UNIVERSITÀ DEGLI STUDI DI PADOVA

DIPARTIMENTO DI INGEGNERIA DELL'INFORMAZIONE

INFORMATION ENGINEERING DEPARTMENT

Master Degree course in Control Systems Engineering

Master Degree Thesis
27/02/2023

An automatic system for bricks production

Supervisor
Prof. Giovanni Boschetti

Candidate
Francesca Olivieri

Academic Year 2022-2023

Abstract

The project consists of a system that has to carry wet clay bricks from one point of the factory to another one, arrange them in a certain pattern and then load the bricks on the floors of a shelf until it is full. At this point the shelf is pushed forward into the oven and another one arrives to be loaded.
The system is composed by transportation components as rollers and conveyor-belts, by a loading system and a cart-pusher. Their movements are controlled using a Siemens PLC, an HMI, two encoders, twenty inverters and many sensors. The devices are almost totally connected to a PROFINET network, just eleven inverters are not linked and directly controlled by the inverters.
All the PLC software and the HMI interface are designed and explained in detail.

Contents

Introduction

The aim of the project is to control a system for the brick production of a factory based in Lambayeque, Peru. Its mechanical design and construction of the machine was done by CMA s.r.l. [1], that has provided the mechanical schemes and components specifications. On the other side Innova s.r.l. [2] designed the electrical part and built the electrical cabinet.

The system is composed by several different elements and the challenge is to create an appropriate communication between each of them in such a way to have a perfect control of each action. The system is under the control of a Siemens PLC that works as its brain.
In addition to the mechanical part, there are different types of devices to control the system. They have to communicate with each other and so a network is built. There are twenty inverter, one for each drive and an ideal situation was having each of them connected to a PROFINET network, that allows a very fast data exchange and allows to have feedbacks about the status of the motor. Unfortunately this was not possible due to the fact that there is a global shortage of electronic chips that comprehends also the communication boards. This shortage is due to various causes, as the effects of Covid-19 lockdown, the growing request of electronic devices and the politics in general.
In fact, just nine of the twenty inverters are provided with this communication module and so connected to the network for this reason. The others do not provided feedbacks and are directly controlled by the inverter as will be explained.
Anyway the system can synchronize and control all of its elements.

An important part of the project is the HMI, upon which an application is created that is able to interface directly with the PLC and to modify parameters or to give commands. The application is created to be very intuitive, since it will be used by factory workers that do not know all the functioning of the system.

In the following each part of the realization of the project will be explained singularly. The first chapter described the mechanical components of the system, the devices that are used and how they communicate to each other.
The second chapter presents the software that is created using Tia Portal, since there is a Siemens PLC. The program is made in such a way that each component is controlled in a single block of the problem, in order to simplify the diagnostics in case of problems.
The third chapter presents the application that is installed on the HMI and allows to simply the interaction with the system. It is explained page by page.

Chapter 1

Project description

The project consists of a system that has to carry wet clay bricks from one point of the factory to another one, arrange them in a certain pattern depending on the type of the brick and then load the bricks on the floors of a shelf until it is full. At this point the shelf is pushed forward into the oven and another one arrives to be loaded.

1.1 Layout

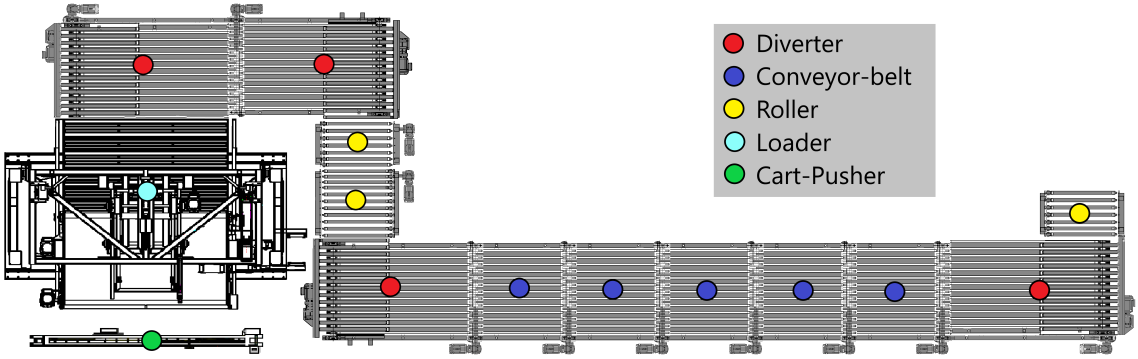


Figure 1.1: General layout of the machine.

To better understand how the machine works, the layout of the system is exploited. As can be seen in Figure 1.1, it is composed by four diverters, five conveyor-belts, three rollers, one elevator and load system and a cart-pusher. It must be considered that the image does not present the true aspect ratios, in fact for example the conveyor-belts are much longer in reality. Notice also that the first roller is not controlled in this project since it belongs to the previous industrial line of the factory, it is there just to show from where the bricks come in.
To control all the elements of the system many devices are used: two encoders, many sensors, twenty inverters, an operator panel and a PLC. In the following sections each component of the system will be exploited in details, for the moment look at how the

machine works in general.

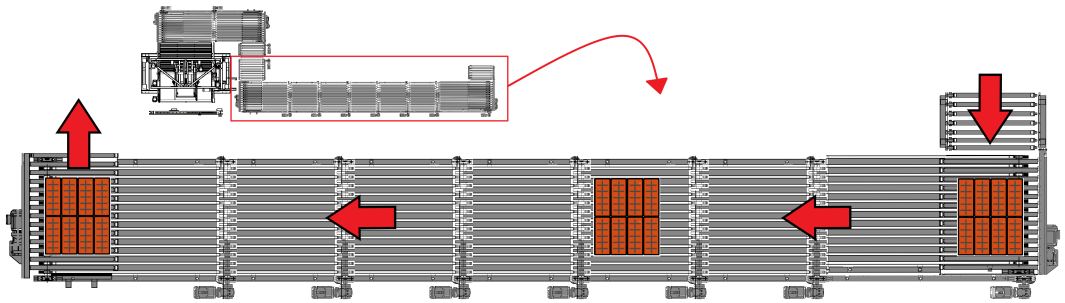


Figure 1.2: Path of a bricks block from the entering point to the second diverter.

The bricks enter the system in blocks of 2 rows, 4 columns and 2 levels each, so 16 bricks at a time. As shown in Figure 1.2, a block arrives through an external roller on the mechanical roller of the first diverter, then it is moved linearly through its belt and through a series of five conveyor-belts. At this point the bricks arrive to the second diverter that push them on another series of two rollers, see Figure 1.3.
They can carry the bricks on the third diverter in two ways: on it can go 4 or 5 rows of bricks, so two groups or two and half are moved on the diverter.

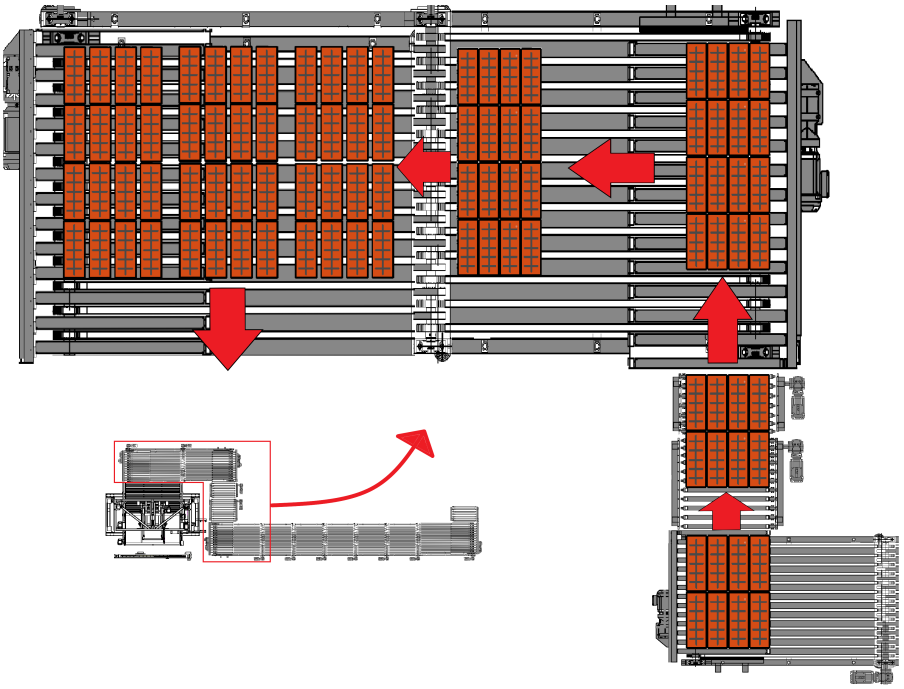


Figure 1.3: Path of the bricks from the second diverter to the loader when the option for 4 rows is selected.

Then the bricks enter a couple of diverters, which have the purpose to create the desired bricks pattern. At the end of the last diverter the bricks are loaded onto the belt of the loader and then stored in a free floor of the shelf that is positioned in front of it. Once that a shelf is full a new one arrives thanks to the action of the cart-pusher.

1.1.1 Stations

Considering that the machine is composed by a lot of different components, to understand its behaviour and to command it in a simpler way, as will be clearer later, the system is divided in four areas: the transportation, the grouper, the loader and elevation system and the cart-pusher. The areas are shown in Figure 1.4 and they are now explained in details.

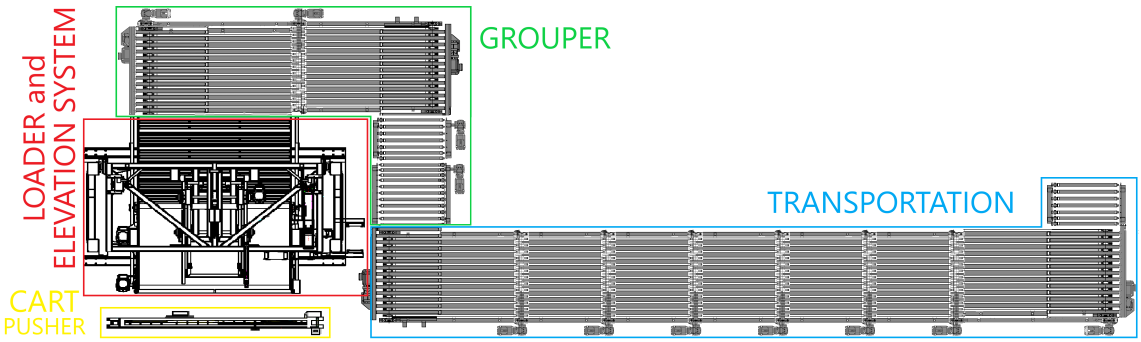


Figure 1.4: Division of the system in four smaller areas.

Transportation

The transportation part consists of all the conveyor-belts, diverters and rollers that carry the bricks from the entering point to the grouper. Look at the blue block of Figure 1.4, in detail is composed by:

- the first diverter;
- a series of five conveyor-belts;
- the second diverter.

Its aim is just to carry the bricks from the entering point of the system to the grouper. The reported pictures are schematic, actually the conveyor-belts are longer than showed, as already said.

Grouper

The grouper is the part that has to composed the bricks pattern and looking at the green block of Figure 1.4 it consists in:

- a series of two rollers;
- a series of two diverters.

Once that the bricks have reached the end of the transportation part, they arrive on the series of two rollers, which have to put on the following diverter the desired number of bricks rows, that can be 4 or 5 and it is selected by the operator panel. When the correct number of brick rows is positioned on the diverter, the creation of the pattern can start.
It is important to notice that the bricks have to fill as much as possible the roller of

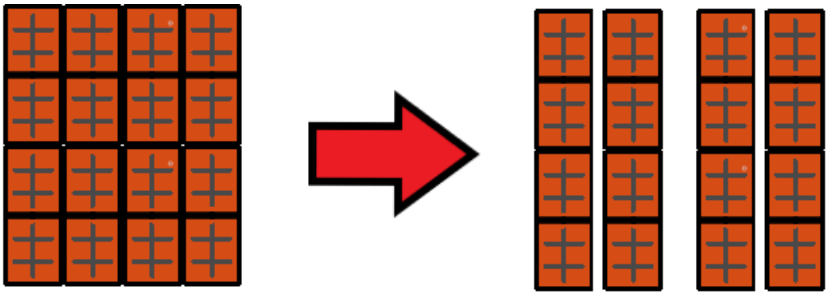


Figure 1.5: Example of bricks pattern creation. In this case there are two groups of two columns each.

the last diverter, in order to optimize the production, so the pattern has the following structure: there are some groups of bricks, that can vary from two to five depending on the type of the bricks; each group is spaced from another one by a distance set in the operator panel. The parameters of each group are defined in the same way, in fact the number of columns per group and the space between each column are set by the panel. These values are parameters and so can be changed in every moment, obviously they have to be standard in such a way to load the shelf in an homogeneous way. In fact a "recipe" - a set of parameter that are stored - is defined for each type of brick and can be selected from the operator panel.
The space between the groups and between each column of the group is created using two different speeds of the conveyor-belts of the two diverters. In fact, if the second one is faster than the previous conveyor, a space between two objects over them is created.
At the end of the roller of the second diverter the bricks are passed to the loader and elevation system.

Loader and Elevation system

As shown in Figure 1.6, the loader and elevation system is composed by two main parts:

- a cart which is equipped with a conveyor-belt and a shovel. Its aim is to load one floor of the shelf at a time;
- the elevator, that allows to change the height of the cart and so loading the different floors of the shelf.

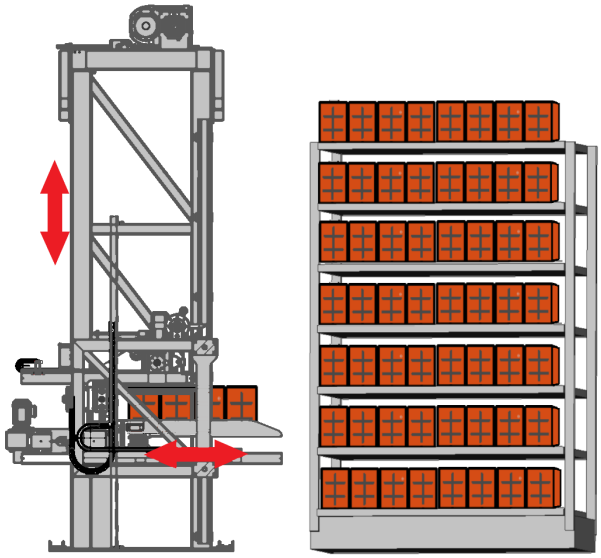


Figure 1.6: Side view of the Loader and elevation system with a full shelf next to it.

When the bricks are arranged in the specific pattern by the grouper, they passed on the conveyor-belt of the loading system. The elevator reaches the free floor of the shelf above the previously loaded one and starts loading it in with a particular procedure since the bricks are made of wet clay. First of all the conveyor-belt starts running forward, moving the bricks toward the shelf; then the shovel goes down and forward, next to the last brick, at this point the conveyor-belt starts running backward while the shovel remains forward, keeping the bricks on the shelf. Using this procedure the bricks do not get stuck to the tape.
Each shelf is composed by seven floors and is stored floor by floor starting from the lowest one. Note that the bricks pattern is created in such a way that it fit perfectly the floors of the shelves.

Cart-pusher

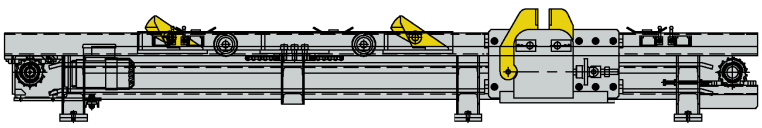


Figure 1.7: Cart-pusher

The cart-pusher is basically composed by a pusher and a locking system. If a free shelf is ready to be loaded, it is positioned in front of the elevator and it is locked in that position. When the loading procedure is finished and so the shelf is full, the system unlocks it and pushes it forward; simultaneously another shelf, that is free, is positioned in front of the elevator to start again the loading procedure.
At the end of the loading each shelf is moved toward an oven in order to be cooked.

1.2 Mechanics

The system works with a general power of 440V AC and auxiliary voltage of 24 DC.
In the following paragraphs all the mechanical components will be explained singularly.

1.2.1 Conveyor-belts

A conveyor-belt is an industrial components that allows to move objects linearly. It has a tape or a number of tapes that are moved parallel to the length of the conveyor-belt.
In this project the movement of its engine is managed by an inverter, that commands the sense of translation and the velocity of the conveyor.
There are five different conveyor-belts in the system, all lined up between the first and the second diverter.

Figure 1.8: An example of a conveyor belt

1.2.2 Rollers

A roller is an industrial component similar to the conveyor-belt. The difference is in the mechanical movement: there are many rollers positioned perpendicular to the sense of movement, their rotation allows to move forward what is positioned over them.
They are controlled exactly in the same way of the conveyor-belts and there are two of them in the system, lined up between the second and the third diverter.

1.2.3 Orthogonal diverters

Figure 1.9: An example of an orthogonal diverter

An orthogonal diverter is an object that allows to make a 90 degrees turn on a conveyor line. It is basically composed by a conveyor-belt and a roller, which can be at the same level of the belt or higher. To have a proper control of the movements, these two elements can not move simultaneously but there are two modalities:

- the roller does not run and it is in the low position, while the belt moves;
- the roller is in the high position and moves, while the belt does not run.

The movements of the roller and of the belt are managed by two different inverters, while the elevation of the roller is commanded by a solenoid valve.

In this system there are four different diverters and each of them is equipped with three sensors: the first checks if the roller is in the high or low position; the second checks if there are bricks over it; the last one checks if the bricks have reached the end of the roller or of the belt of the diverter.

1.2.4 Drives

Each drive of the system is an AC motor by SEW-EURODRIVE, they are twenty in total with different parameters in general, each of them is managed through an inverter. The drives moves:

1. the roller of the first diverter;
2. the belts of the first diverter;
3. the belts number 1;
4. the belts number 2;
5. the belts number 3;
6. the belts number 4;
7. the belts number 5;
8. the roller of the second diverter;
9. the belts of the second diverter;
10. the first roller of the series;
11. the second roller of the series;
12. the roller of the third diverter;
13. the belts of the third diverter;
14. the belts of the fourth diverter;
15. the roller of the fourth diverter;
16. the elevator of the loader;
17. the shovel of the loader;
18. the cart translation of the loader;
19. the conveyor-belt of the loader;
20. the cart-pusher.

Figure 1.10: An example of AC drive by SEW-EURODRIVE

1.3 Devices

In the following paragraphs the various devices will be explained singularly.

1.3.1 PLC

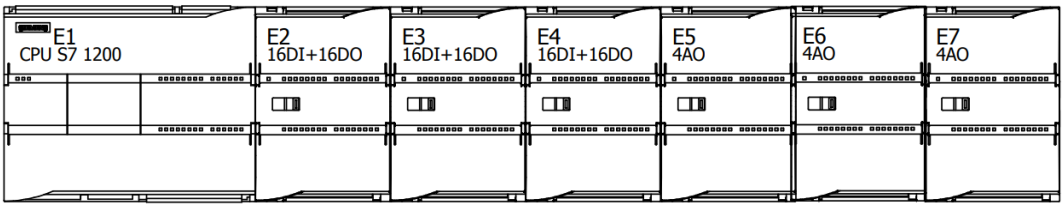


Figure 1.11: Siemens S7 - 1200 PLC with the additional modules

The PLC is the most important device, it is the brain of the system. The PLC aims to control and command all the elements of this section.
The PLC used in this project is a Siemens S7 - 1200, itself has 16 digital inputs, 16 digital outputs, 2 analog inputs and 2 analog outputs. Then it is equipped with three modules with sixteen digital inputs and sixteen digital outputs each and other three modules with four analog outputs each.

1.3.2 HMI

Figure 1.12: eSMART10 operator panel by Exor

The HMI is an eSMART10 by Exor. It has an application installed on it that allows to control the system and its parameters. The application has been made with the JMobile software, creating page by page the program using different elements that can be buttons, images or widgets. It will be explained in details in Chapter 3.
The panel is important since it presents a simple interface for the operator and allows to command and to modify the parameters of the system in real time.

1.3.3 Sensors

There are fifty-one sensors of various type arranged all over the system.

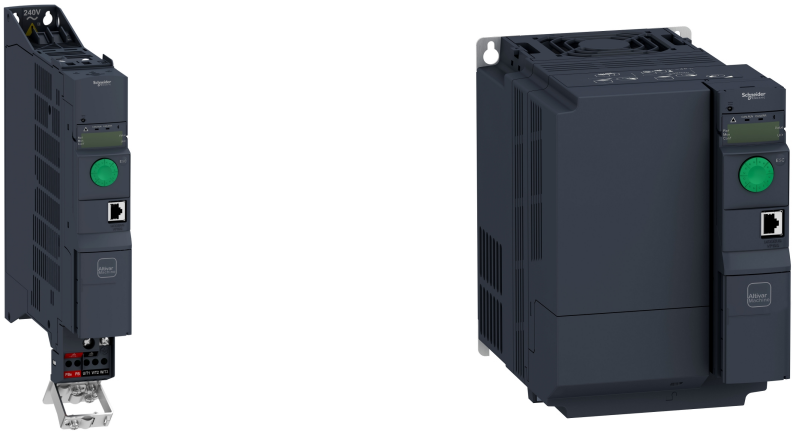
On the conveyor-belts and on the diverters there are sixteen photocell sensors, to check where the bricks are and if they are present. There are also two more photocell sensors on the loader and elevation system.

There are then ten magnetic field sensors: two for each diverter - so eight in total - to check if the their rollers are all the way up or all the way down; the lasts two are on the cart-pusher, to see if its locking system is totally locked or totally open.

Another type of sensor that is used is the proximity sensor. There are eight of them on the loader and elevation system: three of them check the longitudinal-position of the shovel, while other two to check its height; two proximity sensors are used to check the position of the cart; the last one marks the home position of the elevator. There then are another two on the cart-pusher, to check the presence of the shelves.

There are eight limit switches on the loader and elevation system, with other four on the cart-pusher.

1.3.4 Inverters



(a) 0.75kW or 1.5kW (b) 7.5kW

Figure 1.13: Schenider inverter Altivar 320

An inverter is a power electronic device that changes direct current to alternating current. Twenty Schenider Altivar 320 speed variator are used in the system but of three different types. They differ from each other only by the value of power. Among all the inverters:

- there are 7 that manage drives with 0.75kW. These inverters manage the drives that control: the roller and the belts of the first and second diverter, the two rollers

of the series and the roller of the third diverter;

- other 12 manage drive with 1.5kW. These inverters manage the drives that control: the five conveyor-belts of the series, the roller and the belts of the fourth diverter, the belts of the third diverter, the cart-pusher and the shovel, the cart and the belt of the loader;

- the last one manages a drive with 7.5kW. This inverter manage the drive that moves the elevator.

