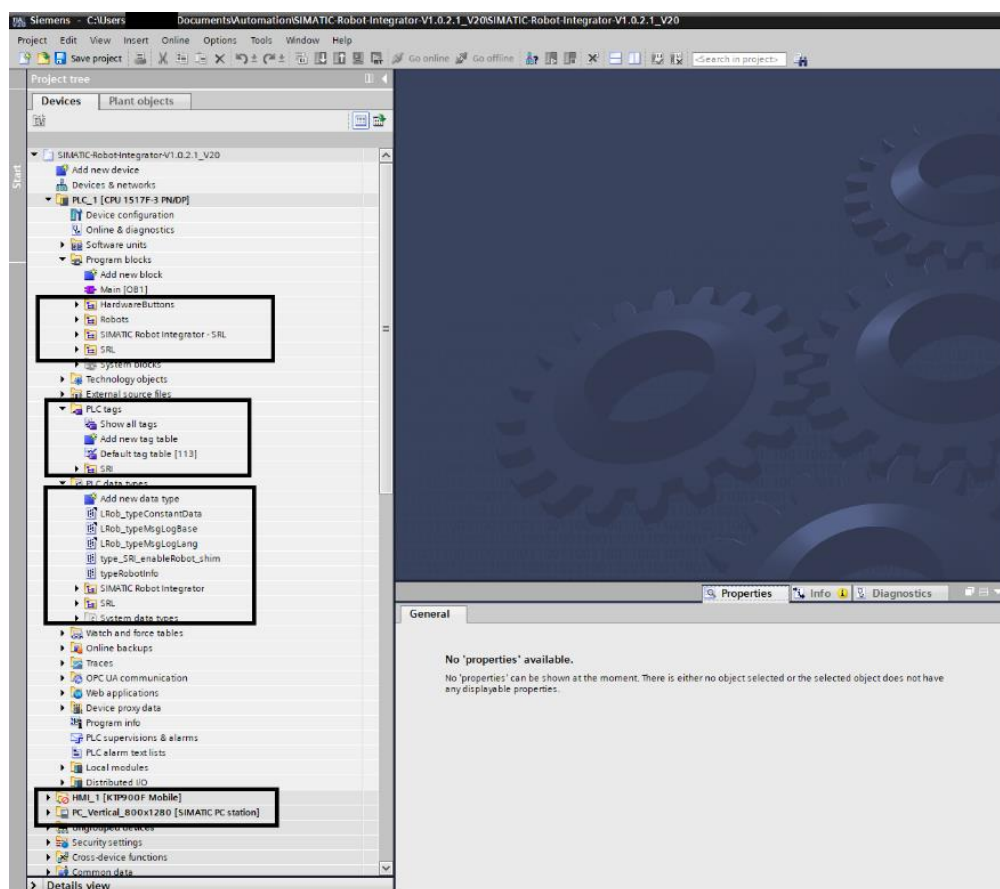


Figure 2-6: SRL/SRI TIA Portal reference project

Once you have the SRL/SRI reference project, you can either copy the required blocks and files into your own TIA Portal project or work directly from the reference project and adapt it to your PLC hardware. The reference project contains everything needed to work with the supported robots (UR, Stäubli, JAKA, etc.); in our case, adapt it for the UR controller.

To do so, follow the steps outlined here:

<https://docs.universal-robots.com/tutorials/communication-protocol-tutorials/srcl-tutorials.html#tia-portal-robot-setup>

After completing this setup, you should be able to go online with your PLC and see the UR device in green status (no errors), as shown below:

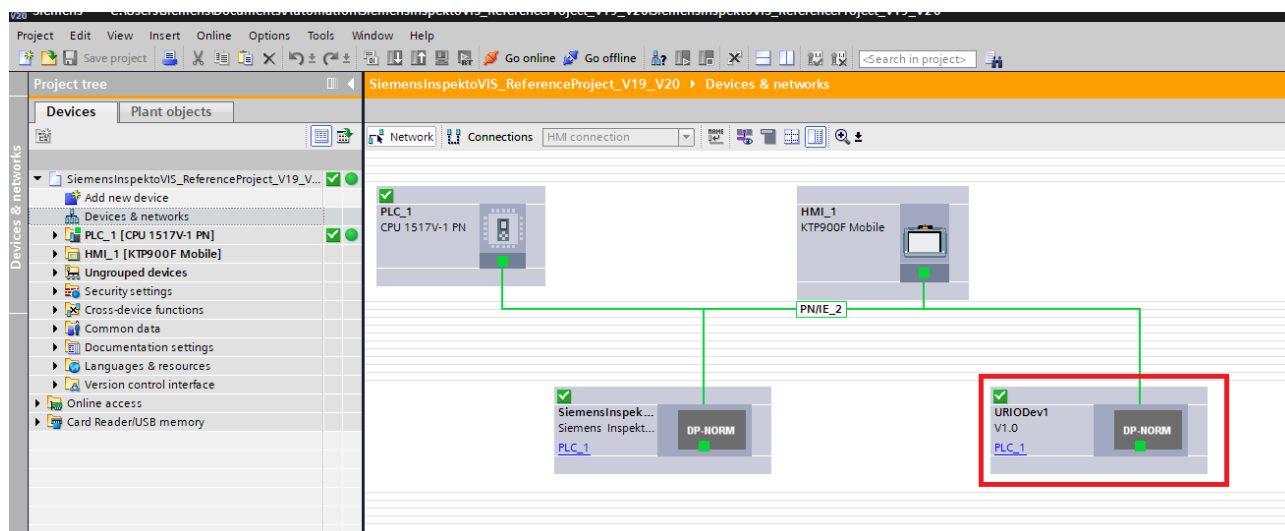


Figure 2-7: UR controller online status

Then, you can operate the robot from the PLC via the **MachineInterfaceSRL DB** (power the robot on or off, reset errors, start or interrupt path execution, and read the current robot status)

Name	Data type	Start value	Retain	Accessible f...	Write...	Visible in ...	Setpoint	Supervision	Comment
robot[5]	"typeUDTMachine"								
command	"typeUDTMachine..."								
automatikMode	Bool	false							Switch to the machine automatikMode, to be...
powerOn	Bool	false							Enable robot
powerOff	Bool	false							Disable robot
reset	Bool	false							Error acknowledgement
abortPath	Bool	false							Abort the robot movement
moveToHome	Bool	false							Move to home position
movePath	Bool	false							Online Program
selectedPath	Int	1							Selected Path for move path
interruptPath	Bool	false							Interrupt a running path
continuePath	Bool	false							Continue an interrupted path
continueOnWaitpoint	Bool	false							Continue a path waiting on a waitpoint
gripperOpened	Bool	false							Continue on the interrupted path after the gri...
gripperClosed	Bool	false							Continue on the interrupted path after the gri...
status	"typeUDTMachine..."								
automatikMode	Bool	false							Operation mode
manuelMode	Bool	false							Operation mode
poweredOn	Bool	false							Robot enabled
busy	Bool	false							A Robot program is selected and started
done	Bool	false							Selected robot program is done
active	Bool	false							Robot is moving on the path
aborted	Bool	false							Selected path is aborted
error	Bool	false							Robot program or Robot has an error. Movem...
errorID	DInt	0							Error ID for the error
robotError	Bool	false							Robot reports an error
gripperOpen	Bool	false							Output to the gripper / tool: open Gripper
gripperClose	Bool	false							Output to the gripper / tool: close Gripper
waitPointActive	Bool	false							TCP is stopped on a defined wait point and is ...
mFunction	Int	0							User defined indicator to get out where TCP is
robot[6]	"typeUDTMachine"								
robot[7]	"typeUDTMachine"								

Figure 2-8: MachineInterfaceSRL data block

NOTE

For a deeper dive into using **MachineInterfaceSRL** and the **SRL/SRI** overall, please refer to the documentation bundled with the SRL/SRI package.

The next step is to create/teach the robot's points and movement paths.

2.4.4.3. Inspekto integration

GSDML Installation

At the time of the publication of this document, the integration of the Inspekto system within TIA Portal is done using its corresponding GSDML file.

A GSDML file (General Station Description Markup Language) is an XML-based configuration file used to describe the properties and capabilities of a PROFINET device. It contains detailed information about the device's hardware, supported communication protocols, parameters, diagnostics, and I/O modules. This standardized format enables engineering tools like TIA Portal to recognize and integrate third-party devices seamlessly into automation projects.

NOTE

The GSDML file of Inspekto will be automatically installed when opening the reference project provided in the "Downloads" section of this SiePortal entry.

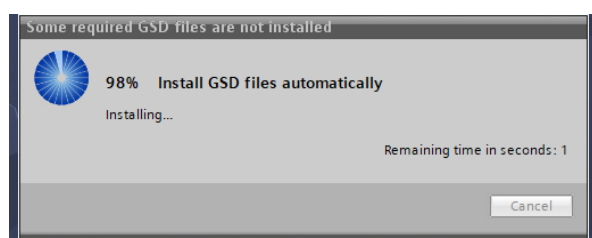


Figure 2-9: GSD installation progress

Once the GSD file of Inspekto is installed it will be available with the in the TIA Portal catalog under the section "Other field devices – Profinet IO – Sensors – Inspekto – Inspekto Cameras – Siemens Inspekto VIS".

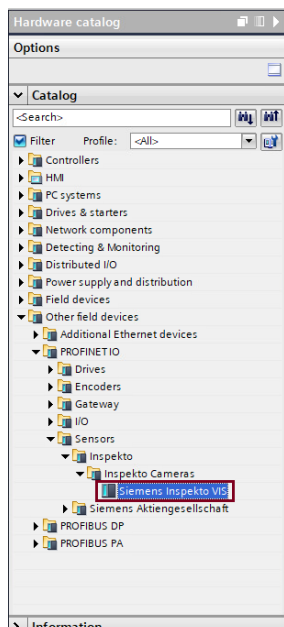


Figure 2-10: Inspekto camera within TIA Portal catalog after installing GSD file

To add the Inspekto device to your automation project, you just need to drag and drop it from the catalog to the "Network view".

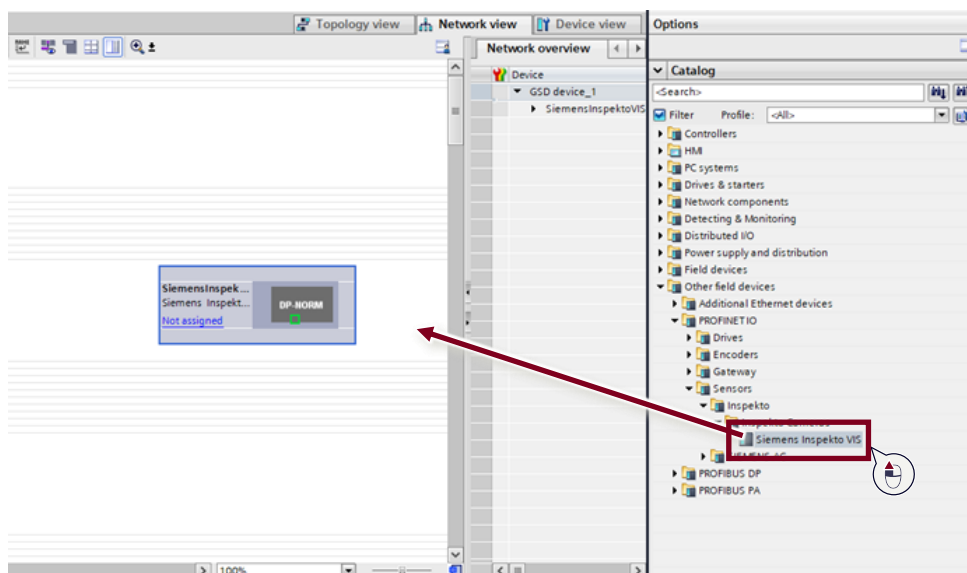


Figure 2-11: Adding Inspekto to the hardware configuration of TIA Portal project

Device configuration

Once the Inspekto device is added into the TIA Portal project, it must be connected to the PROFINET network and assigned to a PLC to ensure it can be controlled using the corresponding TIA Portal function blocks. The following steps should be followed to achieve this:

1. In the Siemens Inspekto VIS device, right click on "Not assigned" and then click on "Assign to new IO controller"

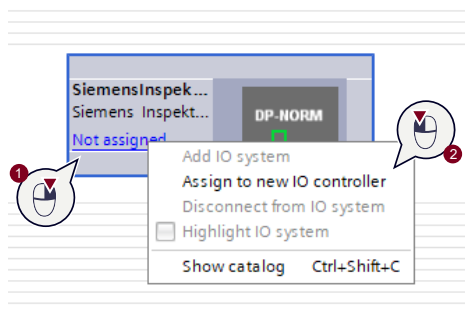


Figure 2-12: Assigning Inspekto to vPLC controller

2. Select the PLC you want to assign Inspekto to and press "OK". The PLC and Inspekto are now connected via a PROFINET network.

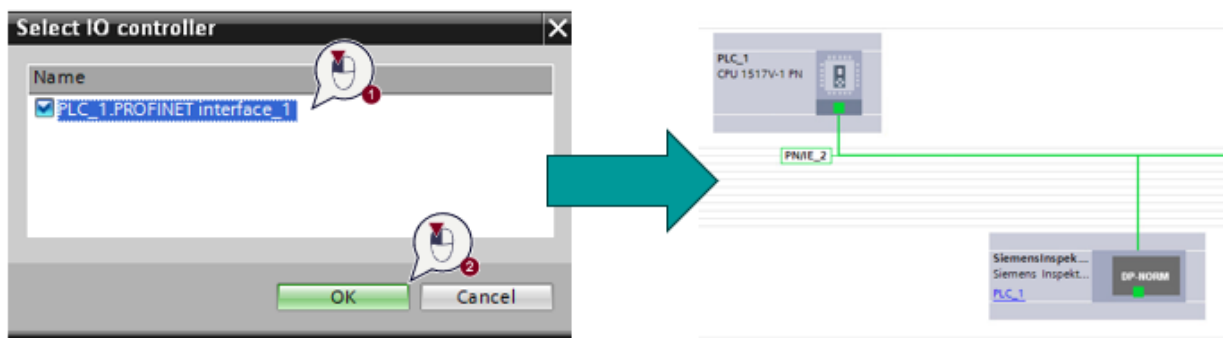


Figure 2-13: Inspekto system assigned to vPLC controller

NOTE

By default, TIA Portal assigns a random IP address to the devices on the default PROFINET subnet 192.168.0.1. Adjust the IP address of your PLC and Inspekto system according to your network configuration.

3. Double click on the "SiemensInspektoVIS" to open the "Device overview" tab and check that all relevant modules of the Inspekto device are present and follow the same structure as below.

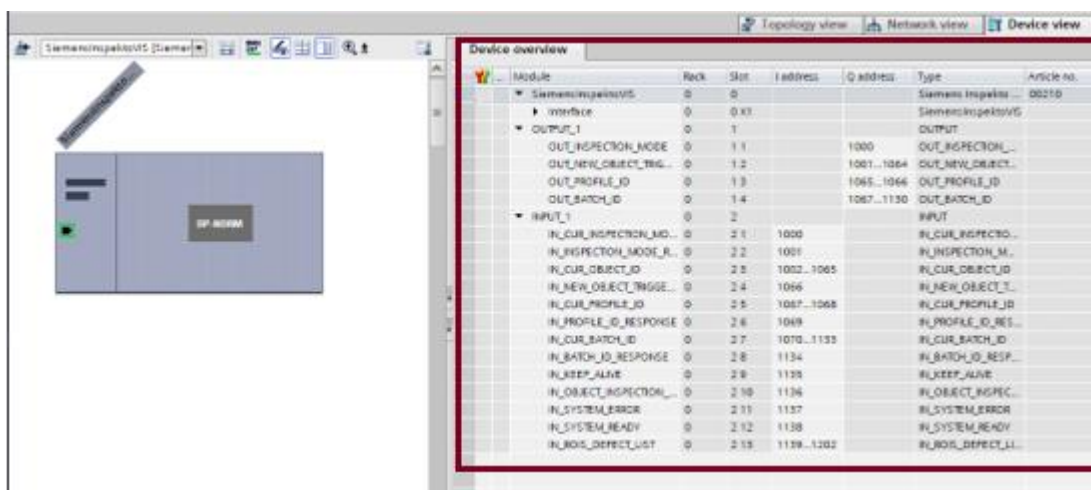


Figure 2-14: Inspekto device overview

NOTE

If any of the "Output" and "Input" modules do not appear in the list. Drag and drop the "Output" and "Input" modules from the catalog to the "Device overview" table

4. Assign the PROFINET device name to the Inspekto system.
 - a. Open the "Device & networks" view.
 - b. Right click on the Inspekto device and click on "Assign device name".

