POLITECNICO DI TORINO

POLITECNICO DI TORINO

Master Degree course in Mechatronic Engineering

Master Degree Thesis

Industrial Process Simulation through Digital Twin Modeling and PLC Programming

Supervisors
Prof. Luigi Mazza
Prof. Enrique Jorge Bernabeu Soler

Candidate
Dennis Barale

Academic Year 2024-2025

Acknowledgements

As I reach the conclusion of this significant chapter in my academic journey, I find myself filled with gratitude and joy. I owe this moment to those who have stood by me from the very beginning, as well as to the people I have been fortunate to meet along the way.

To my supervisors, I extend my deepest thanks for your inspiring guidance, patience, and faith in my work. Your expertise and mentorship have been the driving forces behind this thesis, and I am sincerely grateful for the opportunity to learn and grow under your supervision.

To my family, words are insufficient to convey the depth of my gratitude. Your love and support have been my foundation throughout this journey. Thank you for believing in me even in moments when I faltered, and for your constant encouragement. This achievement belongs as much to you as it does to me.

To my friends, thank you for walking beside me, for the late-night conversations, the laughter, and the moments of calm when I needed them most. Your companionship has enriched this experience and given it greater meaning.

To Valencia, *la terra de les flors, de la llum i de l'amor*, without which this project might never have even been conceived. This magical city has welcomed me, taught me resilience during one of its most difficult times, and blessed me with new friendships and even more, much more than I could have ever imagined.

To all those who offered their loving understanding when I spent time to work instead of being with them, I am profoundly grateful.

This thesis is not merely the product of academic effort, it is also a reflection of the care, kindness, and support I have received from those around me. I am truly and deeply thankful to each of you.

Abstract

This thesis aligns with the principles of Industry 4.0, aiming to explore the potential of Digital Twin technology in the simulation of industrial processes. It leverages personal expertise in PLC programming to automate operations within a virtual environment.

The study begins with a comprehensive analysis of an industrial process, including the identification of sensors and actuators embedded in its machinery.

Based on the data collected during the analysis phase, a Digital Twin model is developed to replicate the observed industrial process. The model is designed with a level of detail sufficient to accurately mirror the mechanical behavior and operational dynamics of the real machines, within the constraints of the simulation environment. This modeling process includes the design of custom components using CAD software, which are subsequently imported into the simulation platform. Once integrated, these components are assembled to form complete machine systems. Controllers are then programmed to animate the machines, ensuring that their virtual behavior aligns with the intended physical operations.

This virtual representation is programmed using PLC languages, and a connection is established between the Digital Twin software and the PLC software via a SoftPLC, enabling real-time interaction and control.

The simulation provides a visual demonstration of the process operation and the logic implemented in the PLC, showcasing how automation can be effectively modeled and tested in a virtual setting.

Contents

Chapter 1

Introduction

1.1 Context and Motivation

The modern industrial environment is undergoing substantial transformation, a phenomenon widely known as Industry 4.0. This term represents the integration of advanced digital technologies into manufacturing and processing operations. [8] [12] [11]

Among the 17 Sustainable Development Goals, established by the United Nations as part of the 2030 Agenda for Sustainable Development, the ninth goal (SDG 9) focuses on building resilient infrastructure, promoting inclusive and sustainable industrialization, and encouraging innovation. [6] In this context, industry is explicitly recognized as a key component of this goal. SDG 9 emphasizes the role of industrial development as a foundation for economic growth. It highlights the importance of resilient infrastructure, innovation, and support for small and medium-sized enterprises. Industry is seen as a key driver for job creation, poverty reduction, and social inclusion, with strong links to education, gender equality, and environmental sustainability. [6] This thesis also aligns with the principles of Sustainable Development Goal 8 (SDG 8), which advocates for sustained, inclusive, and sustainable economic growth, full and productive employment, and decent work for all. [6] SDG 8 emphasizes that economic progress must serve as a positive force globally, ensuring that financial development contributes to the creation of dignified and fulfilling employment opportunities without compromising environmental integrity. [6] This goal calls for the promotion of job creation through a broader access to financial services, thereby enabling greater participation in entrepreneurship and innovation. Among its key targets are: enhancing economic productivity through diversification, technological advancement, and innovation, particularly in high value-added and labor-intensive sectors; supporting development-oriented policies that foster productive activities, decent employment, creativity, and the formalization and growth of micro-, small-, and medium-sized enterprises; and improving global resource efficiency in consumption and production. The latter includes efforts to decouple economic growth from environmental degradation, in line with the 10-Year Framework of Programmes on Sustainable Consumption and Production, with developed nations taking the lead. The work presented in this thesis contributes to these objectives by promoting technological innovation in industrial automation, fostering knowledge development, and supporting

the transition toward more efficient and sustainable production systems. Through the use of Digital Twin technology, the project encourages safer, more informed, and more inclusive approaches to industrial process design and control, ultimately contributing to the broader goals of economic sustainability and social responsibility. [6]

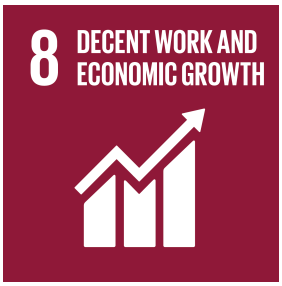


Figure 1.1: SDG 8.

Figure 1.2: SDG 9.

Industry 4.0 constitutes a paradigm shift that is driven by several key innovations that enable unprecedented levels of connectivity, data exchange, and intelligent automation. [9] [11] Central among these are the Industrial Internet of Things (IIoT), Big Data analytics, and Cloud Computing, all of which provide foundational elements for the realization of Cyber-Physical Systems (CPS). [8] [11] [9]

The proliferation of IIoT devices, characterized by embedded sensors and actuators, facilitates real-time connectivity, enabling continuous exchange of data across disparate domains. [8] [11] This extensive data collection underpins the capabilities of Big Data processing techniques, which, coupled with advancements in Artificial Intelligence (AI) and Machine Learning (ML), allow for the extraction of significant knowledge, the identification of hidden patterns, and the enhancement of physical system performance. [8] Cloud Computing further increases these capabilities by providing scalable infrastructures for storing, elaborating, and centralizing vast amounts of data, which can then be visualized and accessed remotely. [8] These technological integrations create a dynamic ecosystem in which systems engage in advanced machine-to-machine communication, enabling data to flow seamlessly between components. [8] At the same time, enhanced human-machine interfaces (HMIs) and visualization platforms facilitate intuitive machine-to-human interaction, allowing experts to monitor, operate, and derive insights from complex systems. [8] [11]

Together, Industry 4.0 technologies transform industrial process management, leading to substantial improvements across several areas. [7] [9] [11] As far as machine and plant design is concerned, integrating product, process and plant (PPP) information is essential to achieve efficient planning and reduce production lead times. Factory layout, a critical aspect of design, requires real-time adaptation to changes originating from product design, process updates, and shop floor modifications. Advancements in layout planning emphasize 3D visualization for optimizing material and resource flow and identifying potential congestion, moving beyond traditional 2D drawings. [7] The ability to simulate, test and compare different possible configurations in a virtual environment enables the optimization and validation of design choices, reducing risks and costs associated with

physical prototypes. [8] [11]

Furthermore, these technologies revolutionize maintenance strategies, transforming them from reactive to highly proactive and intelligent approaches, including preventive, condition-based, predictive, and prescriptive maintenance. [12] [9] Digital solutions enable continuous monitoring of equipment, forecasting of potential issues, and autonomous implementation of "self-healing mechanisms" [8], significantly reducing downtime and improving productivity. [8] [12] [9] Real-time data elaboration enables the estimation of performance indicators, helping detecting possible bottlenecks and suggesting alternatives to avoid them. [7] [11] The shift towards smart manufacturing, underpinned by these innovations, provides a robust foundation for enhanced system control, performance diagnostics, and overall operational efficiency. [7] [8] [11]

In this transformative landscape, the Digital Twin (DT) stands out as a central technology, providing a powerful means of connecting physical and virtual environments. [11] However, even among experts, defining the concept in a universally exhaustive way remains a challenge, as it is inherently complex and open to multiple interpretations. Nevertheless, a clear and consistent understanding is crucial for its effective use in industrial contexts. At its foundation, a Digital Twin represents a virtual counterpart of a physical entity-whether an object, a process, or a system-existing in a digital environment and continuously connected to its real-world version through the exchange of data and information. Far from being just a static model or simulation, it functions as a dynamic, intelligent system that evolves in real time, reflecting the ongoing state and behavior of its physical twin. [7] [8] [9] [11] The Digital Twin is characterized by its ability to:

- "...establish a bijective relationship" [8] with its twin.
- Leverage Artificial Intelligence (AI) to process high-dimensional data, reveal underlying patterns, interpret system behavior, and forecast future issues. [8]
- Employ proactive approaches in maintenance, recommending actions and activating self-healing mechanisms. [8]
- Possess "self-adaptation and self-parametrization capabilities, which allow to resemble the physical twin during its whole life cycle." [8]
- "Provide modeling and simulation applications for representing, in a realistic and natural way, both the current status of the physical twin, and different 'what-if" scenarios." [8]

One of the main intentions behind this work is the need to bridge the gap between the theoretical concepts of Industry 4.0 and their practical application in real industrial environments. Although the project still has an educational orientation, its objective is to address many of the real-world challenges commonly faced in industrial automation. To support this goal, a Digital Twin of an industrial process is created. This model allow users to visualize and interact with the system's behavior. However, the developed Digital Twin does not possess any direct connection with the actual industrial process, namely its Physical Twin. This is because establishing real-time data exchange with the production process would require explicit authorization from a company, which is generally

improbable to obtain. As a result, synchronization between the virtual and the physical system is not addressed within the scope of this thesis. Furthermore, the availability of free software capable of building Digital Twins in a mechatronic context is limited. After thorough research, an accessible software solution was selected such that it could offers enough flexibility to model and control a 3D representation of the plant. Additionally, the solution could enable connectivity with PLC programming software via a SoftPLC, thereby supporting the primary aim of this work: constructing a virtual model that can be programmed using PLC software to emulate the control logic of an actual production line. Through this approach, the thesis demonstrates the potential and scalability of Digital Twins as tools for industrial development and technical training, while being also effective and valuable for educational purposes. At last, this work contributes to the author's expanding understanding of the application of Digital Twins within mechatronic systems.

1.2 State of the Art

When working with industrial processes, having a solid understanding of the products being processed is essential to fully comprehend the operating principles of the machinery involved. For this reason, and considering the author's personal experience, a fruit packing line has been chosen as the reference model. Specifically, the process focuses on the packing of apples. Apples rank among the top ten agricultural products in Italy by volume, according to FAO statistics [10]. The steady increase in production has led companies to adopt higher levels of automation in apple processing. Nevertheless, handling apples remains a complex task due to their perishable nature and their sensitivity to bruising. This is one of the main reasons why apples, unlike stone fruits, are typically processed in water.

After harvesting, the apples are stored in cold rooms inside boxes. These boxes are then sent to a presizing and grading facility, where an electronic sorting system analyzes each fruit based on shape, color, ripeness, and visible defects. The apples are then redistributed into new boxes according to these characteristics. Once presizing is complete, the apples move on to the next stage of processing. At this point, apples may follow different paths: they can be packed into trays of various sizes, placed into bags, or undergo a waxing treatment. Waxing helps reduce moisture loss, protects against microbial contamination, and improves visual appeal.

The simulation developed in this project focuses on the path in which apples are weighed, packed in bags, placed in boxes, and finally palletized for shipment. First of all, boxes of a certain size and quality are chosen to fill the bags according to the net weight required by the customer. Then, the process begins with an in-feed system that submerses the boxes in water and gently tips them to release the apples. A conveyor then collects the apples and transports them out of the water. The next stage is drying, where fans and a specific type of rollers are used to remove the surface moisture. The apples then continue along the conveyor to the weighing station. Here, conveyors and guiding mechanisms arrange the apples in lines and direct them to hoppers, which weigh them in batches. Based on the target weight for each bag, a specific combination of hoppers

opens, releasing apples onto a flap conveyor. This ensures that each group of apples matches the net weight required for a single bag. The flap conveyor guides the apples into bags which are dispensed step by step. Once filled, the bags are sealed and ready for packing. Although this packing step is often performed manually, it can be redesigned and automated, as proposed in this project. When the bags are placed in boxes, the final stage is palletizing, after which the products are ready for shipment.

1.3 Software Implementation

Once the process under investigation had been selected, the next step involved identifying suitable software capable of supporting the modeling of Digital Twins and enabling communication with a PLC programming environment. The search for a modeling platform began with Factory I/O [2], which is an excellent tool for didactic purposes. However, it offers limited flexibility in terms of customization and extensibility. Given the specific nature of the selected process and the custom machinery involved, it was deemed more appropriate to adopt a similar platform with enhanced customization capabilities. *Machine Simulator 4 by Nirtec* [3] was ultimately chosen for this task. Its compatibility with standard CAD workflows, such as designing components in 3D using *SolidWorks* [5], exporting them in `.stl` format, converting them to `.obj`, and subsequently importing them into the simulator, makes it an ideal candidate for building User Defined Components (UDCs).

UDCs are created by incorporating `.obj` files either as standalone components or by appending them to existing objects. These components can then be dynamically controlled using the *Controllers Editor*, which allows defining relative and absolute movements such as rotations and translations along different axes at variable speeds. Furthermore, smoother and more realistic transitions can be achieved using the *Key Frame Animator* feature.

Beyond its flexibility in component design, the most significant advantage of *Machine Simulator 4* lies in its compatibility with various PLC programming environments. Based on prior experience with *CODESYS* [1], it was selected as the PLC development platform for this project. Additionally, the availability of a full license enabled the use of the *CODESYS Gateway* and *CODESYS SoftPLC*, allowing the creation of a virtual PLC directly on the same laptop running the simulation. This setup made it possible to employ an OPC UA protocol [4] within *Machine Simulator 4* to establish communication with *CODESYS*, thereby facilitating the linkage of inputs and outputs between the Digital Twin and the virtual PLC.

1.4 Analysis and Process Layout

First, it is necessary to define an appropriate level of geometric detail to be maintained throughout the development of the virtual model. Given the complexity of the machinery involved and the fact that the Digital Twin is primarily intended for programming applications, the modeling effort focuses on producing a functional representation of the industrial process, with particular emphasis on the positioning and operational behavior

of sensors and actuators. Accordingly, the working principles of mechanisms and machines are modeled with high fidelity. Nevertheless, due to limitations related to software capabilities and simulation performance, certain simplifications have been introduced; these will be discussed in subsequent chapters. It is also important to note that the dimensions and shapes of the machines are not reproduced with engineering accuracy, as this would require detailed technical drawings for each component. Instead, the geometric representation serves to illustrate the operating principles and dynamic behavior of the system.

Moving on to the detailed analysis, attention is first placed on the robotic system responsible for tipping the plastic boxes. This includes the clamping and unclamping mechanisms, the box-tilting system that facilitates the unloading of apples, and the pantograph structure used to stabilize the clamping elements. The positions of motors, pneumatic cylinders, and sensors are examined, and by observing the machine in operation, valuable insights into the programming logic are obtained.

Subsequently, focus shifted to the roller conveyor responsible for extracting apples from the water and initiating the drying process. Although this stage may appear simple at first glance, it proves difficult to model accurately. In the actual drying machine, the rollers are covered with brushes and sponges that assist in removing residual water from the surface of the apples, working in conjunction with forced air provided by fans. However, replicating this roller configuration is not feasible within the limitations of the modeling software, and simplifications have to be adopted.

Following this, one of the most critical and technically demanding stages is addressed: the weighing process. Understanding the mechanisms behind this phase presents significant challenges due to its complexity and the precision required in both mechanical and control aspects. Initially, apples are transferred onto a conveyor belt that carries them toward the weighing system, which consists of multiple channels. Each channel includes a specialized conveyor belt known as a *singulator*, which arranges the apples into a single file line and directs them toward the first flap. This flap remains open while the corresponding hopper is empty. Once the hopper reaches capacity, the flap closes and the system records the weight of the individual hopper. After all hoppers have registered their respective weights, a decision algorithm identifies the combination of hopper flaps whose total weight more closely matches the target bag weight. These selected hoppers are then discharged simultaneously. The apples fall onto a flap conveyor belt, guided by a larger flap, which directs them to the packing machine. This stage involves several sensors and actuators, the most sensitive of which are the load cells within the hoppers. These components are particularly complex to deal with in physical machines and require careful calibration. A specific approach is adopted to replicate them, as described in Section 2.1.3.

In the subsequent phase, empty bags are dispensed from a continuous roll, incrementally unrolled by rubber belts and monitored by a color sensor that detects a black mark indicating the start of each bag. Once filled, the bags are conveyed to the end of the bag-filling unit, where a heated sealing blade hermetically seals and separates them. This machine also contains numerous sensors and actuators within a compact space, making it challenging to identify and program their behavior, particularly in synchronizing the

bag feeding, filling and sealing processes.

Finally, the sealed bags are labeled and transported to a robotic station, where a robotic arm places them in carton boxes. Once three bags are placed in each box, the boxes are closed and sent to the palletizing station. When a pallet is fully loaded, it is wrapped with nylon film and prepared for collection by forklift operators.

For better clarity and more effective organization of the project development, the Digital Twin processing plant was conceptually divided into seven functional modules. Module 1 covers all operations from the input of the filled plastic boxes up to the roller conveyor that transports the apples to the drying unit. The drying machine, along with its associated roller conveyors, is designated as Module 2. Module 3 corresponds to the weighing system, while the bag-filling machine constitutes Module 4. Module 5 includes the robotic arm station, as well as the conveyor belts responsible for handling both bags and carton boxes. Finally, Modules 6 and 7 are assigned to the palletizer and the pallet strapping system, respectively. An additional subsection is dedicated to traceability, which in essence consists of a labeling system distributed along the processing line. Its purpose is to attach all necessary product information to both bags and boxes, ensuring that each item can be accurately tracked throughout the production and distribution process.

Chapter 2

Digital Twin Design and Development

In the following chapter, the attention is turned towards the steps followed in the development of the project. This section describes the fundamental work of modeling, assembling, and animating machines. In addition to this, it also describes the process of programming and the organization of the POUs, along with the declaration of inputs and outputs, and the connection between the Digital Twin software and the PLC development environment.

2.1 3D Modeling

2.1.1 Module 1

The realization of this module begins with the development of a two-axis gantry robot. The process starts by designing rectangular and cylindrical bars, which constituted arms (Fig. 2.3) and shafts (Fig. 2.4) of the pantograph structure. The arms are modeled with circular holes to accommodate the shafts, allowing mechanical assembly of the pantograph system. Subsequently, both the top and bottom bases are modeled. The top base (Fig. 2.2) is connected to the pantograph and equipped with electric motors and speed reducers, whose function is to drive the robot horizontally and to raise or lower the pantograph vertically. The bottom base (Fig. 2.5), in contrast, is fitted with two cylindrical pneumatic pistons actuating onto the clamping brackets (Fig. 2.6) responsible for clamping and unclamping the boxes. These brackets move two articulated mechanisms (Fig. 2.7) that secure the box from below and from the sides, allowing the boxes to be tilted 30 degrees using two additional pneumatic pistons. The upper platform is then outfitted with wheels and mounted onto the beam support structure (Fig. 2.1) to enable translational movement from the conveyors to the water tank (Fig. 2.8). Additionally, the robot is equipped with analog sensors for position detection, and digital sensors to monitor box tilting and clamping states–features essential for PLC integration and control logic implementation. The components described above are designed

in *SolidWorks* and imported as `.obj` files to be assembled within the simulation environment. All remaining parts used are sourced from the standard component library available in *Machine Simulator 4* and are integrated directly into the robotic system.

Figure 2.1: Support beam structure.

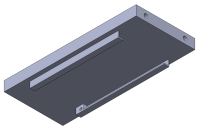


Figure 2.2: Robot top base.

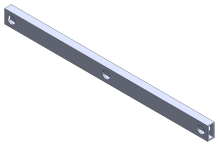


Figure 2.3: Pantograph arm.

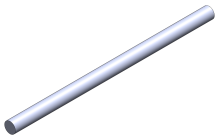


Figure 2.4: Pantograph shaft.

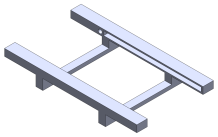


Figure 2.5: Robot bottom base.

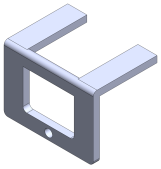


Figure 2.6: Clamping bracket.

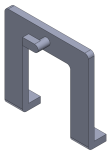


Figure 2.7: Turning bracket.

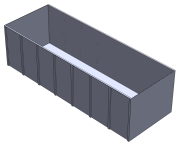


Figure 2.8: Water tank.

The assembly procedure is carried out by importing the individual components, scaling them to the desired dimensions, and appending them to existing components to obtain relative motion. Additionally, it becomes necessary to introduce auxiliary reference frames, as the imported parts have original reference frames positioned in ways that complicated motion programming. To address this, calculations are performed using rotation matrices to define and place auxiliary frames of reference in more convenient locations. Once all components are correctly aligned and positioned, the motion programming phase begins. The robot is configured to respond to analog input variables, enabling controlled motion along the horizontal and vertical axes. The result obtained is shown in Fig. 2.9.

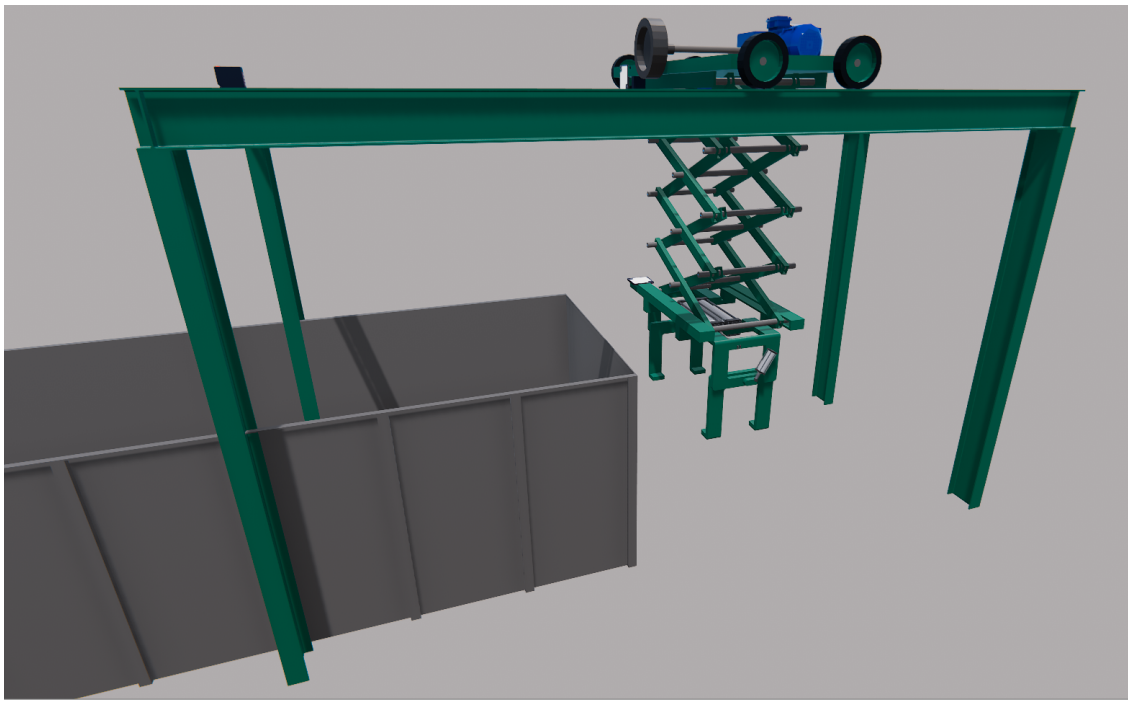


Figure 2.9: Robot assembly.

Now it is time to program the UDC so that it responds to commands and behaves like an actual industrial machine. In a real-world application, the belt driven by an electric motor is responsible for lifting and lowering the bottom platform. However, this mechanical principle cannot be replicated directly in the Digital Twin. Therefore, an alternative approach must be adopted. A kinematic analysis of the pantograph structure (Fig. 2.10) is conducted in order to define an effective motion strategy. It is important to note that the structure on the upper side is partially hinged to the top base and partially allowed to slide within a groove. The same configuration is mirrored on the lower side with respect to the bottom base. It results in having speeds and angles halved in those positions.

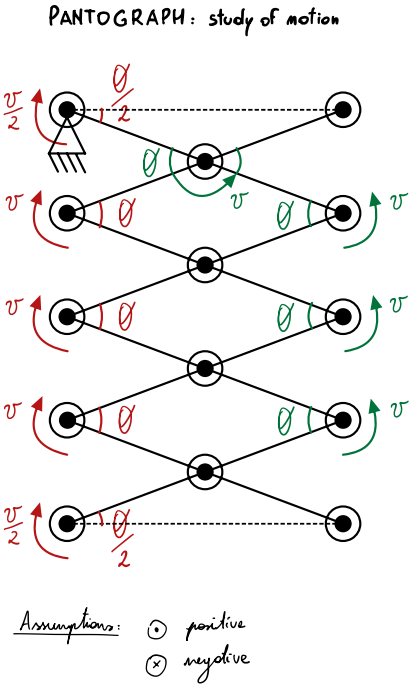


Figure 2.10: Pantograph study of motion.

Since in the physical system two variables are enough to govern the vertical movement, it was necessary (primarily for the sake of visual clarity in the simulation) to derive these variables internally. Within the UDC, the analog output signal from the PLC and a speed parameter were extracted and used to compute the other necessary control values. The following operations were performed as follows.

Figure 2.11: Graph controller board.

$$Analog\ Output = Y_{aim} = \theta \tag{2.1}$$

$$Y_{aimp} = Y_{aim} = \theta$$

$$Y_{aimn} = -Y_{aim} = -\theta$$

$$Y_{aim2p} = \frac{Y_{aim}}{2} = \frac{\theta}{2}$$

$$Y_{aim2n} = -\frac{Y_{aim}}{2} = -\frac{\theta}{2}$$

$$Parameter = Y_{vel} = v \tag{2.2}$$

$$Y_{vel2} = \frac{Y_{vel}}{2} = \frac{v}{2}$$

Once all required data are generated, the controller setup could be completed. Based on the way the pantograph is assembled, only the joints highlighted in red and green in Fig. 2.10 have to be actively controlled. While the others move as a consequence. The red joints are actuated using negative values, while the green ones responded to positive values. As an illustrative example, we consider the controller for the upper hinge attached to the top base (Fig. 2.12). This controller was labeled *Ymotionn*, where the suffix n denotes a negative rotation direction relative to the convention defined in the study of motion. The animation is triggered by the analog output *Yaim* and the digital output *Ymove*. When both conditions are satisfied, the hinge rotates about the x-axis by an angle of *Yaim2n*° at a speed of *Yvel2*,°s. The direction of rotation is determined by the sign of the angle, which in turn depends on whether the platform is to be raised or lowered. For others joints *Yaimn* and *Yveln* or *Yaimp* and *Yvelp* are used according to the rotation n for negative and p for positive.

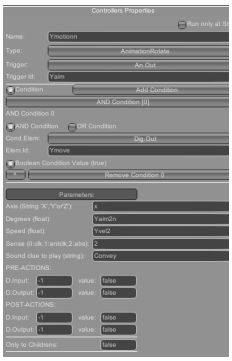


Figure 2.12: Example of Control Properties of a pantograph piece.

To complete the UDC programming, additional controllers must be added to the top base to manage the horizontal positioning of the robot. Furthermore, two more controllers need to be configured within the robot: one to read the angular position of pantograph arms via an analog input that simulates the behavior of a sensor, and another to simulate a sensor that detects the horizontal position of the top base. At this stage, only two motions remain to be implemented: the clamping and the overturning functions. The clamping and unclamping operations are achieved through controllers that actuate the pneumatic pistons in conjunction with the lateral brackets. The right and left brackets are assigned opposite directions of motion to perform the clamping and releasing actions. Additionally, *pre-actions* and *post-actions* can be configured to toggle digital inputs that serve as feedback signals, indicating the completion of specific actions. As for the overturning mechanism, it is necessary to employ the *Key Frame Animator* (Fig. 2.14) editor. By defining the initial and final poses of the relevant elements, the animation can be triggered using the *Graph Controller Board* (Fig. 2.13), where a digital output is assigned to control both the overturning and the return to the initial position. It is important to note that in this configuration, a single signal is generated once the animation is either played or reversed. Consequently, additional logic must be implemented to derive two distinct signals-one for the overturning action and another for the return motion.

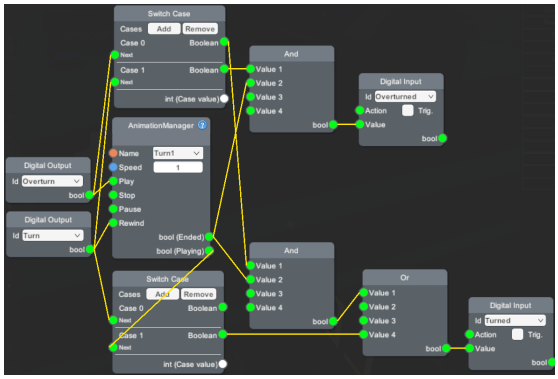


Figure 2.13: Graph controller board.

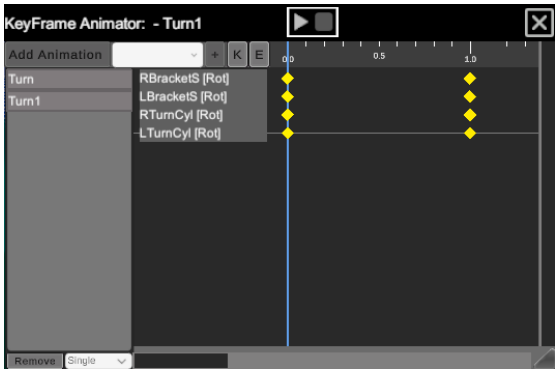


Figure 2.14: Key frame animator.

Once the robot is assembled and saved as a User Defined Component (UDC), it is possible to place it into the project as a machine. A feeding roller conveyor is added to transport the filled boxes to the gantry robot, along with a second roller conveyor to carry the empty boxes away from the robot after processing. In addition to the conveyors, safety elements are incorporated into the layout. These include safety grids, access doors, horizontal floor markings, and protective barriers, all aimed at replicating a realistic and compliant industrial environment (Fig. 2.15).

Figure 2.15: Module 1 assembly in the processing line.

For maintenance purposes now that the module 1 assembly is complete one can have a full list of sensors and actuators employed and their relative PLC variables.

Component	**Actuator**	**Comments**	**I/O**
Roller conveyor 1	Electric motor	Conveyor advance	QM01CR01: BOOL
Roller conveyor 2	Electric motor	Conveyor advance	
Roller conveyor 3	Electric motor	Conveyor advance	QM01CR03: BOOL
Roller conveyor 4	Electric motor	Conveyor advance	QM01CR04: BOOL
Roller conveyor 5	Electric motor	Roller conveyor adv.	
	Pneumatic cylinder	Chain lift/lower	QM01CU01: BOOL
	Electric motor	Chain conveyor adv.	QM01CL01: BOOL
Roller conveyor 6	Electric motor	Conveyor advance	
Roller conveyor 7	Electric motor	Conveyor advance	QM01CR07: BOOL
Roller conveyor 8	Electric motor	Conveyor advance	
UDC Robot	Electric motor	Robot horiz. motion	QM01XMV1: BOOL;
			AQM01XAM1: REAL;
	Electric motor	Robot vert. motion	QM01YMV1: BOOL;
			AQM01YAM1: REAL;
	Pneumatic cylinder	Box clamping	QM01CLM1: BOOL;
			QM01ULM1: BOOL;
	Pneumatic cylinder	Box turning	QM01OTR1: BOOL;
			QM01BTR1: BOOL;

Table 2.1: Actuators and I/O mapping

Component	Sensor	Comments	I/O
Photocell 1	Digital photosensor	Object detected	IM01PS01: BOOL;
Photocell 2	Digital photosensor	Object detected	IM01PS02: BOOL;
Photocell 3	Digital photosensor	Object detected	IM01PS03: BOOL;
Photocell 4	Digital photosensor	Object detected	IM01PS04: BOOL;
Photocell 5	Digital photosensor	Object detected	IM01PS05: BOOL;
Photocell 6	Digital photosensor	Object detected	IM01PS06: BOOL;
UDC Robot	Analog photosensor	Robot horizontal pos.	AIM01XPS1: REAL;
	Analog photosensor	Robot vertical pos.	AIM01YPS1: REAL;
	Mag. prox. sensor	Box clamped	IM01CLD1: BOOL;
	Mag. prox. sensor	Box unclamped	IM01ULD1: BOOL;
	Mag. prox. sensor	Box overturned	IM01OTD1: BOOL;
	Mag. prox. sensor	Box Backturned	IM01BTD1: BOOL;
Roller conv. 5	Mag. prox. sensor	Chains lifted	IM01CT01: BOOL;

Table 2.2: Sensors and I/O mapping

Component	Emergency	Comments	I/O
Siren	Voice coil	Acoustic alarm	QM01SRN1: BOOL;
Beacon	Led	Red light	QM01BCR1: BOOL;
	Led	Yellow light	QM01BCY1: BOOL;
	Led	Green light	QM01BCG1: BOOL;
Barrier 1	Digital photosensor	Object detected	IM01SBR1: BOOL;
Barrier 2	Digital photosensor	Object detected	IM01SBR2: BOOL;
Door	Digital sensor	Door opened	IM01SDO1: BOOL;
	Digital signal	Door open permit	QM01SDP1: BOOL;

Table 2.3: Emergency devices and I/O mapping

Component	HMI	Comments	I/O
ON/OFF	Switch	-	IM01ONF1: BOOL;
	Led	-	QM01LON1: BOOL;
Start	Button	-	IM01STR1: BOOL;
	Led	-	QM01LSR1: BOOL;
Stop	Button	-	IM01STP1: BOOL;
	Led	-	QM01LST1: BOOL;
Pause	Button	-	IM01PSE1: BOOL;
	Led	-	QM01LPS1: BOOL;
Reset	Button	-	IM01RST1: BOOL;
	Led	-	QM01LRS1: BOOL;
Emergency	Twist release button	-	IM01EME1: BOOL;
Open door	Switch	-	IM01XDO1: BOOL;

Table 2.4: HMI devices and I/O mapping

Component	Visibility	Comments	I/O
Box generator	Generator block	Generation of boxes	QM01BG01: BOOL;
Box destructor	Destructor block	Destruction of boxes	QM01BD01: BOOL;
Apple generator	Generator block	Generation of apples	QM01AG01: BOOL;

Table 2.5: Visibility components and I/O mapping

2.1.2 Module 2

This next module concerns the conveying of apples out of the water, their manual sorting, and subsequent drying in preparation for weighing and packaging. While this step is relatively simple from both a modeling and conceptual standpoint, there are certain aspects that warrant closer examination. One such consideration is the decision to use a flap conveyor belt to transport apples from the water tank to the sorting bench. In real industrial applications, a roller conveyor is typically employed, as it performs better by not retaining residual water. However, due to the physics limitations of the simulation environment, the mesh of each apple is approximated as a sphere, while the roller conveyor surface is modeled as a flat plane. This mismatch causes the apples to roll uncontrollably on the conveyor, making it impractical to replicate the actual behavior of the machine. As a result, a flap conveyor belt was chosen as a more stable alternative for simulation purposes.

Figure 2.16: Drier machine assembly.

Figure 2.17: Module 1 assembly in the processing line.

The drying machine itself consists of a tunnel equipped with rotating brush rollers and overhead fans that blow air onto the apple surfaces. This design ensures that the apples exiting the tunnel are sufficiently dry and ready for the weighing and packing stages. Due to the fact that it is not possible to change visual appearance of conveyors, brush rollers are represented as normal rollers. As with the robot described in Sect. 2.1.1, this machine was modeled as a User Defined Component (UDC) as in Fig. 2.16. However, in this case, it was not necessary to create 3D components from scratch, as the software's internal library provided adequate resources. Nevertheless, like all UDCs, the system required controller integration: the fans and tunnel lighting were linked to one digital output, and the conveyor mechanism was connected to another. Once ready the machine was assembled along the processing line. Some roller conveyors were added between the UDC and the tank, as well as after, before Module 3. The final Module 2 look is visible in Fig. 2.17.
Always for maintenance purposes, as in the previous case, it is possible to extract a full list of sensors and actuators employed and their relative PLC variables.

Component	**Actuator**	**Comments**	**I/O**
Flap conveyor 1	Electric motor	Conveyor advance	QM01ACT1: BOOL;
Roller conveyor 1	Electric motor	Conveyor advance	
Roller conveyor 2	Electric motor	Conveyor advance	
UDC Drier	Electric motor	Conveyor advance	
	Electric motor	Fan switching	QM02DVL1: BOOL;
	Switching	Light switching	

Table 2.6: Actuators and I/O mapping

Component	**Emergency**	**Comments**	**I/O**
Siren	Voice coil	Acoustic alarm	QM02SRN1: BOOL;

Table 2.7: Emergency devices and I/O mapping

Component	**HMI**	**Comments**	**I/O**
ON/OFF	Switch	-	IM02ONF1: BOOL;
	Led	-	QM02LON1: BOOL;
Pause	Button	-	IM02PSE1: BOOL;
	Led	-	QM02LPS1: BOOL;
Reset	Button	-	IM02RST1: BOOL;
	Led	-	QM02LRS1: BOOL;
Emergency 1	Twist release button	-	IM02EME1: BOOL;
Emergency 2	Twist release button	-	IM02EME2: BOOL;
Emergency 3	Twist release button	-	IM02EME3: BOOL;

Table 2.8: HMI devices and I/O mapping

2.1.3 Module 3

Module 3 is particularly articulated, with its core functionality centered around the weighing system. This module aims to replicate, as faithfully as possible, the behavior of an actual industrial weighing system. It comprises a feeding conveyor onto which apples arriving from the drying unit are spread and subsequently directed toward individual weighing stations. These stations consist of four principal components:

- a conveyor belt, referred to as the *singulator*;
- a first flap;
- a second flap equipped with an embedded load cell for weighing;
- a photo sensor.

All stations operate in coordination to supply one or two bag fillers, which are described in Sect. 2.1.4. Although industrial systems typically include more weighing stations than those implemented here, a symbolic number of six stations is selected. This limitation is imposed by the computational resources available on the machine used to run the simulation, as well as the complexity of managing a large number of variables simultaneously. In real-world installations, it is not uncommon to find more than ten weighing stations servicing two bag-filling units, thereby enabling higher throughput and operational efficiency.

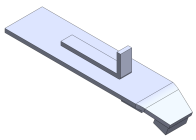


Figure 2.18: Conveyor flap.

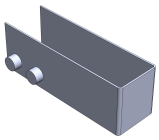


Figure 2.19: Hopper flap.

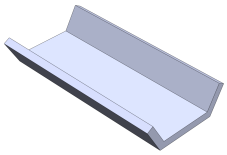


Figure 2.20: Hopper base.

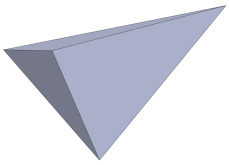


Figure 2.21: Wedge shaped element.

The components illustrated in Fig.2.18 to Fig.2.21 are specifically developed for modeling this module. It is necessary to replicate the two pneumatically actuated flapper elements: one located at the end of the conveyor belt referred to as the *singulator* (Fig.2.18), and the other positioned above the weighing hopper (Fig.2.19). In this latter location, directly beneath the flapper, a weighing plate is placed to receive the apples before the weighing process takes place (Fig. 2.20). Furthermore, an additional component must be designed to prevent apples from becoming stuck between adjacent *singulators* (Fig. 2.21). To address this, a wedge-shaped element is modeled, inspired by solutions used in real processing lines. It is important to note that in *Machine Simulator 4*, every component-whether imported from the internal library or brought in as a 3D model from external CAD software-must be assigned a simplified physical shape. The simulator does not allow physical interactions to be defined directly from the surface mesh of imported models. Instead, predefined bounding geometries such as cubes/parallelepipeds, spheres, cylinders, or ovals must be used. For the aforementioned wedge-shaped component, the oval

approximation is selected, as it best fulfills the functional requirements of the simulation. These components along with other picked from the internal *Machine Simulator 4* library result in the modular shape as in Fig. 2.22 and Fig. 2.23.
As far as the controllers are concerned, most of the actuation logic is implemented using the *Key Frame Animator* feature. This approach allowed the coordinated movement of the pneumatic piston and the upper flap by means of two digital outputs-one for opening and one for closing-while also generating two corresponding digital inputs to indicate when the flap is fully open or fully closed using the logic as in Sect. 2.1.1, Fig. 2.13. The same control strategy is applied to the bottom flap. In contrast, the conveyor belt is simply linked to a digital output to toggle its forward motion. Once all six modules are assembled and integrated into the processing line, the complete weighing unit appears as shown in Fig. 2.24.

Figure 2.22: *Singulator* and upper flap.

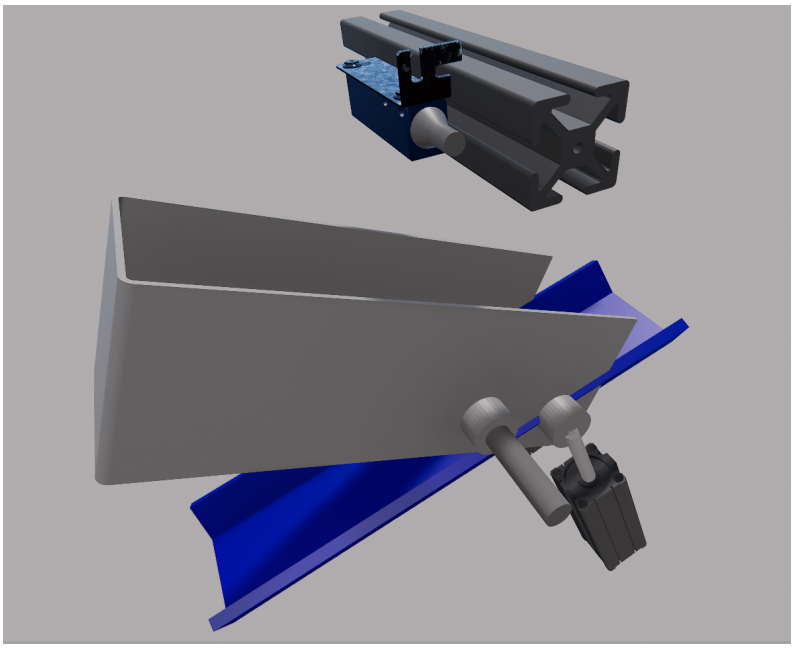


Figure 2.23: Bottom flap.

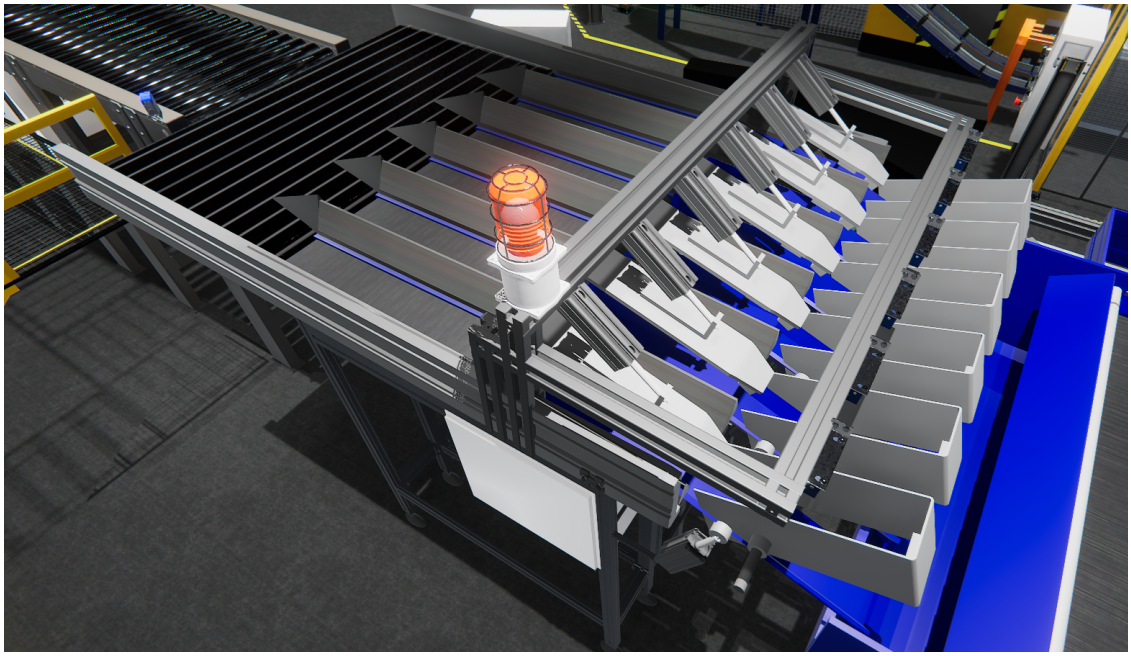


Figure 2.24: Module 3 assembly in the processing line.

Now, since the simulation is not connected to a real process, and all the apples have the same physical shape, in principle they all have the same weight. For this reason, the load cells have to be *programmed* to return some random values of weight each time a digital

output is sent to a specific cell, to obtain variables weight and perform calculation in the PLC environment, that we will deepen in later chapters. As far as this programming topic is concerned the issue is solved by using some lines of code in *C# Scripts Editor* available on *Machine Simulator 4*. First of all it is necessary to import the random number generation function.

```
// Import random numbers generation fuction
System.Random rand = new System.Random();
```

Once the function is defined and available, it becomes possible to proceed with the main program logic. Beginning with the first weighing line, the boolean variable `readWeight1` is declared and mapped to machine digital output number 60. An `if` statement is then used to evaluate the state of this variable. When `readWeight1` is `true`, a random value is generated within the range of 495 to 510 grams, representing the weight of apples contained in the hopper. This value range is chosen based on the fact that the machine typically opens three hoppers simultaneously, and in order to fill bags of approximately 1500 grams, unloading around 500 grams from each hopper is a logical approximation. The generated random weight is stored in the integer variable `randomWeight1`, which is subsequently sent to analog input port 2, making it available for reading and processing by the PLC. The same programming logic holds for the remaining 5 weighing lines simply assigning different ports as digital outputs and analog inputs.

```
public void Main()
{
    // ---------- WEIGHER LINE 301 ----------
    // Read the current state of the Weighing Reading Request
    bool readWeight1 = IOManager.GetOutput(60);

    // Rising edge detection
    if (readWeight1)
    {
        // Generate a random value between 495 and 510
        int randomWeight1 = rand.Next(495, 511); // Upper bound is exclusive
        // Write the random weight to the analog input
        IOManager.SetAInput(2, randomWeight1);
    }

[...]

}
```

Once the machine stops, all the inputs are set to zero avoiding weight misreading when restarting.

```
public void Finish()
{
    IOManager.SetAInput(2, 0);
    IOManager.SetAInput(3, 0);
```

```
5       IOManager.SetAInput(4, 0);
6       IOManager.SetAInput(5, 0);
7       IOManager.SetAInput(6, 0);
8       IOManager.SetAInput(7, 0);
9   }
```

For maintenance purposes it is possible to list in a table all the components with their respective I/O variable names. In this case pieces are redundant, but being controlled independently will appear as many times as needed.