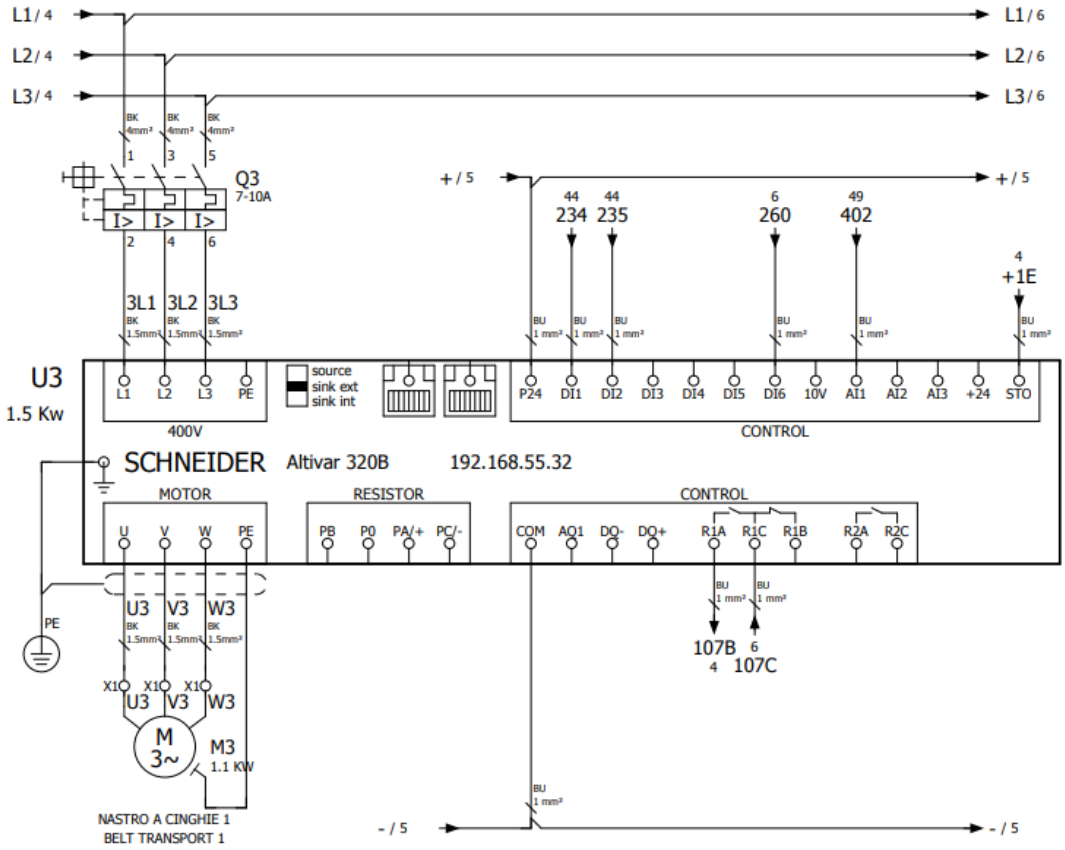


Figure 1.14: Electrical scheme of the inverter U3, the one that controls the first conveyor-belts of the series

The inverters are interfaced with the related drive just with the three-phase power supply, as shown in the bottom left part of Figure 1.14: U, V and W are directly connected to the three phases of the drive.

1.3.5 Encoders

An encoder is an electro-mechanical device that converts the angular position or motion of a shaft or axle to analog or digital output signals.
The ones that are used in this project are absolute rotary encoders by TR Electronic. There are two absolute encoders in the system: one is on the drive that commands the conveyor-belt of the fourth diverter, the one that has to create the bricks pattern; the other one is on the elevator of the loader, in order to have a precise measure of the position of the cart of the loading system when reaching a floor of a shelf.

Figure 1.15: Example of an absolute rotary encoder

1.4 Communication

As said in the introduction there were not enough communications boards for all the devices, so just nine inverters are connected via Ethernet under the Profinet protocol. The others are directly controlled by the PLC.

1.4.1 Profinet Network

PROFINET (Process Field Network) is an Industrial Ethernet solution. It is a communication protocol to exchange data between controllers and devices. PROFINET sits on Layer 7 of the ISO/OSI model since it is an application. It defines cyclic and acyclic communication between components, including diagnostics, functional safety, alarms, and other related information. Also, PROFINET is based on standard Ethernet for its communication medium. High bandwidth, large message size, and versatility are just some of the benefits of having Ethernet on the factory floor. [3]

Figure 1.16: Layout of the PROFINET network

The topology of the Ethernet network, on which the PROFINET one is based, is a star, where a switch is the center; all the devices are connected to it. To have a correct communication it is also important to know the type of data that each device needs as input and that sends as output .They are:

- the PLC with IP address of the PLC is 192.168.55.10;
- the ring network of the inverters U12 - U20. Each of them communicates with 10 words of inputs and 10 words of output, whose meaning is specified in their manual. To unpack the data an ad hoc structure called "UDT_ATV_320" is built following the specifications, as will be explained in the next chapter.
  The communication board that allows the connection of the inverter is the additional module W3A3627, the ones that have it are the inverters that control:

  U12 - the roller of the third diverter with IP address 192.169.55.61;

  U13 - the belts of the third diverter with IP address 192.169.55.62;

  U14 - the belts of the fourth diverter with IP address 192.169.55.63;

  U15 - the roller of the fourth diverter with IP address 192.169.55.64;

  U16 - the elevator of the loading system with IP address 192.169.55.65;

  U17 - the shovel of the loading system with IP address 192.169.55.66;

  U18 - the cart translation of the loading system with IP address 192.169.55.67;

  U19 - the conveyor-belt of the loading system with IP address 192.169.55.68;

  U20 - the cart-pusher with IP address 192.169.55.69.

- the encoders 1 and 2 with IP addresses respectively equal to 192.169.55.40 and 192.169.55.41. Each encoder exchanges:

  Input - one DINT variable for the preset value and 2 bitwise addressed bytes for the preset and control;

  Output - 2 DINT variables for the feedbacks on position and velocity and 2 bitwise addressed bytes for the preset and the status.

- the operator panel with IP address 192.168.55.20. The JMobile application installed on it contains the addresses of some variables of the PLC: when the software is finished, its variables are exported and loaded in the application to be linked to the panel buttons and parameters. In this way the control of the system is in part controlled by the operator panel, as it can modify the variables value.

The PROFINET network allows to exchange data efficiently between the devices and having information about their status. It is built through the Tia Portal software as shown in Figure 1.16, that allows to set all the parameters of the communications.

1.4.2 Directly Controlled Inverters

The first eleven inverters U1-U11 are not equipped with a communication board, so they are not connected to the PROFINET network but they are directly controlled by the PLC. These inverters are the ones that control:

U1 - the roller of the first diverter;

U2 - the belts of the first diverter;

U3 - the belts number 1;

U4 - the belts number 2;

U5 - the belts number 3;

U6 - the belts number 4;

U7 - the belts number 5;

U8 - the roller of the second diverter;

U9 - the belts of the second diverter;

U10 - the first roller of the series;

U11 - the second roller of the series.

Their electrical scheme is modified to use the physical I/O of the inverters to control them. It is important to note that in this way there are no feedbacks from the inverter and so there are no available information about the drive status. Looking at Figure 1.14:

- the pin DI6 resets of the drive;
- the pin DI1 commands the forward running;
- the pin DI2 commands the backward running;
- the analog input AI1 commands the velocity of the drive.

Chapter 2

Software PLC

The program used to create the PLC software is TIA Portal v16, one of the most popular program for control Siemens PLCs.
The software approach that has been used can sometimes be redundant, but allows to manage each component singularly and to re-used the code of a single block in other projects or part of the same project if needed because a block comprehends all the necessary signals and parameters for its aim. The software is so structured in a particular way: the idea is to iteratively divide the control of the machine in smaller control actions. In practice, the software is made by little simple blocks, which command each component singularly, then they are used in bigger blocks that command the stations explained in the precedent section. At the end all the stations are controlled in the main program. The stations created in the software are three: Transportation, that includes the grouper and the transportation explained in the precedent chapter, Loader and Cart-Pusher.
Then there are other smaller blocks for the management of:

- the positioners, that are two. One in the grouper to create the bricks pattern and one in the loader and elevation system to control the height of the elevator;
- the encoders, related to each positioner;
- the inverters, that are twenty in total.

2.1 Variables

First of all look at the variables that will be used in the program. They are divided in groups in order that they can be found in a simpler way: there are the inputs and outputs of the PLC, the general variables, the inputs and outputs of the encoders and the inputs and outputs of the inverters.

2.1.1 PLC Variables

The PLC itself has 16 digital inputs, 16 digital outputs, 2 analog inputs and 2 analog outputs. Then it is equipped with three modules, with 16 inputs and 16 outputs each, and other three modules with 4 analog outputs each, as already said. In total the PLC has 64 digital inputs, 64 digital outputs, 2 analog inputs and 14 analog outputs.

In this paragraph the most important variables of the PLC are listed, divided in groups, with a little description of the related function.

Digital Inputs

The digital inputs are signals that the PLC receives from a physical element, that can be a sensor, a button or something else. The ones used by the PLC are:

- 51 digital inputs for the sensors, called as *I_SensorName*;
- *I_Cancelli_OK*, it indicates that the protections of the machine are all in the right position;
- *I_Emg_OK*, it indicates that the system is not in the emergency state;
- *I_Inverter_OK*, it indicates that the PLC can communicate with the inverters;
- *I_Reset*, it is related to the RESET button on the electrical cabinet;
- *I_Sel_Auto_Man*, it is related to the automatic/manual selector on the electrical cabinet;
- *I_Start*, it is related to the START button on the electrical cabinet;
- *I_Start_Man*, it is related to the START button when in manual mode;
- *I_Stop*, it is related to the STOP button on the electrical cabinet;
- five are not used.

Digital Outputs

The digital outputs are signals that the PLC writes on a physical connected element, that can be an inverter, a solenoid valve, a light or somethig else. The ones used by the PLC are:

- *O_Ux_AV* and *O_Ux_IND*, that are used to command the running to the direct controlled inverters, respectively in forward and backward direction. Those inverters are 11, so there are 22 digital outputs for them;
- *O_YV10A_BloccCarrello_Aperto* and O_YV10C_BloccCarrello_Chiuso, that are used to block and unlock the locking system of the cart-pusher;
- *O_YVxA_Deviatore_x_Alto* and *O_YVxB_Deviatore_x_Basso*, that are used to move respectively up and down the elevator of the diverter x. There are 4 diverter so in total there are 8 digital outputs for them;
- O_Richiesta_Prodotto, that is an output used to make a product request when there are no bricks entering the system;
- *O_Spia_Reset_Emg*, it controls the light that marks how to reset the emergency status. It makes blink the reset button. It is a steady light when there are no alarms;
- *O_Spia_Reset_Cancelli*, it controls the light for the status of the protections. It blinks when they are not in the security position, otherwise it is a steady light;
- *O_Spia_Start*, it controls the light of the start button on the electrical cabinet. It is on when the selector is in automatic mode and there button stop has not be pushed;
- *O_Spia_CmdMan*, it controls a light that blinks when the manual mode is selected, otherwise, if the system is running in automatic mode, it is a steady light;
- *O_Torretta_Verde*, it controls the green light and it is on when *O_Spia_Start* is;
- *O_Torretta_Giallo*, it controls the yellow light and it blinks when there is any active alarm;
- *O_Torretta_Rosso*, it controls the red light and it is on when the emergency occurs;
- *O_Torretta_Sirena*, it controls the alarm siren and it sounds when there is any active alarm.

Analog Outputs

The analog outputs are some values that the PLC passes to another device. In this case they are used to command the velocity of each of the directly commanded inverters, they are called as *AO_Vel_InverterName* and they are 11 in total.

General Variables

The general variables are the ones internally created by the PLC and used in the program. In details, they are:

- *EMG_OK*, that is directly related to the input *I_Emg_OK* with a time filter;
- *BARR_OK*, that is directly related to the input *I_Cancelli_OK* with a time filter;
- *SIC_OK*, that marks when the security conditions are verified. It is *true* if also *EMG_OK* and *BARR_OK* are *true*;
- *STOP_OK*, that causes the stop of the automatic cycle. It is directly related to the input *I_Stop* or it is active when any alarm is active;
- *SEL_AUTO* and *SEL_MAN*, they are referred to the signal of the selector on the electrical cabinet *I_Sel_Auto_Man*. The manual is the *true* and the automatic is the *false*;
- *AUTO_ON*, it is a signal that marks when the system is running in automatic mode and the stop signal *STOP_OK* is not active;
- *MAN_ON*, it is a signal that marks when the manual mode it is activated;
- *ALLARME*, it turns true when any alarm is active;
- *MACCHINA_PRONTA*, it marks when the system is all ready to work. It is active if there are no alarms and if each station presents the bit *Stazione_Pronta* equal to *true*;
- *RESET*, that is the general reset and it is directly related to the button *I_Reset*;
- *CINGHIE_AV* and *CINGHIE_IND*, they are the commands for the movements of the belts of the fourth diverter;
- *SOLLEV_AV* and *SOLLEV_IND*, they are the commands for the movements of the elevator of the loading system;
- *ENC_CINGHIE_Zero* and *ENC_SOLLEV_Zero*, they are the commands to reset the encoders;
- *CINGHIE_Velocita* and *SOLLEV_Velocita*, they are the commanded velocities of the inverters involved in the positioners;

2.1.2 Inverter Variables

Each inverter that is connected with the PROFINET protocol is provided by 10 integer variables for the inputs and 10 for the outputs. As already said for the encoder variables, what is an output for the inverter is an input for the PLC and viceversa.
The input and output variables are respectively called Ux_IW_y and Ux_OW_y, where

x is the number of the inverter and y goes from 1 to 10 indicating the number of the variables. They are 90 and 90 in total, since there are 9 inverter under the PROFINET protocol.

2.1.3 Encoder Variables

Each encoder is provided by 16 digital outputs, 16 digital inputs, 2 DINT outputs and a DINT input. Looking by the PLC side of view, the inputs become outputs and viceversa. Among the inputs, there are:

- 8 bits for the preset, that are called *ENC_x_IN_Preset_y* where x indicates the encoder and y the number of the bit that goes from 0 to 7;
- 8 bits for the status of the encoder *ENC_x_IN_Status_y*;
- a couple of DINT variables used for the position and the velocity read from the encoder, respectively *ENC_x_IN_Pos* and *ENC_x_IN_Vel*.

Among the outputs there are:

- 8 bits for the preset. The first is called ENC_x_OUT_Preset_Mode and the other seven *ENC_x_OUT_Preset_y*;
- 8 bits for the control, that are called *ENC_x_OUT_Control_y*;
- a DINT variable *ENC_x_OUT_Preset* for the preset value.

2.2 Inverter management

As previously explained, due to the fact that there were no enough communication boards, nine inverters are connected to the PROFINET network while the others eleven are directly managed by the PLC. In both cases an FC and a DB is created for each inverter and they are perfectly equal to each other. Besides these, there are two general blocks: an FC called *FC700_General_Inverter* which just calls all the singular FC of each inverter and a DB called *DB700_Params_Inverter* that stores all the parameters of each inverter which are completely filled through the operator panel. The parameters are:

- *Velocity_FW*, *Velocity_BW* and *Velocity_MAN*, the values of the velocity when respectively the motor is running forward, backward and when the manual mode is selected;
- *Velocity_MAX* and *Velocity_MIN*, the values of the maximum and minimum velocity that the motor can perform;
- *Acceleration* and *Deceleration*, the values of the acceleration and deceleration of the motor;

2.2.1 Directly controlled inverters

The inverters directly managed by the PLC are eleven and they are the ones that go from U1 to U11. An ad hoc couple of DB and FC is created for each of these inverters. All the FCs of the directly controlled inverters are equal to each other, obviously using the variables relating to the specific inverter.

Directly controlled inverter Data Structure - DB

All the variables related to the inverter are stored in this Data Structure, that is composed by three different structures:

- *CMD*, that comprehends all the commands of the inverter. They are:
  - *Enable*, the command that enables the movement of the motor;
  - *Run_AV* and *Run_IND*, to forward and backward running of the motor;
- *FBK*, that comprehends the feedbacks that the inverter returns. They are just two boolean: *Ready*, that says if the motor is ready to work; *Running*, that says that the motor is running.
- *PARAMETRI*, that comprehends the parameters of the inverter. That are:
  - *Velocita_MIN* and *Velocita_MAX*, that are respectively the minimum and maximum velocity of the motor. They are copied from *Velocity_MIN* and *Velocity_MAX*;
  - *Velocita_Avanti* and *Velocita_Indietro*, that are respectively the forward and backward velocity of the motor. They are limited in the interval [*Velocita_MIN*, *Velocita_MAX*];

Directly controlled inverter Logic - FC

First of all the structure *FBK* is updated. The inverter is *Ready* as long as the inverter is *enabled*; the inverter is *Running* as long as the inverter is ready and if it is commanded to run forward *Run_AV* or backward *Run_AV*.

Then the commands of the inverter are elaborated. The command *Enable* is *true* as long as there are no emergencies in the system.The command *Run_AV* can be activated both in manual and automatic mode through the specific command of that inverter, which is managed in one of the stations. For example the movement of the roller of the first diverter is controlled by ***AUTO_CMD***.*Rull_Dev_1_AV* in the *Transportation* station when the system is in automatic mode or with *MAN_CMD.Rull_ Dev_1_AV* when the system is in manual mode. The parameter *Velocita_Avanti* is set equal to *Velocity_FW* in automatic mode, otherwise it is set equal to *Velocity_MAN*.
The management of *Run_IND* is done exactly in the same way.
At this point the FC writes the outputs to be communicated to the inverter. The physical output that commands the forward running of the inverter is the boolean *O_Ux_AV* and it is active as long as the following series of conditions is *true*: the inverter is ready and enabled, the command *Run_AV* is active while the command *Run_IND* is inactive.
The output that commands the backward running of the inverter *O_Ux_IND* is managed in an analogous way, with the commands *Run_AV* and *Run_IND* with reversed roles.
The value of the velocity to be communicated to the inverter is selected depending on the direction of the movement: *Velocita_Avanti* if it is running forward or *Velocita_Indietro* otherwise. The velocity value is passed through the specific inverter variable *AO_Vel_InverterName* after an adequate scaling.

2.2.2 Inverters connected to the PROFINET network

The inverters managed through the Profinet protocol are the ones between U12 and U20. The DBs of these inverters are totally equal between each other and have just some little differences between the ones explained in the precedent subsection. The FCs of the inverters are totally equal, just the numbers 16, 17, 18 and 20 have some differences.

Inverters connected to the PROFINET network Data Structure - DB

The DBs of these inverters are pretty equal to the Data Structure explained in the precedent subsection and it is composed by four different structures:

- ***CMD***, that comprehends all the commands of the inverter. They are:
  - *Enable*, the command that enables the movement of the motor;
  - *Run_AV* and *Run_IND*, the commands used to run the motor forward and backward;
  - *Extracorsa_AV* and *Extracorsa_IND*, two bits used to check if the motor is arrived at the maximum forward and backward limit;
  - *Reset*, the command that reset the motor.

- ***FBK***, that comprehends the feedbacks that the inverter returns. They are:
  - *Ready*, that says if the motor is ready to work;
  - *Running*, that says that the motor is running;
  - *Fault*, that indicates if the inverter is in fault;
  - *In_Velocity*, that indicates if the inverter has reached the commanded velocity;
  - *Velocita_Attuale* and *Corrente_Attuale*, two real values that indicate the actual velocity and actual current of the motor;
  - *Codice_Errore*, a word that reports the eventual error code of the inverter;
- ***PARAMETRI***, that comprehends the parameters of the inverter. That are:
  - *Velocita_MIN* and *Velocita_MAX*, that are respectively the minimum and maximum velocity of the motor;
  - *Acceleration* and *Deceleration*, that are respectively the acceleration and deceleration of the motor. Their values are limited in the interval [0,100];
  - *Velocita_Avanti* and *Velocita_Indietro*, that are respectively the forward and backward velocity of the motor. They are limited in the interval [*Velocita_MIN*, *Velocita_MAX*].

  Velocita_MIN, *Velocita_MAX*, *Acceleration* and *Deceleration* are copied from the related parameters of *DB700_Params_Inverter*.
- ***INVERTER***, an additional structure of the custom type *UDT_ATV_320* that manages the communication with the inverter in a more elaborated and precise way. The variable specifications can be found in the inverter manual, look at Figure 2.1 for having an idea. The structure is divided into *INPUTS*, that are:
  - the structure *ETA* reports the status of the inverter;
  - *RFRD* reports the value of the actual velocity;
  - *LCR* reports the actual current;
  - *LFT* reports the error code.

  and *OUTPUTS*, that are:
  - *CMD* reports some command bits;
  - *LFRD* reports the commanded velocity;
  - *ACC* and *DEC* report the parameters of acceleration and deceleration;
  - *HSP* and *LSP* report respectively the maximum and minimum frequency of the motor.

DB712_U12_Roller_Div3

	Nome	Tipo di dati
1	Static	
2	INVERTER	"UDT_ATV_320"
3	INPUT	Struct
4	PAR_MNG	Struct
5	PKE	Word
6	R_W	Word
7	PWE_2	Word
8	PWE_1	Word
9	ETA	Struct
10	Bit_8	Bool
11	Bit_9	Bool
12	Bit_10	Bool
13	Bit_11	Bool
14	Bit_12	Bool
15	Bit_13	Bool
16	Bit_14	Bool
17	Bit_15	Bool
18	Bit_0	Bool
19	Bit_1	Bool
20	Bit_2	Bool
21	Bit_3	Bool
22	Bit_4	Bool
23	Bit_5	Bool
24	Bit_6	Bool
25	Bit_7	Bool
26	RFRD	Word
27	LCR	Word
28	LFT	Word
29	spare_w_1	Word
30	spare_w_2	Word

DB712_U12_Roller_Div3

	Nome	Tipo di dati
1	Static	
2	INVERTER	"UDT_ATV_320"
3	INPUT	Struct
31	OUTPUT	Struct
32	PAR_MNG	Struct
33	PKE	Word
34	R_W	Word
35	PWE_2	Word
36	PWE_1	Word
37	CMD	Struct
38	Bit_8	Bool
39	Bit_9	Bool
40	Bit_10	Bool
41	Bit_11	Bool
42	Bit_12	Bool
43	Bit_13	Bool
44	Bit_14	Bool
45	Bit_15	Bool
46	Bit_0	Bool
47	Bit_1	Bool
48	Bit_2	Bool
49	Bit_3	Bool
50	Bit_4	Bool
51	Bit_5	Bool
52	Bit_6	Bool
53	Bit_7	Bool
54	LFRD	Word
55	ACC	Word
56	DEC	Word
57	HSP	Word
58	LSP	Word

Figure 2.1: Example of structure ***INVERTER*** took from *DB712_Roller_Div3*, which is related to the inverter that control the roller of the third diverter. On the left there are the details of the INPUTS, while on the right there is the OUTPUT structure.

