POLITECNICO DI TORINO

Collegio di ingegneria Informatica, del Cinema e Meccatronica
Master degree in Mechatronic Engineering

# Communication interface between various drives and programmable logic controllers.

Supervisor:
Prof. Eng. Luigi Mazza
Company tutor:
Mr Alessandro Monge

Candidate:
Sebastiano De Luca

April 2019

# SUMMARY

Nowadays, the market pushes companies to make their production flexible and to develop their productivity in order to be competitive. These conditions have led companies in the automation, initially aiming at Industry 3.0 and now at Industry 4.0. The programmable logic controllers (PLCs) are playing a key role in this industrial evolution. The PLC allows a complete control of industrial plants because it is able to interface with all the devices of the factory. Specifically, the control of the electric motors is carried out by means of drive. There are two types of drive: inverter drive and servo drive. Inverter drive controls the speed of an electric motor, while a servo drive adjusts the position of an electric motor. These devices interface with programmable logic controllers through data structures. Therefore, communication is a fundamental aspect for the execution of an electric motor control. Data communication structures depend on the type of drive and the manufacturer of the drive. Particularly, Siemens communication structures are called telegrams, while the Rockwell Automation ones are called data type. In addition, Siemens company provides pre-set blocks to programmers. These blocks facilitate the implementation of the control but they restrict the programmer and user manipulation.

The thesis objective is to construct an interface that eliminates these constraints. The interface block aims to allow the communication between central processing unit (CPU) of PLC and drive regardless of the type of drive and the manufacturer of drive.

To implement the interface block, the Siemens and Rockwell Automation communication structures have been chosen, because they are very different from each other. Generally, other producers base their communication structures on Siemens telegrams and Rockwell Automation data types.

Initially, a thorough analysis of the communication modes between central processing unit of PLC and drive has been performed both in the case of Siemens systems and in the case of Rockwell Automation systems. Next, the interface block has been implemented, and finally a series of tests have been executed to verify the block operation. The interface block has been tested by using firstly a Siemens PLC and a Siemens drive inverter, secondly a Siemens PLC and a Siemens servo drive and finally a Rockwell PLC and a Rockwell inverter.

The test results have been positive: all the features that were required for the block have been executed and the communication between central processing unit of PLC and drive has been successful.

A further development of this thesis could be the construction of a converter that allows a direct conversion from a software environment to another. In this way, it would be possible not only a communication between PLC and drive regardless of the drive type and the drive manufacturer, but also a free implementation of the program regardless of the programming software chosen.

# INDEX

# INTRODUCTION

The main objective of this experimental thesis is the complete and detailed analysis of the programmable logic controller system. Specifically, a communication interface between Programmable Logic Controller (PLC) and drive will be implemented and tested.

My interest in programmable logic controllers is based on the high use of these devices in the industrial environment. Indeed, the demand for better products has stimulated a new industrial development. This industrial evolution registers a shift from the Industry 3.0 to the Industry 4.0. The programmable logic controllers are the key players of this evolution because they have allowed the birth of the Industry 3.0 and now the development towards the Industry 4.0.

The programmable logic controller is a digitally operating electronic system, designed for the industrial environment, which uses programmable memory for implementing specific functions.

The main advantages of PLCs are flexibility, cost, smoothness, modularity, size, safety and durability. These characteristics make them suitable to interface with all the industrial devices and to carry out a complete control of an industrial plant. These characteristics have led to a wide diffusion of PLC in the industrial field.

The study has been divided into three parts: the first chapter shows the state of the art of programmable logic controllers. This kind of analysis has been done to understand the various components that form a system based on PLC and the potentialities of PLCs. Initially PLC definition is reported and then there is a wide and detailed excursus about the PLC characteristics and how PLCs have impacted on today's industrial evolution.

In the second chapter, the communication between central processing unit of PLC and drive will be analyzed on the basis of two main PLC manufacturers and program implemented by me. The control of electric motors by means of PLC is done through servo drives and inverter drives. The communication between PLC and drive is the main feature for control to be properly effectuated. Each company has its own and different communication structure from the other structures on the market. Specifically, there is an analysis of the communication structures of Siemens and Rockwell Automation. The communication structures are called telegrams by Siemens and data type by Rockwell Automation. The telegrams are different from the data type even if both allow the communication between PLC and drive. Based on the analysis of the various communication structures, there is the implementation of a communication interface.

The interface block allows communication between PLC and drive regardless of the drive type and the drive manufacturer.

In the last chapter, the test results on the communication interface are reported. The test results are divided in according to the communication structure tested. Initially, the test results on communication telegrams between Siemens devices are analyzed in detail. Particularly, the Standard Telegram 111 and its modalities are examined deeply and the analysis of the two telegrams for the management of an inverter (Standard Telegram 1 and Siemens Telegram 352) is executed. Finally, the test results about the data types of communication between Rockwell Automation devices are fully described. In this case, the test results concerning the “PowerFlex 525” data type are analyzed.

At the bottom of the thesis there are the appendices with the Siemens and Rockwell Automation communication structures and the interface block implemented in the Siemens and the Rockwell Automation programming software.

# 1. The Programmable Logic Controller

## INTRODUCTION

In this first chapter, it will be shown the state of the art of Programmable Logic Controllers (PLCs). Particularly in the first paragraph, the PLC definition will be reported according to IEC 61131 and main differences between PLC and PC will be analyzed. The second paragraph analyzes the PLC application field by defining the reasons why PLC control system is used. In addition, the paragraph will examine differences between Industry 3.0 and Industry 4.0 and why PLC are fundamental to industrial evolution.

The advantages of the PLC will be listed and analyzed individually in paragraph 1.3.

In "Process control by means of PLC" paragraph, observation and controllability of control process will be described. Moreover, the paragraph will list some of analog and digital devices that can be connected to PLCs.

In paragraph 1.5, PLC architecture will be examined. Specifically, all main devices of PLC system will be listed and then each device will be analyzed and studied in detail. This paragraph is divided into 7 sub-paragraphs because each sub-paragraph will focus on PLC system device.

Paragraph 1.6 is called "Basic PLC operation" because it will talk about the steps that PLC performs to carry out the control.

In paragraph 1.7, the PLC cycle concept and the PLC cycle types will be analyzed, describing cycle algorithms activates.

Successively, paragraph 1.8 will explain the addressing concept, focusing on how the programmer should perform the address of each logical signal.

In paragraph 1.9, programming languages will be listed and examined. In this paragraph, some functions examples will be reported and they will be represented according to different programming languages.

In paragraph 1.10, there will be shown set of operations and data type that the CPUs of PLCs can calculate. The paragraph 1.10 constitutes of 4 sub-paragraphs: each sub-paragraph will study and explain set of operations thought application examples.

Next paragraph will analyze how PLCs work and specifically how they are implemented by programmers.

In paragraph 1.12, an analysis of the communication cable among various devices of a PLC will be described, focusing on “Profinet” network.

Finally, the last paragraph will describe IEC 61131 norm. All companies that produce PLC and programmers must respect this norm so that their control devices system is defined programmable logic controllers.

## *1.1 Programmable Logic Controller definition and difference between PLC and PC*

Nowadays the world economic market pushes companies to greater flexibility and product improvement. Considering these needs, many companies use programmable logic controllers in production control. The Programmable Logic Controller or PLC is "a digitally operating electronic system, designed for use the industrial environment, which uses programmable memory for implementing specific functions such as logic, sequencing, timing, counting and arithmetic, to control, through digital or analog inputs and outputs, various types of machines or processes " [Document No. 1 bibliography].

Programmable Logic Controller (PLC) is a device capable of controlling industrial process. It is a particular industrial control system, because it guides processes in difficult conditions that are characterized by electrical disturbances, vibrations, dust and wide variations in temperature and humidity.

Controlled processes can be different types, like plant for printing newspapers or filling plant or press for plastic molds.

PLC is a controller that is based on logic programming. Actually, PLCs perform not only strictly logical functions such as AND, OR, NOT and NAND, but they are able to perform count, timing, comparison, numerical calculation, and several operations of data manipulation in bit, byte, word, int and Dint format, like CONV, FILL, SHIFT functions.

PLC can be identified as a personal computer with limited potential. As the table 1.1 shows, there are several differences between PC and PLC [Document No. 5 bibliography].

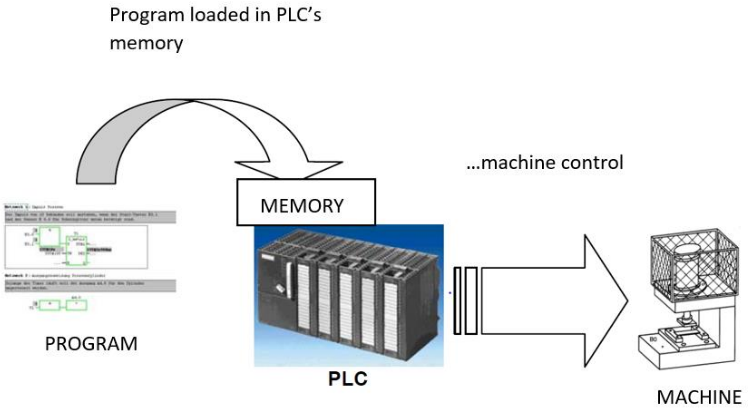
| Features | PC | PLC |
|---|---|---|
| Data movement | >500.00kByte/sec | > 10 kByte/s |
| Size of programs | >10.000 kByte | < 10 kByte |
| Binary operations | movement of 32 bit | operations of 1 bit |
| Microprocessor frequency | >1 GHz | <100MHz |

| | | |
|---|---|---|
| Typical operation | 8 hours per day | 24 hours on 24 |
| Immunity to electrical disturbances | poor | High |
| Environmental conditions | air-conditioned interior | From 0 to 55 °C |
| Programming | compiled in high-level languages | direct, in "machine language" |

(Table 1.1 Differences between PLC and PC)

Focusing on the last feature, programming language is a significant difference between PC and PLC. Contrary to the PC, the PLC language does not have to be particularly complex, so that it can be used by technicians who do not have knowledge oriented to electronics and computer science.

In summary, PLC is able to control industrial production in detail and in difficult environmental conditions.

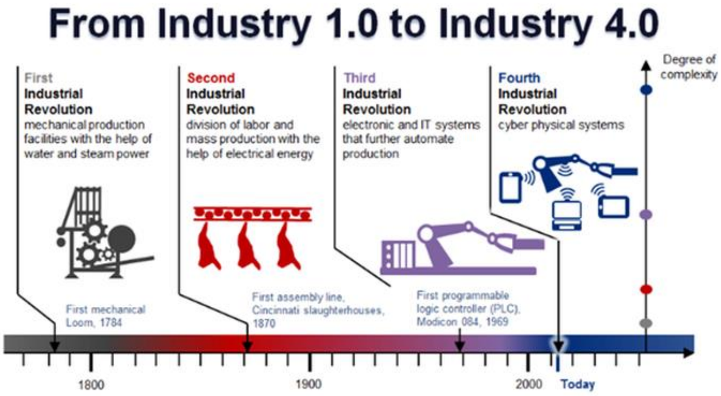


(Fig. 1.1 Structure of control system)

## *1.2 Applications field*

Application fields of PLCs range from construction to vibration control to industrial control for production. Indeed, in order to be competitive, companies have to provide customers with better and constant quality products, they have to make production flexible and to develop productivity. This situation has pushed companies towards automation aiming at Industry 3.0 and at Industry 4.0. In this climate of industrial improvement, PLC has assumed primary role in production, transformation and control sectors. Increase of the PLC use has led to their technological improvement, sure enough they are not simple sequencers with the purpose of controlling production process but they are capable to make self-diagnosis of the possible failures.

Initially, PLCs were implemented in the context of the industry 3.0 (fig. 1.2).

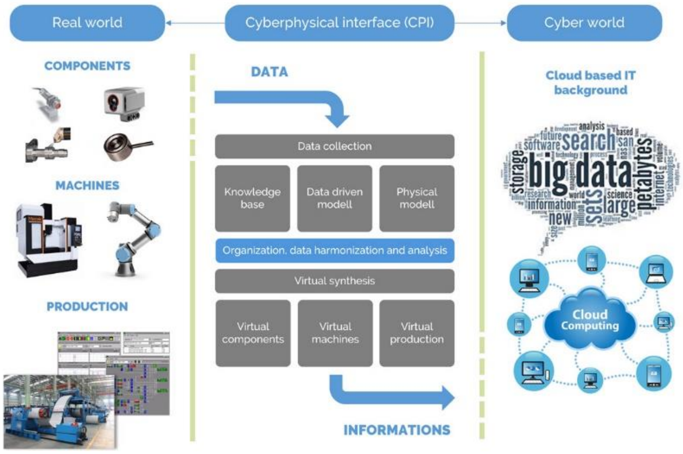

From Industry 1.0 to Industry 4.0
First Industrial Revolution
mechanical production facilities with the help of water and steam power
Second Industrial Revolution
division of labor and mass production with the help of electrical energy
Third Industrial Revolution
electronic and IT systems that further automate production
Fourth Industrial Revolution
cyber physical systems
Degree of complexity
First mechanical Loom, 1784
First assembly line, Cincinnati slaughterhouses, 1870
First programmable logic controller (PLC), Modicon 084, 1969
1800
1900
2000
Today

(Fig. 1.2 Evolution of industry)

The industry 3.0 is characterized by hierarchical approach that is applied to the automation of production processes. This approach is represented by so-called "automation pyramid" and it is called vertical integration.

The automation pyramid consists of following levels: physical production processes, supervision and data acquisition control (SCADA), production control and business processes related to production planning and resource planning [Document No. 4 bibliography].

Evolution of Industry 3.0 is the Industry 4.0, that is based on RAMI 4.0 model (fig 1.3).



(Fig. 1.3 Structure of industry 4.0)

This structure integrates product life cycle and system architecture. System architecture is defined as physical production processes and Company's production.

If factories are based on this model, they are considered like cyber-physical production systems. In these systems components of factories (machines, various equipment and products) communicate directly to exchange information, perform operations and control each other autonomously.

This type of control system needs different objects to communicate reciprocally. In this regard Digital Administration Shell was introduced.

The introduction of Digital Administration Shell was necessary to facilitate the integration of the components. It involves insertion of physical component able to interface all components of the factory allowing a standardized communication for the storage of all data and acquired information.

The creation of data area in which to store and exchange data within a company or group of companies, even in different branches, has generated in a simple and safe way a new type of integration: the horizontal integration. In other words, with horizontal integration we mean the possibility to communicate in real time safety analysis not only all the components of a factory among themselves, but also company with other companies that use data area accessible for all users. Thus, if Industry 3.0 was characterized only by vertical integration, Industry 4.0 is characterized by both vertical integration and horizontal integration.

In the Industry 4.0 perspective, PLCs are perfect components as they enable state observability, control activity and communication activity with external devices.

In conclusion, the scope of PLC use is industrial environment and in the specific Industry 3.0 and 4.0.

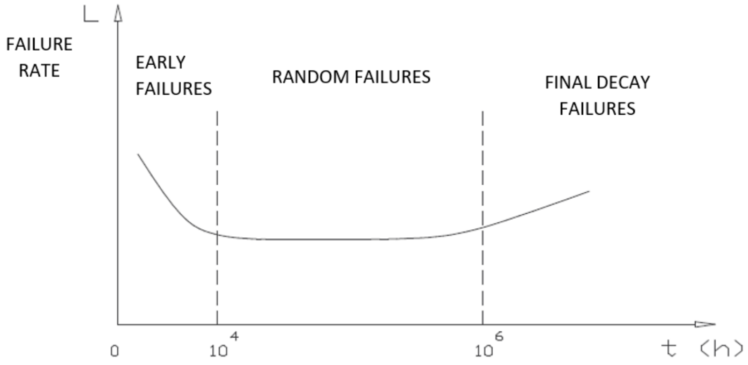
## *1.3 Advantages of PLC*

The use of PLCs is very convenient for companies, because of their flexibility, cost, smoothness, modularity, size, safety and durability [Document No. 1 bibliography]. Before the advent of PLCs, the electromechanical and pneumatic components were used to perform sequential and control activities. They had to be wired to perform a certain function, and the entire wiring had to be re-run whenever a different extension or function was needed. The use of PLC has eliminated this phase, sure enough, in case of expansion, new equipment is added and connect physically without having to modify the connections of the other components; while in case the automation cycle needs to be changed, the program instructions need to be changed simply by leaving the physical connections unchanged. In other words, they are extremely flexible, because the process can be changed very quickly. PLCs are economical:

- if plant must to be eliminated, it can be reused in another application;
- if the pneumatic or electromechanical circuit is obsolete, it can be replaced with PLC, as the PLC is correctly adapted to the existing machines.

Moreover, the PLCs are compact and their structure is modular, so you can configure them according to installation needs. They are able to perform a self-diagnosis and interface with computers. This permit application of the Industry 4.0. Another important advantage of PLCs is their small size, which makes it possible to test and refine program. In addition, program test can be performed by means of suitable simulator software. In this case, programmer chooses type of wanted PLC, program is loaded in the virtual PLC and then the programmer starts simulation. Another advantage of PLC is safety. Particularly fatal accidents cannot happen because in most cases there are voltages up to 30 V.

The cost of management is a key factor for PLC, they also have minimal wear and they require little maintenance (fig 1.4).



(Fig. 1.4 Falure rate trend)

So, benefits of PLCs increase their use in industrial field, improving the control systems.

## *1.4 Process control by means of PLC*

A control system is a system that allows to observe and control a plant. Specifically, the observability is acquisition of process states in real time, while controllability is the ability of PLCs to obtain states that user wants.

States observation is performed by means of input pins. They are connected to all devices that can execute function of signal generators like switches, buttons, sensors, limit switches, photocells, thermostat and thermometer.

These devices can be classified into analog and digital devices:

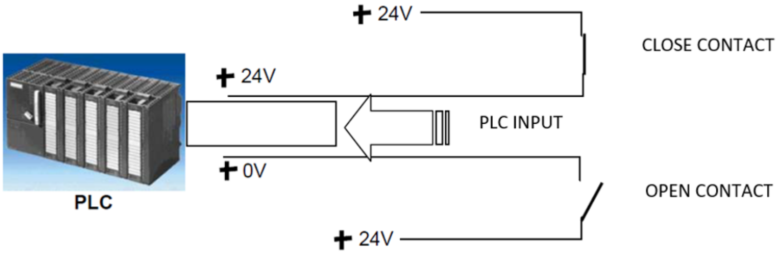
-analog device can take values in a range;

-digital device can take two values ON/OFF.

Switches, buttons, photocell, limit switches, thermostat are digital devices and they produce digital inputs. Specifically, there are two types of contacts: normally open contacts and normally closed contacts. Normally open contacts close when they are actuated; conversely, normally closed contacts open when they are actuated. On the other hand, photocell produces an open signal if it is reached by light, otherwise it produces a closed signal.

Operation of limit switch is the button that is actuated by mechanical action and not by user.

Thermometer instead is an analog device. It is different from thermostat because thermometer measures values of temperature of environment, while thermostat indicates if temperature is above or below the reference value.



(Fig. 1.5 Scheme of process states)

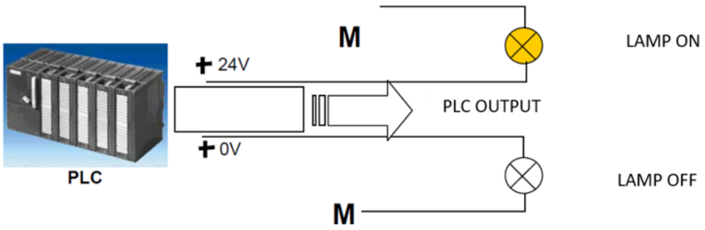
The PLC acquires various states from the signal generators and then it executes instructions of the program stored in its memory. This generates all tasks required to get states that the user wanted.

At this point observability phase ends and the controllability function is performed through the application of the activities determined by the program processing.

System control is performed through specific devices that are able to execute the functions necessary for the process to run properly. These devices are connected to PLC outputs that can be digital or analogue.

PLC commands actuators like motors, solenoid valves and other circuits, via digital outputs, while the analogue outputs are used to operate proportional valves, indicators, recorders, drives and inverters.

In other words, activities are performed by means of starting or stopping an engine, opening or closing a valve, or turning the indicator light on or off (figure 1.6).



(Fig. 1.6 Scheme of process control)

In conclusion, PLC must be able to observe current system states and apply commands to obtain desired states.

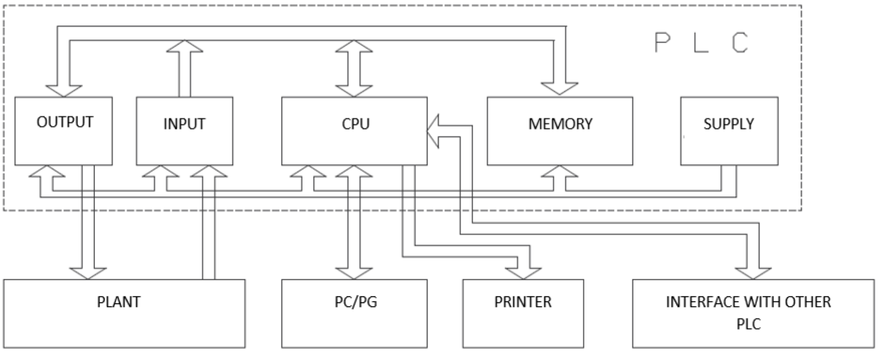
## *1.5 PLC architecture*

PLC-based control system is characterized by a complex architecture. Basically, PLC first acquires all the process states in real time, then it calculates the tasks necessary to implement any changes to be made to current state, and finally it applies the evaluated operations based on the instructions written in program.

PLC consists of the following elements (figure 1.7):

- input modules;
- output modules;
- central processing unit (CPU);
- memory;
- power supply;
- bus system.

In addition, PLC is equipped to connect with peripherals that let human-machine interfaces (HMI), communication with other PLCs, use of mass storage, connection with online servers and paper documentation. These devices collaborate for the execution of an effective control system [Document No. 1 bibliography].



(Fig. 1.7 Scheme of PLC architecture)

### 1.5.1 Input and output modules

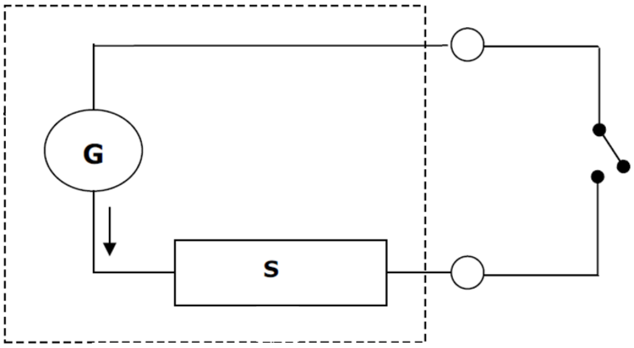
Input and output modules can be defined as interface between PLC and external environment. The input modules are used to connect signal generators to PLC; the output modules, instead, connect PLC with actuators allowing application of functions elaborated by the PLC.

The number of input and output modules is a crucial feature for PLC, because the greater the number of input and output modules the greater the number of sensors and actuators that can be connected. This increases the PLC control capacity and its cost.

Input signals must be converted into signals that can be recognized by the control unit because they are electrical signals and the CPU is unable to read electrical signals.

The conversion of these signals is performed by input boards, which are interfaces that allow communication between PLC and power unit (servo drive, inverter, etc.).

The digital input board unit with internal generator is schematized in the following image (figure 1.8):



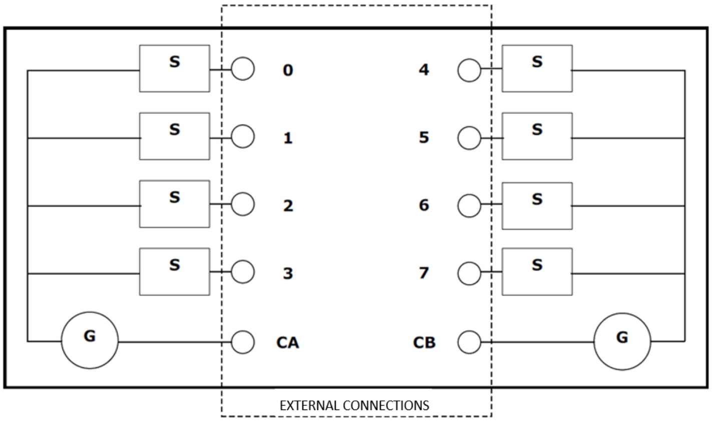
(Fig. 1.8 Unit of digital input board)

Digital input unit of figure 1.9 consists of an internal generator (G) and a sensor (s) or an external switch (s).

An internal generator is required when a passive element is attached, conversely when board has no internal generator, connected device is active because it must have external generator.

Sensor reveals the presence or absence of the current, while the external switch source reacts the sensor state.

Digital input board is divided into groups and each group has more units (figure 1.9).



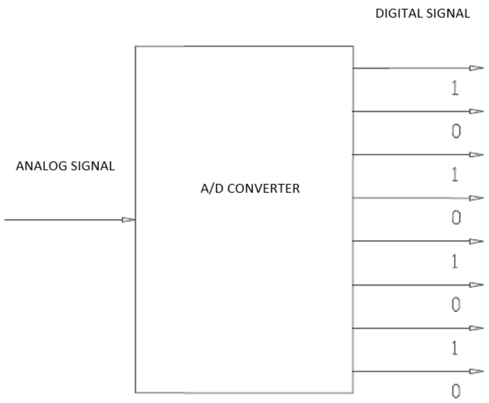
(Fig. 1.9 Digital input board with 2 groups and 4 units)

The input modules signals can be of binary type, i.e. they assume logical value 0 and logical value 1 regardless of the type of signal that could be both digital (buttons, switches) and analog (thermometer, pressure gauge).

In some cases, sure enough, the analogue signal must be treated as a digital binary signal, for example to indicate the absence of voltage or the presence of voltage.

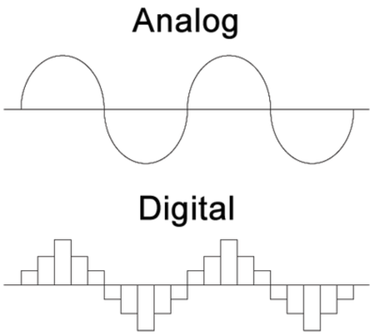
In other cases, the analogue signal must be treated as signal that varies over time within predetermined range of values, such as when the liquid level in the tank or the oven temperature is to be indicated. In these cases, the values of these signals are between minimum value and maximum value. Generally, they are standard: ± 50 mV, ± 1 V, ± 5 V, ± 10 V, 0…10 V, 0…20 mA, ± 20 mA, + 4…20 mA.

The CPU is not able to read the analogue signal, so analog signal must be converted to digital signal. This conversion is performed by A/D Converter (Fig 1.10)



(Fig. 1.10 A/D Converter)

The A / D converter transforms a continuous signal over time into a discrete one i.e. digital signal (figure 1.11).



(Fig. 1.11 Analog and digital waves)

The main characteristics of A/D converter are accuracy and precision, which depends on the resolution and on the number of bits. Indeed, the converter's resolution is:

$$\frac{T_{max} - T_{min}}{2^n}$$

where "n" is the number of bits.

Generally, number of bits of A/D converter are 8, 10 and 12, so the number of possible combinations are: 2 ^ 8 = 256; 2 ^10 = 1024; 2 ^ 12 = 4096.