Component	**HMI**	**Comments**	**I/O**
ON/OFF	Switch	-	IM03ONF1: BOOL;
	Led	-	QM03LON1: BOOL;
Pause	Button	-	IM03PSE1: BOOL;
	Led	-	QM03LPS1: BOOL;
Reset	Button	-	IM03RST1: BOOL;
	Led	-	QM03LRS1: BOOL;
Emergency 1	Twist release button	-	IM03EME1: BOOL;

Table 2.9: HMI devices and I/O mapping

Component	**Emergency**	**Comments**	**I/O**
Siren	Voice coil	Acoustic alarm	QM03SRN1: BOOL;

Table 2.10: Emergency devices and I/O mapping

Component	Actuator	Comments	I/O
Weighing line 1	Pneumatic cylinder	Close flap	QM03W1C1: BOOL;
		Open flap	QM03W1O1: BOOL;
	Pneumatic cylinder	Close flap	QM03W1C2: BOOL;
		Open flap	QM03W1O2: BOOL;
	Electric motor	Conv.advance	QM03W1AD: BOOL;
	Digital signal	Weight reading	QM03RWG1: BOOL;
Weighing line 2	Pneumatic cylinder	Close flap	QM03W2C1: BOOL;
		Open flap	QM03W2O1: BOOL;
	Pneumatic cylinder	Close flap	QM03W2C2: BOOL;
		Open flap	QM03W2O2: BOOL;
	Electric motor	Conv.advance	QM03W2AD: BOOL;
	Digital signal	Weight reading	QM03RWG2: BOOL;
Weighing line 3	Pneumatic cylinder	Close flap	QM03W3C1: BOOL;
		Open flap	QM03W3O1: BOOL;
	Pneumatic cylinder	Close flap	QM03W3C2: BOOL;
		Open flap	QM03W3O2: BOOL;
	Electric motor	Conv.advance	QM03W3AD: BOOL;
	Digital signal	Weight reading	QM03RWG3: BOOL;
Weighing line 4	Pneumatic cylinder	Close flap	QM03W4C1: BOOL;
		Open flap	QM03W4O1: BOOL;
	Pneumatic cylinder	Close flap	QM03W4C2: BOOL;
		Open flap	QM03W4O2: BOOL;
	Electric motor	Conv.advance	QM03W4AD: BOOL;
	Digital signal	Weight reading	QM03RWG4: BOOL;
Weighing line 5	Pneumatic cylinder	Close flap	QM03W5C1: BOOL;
		Open flap	QM03W5O1: BOOL;
	Pneumatic cylinder	Close flap	QM03W5C2: BOOL;
		Open flap	QM03W5O2: BOOL;
	Electric motor	Conv.advance	QM03W5AD: BOOL;
	Digital signal	Weight reading	QM03RWG5: BOOL;
Weighing line 6	Pneumatic cylinder	Close flap	QM03W6C1: BOOL;
		Open flap	QM03W6O1: BOOL;
	Pneumatic cylinder	Close flap	QM03W6C2: BOOL;
		Open flap	QM03W6O2: BOOL;
	Electric motor	Conv.advance	QM03W6AD: BOOL;
	Digital signal	Weight reading	QM03RWG6: BOOL;

Table 2.11: Actuators and I/O mapping

Component	Sensor	Comments	I/O
Weighing line 1	Mag. prox. sensor	Flap closed	IM03W1L1: BOOL;
		Flap opened	IM03W1P1: BOOL;
	Mag. prox. sensor	Flap closed	IM03W1L2: BOOL;
		Flap opened	IM03W1P2: BOOL;
	Digital photosensor	Object detected	IM03W1PS: BOOL;
	Analog load cell	Weight detection	AIM03W1WG: REAL;
Weighing line 2	Mag. prox. sensor	Flap closed	IM03W2L1: BOOL;
		Flap opened	IM03W2P1: BOOL;
	Mag. prox. sensor	Flap closed	IM03W2L2: BOOL;
		Flap opened	IM03W2P2: BOOL;
	Digital photosensor	Object detected	IM03W2PS: BOOL;
	Analog load cell	Weight detection	AIM03W2WG: REAL;
Weighing line 3	Mag. prox. sensor	Flap closed	IM03W3L1: BOOL;
		Flap opened	IM03W3P1: BOOL;
	Mag. prox. sensor	Flap closed	IM03W3L2: BOOL;
		Flap opened	IM03W3P2: BOOL;
	Digital photosensor	Object detected	IM03W3PS: BOOL;
	Analog load cell	Weight detection	AIM03W3WG: REAL;
Weighing line 4	Mag. prox. sensor	Flap closed	IM03W4L1: BOOL;
		Flap opened	IM03W4P1: BOOL;
	Mag. prox. sensor	Flap closed	IM03W4L2: BOOL;
		Flap opened	IM03W4P2: BOOL;
	Digital photosensor	Object detected	IM03W4PS: BOOL;
	Analog load cell	Weight detection	AIM03W4WG: REAL;
Weighing line 5	Mag. prox. sensor	Flap closed	IM03W5L1: BOOL;
		Flap opened	IM03W5P1: BOOL;
	Mag. prox. sensor	Flap closed	IM03W5L2: BOOL;
		Flap opened	IM03W5P2: BOOL;
	Digital photosensor	Object detected	IM03W5PS: BOOL;
	Analog load cell	Weight detection	AIM03W5WG: REAL;
Weighing line 6	Mag. prox. sensor	Flap closed	IM03W6L1: BOOL;
		Flap opened	IM03W6P1: BOOL;
	Mag. prox. sensor	Flap closed	IM03W6L2: BOOL;
		Flap opened	IM03W6P2: BOOL;
	Digital photosensor	Object detected	IM03W6PS: BOOL;
	Analog load cell	Weight detection	AIM03W6WG: REAL;

Table 2.12: Sensors and I/O mapping

2.1.4 Module 4

Module 4 is responsible for the bag-filling operation. It receives apples from the weighing hoppers and conveys them into polyethylene or low-density polyethylene (LDPE) bags. These bags are unrolled from a plastic reel on which they are preformed and stored. Each bag is sealed on three sides and connected to the next one via perforated (dashed) cuts, which facilitate separation once the bag is filled and top-sealed. After the bags are filled, they are transported by a conveyor belt to the sealing jaws, where they are sealed and prepared for packing. To simplify both modeling and control logic, the module has been subdivided into three sub-modules:

1. flap conveyor;
2. bag unroller and tensioning unit;
3. bag-filling unit.

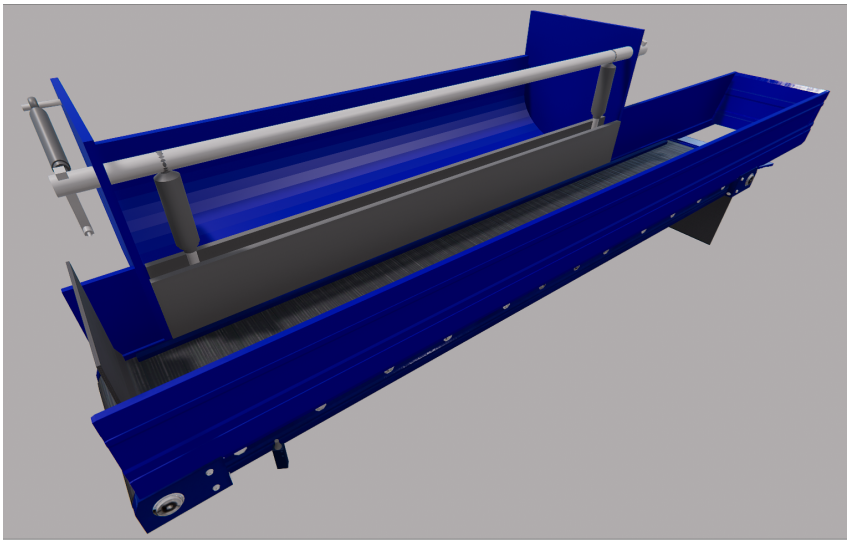


Figure 2.25: Flap conveyor.

Sub-module 1 (Fig. 2.25) has been modeled almost entirely using components from the *Machine Simulator 4* library. First, the flap conveyor is inserted, followed by the surrounding structural elements. Since apples are sensitive to bruising, it is necessary to add a flapper element that gently assists the fruit in rolling down onto the conveyor. Four pneumatic components are involved in operating this flapper: two are used to rotate the flapper by approximately 90°, and the other two lift the flapper element to allow it to return to its initial position in preparation for the next discharge. As in previous modules, controllers are employed to lift the flapper, and the *Key Frame Animator* is used to animate both the turning flapper and the movement of the cylinders. Moreover, the same logic described in previous subsections is used to signal the completion of the flap's turning in both directions. Regarding the conveyor, it includes two flaps used to

push the fruits toward the opening above the bag filler. This conveyor is equipped with an optical photo-sensor that detects the position of the flaps to control motion. Ideally, an encoder should have been used, but the software does not support its integration on this type of conveyor. Therefore, a simpler method is adopted.

Sub-module 2 (Fig. 2.26) concerns the unrolling of the bags. The plastic film must remain under tension to avoid wrinkles and machine jamming. For this purpose, spring-actuated systems are employed to tension the film. Two sets of red rollers are used: one set of two rollers is fixed, while a second set of three rollers is mounted on springs that pull the two sets apart. When the film is pulled by the filling machine, the elastic force is temporarily overcome, and the rollers move closer. During filling, the spring force pushes the rollers apart again. At maximum extension, an inductive sensor is triggered to break the drum onto which the roll is fitted to prevent its uncontrolled spinning. The spring system is modeled using visual effects, while the activation of the braking system is handled in the automation logic. As with the other sub-modules, controllers are used to actuate mechanical elements and to generate input signals reflecting the machine state.

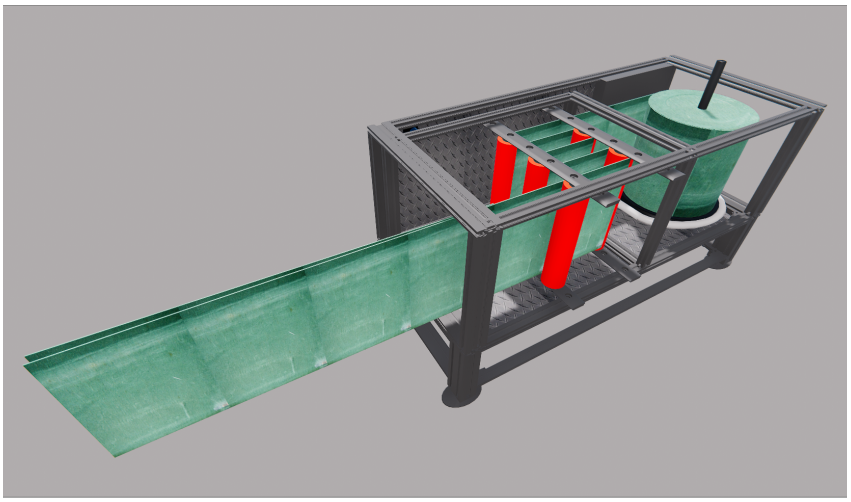


Figure 2.26: Bag unroller and tensioning unit.

Sub-module 3 (Fig. 2.27) is the heart of this fourth module. It consists of an aluminum frame onto which all components are mounted. The upper belts, which run face to face, pinch the plastic film to pull the bags forward. For each new bag, the belts stop to allow for filling. The presence of a new bag is detected by a black stripe on top of it, which is read by a color sensor. When the bag reaches the filling position, a flapper lifts it to avoid apples falling from excessive height. This flapper then lowers as the bag is being filled. After filling, the bag is pulled onto a belt conveyor. However, it remains clamped at the top by the belts, while the bottom conveyor moves at the same speed to support the weight of the filled bag. The bags are then hot-sealed and cut using specialized heat-sealing jaws. Once the next bag is filled, the sealed bag continues on the conveyor to the next module. Additionally, for safety reasons, a barrier sensor is installed on the lateral side of the machine frame. This design choice aims to prevent operators from inserting

their hands into the machine in the event of a jam while it is in operation. Typically, a sliding safety panel would be employed; however, such a solution would obstruct view of the machine's internal mechanisms. Consequently, a non-obstructive barrier sensor is preferred to maintain both safety and operational transparency. Furthermore, additional twist-release emergency stop buttons are integrated into the machine when assembled within the processing line, following the same approach adopted in the previous modules.

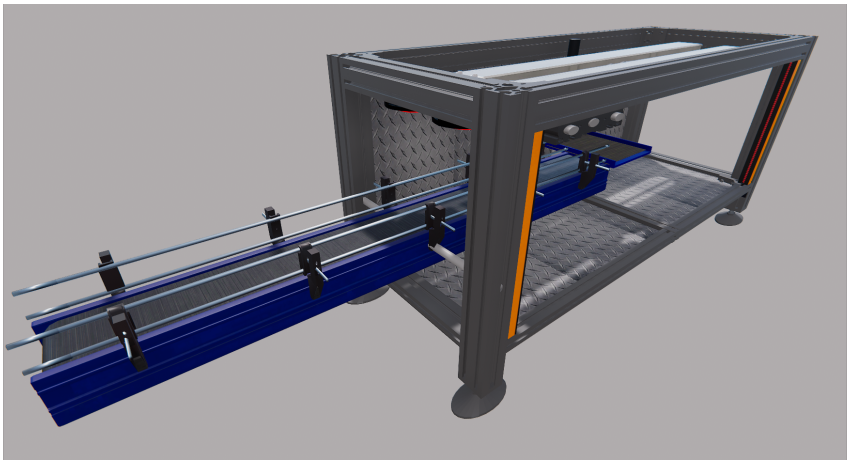


Figure 2.27: Bag-filling unit.

For better clarity, the complete assembly of the machine within the processing line environment is shown in Fig. 2.28.

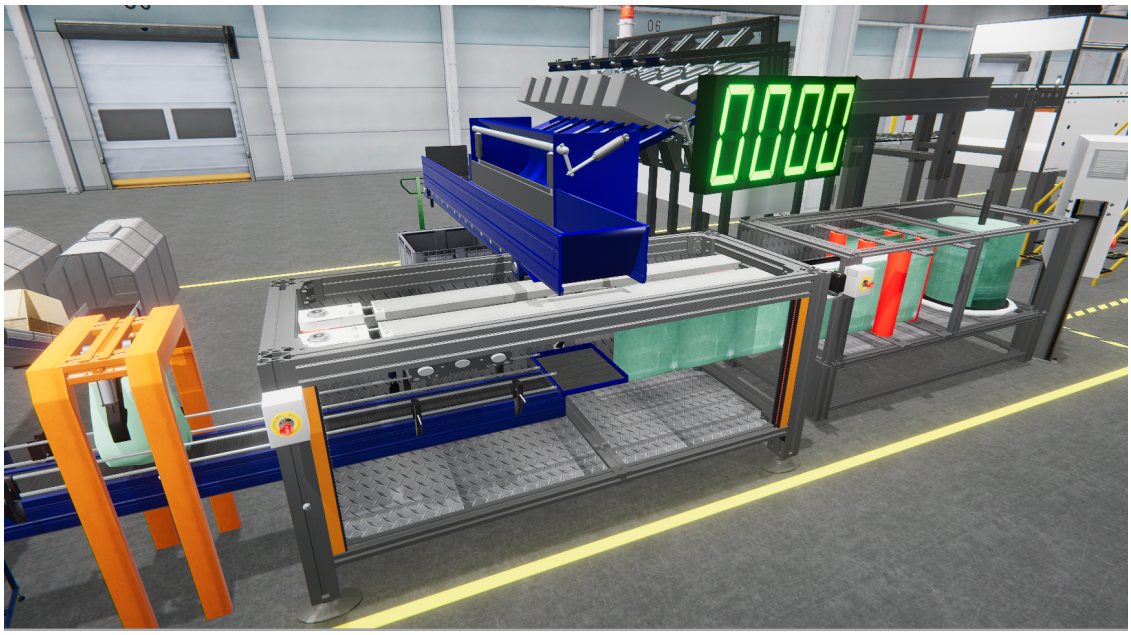


Figure 2.28: Module 4 assembly in the processing line.

For maintenance purposes, it is possible to list in a table all the components with their respective I/O variable names.

Component	Actuator	Comments	I/O
UDC Flap conv.	Pneumatic cylinder	Flap turn	QM04FTR1: BOOL;
		Flap backturn	QM04FBK1: BOOL;
	Pneumatic cylinder	Flap raise	QM04FRS1: BOOL;
	Electric motor	Conv. Advance	QM04FC01: BOOL;
UDC Bags unnroller	Inductive sensor	Sensor activation	QM04ISO1: BOOL;
	Pneumatic cylinder	Break	QM04BRK1: BOOL;
UDC Bags filler	Color sensor	Sensor activation	QM04CSO1: BOOL;
	Electric motor	Conv. Advance	QM04CB01: BOOL;
		Conveyor speed	AQM04CBS1: REAL;
	Pneumatic cylinder	Flap drop	QM04SPU1: BOOL;
	Pneumatic motor	Flap lift	QM04WMU1: BOOL;
		Flap low	QM04WMD1: BOOL;
	Pneumatic cylinder	Heat seal jaws	QM04SL01: BOOL;

Table 2.13: Actuators and I/O mapping.

Component	Sensor	Comments	I/O
UDC Flap conv.	Mag. prox. sensor	Flap turned	IM04FTB1: BOOL;
	Mag. prox. sensor	Flap backturned	IM04FTD1: BOOL;
	Mag. prox. sensor	Flap raise	IM04FRD1: BOOL;
	Digital photosensor	Conveyor ready	IM04FRY1: BOOL;
UDC Bags unroller	Inductive sensor	Object detected	IM04ISA1: BOOL;
UDC Bags filler	Color sensor	Sensor activated	IM04CSA1: BOOL;
		Analog color coding	AIM04CS01: REAL;
	Mag. prox. sensor	Flap up	IM04SUP1: BOOL;
	Digital photosensor	Flap top pos.	IM04STP1: BOOL;
	Digital photosensor	Flap bottom pos.	IM04SBT1: BOOL;
	Digital photosensor	Bag sealing pos.	IM04PS02: BOOL;

Table 2.14: Sensors and I/O mapping.

Component	**Emergency**	**Comments**	**I/O**
Barrier 1	Digital photosensor	Object detected	IM04SBR1: BOOL;

Table 2.15: Emergency and I/O mapping.

Component	**HMI**	**Comments**	**I/O**
Emergency 1	Twist release button		IM04EME1: BOOL;
Emergency 2	Twist release button		IM04EME2: BOOL;

Table 2.16: HMI and I/O mapping.

Component	**Visibility**	**Comments**	**I/O**
Bag generator	Generator block	Generation of bags	QM04BG01: BOOL;
Bags and drum	Digital signal	Bags unrolling	QM04UNR1: BOOL;
Bags flap	Bags advance	Bags adv. on flap sim.	QM04FBM1: BOOL;
Rollers set	Rollers set adv.	Spring element sim.	QM04NBD1: BOOL;
Bag detection	New bag gen.	Bag generated	IM04BGD1: BOOL;

Table 2.17: Visibility and I/O mapping.

2.1.5 Module 5

The following module 5 is particularly articulated; therefore, for the sake of clarity in the modeling description, it will be subdivided into three sub-modules, as done in the previous Sect. 2.1.4. The module is organized as follows:

1. anthropomorphic arm;
2. pneumatic positioner and box closer machine;
3. feeding conveyors.

In sub-module 1 pieces are first modeled in *SolidWorks* and converted into `.obj` files.

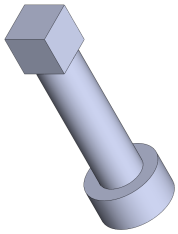


Figure 2.29: Link 1, robotic shoulder.

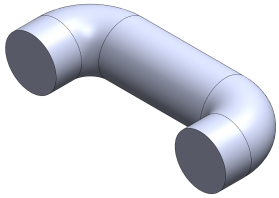


Figure 2.30: Link 2, robotic shoulder.

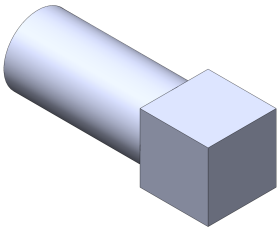


Figure 2.31: Link 3, robotic shoulder.

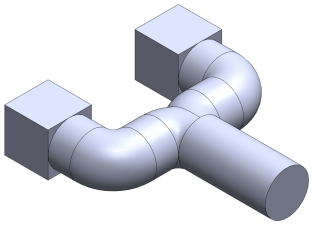


Figure 2.32: Link 1, robotic wrist.

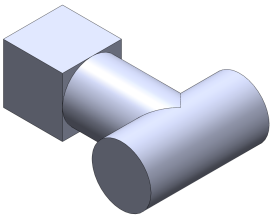


Figure 2.33: Link 2, robotic wrist.

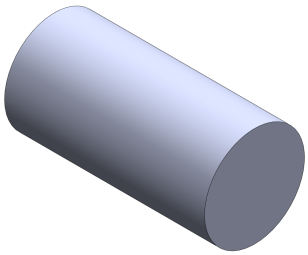
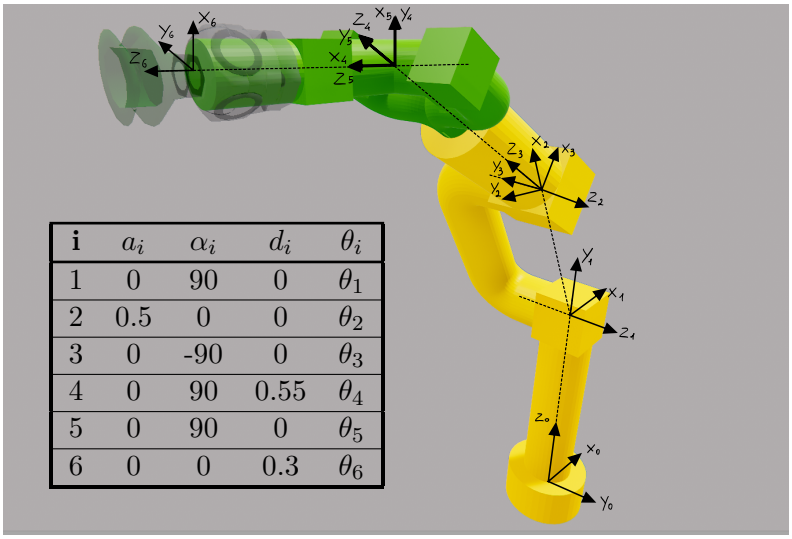


Figure 2.34: Link 3, robotic wrist.

Later, the pieces imported into the *Machine Simulator 4* environment are assembled to create the anthropomorphic arm and wrist as shown in Fig. 2.35. This operation makes it possible to obtain a robot for which all Denavit-Hartenberg (DH) parameters are known-unlike the pre-built robots available in the software library. Although the DH parameter



i	a_i	α_i	d_i	θ_i
1	0	90	0	θ_1
2	0.5	0	0	θ_2
3	0	-90	0	θ_3
4	0	90	0.55	θ_4
5	0	90	0	θ_5
6	0	0	0.3	θ_6

Figure 2.35: Manipulator assembly, DH parameters and reference frames.

matrix does not play a crucial role in this work, it is included as a foundation for potential future developments or advanced control strategies. As a matter of fact, this stage of the processing lines under study is originally performed manually. However, since it was redesigned to align with automation requirements, it was decided to include a robotic manipulator. Although the robot is not a real unit but a 3D prototype, its kinematic characteristics are fully known, which makes it suitable for additional testing or implementation of control algorithms. In particular, the Denavit-Hartenberg matrix proved useful during some MATLAB-based simulations-not discussed in detail here-which resulted in generating joint angles that contribute to smoother and more realistic manipulator movements. It is worth noting that the reference frames shown in Fig. 2.35, defined according to the Denavit-Hartenberg (DH) convention, do not correspond to the reference frames of the robotic arm links in the *Machine Simulator 4* environment. This discrepancy arises because the simulator always places automatically the same reference frame, with the y-axis pointing upwards, for every imported piece. As a consequence, the motion of the links have to be programmed using different reference frames. For instance, due to the upward-oriented y-axis, the first joint angle is referred to the rotation about the y-axis instead of the z-axis, as would be the case in the DH convention. This mismatch, however, is relatively easy to manage, since the reference frames are displayed when the pieces are imported into *Machine Simulator 4*. A limitation remains, nonetheless: when a controller is configured to read the angular positions of the robot joints via analog inputs, the simulator does not alter the representation of rotations around the y and z-axes, but it does affect those around the x-axis. The resulting visualization, illustrated in Fig. 2.36,

will later prove useful during the PLC programming phase.

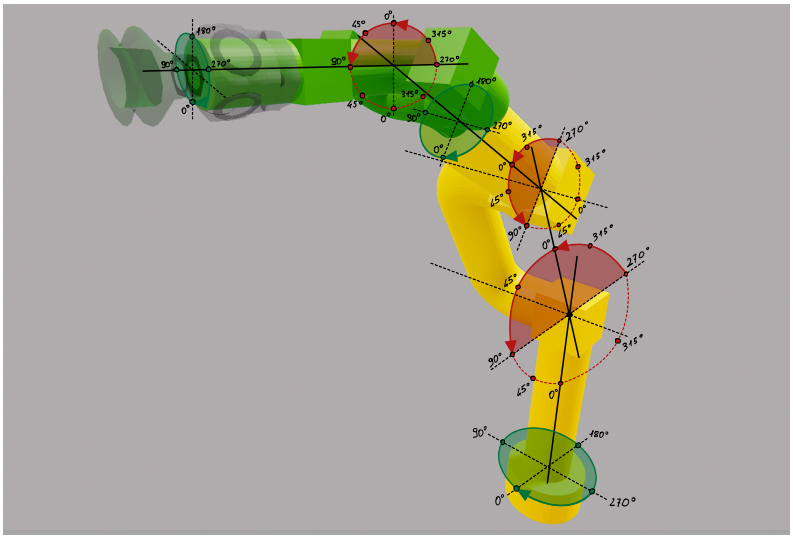


Figure 2.36: Alteration in joint angles reading.

As far as sub-module 2 is concerned, the modeling of the positioner does not require significant effort, as it is already available in the software library. This element is included to ensure the correct positioning of open carton boxes that need to be filled, and since the process has been automated, precise positioning is essential. On the other hand, the box closing machine has to be built from scratch, as it is not included in the simulator's library. However, this component does not involve many controllers or mechanisms, as accurately simulating the transition phase of box closing is nearly impossible. The adopted solution is to visually represent the box as open before entering the machine and then closed once exiting it. Despite the simplification, including this machine is necessary for a complete representation of the processing line. All these machines, along with the manipulator, are enclosed within a safety area using protective fences, similar to what has been done in module 2.1.1.
To conclude the description of this module, sub-module 3 is left. This sub-module includes all the conveyor systems. The box assembly process is omitted on purpose, as it falls outside the main scope of the project, which focuses on the packaging of fresh apples. Therefore, the boxes simply arrive at the box filling station via a conveyor belt from a previous stage. Beneath the box conveyor, enough space is available to accommodate the bag conveyor, which transports the filled apple bags from module 4. As a result, the two conveyor systems run in parallel on two separate planes. Finally, a spiral conveyor brings the boxes down to approximately the same level as the bags, facilitating easier handling by the manipulator. Then, assembling all the sub-modules together, module 5 results connected to module 4 as shown in Fig. 2.37.

Figure 2.37: Module 5 assembly in the processing line.

Component	**HMI**	**Comments**	**I/O**
ON/OFF	Switch		IM05ONF1: BOOL;
	Led		QM05LON1: BOOL;
Pause	Button		IM05PSE1: BOOL;
	Led		QM05LPS1: BOOL;
Reset	Button		IM05RST1: BOOL;
	Led		QM05LRS1: BOOL;
Emergency	Twist release button		IM05EME1: BOOL;
Open door	Switch		IM05XDO1: BOOL;

Table 2.18: HMI and I/O mapping.

Component	**Visibility**	**Comments**	**I/O**
Open box gen.	Generator block	Generation of boxes	QM05OBG1: BOOL;
Open box des.	Destructor block	Destruction of boxes	QM05OBD1: BOOL;
Close box gen.	Generator block	Generation of boxes	QM05CBG1: BOOL;
Close box des.	Destructor block	Destruction of boxes	QM05CBD1: BOOL;
Bag destructor	Destructor block	Destruction of bags	QM05BGD1: BOOL;

Table 2.19: Visibility and I/O mapping.

Component	Actuator	Comments	I/O
M05_BagBelt01	Electric motor	Conv. advance	QM05BGB1: BOOL;
M05_BagBelt02	Electric motor	Conv. advance	
M05_BagBelt03	Electric motor	Conv. advance	
M05_BagBelt04	Electric motor	Conv. advance	
M05_BagBelt05	Electric motor	Conv. advance	
M05_BagBelt10	Electric motor	Conv. advance	QM05BGB2: BOOL;
M05_BoxBelt01	Electric motor	Conv. advance	QM05BXB3: BOOL;
M05_BoxBelt02	Electric motor	Conv. advance	
M05_BoxBelt03	Electric motor	Conv. advance	
M05_BoxBelt04	Electric motor	Conv. advance	
M05_BoxBelt05	Electric motor	Conv. advance	
M05_BoxBelt06	Electric motor	Conv. advance	
M05_BoxBelt10	Electric motor	Conv. advance	QM05BXB1: BOOL;
M05_BoxBelt11	Electric motor	Conv. advance	
M05_BoxBelt20	Electric motor	Conv. advance	QM05BXB2: BOOL;
M06_Belt10	Electric motor	Conv. advance	QM05BXB4: BOOL;
M06_Belt11	Electric motor	Conv. advance	
M06_Belt12	Electric motor	Conv. advance	
M06_Belt13	Electric motor	Conv. advance	
M06_Belt14	Electric motor	Conv. advance	
Positioner	Pneumatic cylinder	Lift positioner	QM05PVP1: BOOL;
	Pneumatic cylinder	Flap home	QM05PHH1: BOOL;
		Flap position	QM05PHP1: BOOL;
UDC Rob. Arm	Suction cup	Bag grip	QM05RTK1: BOOL;
	Electric motor	Angle	AQM05RQ01: REAL;
		Speed	AQM05RQD1: REAL;
	Electric motor	Angle	AQM05RQ02: REAL;
		Speed	AQM05RQD2: REAL;
	Electric motor	Angle	AQM05RQ03: REAL;
		Speed	AQM05RQD3: REAL;
	Electric motor	Angle	AQM05RQ04: REAL;
		Speed	AQM05RQD4: REAL;
	Electric motor	Angle	AQM05RQ05: REAL;
		Speed	AQM05RQD5: REAL;
	Electric motor	Angle	AQM05RQ06: REAL;
		Speed	AQM05RQD6: REAL;

Table 2.20: Actuators and I/O mapping.

Component	**Sensor**	**Comments**	**I/O**
M05_BagBeltPS01	Digital photosensor	Object detected	IM05BGS1: BOOL;
M05_BagBeltPS10	Digital photosensor	Object detected	IM05BGS2: BOOL;
M05_BoxBeltPS10	Digital photosensor	Object detected	IM05BXS1: BOOL;
M05_BoxBeltPS20	Digital photosensor	Object detected	IM05BXS2: BOOL;
M05_BoxBeltPS01	Digital photosensor	Object detected	IM05BXS3: BOOL;
Positioner	Mag. prox. sensor	Bar home	IM05PVM1: BOOL;
	Mag. prox. sensor	Bar top	IM05PVW1: BOOL;
	Mag. prox. sensor	Flap home	IM05PHM1: BOOL;
	Mag. prox. sensor	Flap positioned	IM05PHW1: BOOL;
UDC Rob. Arm	Flow Sensor	Object picked	IM05RPD1: BOOL;
	Digital photosensor	Object detected	IM05RDT1: BOOL;
	Encoder	Angle	AIM05RQA1: REAL;
	Encoder	Angle	AIM05RQA2: REAL;
	Encoder	Angle	AIM05RQA3: REAL;
	Encoder	Angle	AIM05RQA4: REAL;
	Encoder	Angle	AIM05RQA5: REAL;
	Encoder	Angle	AIM05RQA6: REAL;

Table 2.21: Sensors and I/O mapping.

Component	**Emergency**	**Comments**	**I/O**
Siren	Voice coil	Acoustic alarm	QM05SRN1: BOOL;
Beacon	Led	Red light	QM05BCR1: BOOL;
	Led	Yellow light	QM05BCY1: BOOL;
	Led	Green light	QM05BCG1: BOOL;
Door	Digital sensor	Door opened	IM05SDO1: BOOL;
	Digital signal	Door open permit	QM05SDP1: BOOL;

Table 2.22: Emergency and I/O mapping.

2.1.6 Module 6

This module is composed of a palletizing machine, a belt conveyor for open boxes, and a roller conveyor for both empty and filled pallets. Contrary to the previous modules, it is not necessary to build machines from scratch, as the palletizer is already available in the *Machine Simulator 4* library. The modeling, therefore, is quite straightforward. Once the palletizer is placed, the connections with the conveyors are arranged accordingly. As a result, the belt conveyors transport the closed boxes from Module 5 onto the palletizer. The latter receives the pallets from the roller conveyors and stacks the boxes in two layers of four boxes each. Once the pallet is fully loaded, it is discharged onto the roller conveyor, which transports it toward the final module. To summarize the working principle of the palletizer machine, it includes a set of prebuilt digital inputs and outputs that allow the system to detect the presence of a pallet and lift it to a preset, adjustable height. Once the boxes are ready and arranged into a layer according to a specific layout-depending on the dimensions of both the pallet and the boxes-the gates open, allowing the boxes to gently fall onto the pallet. The platform then lowers slightly more than the height of a single layer, so the gates can close without pinching the boxes, and the next layer is prepared. The final assembly of the machine with the conveyors appears as in Fig. 2.38.

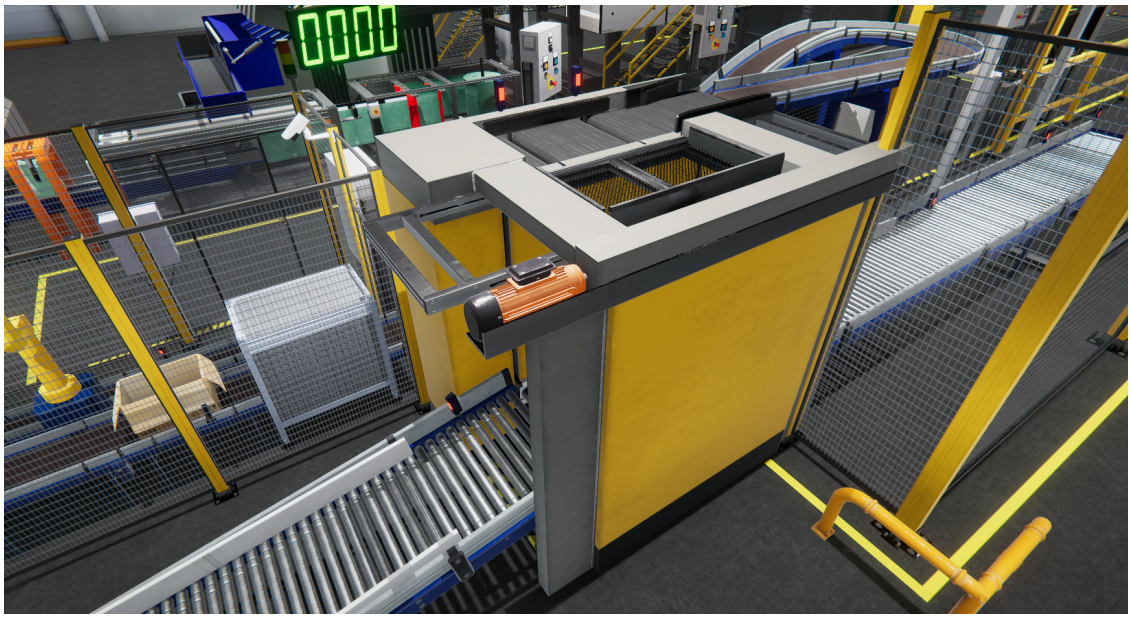


Figure 2.38: Module 6 assembly in the processing line.

Component	Actuator	Comments	I/O
M06_Roller10	Electric motor	Conveyor advance	QM06PIN1: BOOL;
M06_Roller11	Electric motor	Conveyor advance	
M06_Roller12	Electric motor	Conveyor advance	
M06_Roller13	Electric motor	Conveyor advance	
M06_Belt15	Electric motor	Conveyor advance	QM06BIN1: BOOL;
Palletizer	Electric motor	Conveyor advance	
	Pneumatic cylinder	Pusher	QM06PSH1: BOOL;
	Pneumatic cylinder	Gate	QM06OPG1: BOOL;
	Electric motor	Elevator up	QM06EUP1: BOOL;
	Electric motor	Elevator down	QM06ELW1: BOOL;
	Electric motor	Elevator descend	QM06EDS1: BOOL;
	Electric motor	Conveyor advance	QM06POU1: BOOL;
M06_Roller14	Electric motor	Conveyor advance	

Table 2.23: Actuators and I/O mapping.

Component	Sensor	Comments	I/O
Palletizer	Mag. prox. sensor	Pusher back	IM06PBK1: BOOL;
	Mag. prox. sensor	Pusher advanced	IM06PAD1: BOOL;
	Mag. prox. sensor	Gate opened	IM06GOD1: BOOL;
	Digital photosensor	Elevator stopped	IM06EST1: BOOL;
M06_sensor4	Digital photosensor	Pallet detected	IM06PS04: BOOL;
M06_sensor3	Digital photosensor	Pallet detected	IM06PS03: BOOL;
M06_sensor5	Digital photosensor	Pallet detected	IM06PS05: BOOL;
M06_sensor6	Digital photosensor	Pallet detected	IM06PS06: BOOL;
M06_sensor1	Digital photosensor	Box detected	IM06PS01: BOOL;
M06_sensor2	Digital photosensor	Box detected	IM06PS02: BOOL;

Table 2.24: Sensors and I/O mapping.

Component	Visibility	Comments	I/O
Pallet generator	Generator block	Generation of boxes	QM06PTG1: BOOL;
Destructor	Destructor block	Destruction of boxes	QM06DES1: BOOL;
Full pallet generator	Generator block	Generation of boxes	QM06GEN1: BOOL;

Table 2.25: Visibility and I/O mapping.

2.1.7 Module 7

As far module 7 is concerned, it does not include many elements. The key machine is the packing machine, that winds a nylon film or net around the boxes in such a way to keep them tight together and to the pallet in order not to fall and ease the shipping and stocking processes. Therefore the pallet coming from the previous module follows its path onto the roller conveyor till the packing machine. Here it is ready to be winded with the net and secured for being transported. When ready it travel to end of the conveyor and is ready to be collected by forklift operators. The assembly of the machine in the processing line is straight forward it has been added after the palletizer and enclosed with safety fences and a safety door to access the machine as shown in Fig. 2.39.

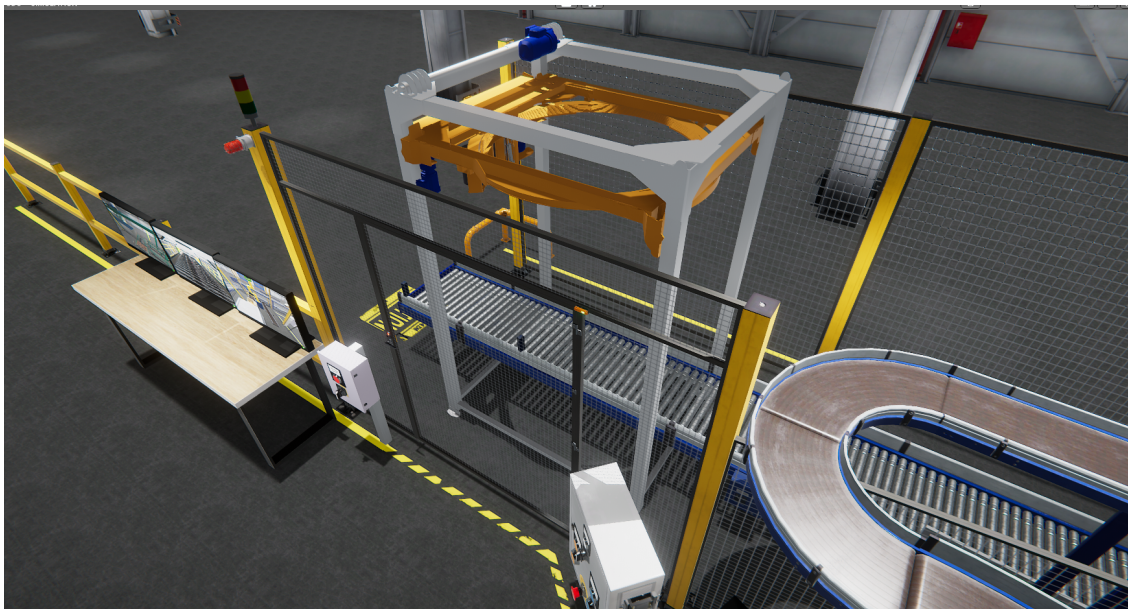


Figure 2.39: Module 7 assembly in the processing line.

Component	Actuator	Comments	I/O
M06_PACKING	Electric motor	Nylon winder	QM07MST1: BOOL;
	Electric motor	Platform height	AQM07YAX1: REAL;
M06_Roller15	Electric motor	Conveyor advance	QM07PIN1: BOOL;

Table 2.26: Actuators and I/O mapping.

Component	Sensor	Comments	I/O
M06_PACKING	Digital photosensor	Winder turn	IM07MTD1: BOOL;
	Analog sensor	Platform height	AIM07YEN1: REAL;
	Analog sensor	Winder postion	AIM07REN1: REAL;
M06_sensor7	Digital photosensor	Pallet detected	IM07PS07: BOOL;

Table 2.27: Sensors and I/O mapping.

Component	Emergency	Comments	I/O
Siren	Voice coil	Acoustic alarm	QM05SRN1: BOOL;
Beacon	Led	Red light	QM06BCR1: BOOL;
	Led	Yellow light	QM06BCY1: BOOL;
	Led	Green light	QM06BCG1: BOOL;
Door	Digital sensor	Door opened	IM06SDO1: BOOL;
	Digital signal	Door open permit	QM06SDP1: BOOL;

Table 2.28: Emergency and I/O mapping.

Component	HMI	Comments	I/O
ON/OFF	Switch		IM06ONF1: BOOL;
	Led		QM06LON1: BOOL;
Pause	Button		IM06PSE1: BOOL;
	Led		QM06LPS1: BOOL;
Reset	Button		IM06RST1: BOOL;
	Led		QM06LRS1: BOOL;
Emergency	Twist release button		IM06EME1: BOOL;
Open door	Switch		IM06XDO1: BOOL;

Table 2.29: HMI and I/O mapping.

Component	Visibility	Comments	I/O
Nylon cover gen.	Generator block	Generation	QM07NYG1: BOOL;
Destructor	Destructor block	Destructor	QM07DES1: BOOL;

Table 2.30: Visibility and I/O mapping.

2.1.8 Labelling machines

The following labeling machines are entitled to place labels onto either boxes and bags. Their assembly on the processing line is quite direct without assembling UDCs first, as their mechanics and structure are just schematic and in reality they can be quite complex. However they are needed for traceability purposes. Therefore they have been assembled directly on the processing line to ensure they could fit and adapt to the existent structure. Both of them are simply composed of two U-shaped components that joined with each other produce the structure with four legs that hold in the first case Fig. 2.40 the arm that pushes the label onto the bag. And in the second case Fig. 2.41 it holds the printer, such that boxes can be labeled from top. Moreover in both cases the structure accommodate for a photo sensor for bags and boxes detection and facilitate programming phase.

Component	Actuator	Comments	I/O
Labeler 1	Arm		QM04LBA1: BOOL;
	Printer		QM04LBL1: BOOL;
	Digital sensor		IM04LBE1: BOOL;
	Photosensor		IM04LBP1: BOOL;
Pause	Printer		QM05LBL1: BOOL;
	Photosensor		IM05LBP1: BOOL;

Table 2.31: Labelers and I/O mapping.

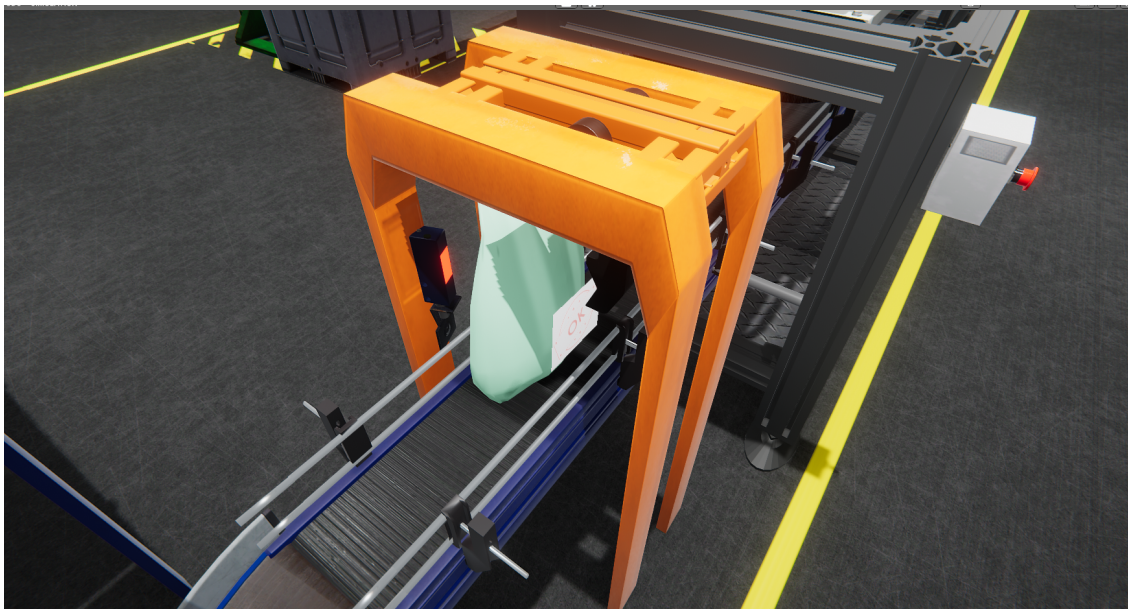


Figure 2.40: Module 4 labeler.