Inverters connected to the PROFINET network - FC

The first rank of the FC is used to read the inputs. The structure *INPUTS* is filled reading a series of words, the ones called Ux_IW_y where y goes from1 to 10, as shown in the example of Figure 2.2.

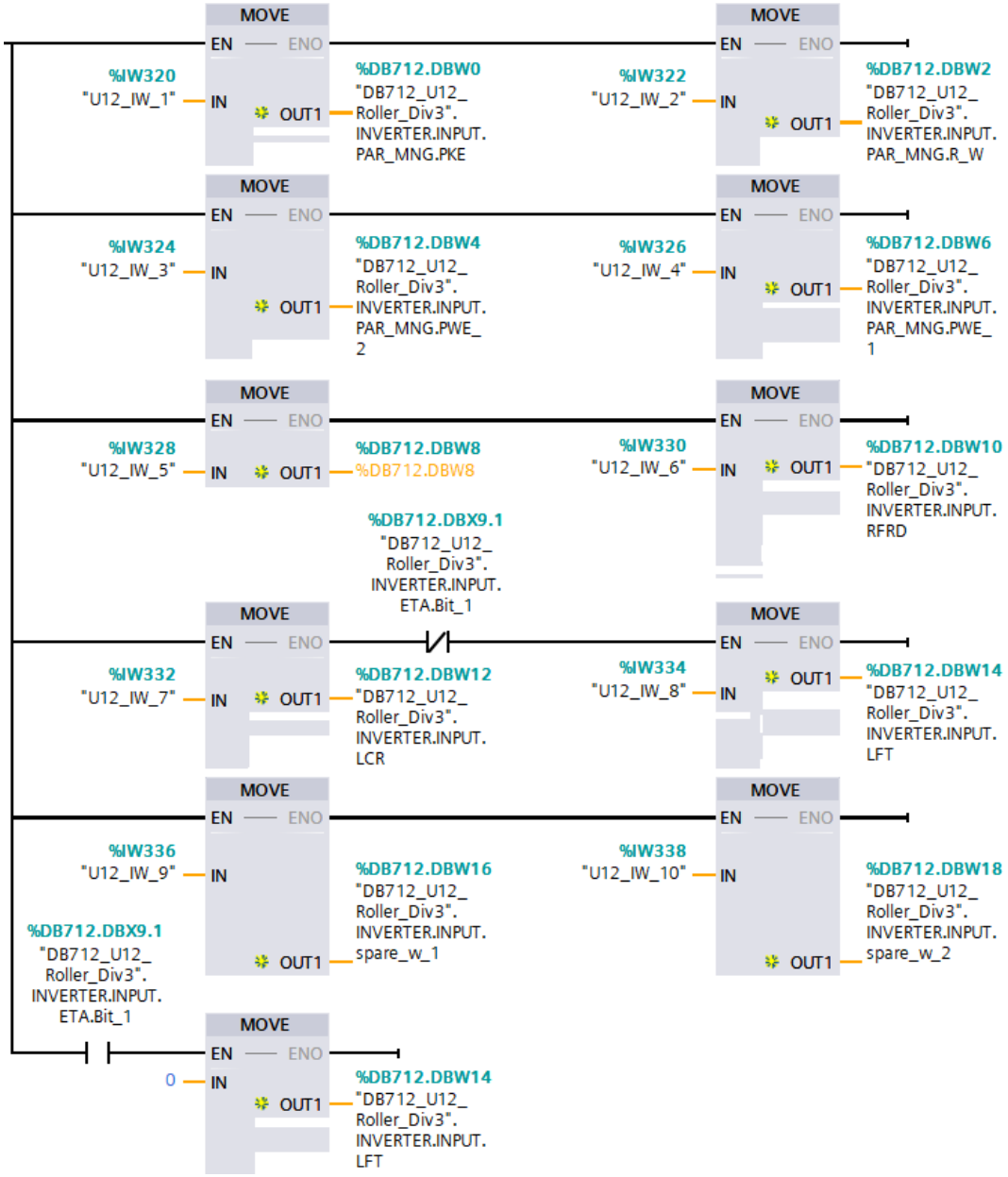


Figure 2.2: Example of filling of the structure *INPUTS* in *FC712_U12_Roller_Div3*

The data are then storage in a more understandable way filling the other structures:

- the inputs *RFRD*, *LCR*, and *LFT* are copied respectively onto *Velocita_Attuale*, *Corrente_Attuale* and *Codice_Errore* of structure *FBK*;
- *Ready* is related to the bit *ETA.Bit1*;
- *Running* is *true* when the inverter is ready and also *ETA.Bit_2* is *true*;
- *Fault* is *true* when one between *ETA.Bit_3* and *ETA.Bit_7* is *true*;
- *In_Velocity* is *true* when the motor is *running* and *ETA.Bit_10* is *true*.

The command *Reset* is related to the general *RESET*.
The commands *Extracorsa_AV* and *Extracorsa_IND* are not considered in general since the most of the inverters control roller or conveyor-belts that can go forward or backward infinitely and so they do not have limit positions. Only the inverters numbers 16, 17, 18 and 20, related respectively to the elevator, to the shovel, to the cart of the loading system and to the cart-pusher, have the management of the overruns. *Extracorsa_AV* is related to the input sensor of its station, for example in the logic of the inverter U16 it is equal to *DB102_Loader_GEN.INPUT.Soll_Extra_ALTO*.
The commands *Run_AV* and *Run_IND* are managed in the same way of the one used in the *Directly controlled inverter Logic*, just the inverters U14 and U16, the ones related to the positioners, are controlled in a different way. For them, *Run_AV* and *Run_IND* are respectively related to *CINGHIE_AV* and *CINGHIE_IND* for U14, to *SOLLEV_AV* and *SOLLEV_IND* for U16.
The same reasoning is applied to the parameters *Velocita_Avanti* and *Velocita_Indietro*, just for the inverters U14 and U16 they are both related respectively to *CINGHIE_Velocita* for the first and to *SOLLEV_Velocita* for the second.

At this point the computed outputs are translated into the structure *OUTPUTS*, that has the specific data type that the inverter needs.
The following actions are performed:

- *CMD.Bit_0* represents the forward running of the inverter and it is *true* when the inverter is *ready*, when there is the command for the forward running *Run_AV*, but not the commands *Run_IND* and the *Extracorsa_AV*;
- *Bit_1* represents the backward running and it is managed exactly as *Bit_0*;
- *Bit_7* represents the command *Reset* and it is active when the inverter is in *Fault* and the command *Reset* is given;
- the parameters *Velocita_MAX*, *Velocita_MIN*, *Acceleration* and *Deceleration* are moved respectively onto *HSP*, *LSP*, *ACC* and *DEC* of the structure *OUTPUT*.

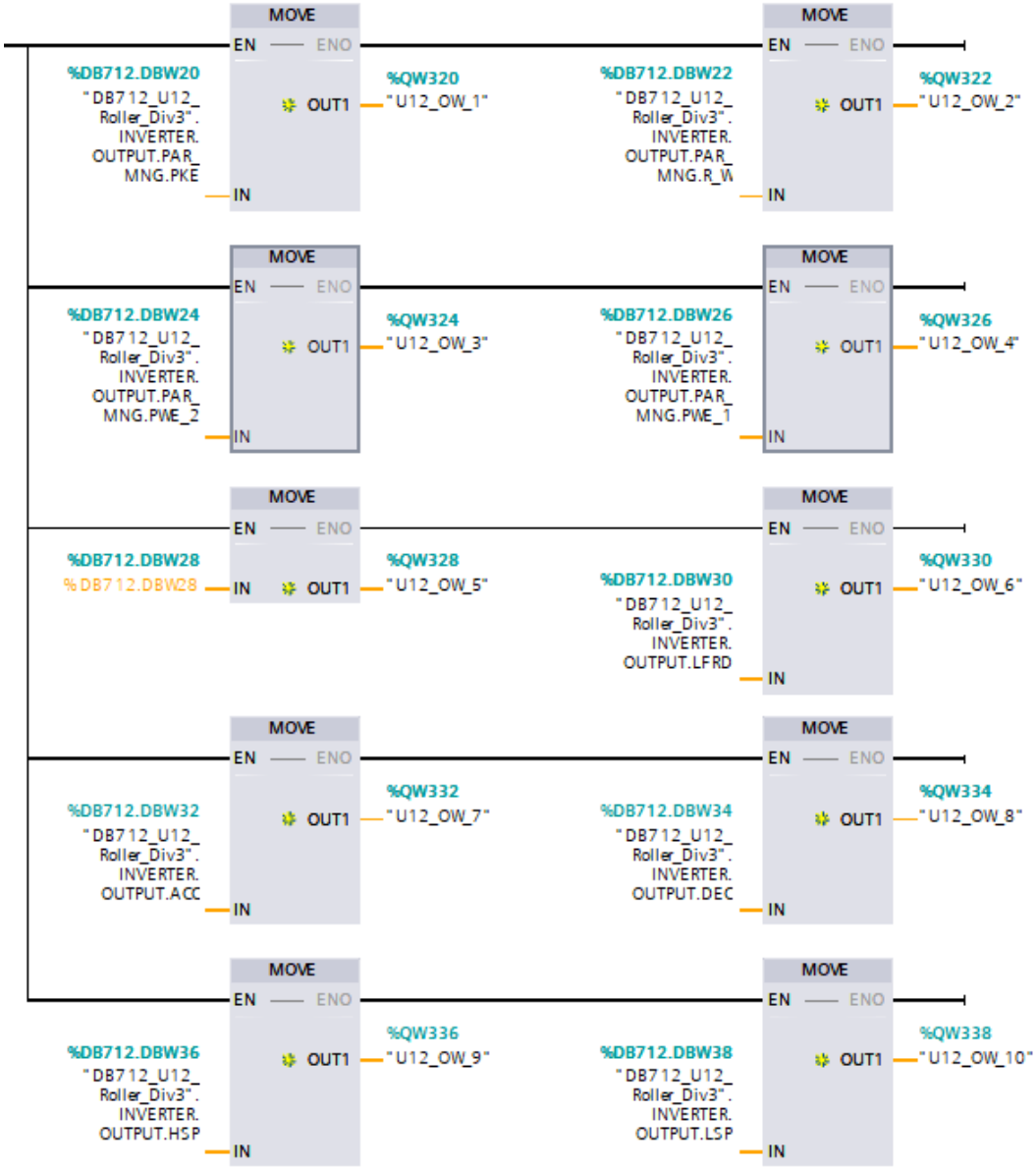


Figure 2.3: Example of writing of the structure *OUTPUTS* onto the output words in *FC712_U12_Roller_Div3*

At the end the outputs are written using ten words, called Ux_OW_y where y goes from 0 to 10, as shown in Figure 2.3, and they are passed to the inverter.

2.3 Encoder management

For the management of the encoders a specif couple of a DB and an FC is created for both of them. They are exactly equal for both the encoders, with just the variable related to its specific Data Structure.

Encoder Data Structure - DB

The variables of the *Encoder Data Structure* are organized in four groups.
ENCODER is a structure that is divided in inputs and outputs, both of them have exactly the specifications of the data type that this particular type of encoder needs. The structure *ENCODER.INPUT* includes two real variables and 16 bits, they are:

- two real values *Position* and *Velocity*, that are respectively the measured position and velocity. They are copied from the I/O variables *ENC_x_IN_Pos* and *ENC_x_IN_Vel*;
- *PRESET.Active*, that is linked to *ENC_x_IN_Preset_0*;
- *PRESET.Out_of_Range*, that is linked to *ENC_x_IN_Preset_7*;
- 6 bits called from *PRESET.Spare_1* to *PRESET.Spare_6*, that are linked to the inputs from *ENC_x_IN_Preset_1* to *ENC_x_IN_Preset_6*;
- *STATUS.Error_Position*, that is linked to *ENC_x_IN_Status_0*;
- *STATUS.Error_Memory*, that is linked to *ENC_x_IN_Status_1*;
- *STATUS.Error_Parametrization*, that is linked to *ENC_x_IN_Status_2*;
- *STATUS.Error_Status_Range*, that is linked to *ENC_x_IN_Status_3*;
- *STATUS.Error_Unspecified*, that is linked to *ENC_x_IN_Status_4*;
- *STATUS.Spare_5*, that is linked to *ENC_x_IN_Status_5*;
- *STATUS.Ready*, that is linked to *ENC_x_IN_Status_6*;
- *STATUS.Error_Collective*, that is linked to *ENC_x_IN_Status_7*.

The structure *ENCODER.OUTPUT* comprehends a real variable and 8 bits for the preset, then there are other 8 bits for the control. They are:

- the variable *PRESET.Preset_Position*;
- a bit *PRESET.Preset_Command*;
- 7 bits called from *PRESET.Spare_1* to *PRESET.Spare_7*;
- *CONTROL.ACK_Error*;
- 7 bits called from *CONTROL.Spare_1* to *CONTROL.Spare_7*.

CMD includes the commands. They are *Reset* and *Zero* that respectively resets and sets to zero the encoder.
FBK comprehends the feedbacks of the encoder. They are:

- a bit *Ready*, it indicates that the encoder is ready to work;
- a bit *Fault*, it indicates that the encoder is in fault;
- a real value for the *position*, it indicates the actual position of the object related to the encoder;
- a real value for the *velocity*, it indicates the actual velocity of the object related to the encoder;

These are, practically, the same information of the structure *INPUT* but organized in such a way to manage them better.
PARAMS includes only a *Preset* parameter.

Encoder Logic - FC

As first, the structure *INPUT* is filled as explained in the *Encoder Data Structure*.

Then it is re-organized in the ***FBK*** structure. *Ready* is linked to the the encoder feedback *STATUS.Ready*, while *Fault* is active when there is any error, so if at least one among the bits of *INPUT.STATUS* that indicate an error is active. The *position* and *velocity* feedbacks are respectively copied from the values of position and velocity of the structure *INPUT*.

At this point the commands are managed. The command *Reset* is related to the global *RESET*, while the *Zero* to a global command that is *ENC_CINGHIE_Zero* or *ENC_SOLLEV_Zero* depending on which encoder is considered. The *Preset* parameter of structure *PARAMS.* is set equal to *POSIZIONAMENTO.PARAMETRI.Pos_Zero* of the DB where the specific encoder is controlled.

As soon as *CMD.Zero* is active the value of *PRESET.Preset_Position* is set equal to the *Preset* parameter, and if the preset is inactive - so *PRESET.Active* is *false* - the *PRESET.Preset_Command* is set.
When the preset is active for a little bit of time, the preset command and the command zero are reset.
the *CONTROL.ACK_Error* turns *true* when the encoder is in fault and the command *Reset* is activated.

At the end the structure *OUTPUT* is copied into the data type that the encoder needs. The I/O variable *ENC_x_OUT_Preset* is copied from *PRESET.Preset_Position*, while ENC_x_OUT_Preset_Mode is set equal to *PRESET.Preset_Command*. All the other 7 preset bits are copied into the related I/O variable *ENC_x_OUT_Preset_y*, where y goes from 1 to 7.

2.4 Positioner management

To handle the positions on the grouper and on the elevator a specific couple of an FC and a DB is built for each positioner.

2.4.1 Positioner Data Structure - DB

The DB is organized in three structures: *INPUT*, *POSIZIONAMENTO* and *OUTPUT*. ***INPUT*** contains all the signals and values to command the positioner. They are:

- *Enable*, a signal that says if the positioner is enable to work;
- *Sensore_Home*, that represents the physical sensor of the positioner if it is present. In the elevator there is a photocell that marks the home position;
- *Start_Home*, a bit that allows to start the homing procedure;
- *Stop*, a bit to book the stop;
- *Jog_AV* and *Jog_IND*, two bits that command the movement in both direction;
- *Extra_AV* and *Extra_IND* are the signals of the sensors which are located at the extreme positions of the positioner.
- *Pos_Attuale*, is the value of the actual position. It is set equal to the feedback *position* provided from the related encoder.

POSIZIONAMENTO is the structure that really commands the positioner. It is composed by:

- *PARAMETRI*, that comprehends all the parameters of the positioner. They are:
  - *Tolleranza*, the tolerance error on the position;
  - *Inerzia_AV* and *Inerzia_IND*, respectively the values of the inertia when running forward and backward;
  - *Dist_Rall*, the distance to the target position at which the movement starts to decelerate;
  - *Pos_Zero*, the home position;
  - *Pos_Min* and *Pos_Max*, respectively the minimum and maximum position that the positioner can reach;
  - *Vel_Homing* and *Vel_Minima*, respectively the velocity value when the positioner is going in the home position and its minimum velocity;
  - *Vel_Jog*, the velocity of the movements when reaching a position. It is copied from *DB700_Params_Inverter.U[x].Velocity_MAN*, the value of manual velocity of the x inverter. It is U14 for the grouper positioner and U16 for the elevator one.

- *FBK* is a group of bits and numbers that encapsulate information about the state of the positioner. They are:
  - *Pos_Attivo* and *Homing_Attivo*, two bits that respectively say that the positioner is set to reach any position and that the positioner is set to reach the home position;
  - *Jog_Attivo*, a bit that says if the positioner is in manual mode;
  - *Homing_Done*, a bit that says if the homing procedure has been completed;
  - *Rall_Attivo* a bit that says if the positioner is decelerating because the actual position belongs to the interval [*Pos_Target*-*Dist_Rall*, *Pos_Target*] if the actual position is smaller than the target one, otherwise the interval is [*Pos_Target*, *Pos_Target*+*Dist_Rall*];
  - *Pos_Attivo_Num* and *In_Pos_Num_Attuale*, two integers that respectively report the number of the position that the positioner must reach and the number of the actual position of the positioner;
- a series of eight positions *POS_x*, one for each of the seven floors of the shelf and one for the loading floor. Each position has:
  - two parameters *Pos_Target* and *Vel_Target*, that respectively represent the position of the target and the velocity that the positioner must have reaching that position. The target position is limited in the interval [*Pos_Min*, *Pos_Max*];
  - *Go_to_Pos*, a bit that activate the positioning for this specific position;
  - *Active*, a bit that remains activated as long as the positioner is moving to the target position;
  - *In_Pos*, a bit that is set to 1 when the positioner reaches the target position.

OUTPUT, that comprehends the commands of the positioner. They are:

- *Marcia_AV* and *Marcia_IND* to control the movement in both directions;
- *Zero* to reset the position counter;
- *Vel*, a value to pass the desired velocity of the movement to the inverter.

The elevator positioner used all these signals, while the positioner of the belts does not considered the homing procedure and all that is related to it, including the extreme positions as it can move in a continuous cycle.

2.4.2 Positioner Logic

Since there are some difference between the two positioner the logic of each one is explained singularly.
Note that the elevator positioner is related to the inverter U16 and to the encoder 2, while the positioner of the grouper is related to the inverter U14 and to the encoder 1.

Elevator Positioner Logic - FC

The first ranks of the FC are used to fill all the ***INPUT*** values.
The positioner is *enabled* (*Enable*) when both the inverter and the encoder related to it are ready (*DB716_U16_Elevator.FBK.Ready* and *DB802_Enc2_Elevator.FBK.Ready* respectively).
The input *Stop* is set in three cases: from the operator panel (*DB102_Loader_ GEN.HMI.* Stop); if the elevator reaches the limit positions (*Extra_AV* or *Extra_IND*); when the emergency occurs.
The inputs *Extra_AV* and *Extra_IND* are equal to the relative inputs *Soll_Extra_ALTO* and *Soll_Extra_BASSO* of *DB102_Loader_GEN*.
Jog_AV and *Jog_IND* are controlled by the related commands of the elevator inverter (*Soll_SALITA* and *Soll_DISCESA* respectively), both in automatic and manual mode, in *FC202*.
The homing procedure can be done only in manual mode and the command to start it *Start_Home* is related to a bit in the logic of the loader station (*Soll_Start_Homing*), while *Sensore_Home* is set equal to *DB102_Loader_GEN.INGRESSI.Soll_Home*.
The activation of the bit *POS_x.Go_to_Pos* marks that the target position is the number x and it is active when the positioner is started (*Soll_Start_Pos*) and the command *Piano_Destinazione* of *DB102_Loader_GEN* is equal to x. This is done exactly in the same way for each position from 1 to 10 and in manual mode.
The value of the actual position *INPUT.Pos_Attuale* is related to the feedback *position* of the second encoder.

Then there is the logic of the homing procedure. The bit *Homing_Attivo* is set, *Homing_Done* is reset and *Pos_Attivo_Num* is set to zero - the home position - when the positioner is enabled, is not stopped, there are no other active procedures and when the command *Start_Home* is *true*.
When the homing procedure is started the temporary variable *Tmp_Homing_IND* is true as long as *Sensore_Home* is not reading.
At this point, when the home procedure is still active and the home sensor starts reading *In_Pos_Num_Attuale* is set to zero, the bit *Homing_Attivo* is reset and if the *Stop* is not active the procedure is set as completed (*Homing_Done*) and the command *Zero* is given. When when the stop is operated and then removed the homing procedure is set as done and the command *Zero* is given.

The commands of the movements are managed in the following way. A temporary bit *Tmp_Jog_AV*, that active the ascent of the elevator, is active as long as the positioner is enabled, there is not the command stop, the homing and the positioning procedures are not active, there is the input command of ascent *Jog_AV* and there is not the descent command *Jog_IND*. The management of the temporary bit *Tmp_Jog_IND* is done exactly in the same way. In both cases the bit *Jog_Attivo* is active and the value *Vel_Jog* is copied into the temporary variable *Tmp_Vel_Jog*.

Then there is the management of the positioning procedure. The bit *POSIZIONAMENTO.POS_x.Active* is *true* and *Pos_Attivo_Num* is set to x, so the target position is the number x, when the following conditions are verified: the positioner is enabled, the stop command is not active, the homing procedure is not active and it is done, the manual mode is not active and the bit *POSx.Go_to_Pos* is active. This is done for every position *POSIZIONAMENTO.POS_x*.
The bit for the positioner status *Pos_Attivo* is *true* as long as at least one between all the positions *POS_x.Active* is active.
To check if the positioner is in the right position a tolerance window is calculated. For each position a minimum and a maximum height are computed: the minimum is the target position *POS_x.Pos_Target* minus the tolerance value *Tolleranza*, the maximum is calculated in the same way adding the tolerance. At the end, depending on which position is active and has to be reached, the related parameters are copied into temporary variables. If the active position is the *POS_y* its couple of maximum and minimum position is copied into the couple of temporary values *Tmp_Pos_max* and *Tmp_Pos_min*, while the related values of target position *POS_y.Pos_Target* and target velocity *POS_y.Vel_Target* are copied into other two temporary values called respectively *Tmp_Pos_Target* and *Tmp_Vel_Target*.