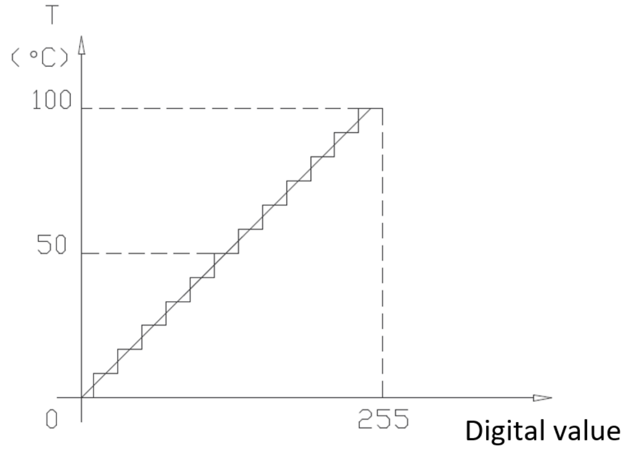
For example, if temperature range is between 0 °C and 100 °C, voltage field is between 0 V and 10 V and number of bits of A/D converter is 8, the resolution of the A/D converter is:

$$\frac{10-0}{2^8} = \frac{10-0}{256} = 0.039\ V$$

This value corresponds to following temperature value:

$$\frac{100-0}{2^8}=\frac{100-0}{256}=0.39\ °C$$

Obviously, the greater the number of bits, the greater the accuracy of digital signal than the analog one. Indeed, in our case resolution would be 0.0976 °C for n = 10 and 0.0244 °C for n = 12.

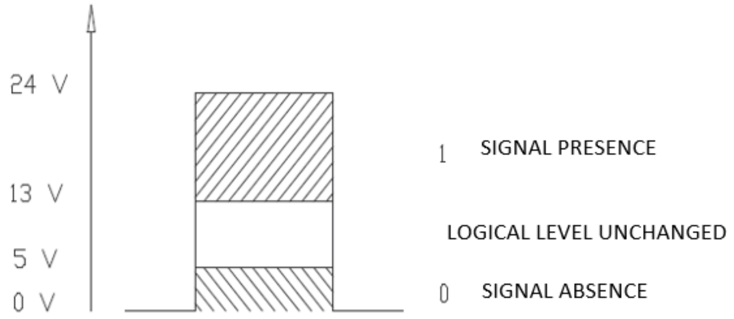


(Fig. 1.12 Digital image of an analogue size)

The digital input boards with logical signal as input carry out a series of activities:

- recognize an ON signal;
- recognize an OFF signal;
- recognize the presence of a noise;
- protect the CPU against short circuits;
- preserve the CPU from overloads;
- shield the CPU from power surges;

The operating principle of the boards is based on the preset operating ranges, for example, if measured voltage value is between 13 V and 24 V, board will provide an ON signal; if, conversely, the value is between 0 V and 5 V, the board will provide an OFF signal; finally, if the value is between 5 V and13 V, the logical signal remains unchanged, which means that if the signal is turned off, it will remain unchanged until the input signal exceeds 13 V, similarly, if the signal is ON, it will remain unchanged until the input signal do not exceed 5 V (figure 1.13).



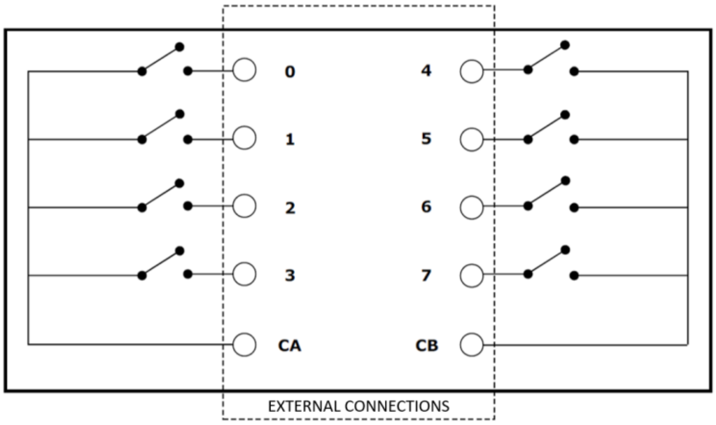
(Fig. 1.13 Example of level bands to recognize variations of status)

Input signals may contain noises, so input boards must filter out parasitic information. The filtration of parasitic information consists in measuring not only the logical level of the signal but also its duration.

As we have already mentioned, the connection between actuators and PLC is via output modules. In other words, the output modules are interfaces that permit the PLC to control actuators.

Like the input modules, the output modules can also be analog and digital.

Particularly, digital output module has some drives and it is divided by group as digital input module.

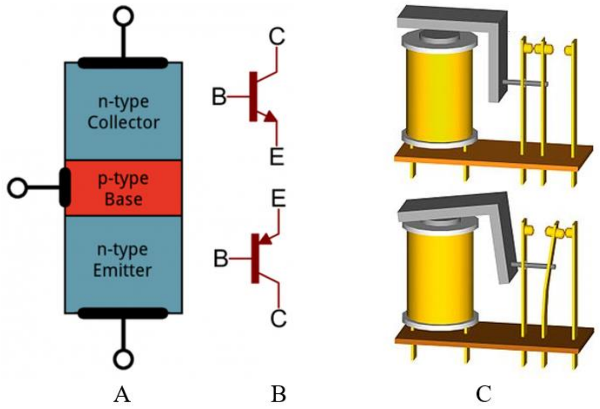


(Fig. 1.14 Scheme of digital output module)

The main feature of the digital output modules is the so-called "output emission time". This is the time interval between the moment when the CPU loads the command in the PLC memory and the moment in which the threshold voltage is reached. This value allows activation of the command.

There are 2 types of digital output modules: relays and static outputs or transistors.

A relay is a switch triggered by the excitation of an electromagnet, while a transistor is a semiconductor-driven switch (figure 1.15).



(Fig. 1.15 (A) transistor structure;(B) transistor scheme; (C) relays structure)

The duration of the emission time of the static output is lower than that of relay, since the time of emission of the latter is caused by time of ascent of relays.

While analog output modules have boards that convert digital signals into analog signals. In other words, the commands processed by PLC in digital mode are converted to analogue quantities so that analogue actuators can easily read them.

The conversion from digital to analogue is performed by D/A converters that make up the analog output modules. They allow to generate an analogue signal proportional to the digital signal. Characteristics of the D/A converter are accuracy, precision and resolution. The calculation of these quantities is analogue to that of the converter.

To conclude, input and output modules are indispensable for communicating between the internal environment and the external environment.

### 1.5.2 Input multiplexing technique

Input multiplexing is a technique that consists in connecting some signals to a single input port of the PLC [Document No. 1 bibliography]. Mainly, this technique is used when a numerical value must be sent to the PLC. Generally, decimal binary encoders (BCD) is the encoding used to perform data transition.

The translation from decimal number to the BCD is done through simple rules represented in the next table.

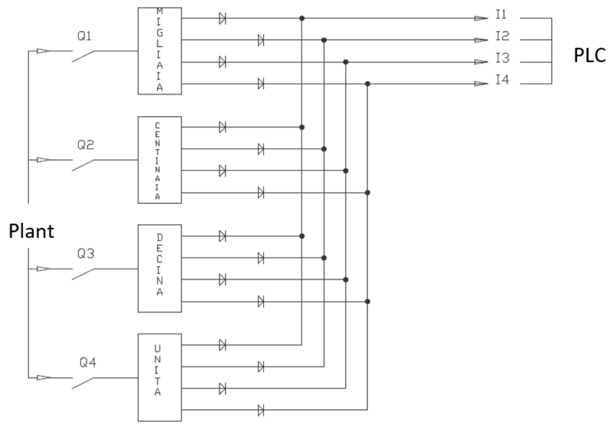
| | MSB | BCD | BCD | LSB |
|---|---|---|---|---|
| Decimal | 8 | 4 | 2 | 1 |
| 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 0 | 1 |
| 2 | 0 | 0 | 1 | 0 |
| 3 | 0 | 0 | 1 | 1 |
| 4 | 0 | 1 | 0 | 0 |
| 5 | 0 | 1 | 0 | 1 |
| 6 | 0 | 1 | 1 | 0 |
| 7 | 0 | 1 | 1 | 1 |
| 8 | 1 | 0 | 0 | 0 |
| 9 | 1 | 0 | 0 | 1 |

(Table 1.2 Rules for BCD code)

The representation of numbers from 1 to 9 requires 4 bits (table 1.2). As a result, the number is divided in according to its position (units, tens, hundreds, thousands etc.) and 4 bits are associated with each position; for example, if value between 10 to 99 needs to be encoded, two blocks of 4 bits are required because one block represents the units and the other the tens. In other words, for each of 4 bits is given an evaluation based on its column value in the binary system.

Moreover, each bit has a specific weight that regulates the transmission of the data in single port of the PLC. Particularly, the least significant bit (LSB) has a value 1, the next bits (going left) take the values 2 and 4, and finally the most significant bit (MSB) assume the value 8 (table 1.2).

The application of this technique is done through the multiplex that transmit the signals of the 4 bits. The figure 1.16 represents application of the multiplexing technique when a 4-digit decimal number must be transmitted (maximum 9999).



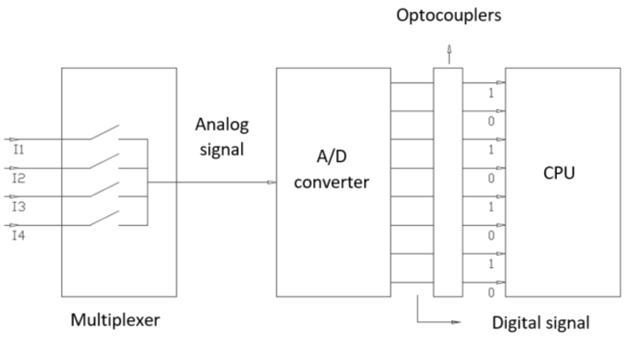
(Fig. 1.16 Multiplex technique)

If this technique was not used, 4 encoders would be needed and each of them would generate 4 signals for total of 16 signals. As a result, PLC should dedicate 16 input pins to this activity.

The application of the multiplexing pin permits to reduce number of input signals to the PLC. Indeed, the 16 signals from the encoders are connected to diodes, which disassemble the signals and transmit them to just 4 PLC inputs (figure 1.16).

The main advantage of this technique is the reduction of the input pins number because only 4 inputs are necessary. However, instead the main disadvantage is the complex management of the software. This disadvantage is not a serious problem because today's PLC are able to handle very complicated software.

The use of the multiplexing technique is performed similarly for analogue cards that can have different inputs. In this case, the inputs are switched to a single converter. The figure 1.17 shows the switching of 4 signals (I1, I2, I3 and I4) in a single analogue signal. This last signal is converted to digital signal by means of the A/D converter and it is sent to the CPU via Optocouplers.



(Fig. 1.17 Multiplex technique for analog signal)

Benefit, also in this case, is the cost reduction thanks to reduction of the input pins; instead, the increase in the data acquisition time is an inconvenience.

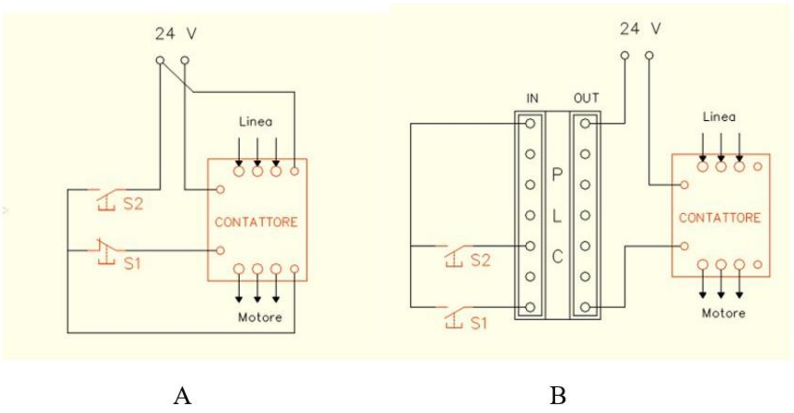
To conclude, multiplexing technique lets to reduce significantly number of PLC pins for transmission of decimal numbers and cost of a control system: PLC, drives and additional modules.

### 1.5.3 Central processing unit (CPU)

Central processing unit (CPU) is the part of automatic system in charge of executing and coordinating all operations required for automation and it is considered the "brain" of the system for this reason. Particularly, CPU task is to process the instructions described by programmer starting from the signals of input pins, and generating the commands per unit of implementation. Its main element is the microprocessor, that is able to perform all functions of calculation and all functions of control. The main feature of CPU microprocessors is the possibility to be programmed, because this characteristic has allowed the transition from wired logic to a programmable logic. The wired logic is more expensive than the programmed one because it consists in proper connection of different devices adapt to fulfill particular function. If the function changes, whole connection must be changed and in some cases some devices have to change.

On the other hand, the programmed logic allows to execute function through predetermined functions and when the function varies, only the instruction must be changed (figure 1.18).

Gear-stop of a motor



(Fig. 1.18 Difference between wired logic and programmed logic)

The 1.18 A image shows physical connection of the "S1" and "S2" switches respectively to motor and to power supply line in the case of the wired logic. Instead, in the case of the programmable logic 2 switches are within the PLC program (figure 1.18 B).

There are many types of CPUs on the market and each manufacturer places the most suitable CPU in the PLC arbitrarily because there is no standardization. For example, Siemens company has some CPUs belonging to family 1200 and 1500.

In conclusion, CPU is that internal device to the PLC that permits program processing and command generation.

### 1.5.4 Memory

Memory is an electronic device that is able to store information and lets other components to read them.

The memories in the PLC can be classified in 2 groups: auxiliary memory and program memory.

Specifically, program memory has both RAM and ROM components.

Read Only Memory (ROM) is used to store the PLC operating system, because it is a read-only memory and is not volatile. In other words, it can be read but it cannot be deleted or written; even if the power supply does not work and the spare battery is discharged, data remains stored.

PLC can have different types of ROM memory: PROM, EPROM and EEPROM. The choice depends on company and potential of the control system hardware.

Random Access Memory (RAM) stores the user program, because it can be read and written. Indeed, in the test phase and run phase the user must be able to modify the program every time he wants it to be according to his needs.

Auxiliary memory stores system inputs and outputs, program variables and intermediate results during program execution. These values change continuously and therefore auxiliary memory is RAM type memory.

Reserved auxiliary memory for intermediate results is called "Flags".

Generally, PLC reads flags as a word or double word, but flags are read as simple bits when only logical value is to be read. Auxiliary memory partition that performs certain functions of the program is called "registers".

Generally, there are at least 4 registers in the PLC:

- working log to store the results of logical arithmetic operations calculated by the CPU following the program's instructions;
- auxiliary register that stores data during special situations;
- stack register to perform the same function as the log when the job is busy;
- status register that stores the system states.

To conclude, the memory is an indispensable element for PLC because it represents place where data are collected, instructions and all information for correct execution of program.

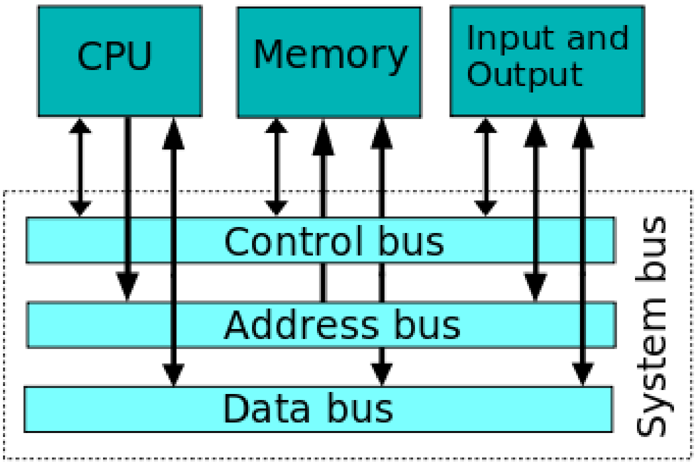
### 1.5.5 Bus system

Bus system is an internal circuit that allows communication between various components that are present in the PLC. It constitutes of a series of internal connections that manages the communication between control units and input modules, communication between processing units and output modules, supply voltage and mass potential.

The bus is divided in according to the functions carried out in:

- address bus, that lets access to the addresses of the individual boards;
- data bus that agrees to the reading of data from both input modules and output modules;
- command bus, that is used to manage command and control signals to perform control of the actuators and to ensure the correct operation performance by the actuator.

In conclusion, the bus system can be defined as that network of cables which carries signals for a correct internal communication.

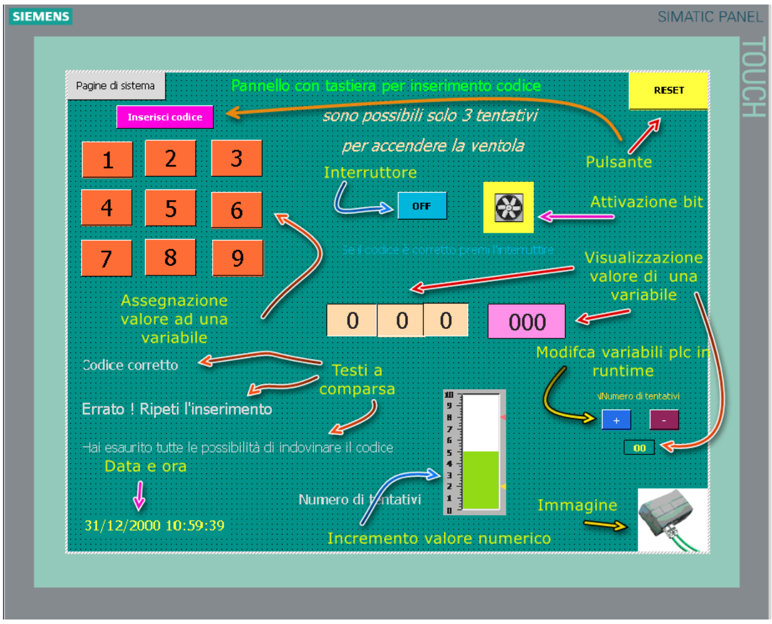


(Fig. 1.19 Bus system structure)

### 1.5.6 External peripherals

External devices are all those devices that communicate with PLC from outside. Indeed, PLCs can communicate with other PLCs, human machine interface devices, printers, devices for emission of alarm signals and devices that perform some functions. Specifically, programmable controller is connected to peripheral units like:

- programming console;
- mass memories;
- simulators;
- EPROM programmers;
- interface for printers and plotters;
- connection modules for personal computers;
- service unit (electronic keys).


SIEMENS
SIMATIC PANEL
TOUCH
Pagine di sistema
Pannello con tastiera per inserimento codice
RESET
Inserisci codice
sono possibili solo 3 tentativi
per accendere la ventola
Pulsante
Interruttore
OFF
Attivazione bit
1
2
3
4
5
6
7
8
9
Se il codice è corretto premi l'interruttre
Visualizzazione valore di una variabile
Assegnazione valore ad una variabile
0 0 0
000
Modifca variabili plc in runtime
Codice corretto
Testi a comparsa
Errato ! Ripeti l'inserimento
Numero di tentativi
+
-
00
Hai esaurito tutte le possibilità di indovinare il codice
Data e ora
Numero di tentativi
10 9 8 7 6 5 4 3 2 1 0
Immagine
31/12/2000 10:59:39
Incremento valore numerico

(Fig. 1.20 Control panel for Siemens PLC)

The connection modules for personal computers are very important because they are able to exchange of information between different computers of the same factory or between different computers of different factories. In this way, it is possible to control the production, the management data relating to process, the remote control and the modification of the parameters and, above all, the detection of eventual faults even at distance, thus allowing the remote diagnosis and the resolution of failures. In other words, they consent to apply the definition of "Industry 4.0" in daily reality of factory at any level.

External peripherals extend the modular horizon of PLC, which is not a simple control device applicable in special cases, but a flexible device capable of performing a large-scale control.

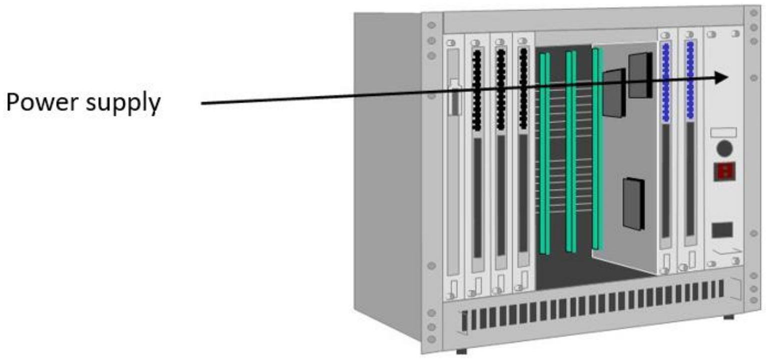
### 1.5.7 Power supply

Power supply is a device that converts line current to direct current. Generally, the power supply voltage of the PLC is either 12 V or 24 V.

Engineering characteristics of power supply depends on the type of PLC and its components, because the power supply must be constant and must not damage all control hardware. It constitutes of transformers, rectifiers for conversion of AC into DC and regulators to provide proper power supply.

The device, which allows the switch to the backup battery, can be considered part of the power supply, indeed it powers PLC when there is no current from electrical grid so that content of RAM is not lost.

In conclusion, power supply is an AC-DC converter that transforms alternating current of transmission line into direct current for operation of the PLC.



(Fig. 1.21 Power supply of PLC)

## *1.6 Basic PLC operation*

The PLC operation is based on sequential and cyclic data processing, because instructions are performed individually from first to last and PLC executes the first instruction after the last processing.

The main key parameters of a sequential operation are the processing time and the presence of contradictory statements.

Processing time is the time taken by the control unit to complete the execution of all program instructions; instead, the presence of contradictory instructions creates situations of uncertainty and even errors, visible even during the programming phase.

Processing time depends on the number of program instructions; also, instruction execution time is not constant but depends on length and complexity of the given instruction. The instruction execution time is the time interval that starts from the instant in which the CPU loads the instruction from memory and ends at the instant the next instruction is read from memory.

During the execution time, the instruction is loaded by means of memory address that is held by the program counter. The instruction is decoded and executed by the CPU, and then the CPU passes to the next instruction.

Another factor influencing reprocessing time is the CPU type of the PLC, which can vary not only among PLC of different brands but also among PLC of same brand.

As a result, the sum of execution times of all the instructions is the program's processing time and is called cycle time or scan time.

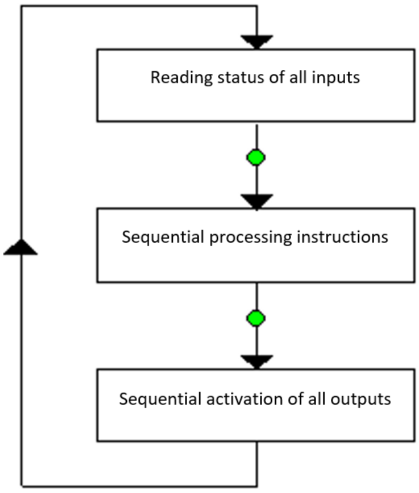
To these factors it is necessary to add a non-constant delay generated during acquisition of new information. This delay is variable because it depends on when the input value changes.

This delay contributes to increase the reaction time that is calculated by adding the delay time during the acquisition of the variation and the duration of the cycle. In other words, reaction time is sum of the cycle duration and a forementioned delay and it is therefore also variable.

The PLC cycle is characterized by following steps (figure 1.22):

1. after PLC is switched ON, the control unit reads input values to assess the presence or absence of voltage. The presence or absence of voltage indicates the status of inputs, which is represented in binary mode: presence of voltage status = 1, absence of voltage status = 0;
2. the processing unit begins to process the program instructions by collecting information about the input states. If it is necessary, during program execution, the control unit accesses various memory cells that contain the variables and characteristics of any counters, timers and markers. The results of each instruction are obtained;
3. the results of each instruction are transferred to output modules, generating actuator controls;
4. a new cycle is performed starting from the first instruction.

In conclusion, basic PLC operation consists of a sequential and cyclic processing of variable duration that characterizes reaction time of control system.


Reading status of all inputs
Sequential processing instructions
Sequential activation of all outputs

(Fig. 1.22 Basic operation of PLC)

## *1.7 Concept of cycle in PLC*

The PLC cycle is marked by 3 phases: the acquisition of input states, the data processing and the sending of commands [Document No. 1 bibliography]. Steps about inputs reading and outputs sending can occur directly or indirectly through a memory.

In the event that the input acquisition takes place directly, whenever state of an input changes, it will be immediately updated in the PLC in real time. In other words, in this case the PLC activities are:

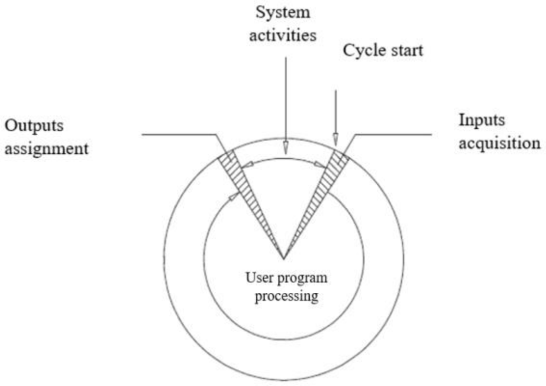
1. acquisition of input states;
2. processing instructions;
3. update states even if the loop is not terminated.

This sequence is similar the one of outputs if sending commands occurs directly.

If there is an intermediate memory, the data is acquired from the CPU and memory at the same time, but the update of any new states will occur at the cycle end. In this case PLC phases are:

1. acquisition of input states;
2. processing instructions;
3. end of cycle;
4. state update.

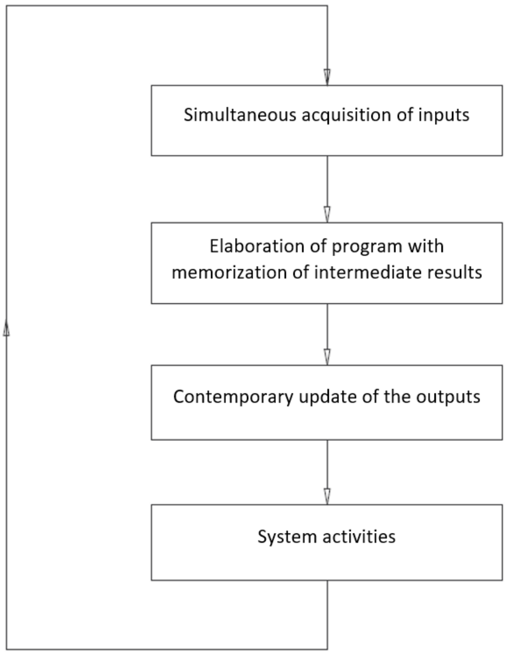
Generally, the timeline of the cycle phases can be represented in the following figure:



(Fig. 1.23 Cycle phases of PLC)

Types of cycle that manage inputs acquisition and the output updating have been standardized by companies of PLC like Siemens and Rockwell. There are 3 different types of cycles: the input and output synchronous cycle, the input synchronous and output asynchronous cycle and the input and output asynchronous cycle.

In the case of a synchronous cycle, the input and output processing phases performed by the PLC are shown in figure 1.23:


Simultaneous acquisition of inputs
Elaboration of program with memorization of intermediate results
Contemporary update of the outputs
System activities

(Fig. 1.23 phases of synchronous cycle )

The synchronous cycle does not perform a direct connection, but it uses intermediate memories called process images. Process images are tables that store input and output states. Especially process images for inputs are indicated with the acronym "PII", while those of outputs are indicated by the acronym "PIQ" [Website No. 3 sitography].