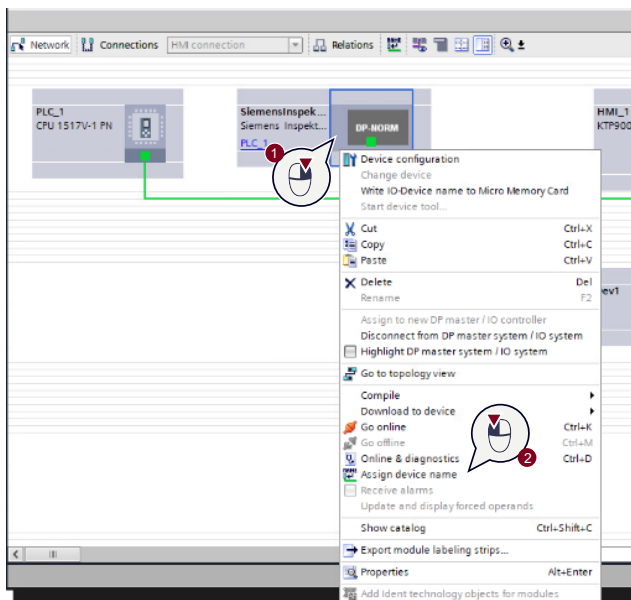


Figure 2-15: Assign PROFINET name to Inspekto system

- c. Select the Inspekto device within the list of the accessible device in the network and click on "Assign name".

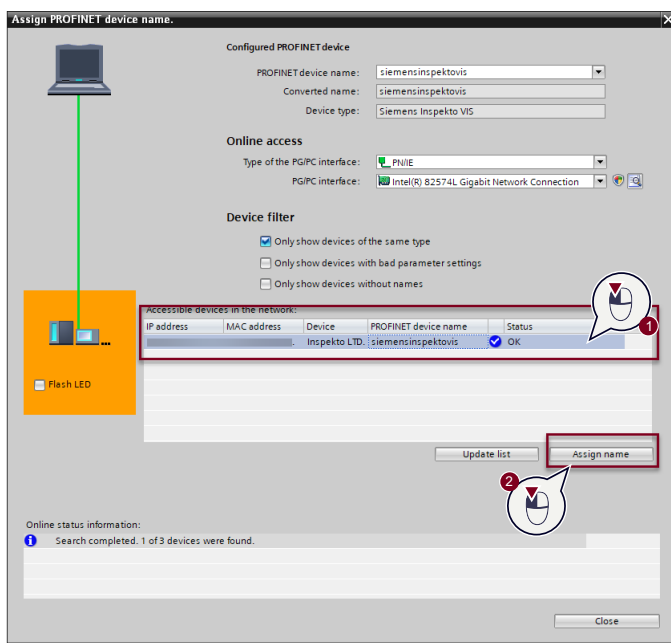


Figure 2-16: Select device for assigning PROFINET name

- d. If the PROFINET name is assigned successfully you should see the following information on the screen.

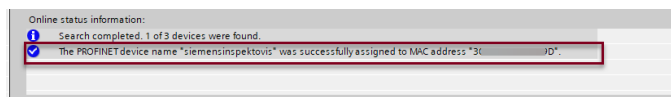


Figure 2-17: PROFINET name assigned successfully

5. Compile and download the project to the PLC.

PLC data type configuration

Some of the Inputs and Outputs of Inspekto, such as OUT_NEW_OBJECT_TRIGGER or IN_CUR_OBJECT_ID are 64 bytes long. A new custom PLC data type needs to be created manually that later can be mapped to the corresponding I/Os. In this case, a custom PLC data type composed of an array of 64 chars will be created.

1. From "Project tree" go to PLC data types, double click on "Add new data type" and rename the new PLC data type to "Array64Chars" and click OK.
2. In the new window, in row 1 under name, type "data" and under "Data type" set it to be Array [0..63] of Char".

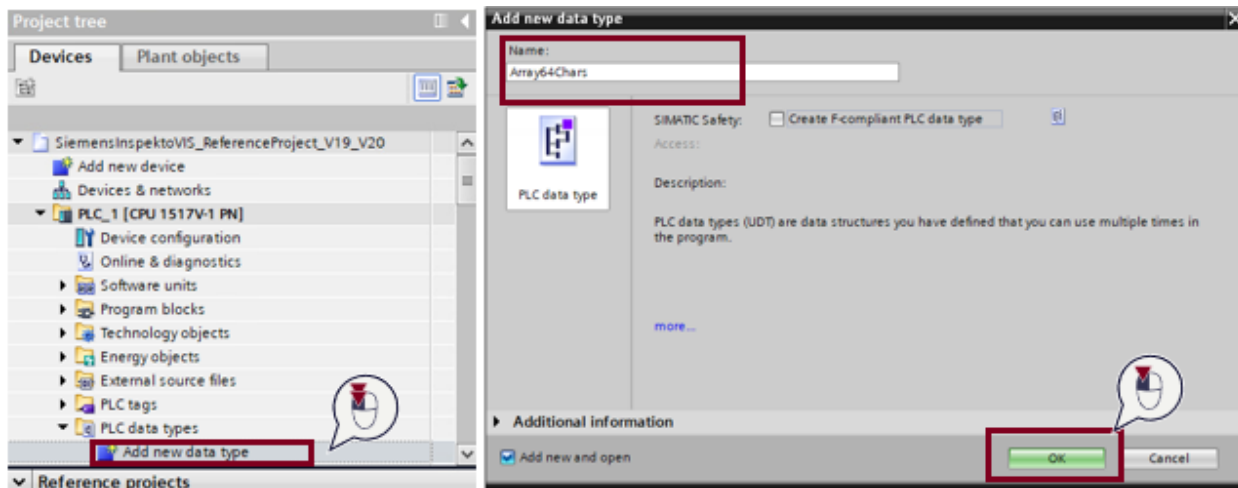


Figure 2-18: Adding new data type for Inspekto tags

Array64Chars									
	Name	Data type	Default value	Accessible f...	Writa...	Visible in ...	Setpoint	Supervision	Comment
1	data	Array[0..63] of Char		☒	☒	☒	☐		

Figure 2-19: Adding data type Array64Chars

Creating Inspekto PLC tags

1. In TIA Portal go to "PLC tags" and double click on "Add new tag table"

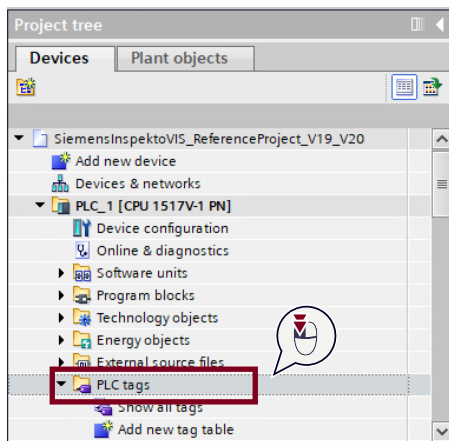


Figure 2-20: PLC Tags

2. Name the tag table "inspekto_tags" and add the following tags with the corresponding I/O addresses based on the previous module configuration.

inspekto_tags									
	Name	Data type	Address	Retain	Acces...	Writa...	Visibl...		
1	IN_BATCH_ID_RESPONSE	Byte	%B1034	☐	☒	☒	☒		
2	IN_CUR_BATCH_ID	"Array64Chars"	%I1070.0	☐	☒	☒	☒		
3	IN_CUR_INSPECTION_MODE	Byte	%B1000	☐	☒	☒	☒		
4	IN_CUR_OBJECT_ID	"Array64Chars"	%I1002.0	☐	☒	☒	☒		
5	IN_CUR_PROFILE_ID	Word	%W1067	☐	☒	☒	☒		
6	IN_INSPECTION_MODE_RESPONSE	Byte	%B1001	☐	☒	☒	☒		
7	IN_KEEP_ALIVE	Byte	%B1035	☐	☒	☒	☒		
8	IN_NEW_OBJECT_TRIGGER_RESPONSE	Byte	%B1066	☐	☒	☒	☒		
9	IN_OBJECT_INSPECTION_RESULT	Byte	%B1136	☐	☒	☒	☒		
10	IN_PROFILE_ID_RESPONSE	Byte	%B1069	☐	☒	☒	☒		
11	IN_ROIS_DEFECT_LIST	"Array64Chars"	%I1139.0	☐	☒	☒	☒		
12	IN_SYSTEM_ERROR	Byte	%B1137	☐	☒	☒	☒		
13	IN_SYSTEM_READY	Byte	%B1138	☐	☒	☒	☒		
14	OUT_BATCH_ID	"Array64Chars"	%Q1067.0	☐	☒	☒	☒		
15	OUT_INSPECTION_MODE	Byte	%B1000	☐	☒	☒	☒		
16	OUT_NEW_OBJECT_TRIGGER	"Array64Chars"	%Q1001.0	☐	☒	☒	☒		
17	OUT_PROFILE_ID	Word	%QW1065	☐	☒	☒	☒		
18	<Add new>			☐	☒	☒	☒		

Figure 2-21: Inspekto_tags PLC tags table

Each of the previous tags represents one of the supported functions by Inspekto which are described in the section 3.1 Inspekto functions.

3. Compile and download the project to the PLC.

Inspekto control TIA Portal function blocks

Once integrated into TIA Portal, you can easily call any of the Inspekto’s supported functions, such as setting the inspection mode, switching the inspection profile or triggering a new inspection using the corresponding TIA Portal function blocks that were built according to the Inspekto functions.

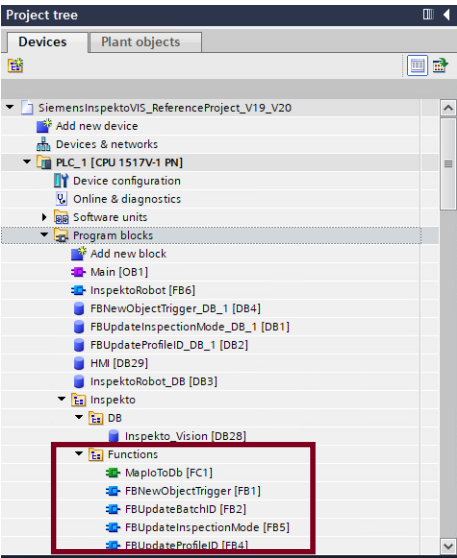


Figure 2-22: Inspekto TIA Porta function blocks

NOTE

The function blocks presented in this documentation are not included by default within TIA Portal and were specifically developed to allow easy and fast control of the Inspekto system via the PLC. These function blocks are included within the reference project available for downloading in the entry of this documentation so you can use them according to your use case.

NOTE

Detailed description of all available TIA Portal function blocks for controlling Inspekto via the PLC can be found in the following link: <https://support.industry.siemens.com/cs/ww/en/view/109977467>

To be able to use the above-mentioned FBs, additional programming elements such as data blocks (DBs) or custom PLC data types are needed:

- PLC data type – Data_In

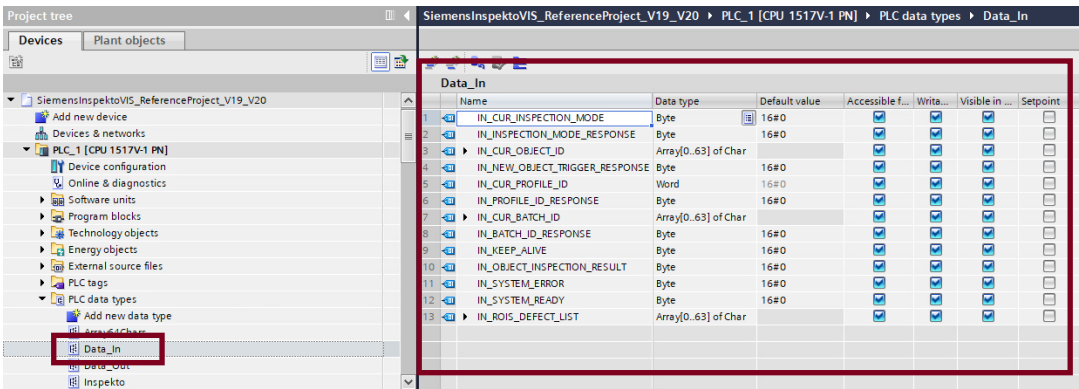


Figure 2-23: Inspekto custom PLC data type – Data_In

- PLC data type – Data_Out

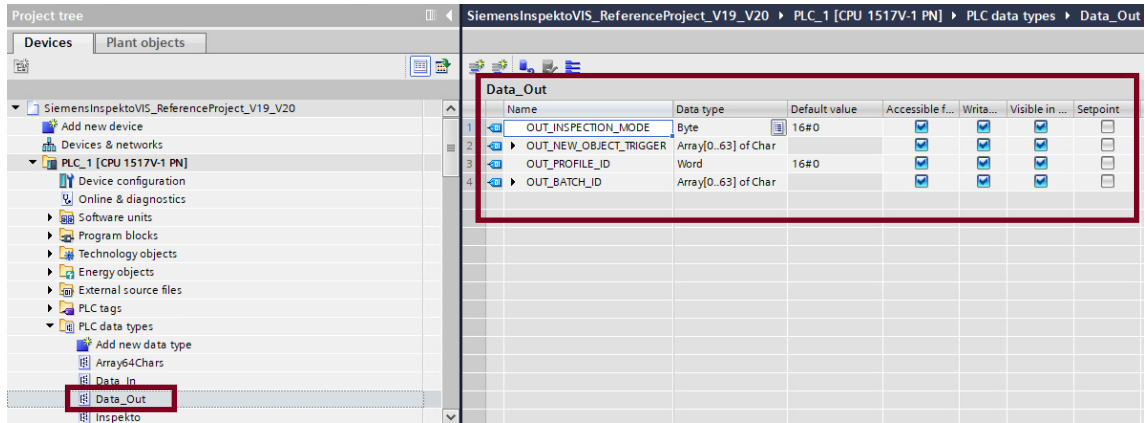


Figure 2-24: Inspekto custom PLC data type – Data_Out

- PLC data type – Inspekto

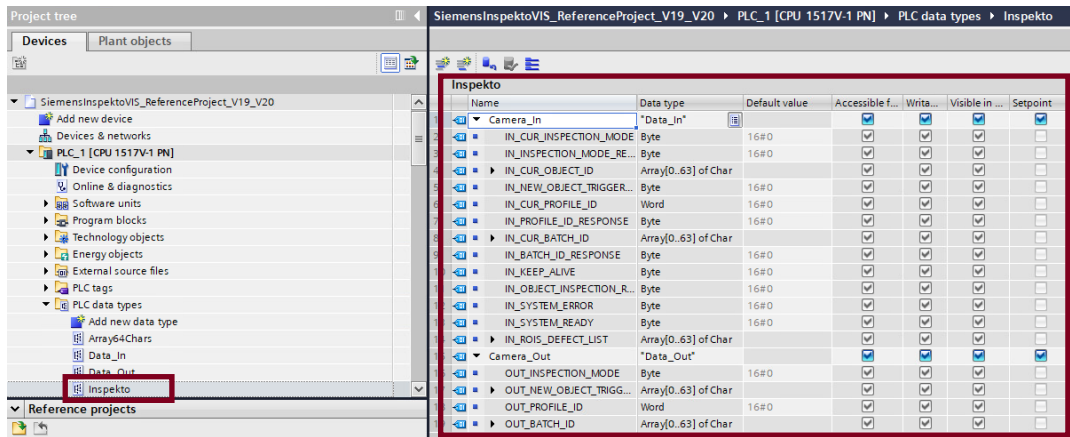


Figure 2-25: Inspekto custom PLC data type – Inspekto

- Data block – Inspekto_Vision

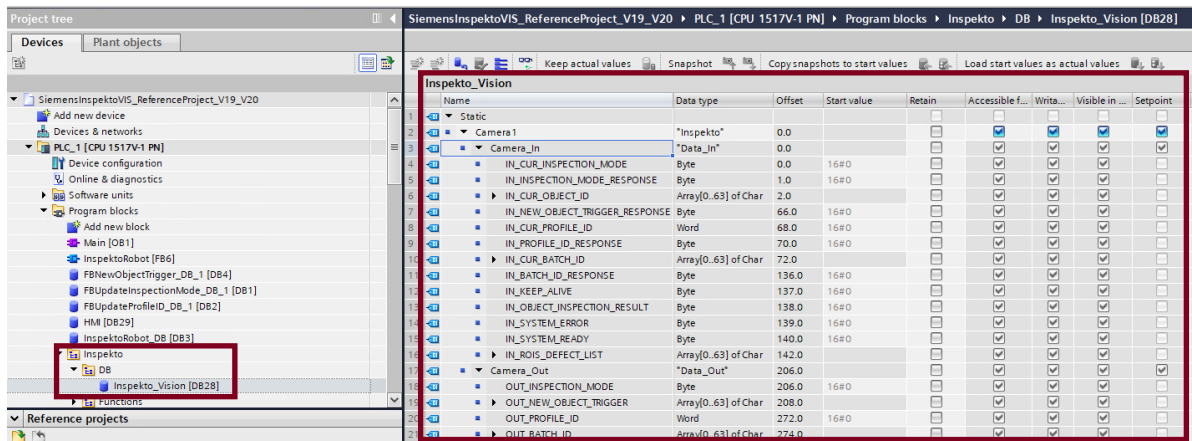


Figure 2-26: Inspekto_Vision TIA Porta data block

NOTE

All the function blocks and auxiliary elements are provided within the reference project. You can copy these elements to a new TIA Portal project as well. Based on your project, additional adjustment of the DB tags addresses might be needed based on the I/O address of your Inspekto device.