At this point some calculation on the target position are needed, in order to consider the inertia of the movements and the slow down when reaching it.
If the actual position *Pos_Attuale* is bigger than the target one and so the the elevator must go up, it is adjusted subtracting the parameter of the ascent inertia *Inerzia_AV*, otherwise it is adjust adding the parameter of the descent inertia *Inerzia_IND*. The new value is stored into *Tmp_Pos_Target_Calc*.
It must be calculated also the position in which the slow down have to start. If the actual position is bigger than the target one, a temporary variable *Tmp_Pos_Rall* is set as *Tmp_Pos_Target* minus *Dist_Rall*, otherwise it is set as *Tmp_Pos_Target* plus *Dist_Rall*.
The slow down is active (*Rall_Attivo*) if the actual position belongs to the interval [*Tmp_Pos_Rall*, *Tmp_Pos_Target*] in the case that the elevator is going up, otherwise it is active when the actual position belongs to the interval [*Tmp_Pos_Target*, *Tmp_Pos_Rall*] in the case that the elevator is going down.
The velocity is scaled linearly from *Tmp_Vel_Target* to *Vel_Minima* in the interval [*Pos_Rall*, *Tmp_Pos_Target*] and stored into *Tmp_Vel_Rall* when the slow down option is active.
A temporary bit *Tmp_Pos_AV*, that active the ascent of the elevator, is active as long as the positioner is enabled, the positioning is active and the actual position *Pos_Attuale* is smaller than the calculated target one *Tmp_Pos_Target_Calc*, so the elevator has not yet reached the target position. The temporary bit *Tmp_Pos_IND* is managed in the same way, checking if the actual position is bigger than *Tmp_Pos_Target_Calc*.

At this point the output *Marcia_AV* is active when the positioner is enabled, when the sensor *Extra_AV* is not reading and one between *Tmp_Jog_AV* and *Tmp_Pos_AV* is active. The output *Marcia_IND* is managed in the same way with *Tmp_Homing_IND*, *Tmp_Jog_IND* and *Tmp_Pos_IND*.

Then some feedbacks are managed. The bit *POSIZIONAMENTO.POS_x.In_Pos*, the one that marks when the elevator is in the specific position x, is active when the actual position is in the interval [*Pos_min_x*, *Pos_max_x*], calculated above.
The bit that marks that a position is the target one *POS_x.Go_to_Pos* is reset when the general positioning is active (*Pos_Attivo*), when the positioning of the specific position *POSIZIONAMENTO.POS_x.Active* is active and when the elevator is marked to be in that position (*POSIZIONAMENTO.POS_x.In_Pos*), in practice when the elevator has reached the desired position. This is done for every *POS_x*.
When *POSIZIONAMENTO.POS_x.In_Pos* is active the feedback on the actual number position *POSIZIONAMENTO.FBK.In_Pos_Num_Attuale* is set equal to x.

At the end there is the control of the output. The output velocity depends on which procedure is active. If the positioning is active and the slow down is not active *Vel* is set equal to *Tmp_Vel_Target*, otherwise equal to *Tmp_Vel_Rall*.
If the jog mode is active *Vel* is set equal to *Tmp_Vel_Jog*, while if the homing procedure is active it is set equal to *Vel_Homing*.
The ascent and descent commands *Marcia_AV* and *Marcia_IND* are related respectively to the general variables *SOLLEV_AV* and *SOLLEV_IND*, the command *Zero* to *ENC_SOLLEV_Zero* and *Vel* is copied into *SOLLEV_Velocita*.

Grouper Positioner Logic - FC

The most of the logic of the grouper positioner is pretty much equal to the elevator positioner one, in the following only the differences between the two are reported. Remember that all that was related to the inverter U16 and to the encoder 2 is now related to the inverter U14 and to the encoder 1.

The input *Stop* is set always to false, as well as the inputs *Extra_AV* and *Extra_IND*, since the object that is controlled is a conveyor that can run infinitely.
Jog_AV and *Jog_IND* are controlled by the related commands of the conveyor-belts inverter (*Raggr_Cinghie_2_AV* and *Raggr_Cinghie_2_IND* respectively), both in automatic and manual mode, in *FC101_Transportation_GEN*.
The value of the actual position *INPUT.Pos_Attuale* is related to the feedback *position* of the first encoder.

The homing procedure is used to reset the position. In practice, *Raggr_Cinghie_2_Azz* is related to *Start_Home* and *Sensore_Home* - since it is not present - is set always to *true*, while everything else in the homing procedure remains the same.
The activation of the bit *POS_x.Go_to_Pos* is done exactly in the same way with the only difference that x is now compared with the command *Pos_Num_Target* of *DB101_Transportation_GEN*.

Obviously also the writing of the FC outputs is different. The forward and backward running commands *Marcia_AV* and *Marcia_IND* are related respectively to the general variables CINGHIE_AV and CINGHIE_IND, the command *Zero* to ENC_CINGHIE_Zero and *Vel* is copied into CINGHIE_Velocita.

2.5 Stations

Each station has pretty much the same structure: there are three FCs and two DBs.
The FCs are:

- *General*, that contains the logic to read the input signals, to do some calculation or timeout and to create the alarms of the station;
- *Cyclic*, that contains the logic to perform the automatic cycle of the station. It comprehends the management of some standard signals, of the cycle steps to perform, of the feedbacks of the station and of the commands;
- *Output*, that writes the outputs.

The DBs are called *General* and *Cyclic*.
The DB *General* contains all the variables related to the station organized in different structures, that are:

- *INGRESSI*, that stores all the signals of the physical sensors;
- *AUTO_CMD*, that comprehends all the commands of the singular components of the station;
- *MAN_CMD*, that are exactly the same in *AUTO_CMD* but they are used when the system is in manual mode;
- *ALLARMI*, that stores all the alarms of the station;
- *HMI*, that stores some variables used in the control panel;
- *APPOGGIO*, that is a series of supporting bits or parameters to do calculations;

The other DB *Cyclic* is divided in:

- *SIGNALS*, that stores the standard signals of the automatic cycle of the station. They are:
  - *Step*, an integer value used when the station is managed in a sequential mode;
  - *Presenza_Pezzo*, a bit that indicates that there is a piece in that station. The meaning of "piece" varies on each station;
  - *RESET_Stazione*, a bit to reset the specific station;
  - *Stazione_Pronta*, a bit that indicates that the station is ready to work;
  - *Lavoro_Completato*, a bit that indicates that the station has completed its work;
  - *NoStepAttivi*, a bit that indicates that the step value is equal to zero;
  - *Passi_Ciclo_ON*, a bit that indicates that the automatic cycle of the station is enabled;

  - *Uscita_Ciclo_ON*, a bit that indicates that the writing of the outputs of the station is enabled;
  - *Stazione_Esclusa*, a bit that indicates that the station is not enabled.

- *PARAMETRI*, that stores the parameter of the automatic cycle of the station;
- *FBK*, that comprehends the feedbacks of the station.

2.5.1 Transportation

This part of the software is used to control the transportation station, that includes all the conveyor-belts, rollers and diverters. This station has a little difference between the standard composition explained above, it has an additional DB called *Timer*.
The automatic logic of the station is managed in a combinatorial mode, so each action is performed every time looking at the specific combination of the inputs. In practice a specific set of inputs determine a specific set of outputs.

Transportation Data Structure - General DB 101

The structure ***INGRESSI*** is composed by the signals of the sensors:

- *B19_Ingresso_Prodotto* the photocell that says when there are bricks at the starting point of the system;
- *B1_Deviatore_1_Pieno* and *B9_Deviatore_2_Pieno* the photocells that say when there are bricks respectively on the first and on the second diverter;
- *B2_Deviatore_1_Stop*, *B10_Deviatore_2_Stop* and *B13_ Deviatore_3_Stop*, the photocells that say when the bricks have reached the end of the roller of respectively the first, the second and the third diverter;
- *B3_FineCinghieDev1*, *B14_FineCinghieDentate_1* and *B15_EmergCinghie Dentate_2*, the photocells that say when the bricks have reached the end of the conveyor-belts of the first, third and the fourth diverter;
- *B4_FineCinghie_1* to *B8_FineCinghie_5* the photocells that say when the bricks have reached the end of the related conveyor-belt;
- *B11_FineRulliera_1* and *B12_FineRulliera_2* the photocells that say when the bricks have reached the end of the first and of the second roller;
- *B16_UscitaRaggruppatore* the photocell that says when the bricks have reached the end of the roller of the fourth diverter and are ready to be stored in the shelves;
- *B17_FineNastroCaricatore* and *B18_CaricatorePieno* the photocells that say when the bricks have reached the end of the conveyor-belt of the loading system and when there are bricks on it;
- *Dev_x_ALTO* and *Dev_x_BASSO* the photocells that say when the x diverter is respectively in the high and low position;

The structure ***AUTO_CMD*** is the list of automatic commands of the station:

- *Dev_x_SALITA* and *Dev_x_DISCESA*, *Rull_Dev_x_AV* and *Rull_Dev_x_IND*, *Cinghia_Dev_x_AV* and *Cinghia_Dev_x_IND*, that command respectively the ascent and the descent of the x diverter, the forward and backward running of the roller of the x diverter and the forward and backward running of the conveyor-belt of the x diverter;

- *Raggr_Cinghie_1_AV* and *Raggr_Cinghie_1_IND*, *Raggr_Cinghie_2_AV* and *Raggr_Cinghie_2_IND* they are the specific commands respectively of the forward and backward running of the conveyor-belt of the third and fourth diverter;
- *Raggr_Cinghie_2_Start_Pos* and *Raggr_Cinghie_2_Azz*, that respectively starts and resets the positioning of the bricks in the grouper, while *Pos_Num_Target* is an integer that represents the target floor of the grouper positioner.

The structure ***MAN_CMD*** is the list of manual commands of the station and they are exactly equal to the ones in ***AUTO_CMD***.
The structure ***ALLARMI*** comprehends the alarms of the station, they are:

- *AL01*, *AL02*, *AL03*, *AL04*, *AL05*, *AL06*, *AL07*, *AL08*, *AL09*, *AL10*, the alarms that rise when the the transfers respectively IN-0, 0-1, 5-6, 6-6A, 6-7, 7-8, 8-9, 7-7A, 9-10, 10-11 are active for too much time and so there is something wrong;
- *AL11* to *AL14*, the alarms that rise when the physical input of the signal that indicates that a diverter is high while the bit of this sensor is not for more than a determined time. It indicates that its mechanical cylinder is blocked;
- *AL18* to *AL24*, the alarms that rise when the respective inverter 12 - 18 is at fault;

The structure ***APPOGGIO*** comprehends a lot of bits and variables used as support in the logic. They are:

- a series of bits that indicates that there is an active transfer. They are called *Trasf_x_y*, where x and y are two numbers that indicate a particular roller or conveyor-belt, as indicated in Figure 2.5. Some of them have also a bit called *Prod_Passato_x_y* that marks the transfer as completed; others transfers are managed just with that bit.
  There is also a series of bits that are used to mark an alarm on a transfer, they are called *Alm_Trasf_x_y* and they are one for each transfer. An alarm turns active when a transfer is marked as active for a time bigger than the time parameter *Timeout_Trasferimento*;
- a series of bits used to create the bricks pattern. They are:
  - *Mattoni_Passati*, it is that marks when the bricks are completely on the third diverter during the *Transfer 7 - 8*. It is reset when that transfer is no more active;
  - *Mattone_Singolo*, it is a bit that indicates when there is just one row of bricks at the end of the second roller of the series;
  - *Rulliera_Piena*, it is a bit that indicates when the brick pattern is completed;
  - *Modulo_Completato*, it is a bit that indicated when the creation of a single group of bricks in the pattern is completed;
  - *Spare_3*, is the option that selects if there are 4 or 5 rows of bricks on the third diverter.

- two integers *Moduli_Posizionati*, that indicates the number of brick groups already placed, and *CNT_Dev_3*, that counts the number of brick blocks on the roller of the third diverter;
- *Vel_Cinghia_1* is the parameter of the velocity of the conveyor-belts of the third diverter, while *K_Differenza_Vel_Cinghie* is the parameter of scaling between the velocities of the conveyor-belts of the grouper.

Transportation Data Structure - Cyclic DB201

The structure ***SIGNALS*** is exactly the standard one described above, where *Presenza_Pezzo* and *Lavoro_Completato* are not used.
The structure ***PARAMETRI*** comprehends:

- a series of times to set the timers for the transfers and the alarms. They are:
  - *Timeout_Trasferimento*, it is a time parameter that it is used in two cases. It is the time limit after that an alarm is raised when a transfer is active; the time after that a transfer is reset; it is also used in *Transfers 1 - 6* as a time after that the bricks have reached the following conveyor;
  - *RitardoSalita_RullieraDev3*, the time after that the bricks are considered as arrived during *Transfer 8 - 9*;
  - *T_PassaggioMattoni_Rull2*, the time parameter of the timer that marks when the bricks are not on the second roller of the series anymore because they are on the third diverter during the *Transfer 7 - 8*;
  - *T_Check_Sensori*, it is the time parameter used in the logic of the alarms *AL11* to *AL14*. They are raised after that value of time when the related actuation is not effective;
  - *T_Marcia_Nastro_IN*, the time after that a bit to marks that there are no bricks in the entering point is activated.
- a series of real values which are used to do calculations about the bricks pattern. They are:
  - *Larghezza_Mattone*, the width of a single brick. It is represented by *A* in Figure 2.4;
  - *File_x_Gruppo*, the number of bricks rows in a single group;
  - *Distanza_Fila*, the distance between two rows in a group. It is represented by *B*;
  - *Distanza_Gruppo*, the distance between two groups. It is represented by *C*;
  - *Larghezza_tot*, the total space that can be used to put the bricks of the pattern. It is represented by *D*.

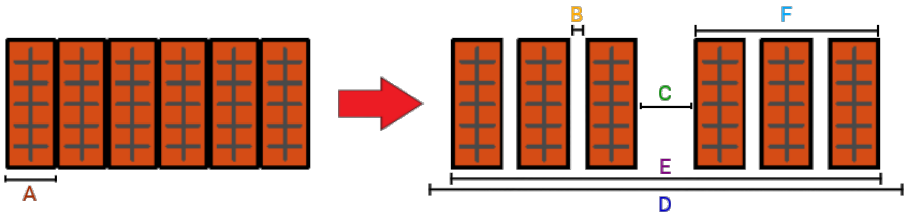


Figure 2.4: Scheme for the creation of the bricks pattern

The structure ***FEEDBACK*** comprehends some results of calculations that were done to create the pattern. They are:

- *Ingombro_Mattoni*, the occupied space in a group by the bricks. It is computed multiplied the width of a single brick *Larghezza_Mattone* by the number of bricks in a group *File_x_Gruppo*;
- *Ingombro_Offset*, the occupied space in a group by the free space between rows. It is computed multiplied the space between each brick in a group *Distanza_Fila* by the number of spaces in a group, that is the number of bricks minus 1 (*File_x_Gruppo* - 1);
- *Larghezza_Gruppo*, the needed space for a group of bricks. It is represented by *F* in Figure 2.4 and it is computed as *Ingombro_Mattoni* plus *Ingombro_Offset*;
- the couple *N_Gruppi*, the approximated number of groups that fit on the conveyor-belt, and *Spazio_Necessario*, the actually busy space by the pattern on the conveyor-belt, that are linked to each other.
  In practice, as first *N_Gruppi* is computed. It is the integer approximation of the result of the computation that divides the total space on the roller *Larghezza_tot* by the sum of the width of each group *Larghezza_Gruppo* and the distance between each group *Distanza_Gruppo*. Using the letters of the scheme it is *N_Gruppi* = ROUND[D/(F+C)]. This is not a precise result, as it may differ by one unit from the exact number of groups that fit in the available space, for this reason now the real occupied space with this number of groups is computed.
  Spazio_Necessario is computed as the multiplication of the sum of the width of each group *Larghezza_Gruppo* and the distance between each group *Distanza_Gruppo* by the computed number of groups *N_Gruppi*, minus the *Distanza_Gruppo*.
  As a formula it is *Spazio_Necessario* = [(F+C)**N_Gruppi*] - C. In Figure 2.4 is the letter *E*. If the results is bigger than the available space *Larghezza_tot* the number of groups is decreased by one and *Spazio_Necessario* is computed again with the new value.
- *Spazio_Libero*, the free space left. It is computed as the difference between the available space *Larghezza_tot* and the occupied space *Spazio_Necessario*. In Figure 2.4 it is *D* - *E*;

Transportation Data Structure - Timer DB 301

This DB contains just some values of time, they are the delay parameters for: the start of the roller (*Rulliera_Dev_3.Rit_Start*) and of the belts (*Cinghia_1_Raggr.Rit_Start*) of the third diverter; for the start and the stop of the elevator of each diverter.
Every time that their moving commands are given, their activation is delayed by that value of time.

Transportation Logic

As first, the structure ***INGRESSI*** is filled with the signals of the sensors. Each signal is filtered with a timer set on the specific sensor.
After that, the management of the output O_Richiesta_Prodotto is done. It reports that there are no bricks at the entering point of the system and it turn *true* when the roller of the first diverter is running forward for a time equal to the parameter *T_Marcia_Nastro_IN*, indicating that no transfer has been made in that time between the first diverter and the first conveyor-belt.
The *SIGNALS* of the Transportation station are managed in the following way: as long as the the station is excluded (*Stazione_Esclusa*), the station is ready so *Stazione_Pronta* is active; each time that the command *RESET_Stazione* is given, the variables *Moduli_Posizionati* and *CNT_Dev_3* are set to zero.
The automatic cycle of the Transportation station is managed by combinatorial logic and each brick transition from a mechanical part to the next one is manage as a single transfer. Look at Figure 2.5 to have a schematic idea of the names that were used.
Moreover, the management of the alarms *AL01 - AL10*, the ones related to the transfers, is done here. In practice, an alarm is set if the related transfer is active for a time bigger than the parameter *Timeout_Trasferimento*. The alarm also reset the related transfer.
Note that, to be clearer, the explanation is done in a sequential way, but remember that every action depends just on some conditions that do not links the entire station. The automatic cycle is explained starting from the entering point of the system and going on sequentially on the next mechanic part. Note also that each action is performed only if *Passi_Ciclo_ON* is active.

IN - 0 The bricks enter the system from an external roller, labelled as *"IN"* in Figure 2.5, and they are moved on the roller of the first diverter, labelled as *"0"*.
The *Transfer IN - 0* is set when all the following conditions are all verified:

- there are bricks at the end of the external roller. So *B19_Ingresso_Prodotto*, the related sensor, is reading;
- the fist diverter is empty, so *B1_Deviatore_1_Pieno* is reading;
- the elevator diverter is not moving and it is in high position.

The transfer actives the movement of the roller of the first diverter with the command.

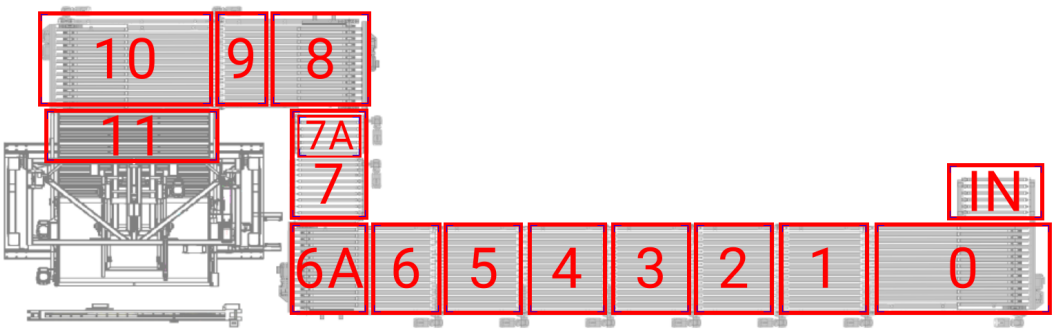


Figure 2.5: Scheme of the transfer names

The transfer is reset, so the roller stops, when the bricks reached the end of the roller of the diverter, so the related sensor *B2_Deviatore_1_Stop* is reading.

Descent 1 At this point the bricks must be put on the conveyor-belt of the diverter. Its descent starts when all the following conditions are all verified:

- the diverter is full, so the related sensor *B2_Deviatore_1_Stop* is reading;
- the roller and the belts of the first diverter are not moving;
- the elevator is not going up and it has not already reached the low position.

When the elevator has actually reached the low position the movement is no more active.

0 - 1 Now the bricks are on the belts of the diverter and must be put on the first conveyor-belt of the series, labelled as *"1"* in Figure 2.5.
The *Transfer 0 - 1* is set when all the following conditions are all verified:

- there are bricks on the diverter, so the related sensor *B1_Deviatore_1_Pieno* is reading;
- the diverter elevator is not moving and it is in low position;
- there are no bricks at the end of the belts of the first diverter, so the related sensor *B3_FineCinghieDev1* is not reading.

The transfer enables the movement of the belts of the first diverter and the bricks are moved to its end.
When they have reached the end - so the sensor *B3_FineCinghieDev1* is reading - the state of the first conveyor-belt is checked: if it is running also the belts of the diverter are moved. This check is done to not hinder the *Transfer 1 - 2*.
The *Transfer 0 - 1* enables also the movement of the first conveyor-belt if the bit *Prod_Passato_1_ 2* - that will be explained in the next transfer - is *false* and as long as the bricks on it have not already reached its end, so as long as the sensor

B4_FineCinghie_1 is not reading.
The bit *Prod_Passato_0_1*, that indicates when the bricks are completely on the first conveyor-belt, is set when the transfer is active and as soon as the sensor on the end of the belts of the first diverter *B3_FineCinghieDev1* does not read anymore. That bit makes the transfer reset, as the bricks have been transferred, and when the transfer is inactive also that bit is reset.

Ascent 1 When the bricks completely arrive on the first conveyor-belt the diverter is empty and it can return in the high position to be load again. Its ascent starts when all the following conditions are all verified:

- the diverter is empty, so the related sensor *B1_Deviatore_1_Pieno* is not reading;
- the roller of the first diverter is not moving;
- the elevator is not going down and it has not already reached the high position.

When the elevator has actually reached the high position the movement is no more active.