The phases of a synchronous cycle (figure 1.23) are:

- reading of input states;
- write the input values to the "PII";
- program execution according to the states in memory;
- the storing of the commands in the "PIQ";
- application of the controls to the actuators.

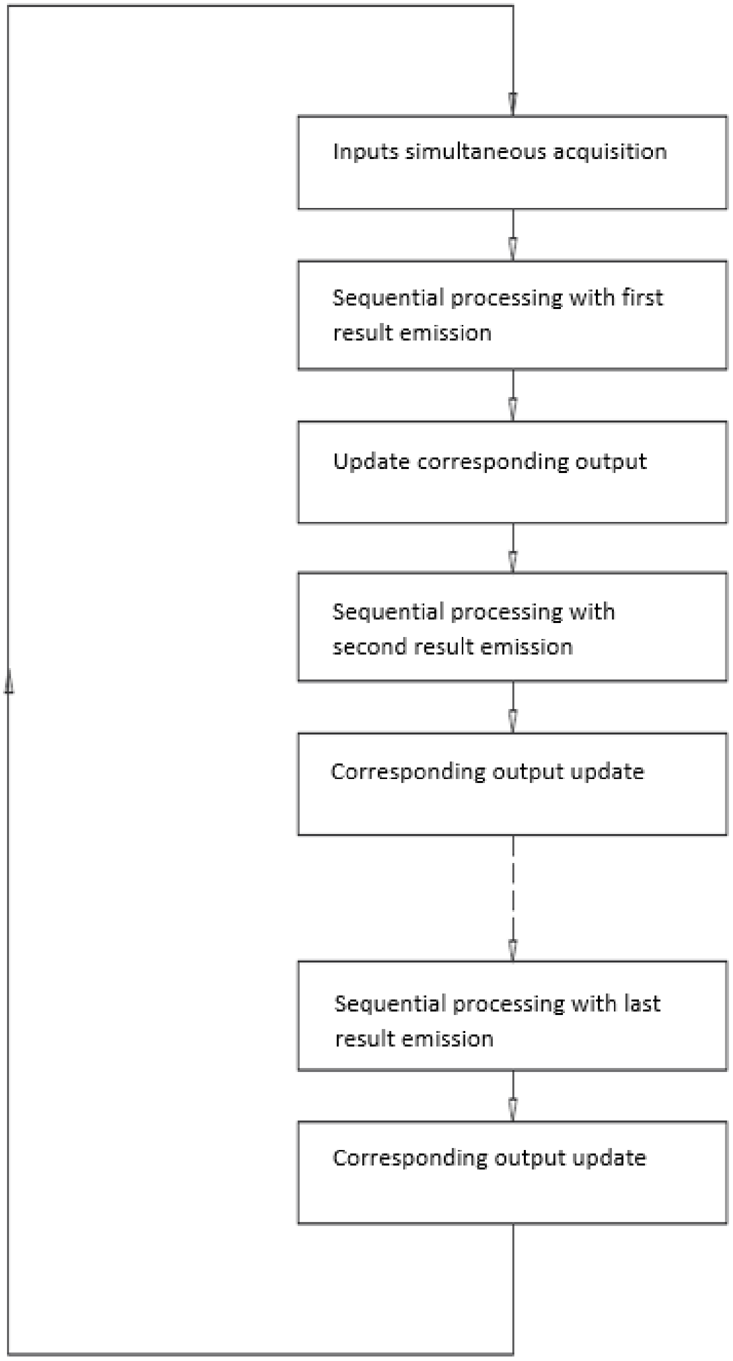
This approach is typical of low-level PLCs.

The second type of loop is the synchronous cycle on the input and the asynchronous cycle on the output.

It differs from the synchronous cycle in input and output for updating outputs, because it manages the inputs by means of auxiliary memory and outputs directly. In other words, calculation of single output and its updating take place at the same time without being used in process image.

The synchronous cycle phases on the input and asynchronous on the output (figure 1.24) are:

- reading of input states;
- write the input values to the "PII";
- calculation of commands according to states in memory and their application to actuators in real time.


Inputs simultaneous acquisition
Sequential processing with first result emission
Update corresponding output
Sequential processing with second result emission
Corresponding output update
Sequential processing with last result emission
Corresponding output update

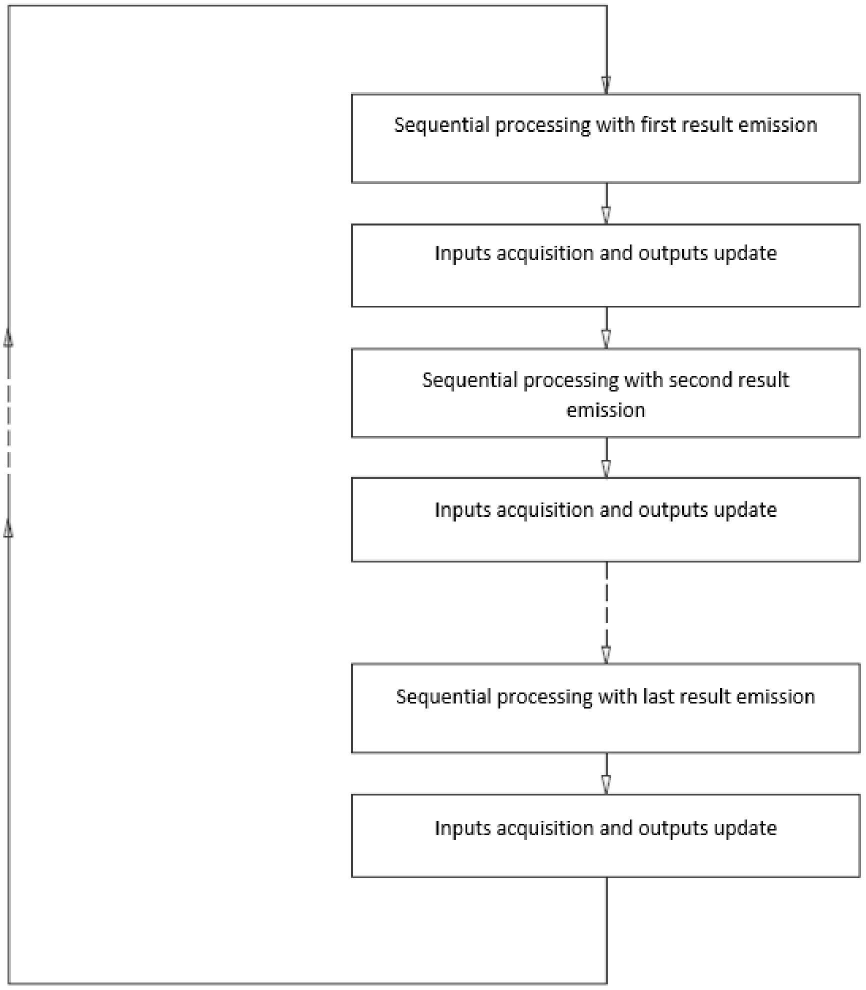
(Fig. 1.24 Phases of synchronous cycle in input and asynchronous in output.)

Finally, the asynchronous input and output cycle applies direct approach to both reading inputs and sending commands.

The phases of asynchronous cycle (figure 1.25) are:

- reading of input states;
- implementation of the first instruction of program;
- application of command and acquisition of new input states.

These tasks are performed for each statement (figure 1.25).


Sequential processing with first result emission
Inputs acquisition and outputs update
Sequential processing with second result emission
Inputs acquisition and outputs update
Sequential processing with last result emission
Inputs acquisition and outputs update

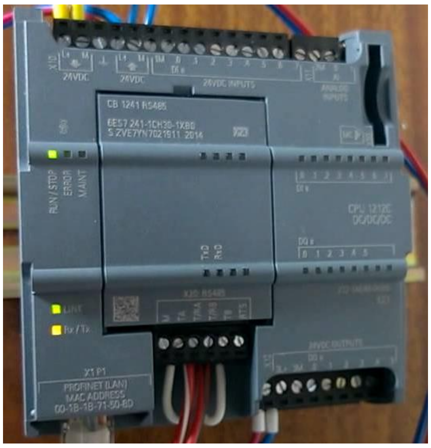
(Fig. 1.25 Phases of asynchronous cycle.)

Generally, this approach is applied by high-level PLCs.

To conclude, the PLC cycle types are categorized in according to the handling of input and output signals, and they are synchronous input and output cycle, synchronous input and asynchronous output cycle, and asynchronous input and output cycle.

## 1.8 Addressing

The objective of the PLC is to manage state of outputs in according to input states and, both the incoming signals and the outgoing signals are necessary. These signals are connected to the PLC via modules that can be inside body containing the CPU (figure 1.26) or outside as additional modules (figure 1.27).



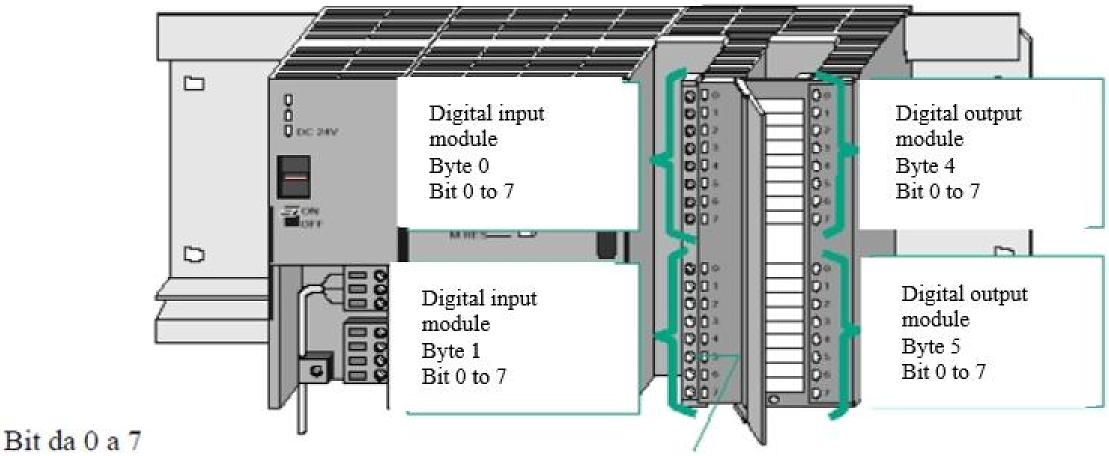
(Fig. 1.26 CPU 1241 Siemens)

(Fig. 1.27 Siemens additional module SM 1231)

Each module has pins to which external devices, such as sensors and actuators, are connected. In the case of input, the signals come from the outside, while in the case of output the signals are sent by the CPU. In both cases, you need an interface that permits a connection between hardware (physical pin) and software (logical signal value). The companies of PLC impose to each bit an address that allows creation of this connection.

Particularly, each digital module has 8 pins which are divided into 2 groups of 4 pins each. Each pin is associated with one bit, and each module has a byte, because a byte corresponds with 8 bits. The byte address is the number that marks a module, instead the single pin is marked with the bit address from 0 to 7, so the address of the first bit of any byte is 0, and the address of the last bit of any byte is 7.

In addition, the input pins are indicated by capital letter "I", while the output pins are indicated with capital letter "Q" (figure 1.28).



(Fig. 1.28 Scheme of Siemens CPU with digital input module 0 and 1, and digital output module 4 and 5 )

As a consequence, complete address of input pin is:

$$I\ X.Y \tag{1.1}$$

where X is the byte address and Y is the pin address.

For example, given the PLC shown in figure 1.28, the address of the fifth input from above is:

$$I\ 0.4 \tag{1.2}$$

because the byte address is 0 and the bit address is 4.

While type of output pin complete address is:

$$Q\ X.Y \tag{1.3}$$

where X is the byte address and Y is the pin address.

For example, the address of the last exit is:

$$Q\ 5.7 \tag{1.4}$$

because the byte address is 5 and the bit address is 7.

This type of addressing applies to both analog and digital signals. Specifically, an analog signal's address indicates a word and not a single bit because analog signals are words.

In conclusion, addressing is activity that permits translation of a physical signal into a virtual logical signal.

## *1.9 Program languages of PLC*

The programming language is a set of elements, blocks and standardized rules that allows the programming of processor to perform defined functions. In other words, the programmer writes a set of instructions that the PLC must perform by means of default programming language.

Generally, the programmers use high-level programming language, while the one of PLC processor is of low level. The task, that allows conversion from a high-level programming language to a low-level one, is called compilation and it is performed by compiler. In other words, the compiler changes the program language by moving from a high-level programming language to a low-level syntax.

All programming languages constitutes of a "set of operations" that the programmer must use to implement any function regardless of its complexity.

Each programming language has its own set of operations and therefore the solution of a single problem can be implemented in a different way; indeed, it is not possible to switch from one programming language to another automatically. Generally, the set of operations are elementary functions that come in logical operations, counts, compare, or timers.

Programming languages are divided on the basis of the visual representation of the functions in two categories: graphical programming languages and literal programming languages.

Graphical programming languages are characterized by the use of graphical symbols, while literal programming languages use mnemonic literal codes.

The programming languages can follow the school of American thought and the school of German thought.

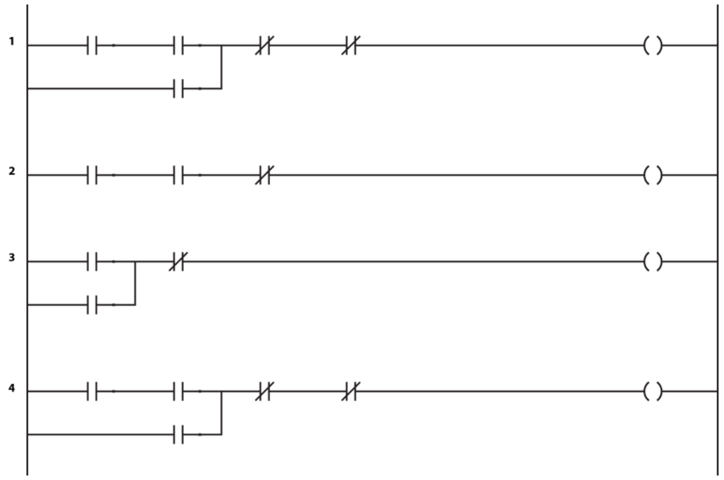
The programming languages of the American school are:

- Ladder diagram, a language based on graphic symbols (figure 1.29);
- Boolean keys, a language based on “mneomonic” literal codes (figure 1.30);

- Functional block, a language based on functional blocks containing an instruction (figure 1.31);
- High-level language (HLL), a language that can also be used by computers (figure 1.32).

The programming languages of the German school were created by Siemens and they are:

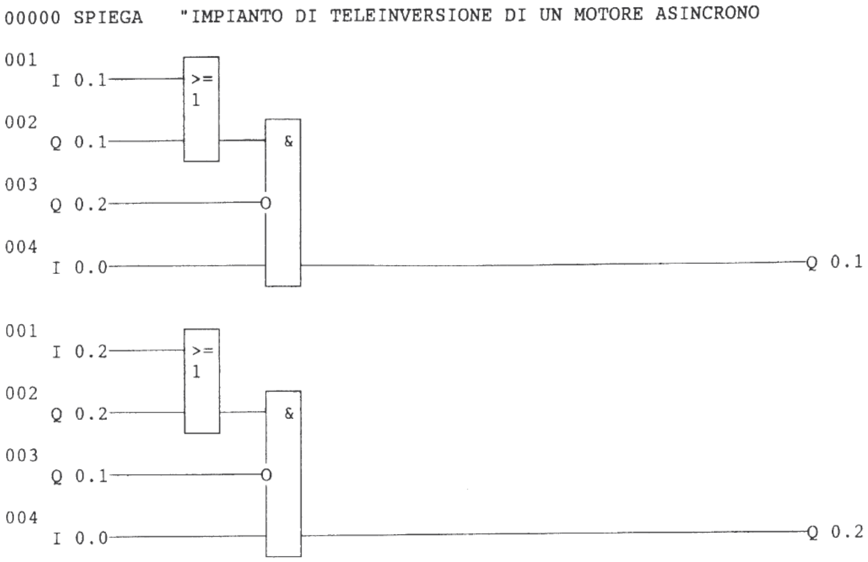
- KOP, a contact programming language based on graphical symbols corresponding to the Ladder diagram (figure 1.29);
- FUP or FBS, defined as a logical scheme corresponding to the functional block (figure 1.31). It is a graphical language because it uses logical blocks as symbols;
- AWL, a list of instructions (STL), which could also be called a symbolic language because desired instructions are written using mnemonic literals that identify the set of operations functions such as counters, timers and logical functions (AND, OR, non, NAND, etc.). It corresponds to the Boolean status keys (figure 1.30).
- Structured Control Language (SCL), a high-level structured programming language. It is at the level of HLL language of American origin (figure 1.33).



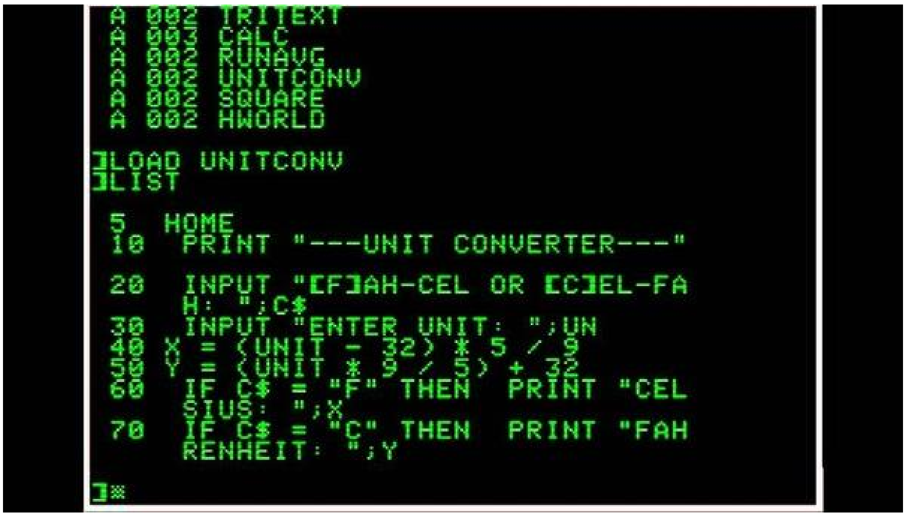
(Fig. 1.29 Example of KOP or LAD programming language)

```
000    L I0.1
001    O Q0.1
002    ANQ0.2
003    A I0.0
004    = Q0.1
005    L I0.2
006    O Q0.2
007    ANQ0.1
008    A I0.0
009    = Q0.2
```

(Fig. 1.30 Example of STL programming language)

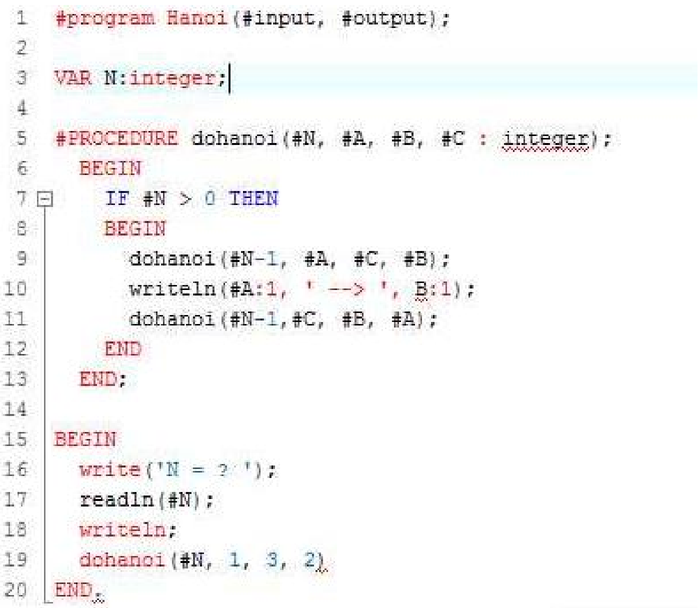


(Fig. 1.31 Example of FUP or FBS programming language)



```
A 002 TRITEXT
A 003 CALC
A 002 RUNAVG
A 002 UNITCONV
A 002 SQUARE
A 002 HWORLD

]LOAD UNITCONV
]LIST

 5  HOME
 10  PRINT "---UNIT CONVERTER---"

 20  INPUT "[F]AH-CEL OR [C]EL-FA
    H: ";C$
 30  INPUT "ENTER UNIT: ";UN
 40 X = (UNIT - 32) * 5 / 9
 50 Y = (UNIT * 9 / 5) + 32
 60  IF C$ = "F" THEN  PRINT "CEL
    SIUS: ";X
 70  IF C$ = "C" THEN  PRINT "FAH
    RENHEIT: ";Y

]
```

(Fig. 1.32 Example of HLL programming language)



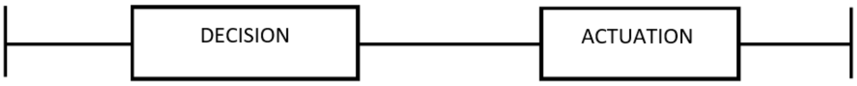
```
#program Hanoi(#input, #output);

VAR N:integer;

#PROCEDURE dohanoi(#N, #A, #B, #C : integer);
  BEGIN
    IF #N > 0 THEN
    BEGIN
      dohanoi(#N-1, #A, #C, #B);
      writeln(#A:1, ' --> ', B:1);
      dohanoi(#N-1,#C, #B, #A);
    END
  END;

BEGIN
  write('N = ? ');
  readln(#N);
  writeln;
  dohanoi(#N, 1, 3, 2)
END.
```

(Fig. 1.33 Example of SCL  programming language)

Especially, the representation of the scale or the programming language KOP is very intuitive, because it consists of sequences of networks. Each network constitutes of two parts: a decision-making part and an implementation part.

Decision block is to the left of the segment and it is a combination of elementary functions belonging to the set of KOP operations.

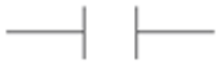
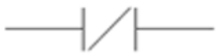
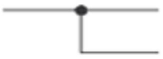
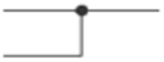
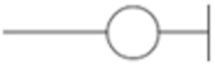
The implementation part is to the right of the segment and it consists of one or more values, which depend on the result of decision-making party (figure 1.34).



(Fig. 1.34 General network of ladder diagram)

Scale diagram is similar to the representation of the electrical circuits, even if there are some differences. For example, the power lines are shown vertically in the ladder diagrams and horizontally in the electrical circuits; in addition, the electrical systems are shown in horizontal position in the ladder diagram and in vertical position in the electrical circuits (figure 1.36).

The ladder scheme consists of switches and circles representing the coils (figure 1.35).

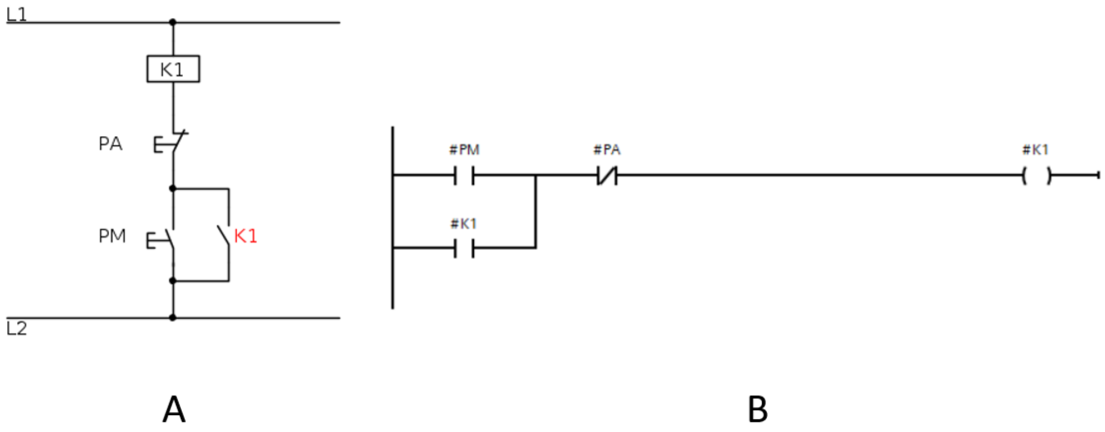
| | |
|---|---|
|  | Operand programmed not denied |
|  | Operand programmed denied |
|  | Opening a branch in parallel |
|  | Closure of a branch in parallel |
|  | Internal or external output |

(Fig. 1.35 Basic elements of ladder diagram)

Switches are called operands and they can be negated (not denied) or no-negated (denied). When a status of 1 is supplied, the no-negated switch activates, while a negated switch turns off. Conversely, when a 0 state is supplied, a no-negated switch is deactivated, and a negated switch activates.

Each operand represents the logical state (not the physical state) of a given signal generator that can be actuator, memory or sensor.

Another difference between the diagram and the electric circuit is the operating logic: the ladder diagram goes from the top to the bottom, while the one of electric circuit goes from left to right.



(Fig. 1.36 A Electrical circuit of self-contained; B ladder diagram of self-contained)

Nowadays, Structured Control Language (SCL) is widely used by programmers, because it eases some functions and it conforms to the PLC standard (IEC 61131).

It is a high-level programming language that is based on the Pascal programming language. It allows structured programming using top-level elements and simple elements such as input, output, timers, counters, etc. In other words, it integrates and expands the functions of other lower-level programming languages such as the KOP.

In conclusion, PLC programming languages let the programmer create a complex software by means of basic functions in a simple and intuitive way.

## *1.10 Set of operations and data type*

The program of PLC must be implemented using programming languages. During the implementation of program, the programmer needs different data types; therefore, the programming languages must be able to evaluate different kinds of variables. The choice of variable takes place in according to the type of operation to be performed. For example, the implementation of the presence or absence of a signal requires a boolean variable, while the measurement of magnitude requires a variable of real or integer type.

PLCs and programming languages are able to evaluate the following variables: boolean, integer, double integer, word and real. Each type of variable has different characteristics that make it suitable or not for certain situations.

The Boolean variable is indicated with the symbol "BOOL" in all programming software. Since it is used to indicate the presence or absence of a condition, the transistor is the physical element to which it is connected: if the transistor is open, the variable value will be 0, and if the transistor is closed, the variable will assume the value 1. The perceived value is stored in memory bit that is associated with the variable through the addressing activity.

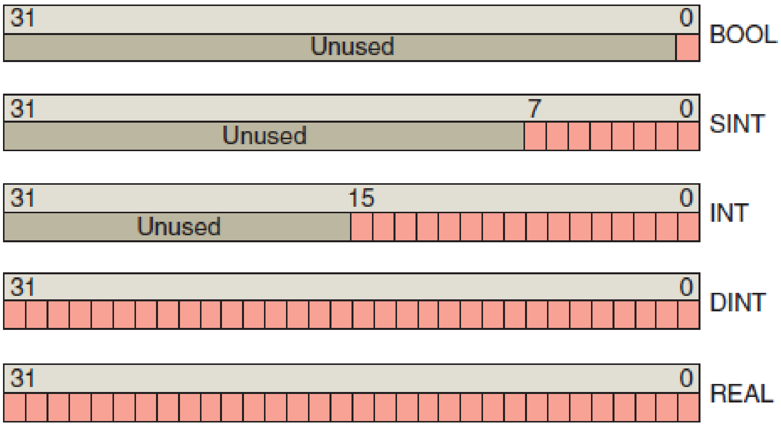
Bits are grouped blocks of memory that are used to store values greater than 1. Particularly, a set of 8 bits generates the byte. The byte is able to store a decimal number between 0 and 255, because it has 256 possible combinations.