The I/O of the Inspekto device and the tags within the DB are linked using the function block FC "MapItoDb" which is parametrized with the starting addresses of the Inputs and Outputs of the Inspekto device and the number of "Inspekto_Vision" DB.

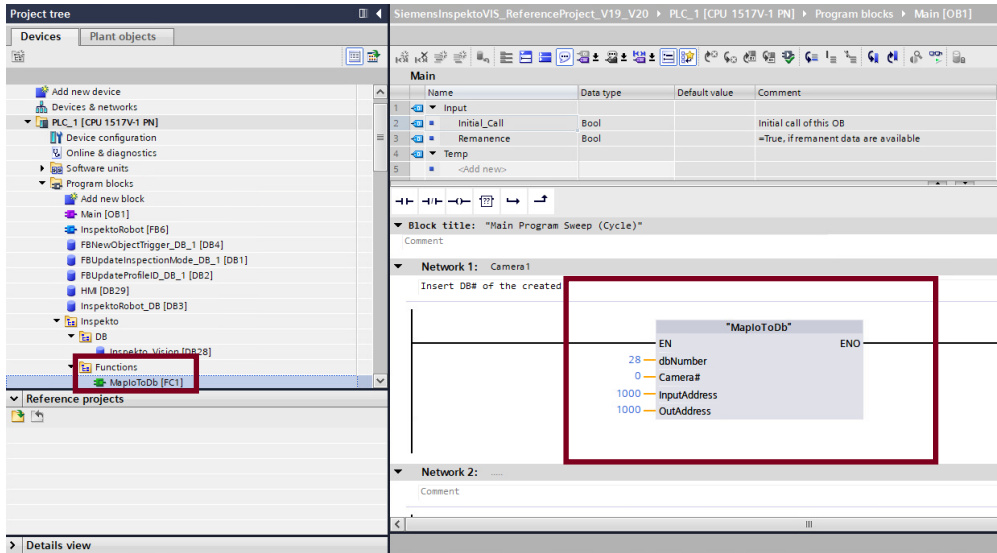


Figure 2-27: FC MaploToDb

Testing configuration and PLC-Inspekto connectivity

The Inspekto system includes a feature called Connectivity Feedback. This feature allows the user tests and ensures correct communication with all types of connectivity options available in the system.

After integrating the SIEMENS Inspekto VIS system within TIA Portal and after establishing the connection with the SIMATIC PLC using the Profinet, it is highly recommended before starting programming the function blocks for controlling the Inspekto system to test the configuration of this interface and verify that:

- Signals are correctly sent between the Inspekto system and the PLC, and
- Signals are received in the right place and form. For example, when sending an OUT_PROFILE_ID signal from the PLC with value of "2" we want to make sure that it is received with value of "2", and in the corresponding field (and not for example as a OUT_BATCH_ID command).

To access the Connectivity Feedback, access your Inspekto system UI, hover over the left side of the screen so the navigation menu opens, and then press on "Settings". Under the general tab scroll down to the connectivity section and press on the "Connectivity Feedback Interface" button:

Inspekto Settings

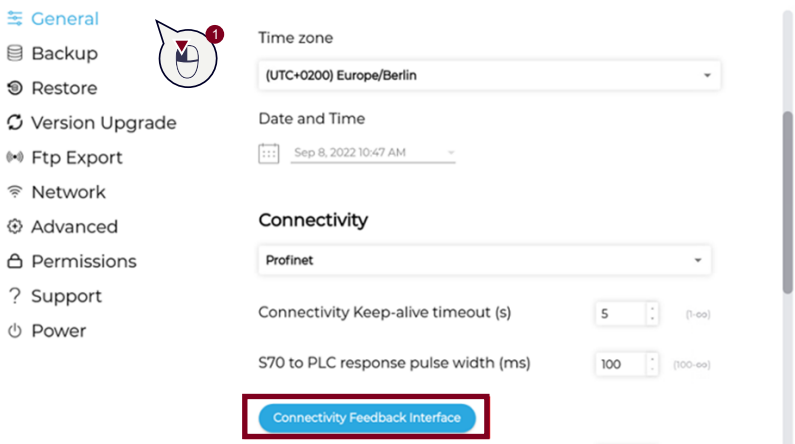


Figure 2-28: Inspekto Connectivity Feedback Interface

Once selected, the system will show a pop-up message stating that the system will now only show signals coming from the PLC but will not act on them. This means that when working on the "Connectivity Feedback Interface" mode, all signals coming from the PLC will be received and shown in the interface, but no action will take place. When closing the Connectivity Feedback Interface all PLC communication will return to normal state automatically.

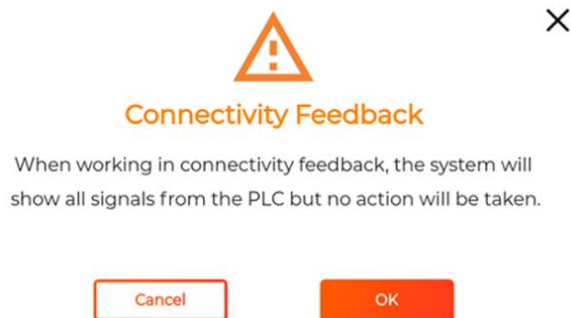


Figure 2-29: Activating Inspekto Connectivity Feedback Interface

The Connectivity Feedback screen has 2 tabs, one for messages going from Inspekto to the PLC, and the other from messages going from the PLC to Inspekto.

1. Inspekto → PLC feedback screen

This screen contains all the possible signals going from the SIEMENS Inspekto VIS system to the PLC known also as Inputs.

 The "Profinet feedback" screen with the "Inspekto → PLC" tab selected. It features a "Reset All" button and a "Send MSG to PLC" button. Under the "From Inspekto" section, there are input fields for:

- Cur. Insp. Mode (0-255)
- Insp. Mode Response (0-255)
- Obj. Inspection Result (0-255)
- Cur. Profile ID (0-65535)
- Cur. Object ID (64 Char)
- Obj. Trigger Response (0-255)
- Profile ID Status (0-255)
- Cur. Batch ID (64 Char)
- Batch ID Response (0-255)
- Sys Error (0-255)

 At the bottom right, there are two status indicators: "System Ready" and "Keep Alive", each with a red square icon.

Figure 2-30: Inspekto → PLC feedback screen

To send a command to the PLC and test the communication and configuration:

- Type a value for a signal you wish to send in the corresponding field and press the "Send MSG to PLC" button:

 The same "Profinet feedback" screen as in Figure 2-30, but with the "Send MSG to PLC" button highlighted by a red rectangle. Additionally, the "Cur. Insp. Mode (0-255)" input field is also highlighted by a red rectangle.

Figure 2-31: Send MSG to PLC from Inspekto

- b. Check that the signal value is received at the corresponding tag within the PLC data block. If the connection is established successfully, you will be able to see the content of the message in the corresponding tag.
- c. Repeat the same with all remaining OUT signals.

2. PLC → Inspekto feedback screen

This screen contains all the possible signals coming from the PLC to Inspekto. These signals are the outputs, OUT_INSPECTION_MODE, OUT_NEW_OBJECT_TRIGGER, OUT_PROFILE_ID and OUT_BATCH_ID.

When the connection is established successfully, the fields will be updated automatically every time a message is being sent from the PLC to Inspekto. To test the configuration and the communication is working properly:

- a. Go online with your PLC, open the Inspekto DB, activate the "Monitor all" feature and modify the value of every output tag.
- b. Check within the Connectivity Feedback Screen that the value of the tag you just modified changed accordingly.

2.5. Robot path programming & Inspekto profile training

Now you need to program the robot, so it moves the camera to each specific position required for the inspection. In parallel, it's advisable to train Inspekto. As you teach the robot to place the camera at a given target point, create and train the corresponding Inspekto profile; when that profile is finished, continue teaching the remaining robot points until you reach the next position that requires training another profile.

Robot programming consists of teaching the exact points where you want the robot to move and then defining the path the robot will follow i.e., the motion sequence that connects those points. This programming is performed with the SRL HMI included in the SRL/SRI package. In our case, we used a Mobile Panel and loaded the HMI project supplied in the SRL/SRI package. This HMI provides everything you need to configure the robot.



Figure 2-32: SRI HMI interface

You can find detailed instructions for using this HMI (point and path programming) in the documentation included with the SRL/SRI package.

2.5.1. Robot moving points

In the SRL/SRI project, some points and paths are provided by default as examples:

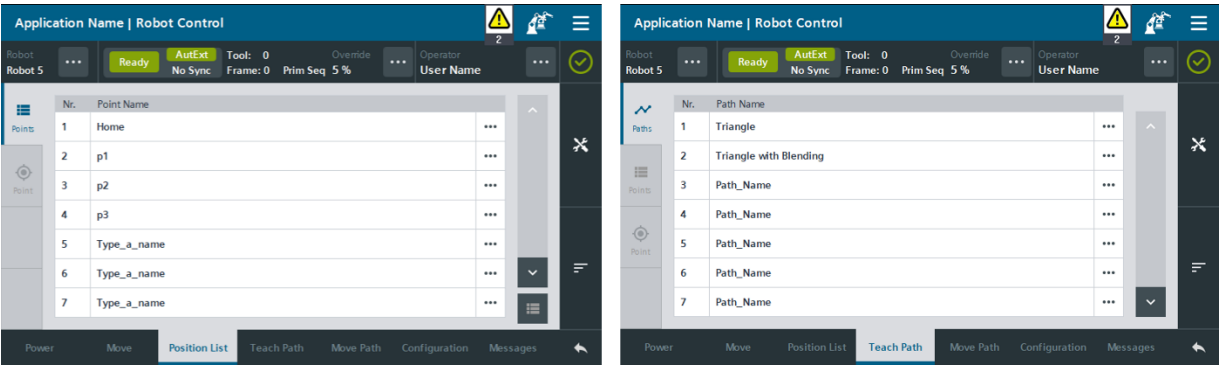


Figure 2-33: Default points/paths in the SRL HMI

In this case, the default points/paths were removed/reset since they will not be used, and new points will be defined from scratch:

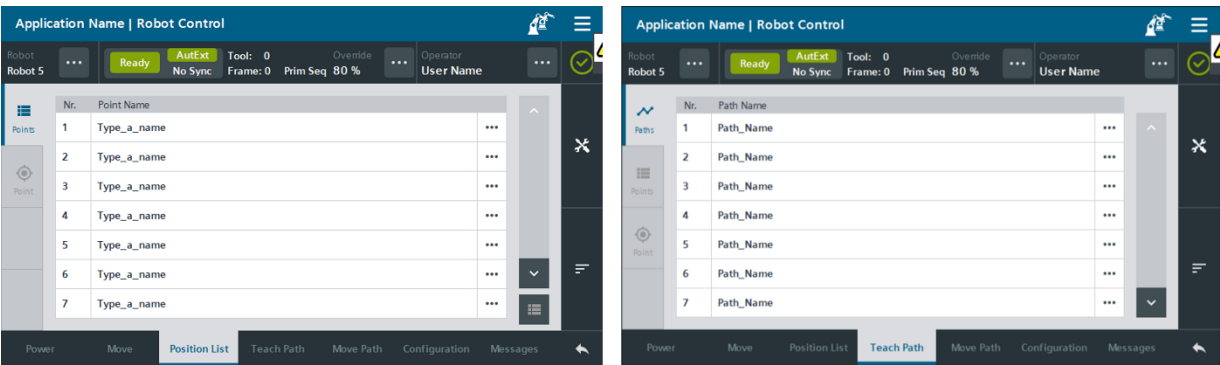


Figure 2-34: Reset points/paths in the SRL HMI

For this application example, the following relevant paths for the use case described were defined using the SRL through the dedicated HMI interface.

2.5.1.1. "Home" position:

This is the first point that was defined. For creating the point where the robot is waiting for a new command, the following steps were performed.

1. Check the robot status

Make sure the robot is powered on and there is no error messages displayed in the HMI screen, as shown below:

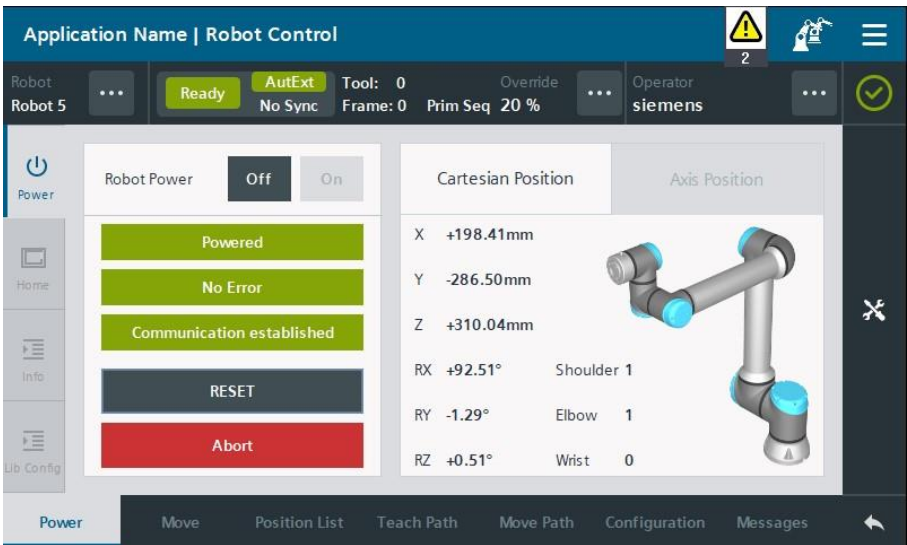


Figure 2-35: SRL HMI – Robot Powered On

2. Teach Position

Jog the robot to your desired position. Once the robot is in the correct position, select "Teach Position" to define the point:

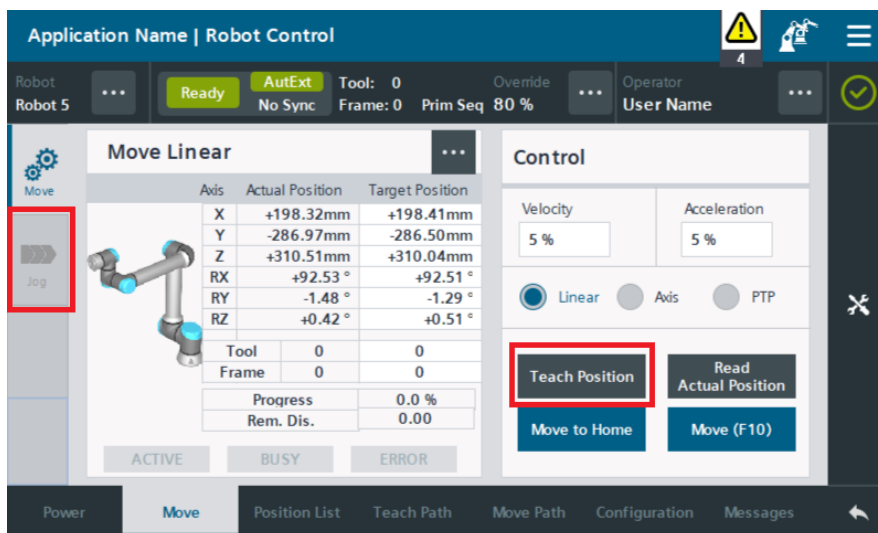


Figure 2-36: SRL HMI – Jog Robot and Teach Position

This is the robot's initial position before any sequence or movement path starts. The robot stays in this position until a new inspection cycle is triggered.



Figure 2-37: Robot's home position

3. Name the point

Assign a name to the newly defined point, in this example: "Home" and click "Teach":

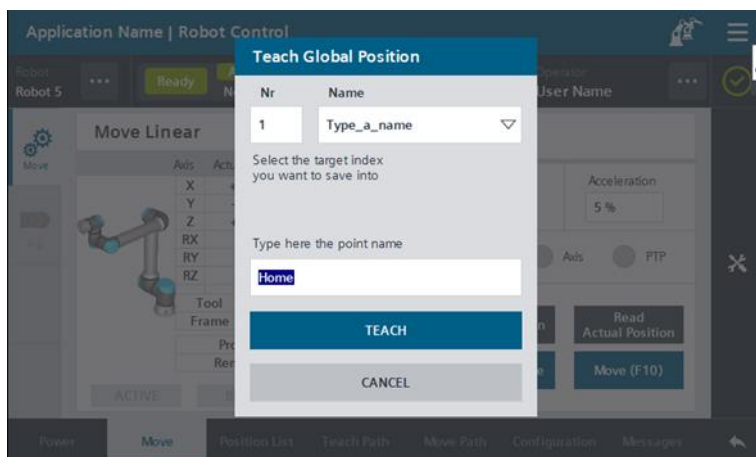


Figure 2-38: SRL HMI – Name point.