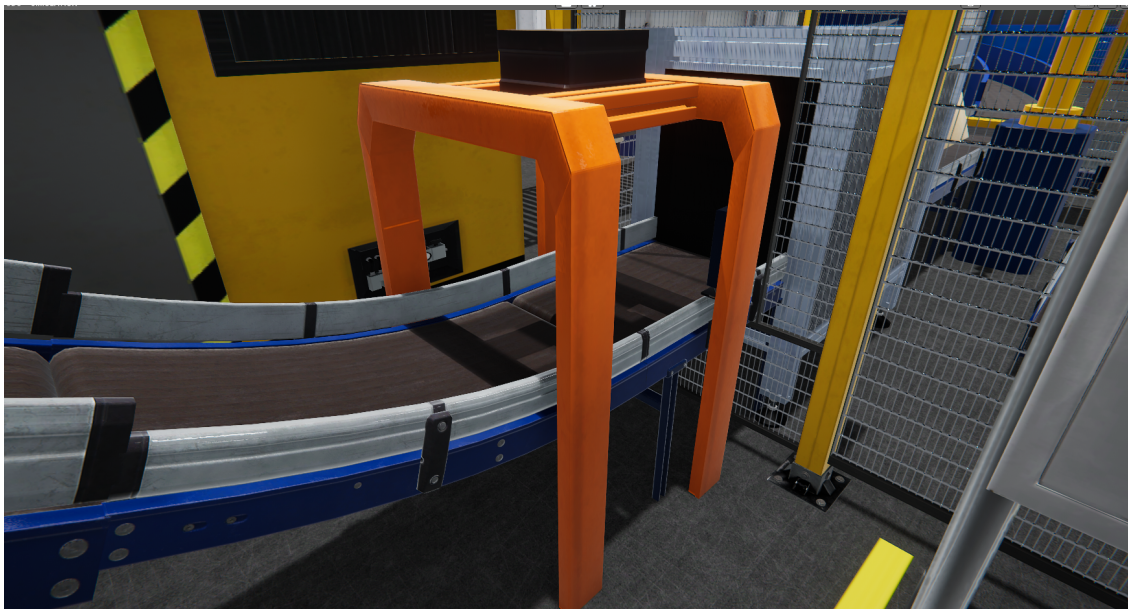


Figure 2.41: Module 5 labeler.

2.2 PLC programming

The PLC programming is designed following the same modular subdivision used in the modeling phase. Each module has been programmed independently using one or more Program Organization Units (POUs), each named according to the corresponding controlled module and machine. Additionally, separate POUs have been defined to manage start, stop, pause, and emergency procedures. Throughout the development process, a consistent programming approach is adopted: all POUs are implemented in pairs: one written in Sequential Function Chart (SFC), analogous to GRAFCET, and the other in Ladder Logic (LD). In this structure, the SFC program handles the conditions that define transitions between steps, while the LD program manages the actions associated with each step. This separation enhances clarity and simplifies program management.

2.2.1 Procedures

This subsection details the Program Organization Units (POUs) responsible for managing general procedures. Starting in order, the first POU is `MAIN_SFC`, which controls the ON/OFF switch that determines the operational state of the machine. The corresponding ladder logic, `MAIN_LD`, is composed of four pages and primarily manages the enabling (or "forcing") of machine modules. Forcing is implemented through the `SFCInit` function, embedded within each SFC POU. When triggered, `SFCInit` resets the associated GRAFCET to its initial step and holds it there. This mechanism enables a clear hierarchical structure within the PLC program: the `MAIN` routine governs the secondary POUs (`M0xSEC_SFCs`), which in turn control the machine-specific POUs within each module. In addition, `MAIN_LD` controls the `SFCInit` for both the `STOP_SFC`, `PAUSE_SFC` and the `MAINEME_SFC`, which manage the stop, pause and emergency cycles, respectively. The `STOP_SFC` is triggered by pressing the stop button for a duration longer than a preset time. When this timer is activated, a cascade of actions is initiated: yellow beacon lights begin blinking, the roller conveyor in Module 1 is emptied of boxes, modules 2 and 3 are emptied of apples (from the tank, drier, and weighing hoppers) and modules 4 and 5 discharge their contents. For Module 5, a box will be closed and ejected even if it contains fewer than the required three bags. This avoids the need for operators to enter the safety area for manual unloading. Modules 6 and 7 will output the remaining pallets, regardless of whether the expected number of boxes has been loaded. The last pallet may need to be manually revised-either by adding missing boxes or replacing empty ones. Once this unloading sequence completes, the machine halts and the red beacon lights activate, signaling that the individual modules and the general ON/OFF switch can be turned off. Alternatively, pressing the start button will resume operation. The `MAINEME_SFC` and the `MAINEME_LD` handle emergency situations triggered by any twist-release push button or safety barrier. When activated, this emergency step propagates a stop signal throughout the entire system. Importantly, the program retains memory of the specific module where the emergency was triggered. Consequently, resetting the emergency can only be done from that same module. This design prevents unauthorized or unsafe resets from other modules. To restore the system after an emergency: clear the obstruction (e.g., retract

from the barrier or release the push button), press the reset push button and the machine resumes operation from the point at which it was interrupted. A similar behavior has the PAUSE_SFC POU. This program has been thought to stop modules in groups: first group contains only module 1, second group modules 2,3,4 and group three modules 5,6,7. The concept behind this architecture is that it is not convenient to pause those modules independently as it could create jams between them (e.g. objects accumulating on conveyors)

2.2.2 Module 1

The POUs related to Module 1 follow the same structure as those in the subsequent modules or module sets. Each module contains a pair of POUs, one in GRAFCET (SFC) and one in Ladder Logic (LD), dedicated to handling the HMI controls for ON/OFF, Pause, and Reset commands, as well as the status of indicator panels and beacon lights, if present. Specifically, M01SEC_SFC and M01SEC_LD serve this purpose. These POUs manage higher-level control, overseeing the operation of all machine-specific POUs within the module. Within Module 1, two primary machine systems are programmed: the robot and the conveyors. The robot's operations are managed by the POUs M01ROB_SFC and M01ROB_LD. The SFC program follows a straightforward sequence: analog signals from sensors determine the robot's position along the vertical and horizontal axes, while digital inputs indicate when discrete actions such as clamping or overturning are complete. At each step, pause and emergency conditions are evaluated. If triggered, they interrupt the robot's movement and freeze it in place. This is reinforced in the LD program, where the robot's current position is saved as the new target, effectively preventing any further motion until the system resumes from the interruption. Though both Pause and Emergency procedures suspend operation, they differ in activation: Emergency can be triggered automatically (e.g., via safety barriers detecting unauthorized access) and is designed for rapid intervention, whereas Pause requires deliberate operator input. Regarding clamping, unlike typical implementations where a sustained signal must be maintained to keep clamping active, this robot uses distinct commands for clamping and unclamping. This means that even if the clamping signal is interrupted (e.g., during an emergency), the object remains secured unless the explicit unclamping command is issued. The conveyor logic is handled by M01CR1_SFC and M01CR1_LD. The conveyor program is more complex due to the need to manage sequential object transitions and potential collisions. The basic concept is as follows: when a photo sensor detects an object, the conveyor activates and continues running until another sensor confirms the object has reached a defined point. At that moment, logic evaluates whether the downstream conveyor is ready to accept the object. If so, both conveyors are activated simultaneously to allow for a smooth transition; otherwise, the upstream conveyor pauses until the path is clear. A key challenge here lies in handling multiple objects. If a second object arrives too quickly, it may trigger the sensor again before the first object has cleared the system, potentially causing duplicate transitions or collisions. This necessitates a careful programming approach that accounts for all such edge cases, ensuring that transitions are not re-triggered prematurely and no object overlaps occur. To coordinate robot and conveyor interactions, the motion of the conveyors is paused during sensitive robot operations, such as clamping, lifting,

or placing a box, to avoid interference. Similarly, when placing an empty box onto the conveyor, the system checks that the previous box has moved forward before proceeding. An additional complexity arises in the section of the conveyor handling empty boxes. This path includes a curve and a special lifting mechanism. This module detects boxes arriving on a roller conveyor and uses a set of chained perpendicular conveyors to lift and shift the box 90 degrees relative to its incoming direction. This redirection ensures the boxes are properly aligned for downstream operations.

2.2.3 Module 2

As anticipated in the modeling phase, this module is relatively simple, and the same applies from a programming perspective. As such, only a few Program Organization Units (POUs) are necessary. Both the control logic and machine functionality are handled within `M02SEC_SFC` and `M02SEC_LD`. These two POUs manage all relevant aspects, from the control panel buttons and indicator lights to the actual operation of the machine. The machine's functionality in this module is limited to basic tasks, such as switching on internal lights and activating fans—operations that do not require complex sequencing or interlocks.

2.2.4 Module 3

This, conversely to the previous, is a considerably more complex module. For the sake of clarity, it is important to follow the same hierarchical order as before, beginning with the highest-level programs: `M03SEC_SFC` and `M03SEC_LD`. These share significant similarities with their counterparts in the previous modules. Their principal role is to manage the control panel interface, buttons and indicator lights, and to handle pause and stop sequences. For instance, once the module is switched on, it cannot be turned off while it is paused or while the stop cycle is active. Additionally, until the module is not switched on, all lower-level programs are forced into their initial states and therefore not working. Proceeding to the POU `M03_ST`, it is necessary to recall the operating principle of the weighing system. The machine waits for all hoppers to be filled, then issues a reading command to acquire and store the weight measurements from each hopper's load cell. These values are then processed by a decision algorithm that determines which hoppers will discharge. The decision is based on a single criterion: identifying the combination of 3 out of 6 hoppers whose cumulative weight most closely approximates the target net weight required to fill a bag. This net weight is determined by customer specifications and, accordingly, the upstream packing line is fed with apples of appropriate size. In this case, the desired net weight is set to 1.5 kg, and the objective is to identify the best trio of hoppers to reach this target. To implement this behavior, a Structured Text POU is employed-an approach that would have been highly impractical using GRAFCET or Ladder Logic due to its algorithmic nature. Specifically, the implementation relies on arrays of `REAL`, `INT`, and `BOOL` types, enabling indexed access via `FOR` loops. The analog weight signals are stored as elements of a `REAL` array, and nested `FOR` loops are used to

evaluate all possible 3-combination subsets. Number of combinations:

$$\text{combinations} = \frac{n!}{(n-k)! \cdot k!} = \frac{6!}{(6-3)! \cdot 3!} = 20 \tag{2.3}$$

Combinations subsets:

1. $\{1, 2, 3\}$
2. $\{1, 2, 4\}$
3. $\{1, 2, 5\}$
4. $\{1, 2, 6\}$
5. $\{1, 3, 4\}$
6. $\{1, 3, 5\}$
7. $\{1, 3, 6\}$
8. $\{1, 4, 5\}$
9. $\{1, 4, 6\}$
10. $\{1, 5, 6\}$
11. $\{2, 3, 4\}$
12. $\{2, 3, 5\}$
13. $\{2, 3, 6\}$
14. $\{2, 4, 5\}$
15. $\{2, 4, 6\}$
16. $\{2, 5, 6\}$
17. $\{3, 4, 5\}$
18. $\{3, 4, 6\}$
19. $\{3, 5, 6\}$
20. $\{4, 5, 6\}$

Before proceeding with any summation, a validity check is performed to ensure that all measured weights are greater than zero. If all values pass this check, the algorithm computes the sum of each combination and compares it to the best (i.e., closest) result found so far. If a closer match is identified, the corresponding indices of the selected hoppers are stored. This process continues until all combinations have been evaluated. Once the optimal trio has been determined, the discharge command (`DSCx`) is issued to the selected hoppers, and the system resets the stored 'closest value' to avoid affecting the next cycle. An alternative operating mode is used during the stop cycle. In this mode, the system cannot wait for all hoppers to be full-doing so could cause jams. Instead, the logic checks each hopper's sensor, and the first three hoppers containing apples are immediately instructed to discharge. This functionality is crucial to prevent product stagnation inside the system. It is important to note that the above explanation describes only the decision-making algorithm. The physical operation of the machine is implemented in `MO3WGT_SFC` and `MO3WGT_LD`. Here, the logic is replicated for each weighing station in the system as parallel branches. The operational core lies in the SFC; while the LD is used exclusively for timing management. Upon startup, the conveyor advance command in the singulator is activated. At this stage, flap 1 is opened, and flap 2-located on the weighing hopper-remains closed. If flap 1 is open, flap 2 is closed, and the presence sensor detects apples for more than 0.3 s, the hopper is considered full. In the next step, flap 1 closes and the conveyor stops. Once flap 1 is fully closed, a weighing command is issued for 0.3 s. Subsequently, the station waits for all other stations to complete the weighing process. The decision algorithm then determines which hoppers must discharge and sends the `DSCx` command accordingly. If a given station is not selected to discharge, it remains in a waiting state until the next cycle. If selected, flap 2 opens, and once the hopper is detected as empty, the program resets to the initial step.

2.2.5 Module 4

Now that the operational logic of the weighing stations has been thoroughly explained, the subsequent stage concerns the programming of the bag filling station-specifically,

M04BGF_SFC and M04BGF_LD. Beginning with the Sequential Function Chart (SFC), the program is structured into parallel branches, each responsible for a specific subsystem behavior. The first branch governs the motion of the large flap, which is initially positioned in its upper orientation to receive apples discharged from the hoppers. The flap is triggered to rotate downward when any of the following steps becomes active: `M03WGT_SFC.S14.x`, `M03WGT_SFC.S24.x`, `M03WGT_SFC.S34.x`, `M03WGT_SFC.S44.x`, `M03WGT_SFC.S54.x`, or `M03WGT_SFC.S64.x`. Upon activation, the flap descends to guide the apples onto the flap conveyor. Once the discharge is complete, it returns to its upper position and awaits the next unloading cycle. The second branch manages the bag film unrolling process. This mechanism relies on analog input from a color sensor, which detects specific reflective patterns-such as the black registration marks-on the film, signifying the presence of a new bag. Upon detection, the unrolling operation halts, and the system waits until the bag is completely filled. Once filling is confirmed, a signal is issued to resume the unrolling sequence. The belts responsible for advancing the empty bags and transporting the filled ones operate in synchronized motion, with the color sensor serving as the master pacing element. A third branch continuously verifies that the color sensor remains operational and properly aligned. The corresponding Ladder Logic POU is responsible for actuating the flapper, bag advancement belts, and the unrolling mechanism. A noteworthy detail concerns the bag unrolling actuator coil, `GVL.QM04UNR1`, which operates as a set-reset coil. It is set each time the second SFC branch restarts from its initial step and is reset upon closure of the contact `GVL.IM04ISA1`, which corresponds to the inductive sensor integrated into the film tensioning system. This tensioning system incorporates a spring that maintains tension in the bag film. However, if the spring reaches its maximum extension while the roll continues to unwind, the film may lose tension, potentially causing mechanical jamming. To mitigate this risk, a pneumatic braking mechanism is installed on the roll drum of the film. When the inductive sensor detects the spring's maximum extension, it triggers the brake and resets the unrolling coil. Additionally, the program includes logic for controlling the elevation platform beneath the bags. This platform ascends during the initial filling phase to minimize apples drop height, thereby reducing the risk of fruit bruising. Finally, several supplementary commands are present only for visualization purposes within the simulation environment.

2.2.6 Module 5

Module 5 concerns the programming of the robotic arm and the conveyor belts feeding the packing station. As with the previous modules, its structure is characterized by a pair of supervisory POUs overseeing the subordinate logic layers: M05SEC_SFC and M05SEC_LD. Analogously to the previous cases, these POUs manage the control panel interface-handling button input, indicator lights, and orchestrating pause and stop cycles. At a lower hierarchical level, several additional POUs are employed. The most complex among them are those responsible for the robotic arm, namely M05ARM_SFC and M05ARM_LD. These are particularly extensive due to the architecture of the robotic system, which consists of six independently actuated arms. At each control step, the position of each arm-defined by its joint angle-must be updated from the `LD` POU. Simultaneously, the `SFC` POU verifies whether each joint has reached its respective target angle. While

conceptually straightforward, this joint-by-joint management renders the code long and repetitive. Through the coordinated control of these joint variables, the manipulator's pose in the task space is precisely managed. The operational logic is as follows: initially, the robotic arm moves to a waiting position located directly above the expected stopping point of an incoming bag. When a bag is detected on the conveyor belt, the arm descends vertically, and the vacuum suction cups are activated to grasp the plastic bag filled with apples. Once the suction sensor confirms the bag has been secured, the arm moves to its next target-positioned above an open box waiting to be filled. Upon arrival, the joint positions are updated once more, and the robot descends vertically to deposit the bag. The vacuum is deactivated, and the arm retracts vertically and returns to the initial waiting position above the conveyor. This process is repeated for the second and third bags, sequentially placing them in the remaining positions within the same box. Once the final bag is placed, the `SFC` governing the arm transitions back to the initial step, ready for a new cycle. Regarding emergency stops and pause events, the same safety logic applied in Module 1 is reused. In both cases, the robot's motion is inhibited by assigning the current joint angles as target values. This ensures the robot freezes in place, preserving its position until normal operation can resume safely. To control the feeding and output conveyor belts, an additional POU has been implemented: `M05CBT_SFC`. The conveyor belts transporting bags are divided into two functional groups: the first consists of the belts nearby the robotic arm, and the second includes the belts from the bag filler (Module 4) to the handoff point just before the robotic arm's zone. Similarly, the conveyor system handling empty and filled boxes is divided into three segments: from the box loading area to the spiral conveyor, from the spiral conveyor to the positioner, and from the positioner to the robotic arm's loading station. Each belt is configured to advance until a new box is detected by a sensor. However, for smooth transitions between sections, adjacent conveyors are synchronized-briefly running together when a box transfers from one segment to the next. The same POU also handles the box positioner. Its programming is relatively straightforward: the horizontal guide arm remains lowered by default. When a new box is detected-arriving from the spiral conveyor-the side flap activates to push the box into its designated position. The flap then retracts, and the positioner raises, allowing the box to continue on the conveyor. Once the box reaches the robotic arm's station, it is detected and the conveyor halts, stopping the box precisely in place. Finally, after the robotic arm signals that the third and final bag has been placed in the box, the conveyor resumes. The filled box advances toward the box-closing machine. Once closed, the box exits the system through the designated ejection point, aided by minor visual cues and tracking logic.

2.2.7 Module 6

Modules 6 and 7 share a common supervisory logic structure, similar to what is adopted for Modules 3 and 4. At the top level, the POUs `M06SEC_SFC` and `M06SEC_LD` are responsible for managing the module's global control logic. This includes ON/OFF transitions, stop and pause cycles, indicator light handling, and forcing of subordinate POUs through

the SFCInit function. The core logic of Module 6 concerns the palletizing process, handled primarily by the POU M06PAL_SFC. This SFC is notably complex and features multiple parallel branches, each managing different subsystems of the palletizing station. The first branch controls the feeding of empty pallets to the palletizing platform. This involves coordinating the roller conveyors leading up to the palletizer and those integrated within the pallet lift. Photo sensors are employed along the path to detect the position of the pallets and enable smooth, collision-free transit. The second branch manages the positioning and stacking of box layers onto the pallet. This includes:

- Controlling the vertical movement of the pallet lift to accommodate new layers.
- Operating mechanical gates used to temporarily hold and then release box groups onto the pallet.
- Counting the number of completed layers to determine when the pallet is full.

A third parallel branch coordinates the grouping of boxes arriving from Module 5 via a belt conveyor. Boxes are counted and grouped in pairs. Once a pair is ready, it is pushed onto a positioning grid. When two pairs are present (i.e., a complete layer), the grid is opened by the previous branch, allowing the boxes to drop onto the pallet. The pallet lift then lowers slightly to accommodate for the layer, and the gates close in preparation for the next cycle. Once the predefined number of layers is reached, the lift lowers entirely, completing the pallet. At this point, visual effects are triggered, after which the final branch is activated to transport the completed pallet to the next module. An additional branch, consisting of a single step, is reserved for stop cycle scenarios. If stop is triggered and the available apples are insufficient to complete the current pallet, this branch bypasses the layering logic and immediately initiates pallet output. The associated LD program, M06PAL_LD, contains the motion control coils for the pallet lift, conveyor belts, and visibility toggles (e.g., generation and destruction instructions). Furthermore, counters are implemented to track:

- The number of boxes arriving on the conveyor belt.
- The number of layers successfully stacked on each pallet.

2.2.8 Module 7

Module 7 concludes the production cycle and involves the pallet wrapping station. The main POUs for this module are M07TYG_SFC and M07TYG_LD. The logic in this module is comparatively simpler. The SFC program consists of two main branches, the first handles pallet detection and automatic deletion. Once a wrapped pallet is detected exiting the system, a timer starts. After a predefined delay-simulating manual collection by a forklift-the pallet is removed from the system. The second branch manages the actual wrapping process. Initially, the machine prepares to receive the incoming pallet by lowering the wrapping platform. Once the pallet is in place, wrapping begins. This is triggered by the LD POU and continues as follows: after each full rotation of the wrapping film around the pallet, an analog sensor reads the angular position of the rotating arm. When a full turn is detected, the platform is incrementally lifted to wrap the next section at a higher level. This vertical lift-and-wrap cycle is repeated-not as a loop but through sequential steps-until the entire pallet is covered. At the end of the process, the platform lifts fully and waits for the next pallet. Meanwhile, the finished pallet is transported via roller

conveyors to the final collection zone, where it remains until removed.

2.2.9 Labeling machines

In this subsection, the functionality of the labelling machines is clarified. These machines, although relatively simple in operation, play a fundamental role in ensuring product traceability by applying identification labels containing essential product information onto both individual bags and final boxes. Due to the critical nature of this functionality within the automation workflow, a dedicated discussion is warranted. The first labelling machine is positioned immediately downstream of the bag filler machine, corresponding to module 4. The relevant control logic can be found in the POU `M04BGF_LD`. In the final section of this Ladder Diagram, the presence of a filled bag is detected, which closes a contact and initiates the extension of the labelling arm. Once the arm reaches its fully extended position, a `TOF` (Time-Off Delay) timer with a preset of 1 second is triggered to activate the label application process. Concurrently, a `TON` (Time-On Delay) timer is used to retract the labelling arm after 0.5 seconds. This sequence ensures precise placement of the label on the bag and prevents misalignment or label displacement due to premature movement. The second labelling machine is located after module 5, responsible for applying labels to the boxes. The logic governing this station is distributed across the POUs `M05ARM_LD` and `M05CBT_SFC`. In the Ladder Diagram `M05ARM_LD`, a photo sensor embedded within the labeller detects the presence of a box. This detection event triggers a `TON` timer, which in turn activates a coil responsible for starting the labelling process. Simultaneously, a `CTU` (Count-Up) counter is incremented, effectively tracking the number of boxes processed and labelled. Furthermore, the same photo sensor signal is used to momentarily halt the output conveyor of Module 5. This short delay allows for proper adhesion of the label onto the box surface, eliminating the risk of misplacement due to motion. This integration ensures not only traceability, but also the mechanical reliability of the labelling phase.

Chapter 3

Results and Validation

3.1 Simplifications and adaptations

As outlined in the project description, adjustments and simplifications have been introduced to compensate for missing data, given that the simulation does not coordinate with a real physical twin. The work originated as an exploratory study aimed at evaluating the feasibility of replicating an industrial process and determining the achievable level of detail within a simulated environment. Moreover, it is required to take into account the performances of the machine where the simulation and PLC are run.

Focusing on modules 1 and 2, the roller conveyor, used in actual processes, responsible for extracting apples from the water must be modeled as a flap conveyor within the simulation environment. As discussed previously, this requirement arises primarily from the geometric simplifications imposed by the software on imported objects. Specifically, apples are approximated as spherical bodies, while roller conveyors are represented with a flat upper surface. This geometric simplification introduces several limitations. Apples, due to their spherical shape, tend to roll freely on the flat surface of roller conveyors. Consequently, it becomes unfeasible to simulate their upward movement on an inclined plane, even with minimal slope angles. Furthermore, apples continuously roll along the conveyor. This behavior poses a significant issue during system pauses or emergency stops, as the apples persist in their motion despite the conveyor having ceased operation. When referring to roller conveyors in this context, it is important to specify that the system in question is known as a feeding roller elevator. This specialized conveyor comprises PVC rollers designed specifically for the inspection and selection of fruit. Unlike conventional roller conveyors where items move by rolling over a static surface, in this configuration, the rollers themselves are in motion. Each roller is mechanically linked to a chain driven by electric motors, enabling synchronized translation. Additionally, the PVC rollers are equipped with end-mounted gears that engage with a specially designed rack. This rack-and-pinion mechanism induces rotation of rollers about their axes as they are pulled along by the chain. This coordinated motion causes the apples to rotate and twist, effectively exposing all sides of each fruit to the operators responsible for visual inspection.

Regarding the weighing machine, as previously described, it represents a simplified

model of the equipment typically used in industrial applications. In this simulation, only six weighing units have been implemented, whereas actual machines generally feature between nine and fourteen units, each equipped with its corresponding conveyor, flapper, hopper, and load cell. This dual feeding configuration increases efficiency, as the system simultaneously feeds two bag fillers. It is a convenient solution because the weighing and discharge operations are significantly faster than the bag filling process. The latter is constrained by the delicate nature of apples, which are highly susceptible to bruising, thereby limiting the permissible conveyor speed. This discrepancy creates a bottleneck that can be mitigated by increasing the number of weighing units and enabling parallel feeding to a couple of bag fillers. Furthermore, the operational logic of the weighing machine can be optimized by allowing more than three hoppers to open simultaneously, and by making the number of these active hoppers variable rather than fixed. For instance, if a hopper contains only a single apple with insufficient weight, the system can combine it with additional hoppers to reach the target weight. Similarly, if hoppers contain an above-average quantity of fruit, the logic can adapt accordingly and open less hoppers. This enhancement is achievable through modifications in the structured text programming and can yield substantial performance improvements.

The simulation of the weight reading process presents certain challenges. These weighing machines are not designed to handle large apples, as such fruits may obstruct the mechanism and are typically packaged in trays rather than bags. It is standard practice to process apples ranging from 40 mm to 85 mm in diameter. For the purpose of this simulation, a nominal apple size of 75 mm was assumed, corresponding to an average weight of approximately 160-170 grams. Consequently, with three apples per hopper, each hopper is expected to contain roughly 0.5 kg of fruit. Opening three hoppers simultaneously would therefore yield a total discharge of approximately 1.5 kg, suitable for bag filling. This represents just one possible operational configuration. Alternatively, the system could be designed to allow only one apple per hopper. This approach simplifies the process of achieving the target weight, although it necessitates either a greater number of hoppers or an increased frequency of weighing cycles.

Concerning the labeling machine implemented in the simulation, it is a simplified representation of the systems used in industrial environments. Real-world labeling machines typically consist of a label roll, a printing unit, and a dedicated label application mechanism. These mechanisms generally fall into three categories: linear, rotary, and blow-type. The linear system employs a pneumatic arm that presses the label onto the top surface of the box. The rotary system utilizes a rotating arm to apply the label to the side of the box. The blow-type system, on the other hand, uses a jet of compressed air to affix the label onto the surface of the object. In the simulation, a hybrid approach combining rotary and blow-type mechanisms has been adopted for bag labeling. This choice was driven by spatial constraints and the irregular geometry of the bag surfaces. For box labeling, a simplified blow-type system has been used, which is sufficient given the regularity of the box surfaces.

Module 5 warrants particular attention, as it represents an experimental stage within the process. In industrial practice, this step is typically performed manually. However, to maintain consistency with the automation-oriented framework of the thesis, the module

has been re-engineered. The concept of employing a robotic manipulator stems from its widespread use in operations such as palletizing, which closely resembles the box-filling task addressed here. Additionally, other robotic systems-such as SCARA robots or two-axis configurations equipped with advanced end effectors-are increasingly being adopted for tray-packing applications. Drawing inspiration from these examples, the integration of a robotic solution for this stage was proposed. The primary challenge lies in handling plastic bags filled with apples, which is inherently complex due to the delicate and deformable nature of the contents. To address this, the use of appropriately dimensioned suction cups, optimized for vacuum pressure and surface contact, is considered essential. Another critical aspect involves the robot's compliance and stiffness characteristics, which must be carefully tuned to prevent damage to the apples during manipulation. Although this topic extends beyond the scope of the current thesis, it represents both a necessary simplification and a promising foundation for future research.

With regard to the palletizing module, it is important to clarify that the simulation includes only two pallet layers. These layers serve a symbolic purpose, as actual industrial pallets have more layers. The simplification was necessary to reduce simulation time and computational load. In a real-world scenario, both the palletizer and the pallets height would be scaled up accordingly. Nonetheless, extending the system to accommodate additional layers would be straightforward from a control perspective. It would require the inclusion of further steps in the PLC program, allowing for the sequential addition of layers without altering the fundamental logic of the process.

The final machine in Module 7 has been adapted to suit the simulation context. In this stage, the machine performs a wrapping operation around the pallet. In actual industrial processes, however, pallet securing is typically achieved through the insertion of angular cardboard reinforcements at the four corners, followed by the application of plastic strapping to prevent load displacement during handling and transport. For simulation purposes, the wrapping method was selected primarily for its visual clarity, allowing easier verification of the machine's functionality. This approach enhances the visibility of the securing process within the virtual environment, despite differing from standard industrial practices.

3.2 Simulation and Testing Methodology

To initiate the simulation process, a sequence of preparatory steps must be executed to ensure a successful connection between *Machine Simulator 4* and *Codesys*. Initially, launch the software environment. Once *Codesys* is operational, proceed to open the designated project. Before connection, it is essential that all variables intended for communication via *OPC UA* are declared within a `Global Variable List (GVL)`. Furthermore, the `Link Always` option located under the `Build` section in the `GVL Properties` of the `GVL` must be activated. Additionally, the `Symbol Configuration` object, configured for *OPC UA* compatibility, must be incorporated into the `Application` tree to facilitate variable management. Subsequently, all functional POUs must be assigned to the following path: `\Project_Name\Device\PLC_Logic\Application\Task_Configuration\Main_Task`, ensuring their inclusion in the SoftPLC download process. Once configuration is complete,

initiate both the *Gateway* and the *SoftPLC* from the *System Tray* in Windows desktop. At this stage, perform a device scan from the `Device` object within *Codesys* to establish a connection with the *SoftPLC*, treating it as a standard device. Following this, execute the `Login` operation to deploy the program onto the *SoftPLC*. During the initial login, user credentials may be required; these can be configured for future sessions or disabled via the `device` menu.

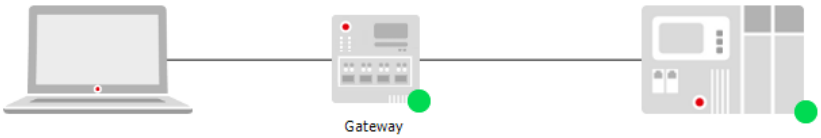


Figure 3.1: Enabled connection through the gateway.

Next, open the machine interface in *Machine Simulator 4* and select `OPC_UA_Driver` under the `Driver Name` field in the `IO Driver` menu. Within the Configuration panel, set the `Type Server` to `opc.tcp://localhost:4840` to initiate the connection. Upon successful variable retrieval, the connection is established, and the `GVL` folder becomes accessible under the following path: `\Objects\Device_Set\CODESYS_Control_Win_V3_x64\Resources\Application\GlobalVars\GVL`. Once the variables are available, they can be assigned by dragging and dropping them into one of the four designated areas at the bottom of the interface, corresponding to PLC Digital Outputs, PLC Digital Inputs, PLC Analog Outputs, and PLC Analog Inputs. At this point, the configuration can be saved and the interface closed.

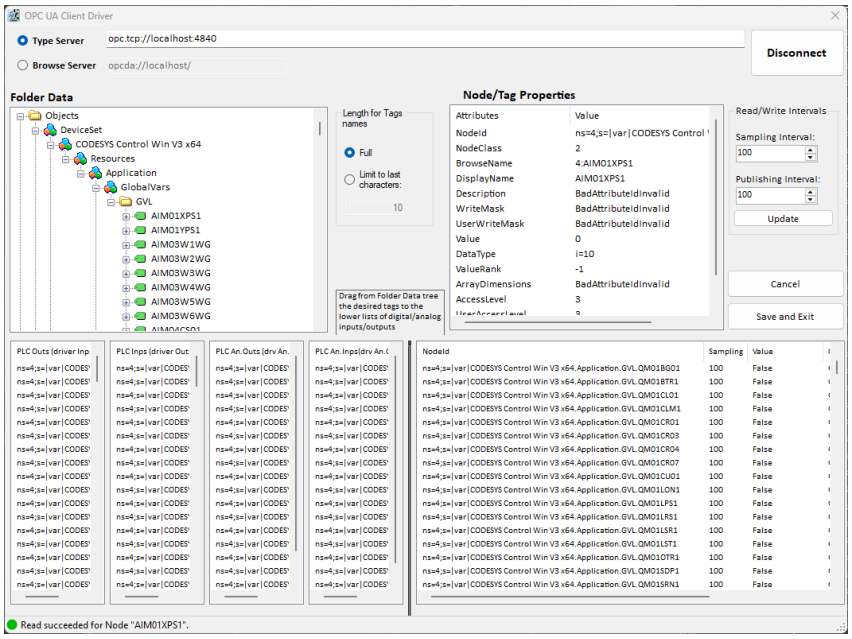


Figure 3.2: OPC UA Configuration from Machine Simulator 4.

The Inputs and Outputs will now be displayed on the left and right sides of the screen, respectively, and can be linked to their corresponding elements within *Machine Simulator*. It is crucial to remember that an output from the simulation serves as an input to the driver, and vice versa. After all connections are established, the configuration can be saved as a custom setup, which may be reloaded in subsequent sessions and edited as needed, since it is stored in a `.txt` file format. To finalize and start the simulation, simply activate the driver and exit, then press the play button within *Codesys*.
As final remark, it is worth noting that the *SoftPLC* of *Codesys*, by default only allows the handling of 100 variables in input and 100 variables in output. This turns out to be a limitation in big projects. Therefore a solution can be adopted. The folder `CODESYSControl.cfg` in the runtime must be found and modified adding the following lines of code:

```
[CmpOPCUAServer]
NetworkPort=4840 ;Configure the port used by the OPC UA server.
   Default:  4840
NetworkAdapter=eth0 ;Configure the name network adapter to be used by the
   OPC UA server.  By default OPC UA will bind to ANY adress.
UseLoopback=1 ; 1 is now default
MaxNodesPerRead=300
MaxNodesPerHistoryReadData=300
MaxNodesPerHistoryReadEvents=300
MaxNodesPerWrite=300
MaxNodesPerHistoryUpdateData=300
MaxNodesPerMethodCall=300
MaxNodesPerBrowse=300
MaxNodesPerRegisterNodes=300
MaxNodesPerTranslateBrowsePathsToNodeIds=300
MaxNodesPerNodeManagement=300
```

In case the 64-bits version is used, it is possible to find the folder at the following path:

```
C:\Windows\System32\config\systemprofile\AppData\Roaming\
   CODESYS\CODESYSControlWinV3x64\AE3C854C
```

While for the 32-bits version the path is:

```
C:\Windows\SysWOW64\config\systemprofile\AppData\Roaming\
   CODESYS\CODESYSControlWinV3\CABE3855
```

Adjusting these parameters the connection will be established without error flags.

3.3 Discussion of the Results

The modeling of the Digital Twin is directed toward the integration of PLC programming, ultimately creating a virtual industrial processing line. This objective is pursued through an initial observation phase, followed by an analysis of the sensors and actuators utilized in actual industrial plants. This approach enabled a comprehensive understanding of the core operational mechanisms, facilitating their accurate replication within a virtual environment.

The processing line is first modeled, adding control mechanisms and animations to simulate its operation. Functional variables are then assigned to corresponding control elements, enabling dynamic interaction within the simulation environment. Subsequently, the industrial system is programmed using PLC programming software, selecting the most suitable programming language for the application. This phase is conducted with consideration of real-world industrial requirements, including standard operating conditions, emergency protocols, stop and pause cycles, and reset procedures. The finalized program is then deployed to a SoftPLC, which governs the entire virtual industrial plant. The anticipated objective of the work is to build a Digital Twin for PLC program implementation. However, a direct outcome is the demonstration of feasibility of the simulation of an industrial environment. One of the most significant results is the establishment of a foundation for future development of Digital Twins representing entire real-world plants, rather than isolated industrial processes. Potential applications include efficiency monitoring, PLC testing, layout optimization, predictive maintenance, training of maintenance personnel, fault detection, and system development. The initial goal is to achieve a fully operational virtual industrial plant programmed using standard software tools. Additionally, it is expected that both the author and readers gain practical insights and technical knowledge from this work. The decision to avoid focusing on a single stage of the industrial process is intentional, as it highlights the scalability of the Digital Twin concept. A primary concern at the outset of the project was the computational load required to simultaneously run the simulation and the PLC on the same machine-an issue specific to this implementation. As a matter of fact, powerful devices are needed as principles of Industry 4.0 are employed, since more and more data need to be handled. As previously mentioned, the modeling process began with the observation of real plants and video footage to develop a solid understanding of both the process and the product being handled. Based on these observations, the modeling approach was structured around replicating the observed mechanisms and identifying the variables necessary for machine control. Each machine's function was first studied and then recreated within the virtual environment. The programming phase, while generally faster due to prior knowledge of machine behavior, is occasionally slowed by the need to revisit and modify the Digital Twin to ensure proper control. *SolidWorks* was used to model custom components, *Machine Simulator 4* was employed for Digital Twin assembly, and *Codesys* was selected for PLC programming and SoftPLC integration via `OPC UA` communication. Due to computational limitations, the simulation is executed with reduced visual fidelity. This compromise is necessary because the PLC software, Digital Twin, and video recording are all running concurrently on the same machine, resulting in performance lags and erratic behavior of machines and 3D objects. With this context established, a detailed analysis of the simulation can begin (Link to the video). Initially, the machine is activated using the general ON/OFF switch, followed by powering on each module via individual switches. Holding the start push button initiates the machine after a predefined delay, at which point the start indicator light is illuminated. The operation of each module proceeds as described in earlier chapters, and it is now essential to verify that the simulation behaves accordingly. The robotic arm responsible for box handling performs with satisfactory accuracy. The tipping mechanism operates smoothly. However, the placement of

empty boxes on the outlet conveyor is occasionally disrupted due to misalignment, likely caused by lag. Specifically, the delay between the photo sensor detecting a box and the motor deactivation allows the box to travel beyond its intended position, complicating the robot's repositioning task. Module 2 functions without issues. Module 3 exhibits a lag-related problem: although programmed to dispense three apples per hopper, delays in the animation of the flapper mechanism sometimes result in more than three apples being deposited. Additionally, when multiple apples accumulate in the hopper, jamming may occur, preventing proper unloading. Module 4 also experiences misalignment due to lag, affecting the flapper mechanism and causing some fruit to miss the bag. From this point onward, the system operates smoothly until Module 5, where the robotic arm occasionally misplaces bags within boxes. However, this module focuses on demonstrating the machine's operational concept, and in real-world applications, precision is managed by dedicated controllers. The palletizer functions flawlessly, both in box placement and layer stacking. The final module presents a minor issue with the alignment of the plastic film, caused by delayed command execution. This overview clearly indicates that the primary challenges in the simulation stem from lag and visual performance limitations. Nonetheless, the core principles of PLC programming and machine control are accurately represented. Even the most intricate operations-such as bag filling and sealing, which require precise timing and synchronization between bag unrolling, conveyor activation, and sealing-are faithfully modeled to reflect real machine behavior.

Regarding the software tools, *Machine Simulator 4* proves to be highly versatile. Its communication with *Codesys* is reliable and consistently fast across simulation runs, depending on program complexity. Through `OPC UA` communication, both digital and analog variables are easily managed. Notably, analog variables are transmitted directly as `REAL` values, eliminating the need for conversion as required by other communication protocols. The Digital Twin developed in this project demonstrates significant usability. It supports unrestricted navigation within the virtual environment, offering both first-person and third-person perspectives, as well as aerial views. Additionally, integration with a VR module is available, enabling immersive exploration in virtual reality. These features make it an ideal candidate for projects of this nature, with strong potential in terms of scalability and adaptability. However there are drawbacks in the software used for modeling and simulation, as one of the main challenges encountered during modeling is the placement of reference frames during machine assembly. As discussed in the Digital Twin modeling phase, this issue is recurrent in *Machine Simulator 4* and can lead to inconsistencies in rotation and translation directions, as well as component orientation-particularly when configuring controllers for machine animation. Another limitation is the default cap on readable and writable variables, which, although initially restrictive, can be easily resolved by increasing the limit as explained in previous chapters. Aside from these aspects, the modeling process is intuitive and allows for the construction of virtually any industrial machinery or environment with minimal effort once the user becomes familiar with the interface.

The passages covered in this work pertain to the initial stages of Digital Twin development, which, in real-world applications, requires more work to make it fully functional. However, once completed, a Digital Twin offers substantial advantages in areas

such as maintenance, layout planning, and performance optimization. The promising results obtained here suggest transformative potential in the management of industrial processes, contributing to enhanced safety and cost efficiency. Predictive maintenance, for instance, enables the reduction of spare parts inventory costs and mitigates the risk of component obsolescence. By forecasting wear and tear, components can be ordered only when necessary. This approach also minimizes unplanned maintenance by identifying parts nearing the end of their life-cycle, thereby preventing operational disruptions and improving overall performance. Layout planning benefits from the ability to visualize spatial arrangements in 3D, offering a realistic perception of space utilization. This allows for design modifications without incurring additional costs, unlike changes made during physical plant assembly. Optimization tools can be applied directly to the virtual model to eliminate unused areas or prevent insufficient space allocation for maintenance and operational tasks-particularly in zones requiring forklift maneuverability. Performance monitoring facilitates the recording of machine up-time and downtime, enabling predictive analysis and recommendations to reduce stoppages and enhance efficiency. This, in turn, contributes to increased competitiveness of the final product in the marketplace.

In conclusion, the automation of production plants presents numerous advantages. Although it necessitates a shift in process management strategies, such changes are essential to fully leverage the benefits offered by these advanced technologies.

Chapter 4

Conclusions and Future Developments

4.1 Recommendations for Future Research

Given the complexity and feature-rich nature of the project, there remains considerable scope for future enhancements and development. A primary area for improvement is the design of a more advanced HMI, aimed at simplifying machine interaction and monitoring. Consolidating all control elements-such as buttons and switches-into a unified panel would streamline operations, while retaining emergency controls adjacent to each machine for safety. This configuration would facilitate easier activation, deactivation, pausing, and resetting of modules. Furthermore, implementing a visualization interface to display metrics such as the number of incoming boxes, production rates of bags and boxes, and pallet output would significantly enhance monitoring capabilities. Leveraging this data, along with machine up-time and downtime logs, would enable the calculation of key performance indicators (KPIs), offering insights into operational efficiency. An interactive panel should also be integrated to allow users to specify the desired weight of apples per bag, configure the box layout on pallets, and define the number of box layers per pallet. In summary, substantial customization potential remains to be explored.
In conjunction with these improvements, the development of motor control systems for belt and roller conveyor actuators would contribute to a more realistic simulation. This would allow dynamic adjustment of production rates through conveyor speed modulation. Further refinement could be applied to the robotic manipulator, where precise control over position, velocity, and acceleration is critical. Additionally, regulating wrist compliance is essential due to the delicate nature of the fruit-filled bags.
From an efficiency standpoint, incorporating an additional weighing station equipped with dual bag fillers would enhance realism. Regarding the initial module, programming the robot to stack empty boxes and unstack filled ones in piles of three would accelerate the feeding process and improve energy efficiency through optimized motion planning.
As a potential extension, the integration of AGVs (Automated Guided Vehicles) could automate the transport of empty and full boxes, as well as pallets. Moreover, two automated storage systems could be developed: one for raw materials, specifically full boxes from

the presizing stage, and another for finished goods, namely pallets ready for shipment.

4.2 Conclusions

In this thesis, a Digital Twin model was developed to simulate and program an industrial process plant. The results confirmed that the model effectively replicated real-world process dynamics, offering a reliable platform for control logic validation. These findings underscore the potential of Digital Twins in industrial automation. However, achieving real-time insights and enabling proactive decision-making demands further development. Despite the encouraging outcomes, significant challenges remain-particularly in system integration and interoperability with existing infrastructure. Moreover, the incorporation of machine learning techniques is essential to enable predictive analytics and adaptive control strategies. A higher level of detail in the model is also required, ideally by inheriting PLC variables and logic to replicate machine behavior with maximum fidelity. Additionally, animations must be synchronized with actual machine cycle times, as command execution in the virtual environment may lag behind real-world performance. These and other considerations discussed throughout the thesis highlight both the substantial progress made and the extensive work still required in this field. Initially, the exploration of automation appeared straightforward; however, it proved to be considerably more complex-not only in terms of technical knowledge but also in understanding the specific process being automated; an aspect is of paramount importance. A thorough comprehension of the product and the associated process has a profound impact on the success of automation. This is where the role of a mechatronics engineer becomes critical. The continuous task involves integrating the characteristics and stages of the industrial process with available technologies-both mechanical and electronic-while consistently accounting for product quality parameters. Up to this point, a single expert can manage the workload. Beyond this stage, however, collaboration among various specialists becomes essential. These include professionals in electrical systems, computer science and communication, and mechanical design. The development process is laborious and each component must be meticulously refined. Ultimately, when the system operates as intended, the benefits and satisfaction derived from the work are substantial.

Bibliography

[1] Codesys - homepage.

[2] Factory i/o.

[3] Nirtec studio.

[4] Opc unified architecture.

[5] Solidworks.

[6] United nations, department of economic and social affairs - sustainable development.

[7] *How to Establish Digital Thread Using 3D Factory*, volume Volume 2B: Advanced Manufacturing of *ASME International Mechanical Engineering Congress and Exposition*, 11 2019.

[8] Barbara Rita Barricelli, Elena Casiraghi, and Daniela Fogli. A survey on digital twin: Definitions, characteristics, applications, and design implications. *IEEE Access*, 7:167653–167671, 2019.