If the number of bits increases, even the maximum decimal number that can be stored increases. For example, a word can represent a maximum decimal value of 65535, because it constitutes of 16 bits (figure 1.37 A) and the number of possible combinations is 65536.

Among the types of variables used by PLCs there are also the integer variables and double integer variables. The integer variable contains 16 bits and is used to store values ranging from -32768 to 32767, while the double integer variable consists of 32 bits and can store values ranging from $-2^{31}$ to $-2^{(31-1)}$.

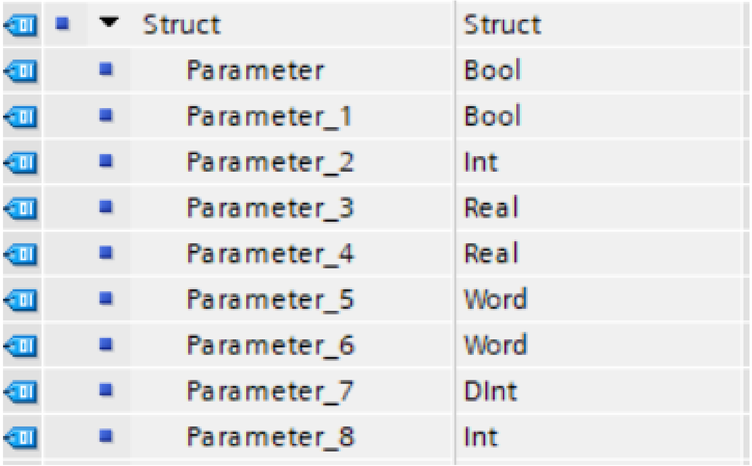
Finally, there is the real variable (real) which is made up of 32 bits and can take values ranging from $-2^{31}$ to $-2^{(31-1)}$.

The main difference between real and integer is that real data store real numbers, while the integer variable only integer numbers.



(Fig. 1.37A Memory struct of data type)

In addition to these types of data related to memory, there are structures. The structure is data type related to the types of information previously described but the programmer can model freely. For instance, it is possible to create a structure composed of 2 words and 1 integer, or 2 words, 2 integers, and 1 bool etc. In other words, the possible combinations are chosen by the programmer arbitrarily and, generally, it is based on the function he wants to elaborate (figure 1.37B).



| Struct | Struct |
|---|---|
| Parameter | Bool |
| Parameter_1 | Bool |
| Parameter_2 | Int |
| Parameter_3 | Real |
| Parameter_4 | Real |
| Parameter_5 | Word |
| Parameter_6 | Word |
| Parameter_7 | DInt |
| Parameter_8 | Int |

(Fig. 1.37B Example of struct)

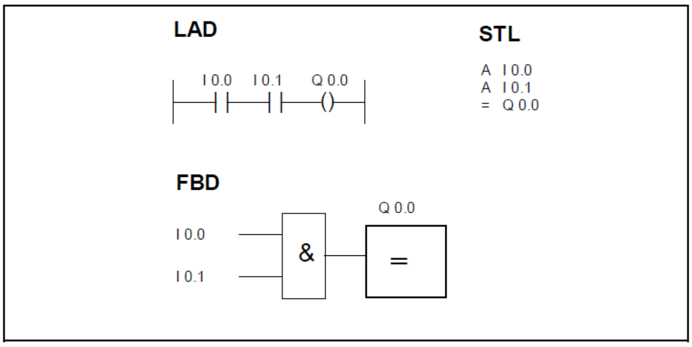
The programming languages have a library of elements that let the implementation of the instructions. This library corresponds to the set of basic functions that are the same for all programming languages, generally. The set of operations can be divided in according to their complexity in:

- basic instructions for the control of electrical circuits;
- instructions for managing the program;
- data manipulation instructions;
- instructions for manipulating numeric data.

### 1.10.1 Basic instructions for the control of electrical circuits

The basic instructions for controlling electrical circuits constitutes of boolean “AND” and “OR” logical functions on individual bits, timers and counter [Document No. 18 bibliography].

Especially, the logic function AND is represented in different ways in programming languages (figure 1.38).



```
A  I 0.0
A  I 0.1
=  Q 0.0
```

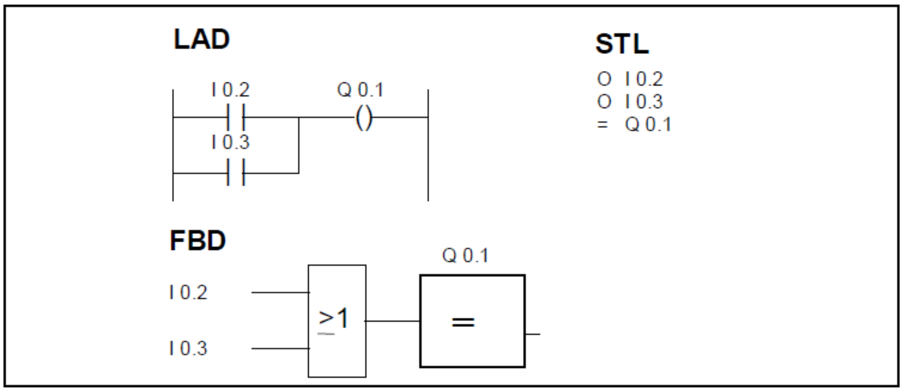
(Fig. 1.38 AND function implementation)

In this case, output Q 0.0 will become true if I 0.0 and I 0.1 switches are both true, as it imposes truth table of the “AND” function (table 1.3).

| I 0.0 | I 0.1 | Q 0.0 |
|---|---|---|
| 0 | 0 | 0 |
| 1 | 0 | 0 |
| 0 | 1 | 0 |
| 1 | 1 | 1 |

(Table 1.3 True table of AND function)

The logical OR function, like the AND function, has several representations that depend on type of programming language (figure 1.39).



(Fig. 1.39 OR function implementation)

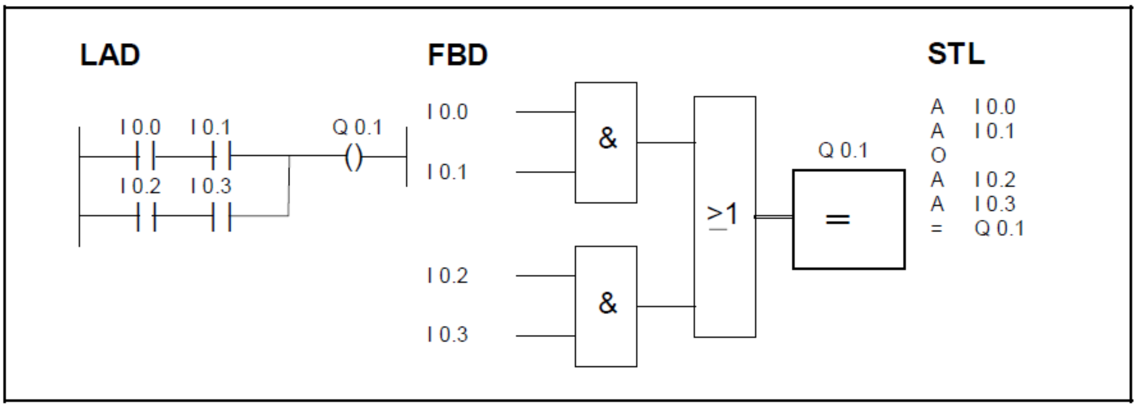
In this case, output Q 0.1 becomes true if I0.2 switch or I0.3 switch is true, as it imposes the truth table of OR function (table 1.4).

| I 0.2 | I 0.3 | Q 0.1 |
|---|---|---|
| 0 | 0 | 0 |
| 1 | 0 | 1 |
| 0 | 1 | 1 |
| 1 | 1 | 1 |

(Table 1.4 True table of OR function)

The application of the logical AND and OR functions is fundamental and often they are used together. For example, figure 1.40 describes the following function:

$$(\,I\ 0.0\ AND\ I\ 0.1)\ OR\ (\,I\ 0.2\ AND\ I\ 0.3) =\ Q\ 0.1 \qquad (1.5)$$



(Fig. 1.40 Implementation of equation 1.5)

The logical AND and OR functions implement standard functions: exclusive-OR operation, pure self-contained, self-contained, set and reset of variables and memories.

Exclusive-OR operation is based on following boolean equation:

$$(I\ 1.0 * \overline{I\ 1.1}) + (\overline{I\ 1.0}\ AND\ I\ 1.1) = Q\ 1.0 \qquad (1.6)$$

It becomes:

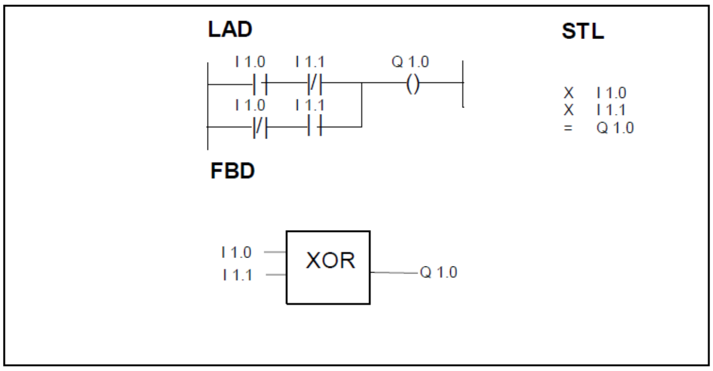
$$(I\ 1.0\ AND\ \overline{I\ 1.1})\ OR\ (\overline{I\ 1.0}\ AND\ I\ 1.1) = Q\ 1.0 \qquad (1.7)$$

Output Q 1.0 is enabled if and only if one of the two inputs (I 1.0 or I 1.1) is active, indeed, output will be false as evidenced by its truth table (table 1.5) if both inputs are true.

| I 1.0 | I 1.1 | Q 1.0 |
|---|---|---|
| 0 | 0 | 0 |
| 1 | 0 | 1 |
| 0 | 1 | 1 |
| 1 | 1 | 0 |

(Table 1.5 True table of XOR function)

The implementation of XOR function depends on type of programming language used (figure 1.41), but in all cases it can only be used with 2 inputs.

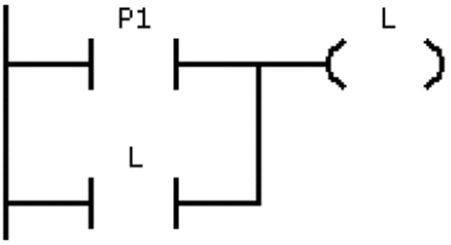


```
X   I 1.0
X   I 1.1
=   Q 1.0
```

(Fig. 1.41 Implementation of XOR operation)

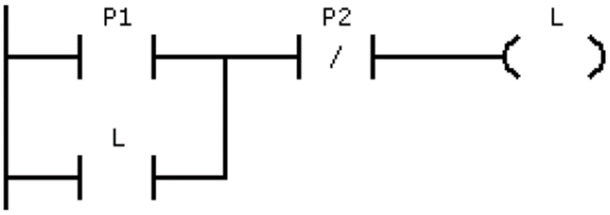
Another standard function is the pure self-contained (figure 1.42).

In this case, the "L" light bulb illuminates when the "P1" switch is active. When the bulb lights up, the "L" switch is activated and light will always remain on even if the "P1" is deactivated. This function is called pure self-contained because it cannot be switched off after turning on the bulb.



(Fig. 1.42 Implementation of pure self-contained by means of ladder diagram )

Instead, the implementation of the self-contained is represented in figure 1.43.



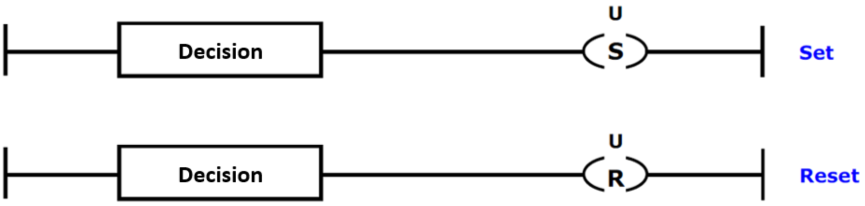
(Fig. 1.43 Implementation of self-contained by means of ladder diagram )

In this case the bulb lights up when the"P1" switch is activated and the "P2" is deactivated. When the light bulb is on, the "L" switch activates and the bulb stays on even if P1 is not active, just like before. The difference is that the "L" bulb can be switched off by pressing the "P2" switch. Indeed, when the "P2" switch is active, the contact opens and the current does not reach the bulb, which turns off and at the same time the "L" contact opens, no longer allowing the passage of the current towards the bulb even if the "P2" is disabled.

In the previous functions the implementation part assumes values that depend on the decision-making part, while in the SET/RESET instructions the one may not happen.

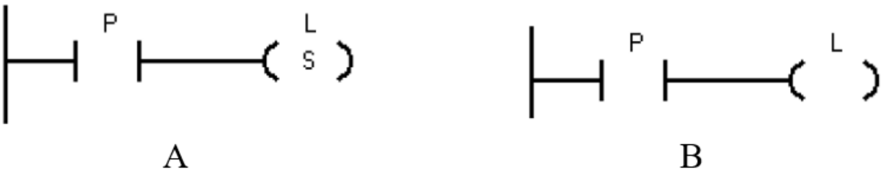
The SET function imposes the value 1 at the output if decision-making part is verified; while the RESET operation equally outputs to 0 if the decision-making part is verified.

The effect of the SET and RESET instructions may also be applied to several outputs at the same time, provided that the addresses of the outputs are different, sure enough the number "n" indicates the number of addresses to which it is connected (figure 1.44).



(Fig. 1.44 Set/Reset function structure)

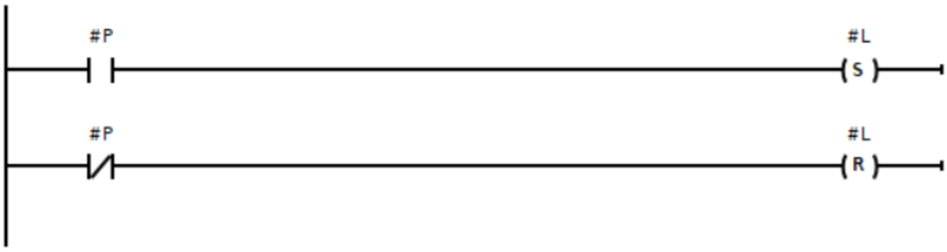
The following figure shows the difference between the use of the SET instruction and the use of a normal coil (figure 1.45).



(Fig. 1.45 A) Set function structure, B) coil structure)

Figure 1.45 B represents a normal coil: the lamp "L" illuminates when the switch "P" is on and it turn off when switch “P” is deactivated. In the case of the SET instruction (figure 1.45 A), the lamp illuminates when the switch "P" is activated and it remains on even when the user switches off the switch “P”.

In the latter case the programmer must add a RESET instruction to switch off the lamp L (figure 1.46).

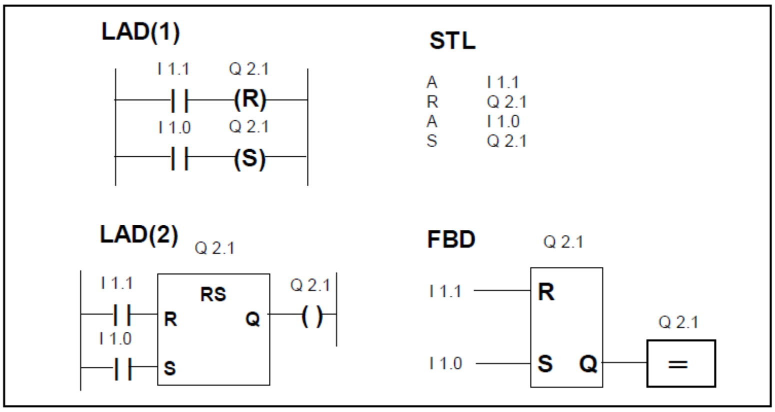


(Fig. 1.46 Example of SET/RESET function)

In the example of figure 1.46 the switch "P" turns on lamp, which will remain lighted until the P switch is deactivated. When P turns off, the RESET function is activated and the lamp goes out. Programmers very often use this statement because it permits them to manage the activation and deactivation of an item according to the conditions that they want.

SET and RESET operations let to implement standard functions such as set dominant and reset dominant.

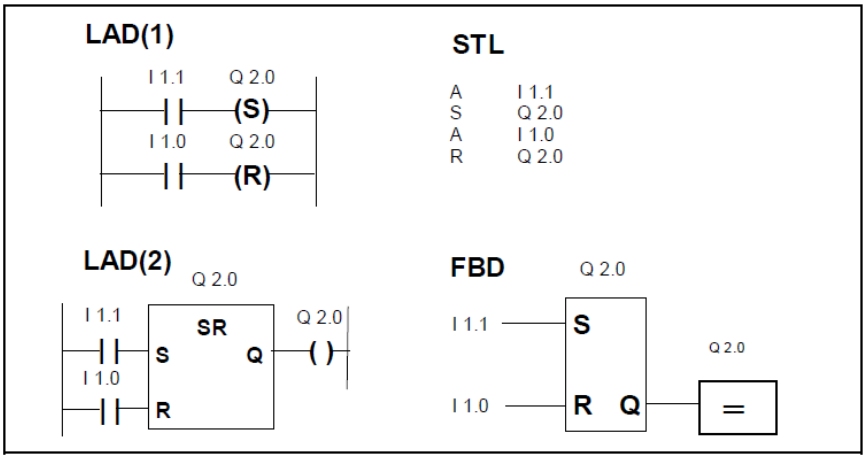
Programmer implements set dominant function when he wants to provide priority to the set function (figure 1.47).



```
A    I 1.1
R    Q 2.1
A    I 1.0
S    Q 2.1
```

(Fig. 1.47 SET Dominant function)

Programmer implements reset dominant function when he wants to provide apriority to the reset function (figure 1.48).

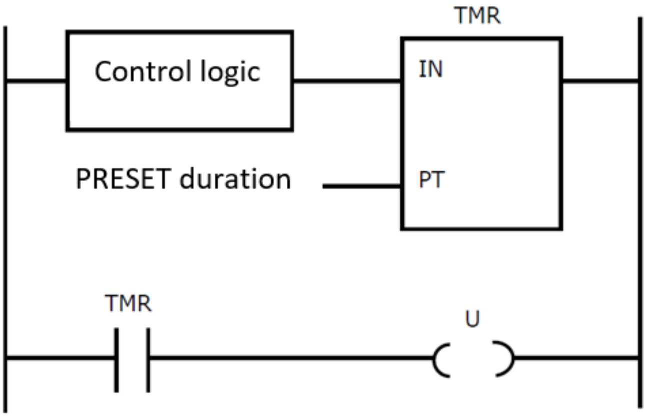


(Fig. 1.48 RESET Dominant function)

AND and OR instructions are the simplest functions that belong to the basic function set. It constitutes of more complex instructions: timers and counter.

The timer is an element that delays the execution of an action such as sending a command.

Figure 1.49 represents a timer using the ladder diagram as the programming language.



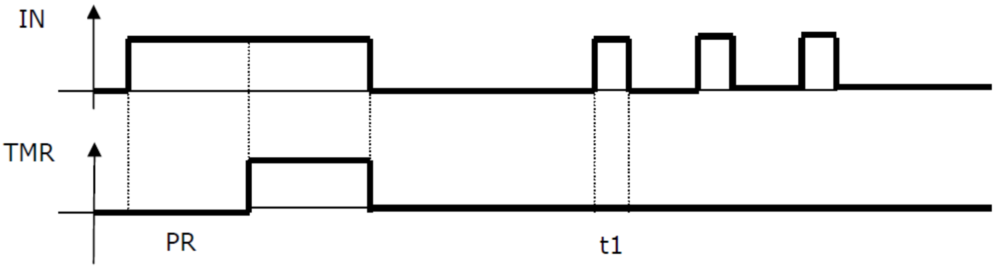
(Fig. 1.49 Working logic of timer)

When control logic is verified, the timer starts counting time, so it activates when it has reached the default delay (PRESET).

Timer activation generates the application of the "U" output. In other words, the control logic does not activate the "U" output immediately but after a predefined delay. The delay depends on the timer resolution used. The resolution is the unit of time used for measuring time. For instance, if the timer resolution is the second, you can set a multiple delay of the second.

There are two types of timers: the timer without memory and the timer with memory.

The time diagram for the timer without memory is shown in the next figure (figure 1.50).



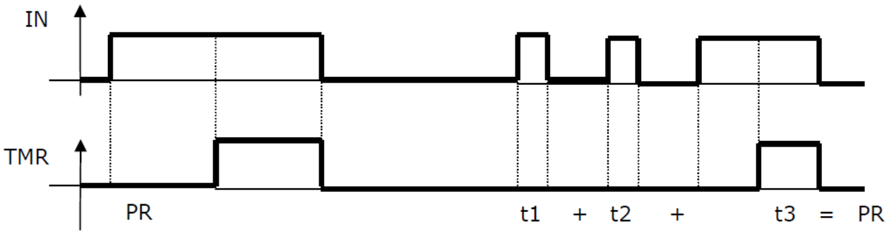
(Fig. 1.50 Timeline for the no-memory timer)

As you can see from figure 1.50, the timer value becomes 1 after the PR delay has elapsed and if the "IN" variable is still active.

If the "IN" input becomes 0 before the delay has elapsed, the timer will not activate.

The timer with memory, however, differs from the timer without memory because it is able to memorize the time for which the condition "IN" has been activated (figure 1.51). As a result, if the "IN" condition has been activated for a period less than the delay, the timer remains in the logical state 0, but this time range is stored and each successive impulse increases it. When time frame has exceeded the default delay, the timer will reach a value of 1.

In addition, the timer with memory needs a reset operation to be reset.



(Fig. 1.51 Timeline for the memory timer)

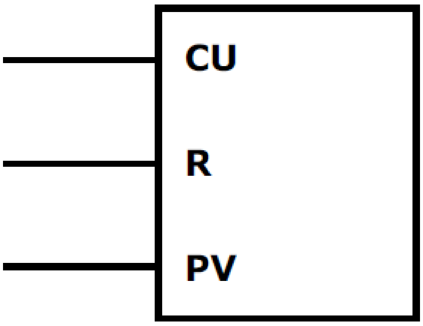
Another element for programming PLC is the counter.

The programmer inserts a counter when he has to count how many times an event happens.

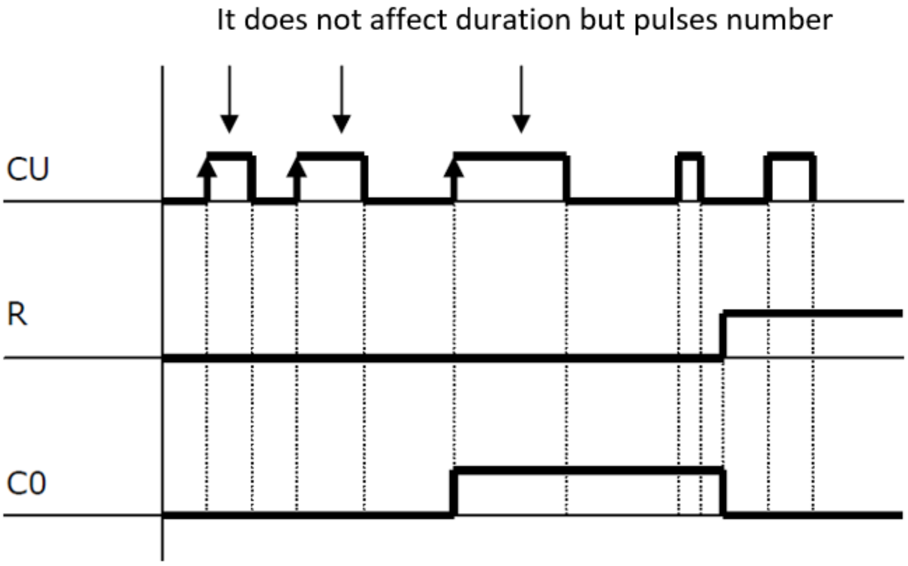
The event to monitor is the input of the counter, so the counter counts the number of pulses that excite its input regardless of the event duration: an event can last 1 s or 50 s but is counted only once (figure 1.51).

There are two types of counters: the forward counter (figure 1.52) and the forward and backward counter (figure 1.54).

The forward counter starts at 0 and it activates its output when the number of pulses from input "CU" reaches the default value of "PV". Thus, user must specify default value by means of input "PV" and the reset condition as input “R”, otherwise the counter will not reset and remain active. Figure 1.52 represents the timeline of forward counter that activates after reaching a value of 3 and turns off only when reset condition occurs.

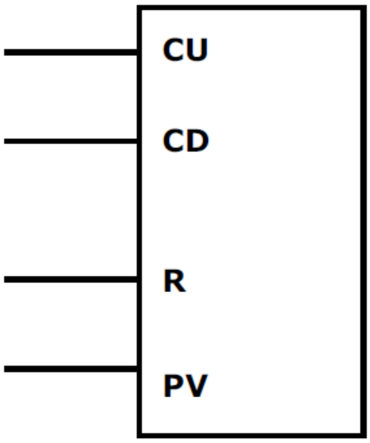


(Fig. 1.52 Forward counter)



(Fig. 1.53 Timeline for forward counter)

The forward and backward counter, on the other hand, is represented in the following image (figure 1.54).



(Fig. 1.54 Forward/Backward counter)

In this type of counter, input CU is dedicated to a forward count, while input “CD” is dedicated to a backward count.

“PV” input is used to record the default value and input "R" is connected to the reset condition. The counter starts at 0 and counts forward when it has to count the number of pulses coming from “CU”; while it starts from the value of “PV” and reaches 0 when it has to count the number of pulses coming from “CD”. The forward/backward counter is activated when it reaches the default “PV” value, which can be both positive and negative.

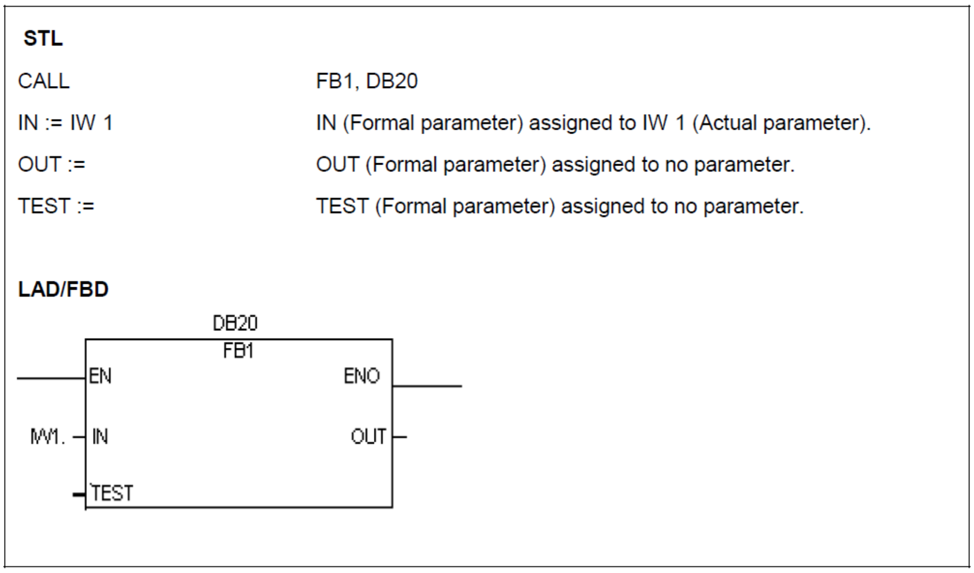
In conclusion, the basic instructions are the indispensable functions to implement a program for electrical circuits control and uncomplicated activities control.