1 - 2 The *Transfer 1 - 2* between the first and second conveyor-belt, labelled as *"2"* in Figure 2.5, is manage in a different way. It is made using only the bit *Prod_Passato_1_2*, which marks when the bricks are completely on the second conveyor-belt.
As long as *Prod_Passato_1_2* is *false* and the sensor at the end of the first conveyor-belt *B4_FineCinghie_1* does not read it is moved.
When the bricks have actually reached the end of the conveyor, so *B4_FineCinghie1* is reading, the status of the second conveyor-belt is check: if it is moving also the movement of the first one is activated. In this way, the *Transfer 2 - 3* is not hindered.
The bit *Prod_Passato_1_2* is set when the first conveyor-belt is moving and its end sensor *B4_FineCinghie1* is not reading for a time set equal to the parameter *Timeout_Trasferimento*, so there is nothing that has passed to the second conveyor for that time.
The bit is reset when one of the end sensor between the one of the first conveyor-belt and the one of the belts of the first diverter starts to read again, so there are bricks moving between *0 - 1* or between *1 - 2*.

2 - 5 All the transfers between *2 - 5* are managed exactly as *Transfer 1 - 2*, obviously with the related variables.

5 - 6 At this point there are bricks at the end of the fifth conveyor-belt, which must be put on the second diverter. This transfer put the bricks on the belts of the last diverter in such a way that they do not hider the movement of the elevator.
The *Transfer 5 - 6* is set when all the following conditions are all verified:

- there are no bricks on the second diverter. So *B9_Deviatore_2_Pieno*, the related sensor, is not reading;

- there are bricks at the end of the fifth conveyor-belt, so the related sensor *B8_FineCinghie_5* is reading;
- the diverter elevator is not moving;
- the *Transfer 6 - 6A* is not active.

The transfer enables both the movements of the fifth conveyor-belt and of the belts of the second diverter. The conveyor-belt runs also if the bit *Prod_Passato_5_6* is *false* and the sensor *B8_FineCinghie_5* is not reading; this consider the case in which the bricks were not waiting at the end of the conveyor when the transfer was activated.
The bit *Prod_Passato_5_6* is managed exactly as *Prod_Passato_1_ 2*. So it is set when the fifth conveyor-belt is running and its end sensor *B8_FineCinghie5* does not read for a time equal to the parameter *Timeout_Trasferimento* and it is reset when one between the end sensor of the fourth conveyor *B7_FineCinghie4* or of the fifth conveyor *B8_FineCinghie5* reads anything.
The bit *Transfer 5 - 6* is reset when the presence sensor of the second diverter *B9_Deviatore_2_Pieno* starts to read.
At the end of this transfer the bricks are completely on the second diverter, but not at the end of its belt, in such a way that in this situation the bricks are not on the roller and the diverter elevator can moves without problems. This allows to save time in the transfer.

6 - 6A Now the bricks are completely on the second diverter, but not at the end of its belt; this transfer moves them to end of the belts of the second diverter.
The *Transfer 6 - 6A* is set when all the following conditions are all verified:

- the elevator of the second diverter is not moving and it is in the low position;
- the roller of the second diverter is not moving;
- the *Transfer 6 - 7* is not active;
- there are bricks on the diverter - so the sensor *B9_Deviatore_2_Pieno* is reading - but there are not at its end - so the related sensor *B10_Deviatore_2_Stop* is not reading.

The belts of the second diverter are running forward as long as the bit *Transfer 6 - 6A* is active. It is reset when the bricks have reached the end of the second diverter, so the related sensor *B10_Deviatore_2_Stop* is reading.

Ascent 2 The bricks are at the end of the second diverter and have to be lifted on its roller. The elevator goes up when all the following conditions are all verified:

- there are bricks at the end of the diverter belts, so the related sensor is reading (*B10_Deviatore_2_Stop*);
- the roller and the belts of the second diverter are not moving;
- the *Transfer 6 - 7* the *Transfer 6 - 6A* are not active;
- the elevator is not going down and it has not already reached the high position.

When the elevator has actually reached the high position the movement is no more active.

6A - 7 At this point the bricks are on the roller of the second diverter and they must enter the series of rollers.
The *Transfer 6A - 7* is set when the following conditions are all verified:

- the *Transfer 7 - 8* and the *Transfer 7 - 7A* are not active;
- the elevator of the second diverter is not moving and it is in the high position;
- the bricks are in the right position, touching the side of the diverter - so the sensor *B10_Deviatore_2_Stop* is reading;
- there are no bricks on the series of rollers, so the sensors at the end of each of them (*B11_FineRulliera_1* and *B12_FineRulliera_2*) are not reading.

The transfer *Transfer 6A - 7* enables the movements of the roller of the second diverter and of both the rollers in the series and it is reset when the bricks have reached the end of the second roller of the series, so the sensor *B12_FineRulliera_2* is reading. In this way at the end of the transfer the bricks are on the end of the second roller of the series.

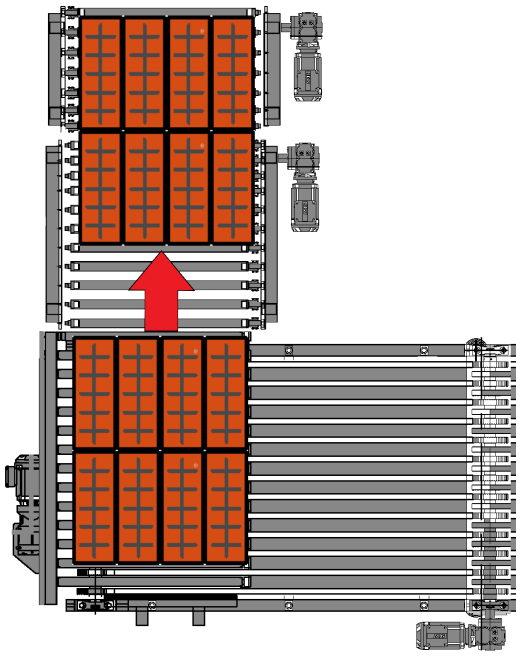


Figure 2.6: Scheme of *Transfer 6A-7*

Descent 2 When there are no more bricks on the roller of the second diverter, it can go down to be loaded again.
Its descent starts when all the following conditions are all verified:

- the *Transfer 6 - 7* is not active;
- there are no bricks on the roller of the third diverter, so the related sensor *B10_Deviatore_2_Stop* is not reading;
- the roller and the belts of the second diverter are not moving;
- the elevator is not going up and it has not already reached the low position.

When the elevator has actually reached the low position the movement is no more active.

7 - 8 Now the bricks are at the end of the second roller of the series and they must go on the third diverter.
The *Transfer 7 - 8* is set when the following conditions are all verified:

- the *Transfer 6A - 7* and the *Transfer 7 - 7A* are not active;
- the elevator of the third diverter is not moving and it is in the high position;
- there are bricks at the end of the second roller of the series - so the sensor *B12_FineRulliera_2* is reading - but there not at the end of the roller of the third diverter - so the related sensor *B13_ Deviatore_3_Stop* is not reading;
- the bit that marks when the bricks are completely on the third diverter *Mattoni_Passati* is *false*.

To be simplify the explanation, the transfer is considered in two cases, depending on which option of *Spare_3*.

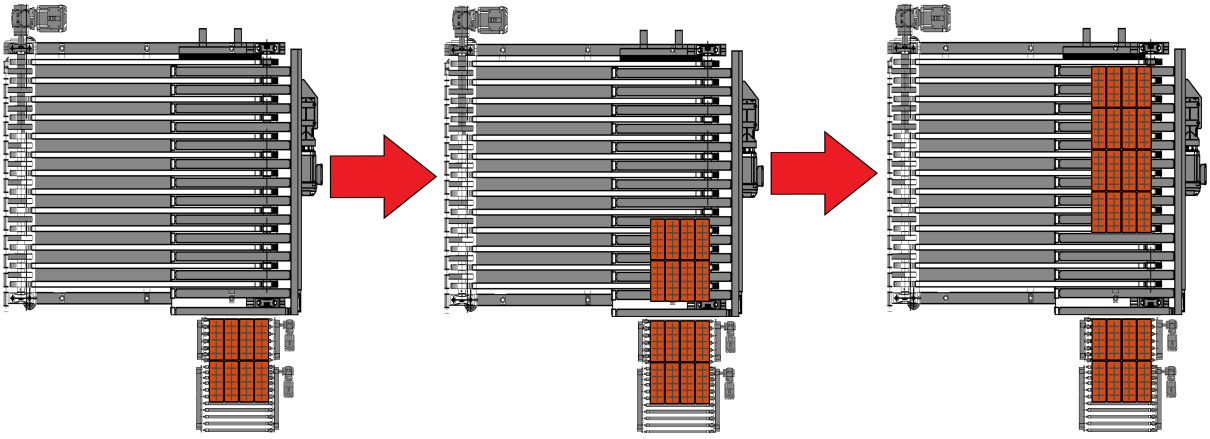


Figure 2.7: Functioning of *Transfer 7 - 8* with 4 rows of bricks

I The option ***Spare_3*** **is activated**, so there must be 4 rows of bricks on the third diverter.
The transfer *Transfer 7 - 8* always enables the movements of the rollers of the

series and of the roller of the third diverter in this condition.
The transfer is reset when the bit *Mattoni_Passati* is *true*, so the bricks are completely on the third diverter, and one of the following conditions is verified: and *CNT_Dev_3* is smaller than 1 or if the sensor at the end of the roller of the third diverter *B13_ Deviatore_3_Stop* reads.
Considering the first activation of the transfer, the variable *CNT_Dev_3* is equal to zero, since there are no bricks on the third diverter. In this case a brick block, so two rows, is transferred completely on the roller of the third diverter, but in its initial part. This makes the transfer to be reset, since *CNT_Dev_3* is smaller than 1.
As soon as the transfer turns to be inactive an there is no the alarm related to the it the variable *CNT_Dev_3* is incremented and it becomes equal to 1.
When the transfer is activated for the second time the second block of bricks is loaded on the third diverter, as already explained.
However, the precedent condition for the reset of the transfer is not verified anymore, since *CNT_Dev_3* is equal to 1, but the transfer is reset when the end sensor of the roller *B13_Deviatore_3_Stop* reads.
When the transfer turns to be inactive the variable *CNT_Dev_3* is incremented again and it becomes equal to 2.
In this way there are 4 rows of bricks at the end of the roller of the third diverter and they are ready to be put on its belt.

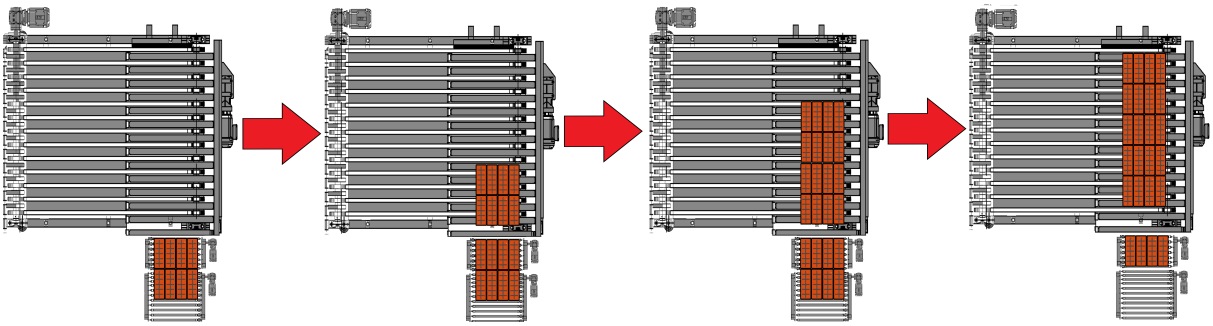


Figure 2.8: Functioning of the first activation of *Transfer 7 - 8* with 5 rows of bricks

II The option ***Spare_3*** **is not activated**, so there must be 5 rows of bricks on the third diverter.
The transfer *Transfer 7 - 8* always enables the movements of the second roller of the series and of the roller of the third diverter. It enables also the running of the first roller of the series but only if one of the following conditions is verified: the variable *CNT_Dev_3* is smaller than 2 or if the bit *Mattone_Singolo* is *true*.
The transfer is reset when the bit *Mattoni_Passati* is *true*, so the bricks are completely on the third diverter, and one of the following conditions is verified:

CNT_Dev_3 is smaller than 2 or if the sensor at the end of the roller of the third diverter *B13_ Deviatore_3_Stop* reads.
On the first two activation the transfer is totally equal to the one explained in point ***I***. So start from the situation in which there are already four rows on the third diverter, but not at its end, and just one is missing. The variable *CNT_Dev_3* is equal to 2.
When a brick block arrives in this conditions at the end of the second roller of the series the the transfer does not active the movement of the first roller of the series, since *CNT_Dev_3* is equal to 2 and the bit *Mattone_Singolo* is *false*. In this way just one row of the block is moved on the third diverter, since the dimension of the second roller of the series is such that it exactly divides a block.
The transfer is reset when the end sensor of the roller *B13_Deviatore_3_Stop* reads, indicating that 5 rows are positioned on the diverter as requested. The variable *CNT_Dev_3* is then equal to 3.
However, it must be considered that there is just a row of bricks at the end of

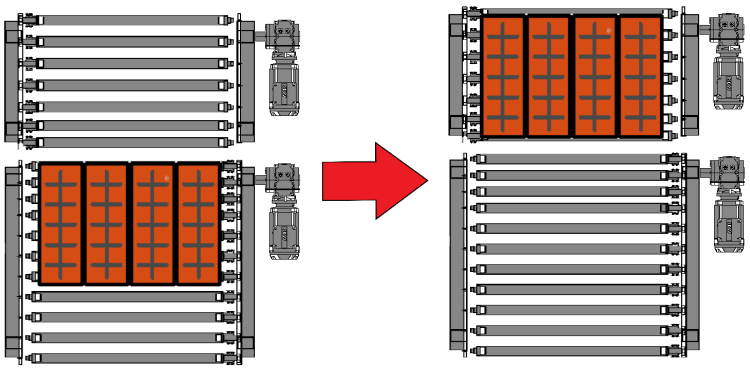


Figure 2.9: Functioning of *Transfer 7 - 7A*

the first roller of the series. In this conditions the *Transfer 7 - 7A*, that moves the bricks to the end of the second roller, is set.
It is activated when the following conditions are all verified:

- the *Transfer 6A - 7* and the *Transfer 8 - 9* are not active;
- the elevator of the third diverter is not moving;
- the belts of the third diverter are not moving;
- there are bricks at the end of the first roller of the series - so the sensor *B11_FineRulliera_1* is reading - but there not at the end of the second one - so the related sensor *B12_FineRulliera_2* is not reading;
- the variable *CNT_Dev_3* is greater than or equal to 3.

The *Transfer 7 - 7A* enables the movement of both the rollers of the series, moving the bricks to its end, and it is reset when the sensor *B12_FineRulliera_2* reads anything. When the *Transfer 7 - 7A* turns to be inactive the bit *Mattone_Singolo* is set, indicating that at the end of this transfer there is a single

row of bricks at the end of the roller series.

At the same time as the *Transfer 7 - 7A* is performed, the elevator of the

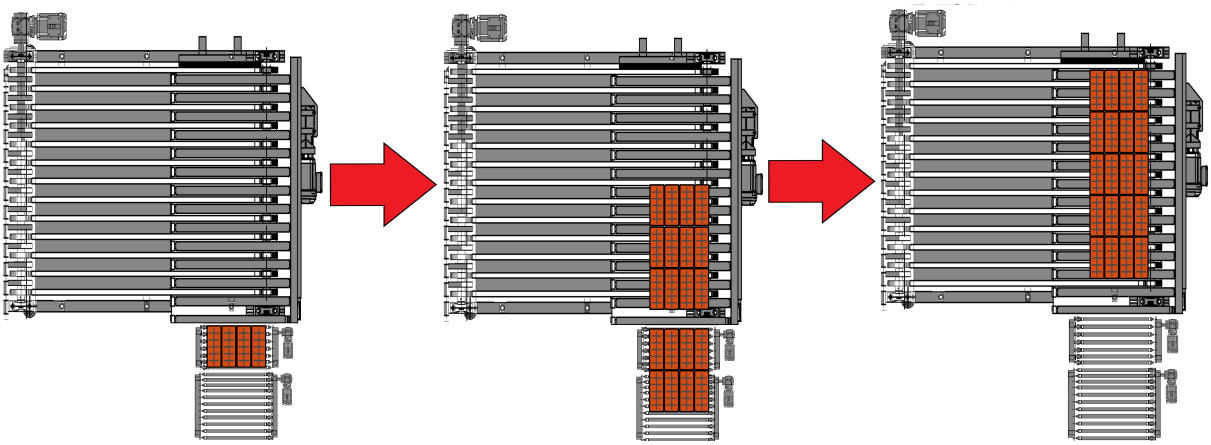


Figure 2.10: Functioning of the second activation of *Transfer 7 - 8* with 5 rows of bricks

third diverter goes down and unload the bricks - look at the following transfers for the details. The variable *CNT_Dev_3* is reset.

At this point the *Transfer 7 - 8* starts again, putting the single row on the diverter. The next two brick blocks are completely load as the bit *Mattone_Singolo* is *true*, so all the roller move.

The bit *Mattone_Singolo* is reset in the final part of *Transfer 7 - 8*. In fact, it is reset when *CNT_Dev_3*, so the diverter is full, and the series of rollers is empty - the related sensors *B11_FineRulliera_1* and *B12_FineRulliera_2* are not reading.

Descent 3 When the roller of the third diverter is completely loaded, it must go down to put the bricks on its belt.

The descent of the third diverter is commanded when the following conditions are all verified:

- the roller and the belts of the third diverter are not moving;
- the *Transfer 7 - 8* and the *Transfer 9 - 10* are not active;
- there are bricks at the end of the roller of the third diverter - so the related sensor *B13_ Deviatore_3_Stop* is reading - but there are not at the end of its belt - so *B14_FineCinghieDentate_1* is not reading;
- the elevator is not going up and it has not already reached the low position.

When the elevator has actually reached the low position the movement is no more active.

8 - 9 The bricks are now on the belts of the diverter and this transfer must move the bricks to the end of the belts of the diverter.
Transfer 8 - 9 is set when the following conditions are all verified:

- the *Transfer 7 - 8* and the *Transfer 9 - 10* are not active;
- the elevator of the third diverter is not moving and it is in the low position;
- the belts of the third diverter are not moving;
- there are bricks at the end of the roller of the third diverter - so the sensor *B13_ Deviatore_3_Stop* is reading - but there not at the end of its belt - so the related sensor *B14_FineCinghieDentate_1* is not reading;
- the bit that marks when the bricks are completely on the third diverter *Mattoni_Passati* is *false*.

The transfer *Transfer 8 - 9* enables the movements of the belts of the diverter.
The bit *Prod_Passato_8_9* is set when the *Transfer 8 - 9* is active and the presence sensor for bricks on the diverter *B13_Deviatore3Stop* does not read for a time set equal to the parameter *RitardoSalita_RullieraDev3*.
The transfer is reset when the bricks have reached the end of the belts of the diverter, so when the sensor *B14_FineCinghieDentate1* is reading. This allows to move the bricks to a position on the belts in such a way that they do no hider the movement of the elevator.
The bit *Prod_Passato_8_9* is reset when the transfer is no more active.

Ascent 3 When the belt of the third diverter is empty, the elevator can go up to be loaded again.
The ascent of the third diverter is commanded when the following conditions are all verified:

- the roller of the third diverter is not moving;
- the *Transfer 7 - 8* is not active;
- *B13_ Deviatore_3_Stop* is not reading and the *Transfer 8 - 9* is not active or *Prod_Passato_8_9* is *true*, so there are definitely no bricks that hider the elevator;
- the elevator is not going down and it has not already reached the high position.

When the elevator has actually reached the high position the movement is no more active.
As soon as the the command for the ascent of the elevator is removed, the variable *CNT_Dev_3* is reset, marking that the roller of the diverter is empty.

9 - 10 The bricks are at the end of the belts of the third diverter and there are no bricks on the last diverter. From this point starts the logic to create the pattern.
The conditions to active the *Transfer 9 - 10* are:

- the *Transfer 8 - 9*, the *Transfer 9 - 10* itself and the *Transfer 10 - 11* are not active;

- the belts of both last two diverters are not moving;
- the last diverter is in the low position;
- the start command of the positioner *Raggr_Cinghie_2_Start_Pos* is inactive;
- the bricks are at the end of the belts of the third diverter - so the sensor *B14_FineCinghieDentate_1* is reading - but the end of the belts of the fourth is free - *B15_EmergCinghie Dentate_2* is not reading;
- the pattern is not complete, so the bit *Rulliera_Piena* is *false*.

On each activation of the *Transfer 9 - 10*, the variable to set the target position for the bricks must be set. In practice, as soon as the transfer is active the variable that indicates the number of the target position to the positioner *Pos_Num_Target* is set equal to the number of groups already placed *Moduli_Posizionati* plus 1, since at the beginning no one has been placed yet the first target position is the first.
The value of each target position is computed after the calculations of the values in ***FEEDBACK*** and depending on the value of *N_Gruppi*.

- If there is just one group to place

$$Pos\_1 = Spazio\_Necessario + (Spazio\_Libero/2) + Y$$

- If there are two groups to place

$$Pos\_1 = Larghezza\_Gruppo + Distanza\_Gruppo$$
$$Pos\_2 = Pos\_1 + Larghezza\_Gruppo + (Spazio\_Libero/2) + Y$$

- If there are N groups to place

$$Pos\_1 = Larghezza\_Gruppo + Distanza\_Gruppo$$
$$Pos\_(x+1) = Pos\_x + Larghezza\_Gruppo + Distanza\_Gruppo$$
$$Pos\_N = Pos\_(N\text{-}1) + Larghezza\_Gruppo + (Spazio\_Libero/2) + Y$$

Where Y is a constant that is the distance between the end of belts of the third diverter and the initial point of the creation of the bricks pattern. Each computed position sets the related *DB601_Belts.POSIZIONAMENTO.Pos_x.Pos_Target*.
When the transfer turns active the belts start moving. In fact, it enables the running of the belts of the third diverter with the command *Raggr_Cinghie_1_AV* at a velocity set by the variable *Vel_Cinghia_1* and starts the positioner on the other belts with the command *Raggr_Cinghie_2_Start_Pos*, that will put the groups in each of the already computed positions.
The space between each brick in a group is creating using a specific different velocities between the two belts. In fact, the velocity of the first belt is computed as

$$Vel\_Cinghia\_1 = \frac{DB700\_Params\_Inverter.U[14].Velocity\_FW}{K\_Differenza\_Vel\_Cinghie} \tag{2.1}$$

with *K_Differenza_Vel_Cinghie* = $\frac{(Larghezza\_Mattone+Distanza\_Fila)}{Larghezza\_Mattone}$.

When the positioner feedback *In_Pos_Num_Attuale* is equal to the commanded position *Pos_Num_Target*, so a group is in the correct position, the bit *Modulo_Completato* is activated and the variable *Moduli_Posizionati* is incremented by one. The transfer is then reset.