[9] Itxaro Errandonea, Sergio BeltrÃ¡n, and Saioa Arrizabalaga. Digital twin for maintenance: A literature review. *Computers in Industry*, 123:103316, 2020.

[10] Food and Agriculture Organization of the United Nations. Commodities by country.

[11] Jessica S. Ortiz, Evelin K. Quishpe, Grace X. Sailema, and Nathaly S. GuamÃ¡n. Digital twin-based active learning for industrial process control and supervision in industry 4.0. *Sensors*, 25(7), 2025.

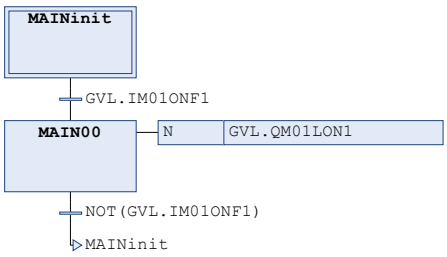
[12] Kimberley Viaron, Nathalie Julien, and Mohammed Hamzaoui. Digital Twin For Maintenance: Overview And Trends. In *ESREL 2024*, Cracow, Poland, June 2024.

Chapter 5

Annex

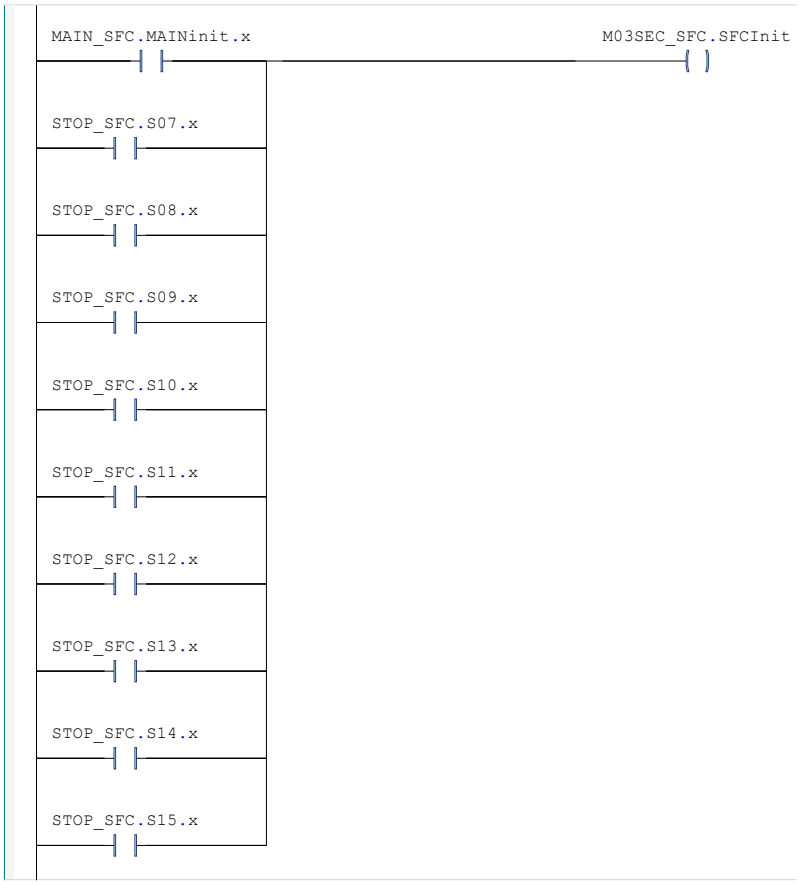
POU: MAIN_SFC

```
1       PROGRAM  MAIN_SFC
2       VAR
3       END_VAR
4
```

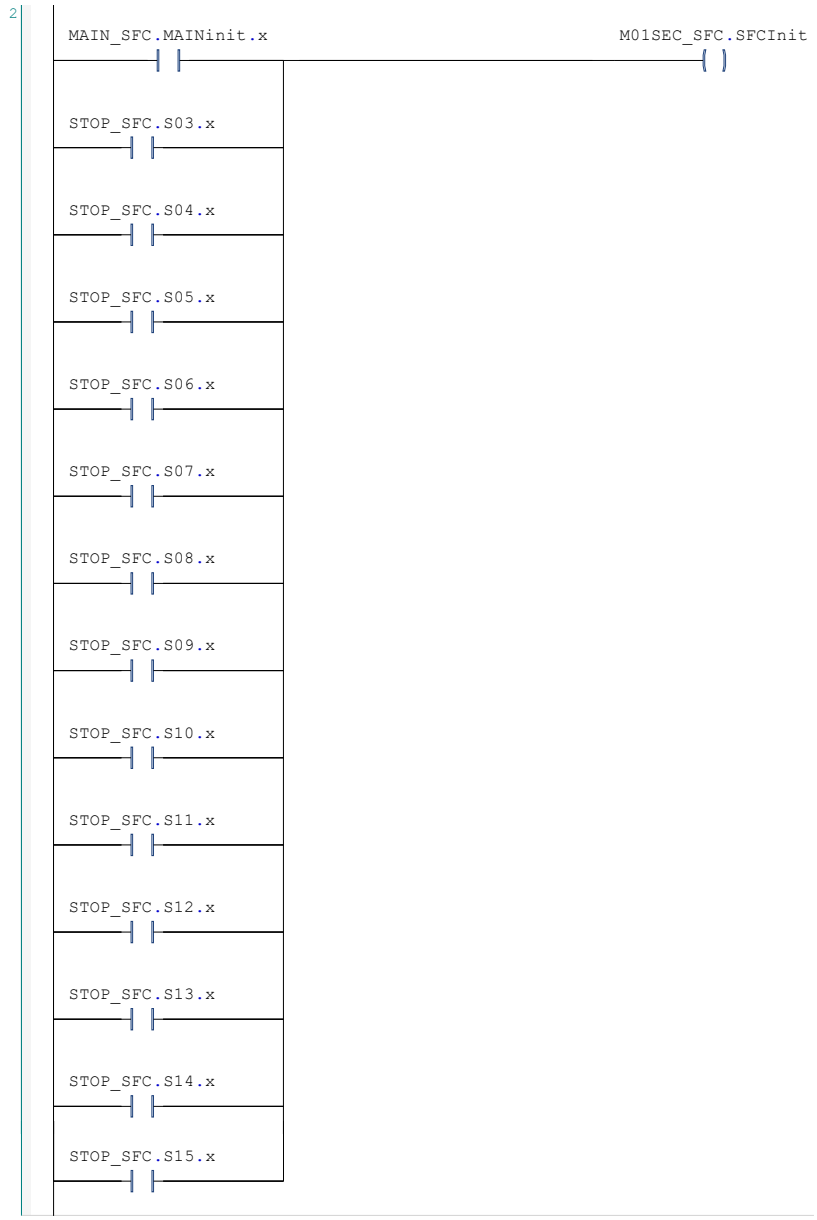


POU: MAIN_LD

```
1   PROGRAM MAIN_LD
2   VAR
3       TON_0 : TON ;
4       TON_1 : TON ;
5       TON_2 : TON ;
6       TON_3 : TON ;
7   END_VAR
8
```



POU: MAIN_LD

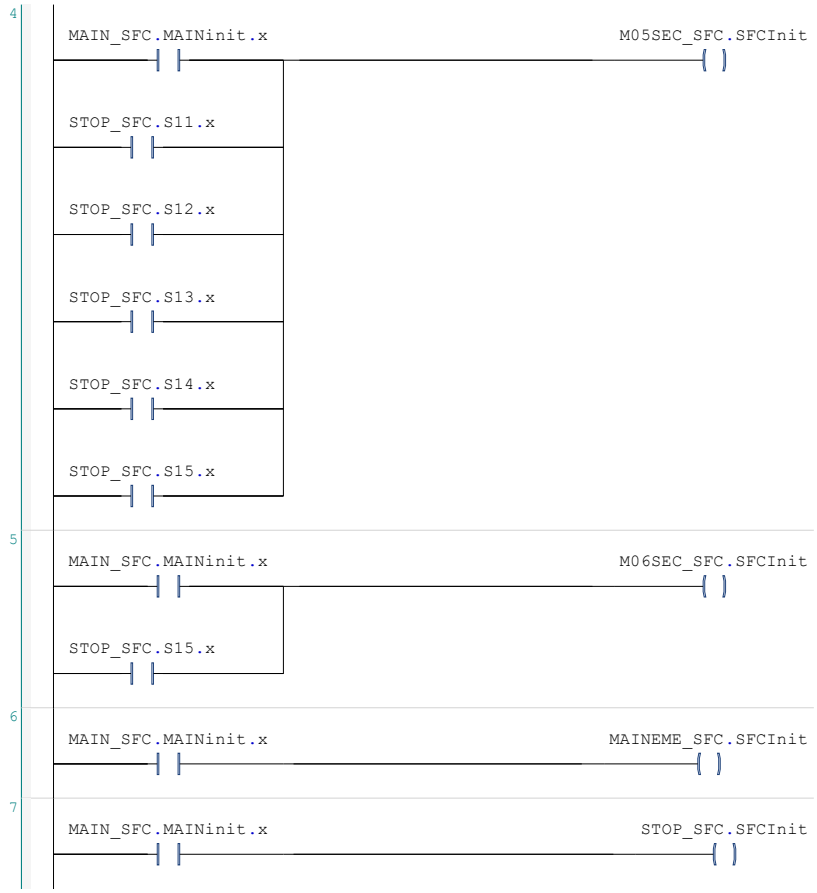

2
MAIN_SFC.MAINinit.x
M01SEC_SFC.SFCInit
STOP_SFC.S03.x
STOP_SFC.S04.x
STOP_SFC.S05.x
STOP_SFC.S06.x
STOP_SFC.S07.x
STOP_SFC.S08.x
STOP_SFC.S09.x
STOP_SFC.S10.x
STOP_SFC.S11.x
STOP_SFC.S12.x
STOP_SFC.S13.x
STOP_SFC.S14.x
STOP_SFC.S15.x

POU: MAIN_LD

ApplePackingLine10x64.project
7/31/2025 10:20 PM

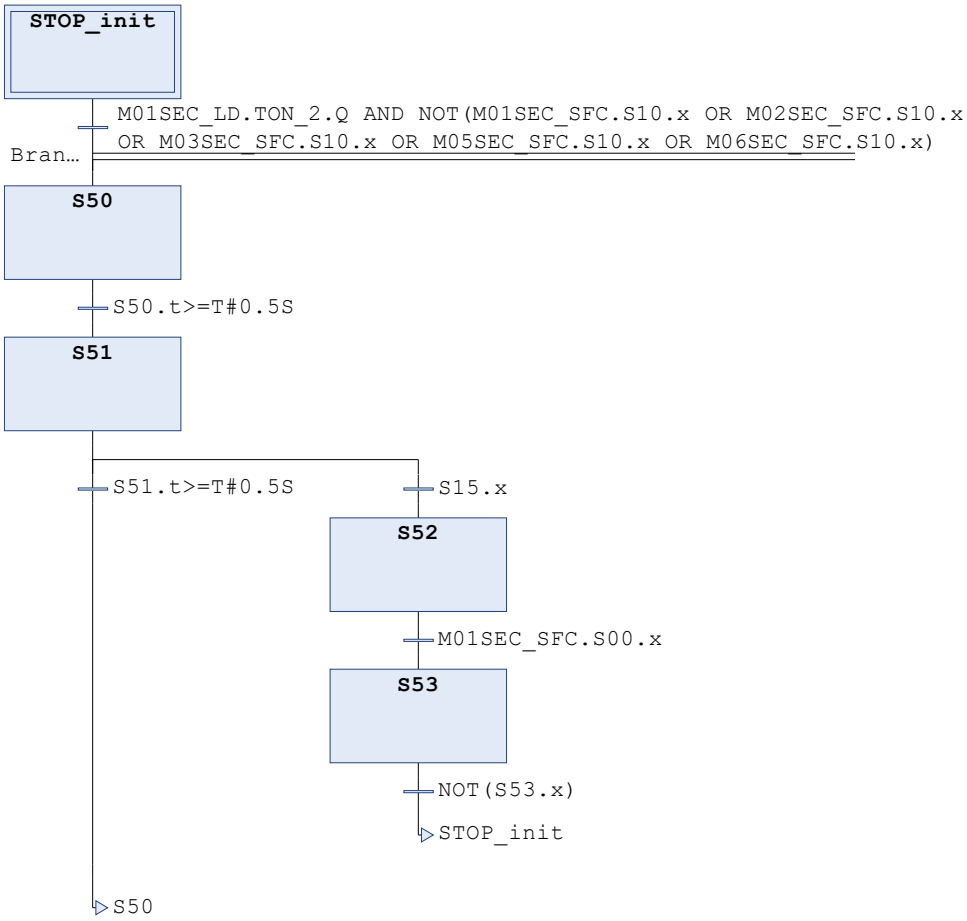
Page 3 of 4

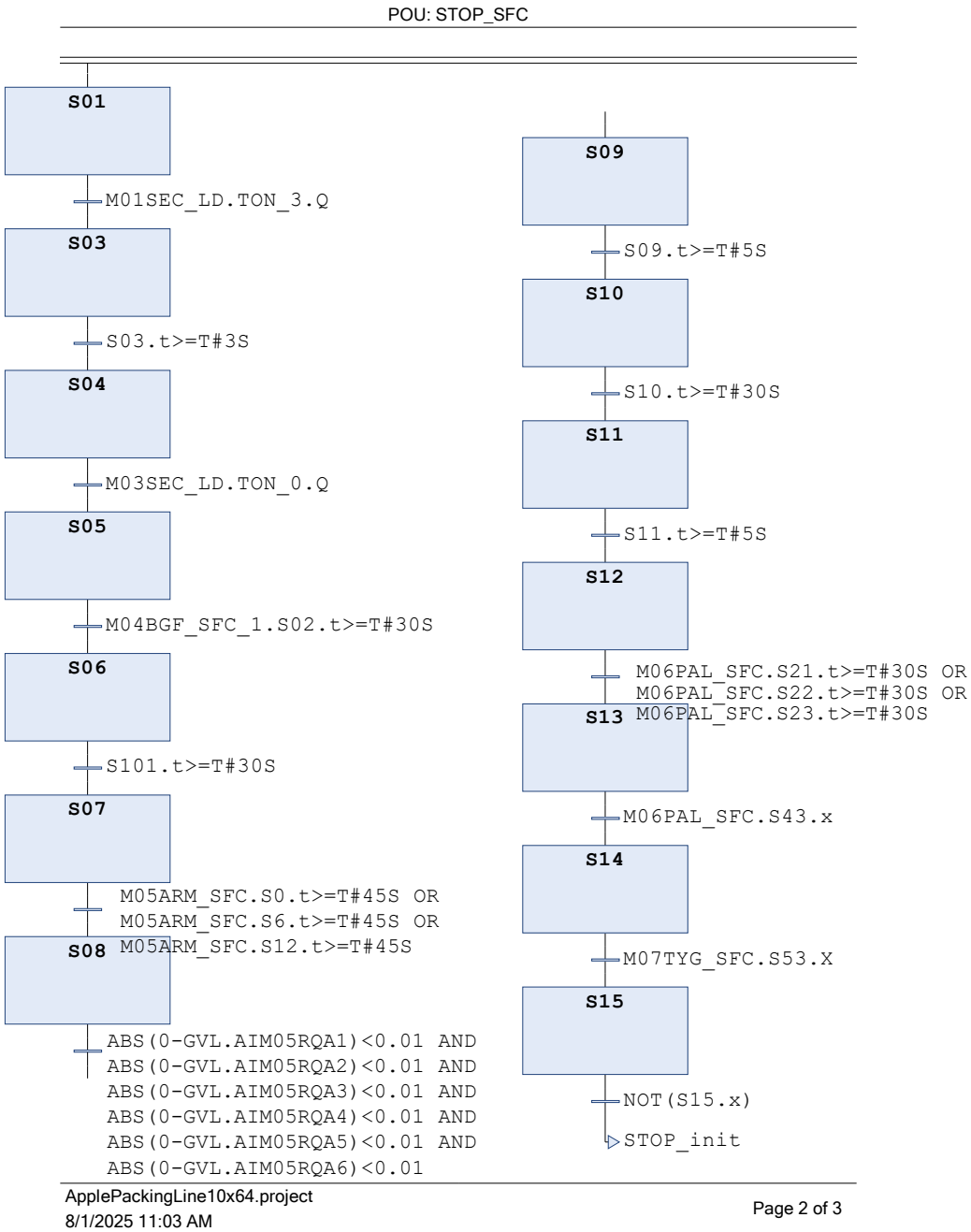
POU: MAIN_LD


4
MAIN_SFC.MAINinit.x
M05SEC_SFC.SFCInit
STOP_SFC.S11.x
STOP_SFC.S12.x
STOP_SFC.S13.x
STOP_SFC.S14.x
STOP_SFC.S15.x
5
MAIN_SFC.MAINinit.x
M06SEC_SFC.SFCInit
STOP_SFC.S15.x
6
MAIN_SFC.MAINinit.x
MAINEME_SFC.SFCInit
7
MAIN_SFC.MAINinit.x
STOP_SFC.SFCInit

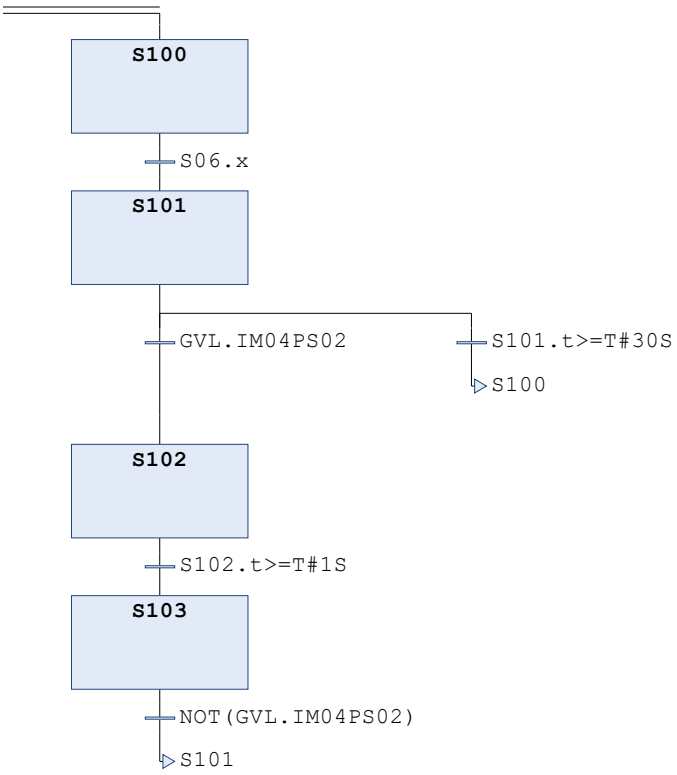
POU: STOP_SFC

```
1       PROGRAM  STOP_SFC
2       VAR
3       END_VAR
4       VAR_INPUT
5           SFCInit  :  BOOL  ;
6       END_VAR
7
```



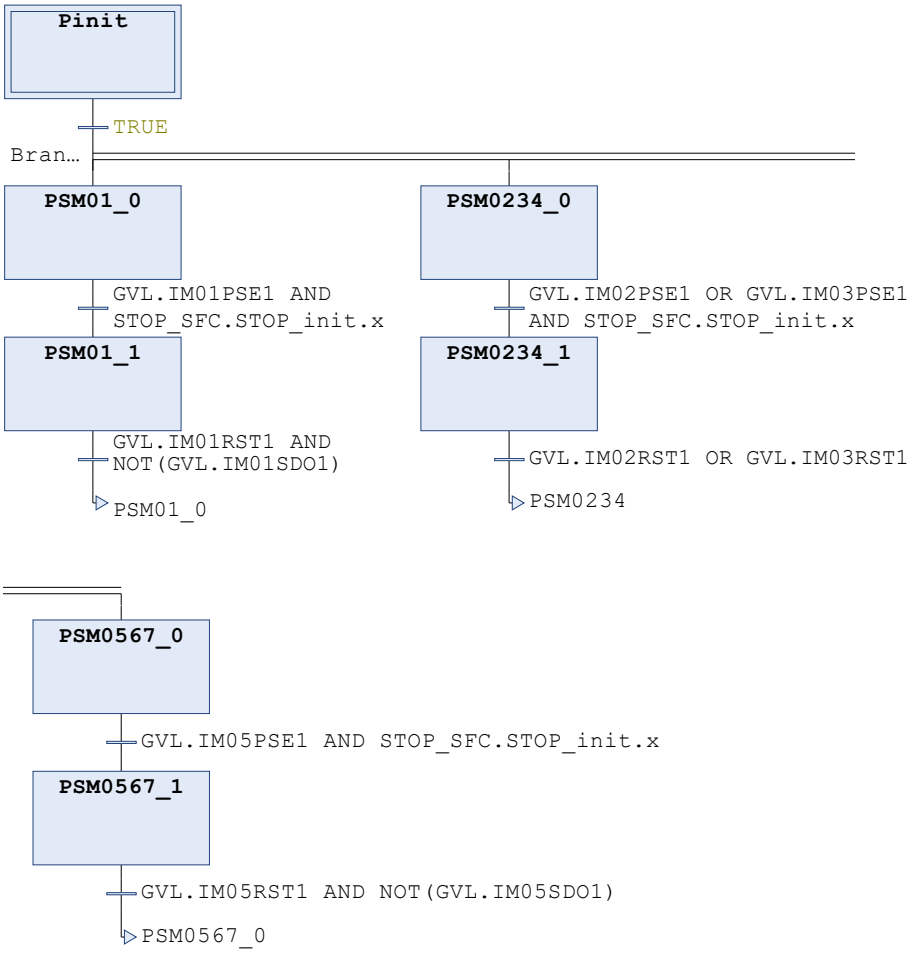

POU: STOP_SFC
S01
M01SEC_LD.TON_3.Q
S03
S03.t>=T#3S
S04
M03SEC_LD.TON_0.Q
S05
M04BGF_SFC_1.S02.t>=T#30S
S06
S101.t>=T#30S
S07
M05ARM_SFC.S0.t>=T#45S OR
M05ARM_SFC.S6.t>=T#45S OR
M05ARM_SFC.S12.t>=T#45S
S08
ABS(0-GVL.AIM05RQA1)<0.01 AND
ABS(0-GVL.AIM05RQA2)<0.01 AND
ABS(0-GVL.AIM05RQA3)<0.01 AND
ABS(0-GVL.AIM05RQA4)<0.01 AND
ABS(0-GVL.AIM05RQA5)<0.01 AND
ABS(0-GVL.AIM05RQA6)<0.01
S09
S09.t>=T#5S
S10
S10.t>=T#30S
S11
S11.t>=T#5S
S12
M06PAL_SFC.S21.t>=T#30S OR
M06PAL_SFC.S22.t>=T#30S OR
M06PAL_SFC.S23.t>=T#30S
S13
M06PAL_SFC.S43.x
S14
M07TYG_SFC.S53.X
S15
NOT(S15.x)
STOP_init
ApplePackingLine10x64.project
8/1/2025 11:03 AM
Page 2 of 3

POU: STOP_SFC


S100
S06.x
S101
GVL.IM04PS02
S101.t>=T#30S
S100
S102
S102.t>=T#1S
S103
NOT(GVL.IM04PS02)
S101

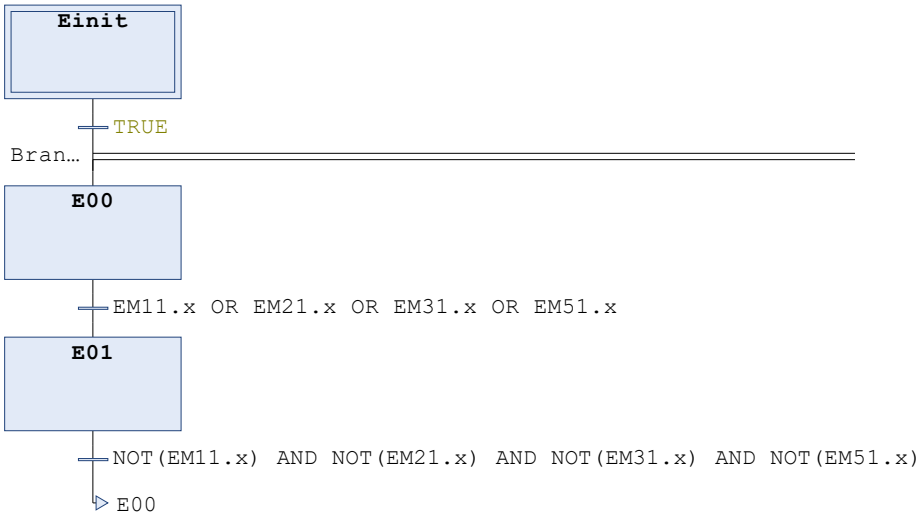
POU: PAUSE_SFC

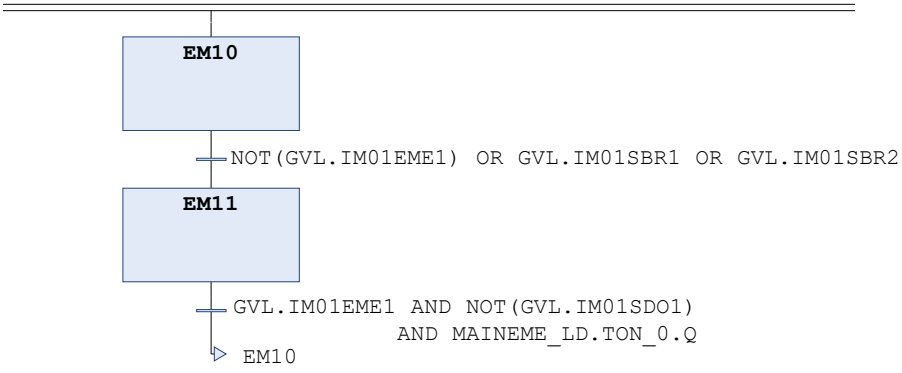
```
1       PROGRAM  PAUSE_SFC
2       VAR
3       END_VAR
4       VAR_INPUT
5           SFCInit  :  BOOL  ;
6       END_VAR
7
```



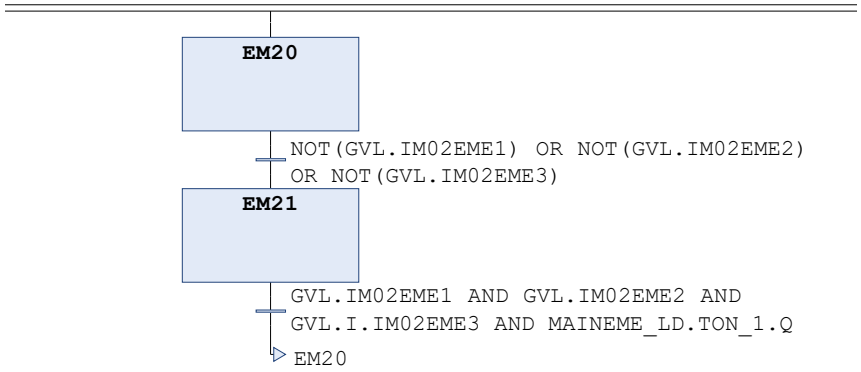
POU: MAINEME_SFC

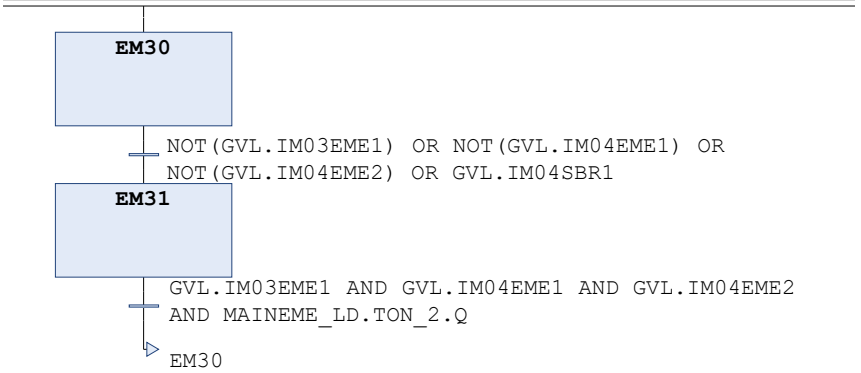
```
1       PROGRAM MAINEME_SFC
2       VAR
3       END_VAR
4       VAR_INPUT
5           SFCInit : BOOL ;
6       END_VAR
7
```



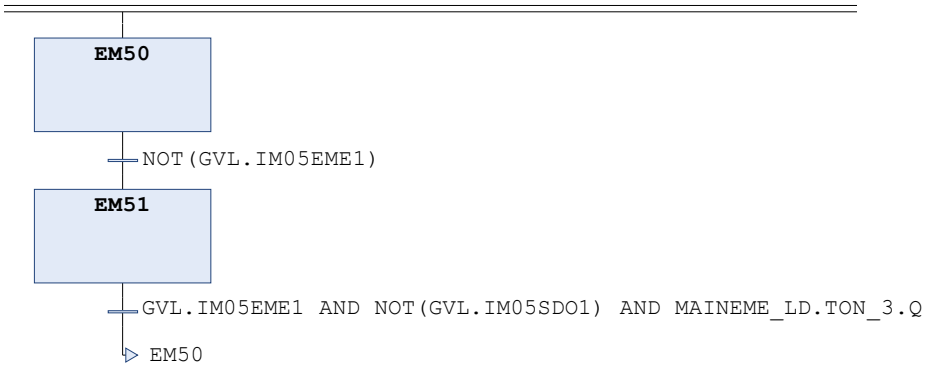


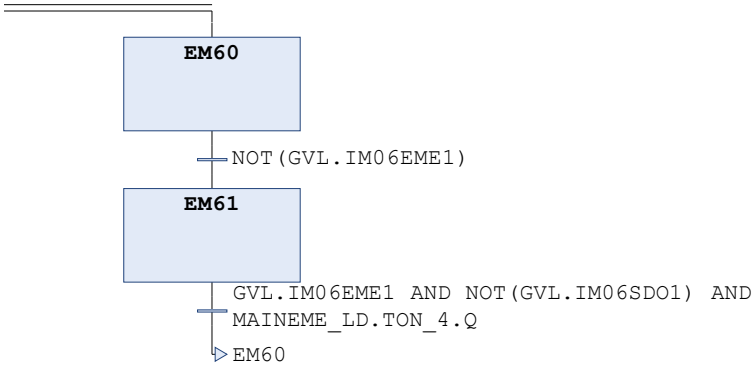
POU: MAINEME_SFC





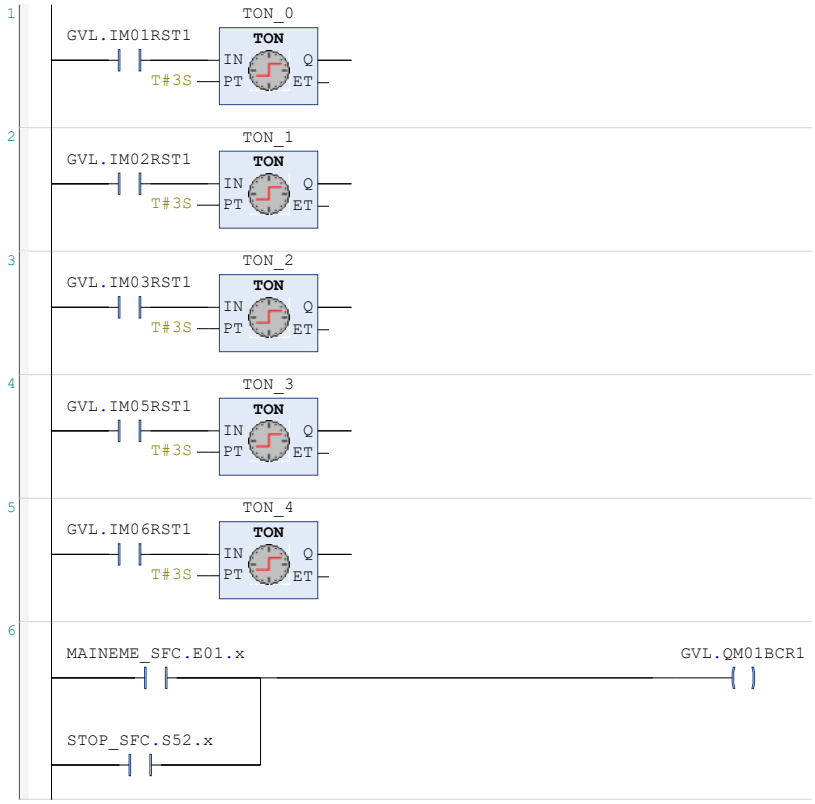
POU: MAINEME_SFC



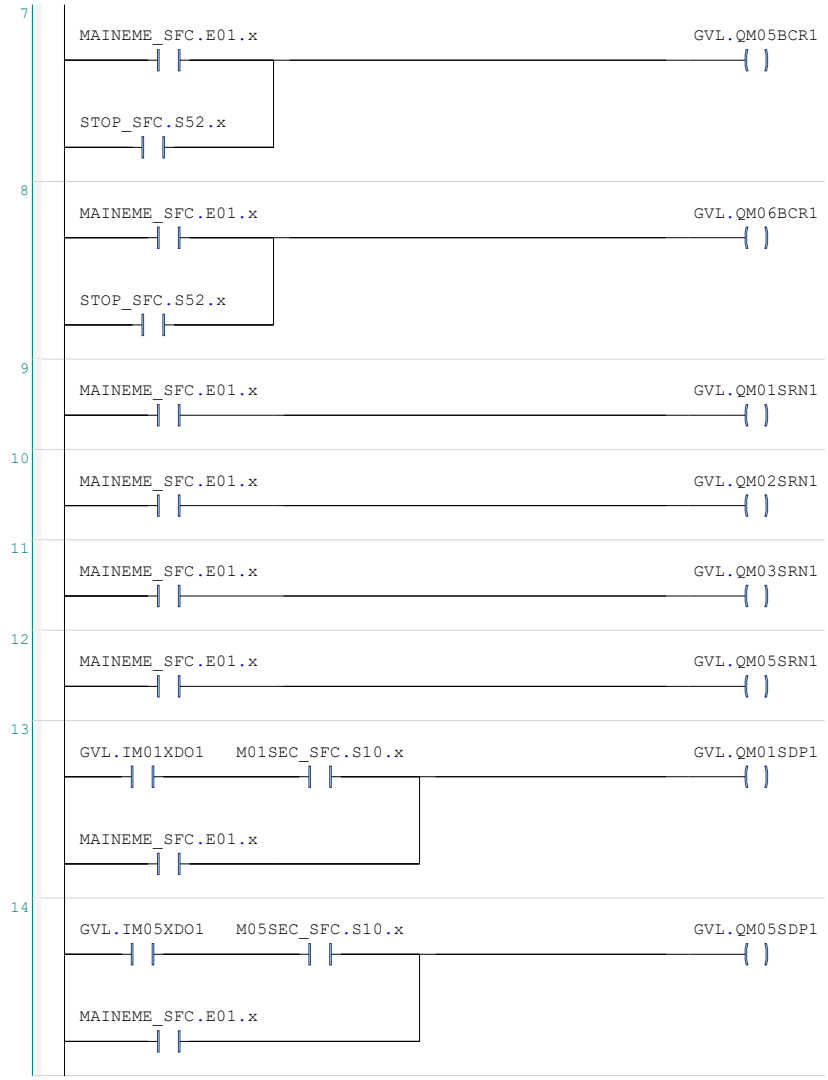


POU: MAINEME_LD

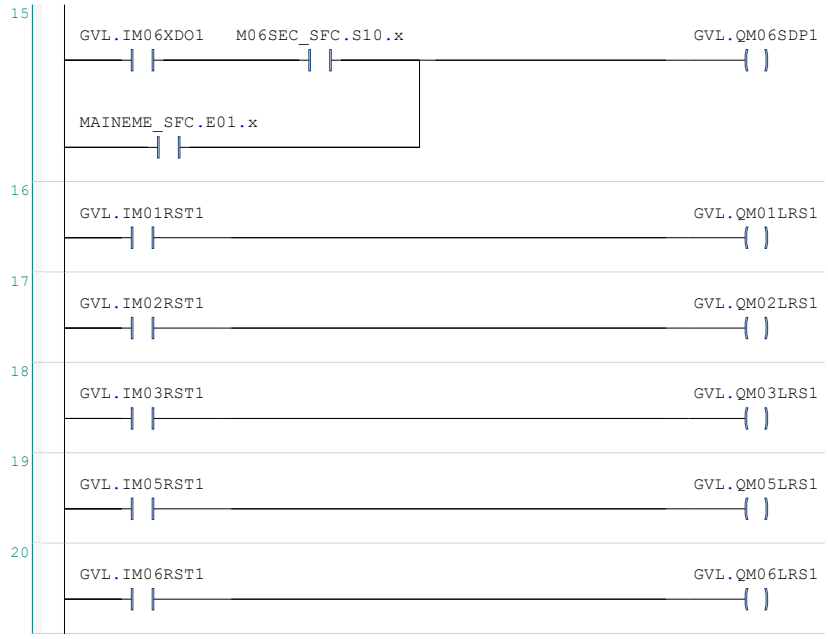
```
1   PROGRAM MAINEME_LD
2   VAR
3       TON_0 : TON ;
4       TON_1 : TON ;
5       TON_2 : TON ;
6       TON_3 : TON ;
7       TON_4 : TON ;
8   END_VAR
9
```



POU: MAINEME_LD

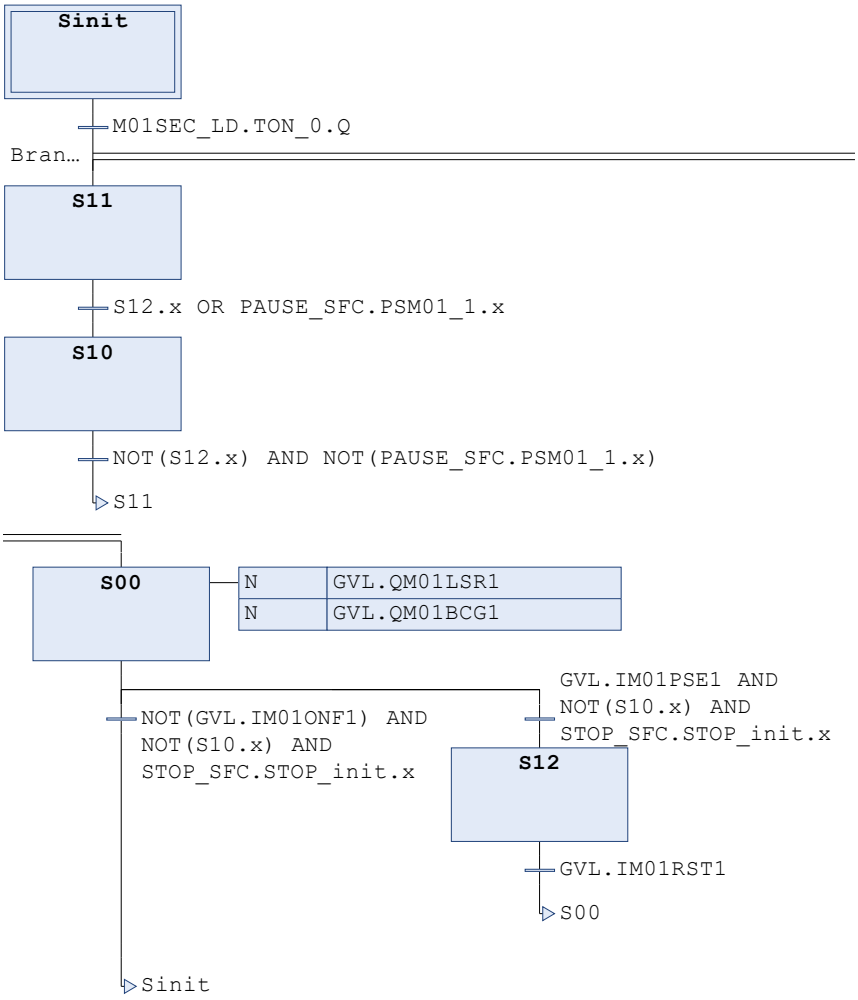


POU: MAINEME_LD



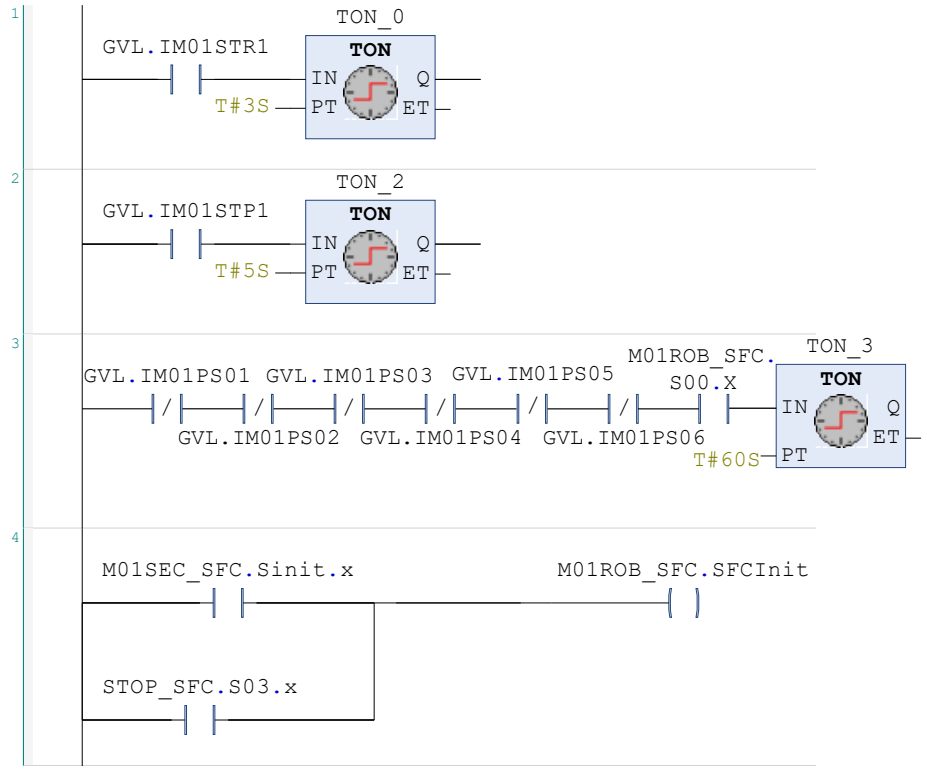
POU: M01SEC_SFC

```
1       PROGRAM M01SEC_SFC
2       VAR
3       END_VAR
4       VAR_INPUT
5           SFCInit : BOOL ;
6       END_VAR
7
```

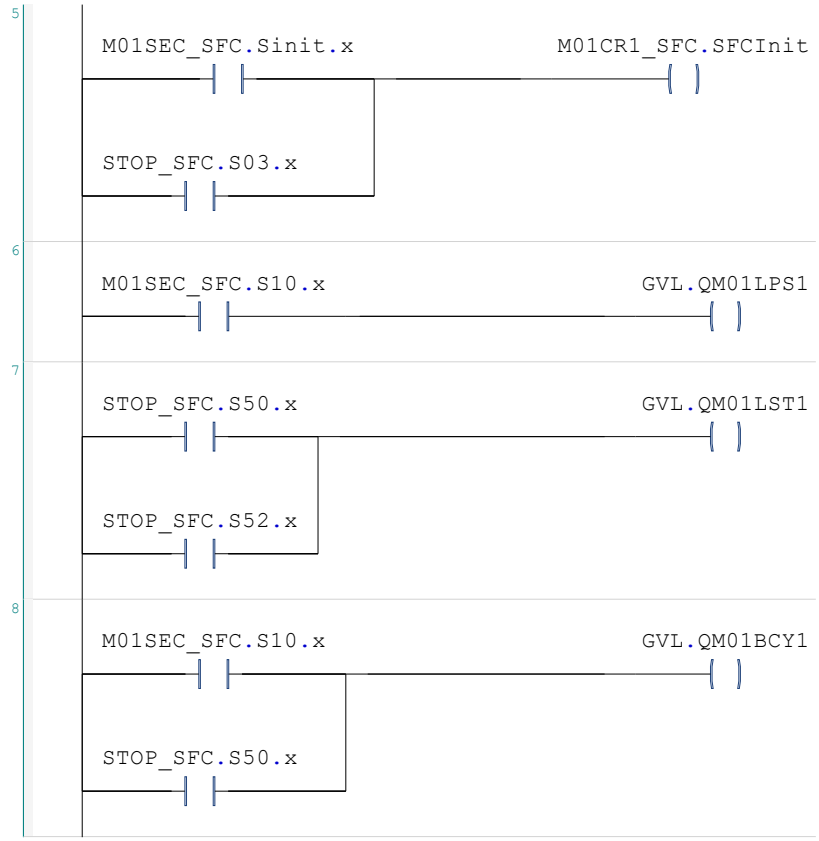


POU: M01SEC_LD

```
1   PROGRAM M01SEC_LD
2   VAR
3       TON_0 : TON ;
4       TON_2 : TON ;
5       TON_3 : TON ;
6   END_VAR
7
```