4. Verify in the point list

If you go to the point list, you will see the new point added:

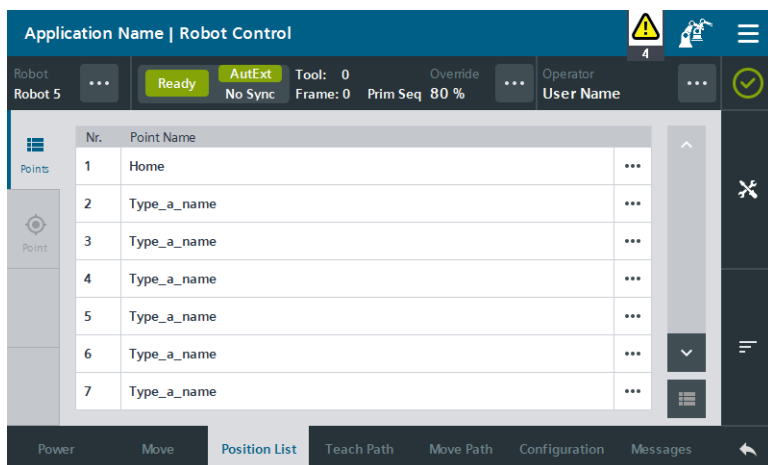


Figure 2-39: SRL HMI – "Home" position on the Position List

Once you have configured the point in **TIA Portal**, it will appear in the **RobotDataSRL DB** as actual value.

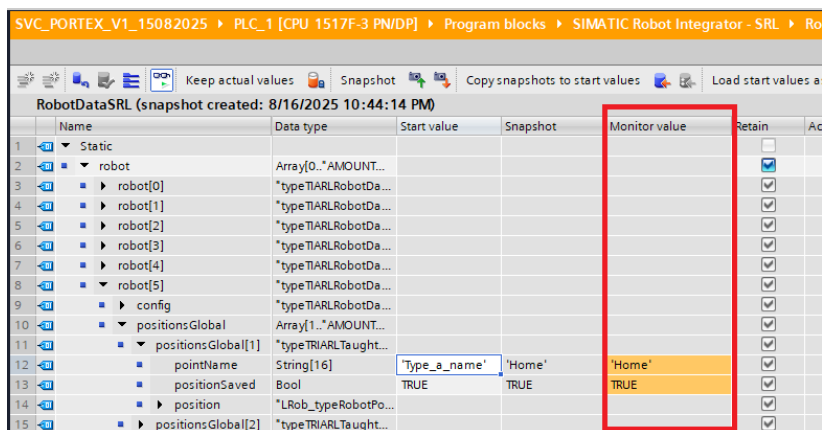


Figure 2-40: RobotDataSRL DB – "Home" as actual value

However, you still need to set these values as *Start Values*:

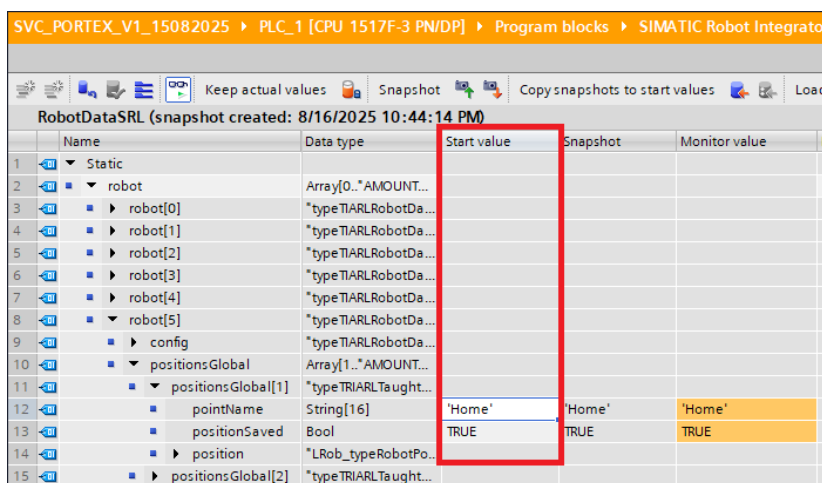


Figure 2-41: RobotDataSRL DB – "Home" as start value

NOTE

In this case, each point was defined and copied individually as a *Start Value*. After all points/paths have been defined and copied as *Start Values* in the **RobotDataSRL DB**, you can download the program to the PLC.

If you do not copy the values as *Start Values* in TIA Portal, you may lose the points/paths you have defined.

The screenshot shows the RobotDataSRL database with a snapshot created on 7/12/2025 at 1:09:29 PM. The table lists various robot data points, with several rows highlighted by red boxes to indicate the 'Home' position data.

Name	Data type	Start value	Retain	Accessible f...	Writa...	Visible in ...	Setpoint	Supervision	Comment
positions:Global	Array[1..-AMOU...		☒	☒	☒	☒	☒	☒	By the user taught positions
positions:Global[1]	*typeTIAARLTaught		☒	☒	☒	☒	☒	☒	By the user taught positions
pointName	String[16]	'Home'	☒	☒	☒	☒	☒	☒	
positionSaved	Bool	TRUE	☒	☒	☒	☒	☒	☒	
position	*LRob_typeRobotPo...		☒	☒	☒	☒	☒	☒	LRob_typeRobotPosition
Cartesian	*LRob_typeRobotCa...		☒	☒	☒	☒	☒	☒	Robot cartesian position data values
Position	*LRob_typeRobotSh...		☒	☒	☒	☒	☒	☒	Cartesian position of the TCP and external axi...
X	Real	198.4072	☒	☒	☒	☒	☒	☒	[mm]: TCP Position on the X-Axis
Y	Real	-286.4971	☒	☒	☒	☒	☒	☒	[mm]: TCP Position on the Y-Axis
Z	Real	310.0373	☒	☒	☒	☒	☒	☒	[mm]: TCP Position on the Z-Axis
RX	Real	92.50785	☒	☒	☒	☒	☒	☒	[°]: Rotation around the X-Axis (roll)
RY	Real	-1.288705	☒	☒	☒	☒	☒	☒	[°]: Rotation around on the Y-Axis (pitch)
RZ	Real	0.5131171	☒	☒	☒	☒	☒	☒	[°]: Rotation around on the Z-Axis (yaw)
Config	*LRob_typeRobotC...		☒	☒	☒	☒	☒	☒	Configuration data of the robot
TurnNumber	Array[0..3] of Byte		☒	☒	☒	☒	☒	☒	Turn number of the axes J1to J6 and E1
E1	Real	0.0	☒	☒	☒	☒	☒	☒	[°(mm)]: Position of first external axis
ExtendedPositi...	*LRob_typeRobotEx...		☒	☒	☒	☒	☒	☒	Position of the external axes E2 to E6
Joint	*LRob_typeRobotSh...		☒	☒	☒	☒	☒	☒	Robot joint position data values
J1	Real	107.1817	☒	☒	☒	☒	☒	☒	[°(mm)]: Position of the axes J1 to J6 and E1
J2	Real	-106.0657	☒	☒	☒	☒	☒	☒	[°(mm)]: Position of first joint of the robot
J3	Real	129.5677	☒	☒	☒	☒	☒	☒	[°(mm)]: Position of second joint of the robot
J4	Real	-20.855	☒	☒	☒	☒	☒	☒	[°(mm)]: Position of third joint of the robot
J5	Real	106.6829	☒	☒	☒	☒	☒	☒	[°(mm)]: Position of fourth joint of the robot
J6	Real	2.238628	☒	☒	☒	☒	☒	☒	[°(mm)]: Position of fifth joint of the robot
E1	Real	0.0	☒	☒	☒	☒	☒	☒	[°(mm)]: Position of sixth joint of the robot
ExtendedJoint	*LRob_typeRobotC...		☒	☒	☒	☒	☒	☒	Position of the external axes E2 to E6
CoordinateSystem	*LRob_typeRobotC...		☒	☒	☒	☒	☒	☒	Cartesian coordinate systems data

Figure 2-42: "Home" position copied as start value and saved into the TIA PORTAL project.

Repeat this procedure for each additional point you need to define.

2.5.1.2. "back0" position:

The robot approaches the rear of the object along the X-axis.



Figure 2-43: "back0" robot's position

SiemensInspektioVIS_ReferenceProject_V19_V20 PLC_1 [CPU 1517V-1 PN] Program blocks SIMATIC Robot Integrator - SRL RobotDataSRL [DB5133]									
RobotDataSRL (snapshot created: 7/12/2025 1:09:29 PM)									
Name	Data type	Start value	Retain	Accessible f...	Write...	Visible in ...	Setpoint	Supervision	Comment
37	CoordinateSystem	"LRob_typeRobotC...							Cartesian coordinate systems data
38	positionsGlobal[2]	"typeTRIARL Taught							By the user taught positions
39	pointName	String[16]							
40	positionSaved	Bool							
41	position	"LRob_typeRobotPo...							LRob_typeRobotPosition
42	Cartesian	"LRob_typeRobotCa...							Robot cartesian position data values
43	Position	"LRob_typeRobotSh...							Cartesian position of the TCP and external axi...
44	X	Real							[mm]: TCP Position on the X-Axis
45	Y	Real							[mm]: TCP Position on the Y-Axis
46	Z	Real							[mm]: TCP Position on the Z-Axis
47	RX	Real							[°]: Rotation around the X-Axis (roll)
48	RY	Real							[°]: Rotation around the Y-Axis (pitch)
49	RZ	Real							[°]: Rotation around the Z-Axis (yaw)
50	Config	"LRob_typeRobotC...							Configuration data of the robot
51	TurnNumber	Array[0..3] of Byte							Turn number of the axes J1 to J6 and E1
52	E1	Real							[°/mm]: Position of first external axis
53	ExtendedPositi...	"LRob_typeRobotEx...							Position of the external axes E2 to E6
54	Joint	"LRob_typeRobotJo...							Robot joint position data values
55	Joint	"LRob_typeRobotSh...							Joint position of the axes J1 to J6 and E1
56	J1	Real							[°/mm]: Position of first joint of the robot
57	J2	Real							[°/mm]: Position of second joint of the robot
58	J3	Real							[°/mm]: Position of third joint of the robot
59	J4	Real							[°/mm]: Position of fourth joint of the robot
60	J5	Real							[°/mm]: Position of fifth joint of the robot
61	J6	Real							[°/mm]: Position of sixth joint of the robot
62	E1	Real							[°/mm]: Position of first external axis
63	ExtendedJoint	"LRob_typeRobotEx...							Position of the external axes E2 to E6
64	CoordinateSystem	"LRob_typeRobotC...							Cartesian coordinate systems data

Figure 2-44: "back0" position copied as start value and saved into the TIA PORTAL project.

2.5.1.3. "back1" position:

The robot rotates the camera to align it with the area of the S7-1500 to be inspected.

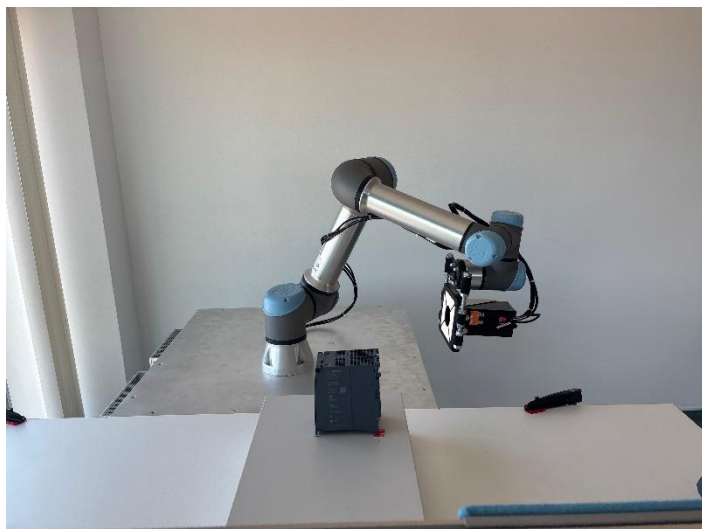


Figure 2-45: "back1" robot's position.

SiemensInspektioVIS_ReferenceProject_V19_V20 PLC_1 [CPU 1517V-1 PN] Program blocks SIMATIC Robot Integrator - SRL RobotDataSRL [DB5133]									
RobotDataSRL (snapshot created: 7/12/2025 1:09:29 PM)									
Name	Data type	Start value	Retain	Accessible f...	Write...	Visible in ...	Setpoint	Supervision	Comment
64	CoordinateSystem	"LRob_typeRobotC...							Cartesian coordinate systems data
65	positionsGlobal[3]	"typeTRIARL Taught							By the user taught positions
66	pointName	String[16]							
67	positionSaved	Bool							
68	position	"LRob_typeRobotPo...							LRob_typeRobotPosition
69	Cartesian	"LRob_typeRobotCa...							Robot cartesian position data values
70	Position	"LRob_typeRobotSh...							Cartesian position of the TCP and external axi...
71	X	Real							[mm]: TCP Position on the X-Axis
72	Y	Real							[mm]: TCP Position on the Y-Axis
73	Z	Real							[mm]: TCP Position on the Z-Axis
74	RX	Real							[°]: Rotation around the X-Axis (roll)
75	RY	Real							[°]: Rotation around the Y-Axis (pitch)
76	RZ	Real							[°]: Rotation around the Z-Axis (yaw)
77	Config	"LRob_typeRobotC...							Configuration data of the robot
78	TurnNumber	Array[0..3] of Byte							Turn number of the axes J1 to J6 and E1
79	E1	Real							[°/mm]: Position of first external axis
80	ExtendedPositi...	"LRob_typeRobotEx...							Position of the external axes E2 to E6
81	Joint	"LRob_typeRobotJo...							Robot joint position data values
82	Joint	"LRob_typeRobotSh...							Joint position of the axes J1 to J6 and E1
83	J1	Real							[°/mm]: Position of first joint of the robot
84	J2	Real							[°/mm]: Position of second joint of the robot
85	J3	Real							[°/mm]: Position of third joint of the robot
86	J4	Real							[°/mm]: Position of fourth joint of the robot
87	J5	Real							[°/mm]: Position of fifth joint of the robot
88	J6	Real							[°/mm]: Position of sixth joint of the robot
89	E1	Real							[°/mm]: Position of first external axis
90	ExtendedJoint	"LRob_typeRobotEx...							Position of the external axes E2 to E6
91	CoordinateSystem	"LRob_typeRobotC...							Cartesian coordinate systems data

Figure 2-46: "back1" position copied as start value and saved into the TIA PORTAL project.

2.5.1.4. "backInspection" position:

Lastly, the robot points the camera directly at the backside of the S7-1500 PLC to be inspected.