When the transfer is active again the variable *Pos_Num_Target* become equal to 2 and the second brick group is positioned. The process goes on until the number of already positioned groups *Moduli_Posizionati* is equal to the total number of groups in the pattern *N_Gruppi*, this activates the bit *Rulliera_Piena* that reset the transfer.
The transfer is reset also if the bricks have reached the end of the diverter, so the sensor at the end of the belts of the last diverter *B15_EmergCinghieDentate_2* is reading.

Ascent 4 At this point the brick pattern is ready on the belts of the last diverter and it has to be put on the roller.
The ascent of the fourth diverter is commanded when the following conditions are all verified:

- the roller and the belts of the fourth diverter are not moving
- the *Transfer 9 - 10* and the *Transfer 10 - 11* are not active;
- the belts positioner is not enabled;
- the brick pattern is ready so the related bit *Rulliera_Piena* is active or the end sensor of the belts of the last diverter *B15_EmergCinghie Dentate_2* is reading;
- the elevator is not going down and it has not already reached the high position.

When the elevator has actually reached the high position the movement is no more active.

10 - 11 At this point the brick pattern is ready on the last diverter and it has to be load on the shelf.
The conditions to active the *Transfer 10 - 11* are:

- the *Transfer 9 - 10* is not active;
- the elevator of the fourth diverter is not moving and it is in the high position;
- the belts of the fourth diverter are not moving;
- the start command of the positioner *Raggr_Cinghie_2_Start_Pos* is inactive;
- the pattern is ready so *Rulliera_Piena* is *true*;

- the loading system is at the loading floor - event marked by *Piano_Carico* - and it is free, so the sensors *B17_FineNastroCaricatore* and *B18_CaricatorePieno* are not reading;
- the *Transportation* station is not excluded - *Stazione_Esclusa* is *false*.

The transfer enables the running of the roller of the last diverter and of the loader belt, as will be explained in the next station. Thanks to these movements the bricks are loaded on the loading system.
The transfer is reset when the bricks are actually loaded and so the sensor at the end of the loader belt *B17_FineNastroCaricatore* is reading.
As soon as the transfer is not active anymore and it is verified that the bricks have reached the end of the loader belt, the number of positioned groups *Moduli_Posizionati* is set to zero and the command *Raggr_Cinghie_2_Azz* is given, resetting the position of the belts positioner. This action is performed also if the alarm related to the transfer is active.

Descent 4 When the the roller of the last diverter is empty, its elevator can return down.
The descent of the fourth diverter is commanded when the following conditions are all verified:

- the roller and the belts of the fourth diverter are not moving
- the *Transfer 9 - 10* and the *Transfer 10 - 11* are not active;
- the belts positioner is not enabled;
- the bit that marks that the brick pattern is ready *Rulliera_Piena* is inactive;
- the roller is free so its sensor *B16_UscitaRaggruppatore* is not reading;
- the elevator is not going up and it has not already reached the low position.

When the elevator has actually reached the low position the movement is no more active.

The *FC301_Transportation_OUT* just writes the command to the ascent or descent of each diverter. Depending on whether the machine is in automatic or manual mode, the output *O_YVxA_Deviatore_x_Alto* is related to the command *AUTOCMD.Dev_x_SALITA* or to the command *AUTOCMD.Dev_x_SALITA*.

2.5.2 Loader

This part of the software is used to control the loader station. Its automatic cycle is managed in a sequential mode, so the operations are divided into phases. The movements of the various parts is performed phase by phase, in this way and they can move only if the automatic cycle is active, in this way it is perfectly known which objects are moved and which are not in each phase. The use of phases allows also to stop the machine without problems, knowing exactly what it was doing and restarting from that specific point of the sequence.

Loader Data Structure - General DB 102

The structure ***INGRESSI*** is composed by some bits that represent the signals of the physical sensors:

- *Ftc_Fine_Nastro* and *Ftc_Pres_Mattoni*, the photocells that say when there are bricks respectively at the end of the conveyor-belt of the loader and on it;
- *Paletta_IND* and *Paletta_AV2*, *Trasl_IND* and *Trasl_AV*, the photocells that say when the shovel and the cart of the loader are respectively in the backward and in the forward position;
- *Fascia_Lenta_DX* to *Fascia_Lenta_SX* the sensors that say when the right and left bands of the elevator are loose;
- *Paletta_ALTA* and *Paletta_BASSA*, the photocells that say when the shovel is respectively at high and at low position;
- *Soll_Extra_ALTO*, *Soll_Extra_BASSO* and *Soll_Home*, the photocell that says when the elevator is respectively at high, low and home position;

The structure ***AUTO_CMD*** is the list of automatic commands of the station:

- *Nastro_AV* and *Nastro_IND* , *Paletta_AV* and *Paletta_IND*, *Trasl_AV* and *Trasl_IND*, that command respectively the forward and backward running of the conveyor-belt, of the shovel and of the cart of the loader;
- *Soll_SALITA* and *Soll_DISCESA*, that command respectively the ascent and the descent of the elevator;
- *Soll_Start_Pos* and *Soll_Start_Homing*, that respectively command the start to the elevator positioner and to its homing procedure;
- *Piano_Destinazione*, an integer that commands the target floor of the shelf to be reached from the elevator.

The structure ***MAN_CMD*** is the list of manual commands of the station and they are exactly equal to the ones in ***AUTO_CMD***.
The structure ***ALLARMI*** comprehends the alarms of the station. *AL01* - *AL04* are

the alarms that rise when the inverter U16 - U19 are not ready. *AL05* is the alarm that rises when the loader station is not ready.
The structure ***HMI*** comprehends the bits for the buttons in the operator panel. One is used to disable the loader station(*Svuotamento*), one to stop the elevator positioner(Stop), the others are called *Disabilita_Piano_x* and are used to disable the loading of the *x* floor of the shelf.
The structure ***APPOGGIO*** comprehends two bits as support in the logic. One is *Mattoni_Caricati*, a bit that indicates that the actual bricks pattern has been loaded; the other is *Piano_Carico*, that indicates when the floor of the shelf that was loading is full.

Loader Data Structure - Cyclic DB 202

The structure *SIGNALS* is exactly the standard one described above but the bit *Presenza_Pezzo* is not used, while *Lavoro_Completato* is used to mark when all the floor of the current shelf has been loaded.
The structure ***PARAMETRI*** comprehends the velocities of the conveyor-belt of the loader (automatic, manual and slow velocities) and an integer *Piano_Target* that indicates the floor that is loading at the moment.
The structure ***FEEDBACK*** comprehends the variable *T_Ciclo_Auto* that indicates the time of cycle of the station and some bits to indicates which phase of the cycle is active. The names of the bits used to indicate each phase, in order, are: *Piano_di_Carico*, *Attesa_Mattoni*, *Carico_Mattoni*, *Piano_Destinazione*, *Scarico_Mattoni* and *Ritorno_in_Posizione*.

Loader Logic - FCs 102 & 202

The first action done in the *FC102_Loader_GEN* is reading the input, so filling with the filtered related physical signals all the bit of the structure ***INGRESSI***.
Also the management of all the alarms(***ALLARMI***) of the station is done here.
The bit *Stazione_Esclusa* allows to disable the work of the loader station on the rising edge of a button on the operator panel (*Svuotamento*). The bit is reset on the falling front of the same button.
The bit *APPOGGIO.Piano_Carico* indicates when the elevator is actually in the loading floor. It is used just between the automatic cycle phases 5 and 10 and it is set equal to the feedback that says if the loader is in home position (*DB602_Elevator. POSIZIONAMENTO.POS_ 10.In_Pos*).
In this FC it is also considered the management of the reset of the station. The bit *Lavoro_Completato* is reset and the target floor *Piano_Destinazione* is set to zero when *RESET_Stazione* is active or at the first scan of the PLC, to initialize it.
Since each shelf has seven floor, the parameter *Piano_Target* is limited between 1 and 7. Here there is also the update of that parameters: when *Mattoni_Caricati* is true, so the current loading floor is full, *Piano_Target* is incremented by one and that bit is reset. If the option *Disabilita_Piano_x* is active, the update of the target floor is equal

to $x + 1$. When the current value of the target floor is 7 and it has been loaded, the bit *DB202_Loader_CYC.SIGNALS.Lavoro_Completato* is set and *Piano_Target* is set to 1, indicating that the current shelf is completely full of bricks.

Look now at *FC202_Loader_CYC*, its first ranks are dedicated to the signals of the station.
The station is set ready(*Stazione_Pronta*) in three cases: when the (*Step*) of the automatic cycle is zero and the shovel and the cart of the loading system are in the back position (*Paletta_IND* and *Trasl_IND* respectively); when the automatic cycle is running, so the step is different from zero; when the station is disabled, so when the bit *Stazione_Esclusa* is active.
Then there is the most important part of the logic, the one that control the automatic cycle of the station. Its phase can change only if the automatic cycle is active and there is a bit to mark each phase in the structure ***FEEDBACK***, they are active only in that specific phase and they are used in the operator panel to show what the loader and elevation system is performing in real time.
The specific phases of the loader station are:

0 - NOTHING MOVES, that is the starting point of the cycle. When the automatic cycle is active and *Lavoro_Completato* is false, so the current shelf is not completely full, the sensor *Ftc_Pres_Mattoni* is checked. If it is false, so there are no bricks on the loading system, the phase is moved to 5 to reach the loading floor, otherwise it means that the bricks are present and the phase is moved to 15, to reach the unloading floor;

5 - LOADING FLOOR, when the elevator of the loader is at the loading floor. In this phase the shovel is moved into the back position (*Paletta_IND*), the command *Piano_Destinazione* is set to 10 - the home position - and when the shovel is actually not in the forward position (*Paletta_AV2*) the command *start* is given to the elevator positioner (*Soll_Start_Pos*). In this way the elevator begins to move towards the loading floor. If the shovel is in the back position (*Paletta_IND*) and when the loading floor has been reached (*DB602_Elevator.POSIZIONAMENTO.POS_10.In_Pos*) the phase is moved to 7;

7 - WAITING FOR THE BRICKS, is the phase where the elevator is in the loading floor waiting for the bricks. After this phase there are two possibilities: if the *Transfer 10 - 11* of the transportation station - the one between the last diverter and the loader - is active, the phase is moved to 10, in which the bricks are loaded onto the cart; otherwise if the bricks have reached the end of the loader tape(*Ftc_Fine_Nastro*) the loading is finished and the phase is moved to 15, where the unloading floor is reached;

10 - LOADING BRICKS ONTO THE BELT, this is the phase where the bricks are loaded onto the belt of the cart. Until the bricks have reached the end of the belt(*Ftc_Fine_Nastro*), it is moved forward(*Nastro_AV*). If the transfer takes too long indicating that there are some problems, the alarm ***APPOGGIO***.*Alm_Trasf_10_11* is raised and the phase is moved to 7 to wait again the loading. Otherwise,

if the *Transfer 10 - 11* has finished and the end of the belt has been reached by the bricks the phase is moved to 15;

15 - DESTINATION FLOOR, in this phase the unloading floor is reached. To perform this action the command *Piano_Destinazione* is set equal to the parameter *Piano_Target* and the elevator positioner is started, moving the cart towards the unloading floor. When *Piano_Destinazione* is actually equal to *POSIZIONAMENTO.FBK.In_Pos_Num_Attuale* the target floor is reached. If also the bit *Lavoro_Completato* is not *true*, so the cycle is not finished, and if *Presenza_Pezzo*, the signal of the cart-pusher station which says that there is a free shelf ready to be loaded, is true then the phase is moved to 20;

20 - FORWARD TRANSLATION, is the first part of the bricks unloading procedure where the cart and the belt are moved forward(*Trasl_AV* and *Nastro_AV* respectively). When the cart has reached the forward position(*Trasl_AV*) the phase is moved to 25;

25 - BRICKS ADVANCE, in this phase the bricks are put on the shelf pushing them forward with the advancement of the shovel(*Paletta_AV*). When the shovel is actually in the forward position(*Paletta_AV2*) the phase is moved to 30;

30 - BACKWARD TRANSLATION, this is the last step of the unloading procedure. The cart is moved backward(*Trasl_IND*) and when it has actually reached the backward position(*Trasl_IND*) the bit that marks the end of the loading procedure *Mattoni_Caricati* is set. At this point the phase is moved to 5 and the cycle starts again.

In this FC there is also a measure of the time spent to perform the unloading procedure, so the time spent between phases 10 and 25. The value is stored in *T_Ciclo_Auto*.

2.5.3 Cart-Pusher

This part of the software is used to control the cart-pusher station. Its automatic cycle is managed in a sequential mode, so the operations are divided into phases as the loader station.

Cart-Pusher Data Structure - General DB 103

The structure ***INGRESSI*** is composed by the signals of the sensors:

- *Pos_Spintore_AV* and *Pos_Spintore_IND* two sensors that mark the position of the pusher back and forth;
- *Blocc_APERTO* and *Blocc_CHIUSO* two sensors that mark the status of the locking system, open or closed;
- *Pres_Carrello_Prelievo* and *Pres_Carrello_Deposito* two sensors that say when there are shelf respectively in the pickup and the storage position.

The structure ***AUTO_CMD*** is the list of automatic commands of the station:

- *Spintore_AV* and *Spintore_IND*, they command the pusher forwards and backwards, respectively;
- *Blocc_APERTURA* and *Blocc_CHIUSURA*, they command the opening and the closing of the locking system, respectively.

The structure ***MAN_CMD*** is the list of manual commands of the station and they are exactly equal to the ones in *AUTO_CMD*.
The structure ***ALLARMI*** comprehends the alarms of the station. ***AL01*** rises when there is the opening or closing command for the locking system but the related sensor takes too much time to become *true*. It indicates that its drive cylinder is blocked. *AL02* rises when the station is not ready but the automatic cycle is at the beginning phase, so the station needs a restart.
The structure ***HMI*** comprehends just a bit for a button in the operator panel. It is used to empty the system at the end of the day and it is called *Svuotamento*.
The structure ***APPOGGIO*** is completed empty for this station.

Cart-Pusher Data Structure - Cyclic DB 203

The structure ***SIGNALS*** is exactly the standard one described above, where the bit *Presenza_Pezzo* indicates the presence of a shelf in front of the loader and elevation system while *Lavoro_Completato* is not used.
The structure ***PARAMETRI*** comprehends only the parameter of time used to filter the sensor input signals, called *T_Check_Sensori*.
The structure ***FEEDBACK*** comprehends the variable *T_Ciclo_Auto* that indicates the time of cycle of the station, some bits to indicate the consents to proceed and some bits to indicates which phase of the cycle is active. The names of the bits used to indicate each phase, in order, are: *Attesa_Comandi*, *Apertura_Bloccaggio*, *Spintore_Avanti*,

Chiusura_Bloccaggio and *Spintore_Indietro*.

Cart-Pusher Logic - FCs 103 & 203

The *FC103_Cart-pusher_GEN* is just used to read the inputs, so filling the structure ***INGRESSI*** with the filtered related physical signals, and to manage all the alarms of the station.

Look now at the *FC203_Cart-pusher_CYC*, the first ranks are dedicated to the signals of the station. The station is ready(*Stazione_Pronta*) when the cart-pusher inverter is ready(*DB720_U20_CartPusher.FBK.Ready*).
The bit *RESET_Stazione* is used to reset the value of the *Step* of the automatic cycle and to move backward the cart-pusher if it is not in the backward position.
Then there is the most important part of the logic, the one that control the automatic cycle of the station. Its phase can change only if the automatic cycle is active(*Passi_Ciclo_ON*) and there is a bit to mark each phase in the structure ***FEEDBACK***, they are active only in that specific phase and they are used in the operator panel to show what the loader and elevation system is performing in real time.
The phases are:

0 - NOTHING MOVES, that is the starting point of the cycle. When the station is ready the sensor of the cart-pusher position *INGRESSI.Pos_Spintore_IND* is checked. If the pusher is in the backward position the cycle can go on and the phase is moved to 2, otherwise the phase is moved to 20 to reset the position of the cart-pusher;

2 - AWAITING CONSENTS, in this phase the cart-pusher waits for some consents. The events that are checked are: the presence of a free shelf in the pick-up position(*INGRESSI.Pres_Carrello_Prelievo*), the absence of a cart in the storage position(*INGRESSI.Pres_Carrello_Deposito*) and the state of the loading. The last one depends on three cases, it is true: if the work of the loader station is finished(*Lavoro_Completato*), so the shelf in front of the loader is full; if there is no shelf in front of the loader(*Presenza_Pezzo*); if the button of the operator panel that commands the emptying of the industrial line(*Svuotamento*) is pushed. When all the consents are obtained the phase is moved to 5;

5 - UNLOCKING, in this phase the locking system is unlocked using the command *AUTO_CMD.Blocc_APERTURA*. When the input *INGRESSI.Blocc_APERTO* marks that the system is actually unlocked, the presence of a shelf in front of the loader(*Presenza_Pezzo*) is reset and the phase is moved to 10;

10 - FORWARD MOVING, in this phase the locking system is still unlocked and the cart-pusher is moved forward(*AUTO_CMD.Spintore_AV*). In this way if there was a full shelf in front of the loader it is pushed in the storage position and at the same time a free shelf is moved in the loading position. When the input

INGRESSI.Pos_Spintore_AV marks that the cart-pusher is actually in the forward position, the phase is moved to 15;

15 - LOCKING, in this phase the locking system is locked using the command *AUTO_CMD.Blocc_CHIUSURA*. When the input *INGRESSI.Blocc_CHIUSO* marks that the system is actually locked, the presence of a shelf in front of the loader (*Presenza_Pezzo*) is set and the phase is moved to 20;

20 - BACKWARD MOVING, in this phase if there is a shelf in front of the loader it remains in that position thanks to the locking system. In fact, at the same time the cart-pusher is moved backward(*INGRESSI.Spintore_IND*). When the position of the cart-pusher is actually backward(*INGRESSI.Pos_Spintore_IND*) the phase is moved to 0 and the cycle restarts.

In this FC also the measure of the time spent to perform the cycle is calculated, in practice the time spent between phase 5 and phase 20. The value is stored in *T_Ciclo_Auto*.

The *FC303_Cart-pusher_OUT* just writes the physical outputs to block and to unlock the shelves. Practically *O_YV10A_BloccCarrello_Aperto* is related to the command *Blocc_APERTURA* both automatic and manual, while O_YV10C_BloccCarrello_Chiuso is related to *Blocc_CHIUSURA* again both automatic and manual.

2.5.4 Diagnostics

To keep under control the the volume of brick production and how much time it takes some logic is implemented. It counts the data as partial or total: using the command reset the partial counts are reset an restart from zero in that moment, while the total counts never reset themselves.

Diagnostic Data Structure - DB006

The *diagnostic variables* are the following ones:

- *Mattoni_Piano*, it it the number of bricks in each floor of a shelf;
- *Mattoni_parz* and *Mattoni_tot*, two integers values that count the partial and total number of loaded bricks;
- *Piani_Riempiti_parz* and *Piani_Riempiti_tot*, two integers values that count the partial and total number of loaded floors;
- *Vagoni_Riempiti_parz* and *Vagoni_Riempiti_tot*, two integers values that count the partial and total number of loaded shelves;
- a bit called *Reset_Parziali* that is used to reset all the partial counts;
- *T_Carico_Vagone*, a time variable that count the time taken for loading a shelf.

Diagnostic Logic - FC006

Consider that in a brick pattern there is a number of bricks equal to

$$Mattoni\_Piano = File\_x\_Gruppo * N\_Gruppi * Number\ of\ rows * 2, \tag{2.2}$$

where the *number of rows* is equal to 4 if the option *Spare_3* is active and equal to 5 otherwise. *Mattoni_Piano* is also the number of bricks in a floor of the shelf.
Each time that the bit that marks the loading of a floor of a shelf as finished *Mattoni_Caricati* turns active, the variables *Piani_Riempiti_parz* and *Piani_Riempiti_tot* are incremented by one, while *Mattoni_parz* and *Mattoni_tot* have both increased by the value of *Mattoni_Piano*.
The number of loaded shelf *Vagoni_Riempiti_parz* and *Vagoni_Riempiti_tot* are incremented by one each time that the bit that marks the loading of a shelf as finished *Lavoro_Completato* in the loader station turns active.

When the command *Reset_Parziali* is given, all the partial counts *Mattoni_parz*, *Piani_Riempiti_parz* and *Vagoni_Riempiti_parz* are set to zero.
When one of the total counts *Mattoni_tot*, *Piani_Riempiti_tot* or *Vagoni_Riempiti_tot* reaches the value 100'000'000, it is set to zero and it starts to count again.

A timer is set to compute *T_Carico_Vagone*. Each time that *Lavoro_Completato* in the loader station turns active, so a shelf has been loaded, the timer is reset and the elapsed time is stored in the variable *T_Carico_Vagone*. So it always indicates how long it took to load the previous shelf.

Chapter 3

HMI

Figure 3.1: Horizontal bar of the operator panel

Figure 3.2: Lateral menu of the operator panel

The HMI (Human-Machine Interface) is a very important element of the system. It is an interactive panel and it can be used to control each component of the system.
It must be intuitive to use as it is utilized by a worker who does not know all the specifications and details of the machine.
Each page of the panel is composed by the following elements:

- the the company logo in the top-left corner;
- an horizontal bar at the top of the page that reports, starting from the left:
  - the recipe currently in use;
  - the username of the currently user;
  - the state of the system that can be: Emergency, Manual, Automatic, Cycle ON and Alarm;
  - the time and the date.
- a menu in the left part of the page that reports all the sections that can be selected.

The sections are work, alarms, settings parameters, recipes and diagnostics, that are explained in the following.
When a section is active there is a green line next to the name of that section.