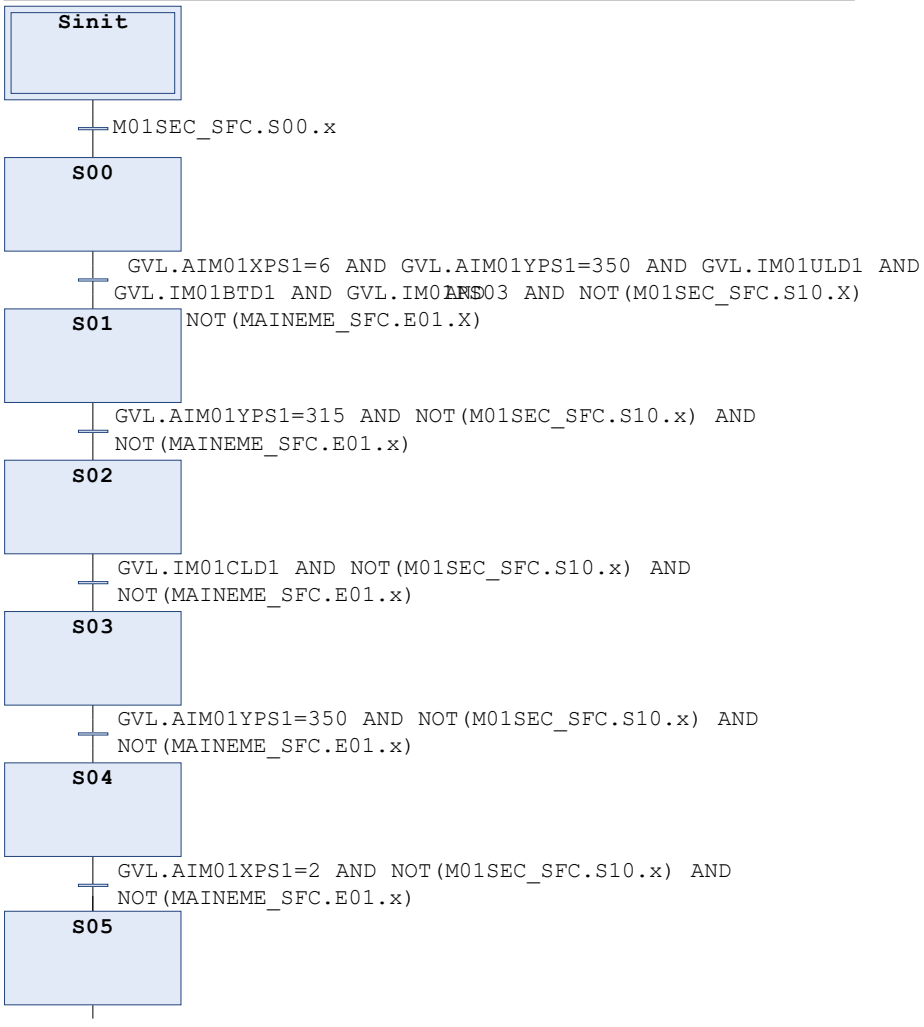


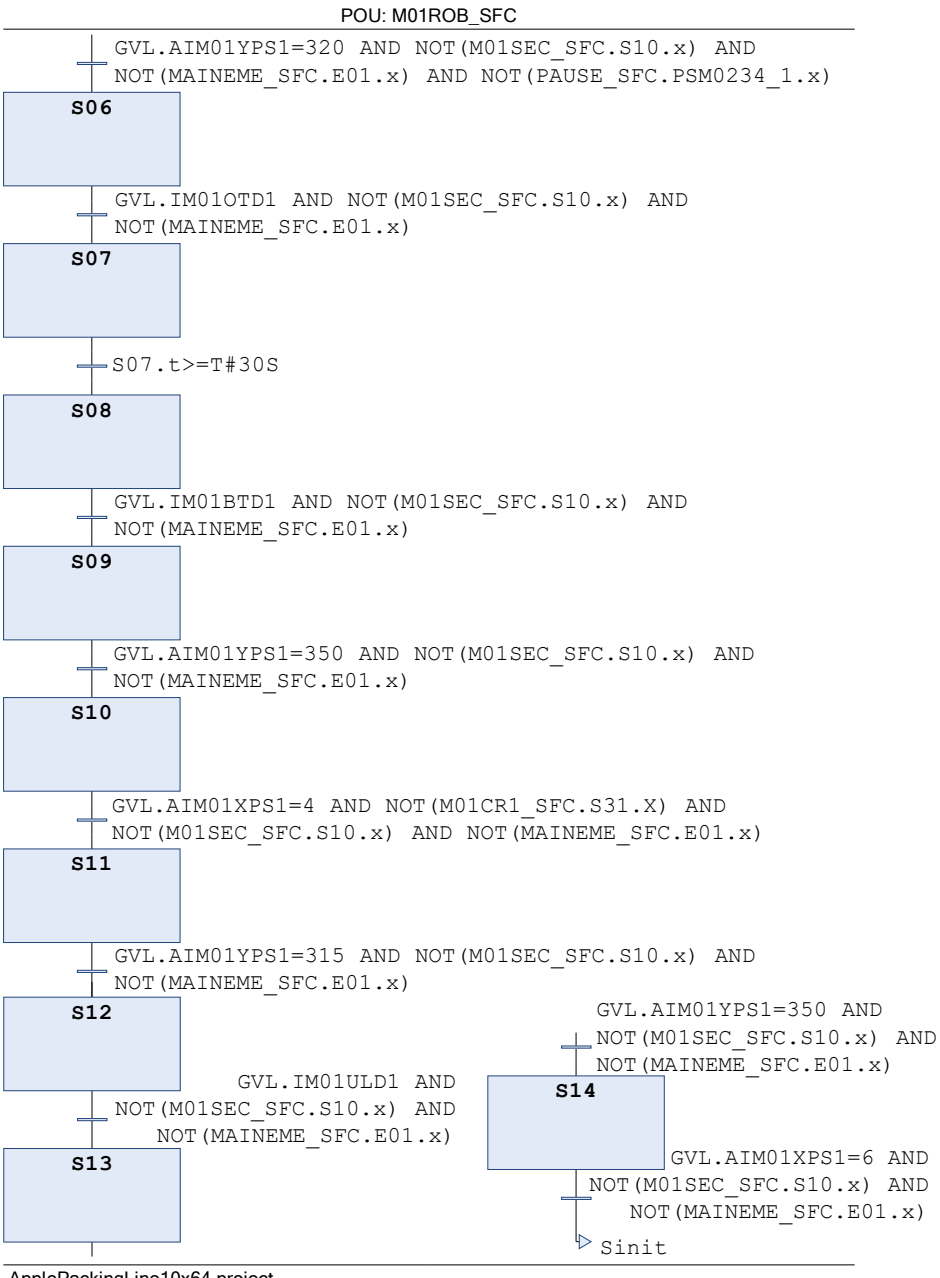
POU: M01SEC_LD



POU: M01ROB_SFC

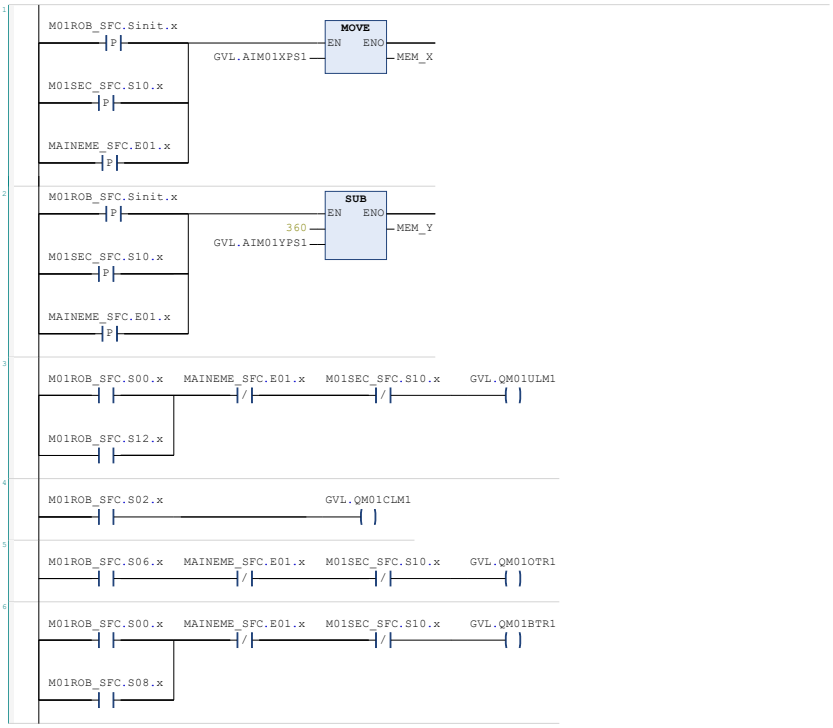
```
1       PROGRAM M01ROB_SFC
2       VAR
3       END_VAR
4       VAR_INPUT
5           SFCInit : BOOL ;
6       END_VAR
7
```



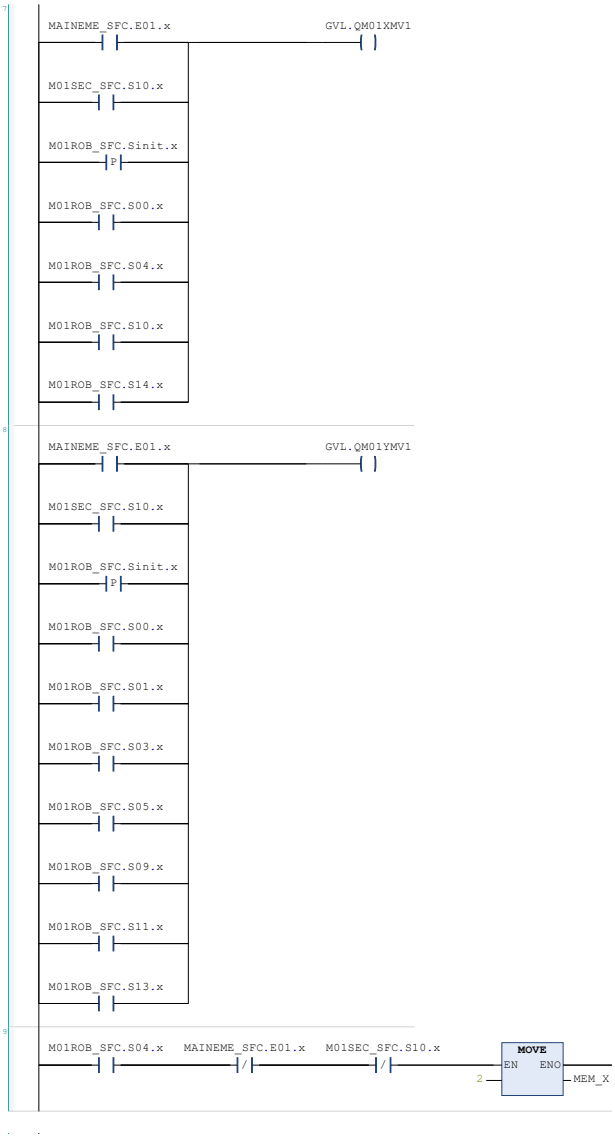

POU: M01ROB_SFC
GVL.AIM01YPS1=320 AND NOT(M01SEC_SFC.S10.x) AND
NOT(MAINEME_SFC.E01.x) AND NOT(PAUSE_SFC.PSM0234_1.x)
S06
GVL.IM01OTD1 AND NOT(M01SEC_SFC.S10.x) AND
NOT(MAINEME_SFC.E01.x)
S07
S07.t>=T#30S
S08
GVL.IM01BTD1 AND NOT(M01SEC_SFC.S10.x) AND
NOT(MAINEME_SFC.E01.x)
S09
GVL.AIM01YPS1=350 AND NOT(M01SEC_SFC.S10.x) AND
NOT(MAINEME_SFC.E01.x)
S10
GVL.AIM01XPS1=4 AND NOT(M01CR1_SFC.S31.X) AND
NOT(M01SEC_SFC.S10.x) AND NOT(MAINEME_SFC.E01.x)
S11
GVL.AIM01YPS1=315 AND NOT(M01SEC_SFC.S10.x) AND
NOT(MAINEME_SFC.E01.x)
S12
GVL.IM01ULD1 AND
NOT(M01SEC_SFC.S10.x) AND
NOT(MAINEME_SFC.E01.x)
S13
GVL.AIM01YPS1=350 AND
NOT(M01SEC_SFC.S10.x) AND
NOT(MAINEME_SFC.E01.x)
S14
GVL.AIM01XPS1=6 AND
NOT(M01SEC_SFC.S10.x) AND
NOT(MAINEME_SFC.E01.x)
Sinit
ApplePackingLine10x64.project
8/1/2025 3:55 PM
Page 2 of 2

POU: M01ROB_LD

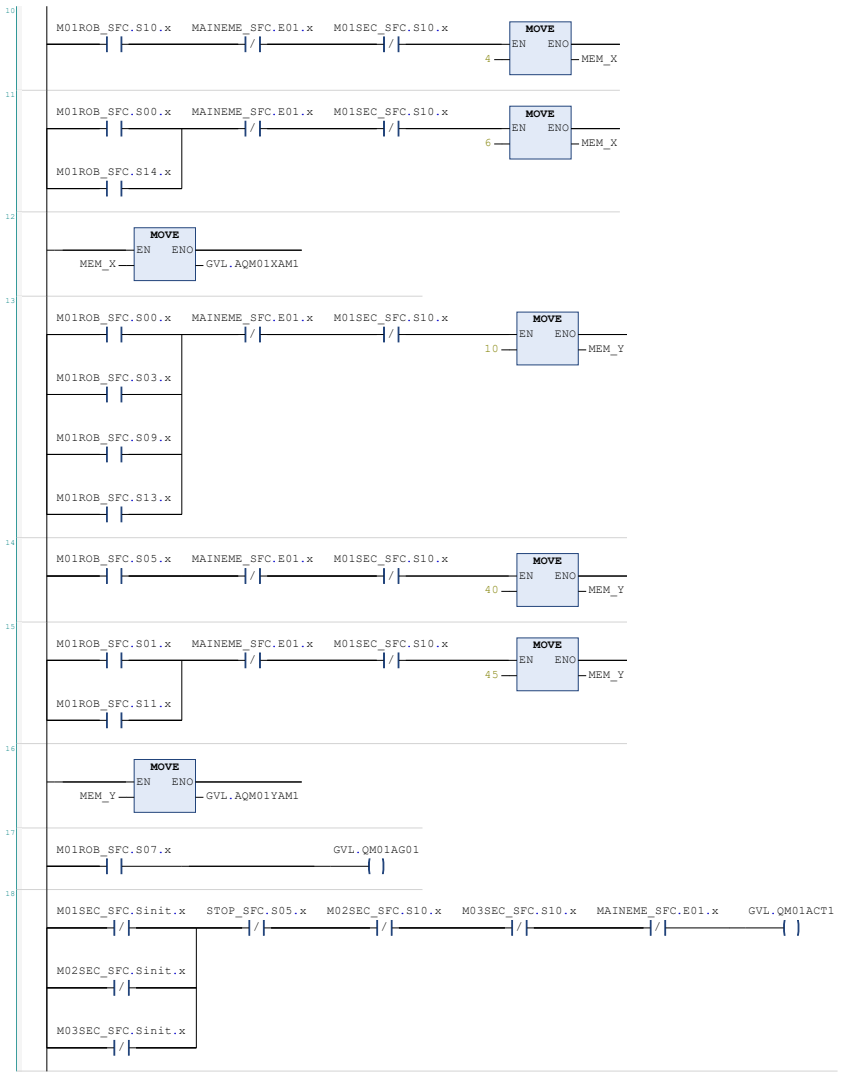
```
1   PROGRAM M01ROB_LD
2   VAR
3       MEM_X : REAL ;
4       MEM_Y : REAL ;
5   END_VAR
6
```



POU: M01ROB_LD

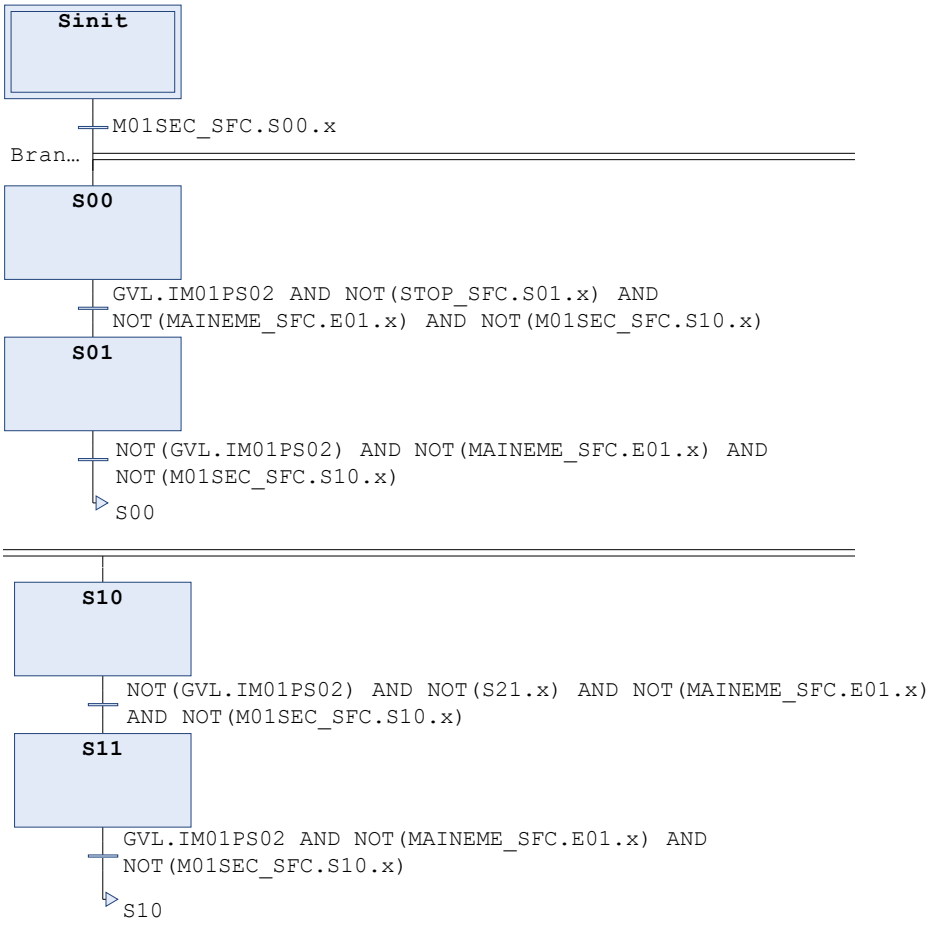


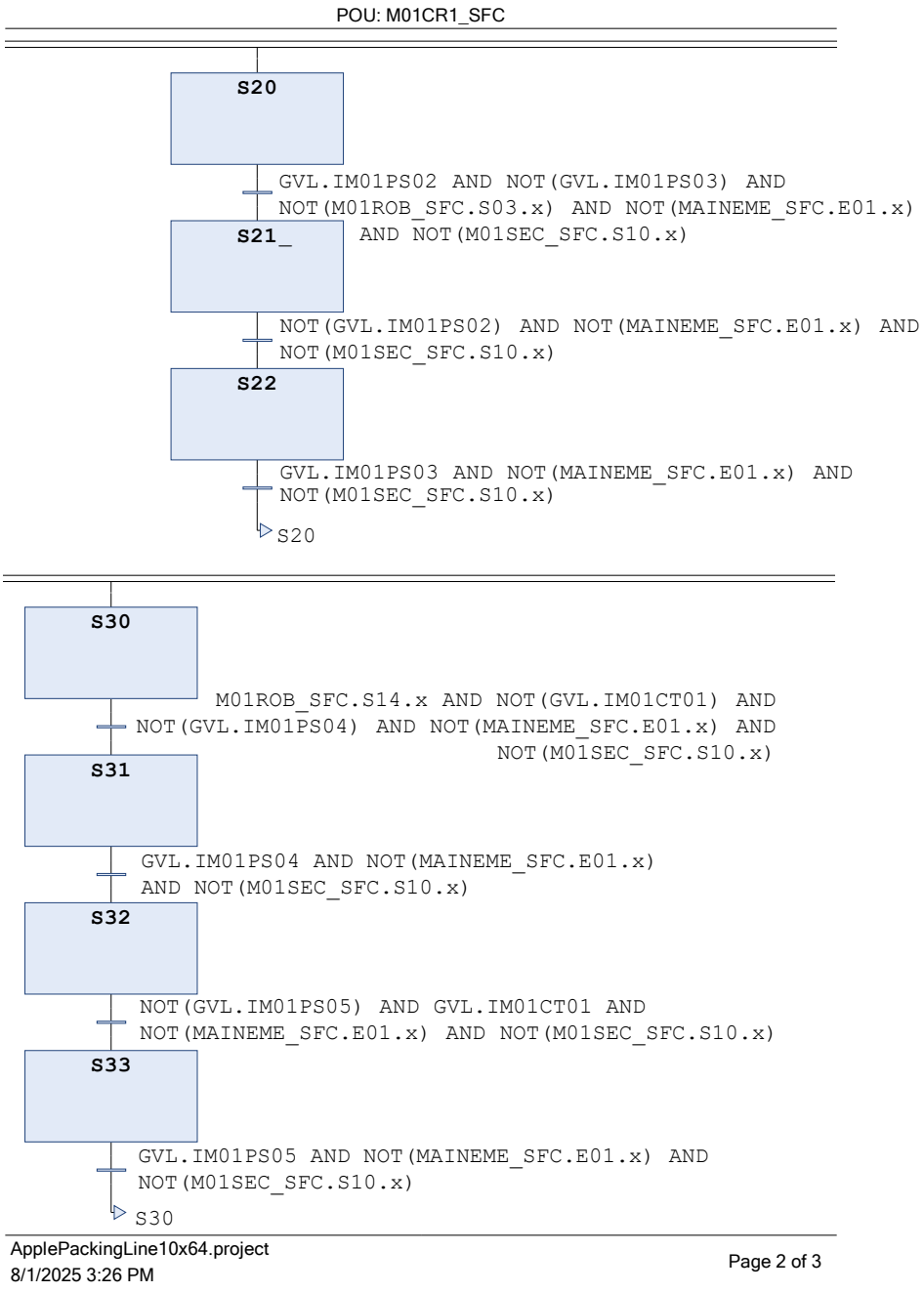
POU: M01ROB_LD

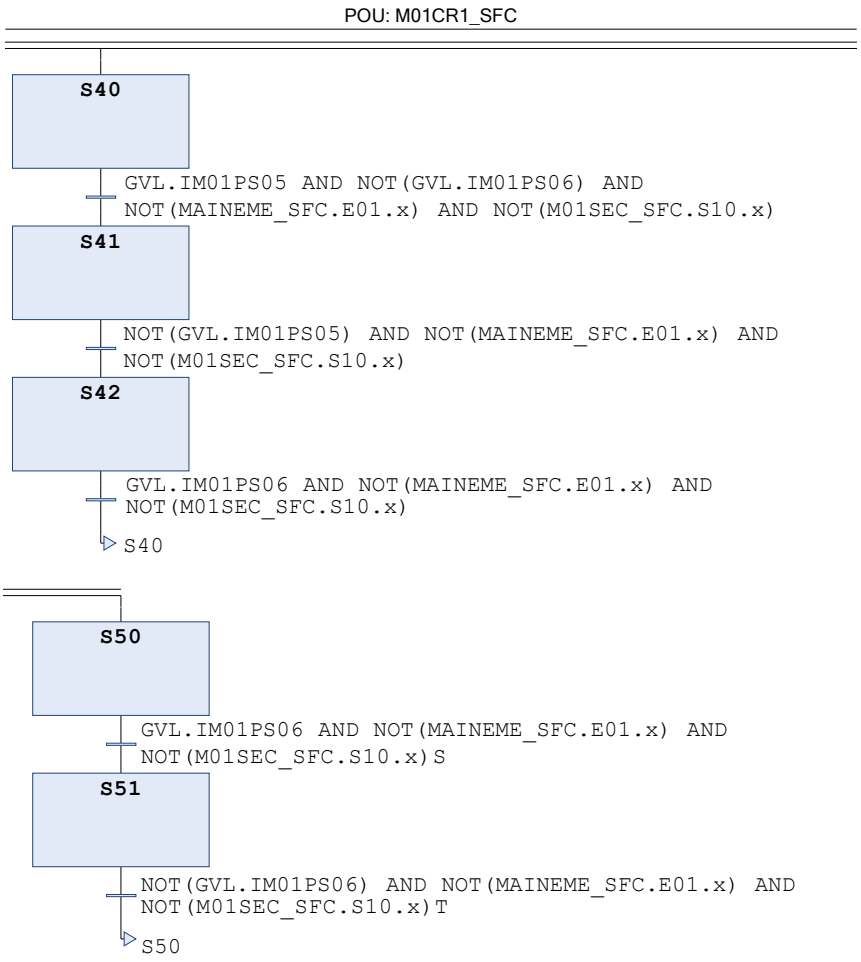


POU: M01CR1_SFC

```
1   PROGRAM M01CR1_SFC
2   VAR
3   END_VAR
4   VAR_INPUT
5       SFCInit : BOOL ;
6   END_VAR
7
```

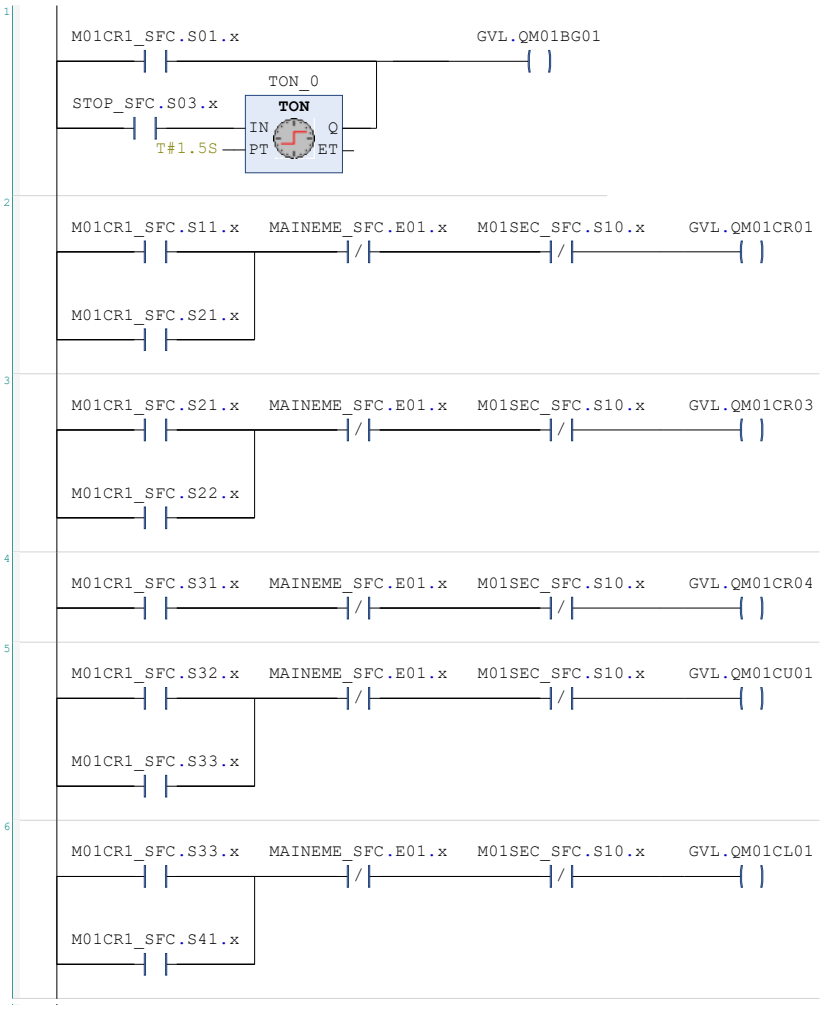



POU: M01CR1_SFC
S20
GVL.IM01PS02 AND NOT(GVL.IM01PS03) AND
NOT(M01ROB_SFC.S03.x) AND NOT(MAINEME_SFC.E01.x)
S21_
AND NOT(M01SEC_SFC.S10.x)
NOT(GVL.IM01PS02) AND NOT(MAINEME_SFC.E01.x) AND
NOT(M01SEC_SFC.S10.x)
S22
GVL.IM01PS03 AND NOT(MAINEME_SFC.E01.x) AND
NOT(M01SEC_SFC.S10.x)
S20
S30
M01ROB_SFC.S14.x AND NOT(GVL.IM01CT01) AND
NOT(GVL.IM01PS04) AND NOT(MAINEME_SFC.E01.x) AND
NOT(M01SEC_SFC.S10.x)
S31
GVL.IM01PS04 AND NOT(MAINEME_SFC.E01.x)
AND NOT(M01SEC_SFC.S10.x)
S32
NOT(GVL.IM01PS05) AND GVL.IM01CT01 AND
NOT(MAINEME_SFC.E01.x) AND NOT(M01SEC_SFC.S10.x)
S33
GVL.IM01PS05 AND NOT(MAINEME_SFC.E01.x) AND
NOT(M01SEC_SFC.S10.x)
S30
ApplePackingLine10x64.project
8/1/2025 3:26 PM
Page 2 of 3

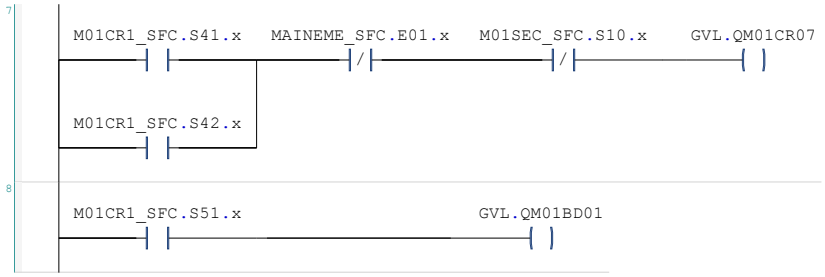

POU: M01CR1_SFC
S40
GVL.IM01PS05 AND NOT(GVL.IM01PS06) AND
NOT(MAINEME_SFC.E01.x) AND NOT(M01SEC_SFC.S10.x)
S41
NOT(GVL.IM01PS05) AND NOT(MAINEME_SFC.E01.x) AND
NOT(M01SEC_SFC.S10.x)
S42
GVL.IM01PS06 AND NOT(MAINEME_SFC.E01.x) AND
NOT(M01SEC_SFC.S10.x)
S40
S50
GVL.IM01PS06 AND NOT(MAINEME_SFC.E01.x) AND
NOT(M01SEC_SFC.S10.x)S
S51
NOT(GVL.IM01PS06) AND NOT(MAINEME_SFC.E01.x) AND
NOT(M01SEC_SFC.S10.x)T
S50

POU: M01CR1_LD

```
1   PROGRAM M01CR1_LD
2   VAR
3       TON_0 : TON ;
4   END_VAR
5
```

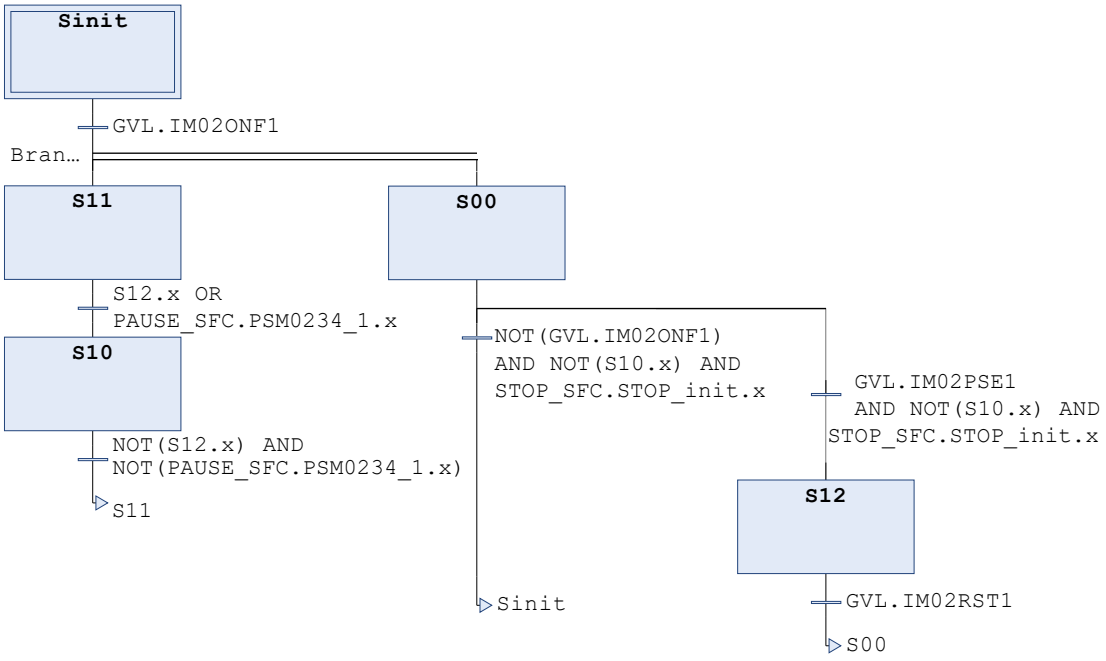


POU: M01CR1_LD



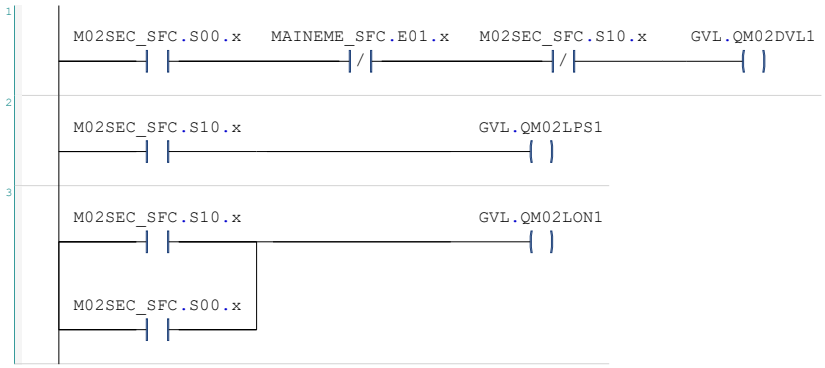
POU: M02SEC_SFC

```
1       PROGRAM M02SEC_SFC
2       VAR
3       END_VAR
4       VAR_INPUT
5           SFCInit : BOOL ;
6       END_VAR
7
```



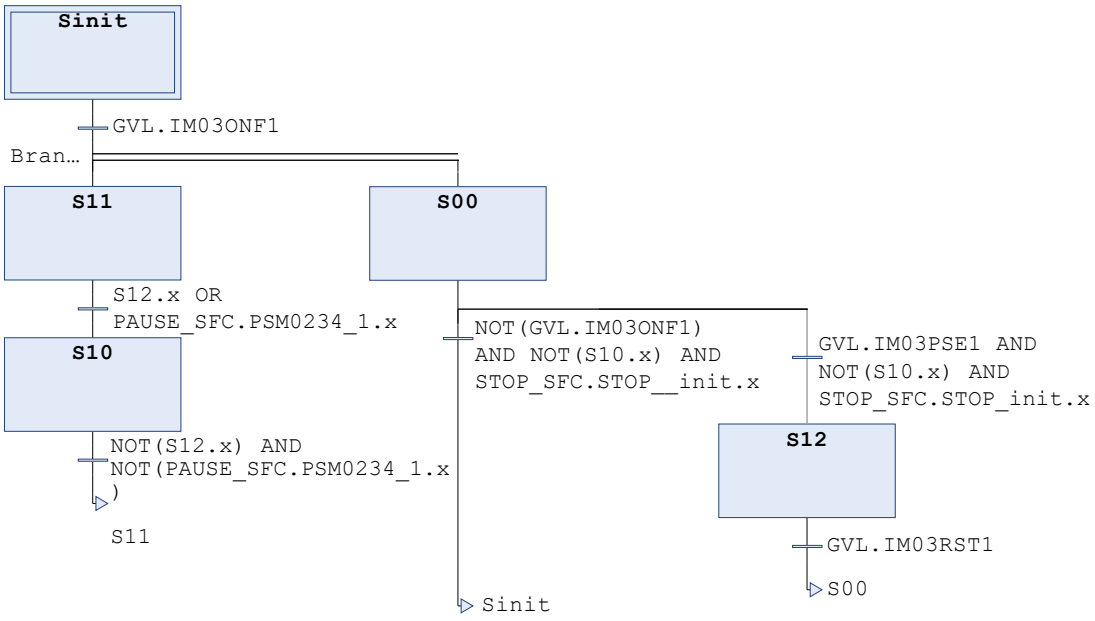
POU: M02SEC_LD

```
1   PROGRAM M02SEC_LD
2   VAR
3
4   END_VAR
5
```



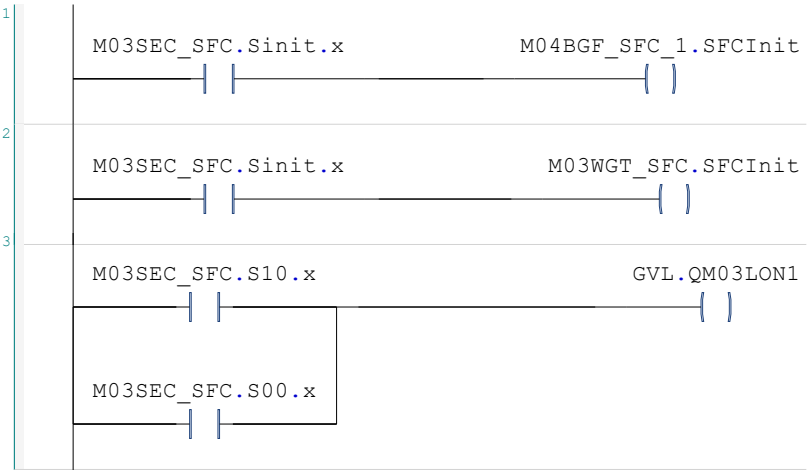
POU: M03SEC_SFC

```
1       PROGRAM M03SEC_SFC
2       VAR
3       END_VAR
4       VAR_INPUT
5           SFCInit : BOOL ;
6       END_VAR
7
```

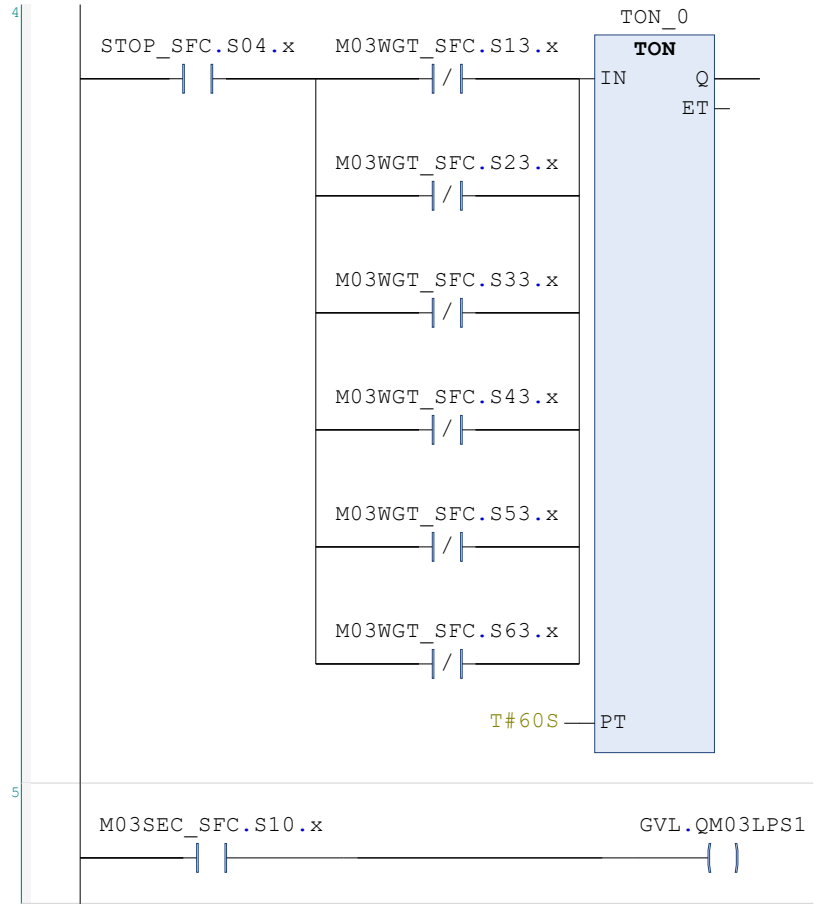


POU: M03SEC_LD

```
1   PROGRAM M03SEC_LD
2   VAR
3       TON_0 : TON ;
4   END_VAR
5
```



POU: M03SEC_LD



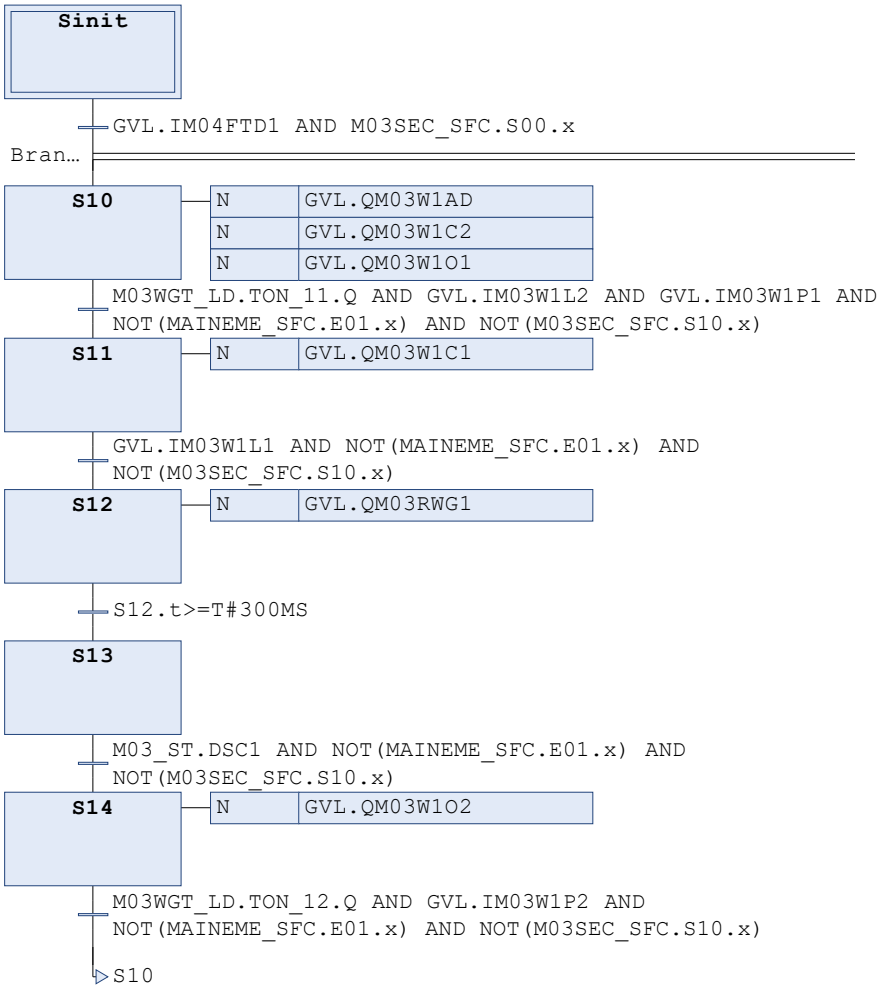
```
PROGRAM M03_ST
VAR
    n,m,i,j,k,l : INT;
    Weights : ARRAY[1..6] OF REAL := [0,0,0,0,0,0];
    BestGroup : ARRAY[1..3] OF INT := [0,0,0];
    Sensors : ARRAY[1..6] OF BOOL;
    ClosestSum : REAL;
    Timer : TON;
    //CountReady : INT;
    //State : INT := 0;
    DSC1 : BOOL;
    DSC2 : BOOL;
    DSC3 : BOOL;
    DSC4 : BOOL;
    DSC5 : BOOL;
    DSC6 : BOOL;
END_VAR

m := 1;
Timer(IN := m = 1, PT := T#1S);
IF Timer.Q THEN
    DSC1 := FALSE; DSC2 := FALSE; DSC3 := FALSE;
    DSC4 := FALSE; DSC5 := FALSE; DSC6 := FALSE;
END_IF
IF NOT(STOP_SFC.S05.x) AND M04BGF_SFC_1.S02.x AND Timer.Q AND M03WGT_SFC.S13.x AND
    M03WGT_SFC.S23.x AND M03WGT_SFC.S33.x AND M03WGT_SFC.S43.x AND M03WGT_SFC.S53.x AND
    M03WGT_SFC.S63.x THEN
    m := 0;
    n := 0;
    Weights[1] := GVL.AIM03W1WG;
    Weights[2] := GVL.AIM03W2WG;
    Weights[3] := GVL.AIM03W3WG;
    Weights[4] := GVL.AIM03W4WG;
    Weights[5] := GVL.AIM03W5WG;
    Weights[6] := GVL.AIM03W6WG;
    ClosestSum := 99999.0;
    BestGroup[1] := 0;
    BestGroup[2] := 0;
    BestGroup[3] := 0;
    FOR i := 1 TO 4 DO
        FOR j := i + 1 TO 5 DO
            FOR k := j + 1 TO 6 DO
                IF Weights[i]>0 AND Weights[j]>0 AND Weights[k]>0 THEN
                    GVL.WeightSum := Weights[i] + Weights[j] + Weights[k];
                    IF ABS(GVL.WeightSum - 1500.0) < ABS(ClosestSum - 1500.0) THEN
                        ClosestSum := GVL.WeightSum;
                        BestGroup[1] := i;
                        BestGroup[2] := j;
                        BestGroup[3] := k;
                    END_IF
                END_IF
            END_FOR
        END_FOR
    END_FOR
```