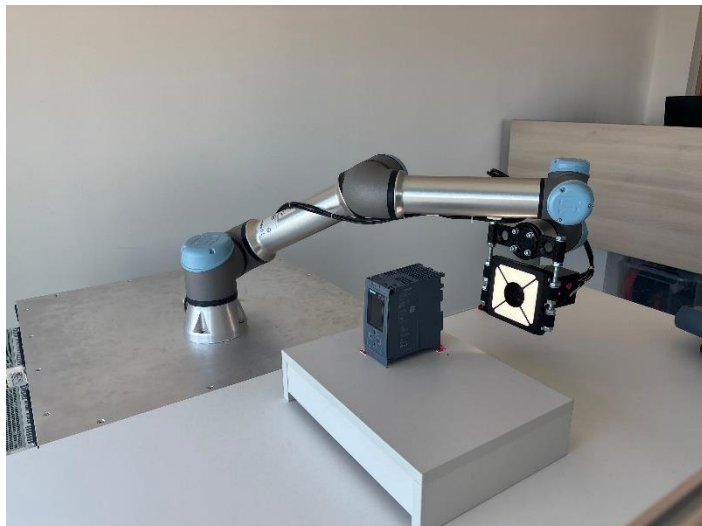


Figure 2-47: "backInspection" robot's position

SiemensInspektoVIS_ReferenceProject_V19_V20 > PLC_1 [CPU 1517V-1 PN] > Program blocks > SIMATIC Robot Integrator - SRL > RobotDataSRL [DB5133]

RobotDataSRL (snapshot created: 7/12/2025 1:09:29 PM)

Name	Data type	Start value	Retain	Accessible f...	Write...	Visible in ...	Setpoint	Supervision	Comment
91. CoordinateSystem	*LRob_typeRobotC...								Cartesian coordinate systems data
92. positionsGlobal[4]	*typeTRARLtaught...								By the user taught positions
93. positionName	String[16]	'backInspection'							
94. positionSaved	Bool	TRUE							
95. position	*LRob_typeRobotPo...								LRob_typeRobotPosition
96. Cartesian	*LRob_typeRobotCa...								Robot cartesian position data values
97. Position	*LRob_typeRobotSh...								Cartesian position of the TCP and external axi...
98. X	Real	482.2047							[mm]: TCP Position on the X-Axis
99. Y	Real	-516.1247							[mm]: TCP Position on the Y-Axis
100. Z	Real	261.397							[mm]: TCP Position on the Z-Axis
101. RX	Real	87.9474							[°]: Rotation around the X-Axis (roll)
102. RY	Real	0.9392985							[°]: Rotation around the Y-Axis (pitch)
103. RZ	Real	-90.10699							[°]: Rotation around the Z-Axis (yaw)
104. Config	*LRob_typeRobotCo...								Configuration data of the robot
105. TurnNumber	Array[0..3] of Byte								Turn number of the axes J1to J6 and E1
106. E1	Real	0.0							[°(mm)]: Position of first external axis
107. ExtendedPositi...	*LRob_typeRobotEx...								Position of the external axes E2 to E6
108. Joint	*LRob_typeRobotio...								Robot joint position data values
109. Joint	*LRob_typeRobotSh...								Joint position of the axes J1 to J6 and E1
110. J1	Real	128.5139							[°(mm)]: Position of first joint of the robot
111. J2	Real	-26.76538							[°(mm)]: Position of second joint of the robot
112. J3	Real	26.21702							[°(mm)]: Position of third joint of the robot
113. J4	Real	3.784563							[°(mm)]: Position of fourth joint of the robot
114. J5	Real	218.6123							[°(mm)]: Position of fifth joint of the robot
115. J6	Real	1.540929							[°(mm)]: Position of sixth joint of the robot
116. E1	Real	0.0							[°(mm)]: Position of first external axis
117. ExtendedJoint	*LRob_typeRobotEx...								Position of the external axes E2 to E6
118. CoordinateSystem	*LRob_typeRobotC...								Cartesian coordinate systems data

Figure 2-48: "backInspection" position copied as start value and saved into the TIA PORTAL project.

To ensure the Inspekto system detects the object to be inspected and the possible defects, the camera should be placed in a position where the inspected object is clearly visible in the Inspekto UI. Once the robot is in the position where the inspection should be triggered, you can start training the inspection profile for that specific side of your object.

In this case, once the UR5e reaches the "backInspection" position, we switch to the Inspekto software to start the creation and training of the inspection profile that checks the electrical connector on the backside of the S7-1500 PLC. When the profile training is complete, you can continue programming the remaining position points for the robot using through the Mobile Panel.

Inspekto Profile training – backside inspection

The Inspekto system performs inspections using inspection profiles. A profile represents a specific view of the object where you define Regions of Interest (ROI), which are the exact portion of the object the system will check for defects.

In this application example, we analyze two views of the inspected object (an S7-1500 PLC), so we configure two profiles—one per view. For this section, we create the "backPLC" profile that will handle the inspection of the pins (bent, missing, or misaligned) of the electric connector located on the backside of the S7-1500 PLC.

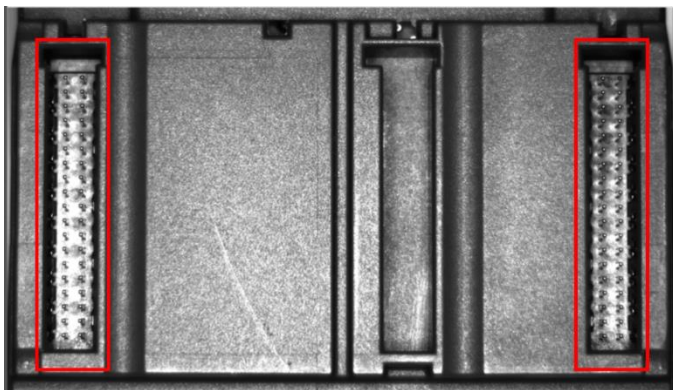


Figure 2-49: Inspection areas of the "backPLC" inspection profile

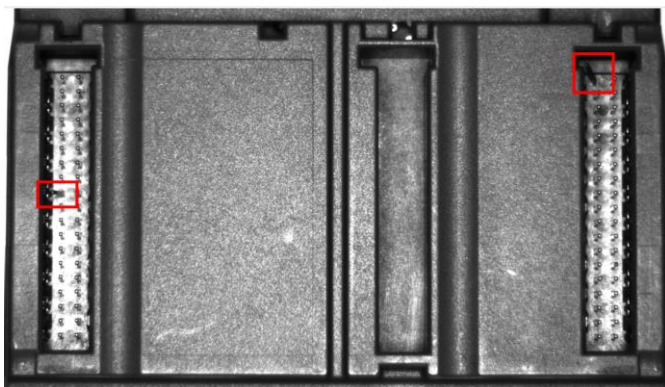


Figure 2-50: Defective pins detected by the Inspekto system

To create the inspection profile, follow the steps below.

NOTE

For detailed step-by-step instructions about profile creating and training please refer to:

[Creating an Inspection Profile](#)

[Easy 6 step setup for PCB inspection](#)

1. Create Profile

Create a new profile as shown in the screenshot below. Inspekto supports two profile types: one for moving objects and one for stationary objects. Given the use case presented, both profiles are "Stationary".

Figure 2-51: Inspekto profile creation starting page

2. Position Camera

Center the camera on the object. Ensure that the inspection area is clearly visible in the field of view of the Inspekto camera.



Figure 2-52: Field of view of Inspekto camera

NOTE

Based on the position the camera was mounted on the robot, the scene captured by the camera might appear inverted in the UI. To avoid disassembling the camera, you can invert the image in the Inspekto software when configuring the camera's zoom.

3. Set Zoom

Adjust how close the camera should frame the object. Using the optical zoom, you can fine-tune the field of view of the camera and get a more detailed image of the object being inspected.

If the image is inverted, rotate it using the "Rotate Image" button within the user interface.

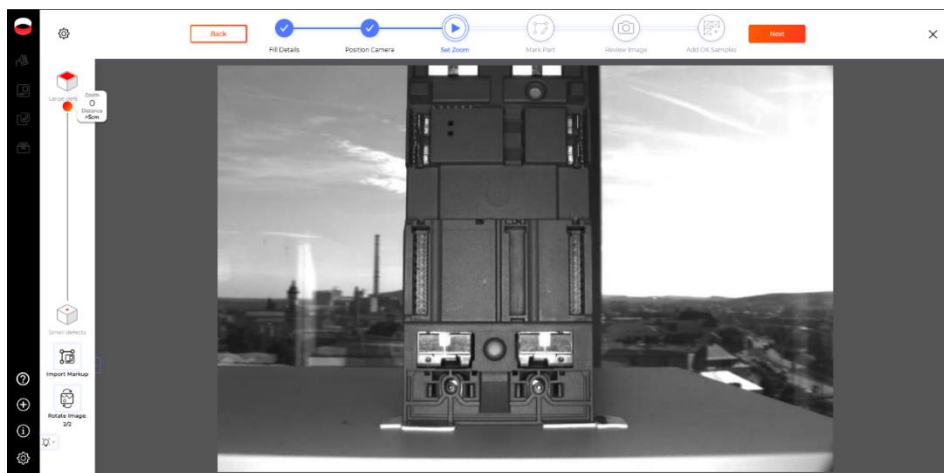


Figure 2-53: Field of view of Inspekto camera after inverting the image

To ensure the pins are clearly visible in the image and that the defects are successfully detected, the camera's zoom was set to 60% for this use case.

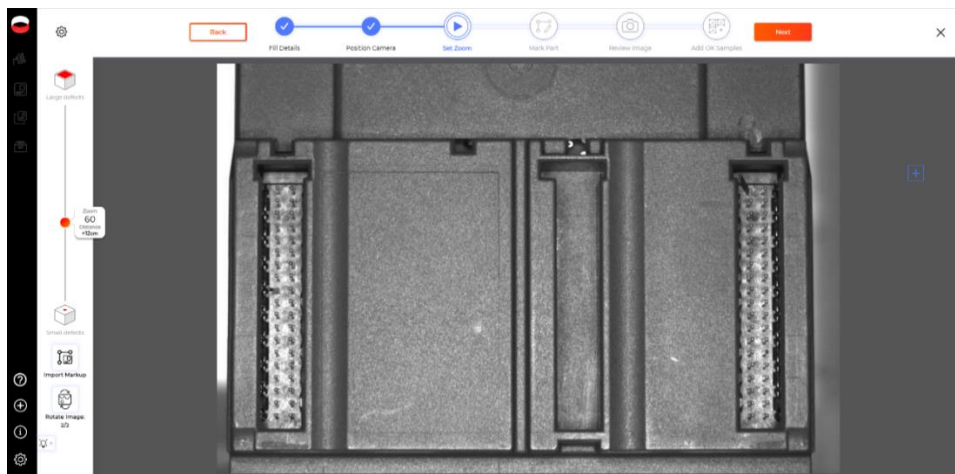


Figure 2-54: Field of view of Inspekto camera with 60% zoom.

4. Mark part

Define the outline of the area of the object to be inspected and the region of inspection (ROIs) where the Inspekto system will perform the inspection. If there are any defects out of the defined ROIs the system will not report the object as faulty.

For this use case Inspection type ROIs are used, because we want Inspekto to detect defects on the connector pins.

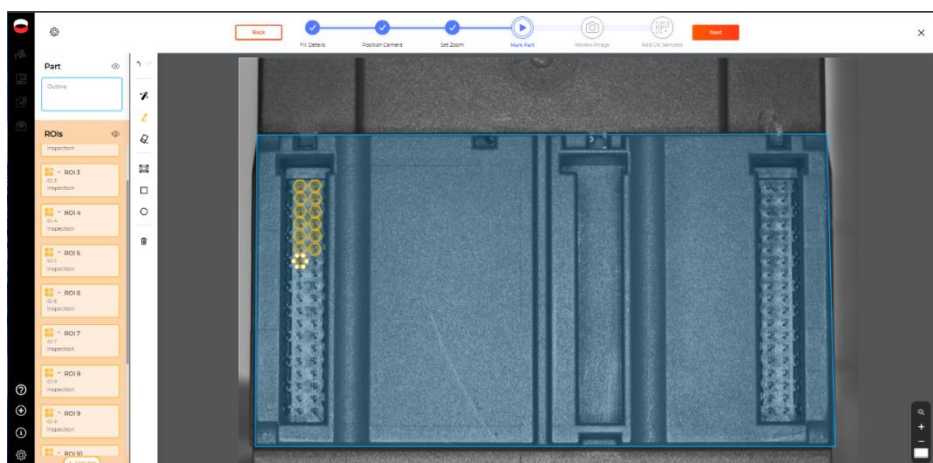


Figure 2-55: Profile markup and ROIs definition

An Inspection ROI was created per each pin of the electrical connector on the back of the PLC.

5. Imaging Mode

After completing the markup, Inspekto lets you choose an image mode. The best choice depends on the object's characteristics.

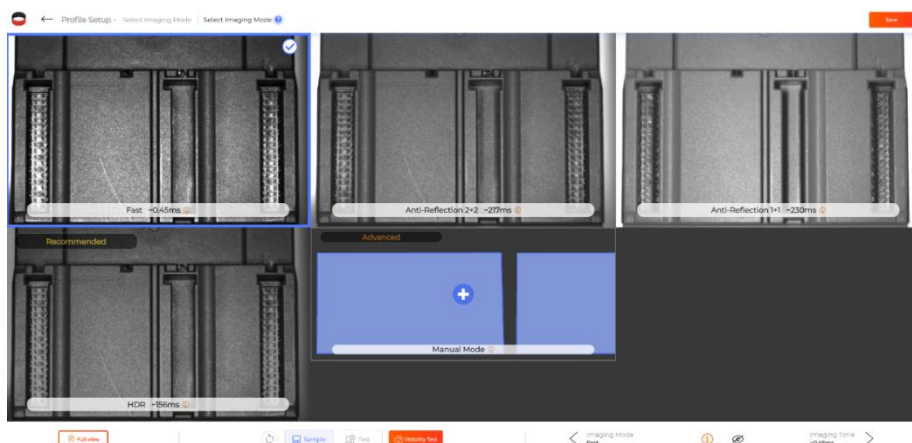


Figure 2-56: Imaging modes

In our case, the object is not highly reflective, so we select the Fast mode, which also shows the estimated capture and processing time. After confirming the imaging mode, the Inspekto captures a picture of the object and add it into the training sample set.

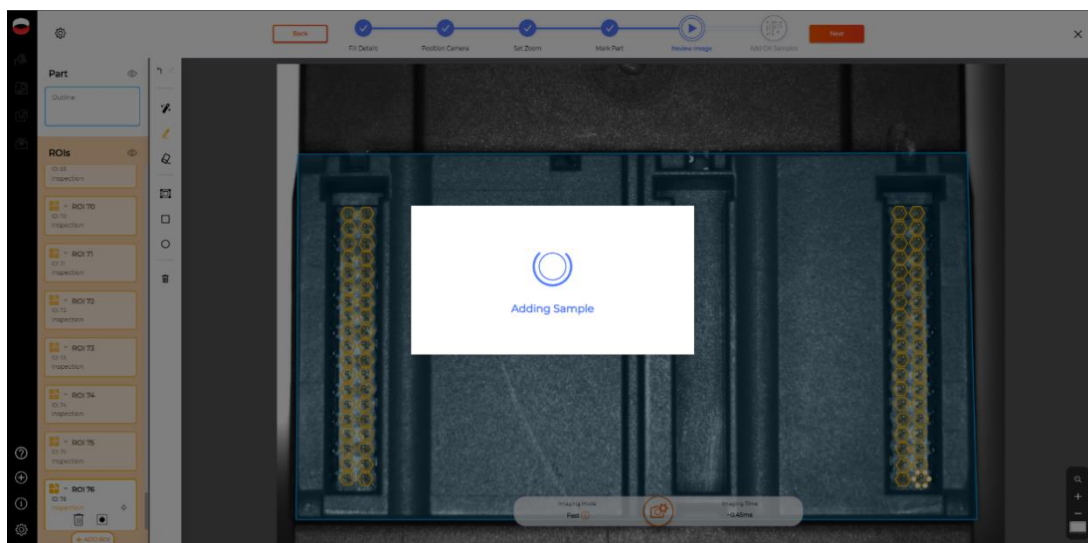


Figure 2-57: Fast mode selected and next step

6. Training sample set

To ensure robust operation and reduce misdetections or false positives, the profile requires at least 20 good (OK) samples during creation.

Place a new S7-1500 PLC in the inspection area and repeat the capture process until the profile shows 20/20 OK samples. Once 20 samples are collected, the profile is ready for training.

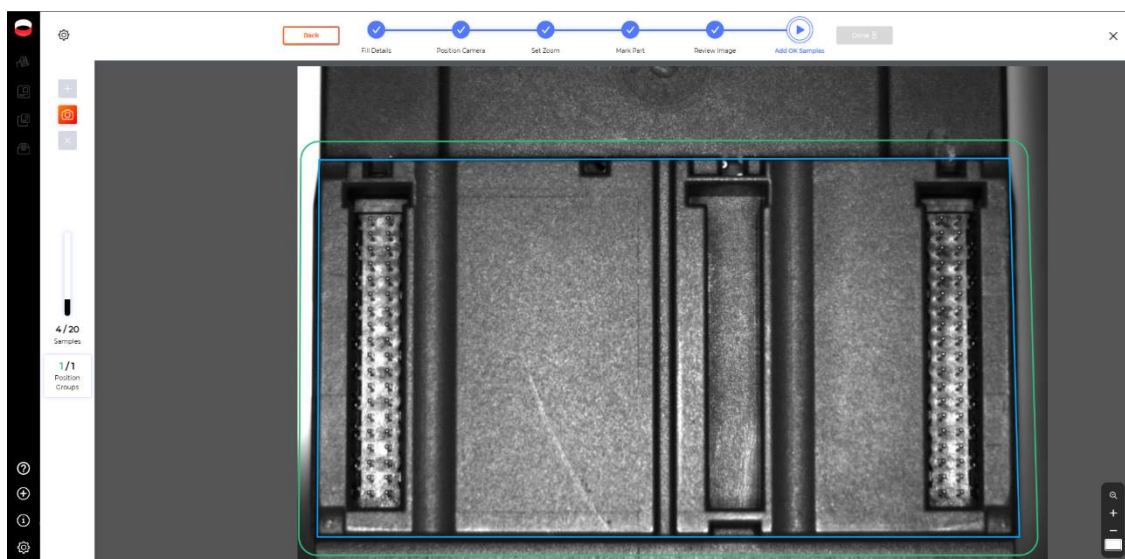


Figure 2-58: Sample set creation

NOTE

You should create as many profiles as there are distinct views of the object you want to inspect.