3.1 Work

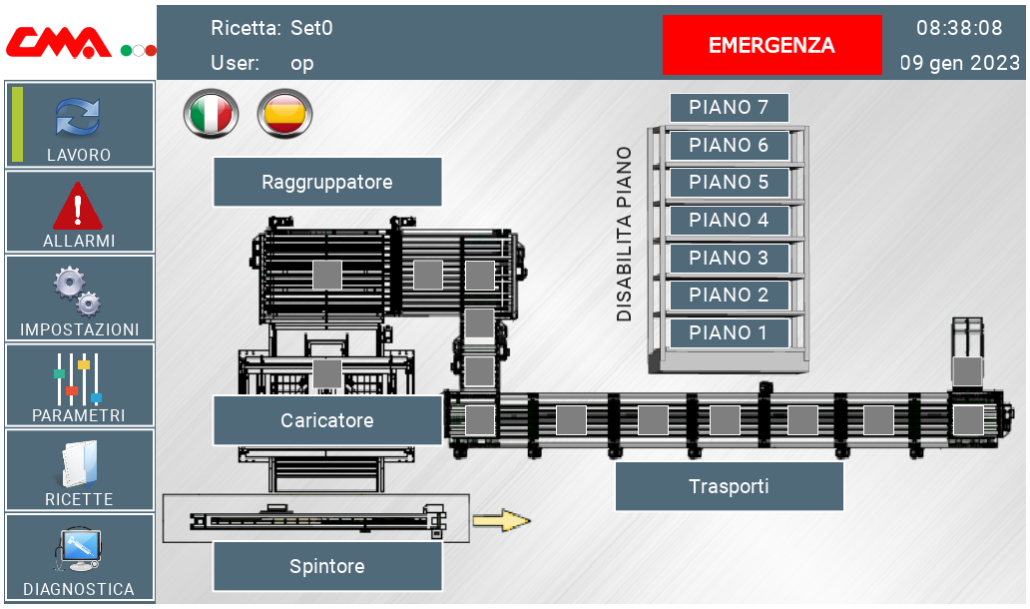


Figure 3.3: *Work* page of the operator panel

This section contains all the commands and the signals of all the sensors of the system. Its pages change as the selector on the electrical cabinet is positioned. If it selects the manual mode the work section allows to move each part and to set the velocity of each movement singularly. Otherwise, if the selector selects the automatic mode the panel shows what each station is doing and there are just few commands that can be given.
The section *Work* is the home of the panel, in its principal page there is a scheme of the system and some buttons. There are two buttons with the Italian and Spanish flags in the top left corner, they allows to change the language of the panel.
There are also some buttons to disable the loading of each floor of the shelf in the top right over the figure of a shelf. The are linked to the variables *Disabilita_Piano_x*.
Over the scheme there are four buttons that open the specific pages of each station: Transportation, Grouper, Loader and Cart-pusher. They follows the same division of the previous chapters except for little changes between Transportation and Grouper station.
There also some little grey squares that are linked to the signals of the photocells. They turn green when the related sensor is reading.

3.1.1 Transportation

Transportation - Manual

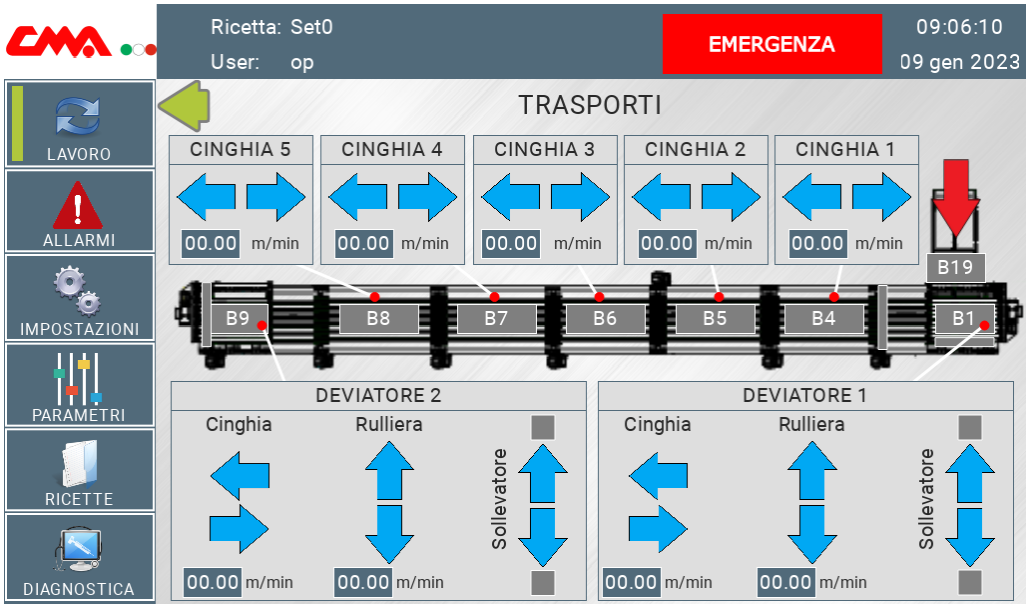


Figure 3.4: *Work* page of the *Transportation* station in manual mode

In Figure 3.4 the manual page of the *Transportation* station is reported. There is a scheme of the system starting from the entering point, highlighted by the red arrow, to the second diverter. Over it there are some rectangles that become green when the related sensor is reading. Look at *DB101_Transportation_GEN.**INGRESSI*** to see the details of each sensor.
There is a square for each mechanical part:

- in the top part of the page there are the commands of each conveyor-belt. The arrows command their forward and backward running, there are also some editable values to set their velocities.
- in the bottom part of the page there are the commands of the diverters. Both their conveyor-belts and their rollers can be moved forward and backward with the related arrows. There are also two arrows to raise or to lower the rollers of the diverters with the related limit position sensors. Also here there are some editable values to set the velocities of each movement.

The commands are linked to the ones in ***MAN_CMD*** of the specific mechanical part, while the editable velocities are linked to the parameter of manual velocity *Velocity_MAN* of the specific inverter.

Transportation - Automatic

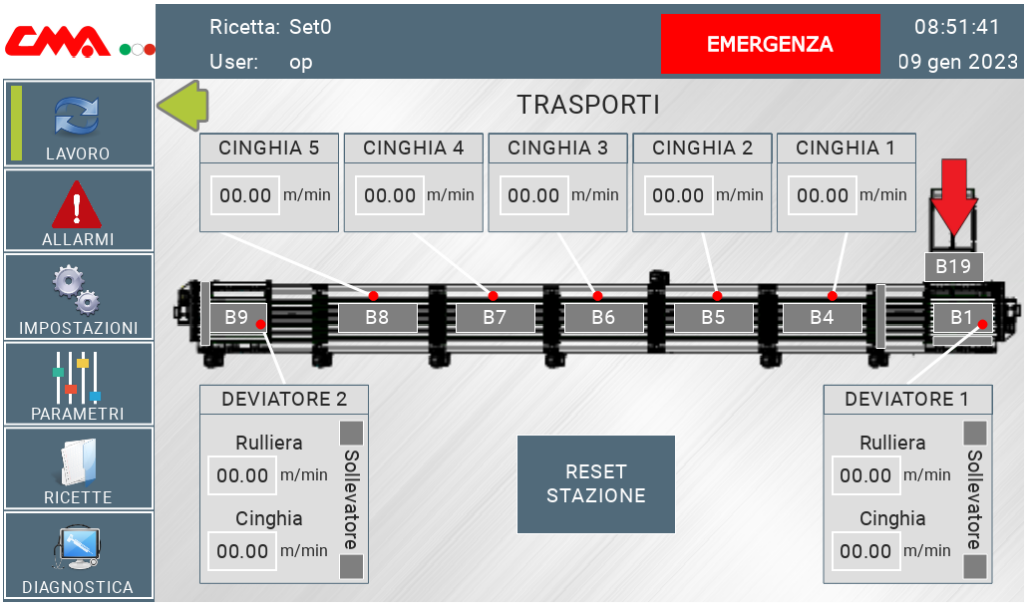


Figure 3.5: *Work* page of the *Transportation* station in automatic mode

When the automatic mode is selected the page appears as in Figure 3.5. It is pretty equal to the previous one, with just some little changes. There are no more commands of the movements, as they are automatically managed by the PLC, and there is the button with the command that reset the station, it is linked to the *RESET_Stazione* of the *Transportation* station.

3.1.2 Grouper

Grouper - Manual

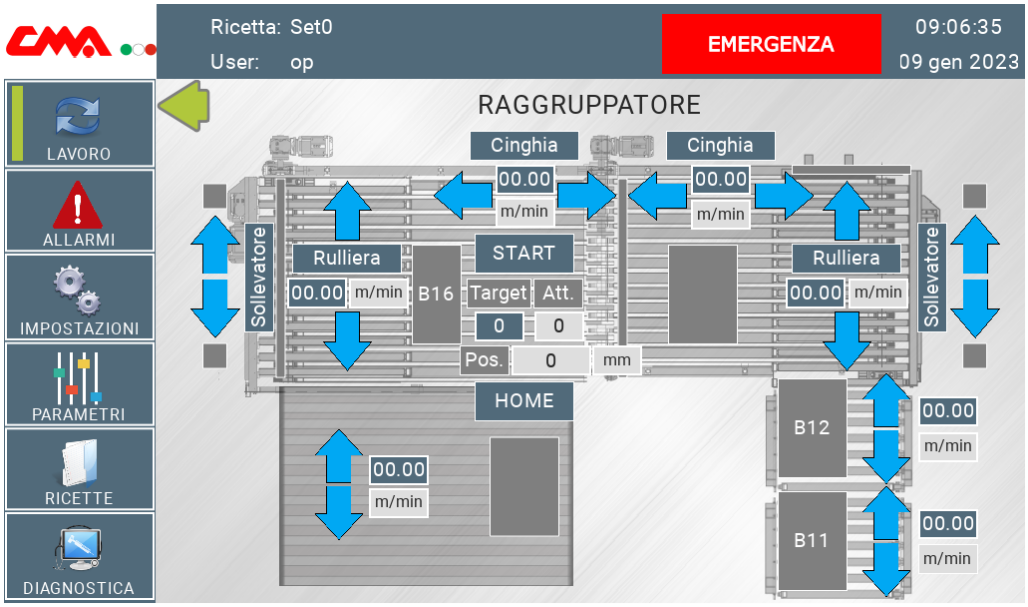


Figure 3.6: *Work* page of the *Grouper* station in manual mode

In Figure 3.6 the manual page of the *Grouper* station is reported. There is a scheme of the station that comprehends the series of two rollers, the last two diverters and the belt of the loading system. Over the image there are some rectangles that become green when the related sensor is reading. The sensors are the ones reported in ***INGRESSI***.
Over the scheme there are also the commands to move the rollers and the diverters, exactly equal as explained in the page *Transportation*.
On the second diverter there are also the commands of the positioner, they are:

- *START*, that is related to *Raggr_Cinghie_2_Start_Pos* and allows to start the logic of the belts positioner;
- two integers labelled as *"Target"* and *"Att"*, the first is editable and allow to set the number of the desired position *Pos_Num_Target*, while the other is related to *In_Pos_Num_Attuale* and just says at which position the positioner is at the moment;
- the last one is the real value *Pos_Attuale*, labelled as *"Pos"*, that is a feedback on the actual position.

Grouper - Automatic

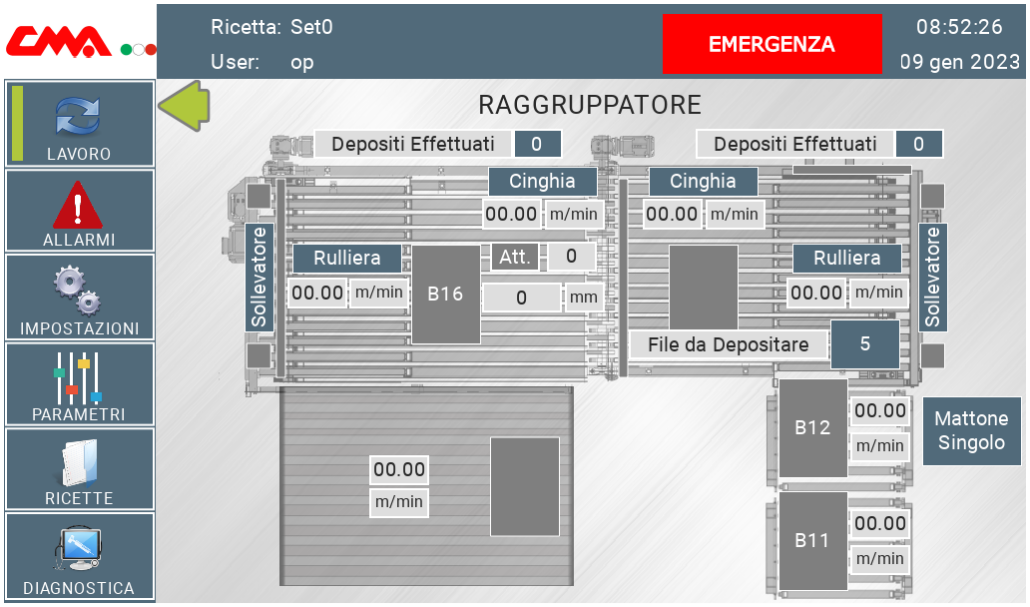


Figure 3.7: *Work* page of the *Grouper* station in automatic mode

When the automatic mode is selected the page appears as in Figure 3.7. As in the manual page the value of the label *"Att"* is related to *In_Pos_Num_Attuale* and just says the actual position of the belts positioner, while the unlabelled value is the real value *Pos_Attuale* that is a feedback on the actual position.

Unlike the manual page, there are no more commands for the movements and the velocity values are no more editable, but there are some counters for the creation of the bricks pattern.
Top left "Depositi Effetuati" is related to *Moduli_Posizionati* and it indicates the number of groups already positioned in the pattern.
Instead, top right "Depositi Effetuati" is related to *CNT_Dev_3* and it indicates the number of brick block already positioned on the roller of the third diverter.
The number next to the label "File da Depositare" is a button that selects the number of rows in the pattern. In fact, it is related to the option *Spare_3*.
The square labelled as "Mattone Singolo" is related to *Mattone_Singolo* and it turns green when it is active. It marks that there is a single row of brick at the end of the series of rollers.

3.1.3 Loader and Elevation System

Loader and Elevation System - Manual

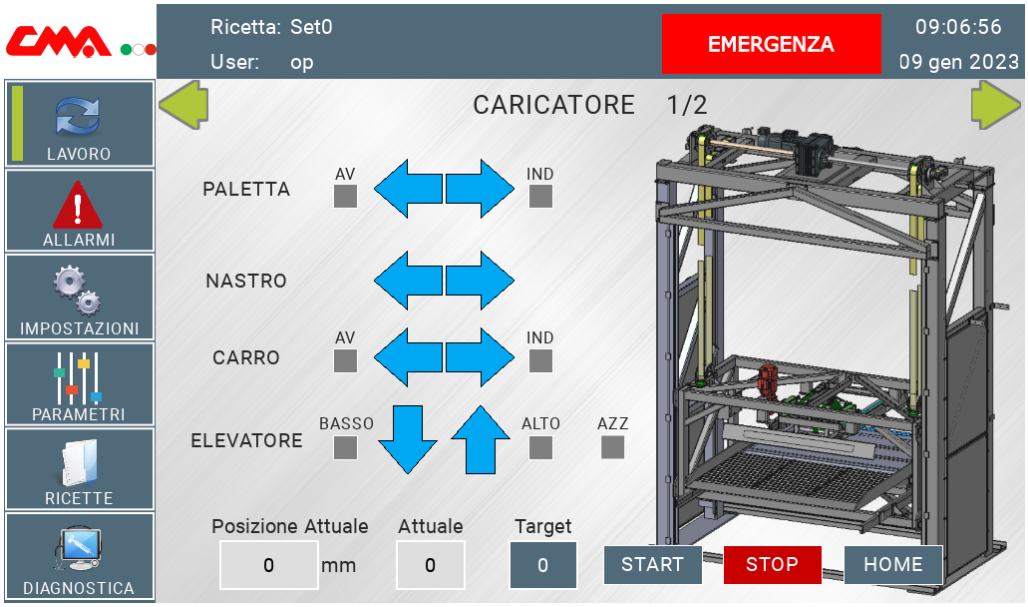


Figure 3.8: First *Work* page of the *Loader and Elevation System* station in manual mode

The *Loader and Elevation System* station is composed by two different pages both in manual and in automatic mode. In the first one, reported in Figure 3.8, there are many objects. In the right part there is an image of the loader, while in the left part there are its manual commands (***MAN_CMD***) for the forward and backward running of the shovel, of the conveyor-belt and of the cart, there is also the control of the ascent and descent of the cart. Next to each arrow there is a grey square that turns green when the limit sensor related to that movement is reading, the ones reported in ***INGRESSI***.
In the bottom left part there are the feedbacks of the positioner: *Pos_Attuale*, labelled as "Posizione Attuale" that indicates the actual position in *mm*, and *In_Pos_Num_Attuale*, labelled as "Attuale" that indicates the actual floor of the elevator.
In the bottom right part there are manual commands of the positioner: *"Target"* is related to *Piano_Destinazione* and commands the target position of the elevator; *"START"* is related to *Soll_Start_Pos* and it starts the positioning on the target position; *"START"* is related to Stop and stops the elevator; *"HOME"* is related to *Soll_Start_Homing* and starts the homing procedure.

Loader and Elevation System - Automatic

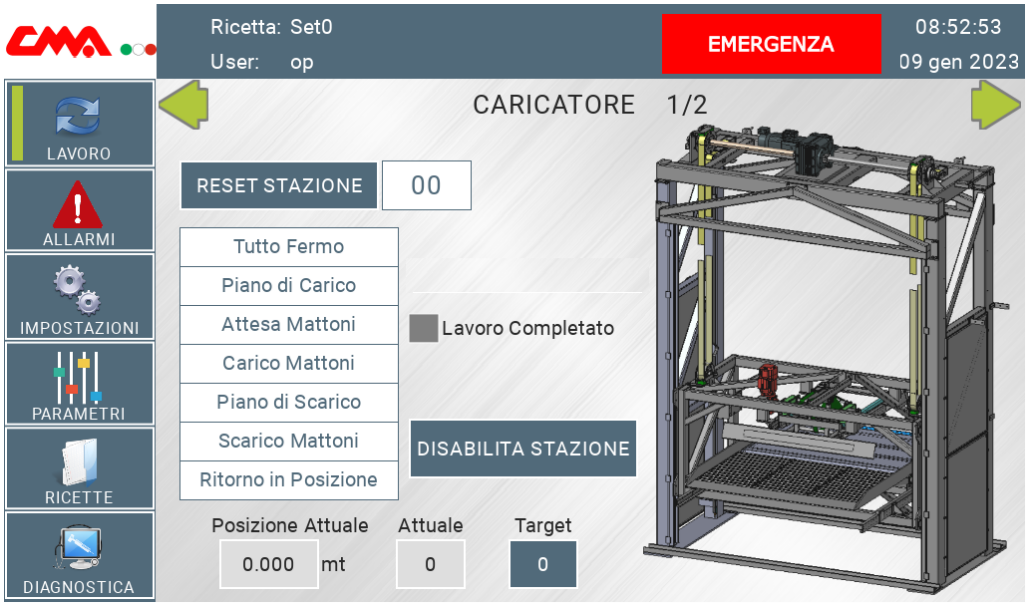


Figure 3.9: *Work* page of the *Loader and Elevation System* station in automatic mode

When the automatic mode is selected the first page appears as in Figure 3.9.
The button *"RESET STAZIONE"* is related to the command *RESET_Stazione* of the Loader station, next to it there is an integer that represents the *Step* value of the automatic cycle.
The white rectangles contain all the phases of the automatic cycle and each of them is related to one bit of the ***FEEDBACK*** structure.
There is the button *"DISABILITA STAZIONE"* that is related to *Stazione_Esclusa*, which disables the station. Over it there is a grey square labelled as *"Lavoro Completato"* that is related to the signal *Lavoro_Completato* of the *Loader* station, it turns green when all the floors of the shelf are loaded.
In the bottom part there is the visualization of the actual position labelled as "Posizione Attuale" and related to *Pos_Attuale*. Next to it there are two integers labelled as *"Attuale"* and *"Target"*, they are not editable and respectively are linked to *In_Pos_Num_Attuale*, that says at which position the positioner is at the moment, and to *Pos_Num_Target*, which visualizes the target position number.

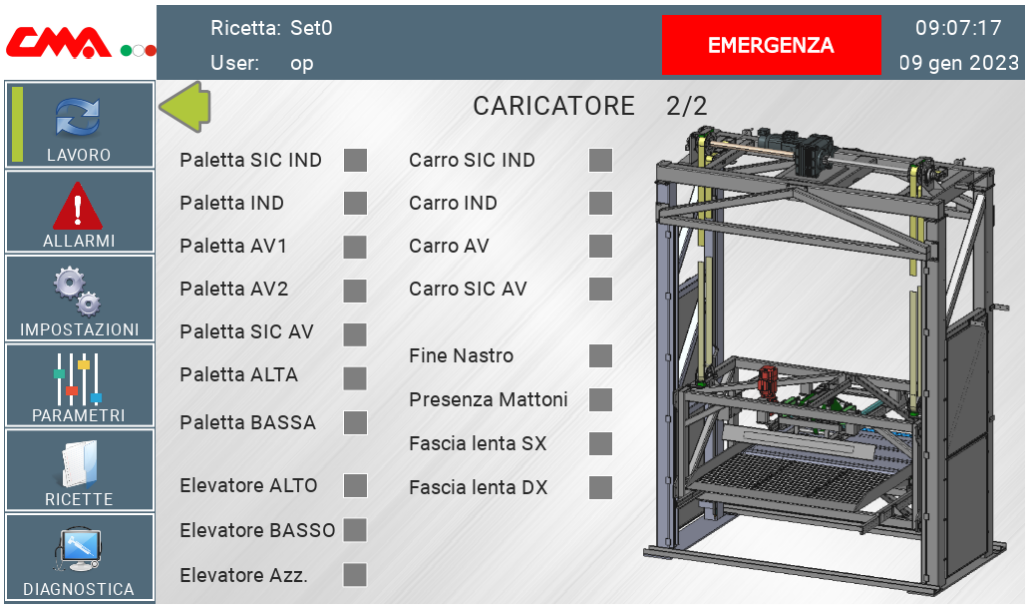


Figure 3.10: Second *Work* page of the *Loader and Elevation System* station in manual mode

The second page of the *Loader and Elevation System* station lists the signals of all the sensors of the station, the ones reported also in ***INGRESSI***. This page is the same both in manual and in automatic mode.

3.1.4 Cart-Pusher

Cart-Pusher - Manual

Figure 3.11: *Work* page of the Cart-Pusher station in manual mode

The *Cart-Pusher* manual page is reported in Figure 3.11, where a image of the Cart-Pusher is set as background. Over the picture there are the commands to push it forward and backward, they are related to *Spintore_AV* and *Spintore_IND*, and an editable value for the velocity, related to the manual velocity of the inverter U20 (*Velocity_MAN*).
Above the image there are the feedback of the sensors, related to the ones in ***INGRESSI***.
Under the image there are the commands and the sensors of the locking system. The commands are *Blocc_APERTURA* and *Blocc_CHIUSURA*.
Pushing the space in the middle of the page above the image, a picture of a shelf appears, as shown in Figure 3.12, and the bit *Presenza_Pezzo* of the Cart-Pusher station is manually set.

Cart-Pusher - Automatic

Figure 3.12: *Work* page of the Cart-Pusher station in automatic mode

When the automatic mode is selected the page appears as in Figure 3.12.
The button *"RESET STAZIONE"* in the bottom left is related to the command *RESET_Stazione* of the Cart-Pusher station, next to it there is an integer that represents the *Step* value of the automatic cycle. In the bottom right there is the button *"DISABILITA STAZIONE"* which is related to the command *Stazione_Esclusa* of the Cart-Pusher station, that disables it.
Between them there are two rectangles to show the state of the bits *Presenza_Pezzo* and *Lavoro_Completato*. The first one is not manually editable but it brings up the image of the shelf when it turns *true*, as in manual mode.
The white rectangles contain all the phases of the automatic cycle and each of them is related to one bit of ***FEEDBACK***.
The button *"PASSO DIRETTO DEL VAGONE"* is related to *Svuotamento* and commands to push forward the loaded shelf without carrying another one to be loaded. it is used to empty the system at the end of the working day.