### 1.10.2 The instructions for program managing

The instructions for managing program are another type of instructions that are provided by the programming languages.

They allow to organize the set of instructions and provide an execution priority.

When program is complex and long, programmer divides the program into blocks. In each block he implements more or less complex functions. Each block must be recalled in the main or another block regardless of programming language. Recalling a block inside another or main is performed using the "CALL" function (figure 1.55)



**STL**

```
CALL            FB1, DB20
IN := IW 1      IN (Formal parameter) assigned to IW 1 (Actual parameter).
OUT :=          OUT (Formal parameter) assigned to no parameter.
TEST :=         TEST (Formal parameter) assigned to no parameter.
```

**LAD/FBD**

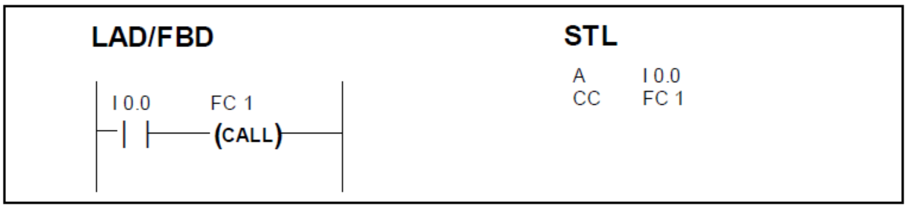
(Fig. 1.55 Implementation of Call function in Siemens software)

In the image 1.55 functional block “FB1” is invoked by associating to it “DB20” data block. It has two inputs and one output:

- inputs IN = IW1 and TEST;
- output is OUT.

Call function is related to the functions "CONDITIONAL CALL (CC)" and "UNCONDITIONAL CALL (UC)".

Conditional call instruction lets to recall a block if and only if the condition has been performed (figure 1.56).



**LAD/FBD**

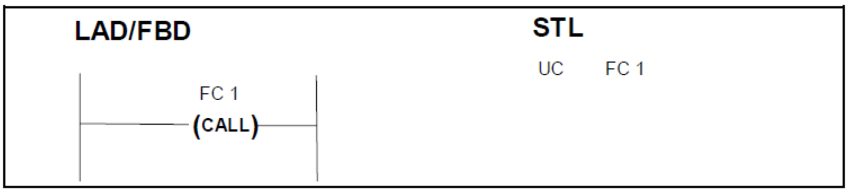
**STL**

```
A    I 0.0
CC   FC 1
```

(Fig. 1.56 Implementation of Conditional Call function in Siemens software)

The FC1 is invoked if only if the condition I 0.0 is true.

Unconditional call instruction permits the programmer recall a block independently of the rest of the program as shown in figure 1.57.

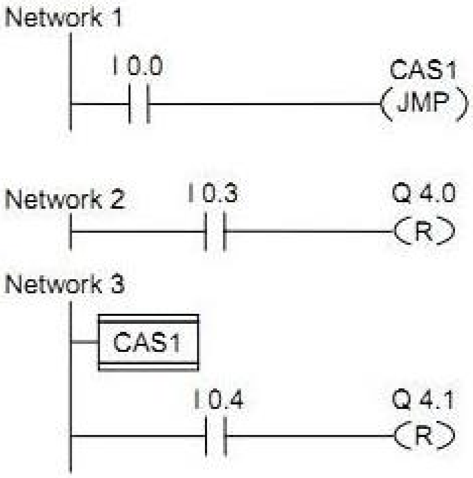


(Fig. 1.57 Implementation of Unconditional Call function in Siemens software)

Another program management instructions example is the Jump function (JMP). It allows PLC to skip over rungs and reach specific network, that is indicated by means of label instruction (figure 1.58)

The jump and label instruction constitute of address number that permits their connection.

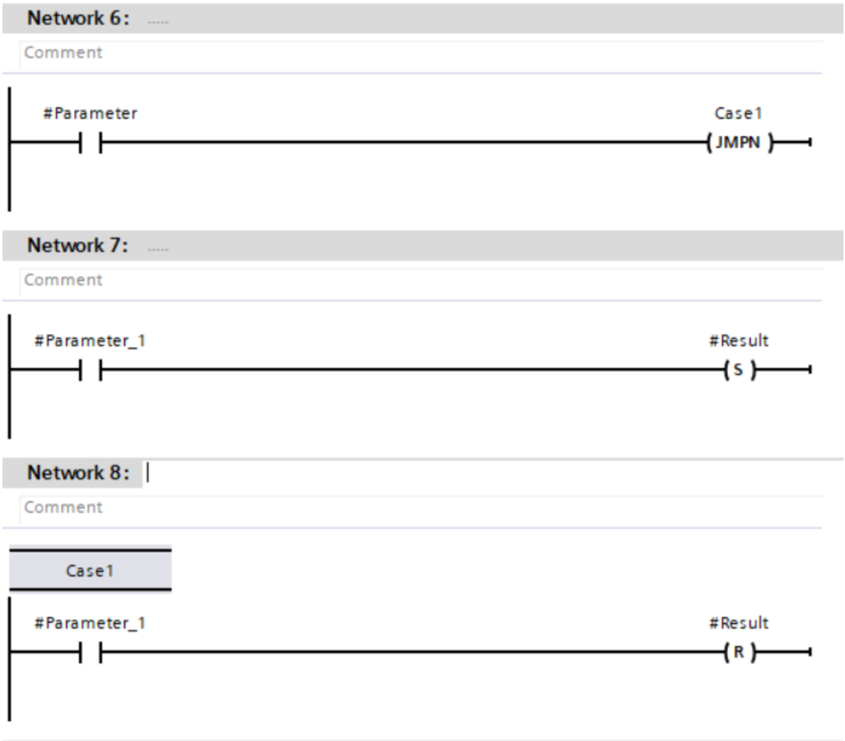
When Jump instruction is true, PLC jumps over rungs and continues to evaluate the instructions following the label element (figure 1.58).



(Fig. 1.58 Implementation of jump function in ladder diagram)

In this case, “Network 2” is jumped when variable I 0.0 is true.

The program management instruction set contains "JUMPN" function. It is analogous to the jump instruction, but, in this case, PLC skips over rungs and continues to evaluate the instructions following the label element when the condition of the JUMPN instruction is false (figure 1.59).



(Fig. 1.59 Implementation of jump function in ladder diagram of Siemens software)

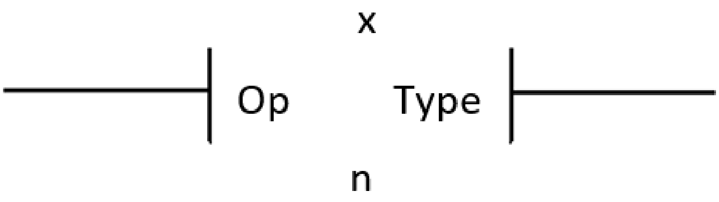
So, network 7 is jumped when variable "Parameter" is false.

In conclusion, program management functions allow the programmer to split the program into different sub-programs and activate them in according to the user needs.

### 1.10.3 Data manipulation instructions

A set of data manipulation instructions constitutes of instructions for moving data into memory, comparing data comparisons, and boolean calculations to bytes and word.

The comparing block is the instruction that allows the comparison operation between numbers. It is made up of 2 values and a comparison mark. In the programming language KOP is represented in the following picture:



(Fig. 1.60 Comparing block in ladder diagram) where:

• "x" is the value to compare;

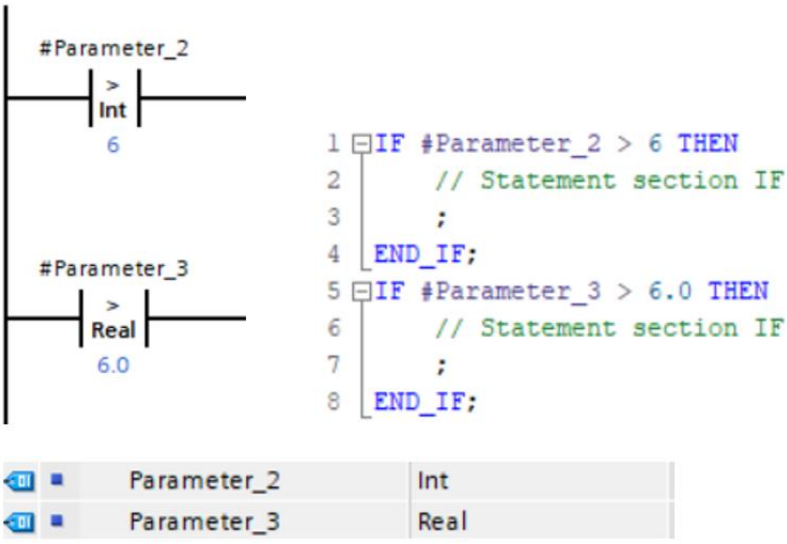
• “n” is the comparison value;

• “Op” is the type of comparison that can be greater (>), minor (<), equals (=), greater than or equal (> =), minor or equal (< =), or different (< >);

• “Type” is the type of data (real, integer...).

Comparing block is activated when the comparison is true.

Comparable numeric variables can be integer, real and double integer, but only numbers of the same type can be compared. For example, if you want to compare two variables, they must be either real or integer or double integer (figure 1.61).



```
IF #Parameter_2 > 6 THEN
    // Statement section IF
    ;
END_IF;
IF #Parameter_3 > 6.0 THEN
    // Statement section IF
    ;
END_IF;
```

| | |
|---|---|
| Parameter_2 | Int |
| Parameter_3 | Real |

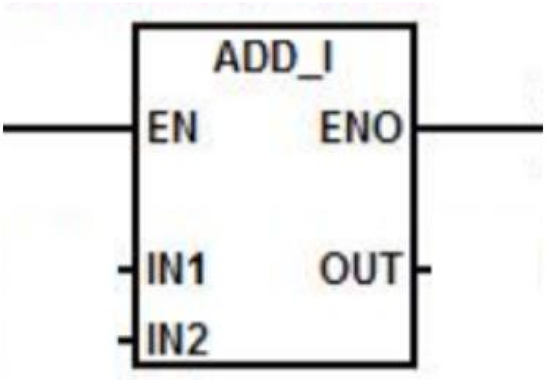
(Fig. 1.61 Comparing blocks by means of LAD and SCL )

In this case the first comparing block becomes true if the variable"Parameter_2" is greater than 6 (integer variable), while the second comparing block becomes true if the variable"Parameter_3" is greater than 6.0 (real variable).

The data manipulation instructions allow programmer to express the conditions that the basic instructions are not able to perform.

### 1.10.4 Instructions for manipulating numeric data

The instructions for manipulating numerical data permit many numeric operations, including basic mathematical calculations, scientific calculations. An example of basic mathematical calculations is the sum. It is performed by means of three variables: the two addendes and the result. The implementation of the sum through the programming language LAD is represented in figure 1.62.



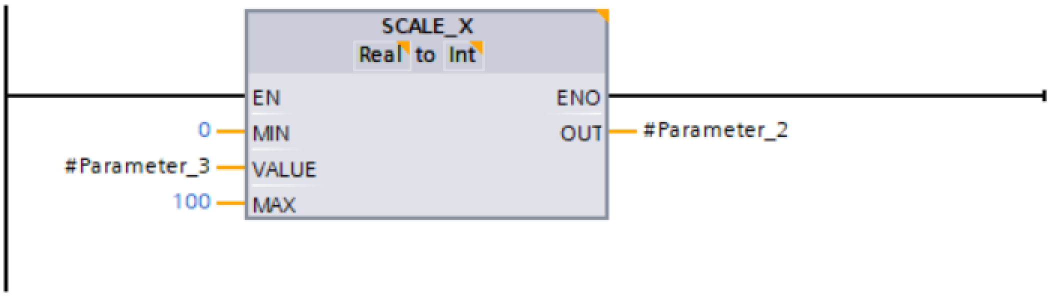
(Fig. 1.62 ADD instruction by means of ladder diagram )

The operation performed by the ADD_I block is as follows:

$$IN1 + IN2 = OUT \tag{1.8}$$

The "SCALE_X" instruction is another function that belongs to the instructions for manipulating numerical data. It lets to scale to floating-point number within value range. Programmer imposes the maximum and minimum value of the range and the scale block provides the result as an integer (figure 1.63). The formula that is applied is as follows:

$$OUT = [VALUE * (MAX - MIN)] + MIN \tag{1.9}$$



(Fig. 1.63 SCALE_X function in ladder diagram )

Where user has set a range of values from 0 to 100, so the block will scale the value of "Parameter_3" within this range and it will impose the result found at "Parameter_2". Particularly, the equation 1.9 becomes:

$$Parameter\_2 = [Paramter\_3 * (100 - 0)] + 0 \qquad (1.10)$$

In conclusion, the manipulation of numerical data lets to perform very simple operations such as sum, multiplication and complex as mapping of given in a range of values

## *1.11 How PLC works*

The programmer implements the set of instructions that the PLC must perform by means of compilers. They are PC software that convert program from a high-level language to a machine-readable and processor-executable language. Particularly, the largest PLC companies, such as Siemens, Rockwell and Schneider, have a PC software that permits the programmer to write and convert the program easily. For example, the manufacturer Siemens provides its customers with software such as T.I.A. Portal, STEP7 and STARTER for writing a program dedicated to Siemens CPUs and Siemens drive; while the management of Rockwell PLC must be done by means of programs created with RsLogix500 and RsLogix5000; finally, software Zelio Soft implements programs for Schneider PLC.

Moreover, the connection between PC and PLC must be done through these software and, specifically, they let to download the program in the PLC and to test it during the implementation phase. The connection between PC and PLC is via a normal cable such as the USB cable or Ethernet cable. After the PC and the PLC are physically connected by a cable, you can make the virtual connection and load the program. The program can be loaded only when there is a connection between PC and PLC or when PC is in online mode.

In the first CPUs as the Siemens S7-300 CPU, loading the program into the PLC could only happen when the CPU was in "STOP" mode, but now in the S7-1200 and S7-1500 CPUs the program can be loaded even when they are in "RUN" mode. It is possible to start the test phase after loading the program. During this phase all operations can be followed and analyzed directly by the PC, because PLC and PC are in the ON-line mode. In case the user needs to make changes, the programmer can only do them in OFF-line mode.

Generally, PLC has RUN and STOP states. When the PLC is in the STOP mode, it is on standby, which permits the program to be downloaded to the PLC. In this stage, all PLC outputs are disabled and therefore it is not possible to change the status of any actuator. The PLC remains in STOP state until a RUN command is applied.

During RUN mode the PLC runs the program and manages the outputs by reading the input states.

This mode can be changed manually by the user or automatically when an error occurs.

To conclude, the standard operation of the PLC depends on programming languages, but the way to implement and the tools for loading the program in the PLC are constrained by companies.

## *1.12 Communication: Profinet network*

The core components of a PLC, such as CPUs and drives, are connected to each other through an Ethernet cable. Communication between the various devices travels within the ethernet cable that creates a network called Profinet. The Profinet network is a leading Industrial Ethernet standard [Document No. 11 bibliography]. It permits high-speed communication and high reliability of controlled processes, which are indispensable for automation.

The Profinet network is useful for the implementation of both Industry 3.0 and Industry 4.0, because it is able to manage vertical and horizontal integration.

It can handle TCP/IP (Transport Communication Protocol/Internet Protocol) without limitation, for instance it allows the integration of PROFIBUS networks and other field bus systems, such as AS-Interface.

Security is one of the most important parameters for a company and the network on which the information travels must be able to guarantee it. The Profinet network has a high level of security because it uses the PROFIsafe profile. The PROFIsafe profile is able to execute standard and safety communication in a single bus cable. In other words, the user is able to monitor all the parameters and information of the Profinet network at any time through internet access.

The PROFIsafe standard is certified in according to “IEC 61508 (up to SIL 3), IEC62061 (up to Sil 3), en ISO 13849-1:2006 (up to PL e), EN 954 (up to category 4), NFPA 79-2002, NFPA 85 as well as for use up to SIL 3 according to EN 62061".

The installation of a profinet-type network is flexible, because it can be applied in the case of a linear, star, shaft and ring network.

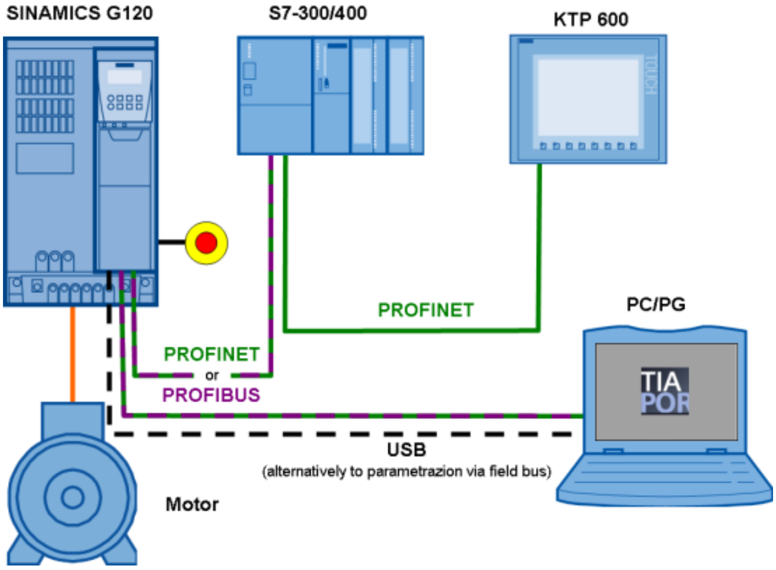
The construction of the Profinet network is based on "Switching" technology with a data transmission speed of 100 Mbit/s, which means that the data can travel in real time even in the case of complex systems.

The application of "Switching" technology permits a free connection with the next switch and therefore each node/partner can send or receive data in every instant. Additionally, devices can send and receive data at the same time because the Profinet network bandwidth this 200 Mbit/s. In this way Profinet allows the application of the Industry 4.0 in a simple and fast way.

Its flexibility also concerns the use of standardized protocols: it can manage the standardized protocols TCP/IP and UDP/IP. In addition, the network profinet uses Simple Network Management Protocol (SNMP) that reads and writes data. In this way, user is able to read and write data in a simple and secure way.

The task of a PLC is to manage actuators as a function of inputs; so PROFIBUS & Profinet International has created the Profinet IO standard. This standard allows a fast and correct communication between the various devices even in a decentralized architecture. It is based on cyclic communication of all data from each device, supporting 1440 bytes/telegram for each field device.

In conclusions, the qualities of the Profinet network permit a fast and secure communication of data and make it better than any field bus tradition.



(Fig. 1.64 Example of PLC control system connection )

## *1.13 Legislation*

PLCs are subject to the International Standard IEC 61131 defined by the International Electrotechnical Commission (IEC) [Website No. 1 sitography]. The IEC is a world organization composed of the National Electrotechnical Committees which aims to define the rules in the electrical and electronic field. These rules are accepted and followed by the international community. Generally, the rules defined by the IEC have an international consensus because they are chosen in agreement with all the national technical committees, therefore it collaborates with the International Organization for Standardization (ISO).

The standard IEC 61131 is made up of 8 parts, which describe every aspect of the PLC and are constantly updated to adapt the legislation to technological development [Document No. 2 bibliography].

The first part provides the definitions of programmable controllers and all peripheral devices such as human-machine interface (HMI), sensors and drives, specifying their main characteristics. In 1992 IEC defined the first edition of IEC 61131-1, which was replaced by the current norm in 2003.

The second part of the IEC 61131 standard was created in 1992, but it was updated in 2003 and in 2007. The current version specifies all tests for programmable controllers and peripherals and the minimum requirements of the PLC and its peripherals. Indeed, it defines the characteristics in the field of functional, electrical, mechanical, environmental, construction, safety, EMC and user programming.

The norm provides constraints on the use of the syntax and semantics of the most common programming languages. In other words, this third part imposes the basic functions that it is possible to use. As a consequence, if manufacturers want to create additional blocks, they should only be based on the basic functions provided by the standard.

In addition, in IEC/TR 61131-4 there is an end user guide. This guide explains to the users how to read the IEC 61131 series and assists them in the choice of devices based on the characteristics of the equipment specified in the norm.

The communication, however, is defined in the fifth part of the a forementioned legislation. The first draft of IEC 61131-5 was written in 2000 and it was updated in 2011. It imposes the type of network to be used for the exchange of information between PLCs and all the devices connected to it. In particular way, this part specifies the way in which PC must interface with PLC to provide information from the application program or services of other devices.

As we have already announced, security is a crucial parameter for PLCs, indeed the International Electrotechnical Commission has dedicated the sixth part of the norm to security. It describes all requirements that programmable controllers and its peripherals must have to be considered suitable for use in a security system E/E/PE. In other words, if PLC complies with IEC 61131-6, it is considered a programmable functional safety logic controller (FS-PLC). FS-PLC constitutes of hardware subsystem or a software subsystem: there are predefined functional blocks in the program that monitor security, generally.

The last two parts of the a forementioned legislation concern Fuzzy control and application of programming languages. Specifically, IEC 61131-7 specifies the type of programming language that must be applied for the implementation of a Fuzzy controller, while the IEC/TR 61131-8 provides a guide for the program implementation that PLC must perform. This guide takes into account the programming languages specified in the third part of the standard in the programming support environments (PES) in writing phase and in the PLC's in run phase.

So, norm IEC 61131 is the standard that all companies of international community have to observe.

# 2. The CPU and drive communication

## INTRODUCTION

In this second chapter, the communication between CPU and drive will be analyzed according to two main PLC manufacturers and program implemented by me.

First paragraph will describe drive types, drive use and main parts that constitutes drive. In addition, there will be how drive communicates with CPU and vice versa.

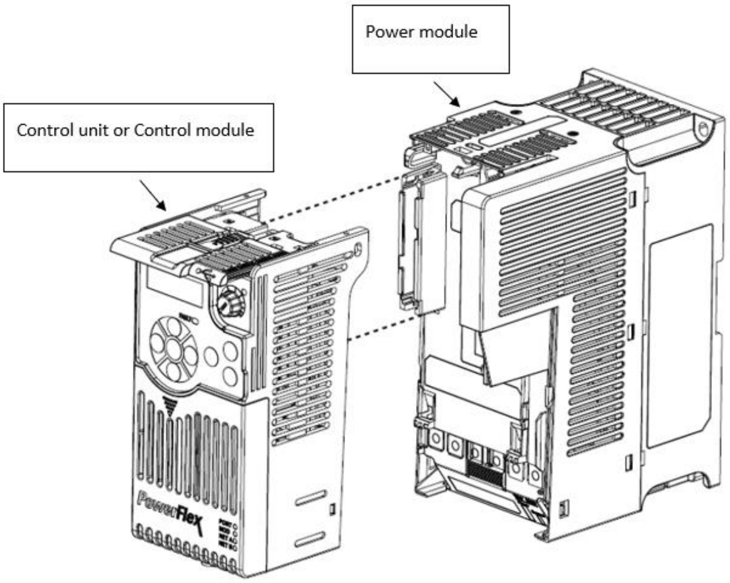
In paragraph 2.2, communication between Siemens CPU and Siemens drive will be analyzed in detail. Particularly, after a brief introduction, the paragraph is divided into 3 subparagraphs: SINA_POS block, SINA_SPEED block and Siemens Telegram 352. First subparagraph will analyze Siemens communication block for a position control and telegram used, while second subparagraph will study Siemens communication block for speed control and communication telegram, finally, third subparagraph will examine Siemens Telegram 352 for a speed control.

Paragraph 2.3 will explain communication between Rockwell Automation CPU and Rockwell Automation drive. Specifically, communication structure of inverter (PowerFlex series) drive will be describe in subparagraph 2.3.1, while the communication structure of a servo drive (Kinetix series) will be studied in subparagraph 2.3.2.

Finally, the last paragraph will analyze technique that I use to implement my program. Input and output signals will be examined and main functions will be described and analyzed in detail. Paragraph 2.4 concludes with an analysis about advantages and disadvantages of my control interface.

## *2.1 Drive types and communication structure*

The control of a process characterized by electric motors is performed by means of a drive. Drives consist of 2 parts: the “Control Unit” (CU) or “Control Module” (CM) and the “Power Module” (PM). CU is a processor that filters information from the CPU, while the power module is the power part that feeds the engine by adjusting the input power.

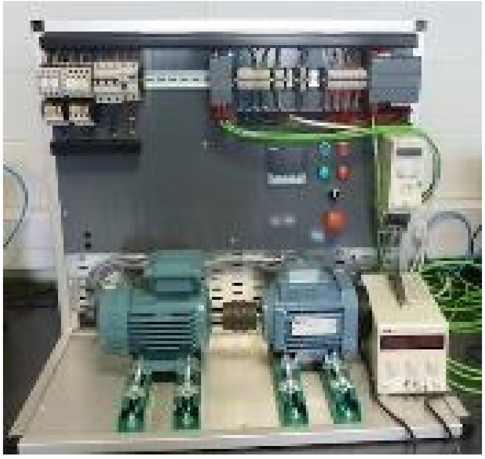


(Fig. 2.1 Disassembling a PowerFlex 525 drive)

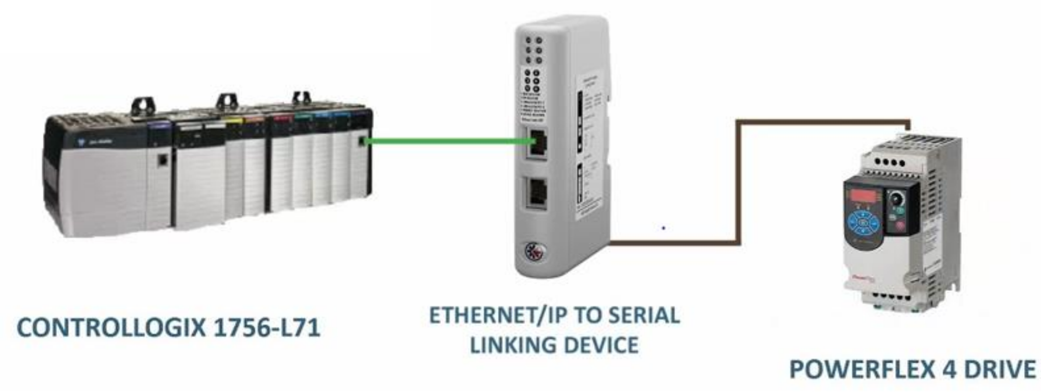
Drives can be classified into two households: inverters and servo drives (positioner). The inverters allow the control of the speed, while the servo drives allow the control of the position. Specifically, the inverters impose only the desired speed, instead the servo drives perform a closed loop by means of an internal encoder that provides the current position.

In other words, the difference between inverter and servo drive is the presence of an internal encoder.

The position control can also be done with an external inverter and encoder. In this case, the PID control uses the current position provided by the encoder to control acceleration, speed and position.