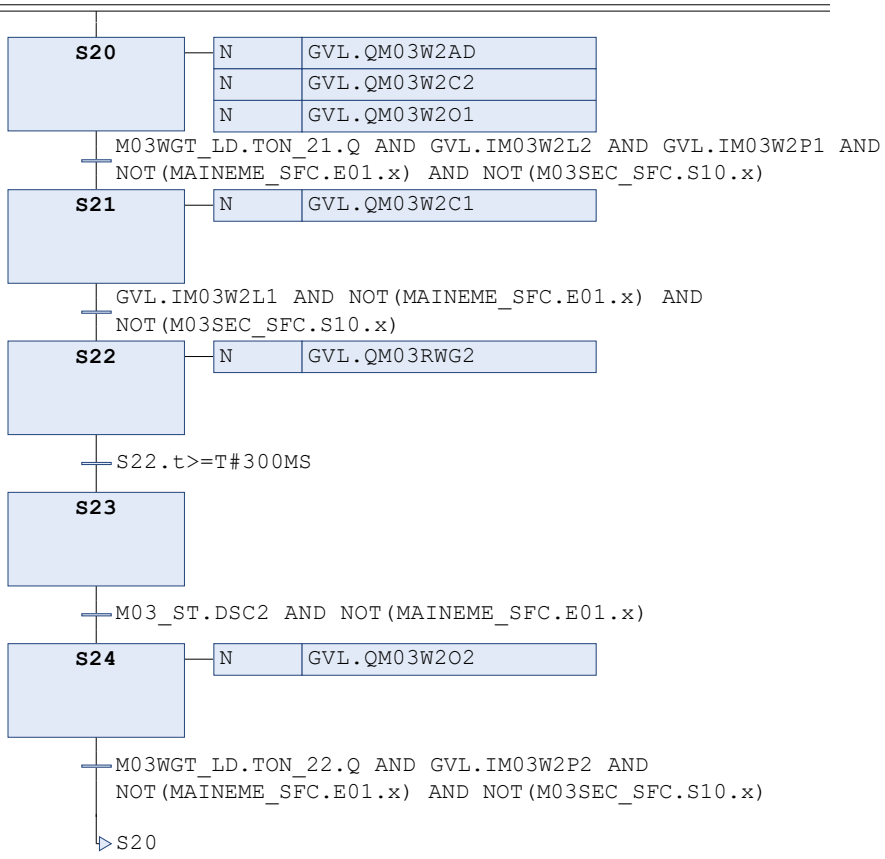
```
    n := 1;
        IF n = BestGroup[1] OR n = BestGroup[2] OR n = BestGroup[3] AND n = 1 THEN
            DSC1 := TRUE;
        END_IF
    n := 2;
        IF n = BestGroup[1] OR n = BestGroup[2] OR n = BestGroup[3] AND n = 2 THEN
            DSC2 := TRUE;
        END_IF
    n := 3;
        IF n = BestGroup[1] OR n = BestGroup[2] OR n = BestGroup[3] AND n = 3 THEN
            DSC3 := TRUE;
        END_IF
    n := 4;
        IF n = BestGroup[1] OR n = BestGroup[2] OR n = BestGroup[3] AND n = 4 THEN
            DSC4 := TRUE;
        END_IF
    n := 5;
        IF n = BestGroup[1] OR n = BestGroup[2] OR n = BestGroup[3] AND n = 5 THEN
            DSC5 := TRUE;
        END_IF
    n := 6;
        IF n = BestGroup[1] OR n = BestGroup[2] OR n = BestGroup[3] AND n = 6 THEN
            DSC6 := TRUE;
        END_IF
ELSIF STOP_SFC.S05.x AND M04BGF_SFC_1.S02.x AND Timer.Q THEN
    m := 0;
    n := 0;
    Weights[1] := GVL.AIM03W1WG;
    Weights[2] := GVL.AIM03W2WG;
    Weights[3] := GVL.AIM03W3WG;
    Weights[4] := GVL.AIM03W4WG;
    Weights[5] := GVL.AIM03W5WG;
    Weights[6] := GVL.AIM03W6WG;
    Sensors[1] := GVL.IM03W1PS;
    Sensors[2] := GVL.IM03W2PS;
    Sensors[3] := GVL.IM03W3PS;
    Sensors[4] := GVL.IM03W4PS;
    Sensors[5] := GVL.IM03W5PS;
    Sensors[6] := GVL.IM03W6PS;
    ClosestSum := 99999.0;
    BestGroup[1] := 0;
    BestGroup[2] := 0;
    BestGroup[3] := 0;
    FOR i := 1 TO 4 DO
        IF Sensors[i] THEN
            FOR j := i + 1 TO 5 DO
                IF Sensors[j] THEN
                    FOR k := j + 1 TO 6 DO
                        IF Sensors[k] THEN
                            IF Weights[i]>0 AND Weights[j]>0 AND Weights[k]>0 THEN
                                GVL.WeightSum := Weights[i] + Weights[j] + Weights[k];
                                //ClosestSum := WeightSum;
                                BestGroup[1] := i;
                                BestGroup[2] := j;
                                BestGroup[3] := k;
```

```
                        END_IF
                    END_IF
                END_FOR
            END_IF
        END_FOR
    END_IF
END_FOR
n := 1;
    IF n = BestGroup[1] OR n = BestGroup[2] OR n = BestGroup[3] AND n = 1 THEN
        DSC1 := TRUE;
    END_IF
n := 2;
    IF n = BestGroup[1] OR n = BestGroup[2] OR n = BestGroup[3] AND n = 2 THEN
        DSC2 := TRUE;
    END_IF
n := 3;
    IF n = BestGroup[1] OR n = BestGroup[2] OR n = BestGroup[3] AND n = 3 THEN
        DSC3 := TRUE;
    END_IF
n := 4;
    IF n = BestGroup[1] OR n = BestGroup[2] OR n = BestGroup[3] AND n = 4 THEN
        DSC4 := TRUE;
    END_IF
n := 5;
    IF n = BestGroup[1] OR n = BestGroup[2] OR n = BestGroup[3] AND n = 5 THEN
        DSC5 := TRUE;
    END_IF
n := 6;
    IF n = BestGroup[1] OR n = BestGroup[2] OR n = BestGroup[3] AND n = 6 THEN
        DSC6 := TRUE;
    END_IF
END_IF
```

POU: M03WGT_SFC

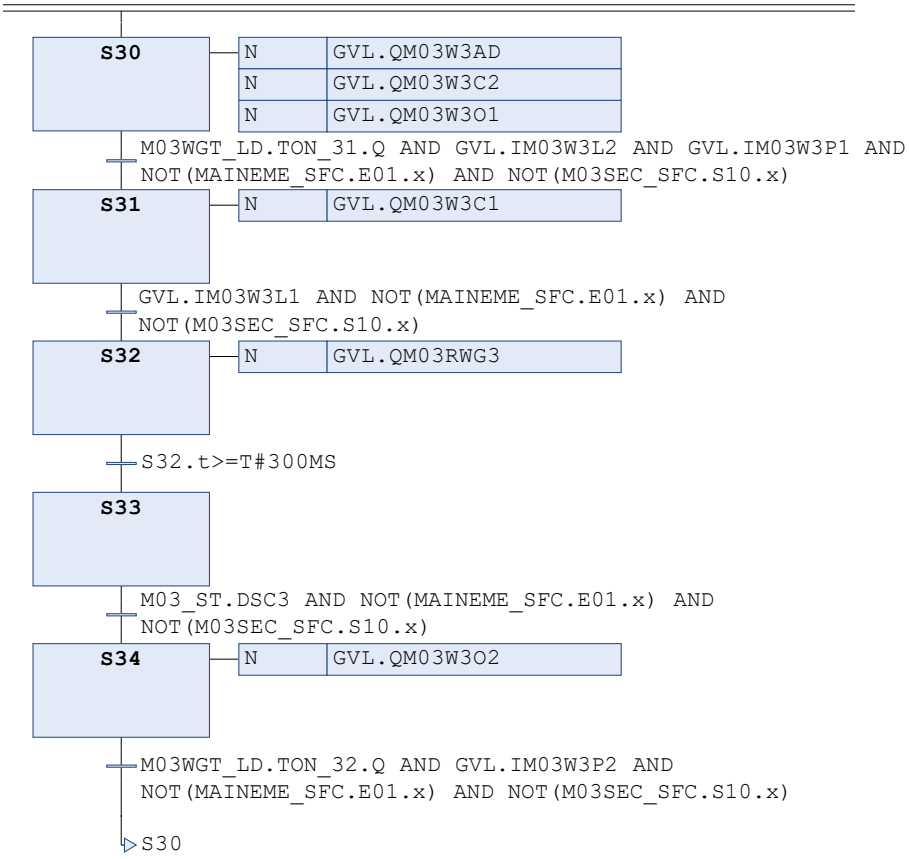
```
1       PROGRAM M03WGT_SFC
2       VAR
3       END_VAR
4       VAR_INPUT
5           SFCInit : BOOL ;
6       END_VAR
7
```

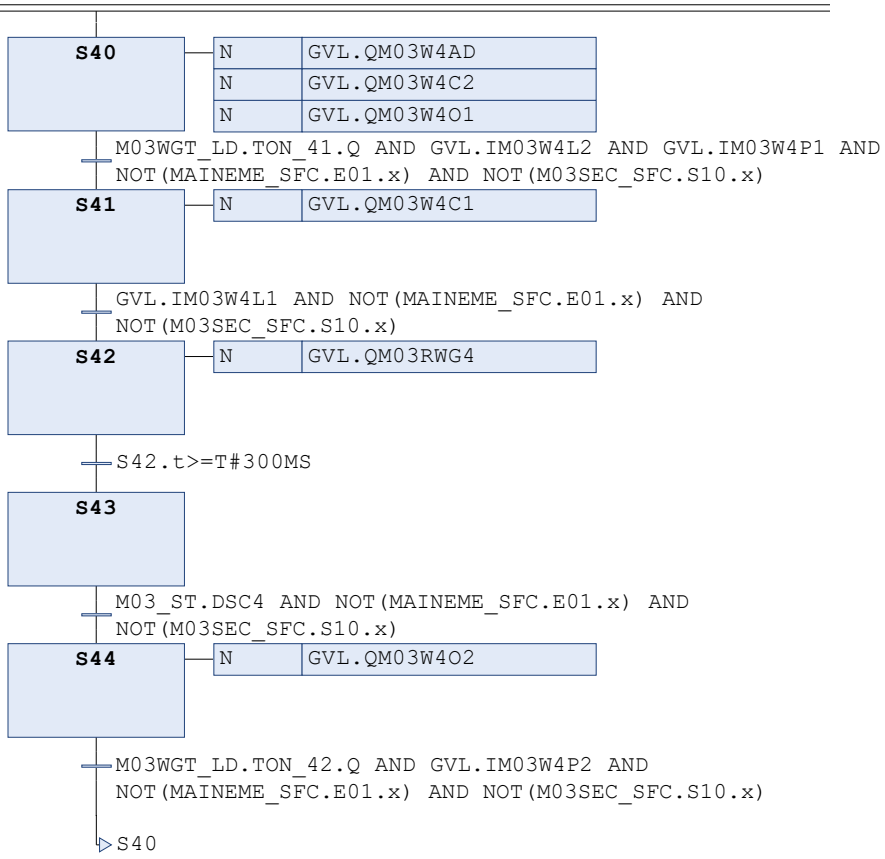


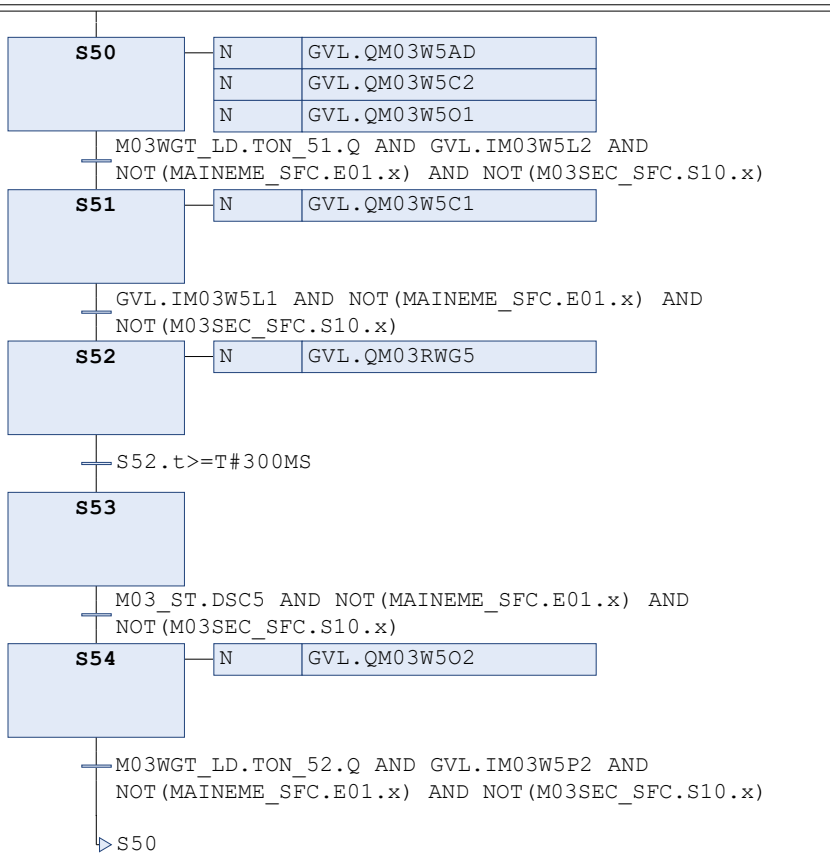

S20
N GVL.QM03W2AD
N GVL.QM03W2C2
N GVL.QM03W2O1
M03WGT_LD.TON_21.Q AND GVL.IM03W2L2 AND GVL.IM03W2P1 AND NOT(MAINEME_SFC.E01.x) AND NOT(M03SEC_SFC.S10.x)
S21
N GVL.QM03W2C1
GVL.IM03W2L1 AND NOT(MAINEME_SFC.E01.x) AND NOT(M03SEC_SFC.S10.x)
S22
N GVL.QM03RWG2
S22.t>=T#300MS
S23
M03_ST.DSC2 AND NOT(MAINEME_SFC.E01.x)
S24
N GVL.QM03W2O2
M03WGT_LD.TON_22.Q AND GVL.IM03W2P2 AND NOT(MAINEME_SFC.E01.x) AND NOT(M03SEC_SFC.S10.x)
S20

3

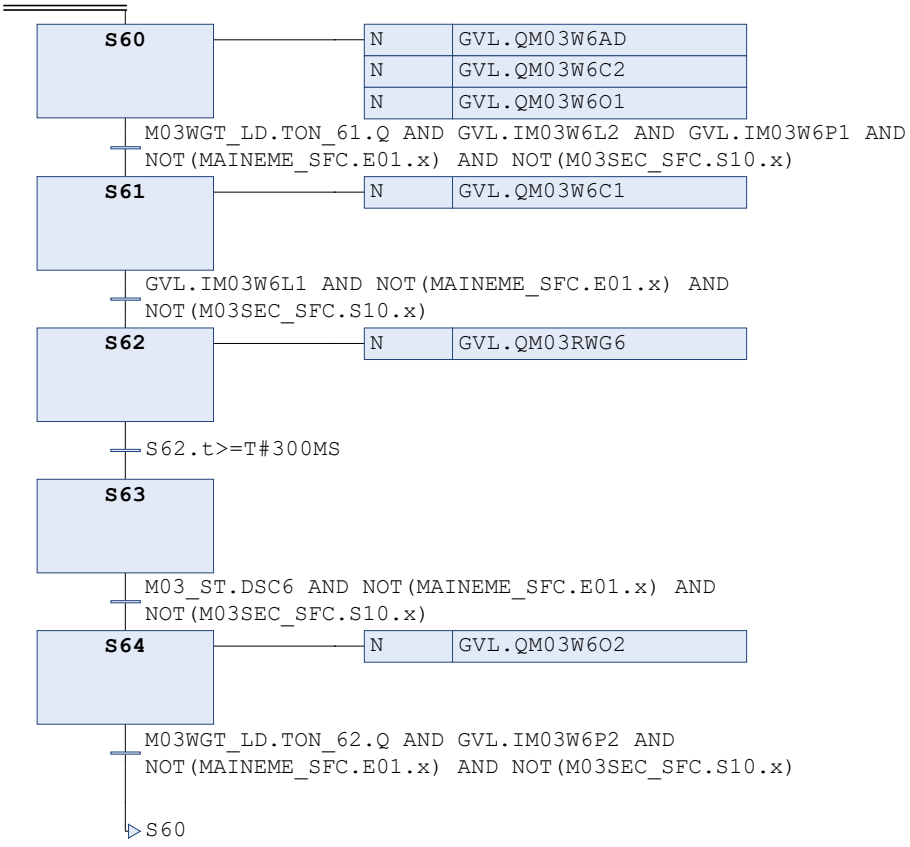
POU: M03WGT_SFC




S40
N GVL.QM03W4AD
N GVL.QM03W4C2
N GVL.QM03W4O1
M03WGT_LD.TON_41.Q AND GVL.IM03W4L2 AND GVL.IM03W4P1 AND NOT(MAINEME_SFC.E01.x) AND NOT(M03SEC_SFC.S10.x)
S41
N GVL.QM03W4C1
GVL.IM03W4L1 AND NOT(MAINEME_SFC.E01.x) AND NOT(M03SEC_SFC.S10.x)
S42
N GVL.QM03RWG4
S42.t>=T#300MS
S43
M03_ST.DSC4 AND NOT(MAINEME_SFC.E01.x) AND NOT(M03SEC_SFC.S10.x)
S44
N GVL.QM03W4O2
M03WGT_LD.TON_42.Q AND GVL.IM03W4P2 AND NOT(MAINEME_SFC.E01.x) AND NOT(M03SEC_SFC.S10.x)
S40

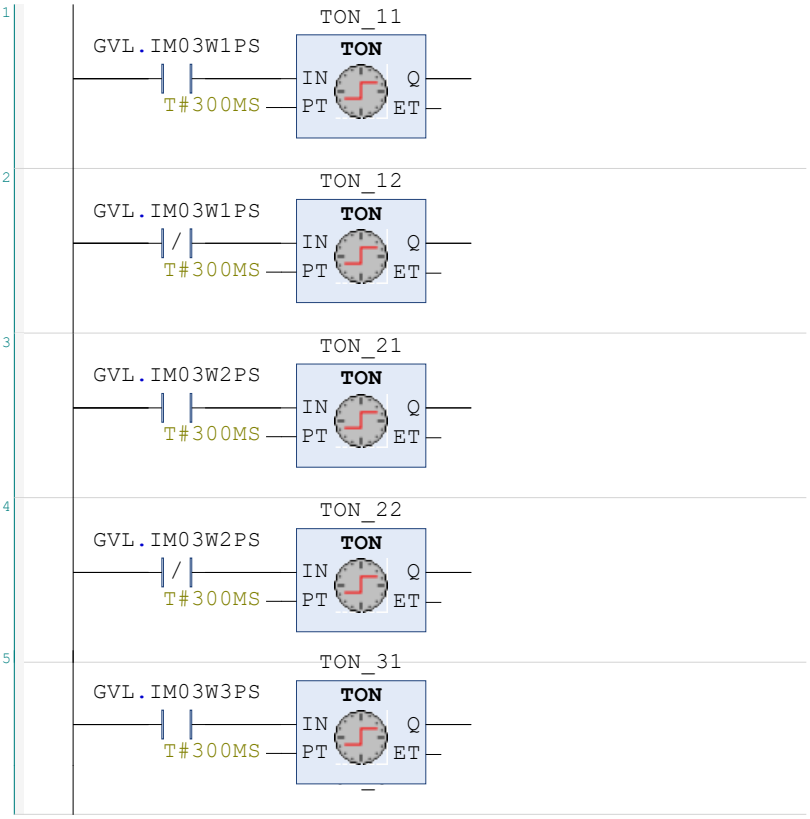

S50
N GVL.QM03W5AD
N GVL.QM03W5C2
N GVL.QM03W5O1
M03WGT_LD.TON_51.Q AND GVL.IM03W5L2 AND NOT(MAINEME_SFC.E01.x) AND NOT(M03SEC_SFC.S10.x)
S51
N GVL.QM03W5C1
GVL.IM03W5L1 AND NOT(MAINEME_SFC.E01.x) AND NOT(M03SEC_SFC.S10.x)
S52
N GVL.QM03RWG5
S52.t>=T#300MS
S53
M03_ST.DSC5 AND NOT(MAINEME_SFC.E01.x) AND NOT(M03SEC_SFC.S10.x)
S54
N GVL.QM03W5O2
M03WGT_LD.TON_52.Q AND GVL.IM03W5P2 AND NOT(MAINEME_SFC.E01.x) AND NOT(M03SEC_SFC.S10.x)
S50

POU: M03WGT_SFC

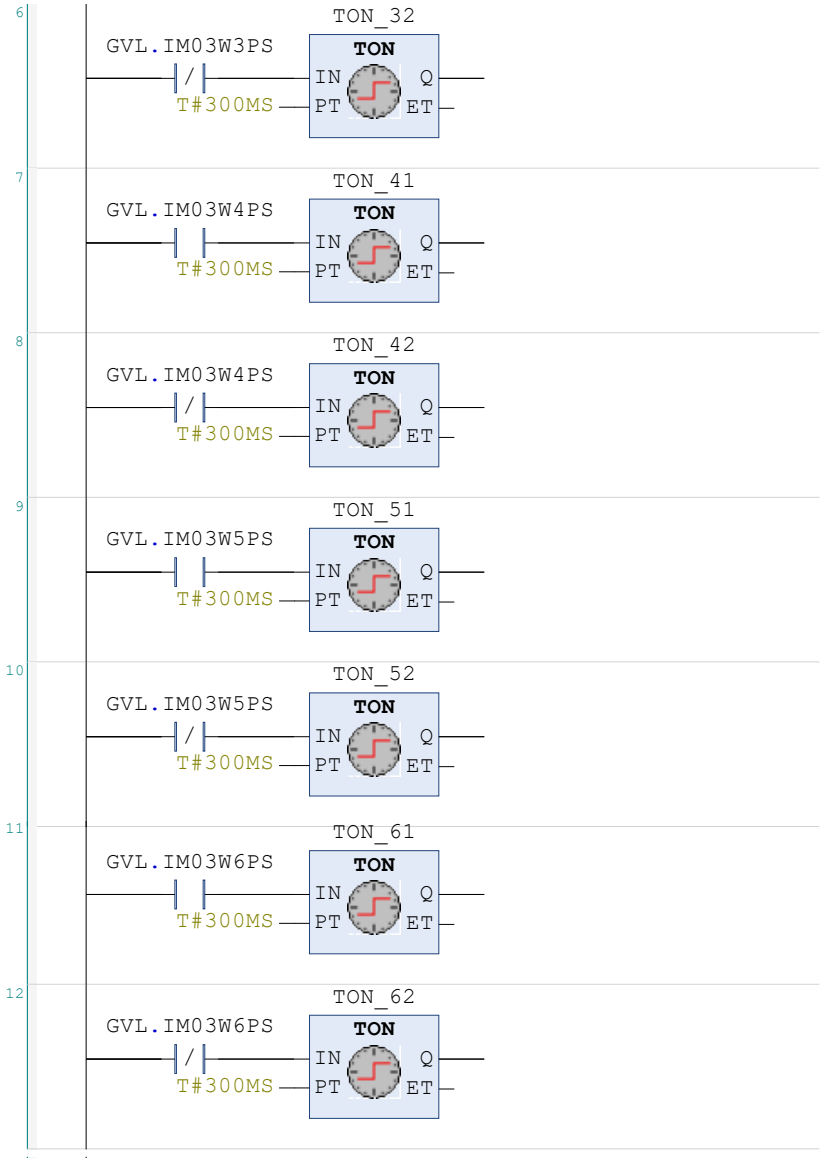


POU: M03WGT_LD

```
1   PROGRAM M03WGT_LD
2   VAR
3       TON_11 : TON ;
4       TON_12 : TON ;
5       TON_21 : TON ;
6       TON_22 : TON ;
7       TON_31 : TON ;
8       TON_32 : TON ;
9       TON_41 : TON ;
10      TON_42 : TON ;
11      TON_51 : TON ;
12      TON_52 : TON ;
13      TON_61 : TON ;
14      TON_62 : TON ;
15  END_VAR
16
17
```

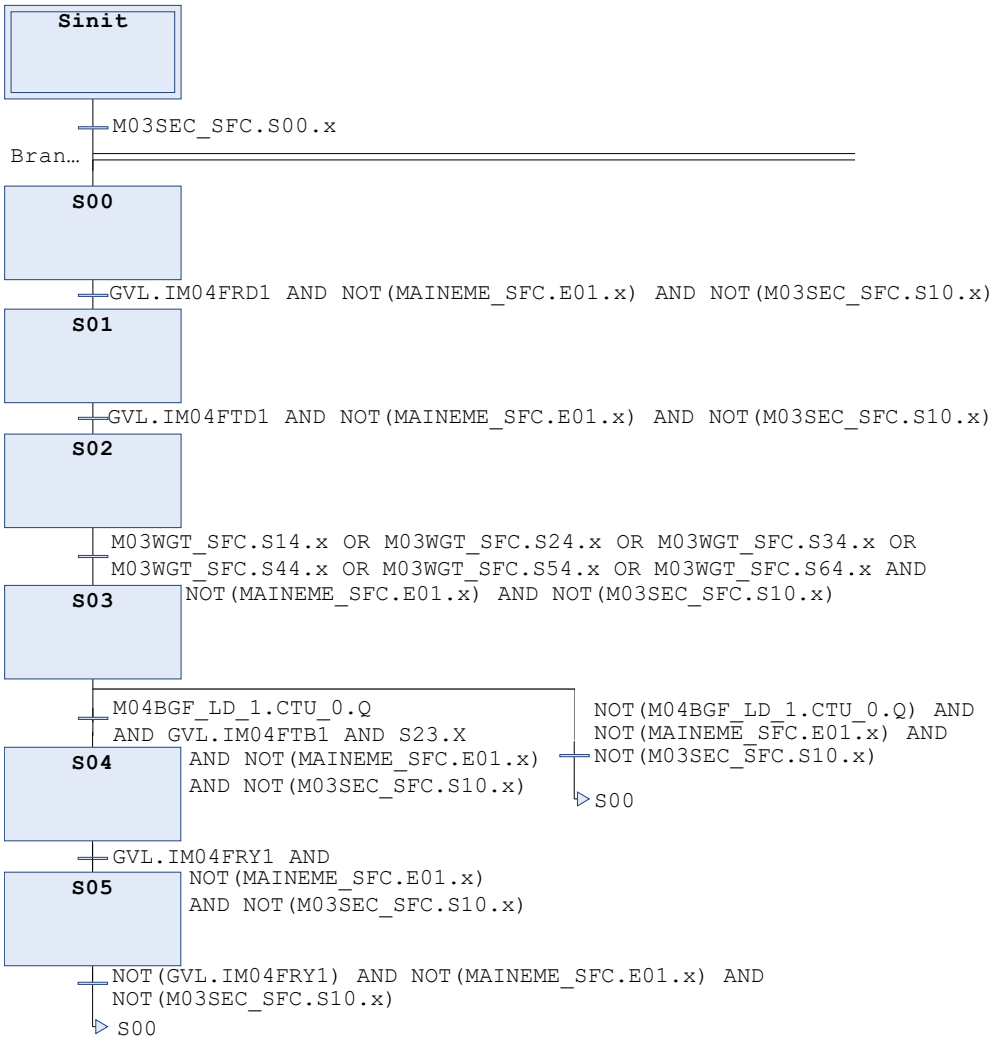


POU: M03WGT_LD



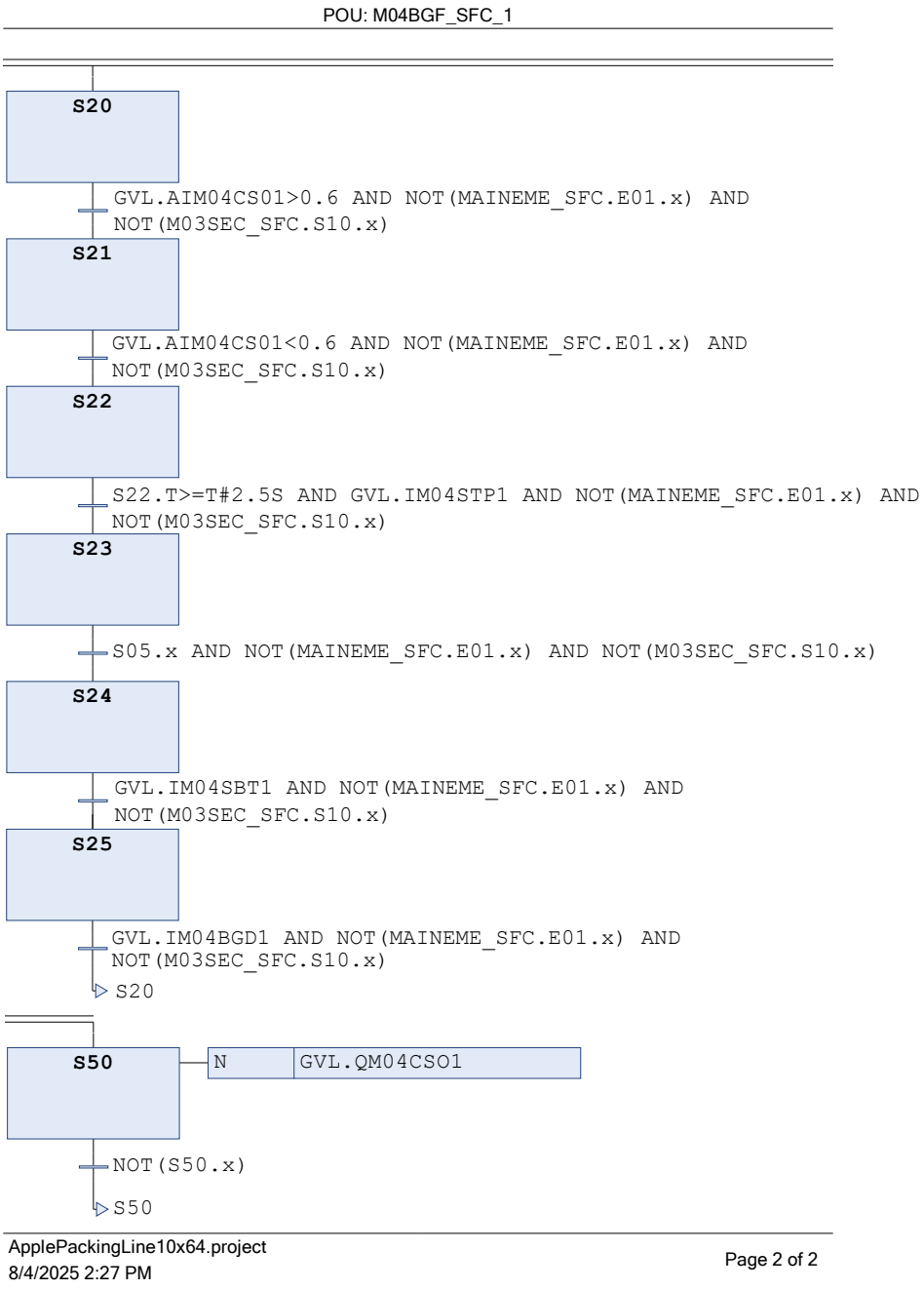
POU: M04BGF_SFC_1

```
1       PROGRAM M04BGF_SFC_1
2       VAR
3       END_VAR
4       VAR_INPUT
5           SFCInit : BOOL
6       END_VAR
7
```



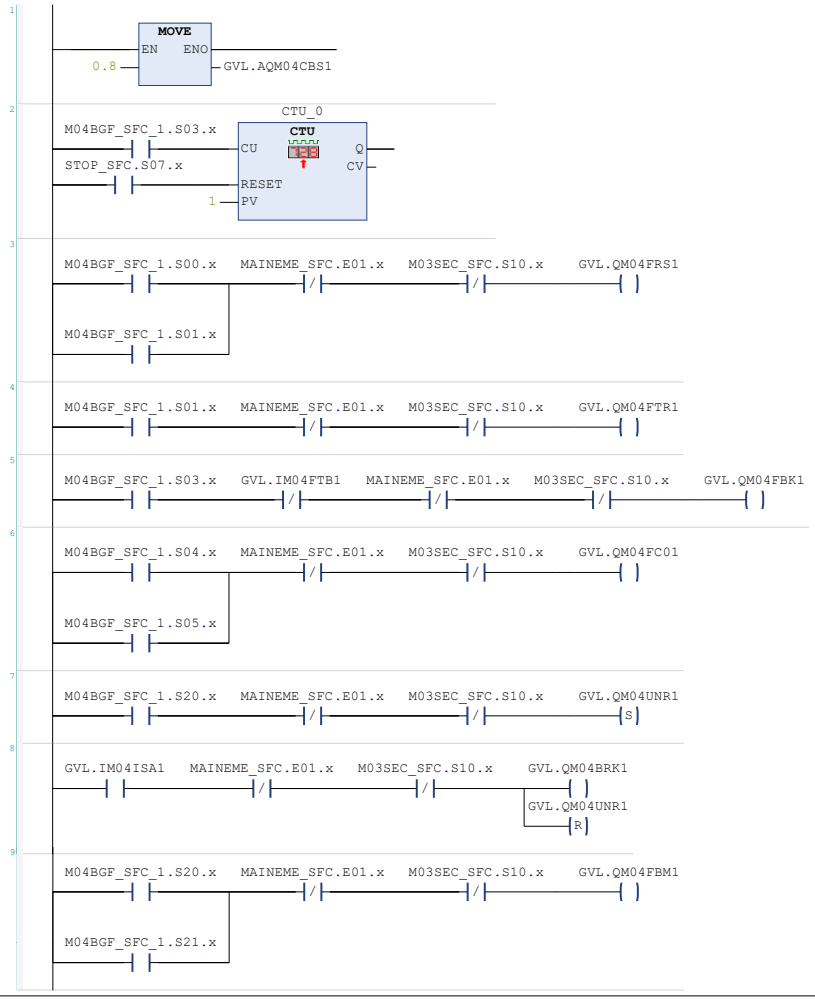
ApplePackingLine10x64.project
8/4/2025 2:27 PM

Page 1 of 2

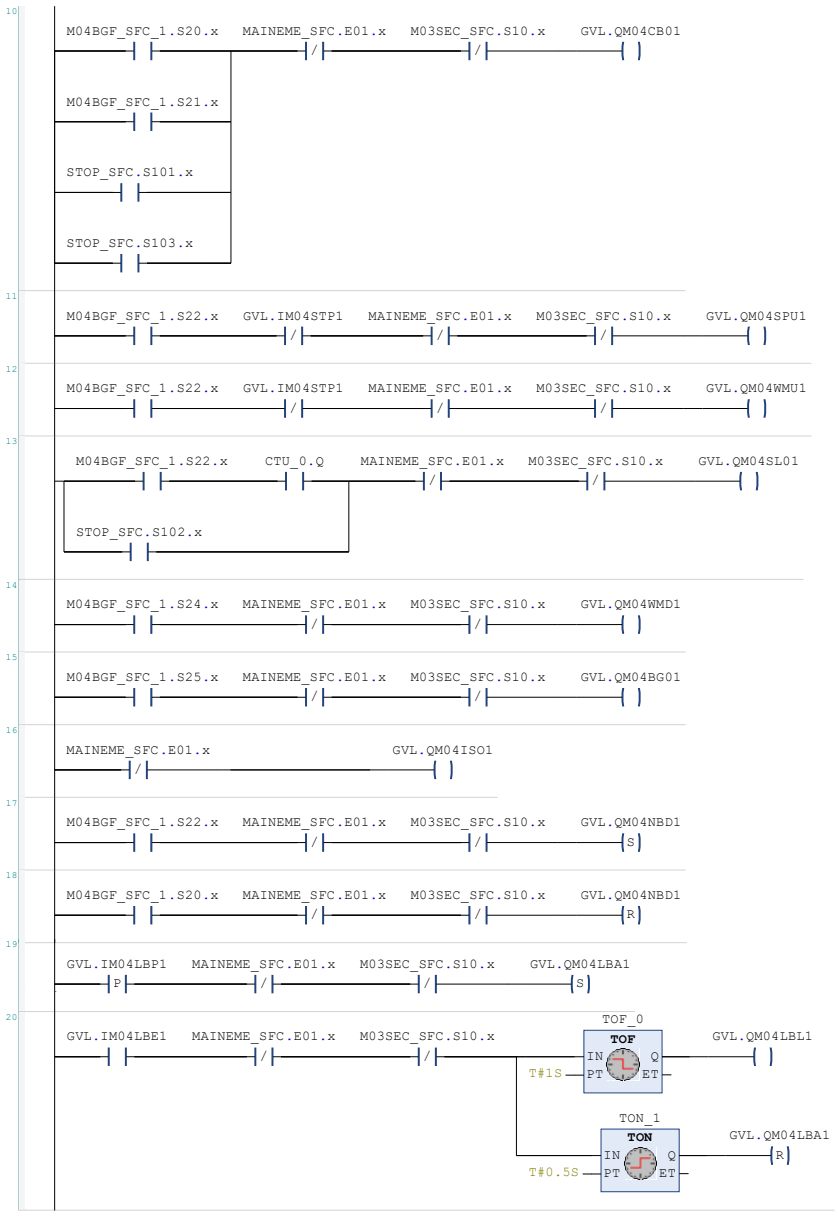

POU: M04BGF_SFC_1
S20
GVL.AIM04CS01>0.6 AND NOT(MAINEME_SFC.E01.x) AND
NOT(M03SEC_SFC.S10.x)
S21
GVL.AIM04CS01<0.6 AND NOT(MAINEME_SFC.E01.x) AND
NOT(M03SEC_SFC.S10.x)
S22
S22.T>=T#2.5S AND GVL.IM04STP1 AND NOT(MAINEME_SFC.E01.x) AND
NOT(M03SEC_SFC.S10.x)
S23
S05.x AND NOT(MAINEME_SFC.E01.x) AND NOT(M03SEC_SFC.S10.x)
S24
GVL.IM04SBT1 AND NOT(MAINEME_SFC.E01.x) AND
NOT(M03SEC_SFC.S10.x)
S25
GVL.IM04BGD1 AND NOT(MAINEME_SFC.E01.x) AND
NOT(M03SEC_SFC.S10.x)
S20
S50
N
GVL.QM04CSO1
NOT(S50.x)
S50
ApplePackingLine10x64.project
8/4/2025 2:27 PM
Page 2 of 2

POU: M04BGF_LD_1

```
1   PROGRAM M04BGF_LD_1
2   VAR
3       TON_0 : TON ;
4       CTU_0 : CTU ;
5       TOF_0 : TOF ;
6
7       TON_1 : TON ;
8   END_VAR
9
10
```

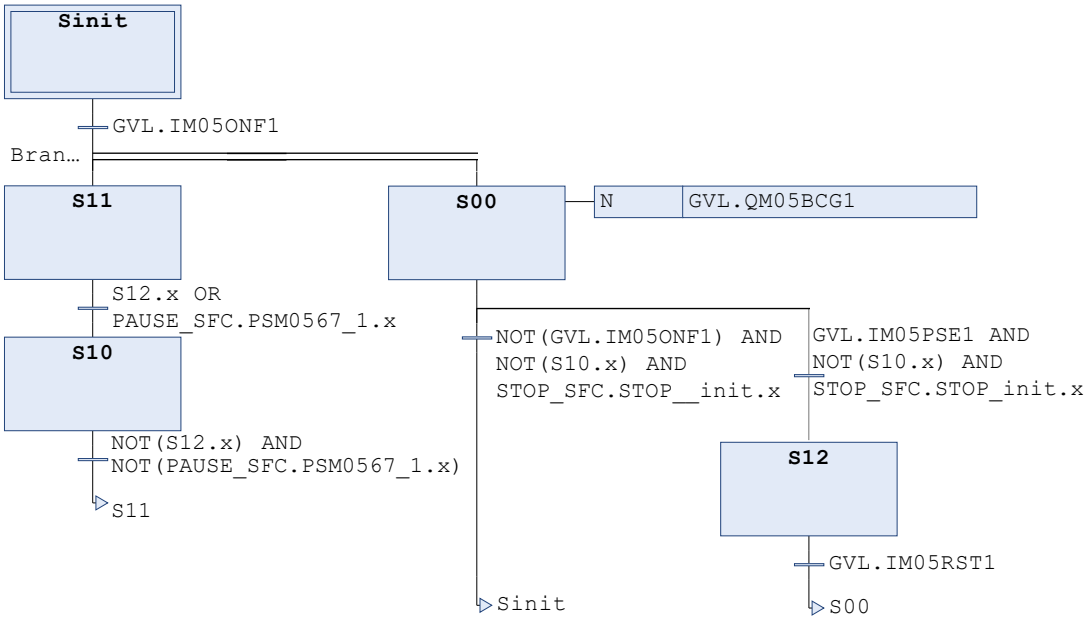


POU: M04BGF_LD_1



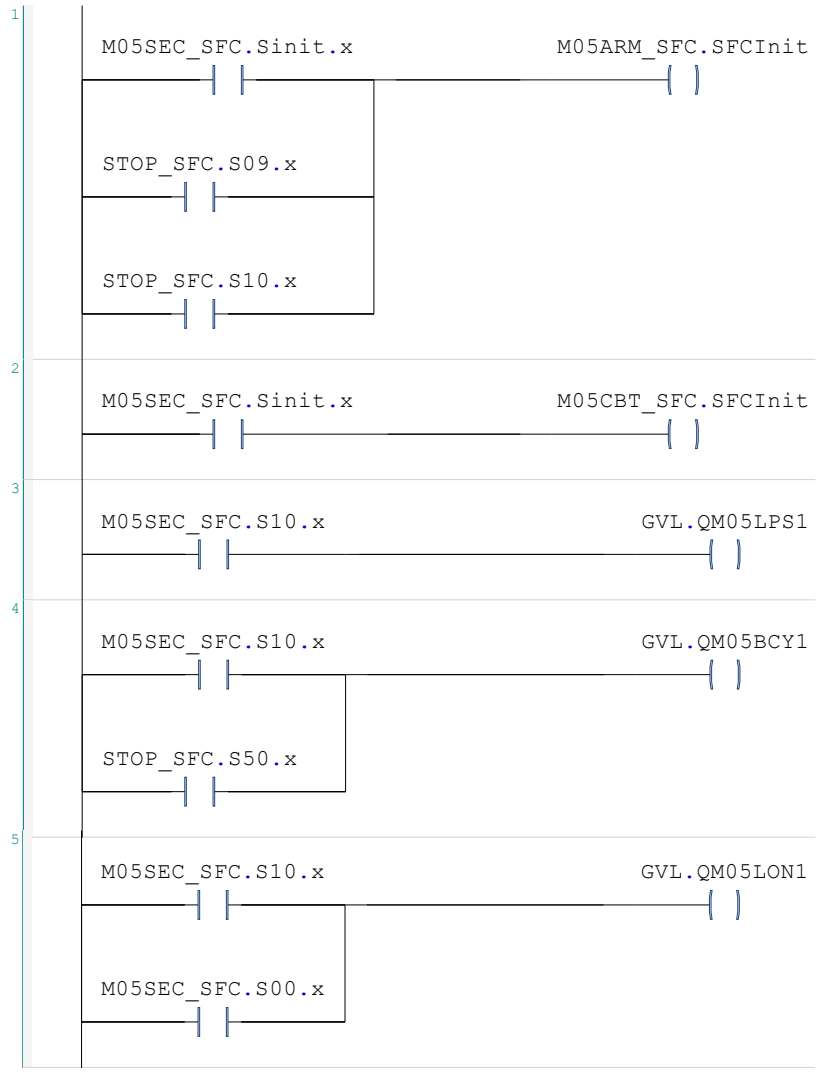
POU: M05SEC_SFC

```
1       PROGRAM M05SEC_SFC
2       VAR
3       END_VAR
4       VAR_INPUT
5           SFCInit : BOOL ;
6       END_VAR
7
```



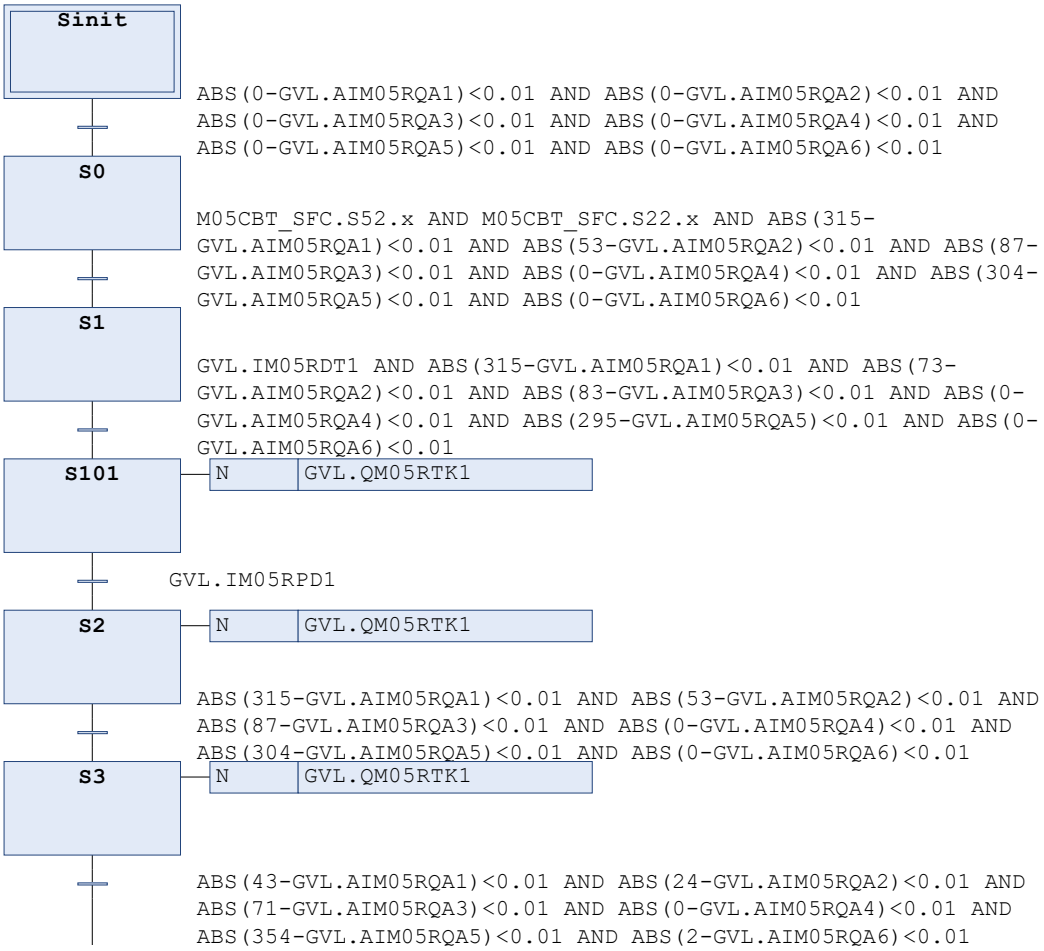
POU: M05SEC_LD

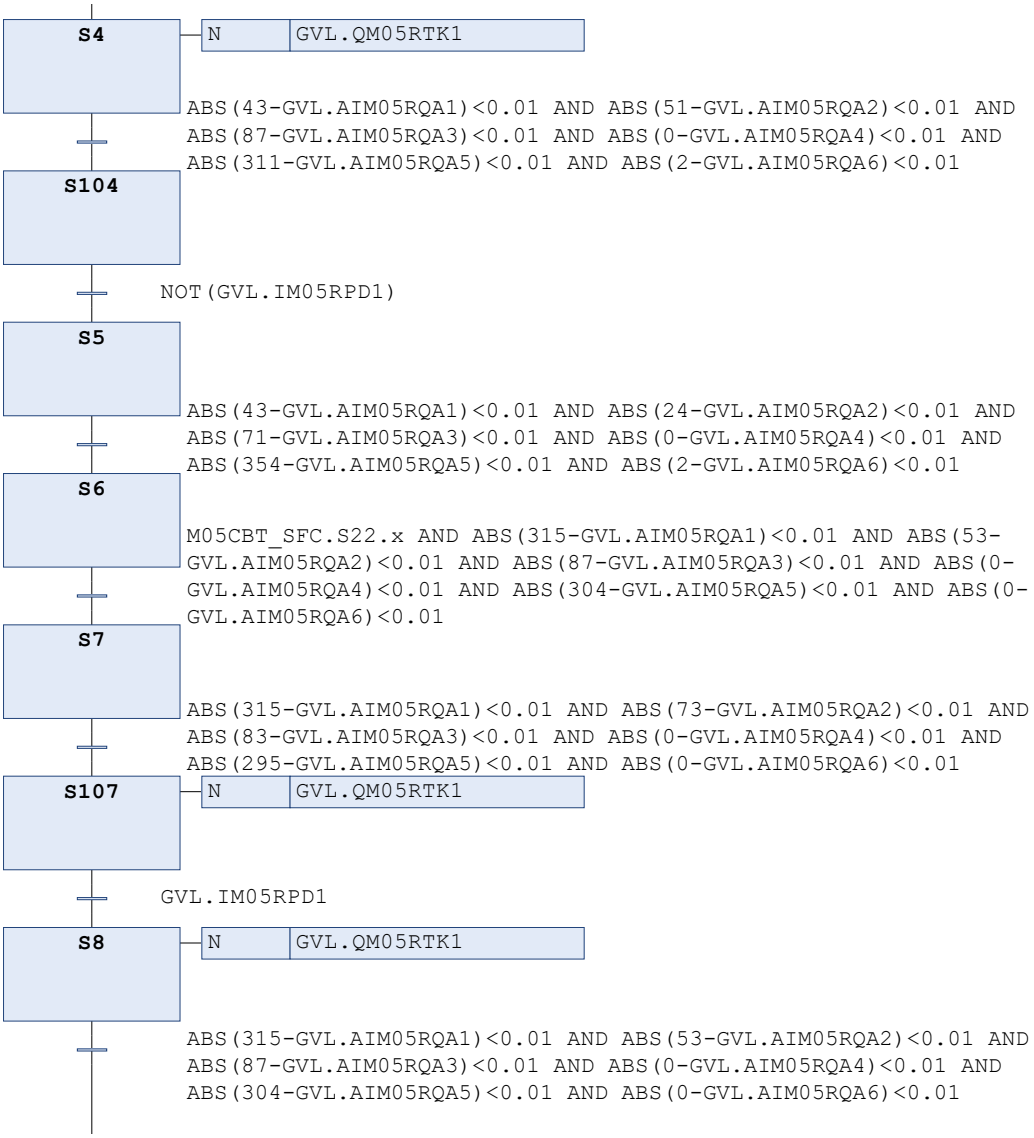
```
1   PROGRAM M05SEC_LD
2   VAR
3
4   END_VAR
5
6
```