To ensure proper operation of the profiles and achieve the highest inspection accuracy while avoiding false positives or misdetections, it is important to perform profile training with the same lighting conditions and in the same environment where the system will run in production.

After uploading the 20 samples, the profile is created and a unique ID is assigned. This ID is later used to switch between profiles from the PLC via the dedicated TIA Portal function blocks. In this case, the Profile ID is 8.

Now that the setup of the first Inspekto profile is finished, you can continue programming the remaining robot points until you reach the next inspection position where another Inspekto profile should be trained.

2.5.1.5. "front0" position:

The robot approaches the front of the object along the X-axis.



Figure 2-59: "front0" robot position

SiemensInspektVIS_ReferenceProject_V19_V20 | PLC_1 [CPU 1517V-1 PN] | Program blocks | SIMATIC Robot Integrator - SRL | RobotDataSRL [DB5133]

RobotDataSRL (snapshot created: 7/12/2025 1:09:29 PM)

Name	Data type	Start value	Retain	Accessible f...	Write...	Visible in ...	Setpoint	Supervision	Comment
CoordinateSystem	*LRob_typeRobotCo...								Cartesian coordinate systems data
positionsGlobal[5]	*typeTRARLTaught...								By the user taught positions
pointName	String[16]	"front0"							
positionSaved	Bool	TRUE							
position	*LRob_typeRobotPo...								LRob_typeRobotPosition
Cartesian	*LRob_typeRobotCa...								Robot cartesian position data values
Position	*LRob_typeRobotSh...								Cartesian position of the TCP and external ax...
X	Real	-153.5395							[mm]: TCP Position on the X-Axis
Y	Real	-355.1254							[mm]: TCP Position on the Y-Axis
Z	Real	310.0073							[mm]: TCP Position on the Z-Axis
RX	Real	92.50979							[°]: Rotation around the X-Axis (roll)
RY	Real	-1.29275							[°]: Rotation around the Y-Axis (pitch)
RZ	Real	0.5087076							[°]: Rotation around the Z-Axis (yaw)
Contg	*LRob_typeRobot...								Configuration data of the robot
TurnNumber	Array[0..3] of Byte								Turn number of the axes J1 to J6 and E1
E1	Real	0.0							[°(mm)]: Position of first external axis
ExtendedPositi...	*LRob_typeRobotEx...								Position of the external axes E2 to E6
Joint	*LRob_typeRobotio...								Robot joint position data values
Joint	*LRob_typeRobotSh...								Joint position of the axes J1 to J6 and E1
J1	Real	32.35442							[°(mm)]: Position of first joint of the robot
J2	Real	-100.8658							[°(mm)]: Position of second joint of the robot
J3	Real	125.843							[°(mm)]: Position of third joint of the robot
J4	Real	-20.59059							[°(mm)]: Position of fourth joint of the robot
J5	Real	31.9659							[°(mm)]: Position of fifth joint of the robot
J6	Real	-2.329849							[°(mm)]: Position of sixth joint of the robot
E1	Real	0.0							[°(mm)]: Position of first external axis
ExtendedJoint	*LRob_typeRobot...								Position of the external axes E2 to E6
CoordinateSystem	*LRob_typeRobotCo...								Cartesian coordinate systems data

Figure 2-60: "front0" position copied as start value and saved into the TIA PORTAL project

2.5.1.6. "front1" position:

The robot rotates the camera.



Figure 2-61: "front1" robot position.

SiemensInspektoVIS_ReferenceProject_V19_V20 | PLC_1 [CPU 1517V-1 PN] | Program blocks | SIMATIC Robot Integrator - SRL | RobotDataSRL [DB5133]

RobotDataSRL (snapshot created: 7/12/2025 1:09:29 PM)

Name	Data type	Start value	Retain	Accessible f...	Write...	Visible in ...	Setpoint	Supervision	Comment
CoordinateSystem	"LRob_typeRobotC...								Cartesian coordinate systems data
positionsGlobal[5]	"typeTRARLtaught...								By the user taught positions
pointName	String[16]	'front0'							
positionSaved	Bool	TRUE							
position	"LRob_typeRobotPo...								LRob_typeRobotPosition
Cartesian	"LRob_typeRobotCa...								Robot cartesian position data values
Position	"LRob_typeRobotSh...								Cartesian position of the TCP and external axis...
X	Real	-153.53995							[mm]: TCP Position on the X-Axis
Y	Real	-355.1254							[mm]: TCP Position on the Y-Axis
Z	Real	310.0073							[mm]: TCP Position on the Z-Axis
RX	Real	92.50979							[°]: Rotation around the X-Axis (roll)
RY	Real	-1.29275							[°]: Rotation around the Y-Axis (pitch)
RZ	Real	0.5087076							[°]: Rotation around the Z-Axis (yaw)
Config	"LRob_typeRobotC...								Configuration data of the robot
TurnNumber	Array[0..3] of Byte								Turn number of the axes J1 to J6 and E1
E1	Real	0.0							[°mm]: Position of first external axis
ExtendedPositi...	"LRob_typeRobotEx...								Position of the external axes E2 to E6
Joint	"LRob_typeRobotJo...								Robot joint position data values
Joint	"LRob_typeRobotSh...								Joint position of the axes J1 to J6 and E1
J1	Real	32.35442							[°mm]: Position of first joint of the robot
J2	Real	-100.8658							[°mm]: Position of second joint of the robot
J3	Real	125.843							[°mm]: Position of third joint of the robot
J4	Real	-20.59059							[°mm]: Position of fourth joint of the robot
J5	Real	31.9659							[°mm]: Position of fifth joint of the robot
J6	Real	-2.329849							[°mm]: Position of sixth joint of the robot
E1	Real	0.0							[°mm]: Position of first external axis
ExtendedJoint	"LRob_typeRobotEx...								Position of the external axes E2 to E6
CoordinateSystem	"LRob_typeRobotC...								Cartesian coordinate systems data

Figure 2-62: "front1" position copied as start value and saved into the TIA PORTAL project

2.5.1.7. "frontInspection" position:

The robot positions the camera directly in front of the area to be inspected.



Figure 2-63: "frontInspection" robot position

SiemensInspektoVIS_ReferenceProject_V19_V20 | PLC_1 [CPU 1517V-1 PN] | Program blocks | SIMATIC Robot Integrator - SRL | RobotDataSRL [DB5133]

RobotDataSRL (snapshot created: 7/12/2025 1:09:29 PM)

Name	Data type	Start value	Retain	Accessible f...	Write...	Visible in ...	Setpoint	Supervision	Comment
CoordinateSystem	"LRob_typeRobotC...								Cartesian coordinate systems data
positionsGlobal[7]	"typeTRARLtaught...								By the user taught positions
pointName	String[16]	'frontInspection'							
positionSaved	Bool	TRUE							
position	"LRob_typeRobotPo...								LRob_typeRobotPosition
Cartesian	"LRob_typeRobotCa...								Robot cartesian position data values
Position	"LRob_typeRobotSh...								Cartesian position of the TCP and external axis...
X	Real	-134.8264							[mm]: TCP Position on the X-Axis
Y	Real	-499.3926							[mm]: TCP Position on the Y-Axis
Z	Real	266.1055							[mm]: TCP Position on the Z-Axis
RX	Real	86.40357							[°]: Rotation around the X-Axis (roll)
RY	Real	-0.1535093							[°]: Rotation around the Y-Axis (pitch)
RZ	Real	90.49969							[°]: Rotation around the Z-Axis (yaw)
Config	"LRob_typeRobotC...								Configuration data of the robot
TurnNumber	Array[0..3] of Byte								Turn number of the axes J1 to J6 and E1
E1	Real	0.0							[°mm]: Position of first external axis
ExtendedPositi...	"LRob_typeRobotEx...								Position of the external axes E2 to E6
Joint	"LRob_typeRobotJo...								Robot joint position data values
Joint	"LRob_typeRobotSh...								Joint position of the axes J1 to J6 and E1
J1	Real	50.91726							[°mm]: Position of first joint of the robot
J2	Real	-62.29543							[°mm]: Position of second joint of the robot
J3	Real	89.50689							[°mm]: Position of third joint of the robot
J4	Real	-21.36081							[°mm]: Position of fourth joint of the robot
J5	Real	-39.60345							[°mm]: Position of fifth joint of the robot
J6	Real	-4.444121							[°mm]: Position of sixth joint of the robot
E1	Real	0.0							[°mm]: Position of first external axis
ExtendedJoint	"LRob_typeRobotEx...								Position of the external axes E2 to E6
CoordinateSystem	"LRob_typeRobotC...								Cartesian coordinate systems data

Figure 2-64: "frontInspection" position copied as start value and saved into the TIA PORTAL project.

At this position, make sure the camera is targeting the required area. Now you can train the new profile for the front-side inspection.

Inspekto Profile training – frontside inspection

For this profile, Inspekto captures the PLC's front view and checks for the presence of the display and labels, as shown below:



Figure 2-65: Front Profile outline and ROIs

For this case, we followed the same steps as for the first profile, except the 1+1 image mode was selected and give then goal of this profile is to detect if the display is mounted Presence ROI instead of Inspection ROI were used.

After uploading the 20 samples, the profile is created and an unique ID is assigned. This ID is later used to switch between profiles from the PLC via the dedicated TIA Portal function blocks. In this case, the Profile ID is 4.

Finally, this is how the points appear on the HMI after being defined:

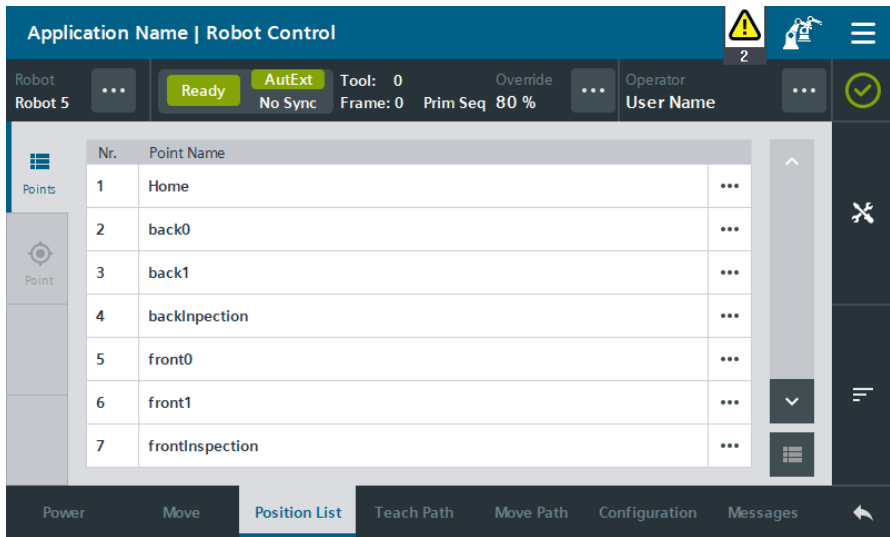


Figure 2-66: SRL HMI – Complete list of points

2.5.2. Robot's Paths

For this application example, we created three paths: "frontFull", "backFull", and "testPath".

2.5.2.1. backFull path

This path moves from Home to the rear inspection point (with a WaitPoint while Inspekto completes the inspection) and then returns to Home.

The points sequence is:

1. Home (PTP)
2. back0 (Linear)
3. back1 (PTP)
4. backInspection (Linear — WaitPoint)
5. back1 (Linear)
6. back0 (PTP)
7. Home (Linear)

To create this path, go to the path list and select an unused one. Then, open the "Points" window to assign points to the selected path:

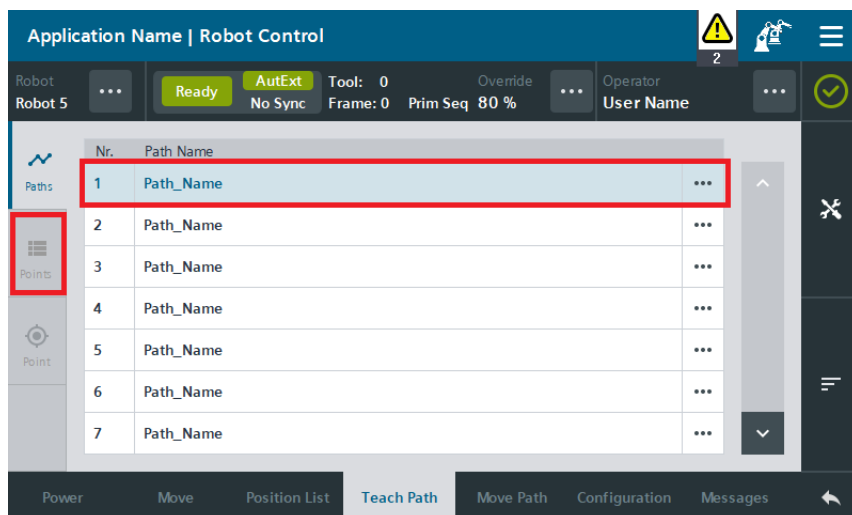


Figure 2-67: SRL HMI – Select Path to assign points

Here is where we select the sequence of points that this path will follow, including additional parameters such as the type of movement the robot should execute when moving to a specific point, movement speed, waiting points, and so on. To add the first point (Home), simply select an unused point and choose *Copy Point from*:

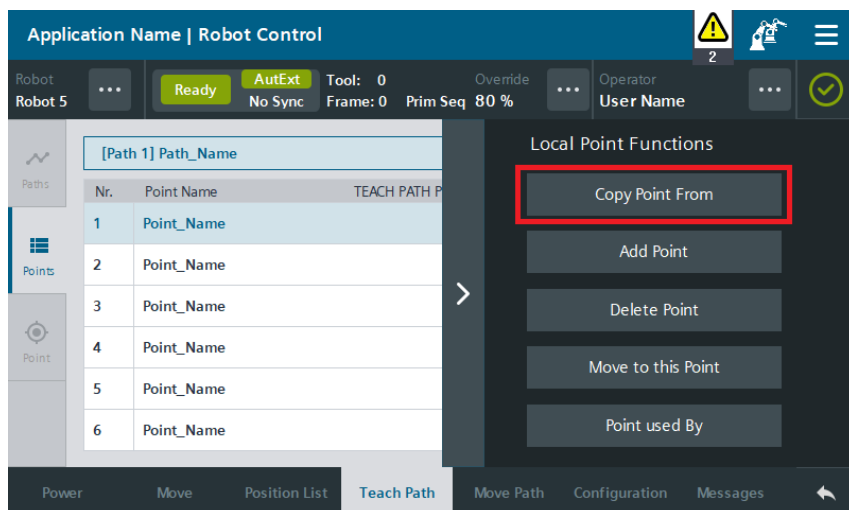


Figure 2-68: SRL HMI – Select point to copy

From the dropdown menu, select the point Home and set the movement type to PTP:

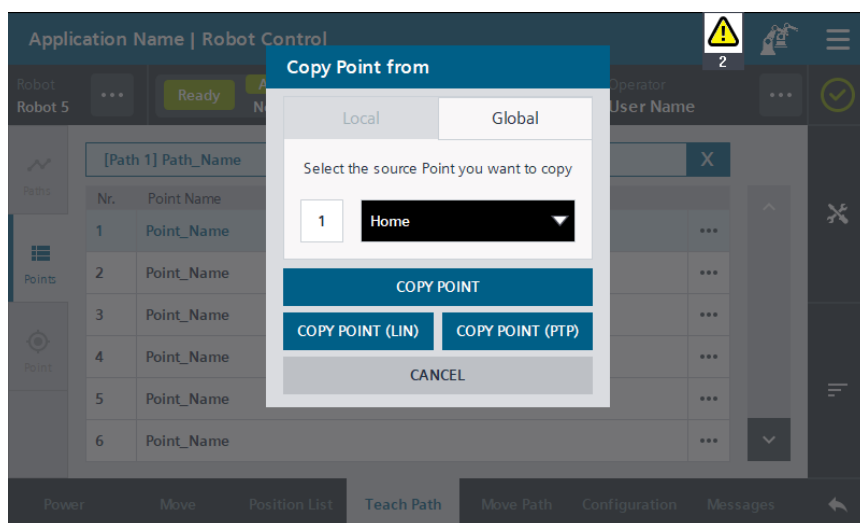


Figure 2-69: SRL HMI – Copy the Home point

The same procedure was applied to the remaining points, with the exception that the point **backInspection** must be declared as a **wait point**, meaning the robot will hold at this position until it is explicitly told to continue. In the PLC logic, the robot resumes the path once Inspekto reports that the inspection is complete.