(Fig. 2.2 Application of Siemens CPU and drive for motor control)



(Fig. 2.3 Example of application of Rockwell Automation CPU and drive)

The communication between these CPUs and drives depends on the manufacturer. Indeed, every manufacturer has a defined structure that binds the user to the use of their components. For example, the communication between CPU and Siemens drive comes through the structures called telegrams (appendix A), while the communication between the CPU and the Rockwell drive comes through the use of data types defined by the Rockwell (appendix B). In addition, the position control through a CPU and a Siemens drive is achieved through standardized Siemens blocks that cannot be applied in the Rockwell software and for Rockwell devices. Sure enough, although it is possible that a Siemens CPU communicates with a Rockwell drive and vice versa, but this activity remains quite difficult because it is not possible to use standardized Siemens or Rockwell blocks. As a consequence, it is necessary to construct an ad hoc software block.

So, the communication between CPU and drive is a key factor for the control of production processes in the industrial field.

## *2.2 Communication between Siemens CPU and Siemens drive*

The communication between Siemens CPU and Siemens drive is managed directly through predefined telegrams [Document No. 14 bibliography]. Telegrams are data structures. They have been pre-set by Siemens and the programmer is bound to them. These structures represent a series of word and each word has a precise address. In other words, the commands provided by the CPU are written in these data packets, which are sent to the drive. Then, the drive split these data packets and activates the functions based on the transmitted value. The data packets consist of “Words”. There are several Siemens telegrams, each of which has its own internal structure. The internal structure depends on the amount and type of information that is communicated.

Siemens provides several telegrams for communication between CPU and drive; The most common are: Standard Telegram 1, Standard Telegram 20, Siemens Telegram 350, Siemens Telegram 352, Siemens Telegram 353 and Siemens Telegram 354 [Document No. 12 bibliography].

The choice of the type of telegram that is used depends on the purpose of the control. For example, if a speed control is needed, a Standard Telegram 1 or a Siemens Telegram 352 is used, and if a position control is required, the programmer uses the Standard Telegram 111.

The choice of telegrams is driven by Siemens, indeed the company has designed standardized blocks Siemens that allow programmer to manage control position and control speed. In other words, these blocks allow the self-programming of Siemens PLC.

These blocks respect the standard on PLCs because they are designed through the proper functions of programming languages, but can be used only with Siemens products (programming software, CPU, drive, etc.).

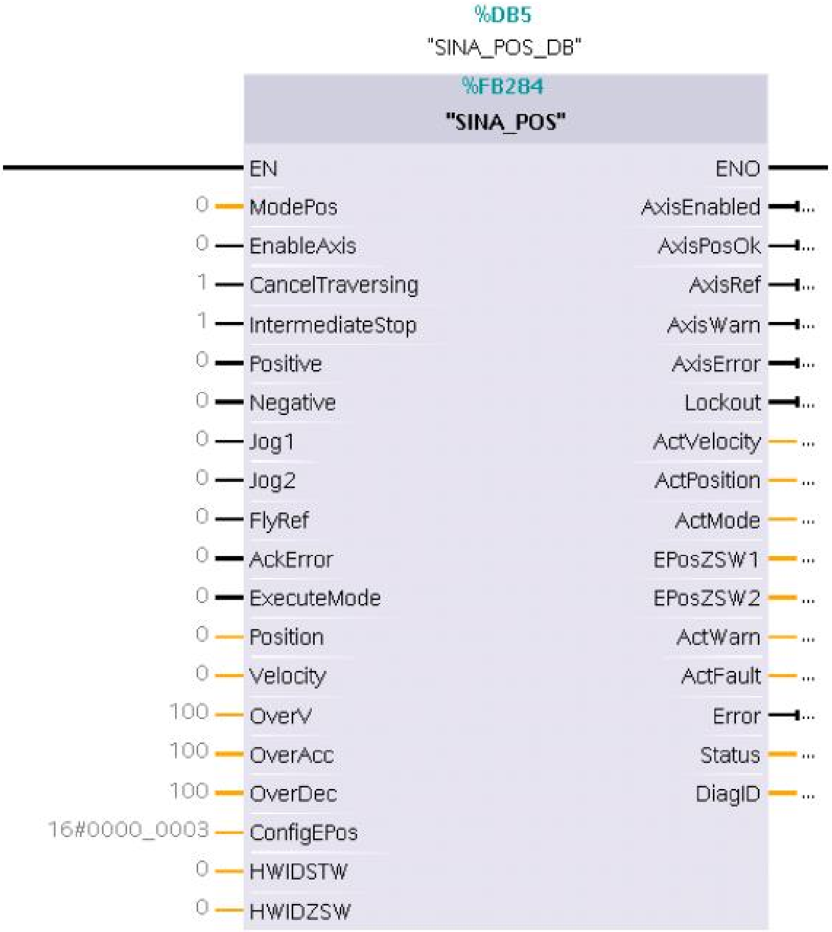
The main blocks are SINA_POS and the SINA_SPEED.

The SINA_POS is a block that allows the control of the drive position, while the SINA_SPEED allows the speed control.

### 2.2.1 SINA_POS block

The SINA_POS block is a standardized block for drive position command. It commands servo drives like S110 and S120. SINA_POS manages the communication between CPU and drive and the programmer must provide the right addressing for the connection with the buttons of the HMI. The communication occurs through a predefined structure: Standard Telegram 111. The Standard Telegram 111 contains all the bits that must be enabled for control implementation and they cannot be changed. As a result, the programmer and the user are bound to use of this structure.

The recall of the SINA_POS block is done through the call function [Document No. 14 bibliography]. The programmer calls function block FB 284. The graphical interface of this block is represented in figure 2.4.



(Fig. 2.4 Siemens SINA_POS block interface)

The SINA_POS block has a series of inputs and a series of outputs: the inputs are the commands that the programmer or directly the user wants to apply, while the outputs are the states of the drive that the user can read.

Specifically, the number of inputs is 19 and the number of outputs is 16. Inputs and outputs signals are different type because the setting of the drive requires different types of data (tables 2.1, 2.2).

| Input signal | Type | Default value | Meaning |
|---|---|---|---|
| ModePos | INT | 0 | Operation modes. |
| EnableAxis | BOOL | 0 | Activation command. |
| CancelTravercing | BOOL | 1 | It does not reject active traversing task. |
| IntermediateStop | BOOL | 1 | It does not active traversing command is interrupted. |

| | | | |
|---|---|---|---|
| Positive | BOOL | 0 | It sets the positive direction of movement |
| Negative | BOOL | 0 | It sets the negative direction of movement |
| Jog1 | BOOL | 0 | When the mode is jog, it performs a movement in the negative direction. |
| Jog2 | BOOL | 0 | When the mode is jog, it performs a movement in the positive direction. |
| FlyRef | BOOL | 0 | It sets flying referencing. |
| AckError | BOOL | 0 | When an unidentified error from the PLC occurs, it allows the user to block the activity. |
| ExecuteMode | BOOL | 0 | Depending on the mode of operation set, it activates traversing task, setpoint accepts e activates reference function. |
| Position | DINT | 0 [LU] | It sets the position that the user wants. |
| Velocity | DINT | 0 [LU/min] | It sets the maximum speed of the engine. |
| OverV | INT | 100 [%] | Speed ovveride. |
| OverAcc | INT | 100 [%] | Acceleration ovveride. |
| OverDec | INT | 100 [%] | Deceleration ovveride. |
| ConfigEpos | DWORD | 3h | This array value is imposed by Siemens company. |
| HWIDSTW | HW_IO | 0 | Drive's state address. |
| HWIDZSW | HW_IO | 0 | Drive's command address. |

(Table 2.1 Input signals of SINA_POS block)

| Output signal | Type | Default value | Meaning |
|---|---|---|---|
| AxisEnabled | BOOL | 0 | Drive is on and ready. |
| AxisPos | BOOL | 0 | Target position is reached. |
| AxisRef | BOOL | 0 | Reference point is setted. |
| AxisWarn | BOOL | 0 | There is drive alarm. |
| AxisError | BOOL | 0 | There is drive fault. |
| Lockout | BOOL | 0 | Switching-on inhibit. |
| ActVelocity | DINT | 0 | Actual velocity. |

| | | | |
|---|---|---|---|
| ActPosition | DINT | 0 [LU] | Actual position. |
| ActMode | INT | 0 | Currently active mode. |
| EPosZSW1 | WORD | 0 | Status of EPos ZSW1. |
| EPosZSW2 | WORD | 0 | Status of EPos ZSW2. |
| ActWarn | WORD | 0 | Actual alarm number. |
| ActFault | WORD | 0 | Actual fault number. |
| Error | BOOL | 0 | Presence of error in drive. |
| Status | INT | 0 | Actual status. |
| DiagID | WORD | 0 | Presence of error during FB call. |

(Table 2.2 Output signals of SINA_POS block)

The SINA_POS for the management of the position allows the programmer to operate in 8 different modes on the drive.

The 8 operating modes are:

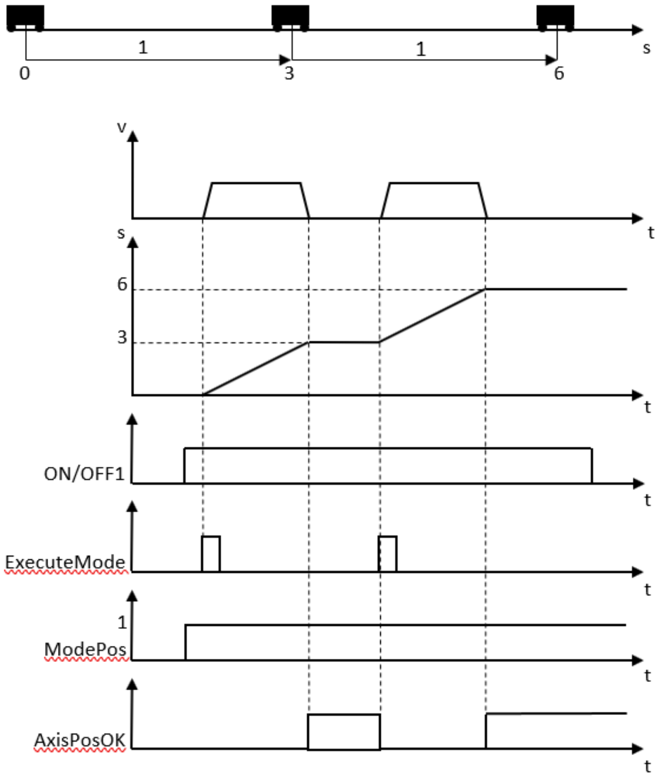
- “Relative positioning”;
- “Absolute positioning”;
- “Setup mode”;
- “Reference point approach”;
- “Set reference point”;
- “Traversing block”;
- “Jog mode”;
- “Incremental jogging”.

In the positioning mode, the position of the motor rotor changes by taking the current position as the initial point. In other words, when the user activates the enablement, the motor rotor switches from position 0 to position 1 and the new position will be set as position 0 automatically.

Activating the positioning mode requires the activation of following inputs: Modepos = 1 and EnableAxis = 1. This mode activates "MDI relative positioning" drive function.

The velocity is another parameter that user must set through the "velocity" input. Instead, it is not necessary to set the direction, because it always remains positive and therefore the movement in negative direction is not possible.

Figure 2.5 shows how the relative positioning mode works. After setting the Modepos = 1 and EnableAxis = 1 parameters, the rotor is ready to perform relative positioning mode. If the motor moves a carriage and position 1 is 3 meters from position 0, when the user sets the position 1 and starts the movement, the trolley will run a path of 3 meters. This new location will be identified as position 0. Then, if the user starts the movement by setting position 1, the rotor will rotate and the trolley will run another 3 m. So, the final position is 6 m relative to the initial position (figure 2.5).



(Fig. 2.5 Timeline of relative positioning mode)

The second mode of operation of the block is absolute positioning.

The "MDI absolute positioning" drive function is activated when absolute positioning mode must be performed.

In the absolute positioning mode, the position is always calculated relative to position 0.

Activation of the absolute Positiong mode requires activation of specific inputs: Modepos = 2 and EnableAxis = 1.

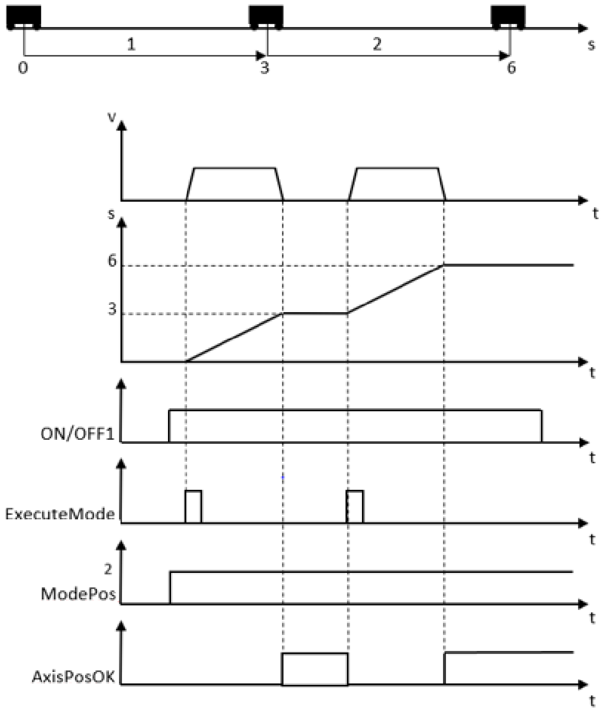
Also, in this case, the programmer or user sets the speed through the velocity input. In this case, the movement direction can be either automatically set or forced by the user. When absolute positioning mode is performed and the direction is not forced, it depends on the distance between the current and the desired position. In this case the positive and negative inputs remain at zero.

If the user forces the direction, the movement of the motor will follow the direction set. The direction setting makes happen through the "Positive" input if the rotor has to go forward, "Negative" if the rotor has to go backwards.

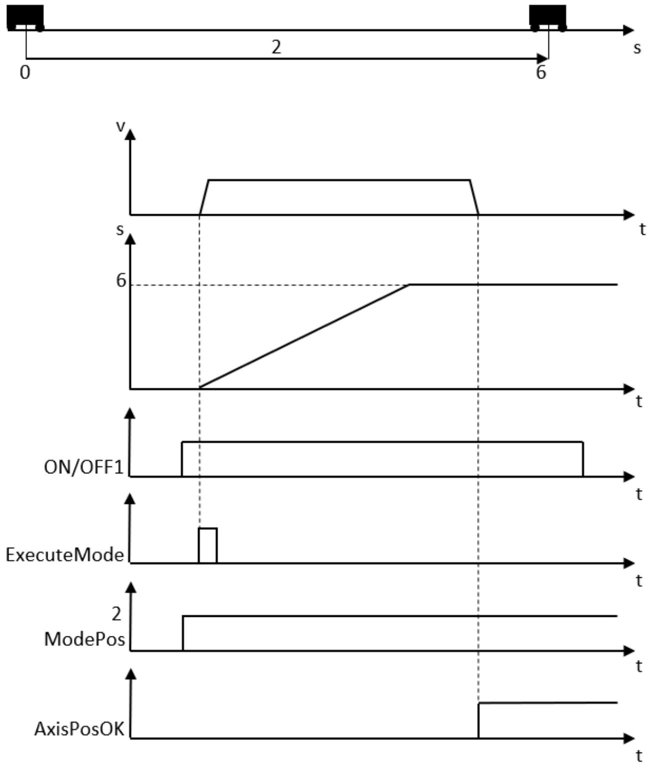
The absolute positioning mode operation is represented in figures 2.6 and 2.7.

After setting the Modepos = 2 and EnableAxis = 1 parameters, the rotor is ready to run the absolute positioning mode. If the motor moves a trolley and position 1 is 3 m from position 0, when the user sets the position 1 and starts the movement, the trolley will run a path of 3 m. If the user starts the movement by setting position 1, the rotor will not turn and the trolley will remain stationary. This position is 3 m relative to the initial position (figure 2.6).

If the user wants the distance between position 0 and end position to be 6 m, he must set another position: “Position 2” = 6 m. Thus, when applying position 2, the trolley will be 6 meters away from position 0, irrespective of position 1 (figures 2.6 and 2.7).



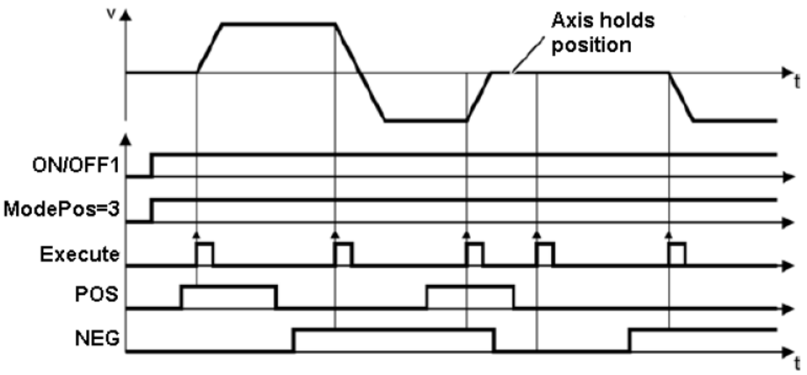
(Fig. 2.6 Timeline of absolute positioning mode (1° example))



(Fig. 2.7 Timeline of absolute positioning mode (2° example))

Absolute positioning and relative positioning modes are indispensable for controlling the position of an electric motor.

The third mode of operation is the Setup mode. It allows the position-controlled traversing of the axis without target position. The user must force the direction. The direction setting makes happen as for absolute positioning: forcing the "Positive" input if the rotor has to go forward or forcing the "Negative" input if the rotor has to go backwards. It uses the "MDI set up" mode drive function. To perform this mode, the user must set following inputs: “velocity”, “Modepos = 3” and “EnableAxis = 1”.

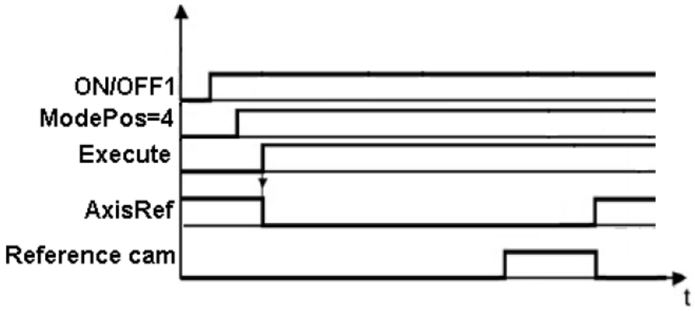


(Fig. 2.8 Timeline of setup mode)

It should be noted that, when positive input is active, the speed is positive and the rotor turns forward, while when negative input is active the speed is negative and the rotor rotates backwards (figure 2.8).

The Reference point approach mode is the fourth mode of operation of the engine. This mode activates the “active referencing” drive function. Both the speed and the direction are set by the user. Particularly, the speed is set by means of the “velocity” input and the direction is set through negative and positive input.

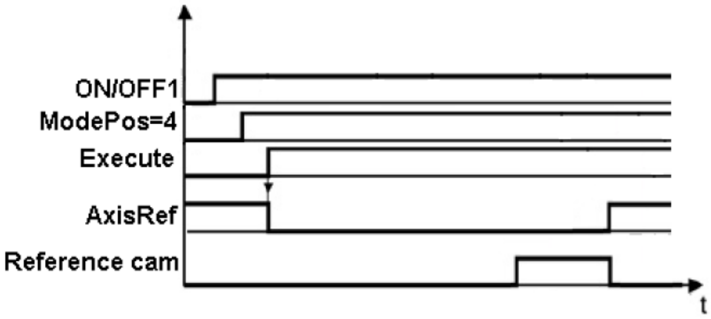
Its activation requires the setting of the inputs Modepos = 3 and EnableAxis = 1; The axis must also be stationary (figure 2.9).



(Fig. 2.9 Timeline of reference point approach mode)

Set reference point mode, on the other hand, enables axis referencing. It defines the current position as the starting position. Therefore, it calculates all subsequent shifts depending on this position. The function of the drive used is called “Set reference point”.

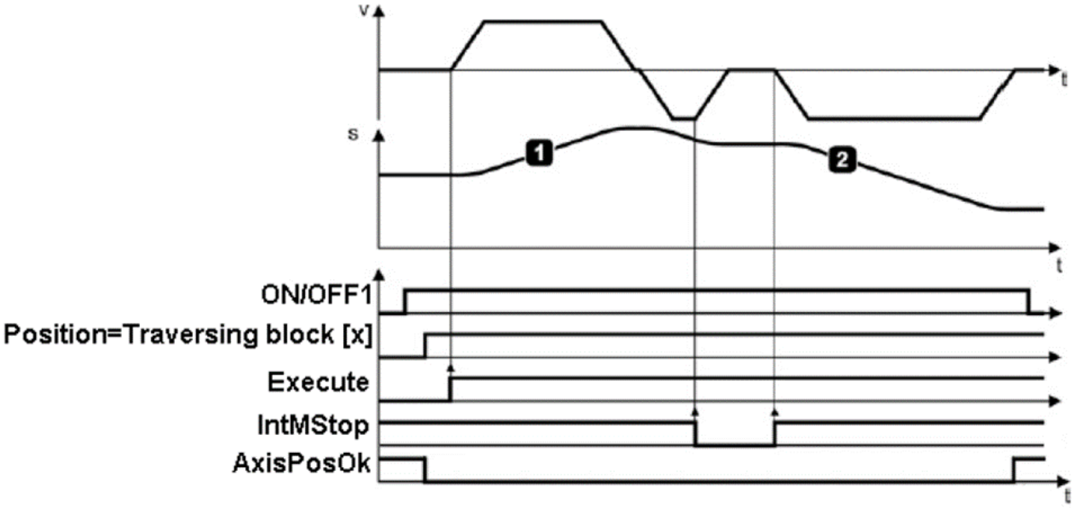
Its activation requires the stationary axis and the setting of the inputs Modepos = 5 and EnableAxis = 1. Indeed, the drive will consider the current position like zero set point (figure 2.10).



(Fig. 2.10 Timeline of zero set point mode)

The sixth mode of operation is traversing block. It generates programs in an automatic way that allows the stop of the movements and the set and reset of the commands. It is performed by means of the “traversing block” drive function.

The user must set the inputs Modepos = 6 and EnableAxis = 1 to implement this task, and the axis must also be stationary (figure 2.11).



(Fig. 2.11 Timeline of traversing block mode)

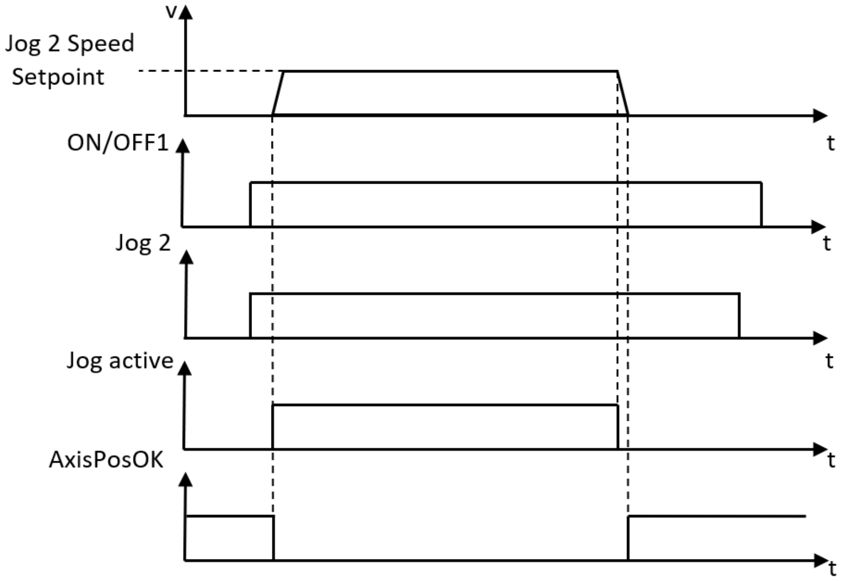
Jog mode, on the other hand, allows the control of the position manually and directly. The speed of motion does not depend on the speed provided as input, but drive computes the speed automatically based on the maximum speed the engine can reach. This mode is performed using the "Jog" drive function.

Its operation is described in the following figure.

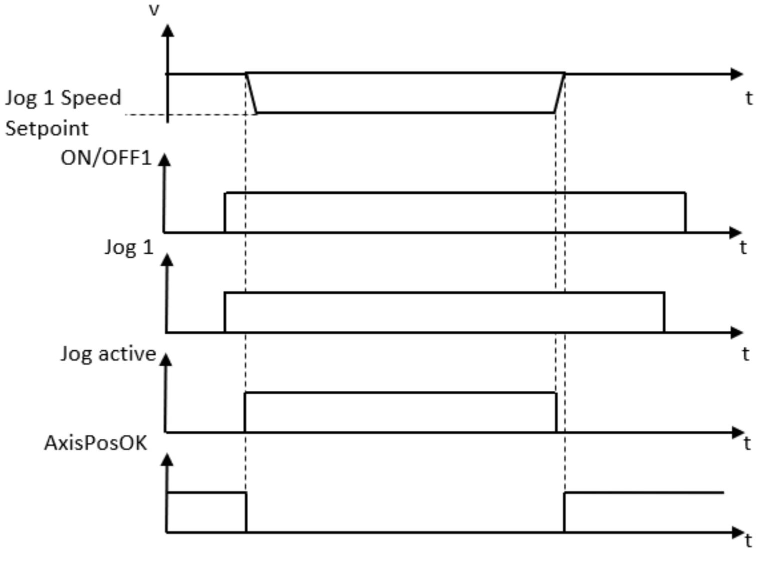
If the user needs jog mode, he must set the inputs Modepos = 7 and EnableAxis = 1 and the axis must also be stationary.

After setting the Modepos = 7 and EnableAxis = 1 parameters, the rotor is ready to run the jog mode. When the user sets Jog2 input equal to 1, the rotor will move in a positive direction until the “jog active” input is active (figure 2.12).

Instead, if the user sets Jog1 input equal to 1, the rotor will move in a negative direction until the jog active remains active (figure 2.13).

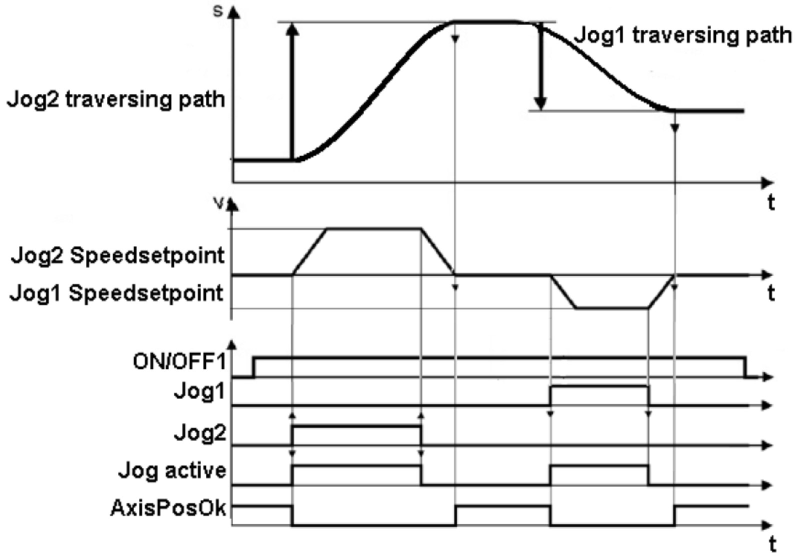


(Fig. 2.12 Timeline of Jog2 mode)



(Fig. 2.13 Timeline of Jog1 mode)

The incremental jogging mode enables the position-controlled and distance-dependent traversing of axes. As in the previous case, the speed of movement does not depend on the speed provided as input, but is calculated automatically based on the maximum speed that the engine can reach. In addiction in a manner similar to jog mode, it is performed by means of the "Jog" drive function. The requirements of this mode are Modepos = 8 and EnableAxis = 1 and standstill axis. The operation of the incremental jog mode is the same as the jog mode (figure 2.14).