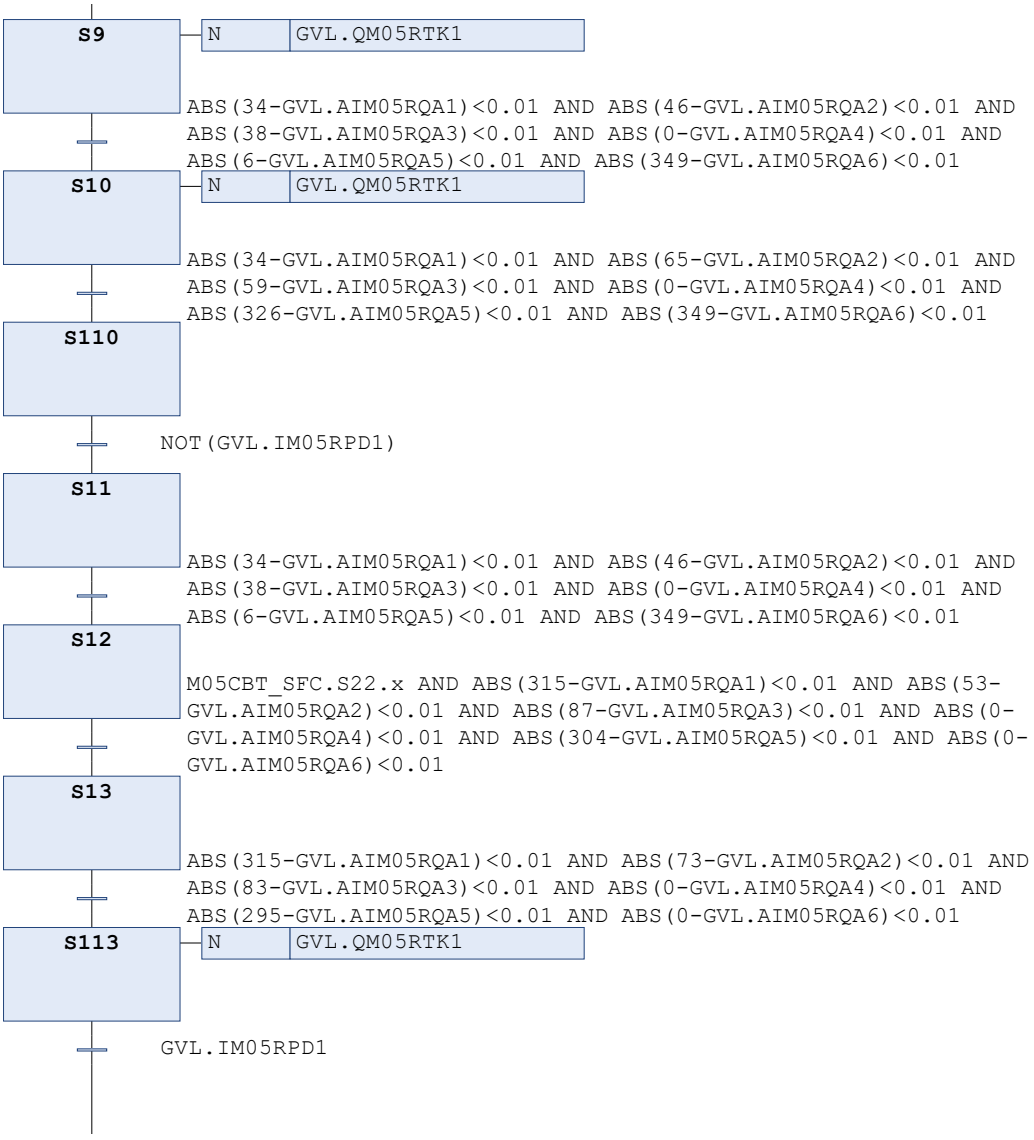


POU: M05ARM_SFC

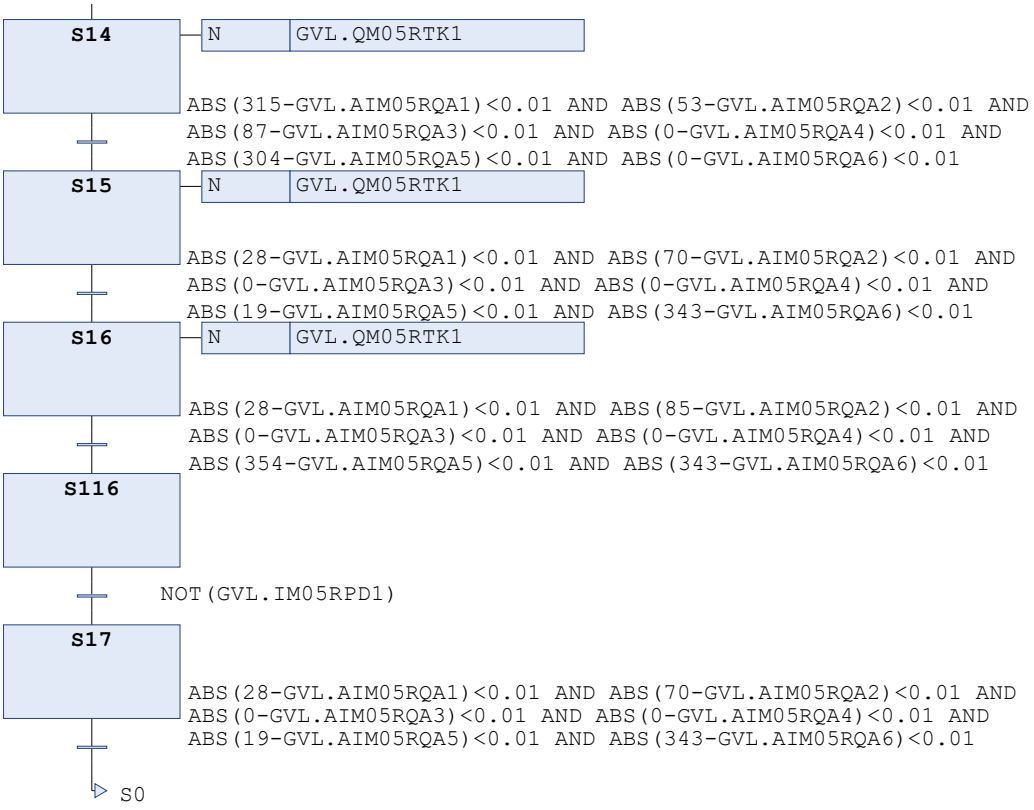
```
1   PROGRAM M05ARM_SFC
2   VAR
3   END_VAR
4   VAR_INPUT
5       SFCInit : BOOL ;
6   END_VAR
7
```




S4
N
GVL.QM05RTK1
ABS(43-GVL.AIM05RQA1)<0.01 AND ABS(51-GVL.AIM05RQA2)<0.01 AND
ABS(87-GVL.AIM05RQA3)<0.01 AND ABS(0-GVL.AIM05RQA4)<0.01 AND
ABS(311-GVL.AIM05RQA5)<0.01 AND ABS(2-GVL.AIM05RQA6)<0.01
S104
NOT(GVL.IM05RPD1)
S5
ABS(43-GVL.AIM05RQA1)<0.01 AND ABS(24-GVL.AIM05RQA2)<0.01 AND
ABS(71-GVL.AIM05RQA3)<0.01 AND ABS(0-GVL.AIM05RQA4)<0.01 AND
ABS(354-GVL.AIM05RQA5)<0.01 AND ABS(2-GVL.AIM05RQA6)<0.01
S6
M05CBT_SFC.S22.x AND ABS(315-GVL.AIM05RQA1)<0.01 AND ABS(53-
GVL.AIM05RQA2)<0.01 AND ABS(87-GVL.AIM05RQA3)<0.01 AND ABS(0-
GVL.AIM05RQA4)<0.01 AND ABS(304-GVL.AIM05RQA5)<0.01 AND ABS(0-
GVL.AIM05RQA6)<0.01
S7
ABS(315-GVL.AIM05RQA1)<0.01 AND ABS(73-GVL.AIM05RQA2)<0.01 AND
ABS(83-GVL.AIM05RQA3)<0.01 AND ABS(0-GVL.AIM05RQA4)<0.01 AND
ABS(295-GVL.AIM05RQA5)<0.01 AND ABS(0-GVL.AIM05RQA6)<0.01
S107
N
GVL.QM05RTK1
GVL.IM05RPD1
S8
N
GVL.QM05RTK1
ABS(315-GVL.AIM05RQA1)<0.01 AND ABS(53-GVL.AIM05RQA2)<0.01 AND
ABS(87-GVL.AIM05RQA3)<0.01 AND ABS(0-GVL.AIM05RQA4)<0.01 AND
ABS(304-GVL.AIM05RQA5)<0.01 AND ABS(0-GVL.AIM05RQA6)<0.01

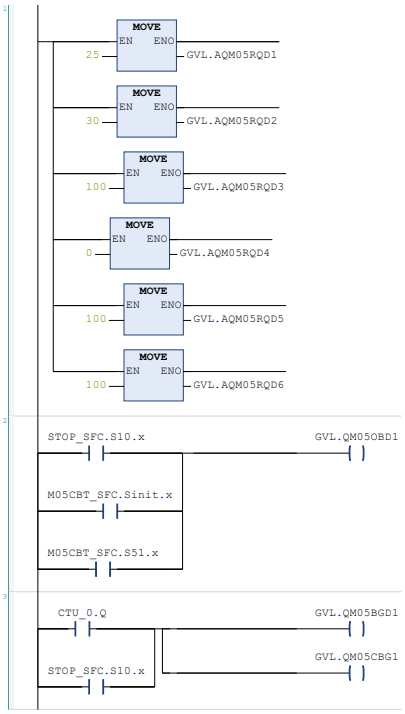

S9
N
GVL.QM05RTK1
ABS(34-GVL.AIM05RQA1)<0.01 AND ABS(46-GVL.AIM05RQA2)<0.01 AND
ABS(38-GVL.AIM05RQA3)<0.01 AND ABS(0-GVL.AIM05RQA4)<0.01 AND
ABS(6-GVL.AIM05RQA5)<0.01 AND ABS(349-GVL.AIM05RQA6)<0.01
S10
N
GVL.QM05RTK1
ABS(34-GVL.AIM05RQA1)<0.01 AND ABS(65-GVL.AIM05RQA2)<0.01 AND
ABS(59-GVL.AIM05RQA3)<0.01 AND ABS(0-GVL.AIM05RQA4)<0.01 AND
ABS(326-GVL.AIM05RQA5)<0.01 AND ABS(349-GVL.AIM05RQA6)<0.01
S110
NOT(GVL.IM05RPD1)
S11
ABS(34-GVL.AIM05RQA1)<0.01 AND ABS(46-GVL.AIM05RQA2)<0.01 AND
ABS(38-GVL.AIM05RQA3)<0.01 AND ABS(0-GVL.AIM05RQA4)<0.01 AND
ABS(6-GVL.AIM05RQA5)<0.01 AND ABS(349-GVL.AIM05RQA6)<0.01
S12
M05CBT_SFC.S22.x AND ABS(315-GVL.AIM05RQA1)<0.01 AND ABS(53-
GVL.AIM05RQA2)<0.01 AND ABS(87-GVL.AIM05RQA3)<0.01 AND ABS(0-
GVL.AIM05RQA4)<0.01 AND ABS(304-GVL.AIM05RQA5)<0.01 AND ABS(0-
GVL.AIM05RQA6)<0.01
S13
ABS(315-GVL.AIM05RQA1)<0.01 AND ABS(73-GVL.AIM05RQA2)<0.01 AND
ABS(83-GVL.AIM05RQA3)<0.01 AND ABS(0-GVL.AIM05RQA4)<0.01 AND
ABS(295-GVL.AIM05RQA5)<0.01 AND ABS(0-GVL.AIM05RQA6)<0.01
S113
N
GVL.QM05RTK1
GVL.IM05RPD1

POU: M05ARM_SFC

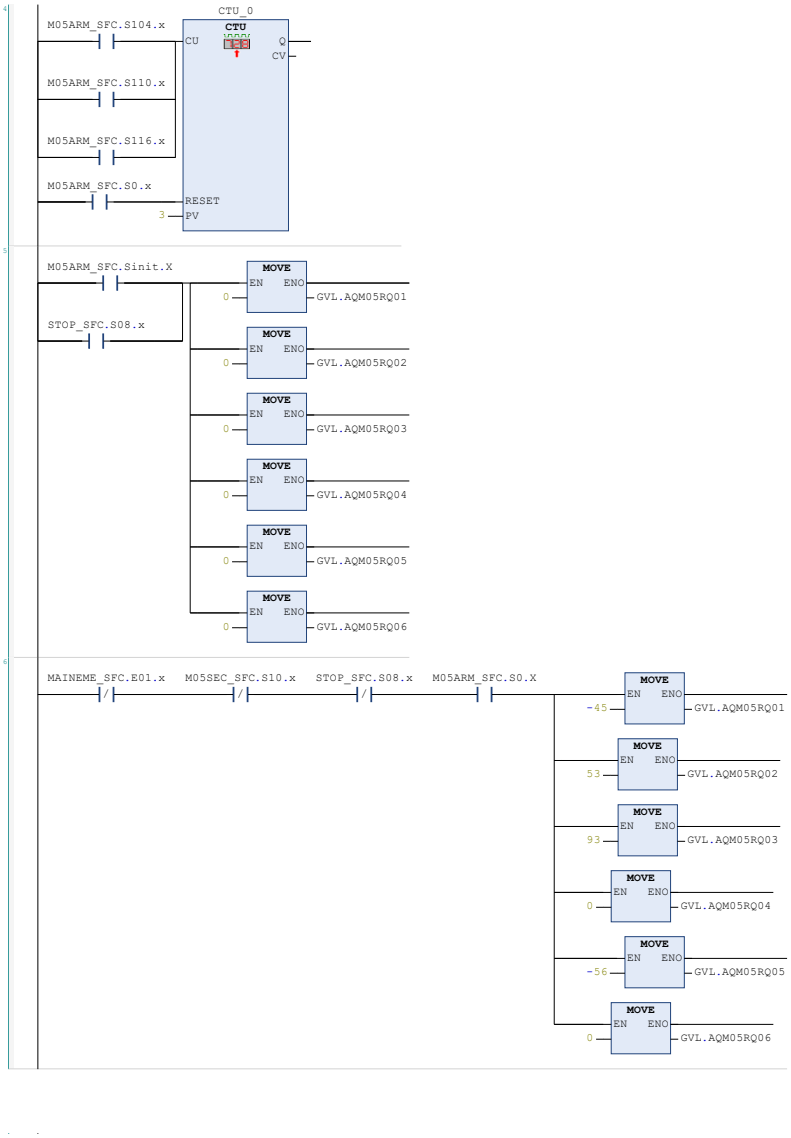


POU: M05ARM_LD

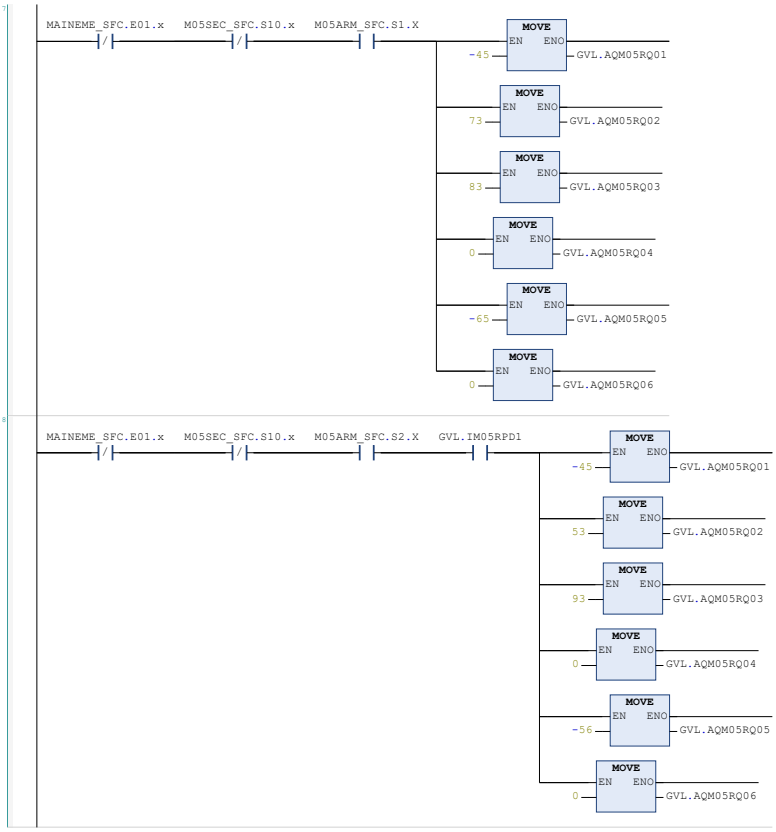
```
1       PROGRAM  M05ARM_LD
2       VAR
3           CTU_0 :  CTU ;
4           TOF_0 :  TOF ;
5           BAGS :  CTU ;
6       END_VAR
7
8
```



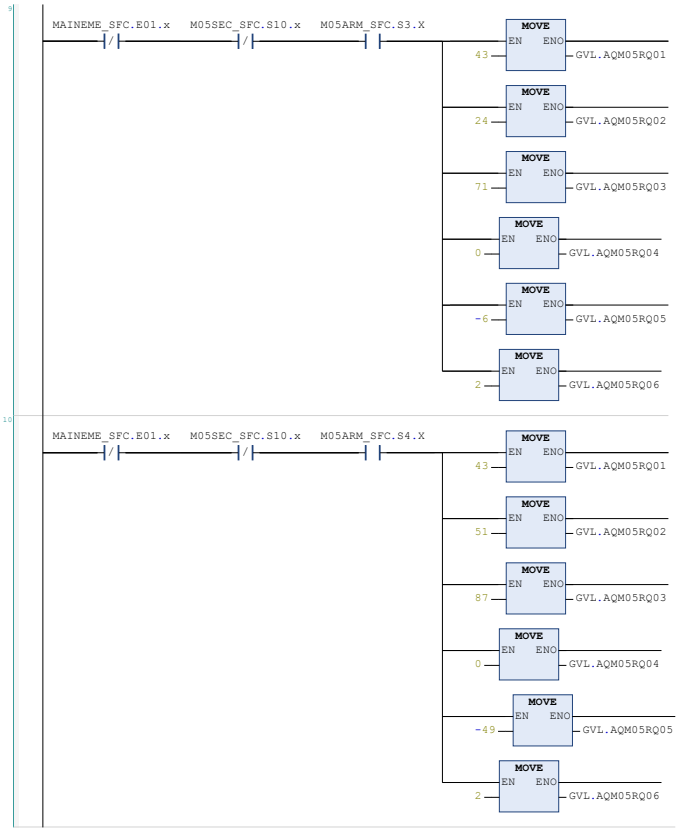
POU: M05ARM_LD



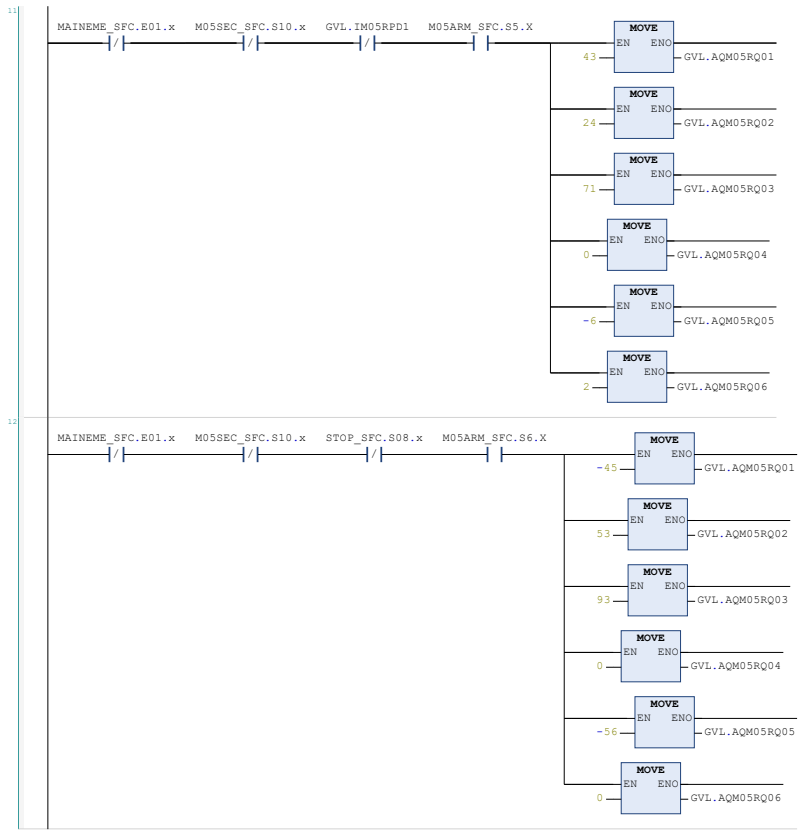
POU: M05ARM_LD



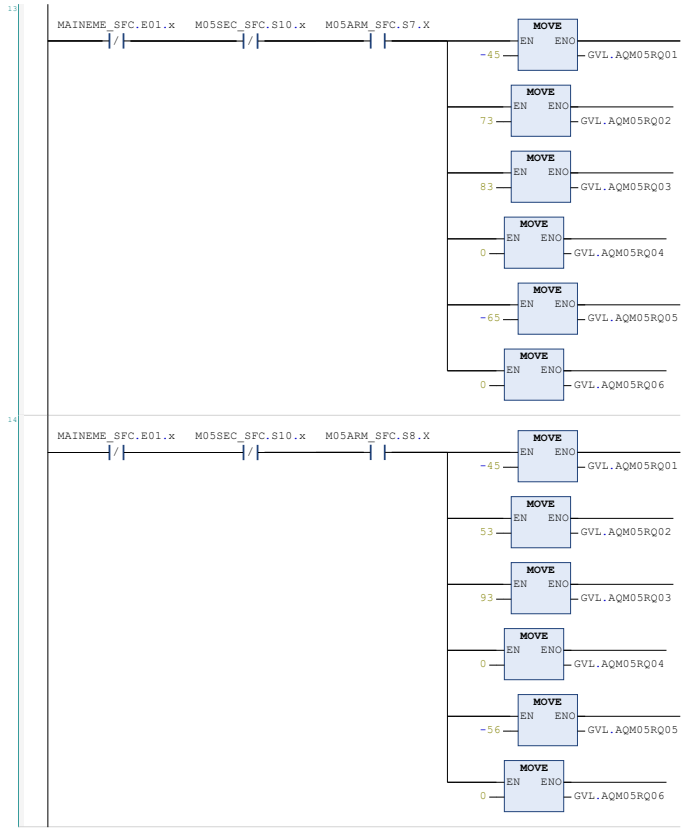
POU: M05ARM_LD



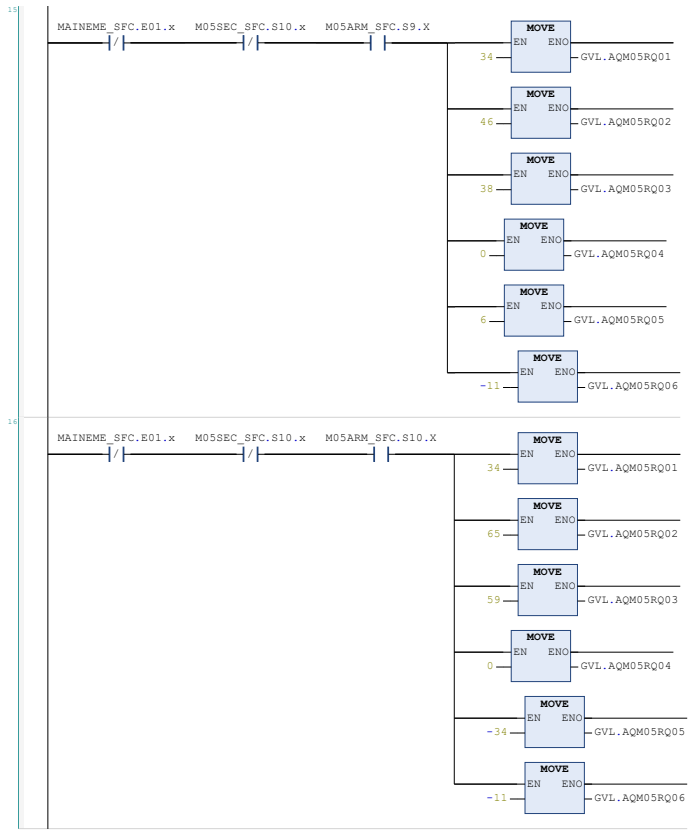
POU: M05ARM_LD



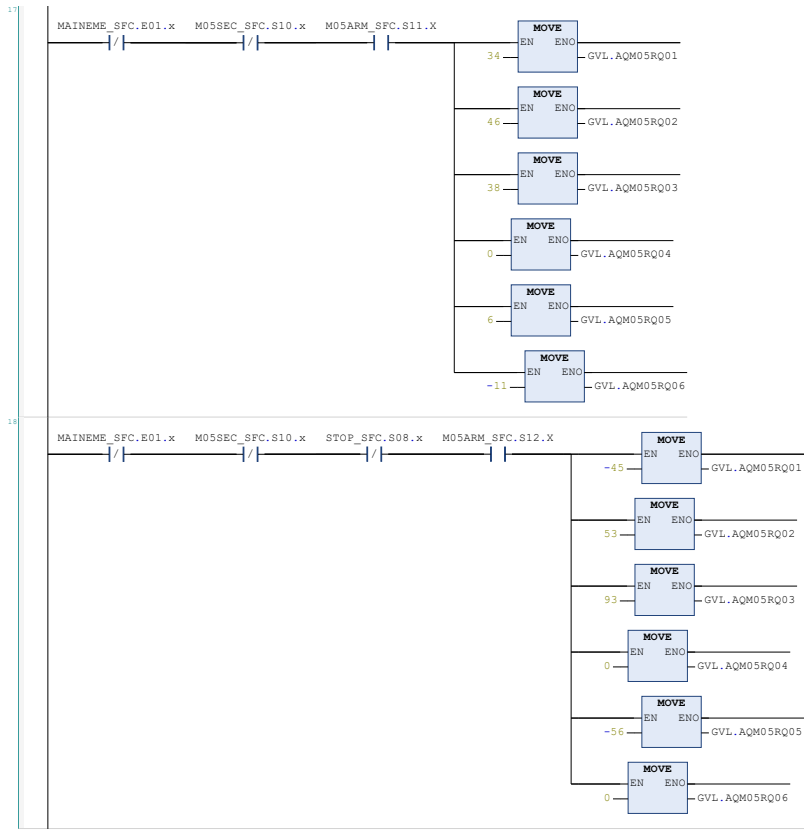
POU: M05ARM_LD



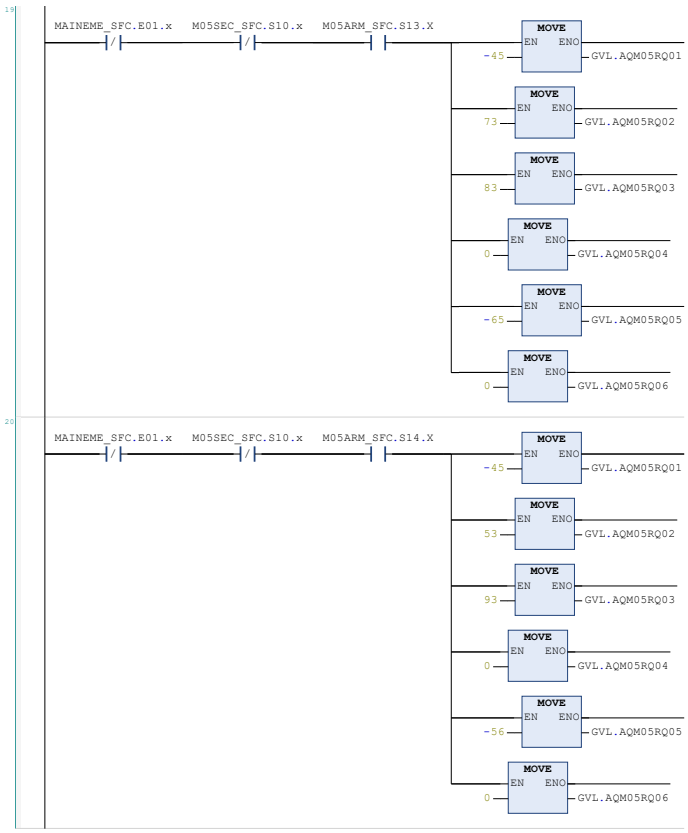
POU: M05ARM_LD



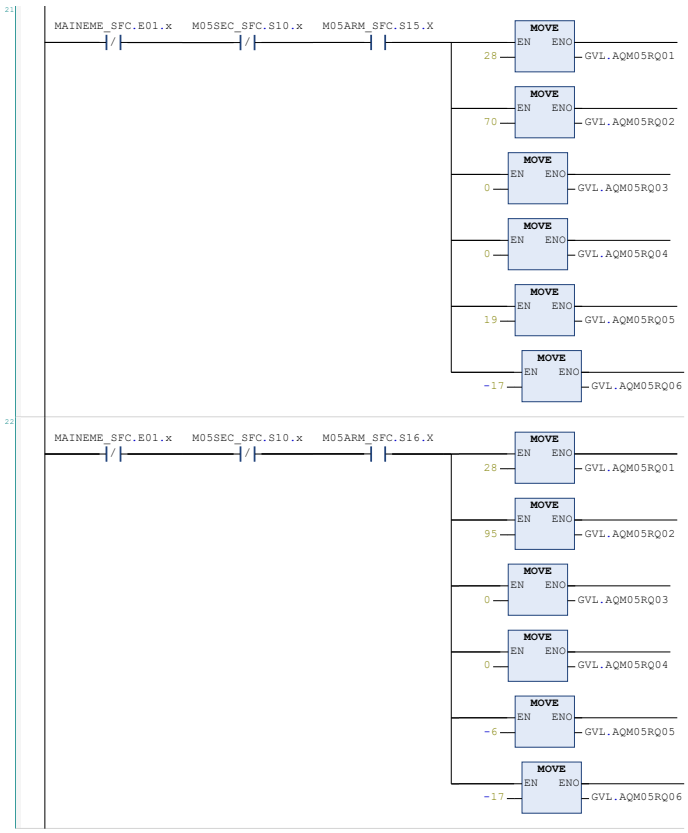
POU: M05ARM_LD



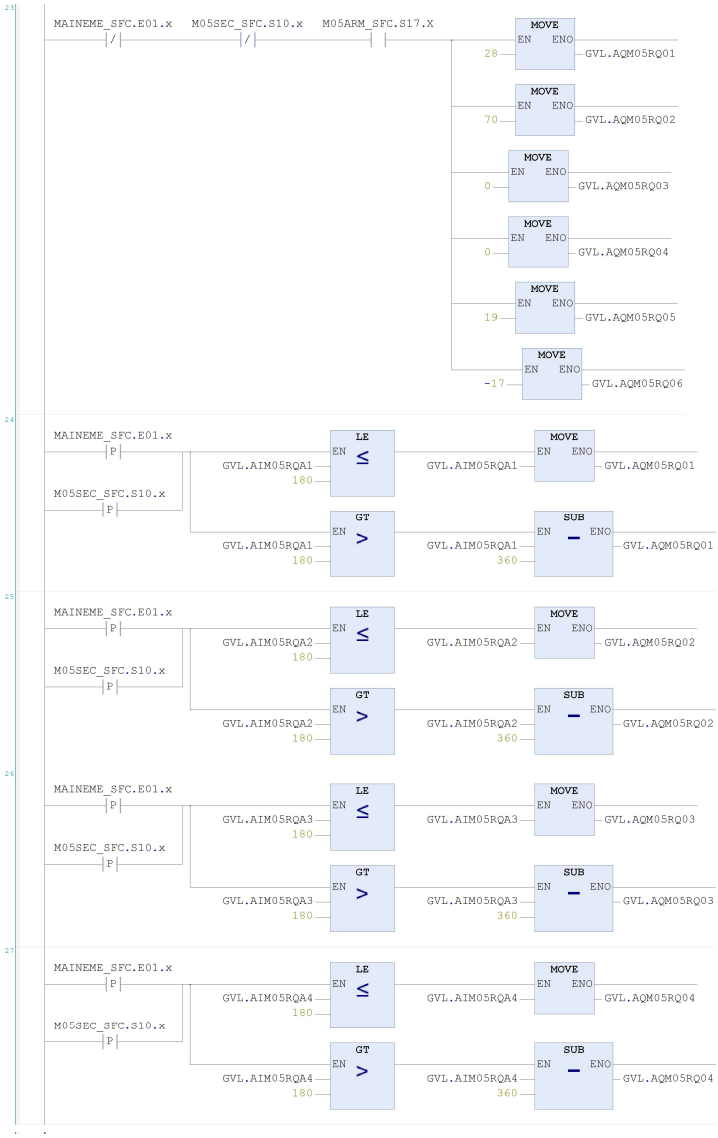
POU: M05ARM_LD



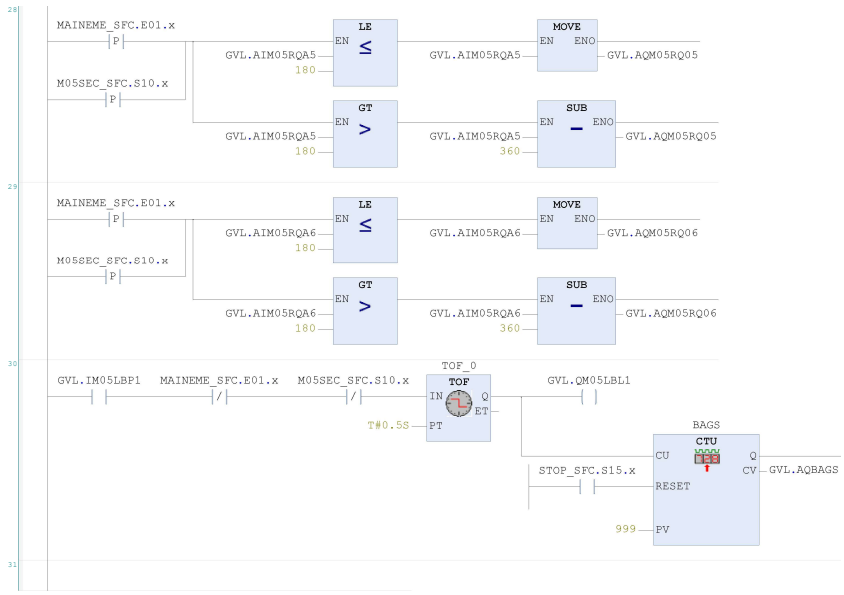
POU: M05ARM_LD



POU: M05ARM_LD

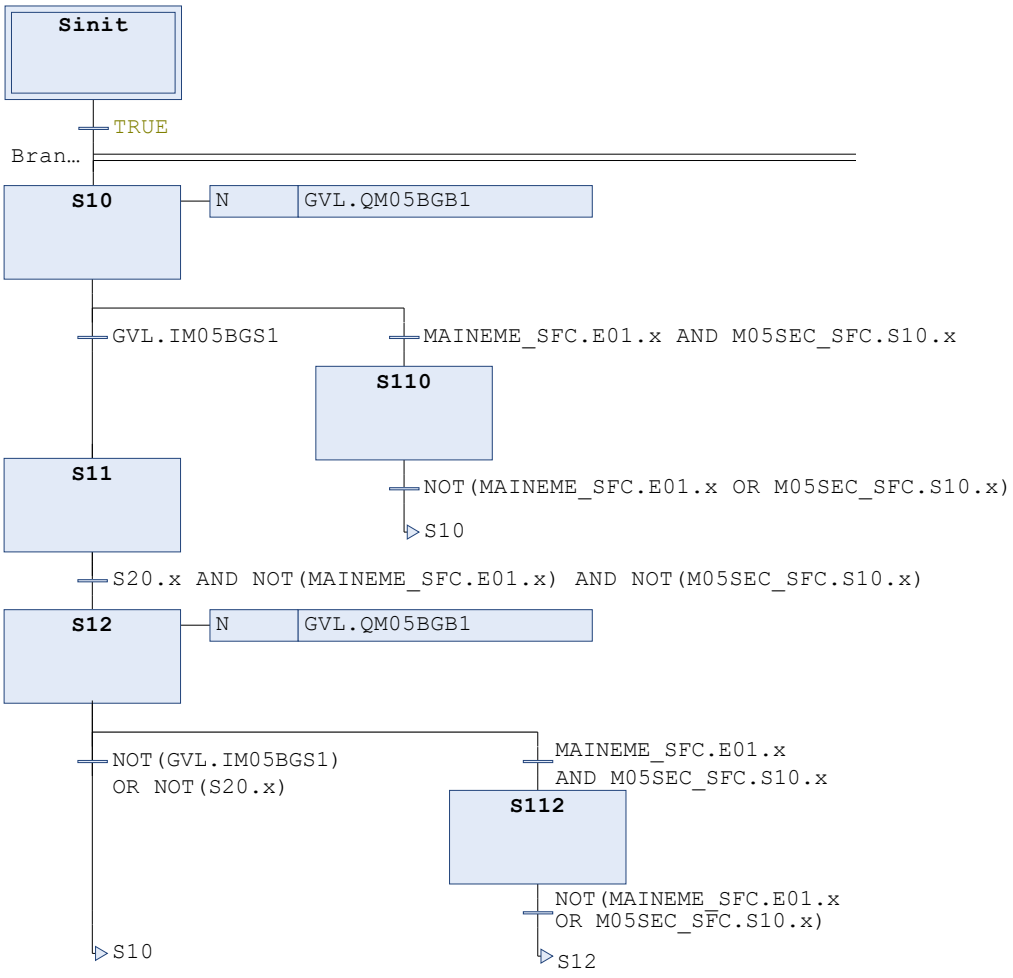


POU: M05ARM_LD

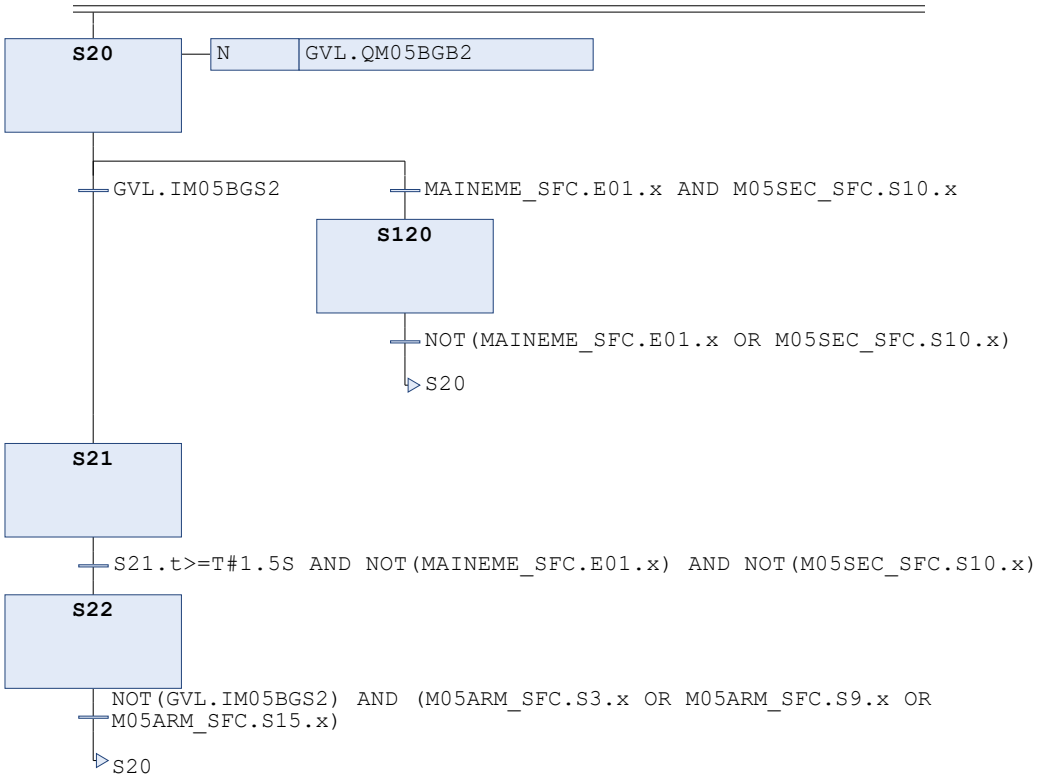


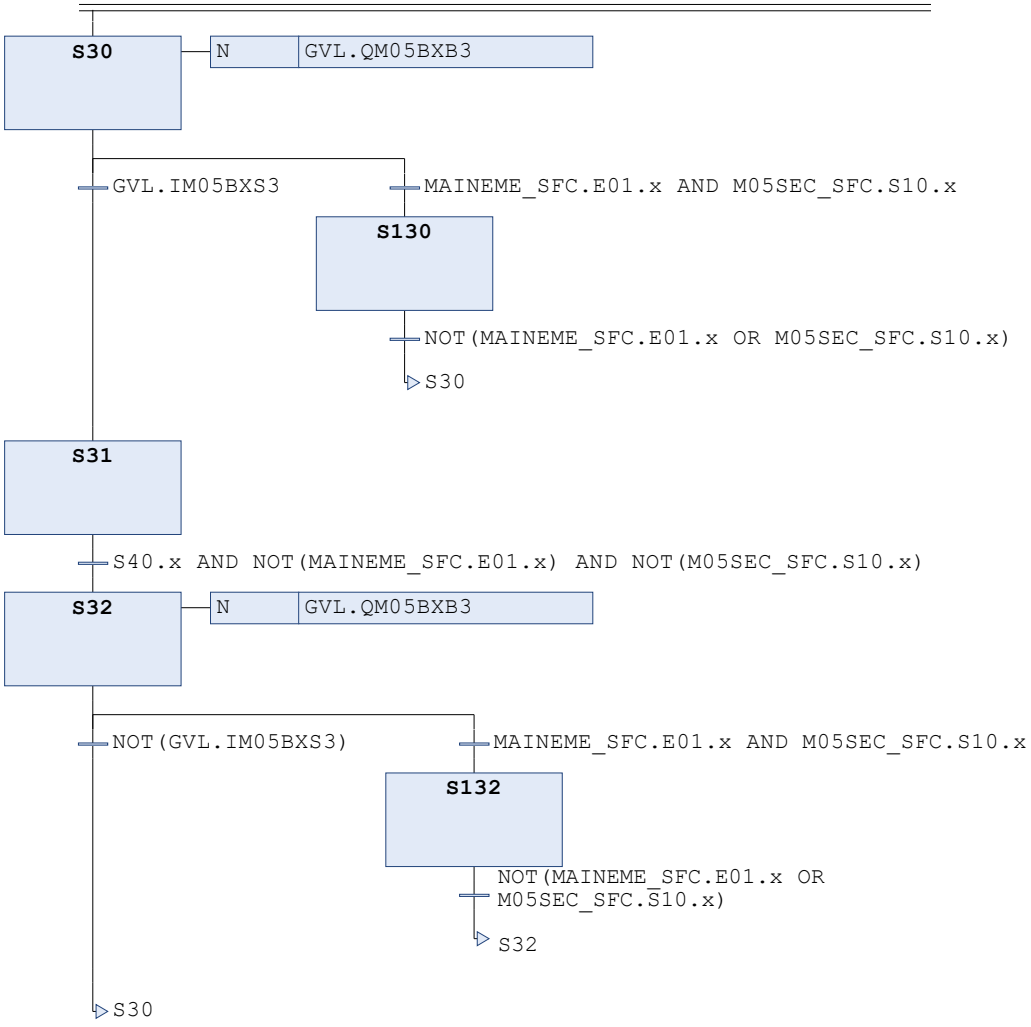
POU: M05CBT_SFC

```
1       PROGRAM M05CBT_SFC
2       VAR
3       END_VAR
4       VAR_INPUT
5           SFCInit : BOOL ;
6       END_VAR
7
```