Additionally, we must assign an **mFunction** to this point, which is useful for programming the logic since it provides a unique identifier for the point.

To do this, after creating the point, select it and go to the Point tab, where you will find more details and additional configuration options. In this case, we configure it as a wait point:

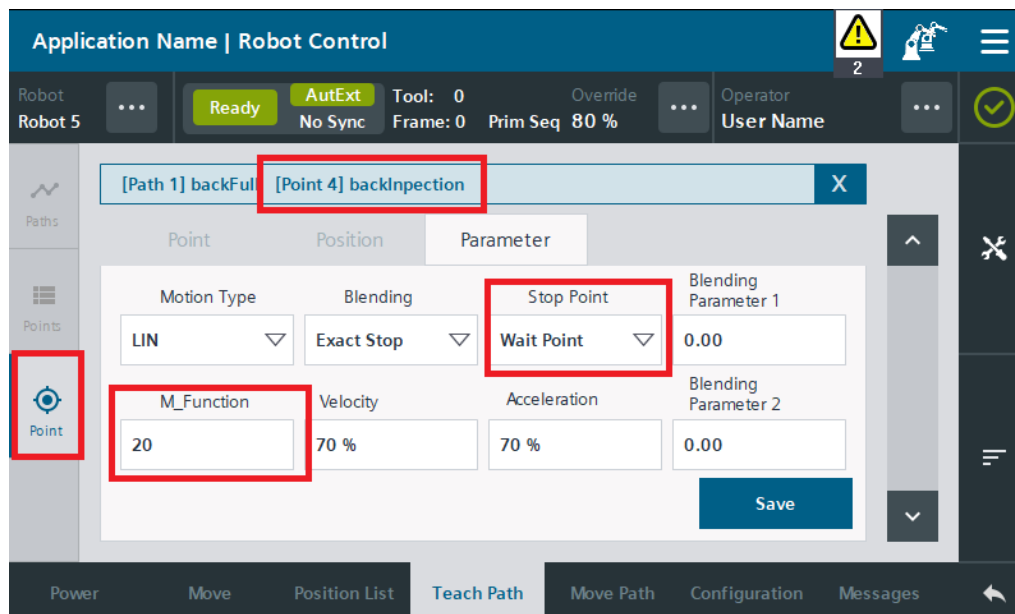


Figure 2-70: SRL HMI – backInspection set as Wait Point.

Now, after finishing the path go to **RobotDataSRL** DB and copy values as start values and save TIA PORTAL project.

In the "**RobotDataSRL**" DB you can find the full path details, including each point's motion type, wait/stop configuration, velocity, acceleration, "**mFunction**" of points, etc.

SiemensInspektoVIS_ReferenceProject_V19_V20 - PLC_1 [CPU 1517V-1 PN] - Program blocks - SIMATIC Robot Integrator - SRL - RobotDataSRL [DB5133]										
RobotDataSRL (snapshot created: 7/12/2025 1:09:29 PM)										
	Name	Data type	Start value	Retain	Accessible f...	Write...	Visible in ...	Setpoint	Supervision	Comment
8	robot[5]	*typeIARobotDa...								
9	config	*typeIARobotDa...								
10	positionsGlobal	Array(1..*AMOUNT...								By the user taught positions
11	positionsOnlineProgramming	Array(1..*AMOUNT...								
12	positionsOnlineProgramming[1]	*typeIARPathstru...								
13	pathName	String[32]	'backFull'							
14	pathSaved	Bool	FALSE							
15	pathOpened	Bool	FALSE							
16	pathCanNotBeSaved	Bool	FALSE							
17	pathEditable	Bool	TRUE							
18	pathOnMove	Bool	FALSE							
19	reserve	Array(1..10) of Bool								
20	point	Array(1..*AMOUNT...								
21	point[1]	*typeIARLineInfor...								
22	point[2]	*typeIARLineInfor...								
23	point[3]	*typeIARLineInfor...								
24	point[4]	*typeIARLineInfor...								
25	pointName	String[16]	'backInspection'							Point Name
26	pointSaved	Bool	TRUE							Point_Defined
27	lastPointOfThePath	Bool	FALSE							This is the last point before a point without "MotionType"
28	collisionDetection	Bool	FALSE							Collision_Detection
29	cont	Bool	FALSE							Continuous Movement
30	externalTCP	Bool	FALSE							True: TCP fixed, False: On the flanch
31	pointWithoutMotionType	Int	2							Point name existing, but without MoveType
32	stopWaitPoint	Int	3							0: No stop or wait point, 1: Open gripper, 2: Close gripper, 3: Wait point
33	motionType	Int	2							0: End, 1: PTP, 2: LIN, 3: CIRC
34	pathVelocity	Int	70							LIN in [m/s], PTP and Axis in [%]
35	acceleration	Int	70							LIN in [m/s^2], PTP and Axis in [%]
36	cornerDistance	Int	100							Blending radius [mm]
37	orientationType	Int	1							0: Standard, 1: Hand PTP, 2: Constant
38	mFunction	Int	20							Type an integer to give out the actual segment for the User
39	reserveInt	Int	0							Reserve
40	position	*LRob_typeRobotPo...								Position
41	BlendingMode	USInt	0							
42	BlendingParameter	Array(0..1) of Real								

Figure 2-71: Detailed view of "backFull" path copied as start value and saved on TIA Portal programmed from SRL HMI

This process was repeated for the rest of the paths.

2.5.2.2. frontFull Path

This path moves from Home to the front inspection point (with a WaitPoint while Inspekto completes the inspection) and then returns to Home. The points sequence is:

1. Home (PTP)
2. front0 (Linear)
3. front1 (PTP)
4. frontInspection (Linear — WaitPoint)
5. front1 (Linear)
6. front0 (PTP)
7. Home (Linear)

In the "RobotDataSRL" DB you can see the full path details, including each point's motion type, wait/stop configuration, velocity, acceleration, mFunction of points, etc.

Name	Data type	Start value	Retain	Accessible f...	Write...	Visible in ...	Setpoint	Supervision	Comment
robot[5]	"type TAILRobotDa...								
config	"type TAILRobotDa...								
positionsGlobal	Array[1..4] of "AMOUNT...								By the user reached positions
positionsOnlineProgramming	Array[1..4] of "AMOUNT...								
positionsOnlineProgramming[1]	"type TAILRobotDa...								
positionsOnlineProgramming[2]	"type TAILRobotDa...								
pathName	String[32]	"frontFull"							
pathSaved	Bool	FALSE							
pathOpened	Bool	FALSE							
pathCanNotBeSaved	Bool	FALSE							
pathEditable	Bool	TRUE							
pathOnLine	Bool								
reserve	Array[1..10] of Bool								
point	Array[1..4] of "AMOUNT...								
point[1]	"type TAILRobotDa...								
point[2]	"type TAILRobotDa...								
point[3]	"type TAILRobotDa...								
point[4]	"type TAILRobotDa...								
pointName	String[16]	"frontInspection"							Point Name
pointSaved	Bool	TRUE							Point_Defined
lastPointOfThePath	Bool	FALSE							This is the last point before a point without "MotionType"
collisionDetection	Bool	FALSE							Collision_Detection
cont	Bool	FALSE							Continuous Movement
externalTCP	Bool	FALSE							True: TCP fixed, False: On the flange
pointWithoutMotionType	Int	2							Point name existing, but without MotionType
stopWaitPoint	Int	3							0: No stop or wait point, 1: Open gripper, 2: Close gripper, 3: Wait point
motionType	Int	2							0: End, 1: PTP, 2: LIN, 3: CRC
pathVelocity	Int	70							LIN in [mm/s], PTP and CRC in [%]
acceleration	Int	70							LIN in [m/s²], PTP and CRC in [%]
cornerDistance	Int	100							Blending radius [mm]
orientationType	Int	1							0: Standard, 1: Hand PTP, 2: Constant
mFunction	Int	25							Type an integer to give out the actual segment for the User
reserveInt	Int	0							Reserve
position	"type TAILRobotDa...								Position
BlendingMode	USint	0							
BlendingParameter	Array[0..1] of Real								

Figure 2-72: Detailed view of "frontFull" path copied as start value and saved on TIA Portal programmed from SRL HMI

In the figure above, the "frontInspection" point is configured as a WaitPoint, meaning the robot will hold at this position until it is explicitly told to continue. In the PLC logic, the robot resumes the path once Inspekto reports that the inspection is complete.

2.5.2.3. multiSide Path

This path performs a multi-side inspection of the object: it moves the robot so Inspekto can execute both the back and front inspections within a single run. The sequence of points is:

1. Home (PTP)
2. back0 (Linear)
3. back1 (PTP)
4. backInspection (Linear — WaitPoint)
5. back1 (Linear)
6. back0 (PTP)
7. front0 (Linear)
8. front1 (PTP)
9. frontInspection (Linear — WaitPoint)
10. front1 (Linear)
11. front0 (PTP)
12. Home (Linear)

In the "RobotDataSRL" DB you can view the full path definition—including each point's motion type, wait/stop configuration, velocity, acceleration, and mFunction—as configured via the SRL HMI.

SiemensInspektoVIS_ReferenceProject_V19_V20 ▶ PLC_1 [CPU 1517V-1 PN] ▶ Program blocks ▶ SIMATIC Robot Integrator - SRL ▶ RobotDataSRL [DB5133]

RobotDataSRL (snapshot created: 7/12/2025 1:09:29 PM)

Name	Data type	Start value	Retain	Accessible f...	Writa...	Visible in ...	Setpoint	Supervision	Comment
positionsOnlineProgramming[7]	*typeTIArLPathstru...		☒	☒	☒	☒	☒		
pathName	String[32]	'multiSide'	☒	☒	☒	☒	☒		
pathSaved	Bool	FALSE	☒	☒	☒	☒	☒		
pathOpened	Bool	FALSE	☒	☒	☒	☒	☒		
pathCanNotBeSaved	Bool	FALSE	☒	☒	☒	☒	☒		
pathEditable	Bool	TRUE	☒	☒	☒	☒	☒		
pathOnMove	Bool	FALSE	☒	☒	☒	☒	☒		
reserve	Array[1..10] of Bool		☒	☒	☒	☒	☒		
point	Array[1..AMOUNT...		☒	☒	☒	☒	☒		
point[1]	*typeTIArLInLinefor...		☒	☒	☒	☒	☒		
point[2]	*typeTIArLInLinefor...		☒	☒	☒	☒	☒		
point[3]	*typeTIArLInLinefor...		☒	☒	☒	☒	☒		
point[4]	*typeTIArLInLinefor...		☒	☒	☒	☒	☒		
pointName	String[16]	'backInspection'	☒	☒	☒	☒	☒		Point Name
pointSaved	Bool	TRUE	☒	☒	☒	☒	☒		Point_Defined
lastPointOfThePath	Bool	FALSE	☒	☒	☒	☒	☒		This is the last point before a point without "..."
collisionDetection	Bool	FALSE	☒	☒	☒	☒	☒		Collision_Detection
cont	Bool	FALSE	☒	☒	☒	☒	☒		Continuous Movement
externalTCP	Bool	FALSE	☒	☒	☒	☒	☒		True: TCP fixed, False: On the flanch
pointWithoutMotionType	Int	2	☒	☒	☒	☒	☒		Point name existing, but without MoveType
stopWaitPoint	Int	3	☒	☒	☒	☒	☒		0: No stop or wait point, 1: Open gripper, 2: ...
motionType	Int	2	☒	☒	☒	☒	☒		0: End, 1:PTP, 2: LIN, 3: CIRC
pathVelocity	Int	70	☒	☒	☒	☒	☒		LIN in [m/s], PTP and Axis in [%]
acceleration	Int	70	☒	☒	☒	☒	☒		LIN in [m/s²], PTP and Axis in [%]
cornerDistance	Int	100	☒	☒	☒	☒	☒		Blending radius [mm]
orientationType	Int	1	☒	☒	☒	☒	☒		0: Standard, 1: Hand PTP, 2: Constant
mFunction	Int	5	☒	☒	☒	☒	☒		Type an integer to give out the actual segme...
reserveInt	Int	0	☒	☒	☒	☒	☒		Reserve
position	*LRob_typeRobotPo...		☒	☒	☒	☒	☒		Position
BlendingMode	USInt	0	☒	☒	☒	☒	☒		
BlendingParameter	Array[0..1] of Real		☒	☒	☒	☒	☒		
point[5]	*typeTIArLInLinefor...		☒	☒	☒	☒	☒		
point[6]	*typeTIArLInLinefor...		☒	☒	☒	☒	☒		
point[7]	*typeTIArLInLinefor...		☒	☒	☒	☒	☒		
point[8]	*typeTIArLInLinefor...		☒	☒	☒	☒	☒		
point[9]	*typeTIArLInLinefor...		☒	☒	☒	☒	☒		
pointName	String[16]	'frontInspection'	☒	☒	☒	☒	☒		Point Name
pointSaved	Bool	TRUE	☒	☒	☒	☒	☒		Point_Defined
lastPointOfThePath	Bool	FALSE	☒	☒	☒	☒	☒		This is the last point before a point without "..."
collisionDetection	Bool	FALSE	☒	☒	☒	☒	☒		Collision_Detection
cont	Bool	FALSE	☒	☒	☒	☒	☒		Continuous Movement
externalTCP	Bool	FALSE	☒	☒	☒	☒	☒		True: TCP fixed, False: On the flanch
pointWithoutMotionType	Int	2	☒	☒	☒	☒	☒		Point name existing, but without MoveType
stopWaitPoint	Int	3	☒	☒	☒	☒	☒		0: No stop or wait point, 1: Open gripper, 2: ...
motionType	Int	2	☒	☒	☒	☒	☒		0: End, 1:PTP, 2: LIN, 3: CIRC
pathVelocity	Int	70	☒	☒	☒	☒	☒		LIN in [m/s], PTP and Axis in [%]
acceleration	Int	70	☒	☒	☒	☒	☒		LIN in [m/s²], PTP and Axis in [%]

Figure 2-73: Detailed view of "multiSide" path copied as start value and saved on TIA Portal programmed from SRL HMI

In the figure above, both *backInspection* and *frontInspection* are configured as WaitPoints, meaning the robot remains at those poses until explicitly commanded to continue. In PLC logic, the robot resumes the path once Inspekto reports the inspection is complete.

By now, the list of paths looked like this:

Application Name | Robot Control

Robot 5 ... Ready AutExt No Sync Tool: 0 Frame: 0 Prim Seq 80 % Operator User Name

Nr.	Path Name	
1	backFull	...
2	frontFull	...
3	multiSide	...
4	Path_Name	...
5	Path_Name	...
6	Path_Name	...
7	Path_Name	...

Power Move Position List Teach Path Move Path Configuration Messages

Figure 2-74: SRL HMI – complete path list

Finally, up to this point we have been copying the new points/paths as *Start Values* in the RobotDataSRL and saving them in TIA Portal. Now, download the program to the PLC so that the new points/paths are stored in the PLC.

2.6. Controlling Inspekto via PLC

2.6.1. Working sequence

After configuring the profiles in the Inspekto software according to the areas and sides of the object to be inspected, you can start programming the TIA Portal function blocks that will control the Inspekto system according to your automation process and inspection scenario. Given the use case presented in this application example, the PLC needs to trigger the following actions on the Inspekto system:

1. Load the inspection mode that allows the image capture and inspection to be triggered via the PLC.
2. Load the corresponding inspection profile for the back side of the S7-1500 PLC.
3. Trigger the inspection of the backside of the S7-1500 PLC once the robot is in the corresponding position.
4. Load the corresponding inspection profile for the front side of the S7-1500 PLC.
5. Trigger the inspection of the frontside of the S7-1500 PLC once the robot is in the correct position.

2.6.2. Operating modes

For demo purposes, the use case presented in this application example contains 2 operating modes:

- **Auto:** In this mode the system performs a complete back-front inspection of the object. Additionally in this mode, the user can enable the "Continuous" mode in which case the system will perform consecutive back-front inspections until the user stops the loop using the "Stop" button.
- **Manual:** In this mode, the user can select which side, back or front, of the object must be inspected and the system will perform an one-time inspection. The operation can be repeated by pressing any of the 2 dedicated buttons again.

NOTE

The previously described operation modes are provided solely as examples to illustrate a potential working scenario of the system. They do not represent actual trigger inspection modes of the Inspekto system.

2.6.3. Programming Inspekto function blocks

This section details how to program the dedicated TIA Portal function blocks to control Inspekto and to trigger the correct actions based on the working sequence described in the previous section and the use case presented in this application example.

NOTE

Each function block should be programmed based on your specific use case, automation process and inspection scenario. The example presented serves only as a reference to help users better understand the steps required to integrate Inspekto with an industrial robot arm for inspecting a manufactured object from multiple angles.

Before programming the Inspekto function blocks it is highly recommended to review the official documentation and understand how the function blocks work and avoid undesirable behaviors.