(Fig. 2.14 Timeline of incremental jogging mode)

The operating modes can be activated if and only if "canceltraversing" and "intermediatestop" inputs have a value of 1. Specifically, "canceltraversing", it manages the activation and the deactivation of the deceleration ramp: if it is at 1, the deceleration ramp is at 0%; on the contrary the deceleration ramp is at 100% if the parameter "canceltraversing" is equal to 0. A deceleration ramp at 100% means that the movement speed tends to 0.

The “intermediatestop” input allows the user to stop the drive immediately when it is equal to 0 and allows movement when it is equal to 1.

As we have previously highlighted, another crucial activity is the setting of position and speed. The “Position” and “velocity” inputs belong to the double integer data type (DInt) and can store a range of values from (-2)^31 to 2^((31-1)).

The unit of measurement used is the Siemens engineering unit named LU. Specifically, “LU” is the position unit of measurement and “LU/min” is that of the velocity. There is no standardized conversion, but it is set by the programmer within the list of drive experts during the programming phase. The drive experts list is a list in which the programmer can modify the drive internal parameters. These parameters can be modified only by the programmer in commissioning phase. During this phase the programmer writes the engine's technical characteristics in the drive, the value of the PID controller coefficients and the ratio of “LU” to “rpm”. The programmer chooses this report without any specific constraint. In addition, he must set the current, the voltage, the maximum speed and the motor frequency. In this way the drive can recognize the motor connected to it and manage it correctly.

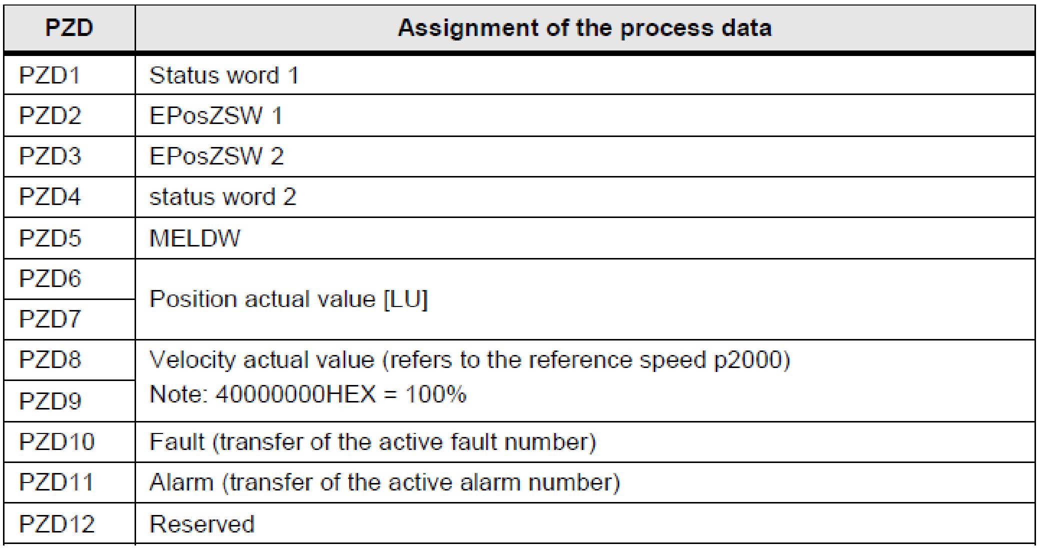
The speed that must be entered is the maximum speed that the drive can reach in LU, while the desired speed is managed by the speed override. It varies from 0% to 100%. For example, if the speed override is 0% the drive will be standstill, if it is at 50%, the drive speed is equal to half of the maximum speed, of course if the drive has to move at maximum speed the user is going to turn the speed override to 100%.

Acceleration and deceleration are set by their overrides. Their default values are 100%, but they can be changed according to customer's needs.

This block allows the control of the position desired by the user starting from two types of input: those managed by the user and the states of the drive.

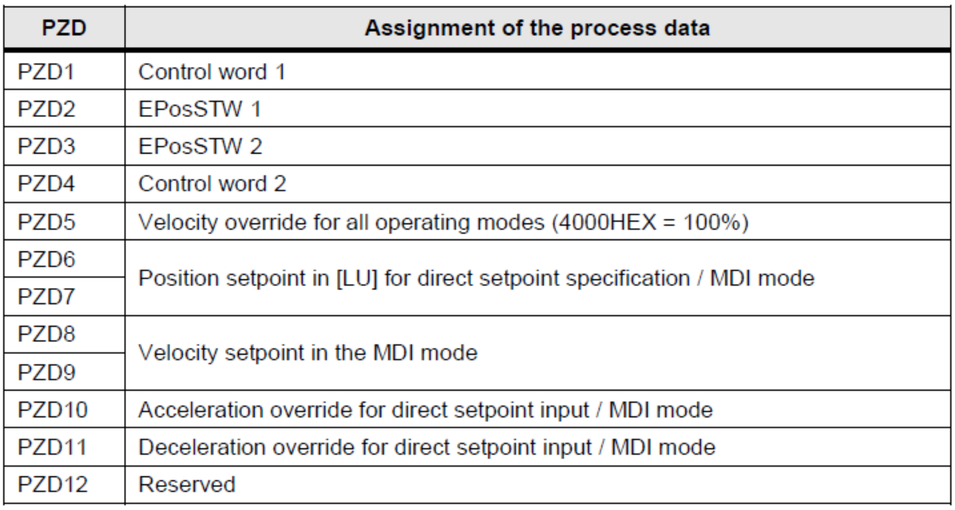
The states of the drive are obtained through the Standard Telegram 111. The use of this preset block only allows the use of the Standard Telegram 111: other telegrams cannot be used because errors would be created during compile phase and the program would not be loaded into CPU memory.

The structure of the Standard Telegram 111 constitutes of 24 words grouped in two parts: status words (figure 2.15) and command words (figure 2.16).



| PZD | Assignment of the process data |
|---|---|
| PZD1 | Status word 1 |
| PZD2 | EPosZSW 1 |
| PZD3 | EPosZSW 2 |
| PZD4 | status word 2 |
| PZD5 | MELDW |
| PZD6 | Position actual value [LU] |
| PZD7 | |
| PZD8 | Velocity actual value (refers to the reference speed p2000)<br>Note: 40000000HEX = 100% |
| PZD9 | |
| PZD10 | Fault (transfer of the active fault number) |
| PZD11 | Alarm (transfer of the active alarm number) |
| PZD12 | Reserved |

(Fig. 2.15 Status word list of Standard telegram 111)



| PZD | Assignment of the process data |
|---|---|
| PZD1 | Control word 1 |
| PZD2 | EPosSTW 1 |
| PZD3 | EPosSTW 2 |
| PZD4 | Control word 2 |
| PZD5 | Velocity override for all operating modes (4000HEX = 100%) |
| PZD6 | Position setpoint in [LU] for direct setpoint specification / MDI mode |
| PZD7 | |
| PZD8 | Velocity setpoint in the MDI mode |
| PZD9 | |
| PZD10 | Acceleration override for direct setpoint input / MDI mode |
| PZD11 | Deceleration override for direct setpoint input / MDI mode |
| PZD12 | Reserved |

(Fig. 2.16 Command word list of Standard telegram 111)

Particularly, status words are data that would let to identify the drive states. The block reads the states desired by the user and the states of the drive and it processes the right command for them to coincide. In particular the block reads the state “Status_ready”, “Status_IOp” and “Status_fault” to start the activities. The conditions required to run the control are:

- “Status_ready = 1”;
- “Status_IOp = 1”;
- “Status_fault = 0”.

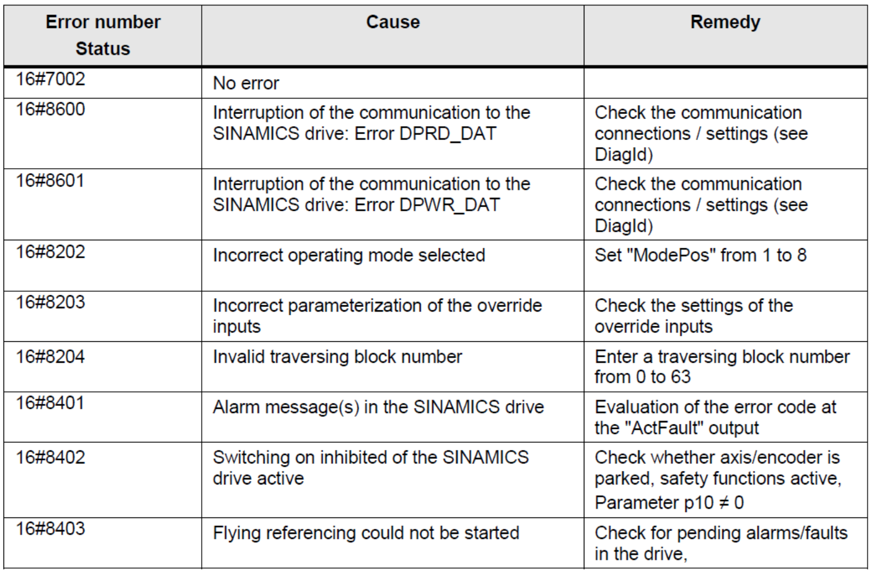
If the condition of the drive state is correct, the block reads user inputs and processes the commands. The sending of the commands is done by means of the 12 command words (appendix A). In other words, the block imposes a specific value on 12 words and sends them to the drive. Conversely, when the block reads the status words (appendix A), the drive imposes values on the word and sends it to the block. The reading and writing of the word is done through addressing. This procedure is analogous to the one described for the individual bits in chapter 1. The addressing of status word and command word status is performed separately: the programmer must set the addresses in two different inputs even if the write address is the same as the reading one. Specifically, the HWIDSTW input maintains the address for reading the states, while the HWIDZSW input sets the address for writing commands to the drive.

Status word (appendix A) shows that the commands sent from the drive to the block do not correspond to the user-visible block outputs (table 2.2). The outputs of the SINA_POS block are a limited number of states that the user is able to read. These data are all important to the user, but "status" output is critical. Indeed, it describes the state of the drive and the PLC, both in case they work correctly or in the presence of errors. The value obtained from this output is a number because the data type is integer. As a result, the user needs a manual to convert the numeric value to a specific state (table 2.3).

| Status | Meaning |
|---|---|
| 16#7002 | No fault – Correct execution. |
| 16#8401 | Drive fault. |
| 16#8402 | Switching-on inhibit. |
| 16#8403 | Flying referencing could not be started. |
| 16#8600 | Error DPRD_DAT. |
| 16#8601 | Error DPWR_DAT. |
| 16#8202 | Incorrect operating mode selected. |
| 16#8203 | Incorrect setpoints parameterized. |
| 16#8204 | Incorrect. |

(Table 2.3 Status code list of Standard telegram 111)

Conversely, if there is an error, "Actfault" output gives us the type of error present. Similarly, to the status it is an integer and therefore a conversion table is necessary (figure 2.17).



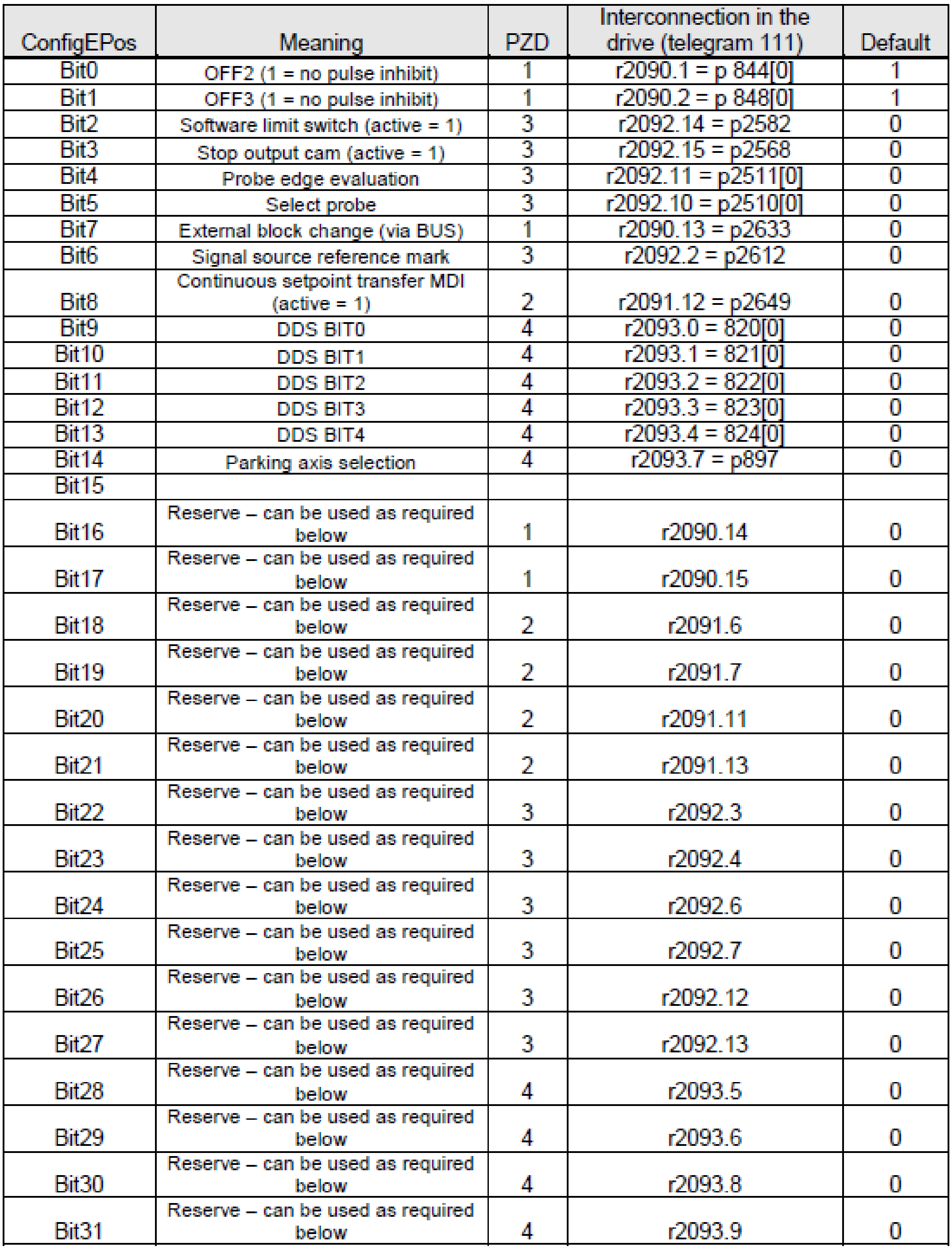
| Error number<br>Status | Cause | Remedy |
|---|---|---|
| 16#7002 | No error | |
| 16#8600 | Interruption of the communication to the SINAMICS drive: Error DPRD_DAT | Check the communication connections / settings (see DiagId) |
| 16#8601 | Interruption of the communication to the SINAMICS drive: Error DPWR_DAT | Check the communication connections / settings (see DiagId) |
| 16#8202 | Incorrect operating mode selected | Set "ModePos" from 1 to 8 |
| 16#8203 | Incorrect parameterization of the override inputs | Check the settings of the override inputs |
| 16#8204 | Invalid traversing block number | Enter a traversing block number from 0 to 63 |
| 16#8401 | Alarm message(s) in the SINAMICS drive | Evaluation of the error code at the "ActFault" output |
| 16#8402 | Switching on inhibited of the SINAMICS drive active | Check whether axis/encoder is parked, safety functions active, Parameter p10 ≠ 0 |
| 16#8403 | Flying referencing could not be started | Check for pending alarms/faults in the drive, |

(Fig. 2.17 Error code list of Standard telegram 111)

The user can read only the main states for proper operation of the control.

The main advantage of the SINA_POS is the ability to manage a control in place easily. Sure enough, the programmer does not have to implement complex functions but simply he has to set the block inputs.

On the contrary, the main disadvantage of using this block is the reduced flexibility. Indeed, the programmer cannot set single bit of drive structure manually, because drive structure bits are set by block based on user input. Furthermore the "ConfigEPos" input is a vector that the user or programmer cannot easily manage because it is set correctly by default (figure 2.18) and any of its modifications could block the communication between CPU and drive.



| ConfigEPos | Meaning | PZD | Interconnection in the drive (telegram 111) | Default |
|---|---|---|---|---|
| Bit0 | OFF2 (1 = no pulse inhibit) | 1 | r2090.1 = p 844[0] | 1 |
| Bit1 | OFF3 (1 = no pulse inhibit) | 1 | r2090.2 = p 848[0] | 1 |
| Bit2 | Software limit switch (active = 1) | 3 | r2092.14 = p2582 | 0 |
| Bit3 | Stop output cam (active = 1) | 3 | r2092.15 = p2568 | 0 |
| Bit4 | Probe edge evaluation | 3 | r2092.11 = p2511[0] | 0 |
| Bit5 | Select probe | 3 | r2092.10 = p2510[0] | 0 |
| Bit7 | External block change (via BUS) | 1 | r2090.13 = p2633 | 0 |
| Bit6 | Signal source reference mark | 3 | r2092.2 = p2612 | 0 |
| Bit8 | Continuous setpoint transfer MDI (active = 1) | 2 | r2091.12 = p2649 | 0 |
| Bit9 | DDS BIT0 | 4 | r2093.0 = 820[0] | 0 |
| Bit10 | DDS BIT1 | 4 | r2093.1 = 821[0] | 0 |
| Bit11 | DDS BIT2 | 4 | r2093.2 = 822[0] | 0 |
| Bit12 | DDS BIT3 | 4 | r2093.3 = 823[0] | 0 |
| Bit13 | DDS BIT4 | 4 | r2093.4 = 824[0] | 0 |
| Bit14 | Parking axis selection | 4 | r2093.7 = p897 | 0 |
| Bit15 | | | | |
| Bit16 | Reserve – can be used as required below | 1 | r2090.14 | 0 |
| Bit17 | Reserve – can be used as required below | 1 | r2090.15 | 0 |
| Bit18 | Reserve – can be used as required below | 2 | r2091.6 | 0 |
| Bit19 | Reserve – can be used as required below | 2 | r2091.7 | 0 |
| Bit20 | Reserve – can be used as required below | 2 | r2091.11 | 0 |
| Bit21 | Reserve – can be used as required below | 2 | r2091.13 | 0 |
| Bit22 | Reserve – can be used as required below | 3 | r2092.3 | 0 |
| Bit23 | Reserve – can be used as required below | 3 | r2092.4 | 0 |
| Bit24 | Reserve – can be used as required below | 3 | r2092.6 | 0 |
| Bit25 | Reserve – can be used as required below | 3 | r2092.7 | 0 |
| Bit26 | Reserve – can be used as required below | 3 | r2092.12 | 0 |
| Bit27 | Reserve – can be used as required below | 3 | r2092.13 | 0 |
| Bit28 | Reserve – can be used as required below | 4 | r2093.5 | 0 |
| Bit29 | Reserve – can be used as required below | 4 | r2093.6 | 0 |
| Bit30 | Reserve – can be used as required below | 4 | r2093.8 | 0 |
| Bit31 | Reserve – can be used as required below | 4 | r2093.9 | 0 |

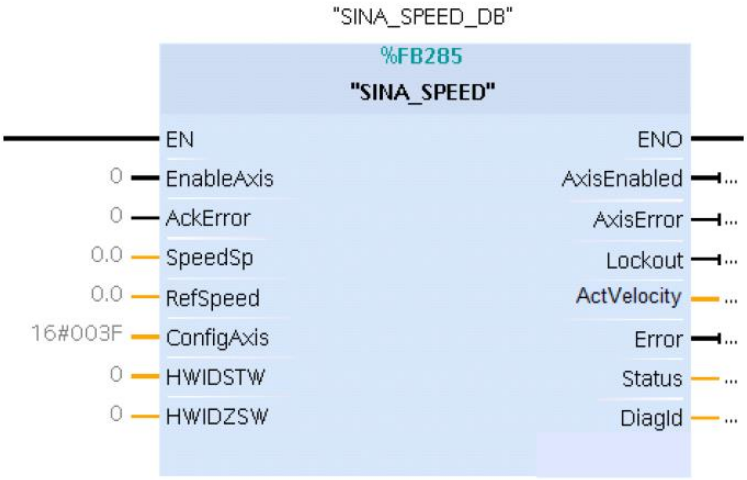
(Fig. 2.18 ConfigEPos structure of Standard telegram 111)

In conclusion, the SINA_POS is a block capable of controlling the position of an engine through servo drives such as S110 and S120. This block is a first example of PLC self-programming because the programmer must only set the block inputs without implementing particular functions, limiting the manipulation of the command bits.

### 2.2.2 SINA_SPEED block

The SINA_SPEED block is a standardized Siemens block for managing the speed of a drive. It controls inverters such as G110 and G120. This block manages the communication between the CPU and the inverter by means of a predefined structure: Standard Telegram 1. This structure contains the individual bits that must be enabled for speed control implementation and cannot be changed. As a result, the programmer and the user are bound by the use of this telegram. If a different telegram is used, a error would happen during compile phase, interrupting the programming of the PLC.

The recall of the SINA_SPEED block is done through the call function [Document No. 14 bibliography]. The programmer calls the SINA_SPEED block by recalling the function block FB285. The block graphical interface is represented by the following image.



(Fig. 2.19 Siemens SINA_SPEED block interface)

The SINA_SPEED block has a series of inputs and a series of outputs: the inputs are the commands that the programmer or user manages, while the outputs are the states of the drive that the user can read. From figure 2.19 it should be noted that the input parameters (table 2.4) and output (table 2.5) are lower than the SINA_POS block. This is because speed management is simpler than position management: to manage the position the user has to handle both position and velocity, while to manage the speed the position is not considered.

| Input signal | Type | Default value | Meaning |
|---|---|---|---|
| EnableAxis | BOOL | 0 | Activation command. |
| AckError | BOOL | 0 | When an unidentified error from the PLC occurs, it allows the user to block the activity. |
| SpeedSp | REAL | 0.0 [rpm] | Speed setpoint. |
| RefSpeed | REAL | 0.0 [rpm] | Rated speed of the drive. |
| ConfigAxis | WORD | 3 | Value imposed by Siemens. |
| HWIDSTW | HW_IO | 0 | Drive's states address. |
| HWIDZSW | HW_IO | 0 | Drive's command address. |

(Table 2.4 Input signals of SINA_SPEED block)

| Output signal | Type | Default value | Meaning |
|---|---|---|---|
| AxisEnabled | BOOL | 0 | Drive is on and ready. |
| Lockout | BOOL | 0 | Switching-on inhibit. |
| ActVelocity | REAL | 0.0 [rpm] | Actual velocity. |
| Error | BOOL | 0 | There is an error in the drive. |
| Status | INT | 0 | Actual status. |
| DiagID | WORD | 0 | Presence of error during FB call. |

(Table 2.5 Output signals of SINA_SPEED block)

The SINA_SPEED block has only one operating mode, so it does not have the “Modepos”. Drive starter occurs through the "EnableAxis" input, which initiates speed control. During the commissioning phase the programmer sets the technical characteristics of the motor. In particular, the maximum speed of the engine must be written in the "p2000" the list of drive experts. The drive calculates the desired speed by the user according to the p2000 parameter. For this reason, generally, the programmer manages the desired speed in percentage. For example, if the rotor has to go at a speed equal to 20% of the maximum engine speed, the programmer will set the following values:

- “SpeedSp” = 20;
- “Refspeed” = 100.

The code inside the block will process these inputs and will resize the drive speed to reach the desired one.

As a result, the SINA_SPEED block allows the user to control the desired speed starting from two types of inputs: user-managed and drive states.

The drive states are obtained through Standard Telegram 1. Using this preset block only allows the use of Standard Telegram 1: other telegrams cannot be used because errors would be created at compile time and the download to the CPU would not run.

The Standard Telegram 1 consists of two parts: status word (table 2.7) and command word (table 2.6). Each of these parts consists of 2 words: the first word is divided into single bits and the second word no (appendix A).

| PZD | Assigment of the process data |
|---|---|
| PZD 1 | Control word |
| PZD 2 (bits 16 to 32) | Speed setpoint |

(Table 2.6 Command word list of Standard Telegram 1)

| PZD | Assigment of the process data |
|---|---|
| PZD 1 | Status word |
| PZD 2 (bits 16 to 32) | Bits 16 – 31 → actual speed value |

(Table 2.7 Status word list of Standard Telegram 1)

As with the Standard Telegram 111, status words are data that would let to locate the drive states. The block reads the states desired by the user and the states of the drive and processes the right command for them to coincide. Particularly, the block reads the status "ready to start", "ready to operate" and "fault active" to start the activities. The conditions required to run the control are:

- “ready to start = 1”;
- “ready to operate = 1”;

- "fault active = 0".

The block produces commands that are sent to the drive through the Standard Telegram 1. If the condition of the drive state is correct, the block reads user inputs and it processes the commands. The sending of the commands is done by means of the 2 command words (appendix A). In other words, the block imposes a specific value on the 2 words and sends them to the drive. Conversely, when reading status word (appendix A), the drive imposes values at 2 words and sends them to the block.

In a manner similar to SINA_POS block, the addressing of word and command word status is performed separately: the programmer must set the addresses in two different inputs even if the two addresses are equal. Specifically, the HWIDSTW input maintains the address for reading the states, while the HWIDZSW input sets the address for writing commands to the drive. The programmer must fill in the correct addresses so that the communication happens correctly.

The reading of the drive states (appendix A) is very important, because it not only indicates the presence or absence of fault in the drive but also the values of speed over time. These data provide information about the drive that is useful not only to the block but also to the user. Sure enough, the block outputs that are read by the user are connected to the drive states. For example, "actvelocity" gives the user the current speed of the drive and "Error" is set if there are any errors.