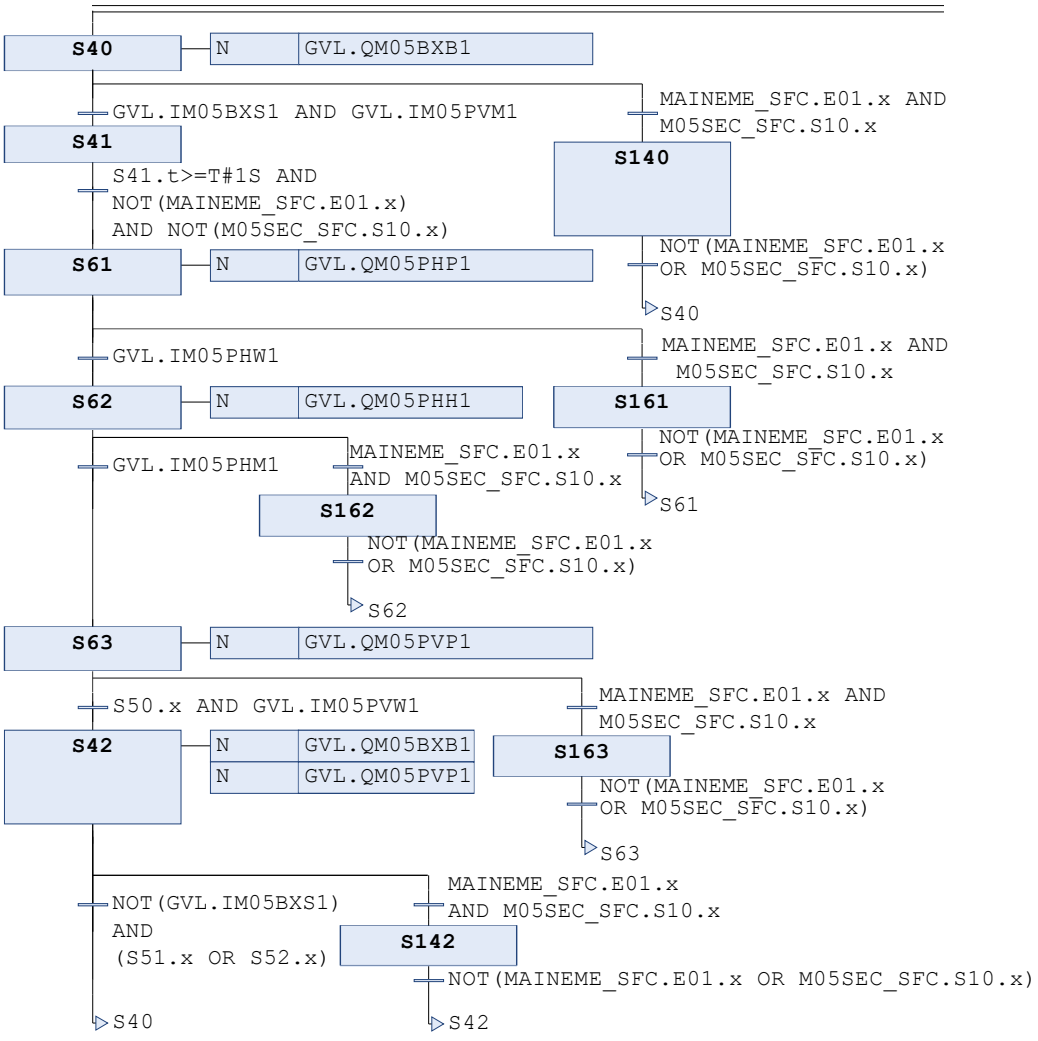


POU: M05CBT_SFC

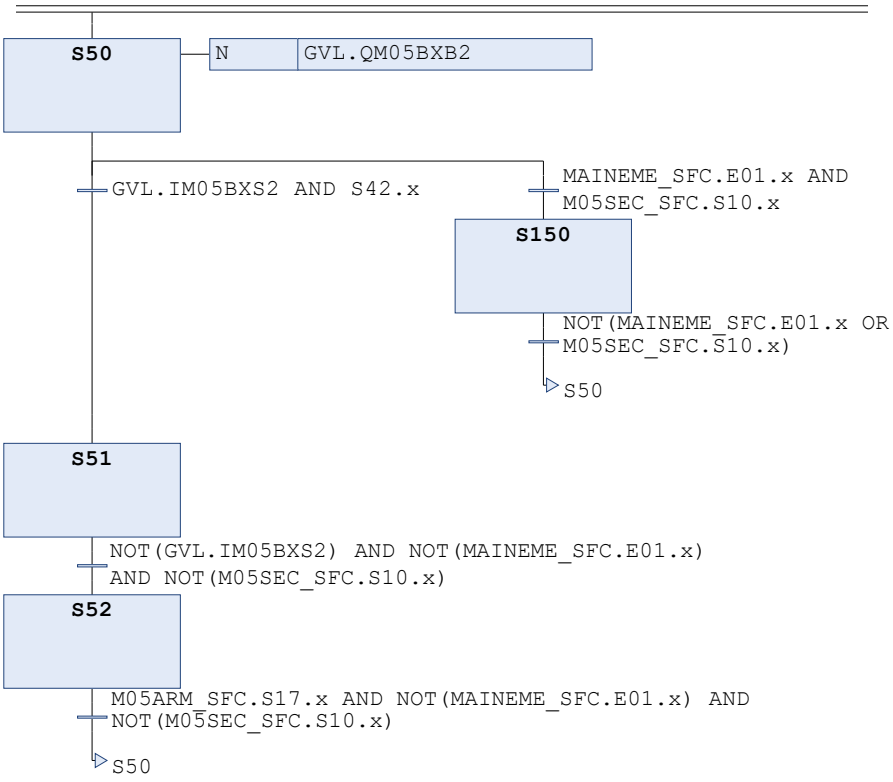


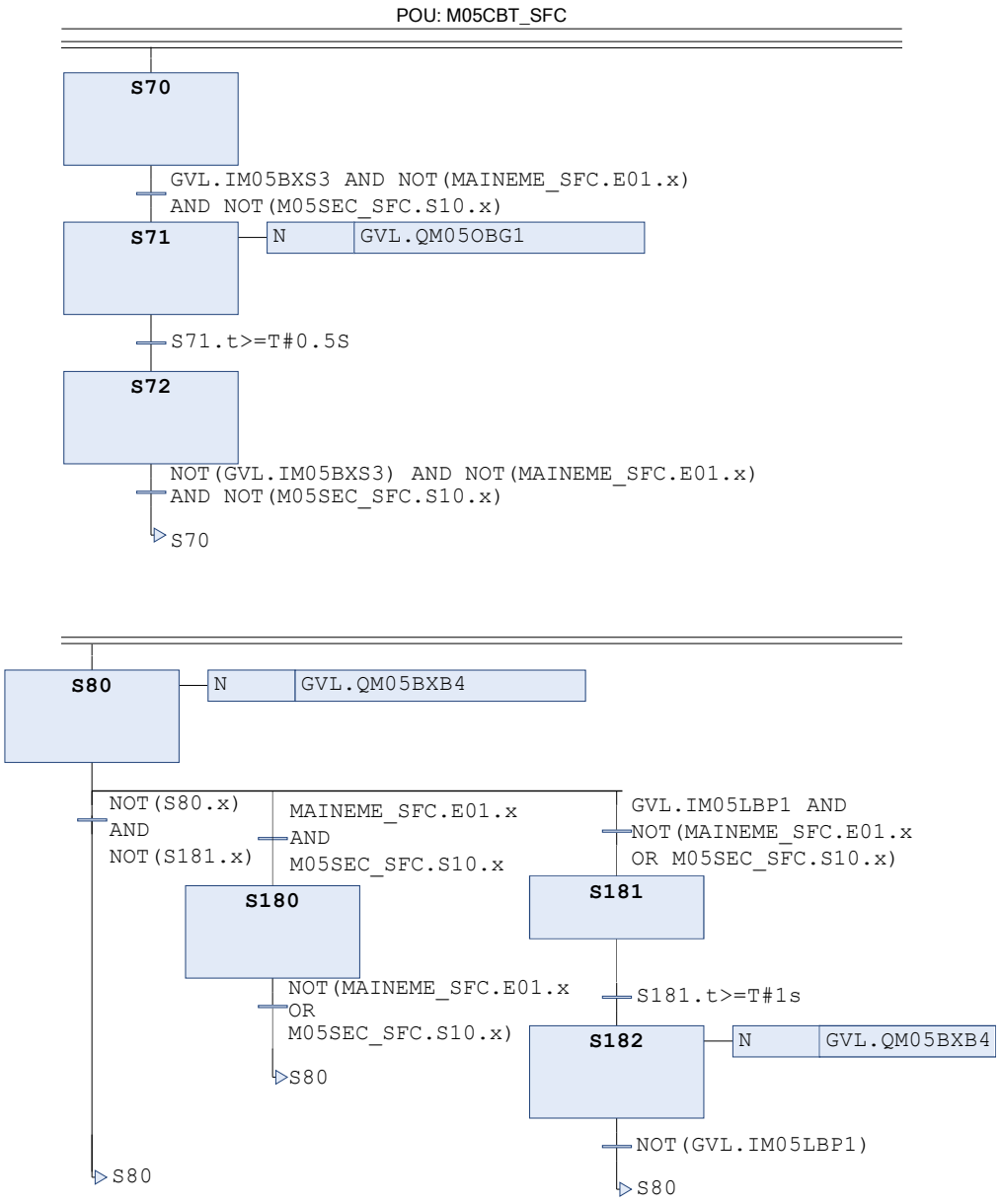

S30
N
GVL.QM05BXB3
GVL.IM05BXS3
MAINEME_SFC.E01.x AND M05SEC_SFC.S10.x
S130
NOT(MAINEME_SFC.E01.x OR M05SEC_SFC.S10.x)
S30
S31
S40.x AND NOT(MAINEME_SFC.E01.x) AND NOT(M05SEC_SFC.S10.x)
S32
N
GVL.QM05BXB3
NOT(GVL.IM05BXS3)
MAINEME_SFC.E01.x AND M05SEC_SFC.S10.x
S132
NOT(MAINEME_SFC.E01.x OR
M05SEC_SFC.S10.x)
S32
S30

POU: M05CBT_SFC



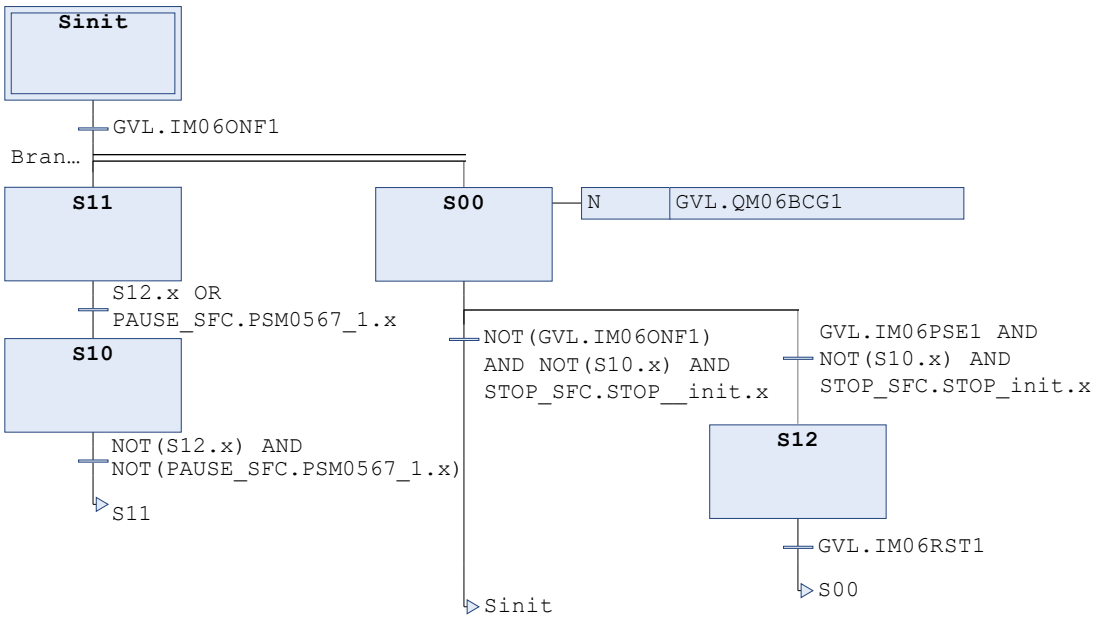
POU: M05CBT_SFC




POU: M05CBT_SFC
S70
GVL.IM05BXS3 AND NOT(MAINEME_SFC.E01.x)
AND NOT(M05SEC_SFC.S10.x)
S71
N
GVL.QM05OBG1
S71.t>=T#0.5S
S72
NOT(GVL.IM05BXS3) AND NOT(MAINEME_SFC.E01.x)
AND NOT(M05SEC_SFC.S10.x)
S70
S80
N
GVL.QM05BXB4
NOT(S80.x)
AND
NOT(S181.x)
MAINEME_SFC.E01.x
AND
M05SEC_SFC.S10.x
GVL.IM05LBP1 AND
NOT(MAINEME_SFC.E01.x
OR M05SEC_SFC.S10.x)
S180
S181
NOT(MAINEME_SFC.E01.x
OR
M05SEC_SFC.S10.x)
S80
S181.t>=T#1s
S182
N
GVL.QM05BXB4
NOT(GVL.IM05LBP1)
S80
S80

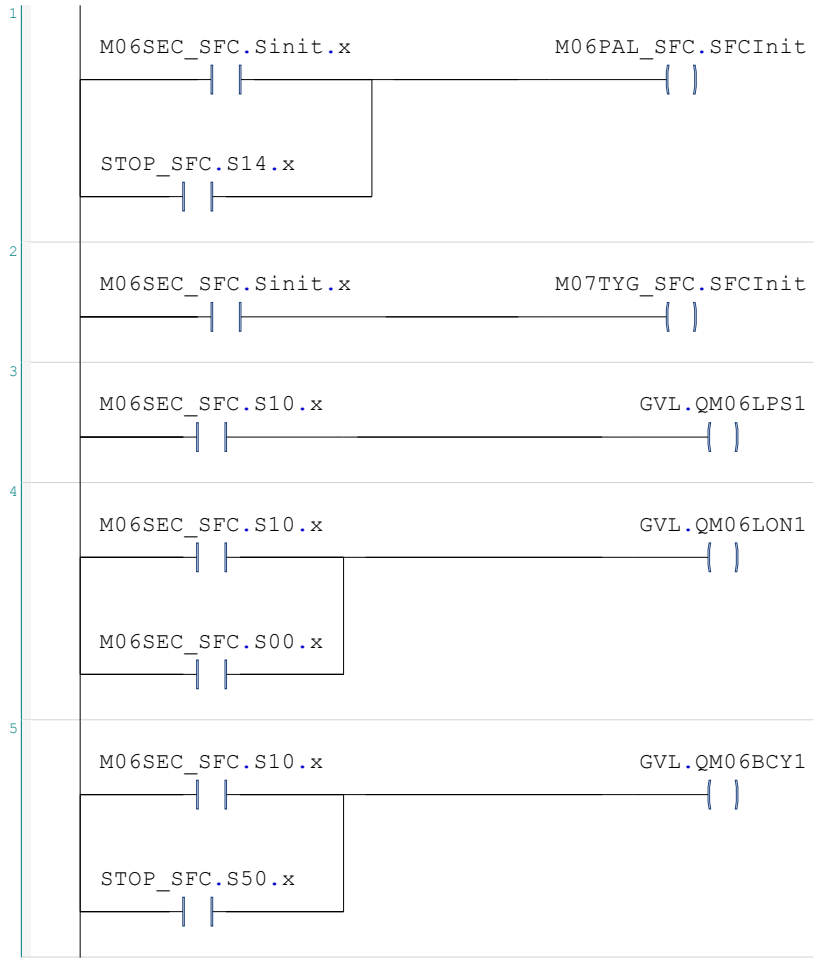
POU: M06SEC_SFC

```
1       PROGRAM M06SEC_SFC
2       VAR
3       END_VAR
4       VAR_INPUT
5           SFCInit : BOOL ;
6       END_VAR
7
```



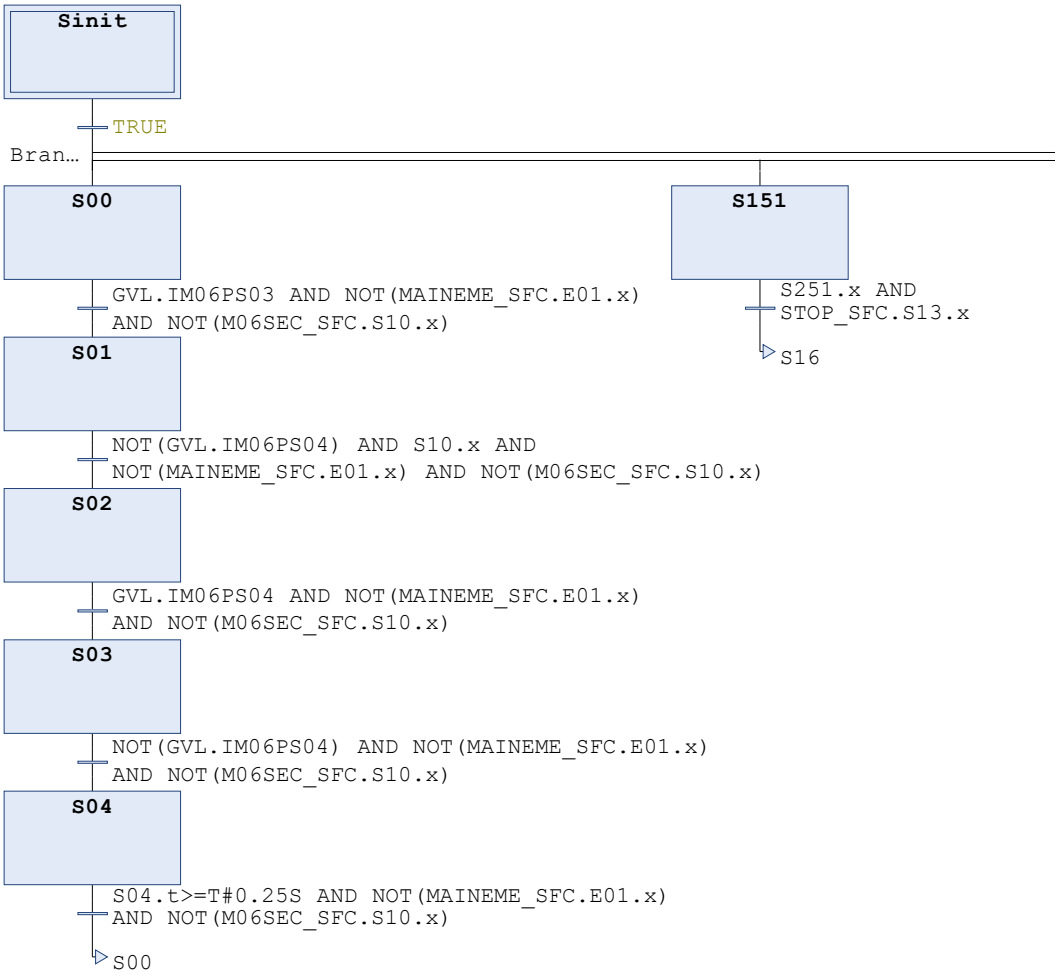
POU: M06SEC_LD

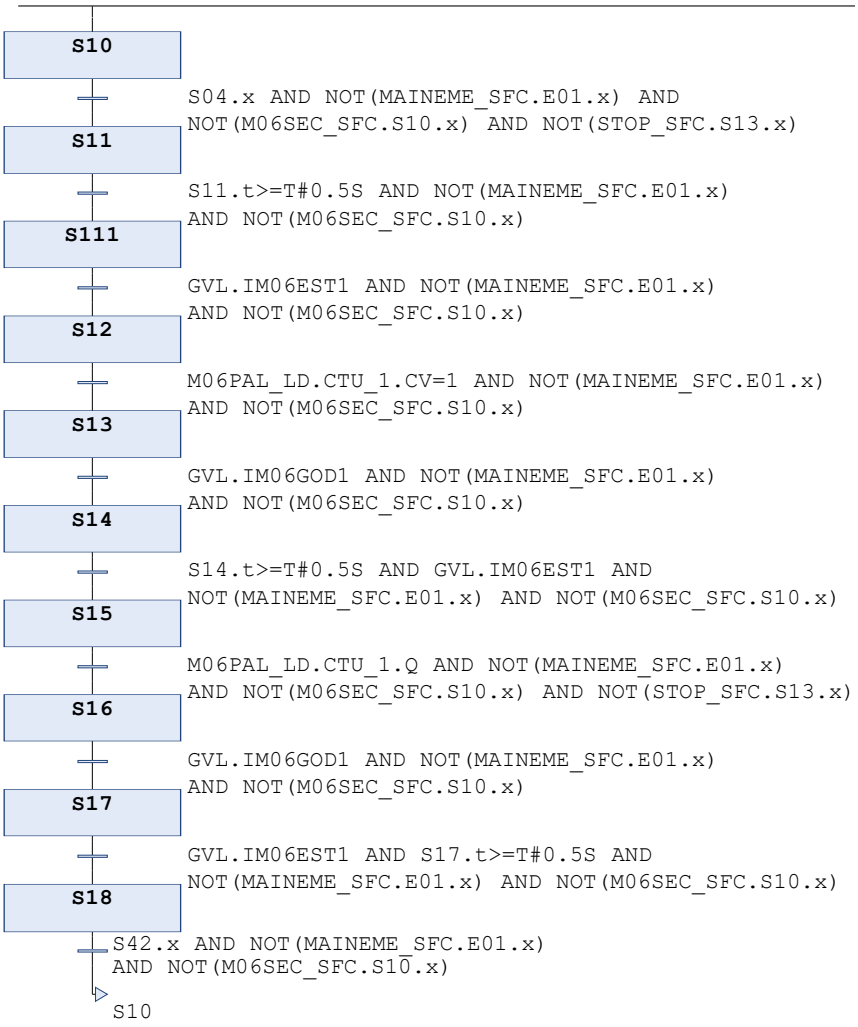
```
1   PROGRAM M06SEC_LD
2   VAR
3
4   END_VAR
5
6
```

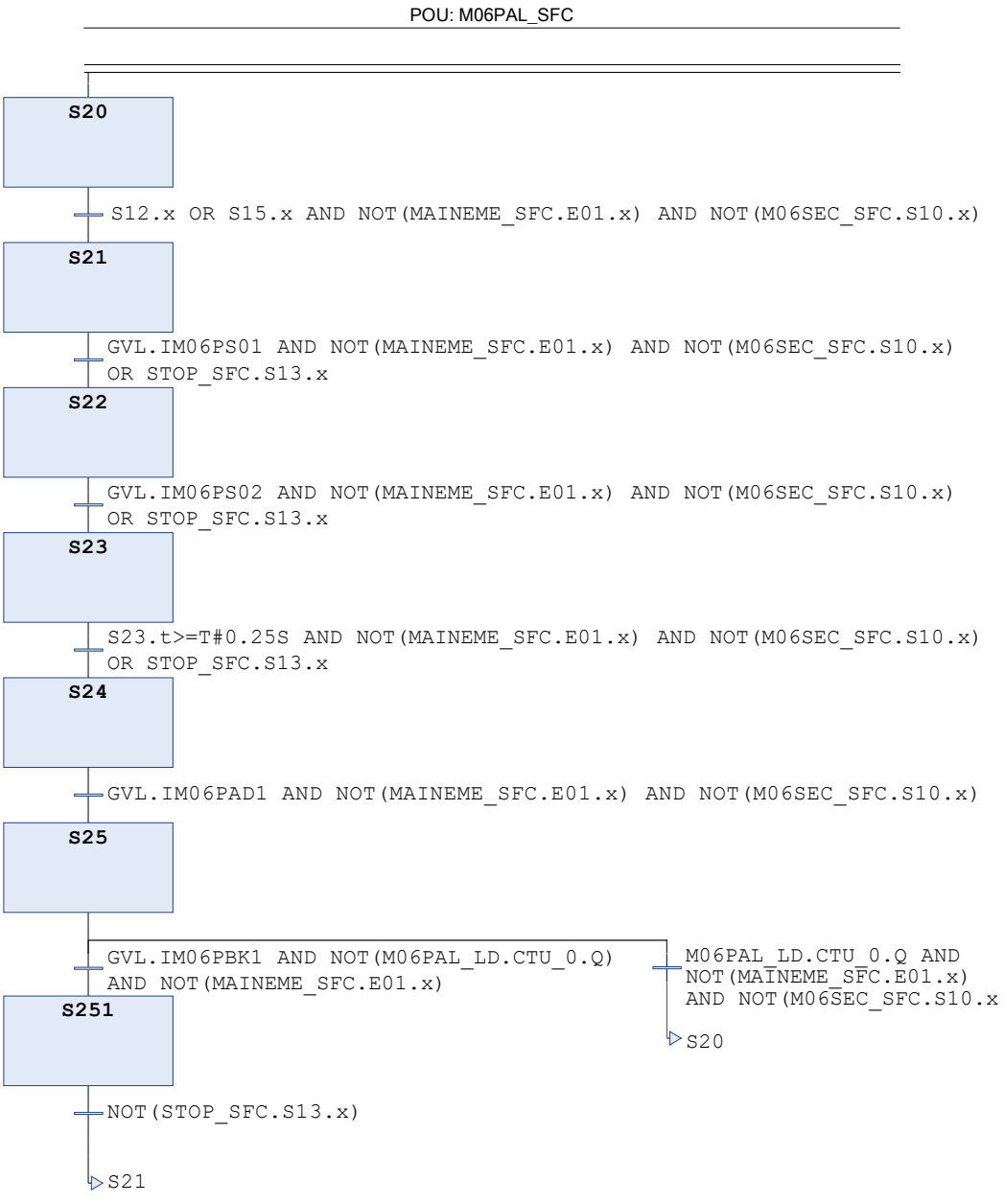


POU: M06PAL_SFC

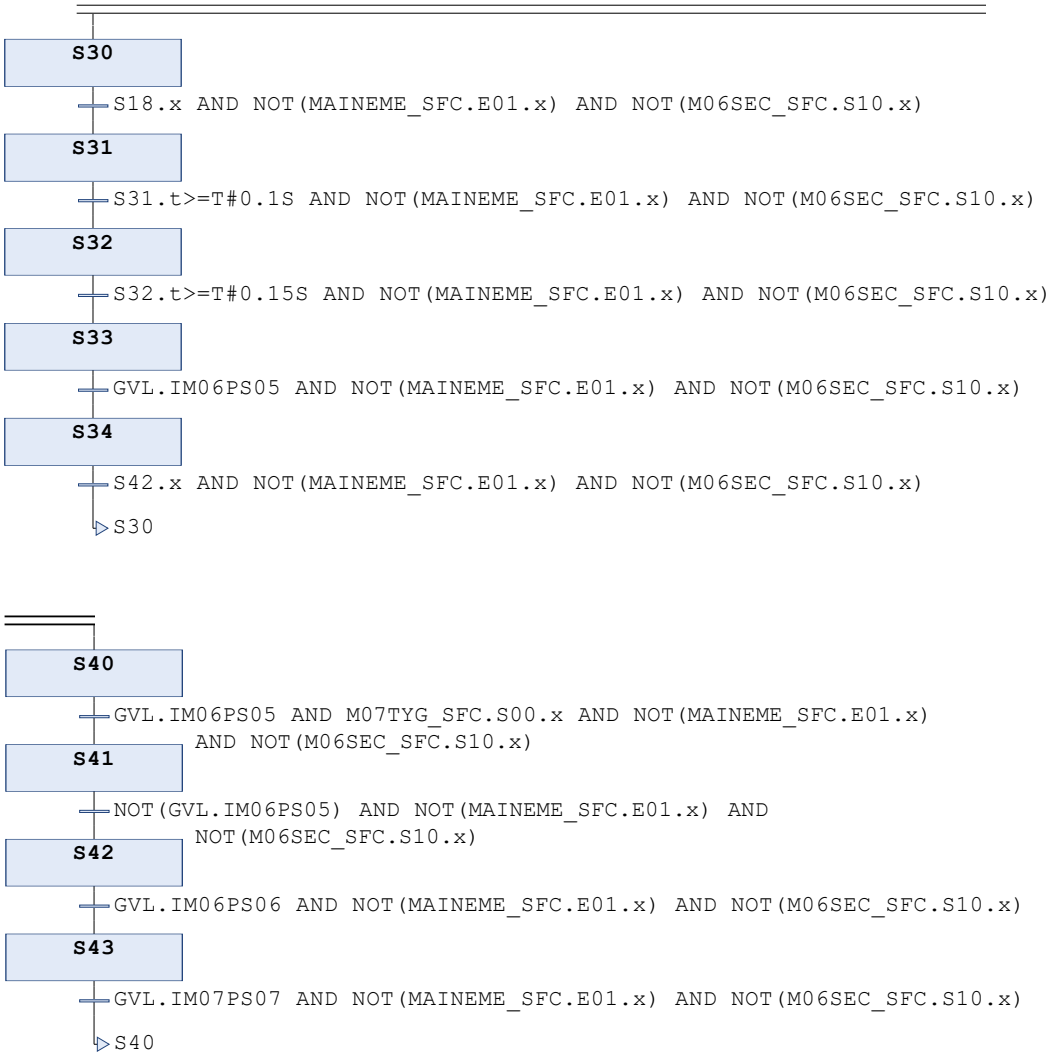
```
1       PROGRAM M06PAL_SFC
2       VAR
3       END_VAR
4       VAR_INPUT
5           SFCInit : BOOL ;
6       END_VAR
7
```




S10
S04.x AND NOT(MAINEME_SFC.E01.x) AND
NOT(M06SEC_SFC.S10.x) AND NOT(STOP_SFC.S13.x)
S11
S11.t>=T#0.5S AND NOT(MAINEME_SFC.E01.x)
AND NOT(M06SEC_SFC.S10.x)
S111
GVL.IM06EST1 AND NOT(MAINEME_SFC.E01.x)
AND NOT(M06SEC_SFC.S10.x)
S12
M06PAL_LD.CTU_1.CV=1 AND NOT(MAINEME_SFC.E01.x)
AND NOT(M06SEC_SFC.S10.x)
S13
GVL.IM06GOD1 AND NOT(MAINEME_SFC.E01.x)
AND NOT(M06SEC_SFC.S10.x)
S14
S14.t>=T#0.5S AND GVL.IM06EST1 AND
NOT(MAINEME_SFC.E01.x) AND NOT(M06SEC_SFC.S10.x)
S15
M06PAL_LD.CTU_1.Q AND NOT(MAINEME_SFC.E01.x)
AND NOT(M06SEC_SFC.S10.x) AND NOT(STOP_SFC.S13.x)
S16
GVL.IM06GOD1 AND NOT(MAINEME_SFC.E01.x)
AND NOT(M06SEC_SFC.S10.x)
S17
GVL.IM06EST1 AND S17.t>=T#0.5S AND
NOT(MAINEME_SFC.E01.x) AND NOT(M06SEC_SFC.S10.x)
S18
S42.x AND NOT(MAINEME_SFC.E01.x)
AND NOT(M06SEC_SFC.S10.x)
S10

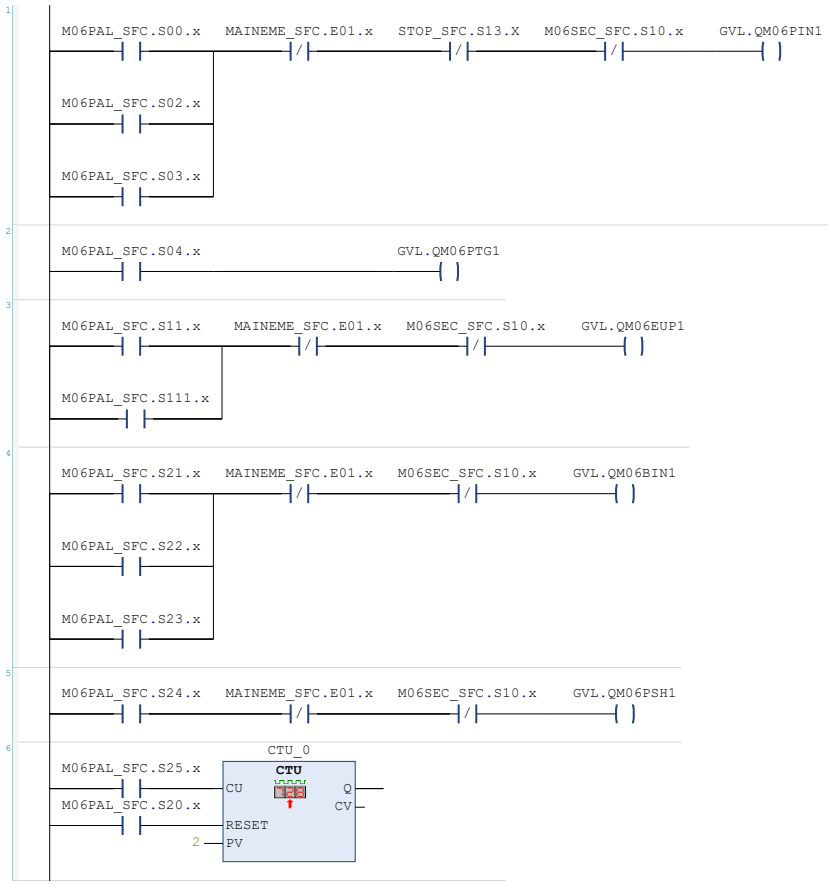

POU: M06PAL_SFC
S20
S12.x OR S15.x AND NOT(MAINEME_SFC.E01.x) AND NOT(M06SEC_SFC.S10.x)
S21
GVL.IM06PS01 AND NOT(MAINEME_SFC.E01.x) AND NOT(M06SEC_SFC.S10.x)
OR STOP_SFC.S13.x
S22
GVL.IM06PS02 AND NOT(MAINEME_SFC.E01.x) AND NOT(M06SEC_SFC.S10.x)
OR STOP_SFC.S13.x
S23
S23.t>=T#0.25S AND NOT(MAINEME_SFC.E01.x) AND NOT(M06SEC_SFC.S10.x)
OR STOP_SFC.S13.x
S24
GVL.IM06PAD1 AND NOT(MAINEME_SFC.E01.x) AND NOT(M06SEC_SFC.S10.x)
S25
GVL.IM06PBK1 AND NOT(M06PAL_LD.CTU_0.Q)
AND NOT(MAINEME_SFC.E01.x)
M06PAL_LD.CTU_0.Q AND
NOT(MAINEME_SFC.E01.x)
AND NOT(M06SEC_SFC.S10.x
S20
S251
NOT(STOP_SFC.S13.x)
S21
ApplePackingLine10x64.project
8/6/2025 10:44 AM
Page 3 of 4

POU: M06PAL_SFC

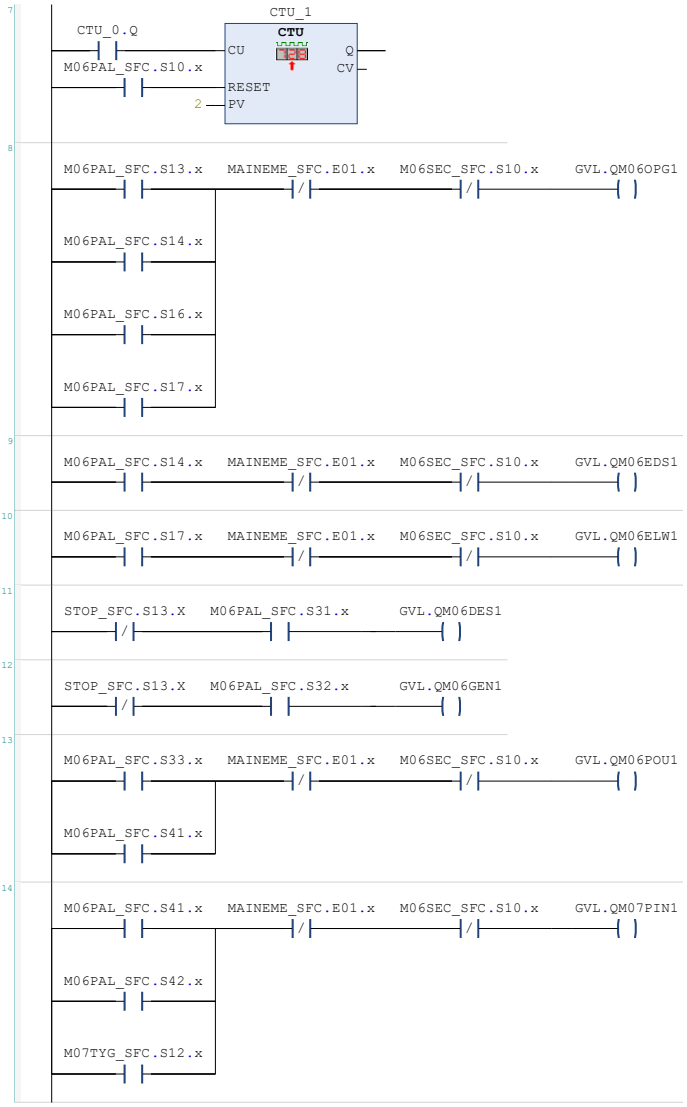


POU: M06PAL_LD

```
1   PROGRAM M06PAL_LD
2   VAR
3       CTU_0 : CTU ;
4       CTU_1 : CTU ;
5   END_VAR
6
7
```

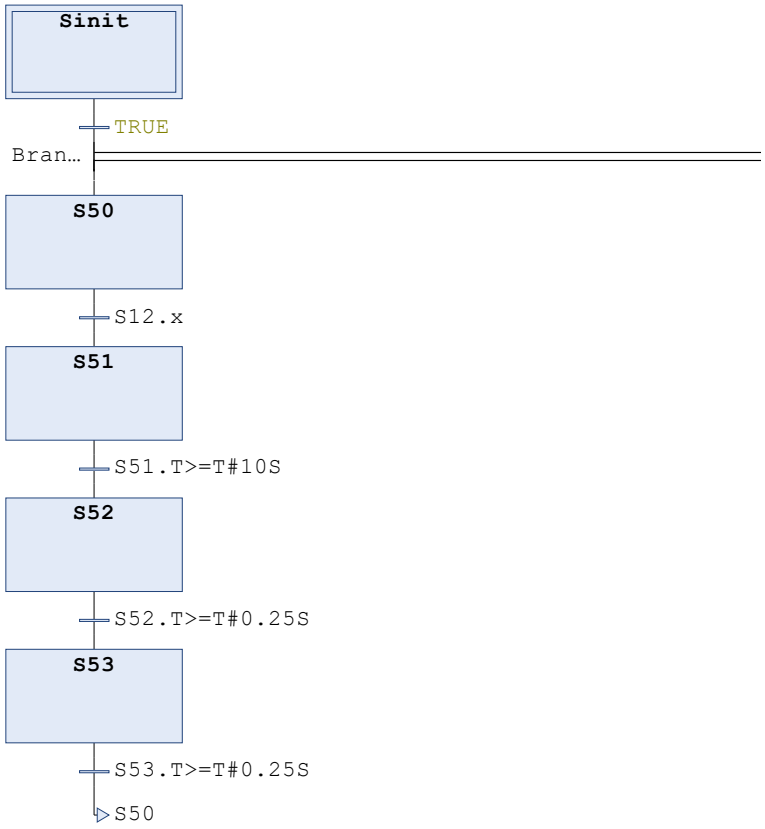


POU: M06PAL_LD

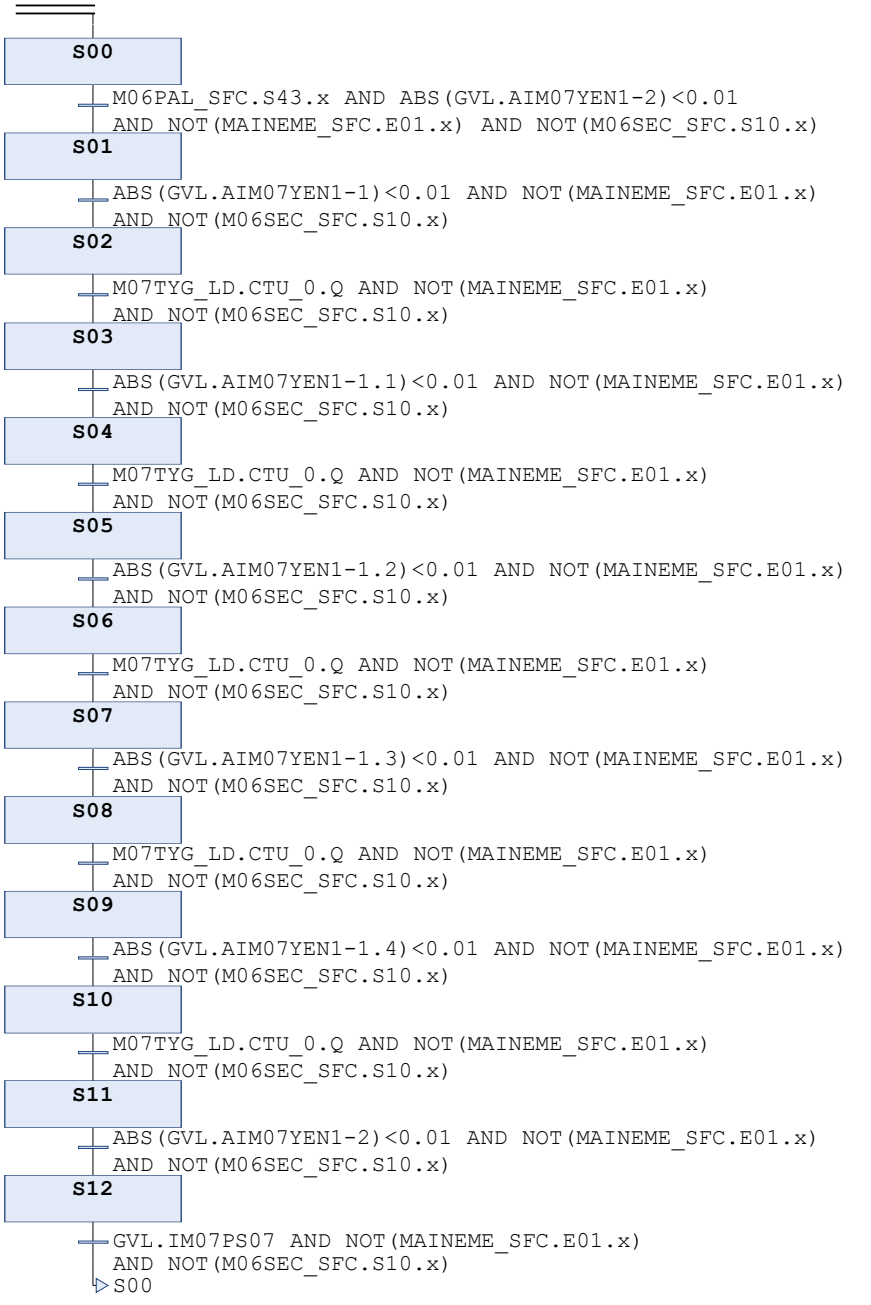


POU: M07TYG_SFC

```
1   PROGRAM M07TYG_SFC
2   VAR
3   END_VAR
4   VAR_INPUT
5       SFCInit : BOOL ;
6   END_VAR
7
```

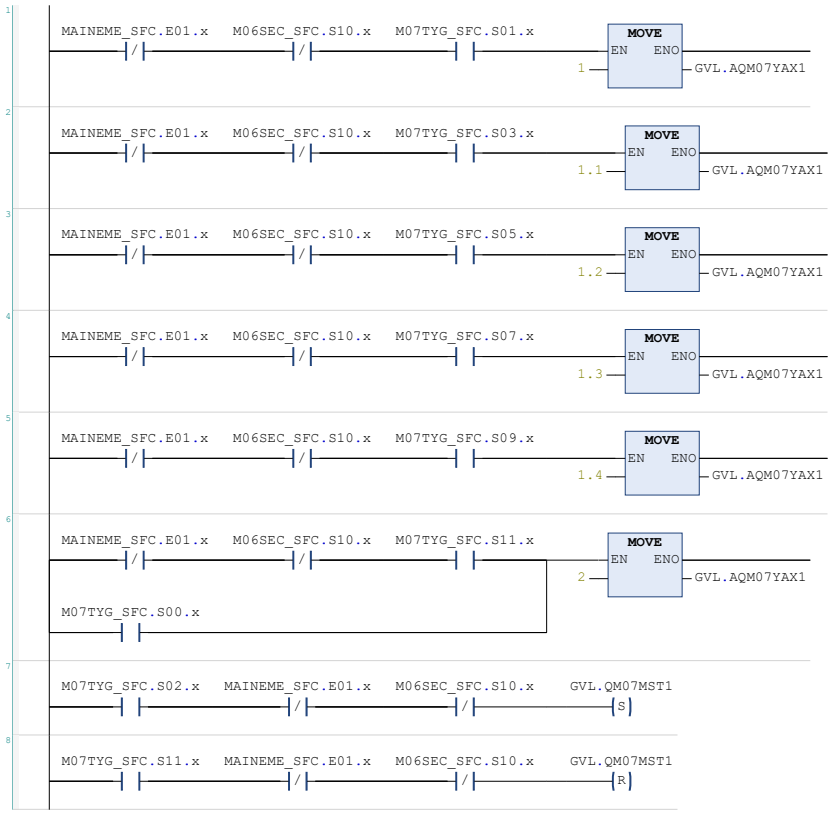


POU: M07TYG_SFC



POU: M07TYG_LD

```
1       PROGRAM  M07TYG_LD
2       VAR
3           CTU_0 :  CTU ;
4       END_VAR
5
6
```



POU: M07TYG_LD

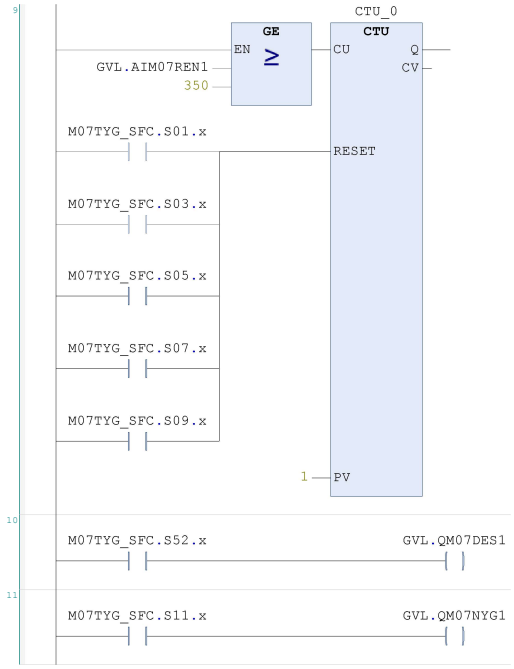


Figure 5.1: Video of the simulation. (Link: https://youtu.be/7TkJYdTqgng)