Working with the SIEMENS Inspekto VIS function blocks

Detailed description of Inspekto TIA Portal function blocks

[Link](#)

When working with the Inspekto TIA Portal function blocks, please consider:

- Before sending any command, the PLC needs to verify that "IN_SYSTEM_READY" signal is =1.
- After sending any command, "IN_SYSTEM_READY" will change to "0" while the command is being performed and change back to "1" when the operation was completed.
- "IN_SYSTEM_ERROR" refers to the status of the last command received, and it is not a system state. If "IN_SYSTEM_ERROR" =1, when "IN_SYSTEM_READY" =1 the system is ready to receive the next command.

- It is recommended to verify any of the possible relevant variables of the Inspekto system before sending any command.

Set inspection trigger mode – FBUpdateInspectionMode

Inspekto allows different inspection triggers modes

To be able to control the trigger of the inspection via the PLC, the inspection trigger mode of the Inspekto system must be configured to one of the available "Direct Trigger" options.

Inspection Trigger Mode	Value
OFF	0
Auto - Part ID provided by the PLC	1
Auto - Part ID generated by the system	2
Manual	3
External - Part ID generated by the system	4
External - Part ID provided by the PLC	5
Direct Trigger - Part ID generated by the system	6
Direct Trigger - Part ID provided by the PLC	7
Auto - Part ID generated by the external code reader	8

Table 2-2: Inspekto trigger modes values for PLC integration

NOTE

Each inspection mode described in detail in the official [Inspekto documentation](#)

The inspection mode is selected via the PLC using the function block, "FBUpdateInspectionMode".

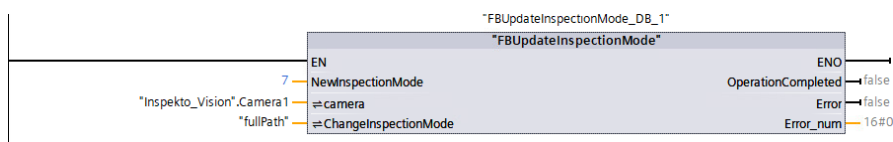


Figure 2-75: TIA Portal function block for setting the inspection mode from the PLC

To set the inspection mode:

- Navigate to the call of the "FBUpdateInspectionMode" function block. In the provided reference project, the FB "FBUpdateInspectionMode" is called in the FB "InspektoRobot".
- Select your desired Direct Trigger mode based on your use case and assign to the "NewInspectionMode" input parameter.
- Assign the "Inspekto_Vision" DB to the "camera" input parameter
- Define the trigger logic for selecting the inspection mode and assign it to the "ChangeInspectionMode" input parameter. A raising edge in this input parameter will set the selected inspection mode in the Inspekto system.

The status of the executed command can be tracked using the output parameters "OperationCompleted", "Error" and "Error_num". You can check if the inspection mode was changed accordingly also in the Inspekto UI.

If you want to change the inspection mode again:

- Change the value assigned to the input parameter "NewInspectionMode".
- Generate a new rising edge in the "ChangeInspectionMode".

NOTE

If you want to assign a specific identifier to the image captured by the system, ensure you are using the inspection mode "Direct trigger – Part ID provided by the PLC"

NOTE

The programming of the "FBUpdateInspectionMode" within the reference project should be adapted according to your automation process and inspection scenario.

Load inspection profile - FBUpdateProfileID

To trigger the inspection from the PLC, a profile with the corresponding view and areas of the object you want to inspect must be also loaded beforehand to the Inspekto system.

Each profile has an ID associated within the systems which can be checked in the Profile Settings menu in the Inspekto software. This ID is important to ensure that the right profile is loaded into the system.

Profile settings

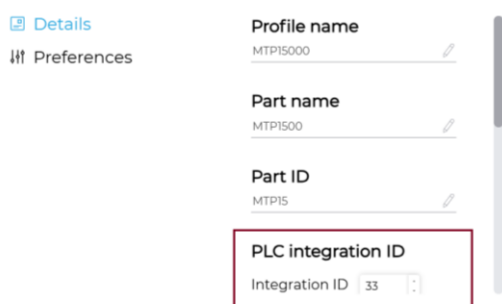


Figure 2-76: Profile PLC integration ID in Inspekto UI

To load a profile via the PLC the function block FBUpdateProfileID proceed as follows:

1. Assign the tag that contains the ID of your profile to the "NewProfileID" input parameter. Using tags instead of static values you can change the profile during the operation of the system according to your automation process and inspection scenario.
2. Assign the "Inspekto_Vision"."Camera" struct to the "camera" input parameter
3. Define the trigger logic for loading the desired profile and assign it to the "LoadProfile" input parameter. A raising edge in this input parameter will load the selected profile into the Inspekto system.

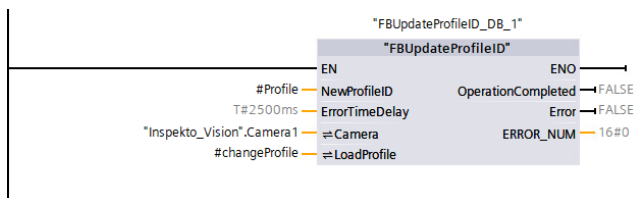


Figure 2-77: TIA Portal function block for loading a profile from the PLC

The status of the executed command can be tracked using the output parameters "OperationCompleted", "Error" and "Error_num". You can check if the selected profile was loaded successfully by accessing the Inspekto software,

If you want to change the profile again:

1. Ensure the value assigned to the input parameter "NewProfileID" changed to a value that corresponds to an existing profile.
2. Generate a new rising edge in the "LoadProfile" input parameter to trigger the profile loading.

NOTE

Based on the characteristics of your profiles, changing a profile might take up to 3 seconds. Please consider this aspect when defining the logic of the profile changing trigger to ensure it meets your automation process and use case.

NOTE

The programming of the "FBUpdateProfileID" within the reference project must be adapted according to your automation process and inspection scenario.

Trigger inspection – FBNewObjectTrigger

After setting the inspection mode and loading the correct inspection profile, you are ready to trigger inspections via the PLC. This action is done using the "FBNewObjectTrigger" function block.

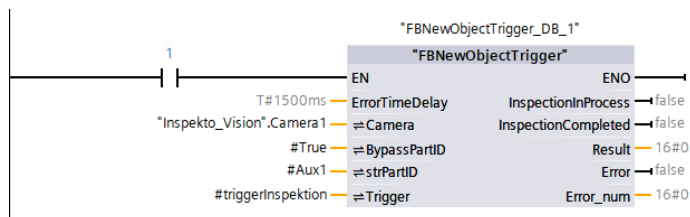


Figure 2-78: TIA Portal function block for triggering a new inspection from the PLC

1. Assign the "Inspekto_Vision" DB to the "Camera" input parameter
2. If you are using the inspection mode "Direct trigger – Part ID provided by the PLC", set the input parameter "BypassPartID" to False otherwise set it to True.
3. If you set the input parameter "BypassPartID" to False, assign to "strPartID" the value you want to provide to the captured image by Inspekto. To trigger the inspection successfully you must ensure this value changes before any new inspection.
4. Define the trigger logic for triggering the inspection according to your automation process and inspection use case. A raising edge in this input parameter will load the selected profile into the Inspekto system. In the application example presented in this document the inspection is triggered when the robot reaches the two defined waiting points.

NOTE

To trigger a new inspection the following conditions must be met:

- The input parameter "Trigger" receives a new rising edge.
- The value of the input parameter "strPartID" has changed from the last inspection if "BypassPartID" is set to False.

NOTE

The programming of the "FBNewObjectTrigger" within the reference project must be adapted according to your automation process and inspection scenario.

The status of the executed command as well as the inspection result can be tracked using the corresponding output parameters of the function block. You can check if the selected profile was loaded successfully by accessing the Inspekto software,

If you want to trigger a new inspection:

1. If the parameter "BypassPartID" is set to False, ensure the value assigned to the input parameter "strPartID" changed.
2. Generate a new rising edge in the "Trigger" input parameter to trigger the new inspection.

2.7. HMI programming

Lastly, for HMI engineering we incorporated a screen called Demo, which is where the application is executed:

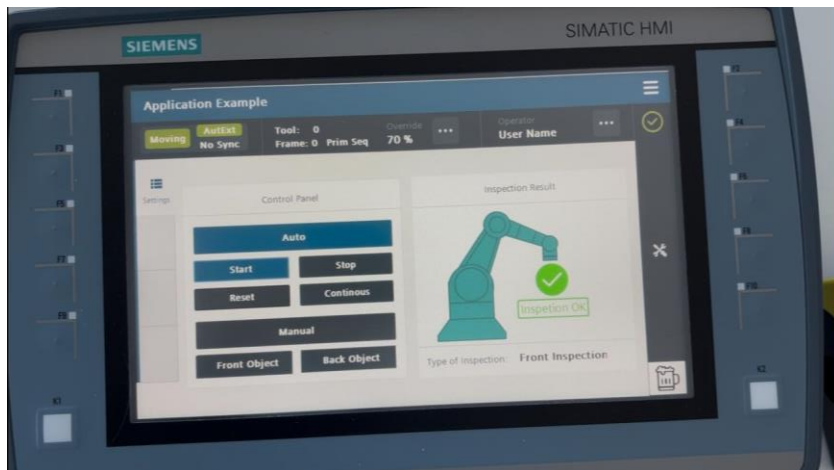


Figure 2-79: HMI screen for application control

The HMI design was created by reusing components/elements from the SRL/SRI HMI, which is compliant with the Siemens Styleguide.

NOTE

Find more information about the Siemens Styleguide [here](#).

You could also find useful the HMI Template Suite [here](#).

Through the HMI, we can run the different operation modes described [here](#).

For example, when pressing the Manual operation mode and selecting Back Object, the PLC will command the robot to move using the backFull path (Path #1) and simultaneously load the backPLC profile to Inspekto (Profile #8). The robot will then move until it reaches the waiting point. At this stage, once the PLC receives the waiting point command, it triggers Inspekto. After the inspection is completed, the path continues as shown in the picture below:

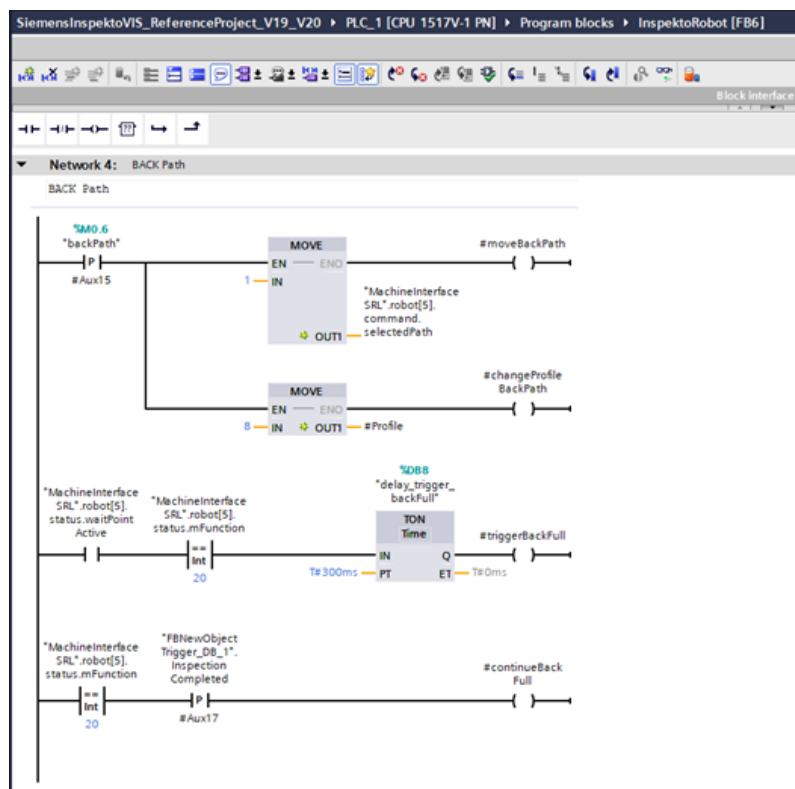


Figure 2-80: PLC/HMI Logic to control Robot&Inspekto

3. Additional information

3.1. Inspekto functions

The SIEMENS Inspekto VIS system supports the following functions:

Function	Data Type	Direction	Explanation (see below for detailed explanations)
OUT_INSPECTION_MODE	Byte	PLC -> INSPEKTO	Sets the system's triggering mode.
OUT_NEW_OBJECT_TRIGGER	Array64Chars	PLC -> INSPEKTO	Sets a new inspection trigger
OUT_PROFILE_ID	Word	PLC -> INSPEKTO	Load inspection profile with profile connectivity ID XX
OUT_BATCH_ID	Array64Chars	PLC -> INSPEKTO	Start a new batch with batch ID XX
IN_CUR_INSPECTION_MODE	Byte	INSPEKTO -> PLC	Report the current inspection mode (trigger mode)
IN_INSPECTION_MODE_RESPONSE	Byte	INSPEKTO -> PLC	a bit to acknowledge that the OUT_INSPECTION_MODE command has been performed
IN_CUR_OBJECT_ID	Array64Chars	INSPEKTO -> PLC	Report the current object ID
IN_NEW_OBJECT_TRIGGER_RESPONSE	Byte	INSPEKTO -> PLC	a bit to acknowledge that the OUT_NEW_OBJECT_TRIGGER command has been performed
IN_CUR_PROFILE_ID	Word	INSPEKTO -> PLC	Report the current inspection profile ID
IN_PROFILE_ID_RESPONSE	Byte	INSPEKTO -> PLC	a bit to acknowledge that the OUT_PROFILE_ID command has been performed
IN_CUR_BATCH_ID	Array64Chars	INSPEKTO -> PLC	Report the current Batch ID
IN_BATCH_ID_RESPONSE	Byte	INSPEKTO -> PLC	a bit to acknowledge that the OUT_BATCH_ID command has been performed
IN_KEEP_ALIVE	Byte	INSPEKTO -> PLC	a bit that periodically being updated between values of 0 and 1 to indicate the system is online. The flipping period can be configured.
IN_OBJECT_INSPECTION_RESULT	Byte	INSPEKTO -> PLC	Report the last inspection cycle result
IN_SYSTEM_ERROR	Byte	INSPEKTO -> PLC	report an error in case there is one, values may differ given the error state and the command that was sent to the INSPEKTO S70
IN_SYSTEM_READY	Byte	INSPEKTO -> PLC	report whether the system is ready to receive a new command or not
IN_ROIS_DEFECT_LIST	Array64Chars	INSPEKTO -> PLC	reports defects per ROI, up to 64 ROIs

Figure 3-1: Inspekto available functions

Detailed information about the Inspekto functions is available in the official documentation in the following link:

<https://docs.industrial-operations-x.siemens.cloud/r/en-us/6.0/profinet-integration-function-blocks/detailed-functions-explanations/plc-siemens-inspekto-vis>.

3.2. Inspekto system installation

Detailed information about the steps needed to install the hardware components for the Inspekto system are available under the following link:

[SIEMENS Inspketo System Installation](#)

4. Appendix

4.1. Service and support

SiePortal

The integrated platform for product selection, purchasing and support - and connection of Industry Mall and Online support. The SiePortal home page replaces the previous home pages of the Industry Mall and the Online Support Portal (SIOS) and combines them.

- **Products & Services**
In Products & Services, you can find all our offerings as previously available in Mall Catalog.
- **Support**
In Support, you can find all information helpful for resolving technical issues with our products.
- **mySiePortal**
mySiePortal collects all your personal data and processes, from your account to current orders, service requests and more. You can only see the full range of functions here after you have logged in.

You can access SiePortal via this address: sieportal.siemens.com

Technical Support

The Technical Support of Siemens Industry provides you fast and competent support regarding all technical queries with numerous tailor-made offers – ranging from basic support to individual support contracts.

Please send queries to Technical Support via Web form: support.industry.siemens.com/cs/my/src

SITRAIN – Digital Industry Academy

We support you with our globally available training courses for industry with practical experience, innovative learning methods and a concept that's tailored to the customer's specific needs.

For more information on our offered trainings and courses, as well as their locations and dates, refer to our web page: siemens.com/sitrain

Industry Online Support app

You will receive optimum support wherever you are with the "Industry Online Support" app. The app is available for iOS and Android:



4.2. Application support

Siemens AG
Digital Factory Division
Factory Automation
SIMATIC System Support
DI FA S SUP
Gleiwitzer Str. 555
90475 Nürnberg, Germany

mailto: inspektosupport.industry@siemens.com

4.3. Links and literature

No.	Topic
-----	-------

\1\	SiePortal https://sieportal.siemens.com
-----	--

\2\	Link to this entry page of this application example https://support.industry.siemens.com/cs/ww/en/view/109995085
-----	--

4.4. Documentation change

Version	Date	Modification
V1.0	09/2025	First version