3.2 Alarms

Figure 3.13: *Alarms* page of the operator panel

The alarms page shows all the active alarms of the system. The button "RESET ALLARMI" is used to reset the alarms, while "STORICO ALLARMI" opens the related page.

The alarms history page and shows all the alarms that the system has had. Through the drop down menu next to the label "Durata", an interval of time can be selected and only the alarms that were active in that period are shown. The button "AGGIORNA" just refresh the lists for the selected interval, while "INDIETRO" and "AVANTI" are used to navigate between the pages of the list.

Figure 3.14: *Alarms history* page of the operator panel

3.3 Settings

Figure 3.15: *Settings* page of the operator panel

The settings page is used just to do the login and the logout of the users.

3.4 Parameters

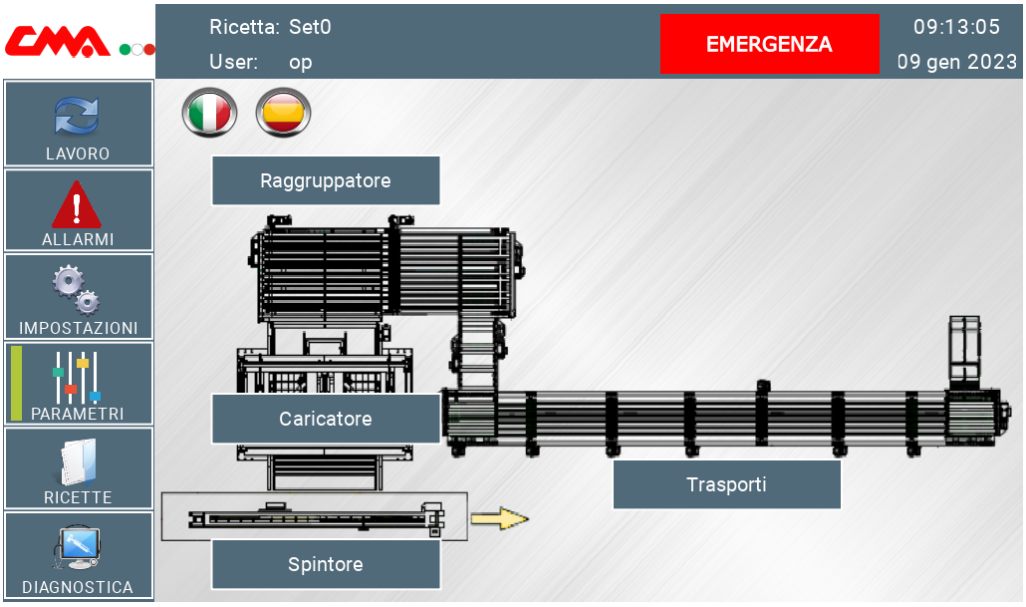


Figure 3.16: Main *Parameters* page of the operator panel

This section allows to modify all the parameters of the system that are divided by stations.
In the page there is a picture of the system, above which there are four buttons that open the specific pages of each station: *Transportation, Grouper, Loader* and *Cart-pusher*. The stations are the same as in section *Work*.

3.4.1 Transportation

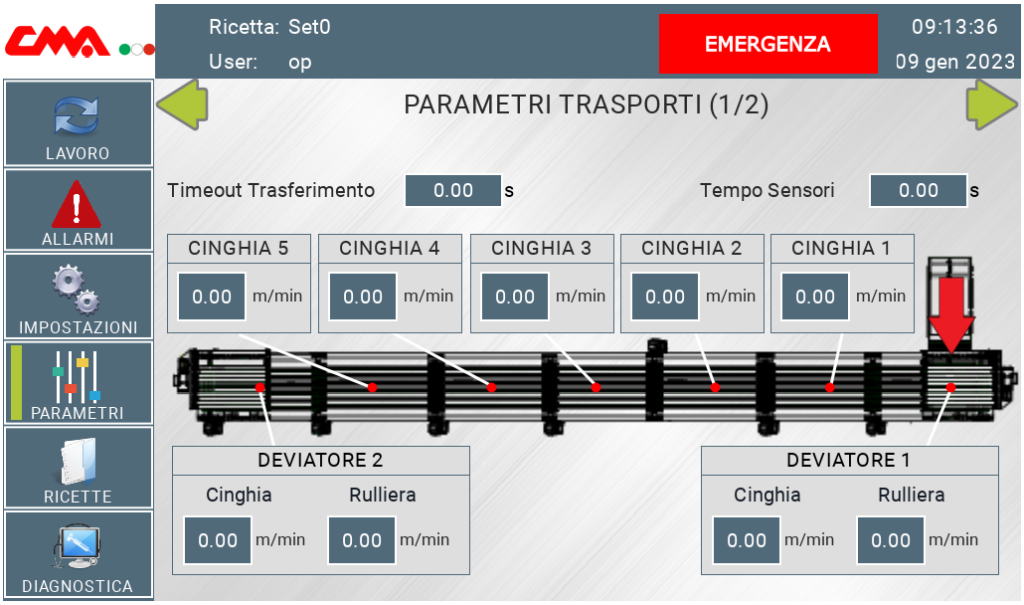


Figure 3.17: First *Parameter* page of the *Transportation* station

In Figure 3.17 the first page for the parameters of the *Transportation* station is reported. There is a scheme of the system starting from the entering point, highlight by the red arrow, to the second diverter.
Around the image there is a square for each mechanical part: in the top part of the page there are the conveyor-belts; in the bottom part there are the rollers and the conveyor-belts of the diverters. They contain the editable parameters of forward velocity, related to the variable *Velocity_FW* of the specific inverter.
In the top part of the image there are the time parameters for the timeout of the transfers and for the filtering of the sensor signals, respectively they change the values of *Timeout_Trasferimento* and *T_Check_Sensori*.

Figure 3.18: Second *Parameter* page of the *Transportation* station

In the second page, shown in Figure 3.18 are reported just three parameters. They are:

- *"Ritardo Salita Rulliera Deviatore 3"*, that is related to *RitardoSalita_RullieraDev3*;
- *"T. Passaggio Mattoni Rulliera 2"*, that is related to *T_PassaggioMattoni_Rull2*;
- *"T. Marcia Nastro di Ingresso"*, that is related to *T_Marcia_Nastro_IN*.

3.4.2 Grouper

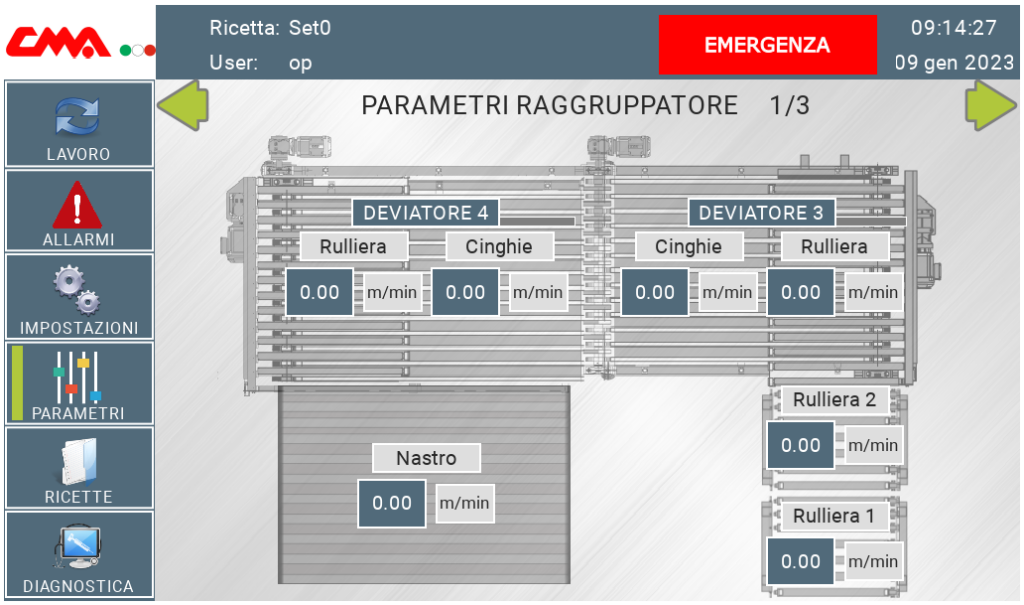


Figure 3.19: First *Parameter* page of the *Grouper* station

In Figure 3.19 the first *Parameter* page of the *Grouper* station is reported. It allows to edit the parameters of forward velocity, related to the variable *Velocity_FW* of the specific inverter: the two rollers, the last two diverters and the belt of the loading system.

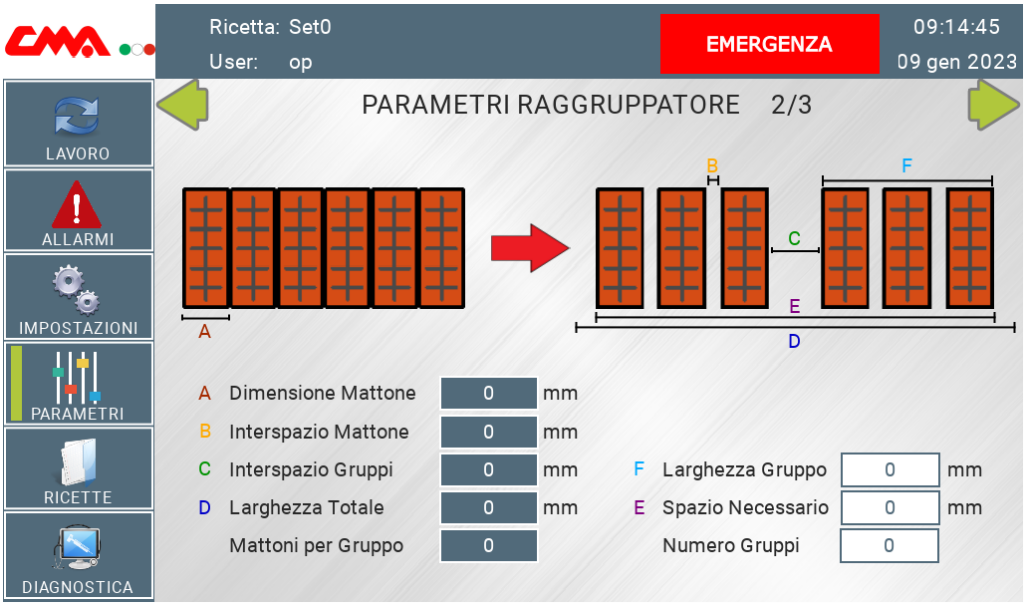


Figure 3.20: Second *Parameter* page of the *Grouper* station

The second page reports the parameters for the creation of the bricks pattern. There is an explanatory image - see Figure 3.20 - in which the meaning of the parameters is shown. The left column reports the editable values, while the right one reports the values of some quantities that are computed using the left column.
"A - Dimensione Mattone" is the value of the width of a brick and it is related to *Larghezza_Mattone*.
"B - Interspazio Mattone" is the width of the space between two bricks in a group, it is related to *Distanza_Fila*.
"C - Interspazio Gruppi" is the width of the space between two groups, it is related to *Distanza_Gruppo*.
"D - Larghezza Totale" is the value of the available space on the last diverter to create the pattern and it is related to *Larghezza_tot*.
"Mattoni per Gruppo" is the number of bricks for each group, it is related to *File_x_Gruppo*.
"F - Larghezza Gruppo" is the value of the occupied space by a group and it is related to *Larghezza_Gruppo*.
"E - Spazio Necessario" is the value of necessary space to create the pattern with the selected parameters and it is related to *Spazio_Necessario*.
"Numero Gruppi" is the calculated maximum number of groups that fits the space on the diverter with the precedent parameters and it is related to *N_Gruppi*.

Figure 3.21: Third *Parameter* page of the *Grouper* station

The third page, shown in Figure 3.21, reports the editable parameters of the positioner - *PARAMETRI* - and the target position (*Pos_Target*) and velocity (*Vel_Target*) of each position of *FC601_Belts.POSIZIONAMENTO*. The last ones are no editable and are calculated with the parameters of the precedent page.

3.4.3 Loader and Elevation System

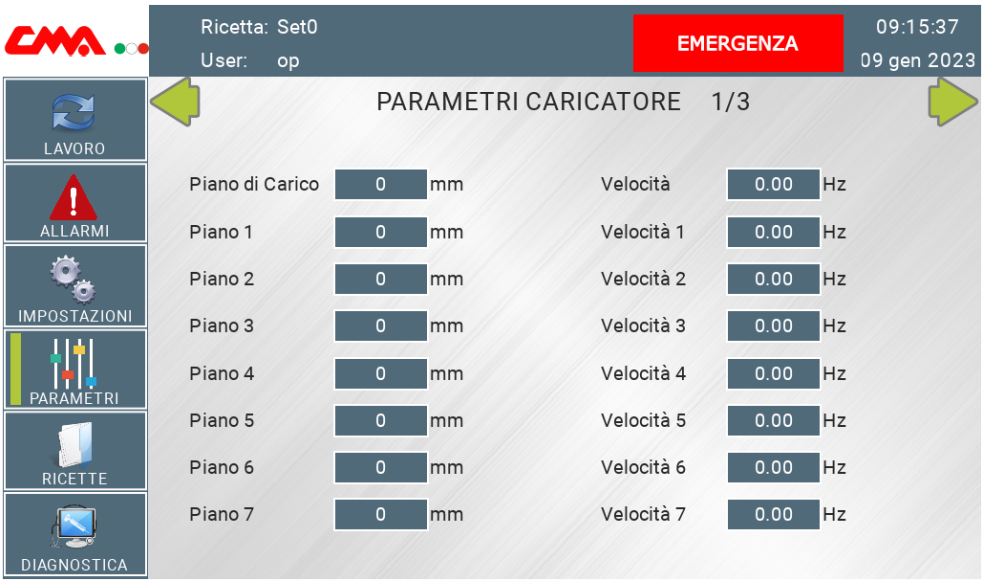


Figure 3.22: First *Parameter* page of the *Loader and Elevation System*

The first *Parameter* page of the *Loader and Elevation System*, shown in Figure 3.22, reports the editable target position (*Pos_Target*) and velocity (*Vel_Target*) of each position of *FC602_Elevator.POSIZIONAMENTO*. The first row is related to the loading floor and it is linked to the position number 10, the others to the seven floors of the shelf and they are linked to the positions 1-7.

Figure 3.23: Second *Parameter* page of the *Loader and Elevation System*

The second *Parameter* page of the *Loader and Elevation System* reports just the editable parameters of the elevator positioner. They are the ones in *DB602_Elevator. POSIZIONAMENTO.PARAMETRI*.

The last *Parameter* page of the *Loader and Elevation System* reports the editable pa-

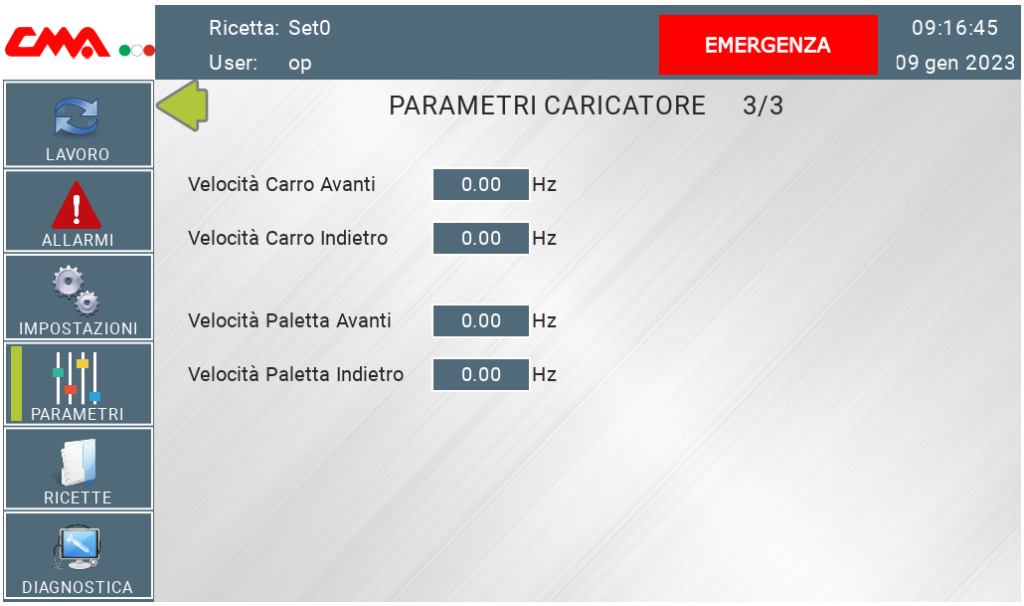


Figure 3.24: Third *Parameter* page of the *Loader and Elevation System*

rameters for the forward and backward velocity of the cart and of the shovel. They are linked to *Velocity_FW* and *Velocity_BW* of the inverters U17 and U18.

3.4.4 Cart-Pusher

Figure 3.25: *Parameter* page of the Cart-Pusher station

The *Parameter* page of the *Cart-Pusher*, shown in Figure 3.25, reports the editable parameters for the forward and backward velocity of the Cart-Pusher - *U[20].Velocity_FW* and *U[20].Velocity_BW* - and the time parameter *T_Check_Sensori* for the alarm ***AL01***.

3.5 Recipes

Figure 3.26: *Recipes* page of the operator panel

The page *Recipes* is used to manage the recipes. A recipe is a list of values of some specific variables, in this case they are all the parameters of *PARAMETRI* of both the positioners, ***PARAMETRI*** of the *Transportation* station and *T_Check_Sensori* of the *Cart-Pusher* station .
The drop down menu allows to selected a recipe between to the one that are saved, while the grey bar allow to rename its name.
Then there are some buttons.

- "NUOVA" creates a new recipe;
- "ELIMINA" removes the current one;
- "BACKUP" creates a copy of the stored recipes on a USB drive;
- "RESTORE" load the recipes of the USB drive;
- "SALVA" stores the current recipes;
- "SCARICA" download the current recipe on the PLC.

3.6 Diagnostics

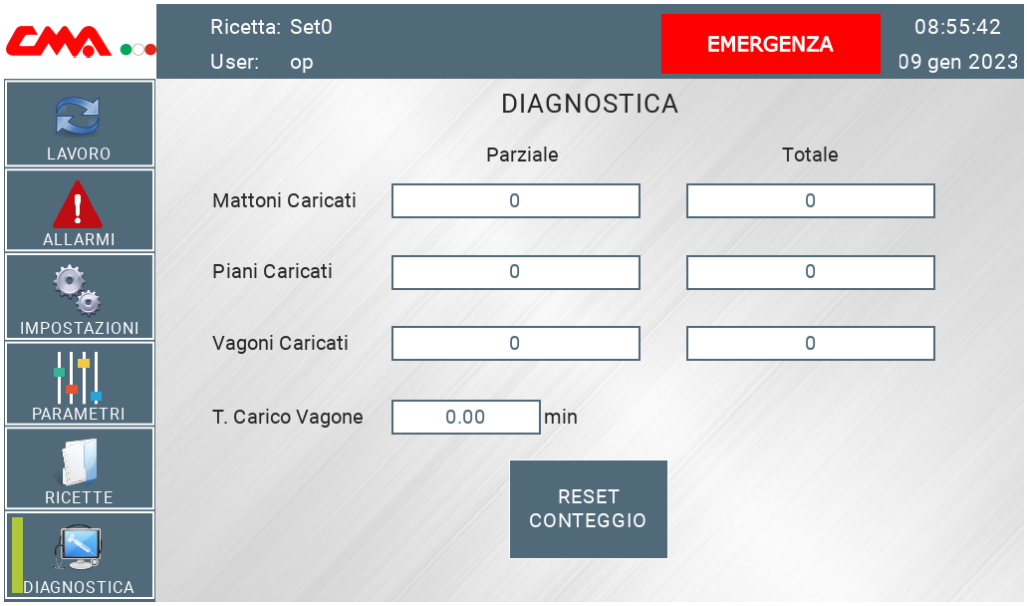


Figure 3.27: *Diagnostics* page of the operator panel

The section *Diagnostic* allows to supervise the production.
It computes the counts of the brick in various phases of the system. There are two counts for each object: one is partial ("Parziale") and one is total ("Totale"). The partial starts from zero each time that the button "RESET CONTEGGIO" is pushed.
"Mattoni Caricati" is the value of bricks already loaded.
"Piani Caricati" is the value of floors already loaded.
"Vagoni Caricati" is the value of shelves already loaded.
"T. Carico Vagone" is the value of elapsed time during the last loading of a shelf.
All the values are linked to the *diagnostic variables* of *DB006*.

Conclusions

The system is created for a factory based in Lambayeque, Peru. The mechanical design and construction of the machine was done by CMA s.r.l. [1], while the electrical design and realization was made by Innova s.r.l. [2].

When the software was finished, the Innova team went to Peru for make it works. As first the mechanical part was checked and then the electrical connection was made. At this point the system was tested and activated.
During this phase some problems arose. As first some mechanical measurements were not correct, for example there were too much space between the rollers of the series. In fact, the bricks were falling between one to the other. To resolve the problem a single reel has been added to the ones of the second roller.
The management of the entering roller was not meant to be managed originally, but then it was decided to control it. Practically O_Richiesta_Prodotto does not rise a signal but directly control that roller. It is interfaced with the inverter of the previous machine and command just the forward running at a fixed velocity.

A big change in the software was made since at the beginning it was designed to manage every brick pass as a transfer, while now most of them are controlled using timers. This choice was requested by the factory to save time during the transfers.

Moreover the system was created to work for different type of bricks, but at the moment it has been tested with just one. Its production is going on without problems. In the future, when the system will be used for other types of bricks, it will need to change the recipe and the related parameters will have to be set in a test phase before going into actual production. The PLC software does not need to be modified, the parameters will be changed through the HMI. This is the strength of having made the software in this way, once that the system actually works no software engineer needs to go there to do anything.

Bibliography

[1] CMA Impianti. Cma impianti. https://www.cmaimpianti.com/, 2023. [Online; accessed 15-January-2023].

[2] Innova. Innova - the heart of automation. https://www.innova.sm/, 2023. [Online; accessed 12-January-2023].

[3] PI North America. Profinet vs ethernet: Definitions and a comparison. https://us.profinet.com/profinet-vs-ethernet-definitions-and-a-comparison/, 2023. [Online; accessed 5-January-2023].