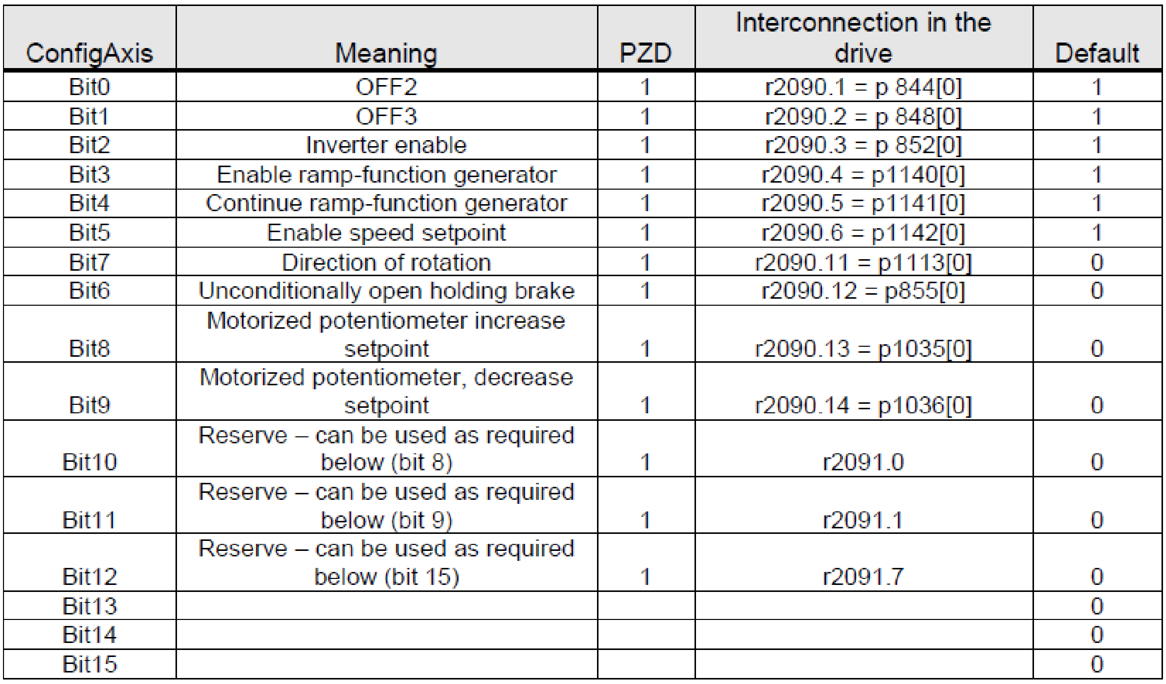
Specifically, "status" output describes the drive state and the PLC, both in case they function correctly and in the presence of errors. The value obtained from this output is a number because the data type is integer. As a result, the user needs a manual to convert the numeric value to a specific state (table 2.8).

| Error number status | Meaning | Remedy |
|---|---|---|
| 16#7002 | No fault active | |
| 16#8401 | Drive fault active | Evaluate active faults of the SINAMICS via the acyclic communication |
| 16#8402 | Drive switching on inhibited active | Check whether axis is parked, safety active, parameter p10 ≠ 0 |
| 16#8600<br>16#8601 | Error of the SFB call active | Correction of the communication fault |

(Table 2.8 Error code list of Standard Telegram 1)

The main advantage of the SINA_SPEED is the ability to manage a speed control easily. Sure enough, the programmer must set the block inputs without implementing complex functions.

On the contrary, the main disadvantage of using this block is the reduced flexibility. Indeed, the programmer cannot set single bit of drive structure manually, because drive structure bits are set by block based on user input. Furthermore the "Configaxis" input is a vector that the user or programmer cannot easily manage because it is set correctly by default (figure 2.20) and any of its modifications could interrupt the communication between CPU and drive.



| ConfigAxis | Meaning | PZD | Interconnection in the drive | Default |
|---|---|---|---|---|
| Bit0 | OFF2 | 1 | r2090.1 = p 844[0] | 1 |
| Bit1 | OFF3 | 1 | r2090.2 = p 848[0] | 1 |
| Bit2 | Inverter enable | 1 | r2090.3 = p 852[0] | 1 |
| Bit3 | Enable ramp-function generator | 1 | r2090.4 = p1140[0] | 1 |
| Bit4 | Continue ramp-function generator | 1 | r2090.5 = p1141[0] | 1 |
| Bit5 | Enable speed setpoint | 1 | r2090.6 = p1142[0] | 1 |
| Bit7 | Direction of rotation | 1 | r2090.11 = p1113[0] | 0 |
| Bit6 | Unconditionally open holding brake | 1 | r2090.12 = p855[0] | 0 |
| Bit8 | Motorized potentiometer increase setpoint | 1 | r2090.13 = p1035[0] | 0 |
| Bit9 | Motorized potentiometer, decrease setpoint | 1 | r2090.14 = p1036[0] | 0 |
| Bit10 | Reserve – can be used as required below (bit 8) | 1 | r2091.0 | 0 |
| Bit11 | Reserve – can be used as required below (bit 9) | 1 | r2091.1 | 0 |
| Bit12 | Reserve – can be used as required below (bit 15) | 1 | r2091.7 | 0 |
| Bit13 | | | | 0 |
| Bit14 | | | | 0 |
| Bit15 | | | | 0 |

(Fig. 2.20 ConfigAxis structure of Standard telegram 1)

In conclusion, the SINA_SPEED is a block capable of controlling an engine speed through inverters such as G110 and G120. This block is an example of self-programming of PLCs because the programmer does not implement particular functions, but he simply set the block inputs. The only disadvantage is the reduction of flexibility.

### 2.2.3 Siemens Telegram 352

The speed control can be carried out through the use of the Siemens Telegram 352 [Document No. 12 bibliography]. In this case it is not possible to use the SINA_SPEED block, as this block is closely linked to Standard Telegram 1. The Siemens Telegram 352 is an extension of the Standard Telegram 1, because it allows a speed control and transmits a greater number of information on the drive. In particular, the total number of words of the Telegram 1 is 4, while the total number of words of the Siemens Telegram 352 is 12. The Siemens Telegram 352 is composed of 6 status word (table 2.10) and 6 control word (table 2.9). In addition, the bit position of the first 2 command words of the Siemens Telegram 352 is equal to the bit position of the command words of Standard Telegram 1; even the first 2 status words of the telegram 352 are equal to the word status of Standard Telegram 1 (appendix A).

| PZD | Assigment of the process data |
|---|---|
| PZD | Control word in bit |
| PZD | NSOLL_A → Speed setpoint |
| PZD | Spare word |
| PZD | Spare word |
| PZD | Spare word |
| PZD | Spare word |

(Table 2.9 Command word list of Standard telegram 352)

| PZD | Symbol |
|---|---|
| PZD 1 | Status word |
| PZD 2 | NIST_A_GLATT → Smoothed speed actual value |
| PZD 3 | IAIST_GLATT → Smoothed actual corrent value |
| PZD 4 | MIST_GLATT → Actual torque |
| PZD 5 | WARN_CODE → Alarm number |
| PZD 6 | FAULT_CODE → Fault number |

(Table 2.10 Status word list of Standard telegram 352)

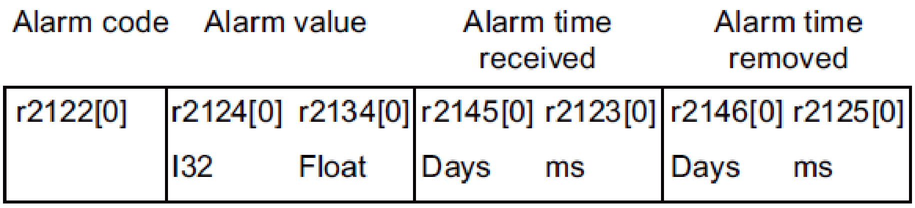
The command words structure has 2 predefined words from Siemens and 4 free word. These 4 words can be freely managed by the programmer based on the customer's requests.

As we said previously, the drive is made up of a list of the experts that has all its states. The telegrams do not transmit all the parameters of the expert list but they transmit only the most important. So, if the customer needs to read further states of the drive, the programmer can transmit additional states using spare words. In practice, the programmer connects the 4 words to the desired parameters and attaches them to a visible variable in the HMI.

The first command word handles the operation mode activation and deactivation commands, while the second word controls the speed the drive must reach.

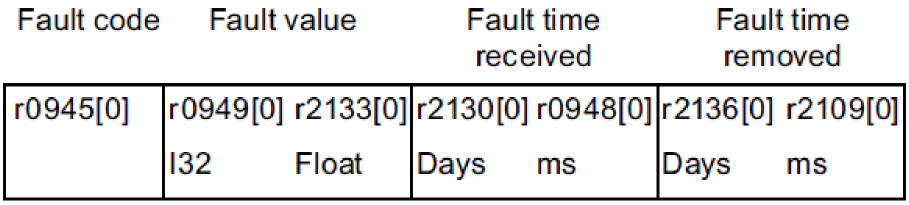
The status words of the Siemens Telegram 352 provide more information than the word status of Standard Telegram 1: for example, Standard Telegram 1 transmits only the presence of an error or a fault, while the Siemens Telegram 352 also transmits the error code and the fault code.

Specifically, PZD 5 is linked to the parameter of the list of experts "r2122 [0]" (figure 2.21), while PZD 6 is connected to the parameter of the list of experts "r0945 [0]" (figure 2.22).



| Alarm code | Alarm value | | Alarm time received | | Alarm time removed | |
|---|---|---|---|---|---|---|
| r2122[0] | r2124[0] | r2134[0] | r2145[0] | r2123[0] | r2146[0] | r2125[0] |
| | I32 | Float | Days | ms | Days | ms |

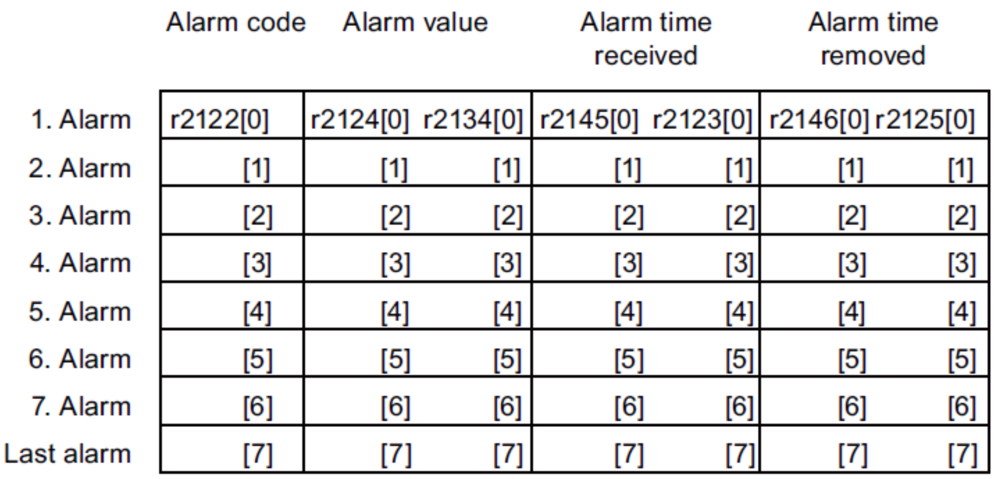
(Fig. 2.21 Parameters to display the alarm characteristics)



| Fault code | Fault value | | Fault time received | | Fault time removed | |
|---|---|---|---|---|---|---|
| r0945[0] | r0949[0] | r2133[0] | r2130[0] | r0948[0] | r2136[0] | r2109[0] |
| | I32 | Float | Days | ms | Days | ms |

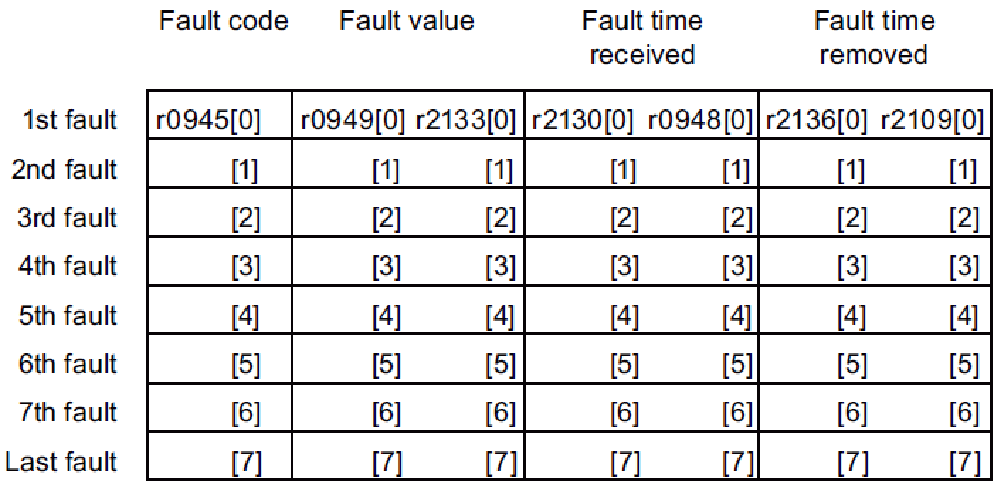
(Fig. 2.22 Parameters to display the fault characteristics)

Drives are able to store up to 8 alarms and 8 faults in their buffer. Alarms and faults remain in memory even if they have been resolved. If a ninth alarm or fault is received and none of the last eight alarms have been removed then the next to last alarm is overwritten. Alarms (figure 2.23) and faults (figure 2.24) are saved according to the date and time with a precision of the millisecond.



| | Alarm code | Alarm value | | Alarm time received | | Alarm time removed | |
|---|---|---|---|---|---|---|---|
| 1. Alarm | r2122[0] | r2124[0] | r2134[0] | r2145[0] | r2123[0] | r2146[0] | r2125[0] |
| 2. Alarm | [1] | [1] | [1] | [1] | [1] | [1] | [1] |
| 3. Alarm | [2] | [2] | [2] | [2] | [2] | [2] | [2] |
| 4. Alarm | [3] | [3] | [3] | [3] | [3] | [3] | [3] |
| 5. Alarm | [4] | [4] | [4] | [4] | [4] | [4] | [4] |
| 6. Alarm | [5] | [5] | [5] | [5] | [5] | [5] | [5] |
| 7. Alarm | [6] | [6] | [6] | [6] | [6] | [6] | [6] |
| Last alarm | [7] | [7] | [7] | [7] | [7] | [7] | [7] |

(Fig. 2.23 Alarms list and their characteristics)



| | Fault code | Fault value | | Fault time received | | Fault time removed | |
|---|---|---|---|---|---|---|---|
| 1st fault | r0945[0] | r0949[0] | r2133[0] | r2130[0] | r0948[0] | r2136[0] | r2109[0] |
| 2nd fault | [1] | [1] | [1] | [1] | [1] | [1] | [1] |
| 3rd fault | [2] | [2] | [2] | [2] | [2] | [2] | [2] |
| 4th fault | [3] | [3] | [3] | [3] | [3] | [3] | [3] |
| 5th fault | [4] | [4] | [4] | [4] | [4] | [4] | [4] |
| 6th fault | [5] | [5] | [5] | [5] | [5] | [5] | [5] |
| 7th fault | [6] | [6] | [6] | [6] | [6] | [6] | [6] |
| Last fault | [7] | [7] | [7] | [7] | [7] | [7] | [7] |

(Fig. 2.24 Faults list and their characteristics)

Build order is chronological: from most recent to oldest.

In other words, the programmer can read the alarms and faults history up to a maximum of 8, while the user is able to read only the most recent alarm by means of the Siemens Telegram 352.

Also in this case, as in the Standard Telegram 111, the drive transmits a number that can be "fixed point" or "floating point" number (table 2.11).

| Number | Cause | Remedy |
|---|---|---|
| F01000 | Software fault in CU | Replace CU. |
| F01001 | Floating Point Exception | Switch CU off and on again. |
| F01015 | Software fault in CU | Upgrade firmware or contact technical support. |
| F01105 | CU: Insufficient memory | Reduce number of data records. |
| F01250 | CU hardware fault | Replace CU. |
| A01028 | Configuration error | Explanation: Parameterization on the memory card has been created with a different type of module (order number, MLFB)<br>Check the module parameters and recommission if necessary. |
| A01920 | PROFIBUS: Cyclic connection interrupt | Explanation: The cyclic connection to PROFIBUS master is interrupted.<br>Establish the PROFIBUS connection and activate the PROFIBUS master with cyclic operation. |
| A03520 | Temperature sensor fault | Check that the sensor is connected correctly. |
| A05000<br>A05001<br>A05002<br>A05004<br>A05006 | Power Module overtemperature | Check the following:<br>- Is the ambient temperature within the defined limit values?<br>- Are the load conditions and duty cycle configured accordingly?<br>- Has the cooling failed? |

(Table 2.11 Some error and fault code list of Siemens inverter drive)

So, the Siemens Telegram 352 with 12 words is an extension of the Standard Telegram 1, because Siemens Telegram 352 allows an information exchange greater than that of the Standard Telegram 1.

## *2.3 Communication between Rockwell Automation CPU and Rockwell Automation drive*

The communication between Rockwell Automation CPU and Rockwell Automation drive comes through a standardized communication structure provided by Rockwell Automation. These structures are named "Data type" and they depend on the type of drive being used. Rockwell company does not provide standardized blocks such as SINA_POS and SINA_SPEED but it is necessary to construct a block that adapts to the drive's communication structure.

The most used Rockwell drives belong to the Powerflex520 and PowerFlex750 series for speed control, while for position control, Kinetix5500 are widely used.

Each drive family has a data type that manages the communication with the Rockwell Automation CPU. Particularly, appendix B shows the data type that allows communication between an inverter and a CPU.

### 2.3.1 Rockwell inverter: PowerFlex series

Rockwell Automation makes available to programmers of servo drives and inverters. Specifically, the Powerflex 520 is a family of frequency converter that manages the rotational speed of an engine (figure 2.25).

(Fig. 2.25 Example of PowerFlex 523 series configurations)

Figure 2.25 shows different configurations of the PowerFlex 523 drive. Particularly, the CU of the drive is the same in all the configurations, while the power module changes: the range of the power module goes from 0.2 kw (0.25 hp) to a maximum of 11 kw (15 hp) [Website No. 23 sitography].

The data type of Allen Bradley's Powerflex 525 is described in appendix B. Similar to Siemens telegrams, this structure is composed of words. Particularly, the PowerFlex 525 data type for speed consists of 2 command words and 2 status words. The first command word describes bit commands to run the speed control, and the second command word is used to communicate the desired speed. Unit of measurement of the speed must be in Hz. Indeed, the Rockwell Automation drives are frequency converters and therefore they manage the speed according to the frequency sent to the electric motor. In this case the conversion between "rpm" and "Hz" is standard and it should not be set by the programmer during the software configuration phase of the drive (equation 2.1):

$$60\, rpm \;=\; 1\, Hz \tag{2.1}$$

This conversion derives from the definition of "Hz" and "rpm":

• A "Hz" is the number of times a periodic event repeats each 1 second;

• An "rpm" is the number of revolutions in one minute.

As a result:

$$1\, Hz \;=\; 1\, \frac{revolution}{second} \rightarrow 60\, revolutions\ per\ minute \;=\; 60\, rpm \tag{2.2}$$

The precision of the PowerFlex 525 drives is per cent, so the programmer must impose the speed using the Hz cents. For example, if the engine is to run at a speed of 600 rpm, the programmer must set a value equal to 1000 (equation 2.3):

$$600\, rpm \;=\; 10\, Hz \;=\; 1000\, hundredths\ of\ Hz \tag{2.3}$$

Rockwell Automation also provides the user with a keyboard called keypad. It is used to command the drive in a direct way i.e. without a CPU. Obviously in this case programmer cannot implement a program but user can start or stop the drive and set the parameters. For example, user can increase the speed by means of the eighth bit of the first command word. In this case the speed override depends on the setting that the programmer has included in the list of drive experts. This function takes the name of MOP increment. Similarly, user can perform a MOP decrement by setting the fifteenth bit of the first word. In addition, the list of drive experts contains 3 parameters that can store three different speeds. These parameters are:

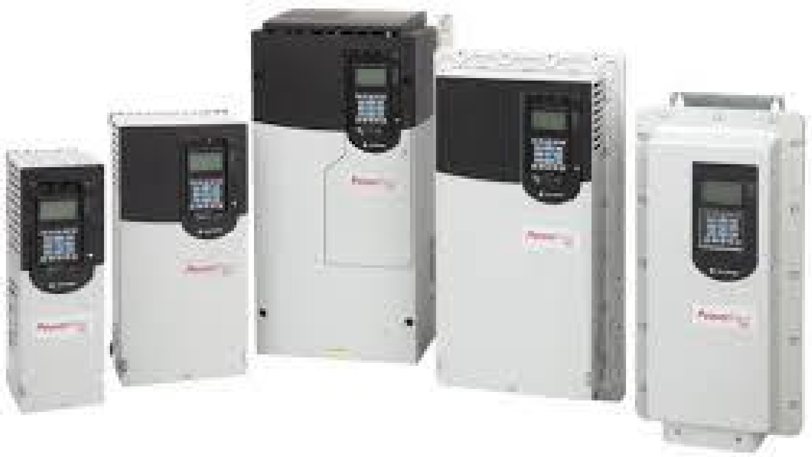
- "P047" corresponds to "Speed reference 1";
- "P049" corresponds to "Speed reference 2";
- "P051" corresponds to "Speed reference 3".

The user can decide rotor speed by selecting it directly through the Keypad. Specifically, the bits to be activated are:

- "12" for "Speed reference 1";
- "13" for "Speed reference 2";
- "14" for "Speed reference 3";

Like Siemens inverters, the PowerFlex 520 is able to control a position if it is coupled to an encoder. If the programmer implements such a system, the encoder would provide the rotor current position while the PID control would handle the rotor speed to reset it to the desired position [Document No. 17 bibliography].

Like PowerFlex 520 series, the PowerFlex 750 series is also a series of frequency converters that handles the rotational speed of an engine (figure 2.25).



(Fig. 2.26 Example of PowerFlex 750 series configurations)

Figure 2.26 shows different configurations of the Powerflex 750 drive. In particular, drive CU is the same in all configurations, while power module changes.

PowerFlex 750 executes position control by means of external encoder. In this case, the data type of Allen Bradley's PowerFlex 750 is described in appendix B. The data type of the PowerFlex 750 consists of 2 command words and 2 status words, as in the case of the PowerFlex 520.

The internal structure of the first command word is very similar to the data type of the PowerFlex 520, but there are some differences. The main differences concern the presence of the "Find Home" bit and the communication between CPU and drive if keypad is used. For example, there is no "Find Home" bit in the case of a speed control while there is a position control. In addition, if in the case of speed control, the bits 12, 13 and 14 set only the desired frequency, in the case of a position control the bits 8, 9 and 10 set both position and frequency. Note also that second command word sets the position and not the speed, indeed unit of measure are degrees. In addiction PowerFlex 750 power range of three-phase 240v input drives runs from 0.5 hp to 200 hp.

In conclusion, Rockwell Automation PowerFlex series is made up of inverters. They perform a speed control and a control in position when coupled to an external encoder. The communication between drive and CPU is through a structure similar to the Siemens inverters.

### 2.3.2 Rockwell Automation servo drive: Kinetix series

The servo drives made available by Rockwell Automation belong to the Kinetix family of drives. In particular, Kinetix 300 is servomotor which has an internal encoder. So, the Kinetix 300 series manages the position of an engine (figure 2.27).

(Fig. 2.27 Example of Kinetix 300 series configurations)

Power range of servo drives Kinetix 300 ranges from 0.4 kw to 3 kw and input voltage range goes from 115 V to 240 V for CA single-phase and from 230 V to 480 V for CA three-phase [Website No. 27 sitography].

The data type of Allen Bradley's Kinetix 300 is described in appendix B.

It should be not that the structure is subdivided in bytes, but also in this case there are the main commands and the words for the speed and position settings. In other words, Rockwell Automation drives structures are similar to each other regardless of the subdivision type.

Again, speed must be expressed in cents of Hz and position in degrees.

The Rockwell Automation Company's Kinetix series corresponds to the Siemens company S series (S 110 and S120).

There are several devices that belong to the Kinetix series like the Kinetix 5500 or the Kinetix 7500, but the communication structures are the same.

In conclusion, Rockwell Automation's Kinetix series constitutes of servo drives that allow a control in position by means of a communication structure between drives and CPUs in the same way as Siemens drives.

## *2.4 Communication interface*

A key factor for the communication between CPU and drive is the words structure. The program is designed to create an interface capable of communicating drives and CPUs independently of the word structure.

The ladder diagram is the programming language used to write the program. The choice of this programming language allows the use of the block for the programming of all PLC on the market, because the ladder diagram is in all the PLC programming software.

The program consists of two blocks: the first one performs the functions of control, writing of command and reading of the drive states. The second block is internal to the first one and it is the block that processes commands to run according to the drive states and the user's commands.

The program allows the management of the communication through the single bit in case of logic commands and whole word in case of transmission of numerical values. This is possible because each word in the communication structure consists of bits (appendices B and A).

Specifically, the interface block has two types of inputs: user inputs (table 2.12) and drive inputs (table 2.13).

| Name | Data type | Meaning |
|---|---|---|
| IN_CmdSpeed | Bool | Speed mode command |
| IN_CmdJogPos_Speed | Bool | Positive jog speed mode command |
| IN_CmdJogNeg_Speed | Bool | Negative jog speed mode command |
| IN_CmdAbsolutePositioning | Bool | Absolute positioning mode command |
| IN_CmdHomeRes | Bool | Home Res mode command |
| IN_Home_Direction | Bool | Homing direction definition 0=Positive, 1= negative. |
| IN_CmdHomeSet | Bool | Home Set mode command |
| IN_CmdJogPos | Bool | Positive jog mode command |
| IN_CmdJogNeg | Bool | Negative jog mode command |
| IN_StopCycle | Bool | Stop cycle command |
| IN_Direction | Bool | Direction definition 0=positive, 1= negative. |
| IN_PosTargetDest | Real | Position target |
| IN_SpeedMax | DInt | Max speed |
| IN_Override | Int | Wanted speed |
| IN_Enable_Axis | Bool | Axis activation command |
| IN_CancelTraversing | Bool | 0 = reject active traversing task. |
| IN_FlyRef | Bool | 1 = select flying referencing |
| IN_MDI_Mode | Bool | Activate traversing task / setpoint acceptance / activate reference function |
| IN_Dec_Override | Int | Deceleration override active 0-100% |
| IN_Acc_Override | Int | Acceleration override active 0-100% |
| IN_ResetAlarm | Bool | Fault/Alarm cancellation |
| IN_Cmd_RockMOPIn | Bool | MOP increment command |
| IN_Cmd_RockMOPDec | Bool | MOP decrement command |
| Always_OFF2 | Bool | OFF2 activation |
| Always_OFF3 | Bool | OFF3 activation |
| Always_EnableControlPLC | Bool | Control PLC enabling |
| Always_EnableRamp | Bool | Ramp enabling |
| Always_ContinueRamp | Bool | Ramp continue correctly |
| Always_EnableOperation | Bool | Operation enabling |
| Always_EnableSpeed | Bool | Speed enabling |
| Always_EnablePos | Bool | Position enabling |

(Table 2.12 User input signals of “R101_Control_Interface” block)

The user sets the desired settings by means of these inputs. Particularly, inputs are always activated by default, but they can be managed by the user by connecting them to the HMI. In other words, user inputs allow the choice of the control type and how the control should be executed.

The inputs dedicated to the drive, however, read the states of the drive.

| Name | Data type | Meaning |
|---|---|---|
| Status_Alarm | Bool | 1= there is alarm |
| Status_Ready | Bool | Drive is ready |
| Status_Blocked | Bool | 1=there is a fault → the drive is stopped |
| Status_AxisEnabled | Bool | Axis is enabled |
| Status_PositiveDir | Bool | Positive direction is defined |
| Status_NegativeDir | Bool | Negative direction is defined |
| Status_Direction | Bool | Direction type, if drive structure has only one bit. |
| Status_ReferenceDone | Bool | Homing executed |
| Status_SpeedReached | Bool | Speed setpoint reached |
| Status_PositionReached | Bool | Position setpoint reached |
| Status_AlarmCode | Word | Alarm code |
| Status_WarningCode | Word | Warning/ fault code |
| Status_ActualSpeed | DInt | Current speed value |
| Status_ActualPosition | DInt | Current position value |

(Table 2.13 Drive input signals of R101_Control_Interface” block)

Specifically, “Status_Ready”, “Status_Blocked” and “Status_AxisEnable” inputs have been used as conditions for the execution of the operating mode. While the remaining states are transmitted to the user because they allow to monitor the drive operation (current speed, current position, fault code and current alarm if there is a fault or an alarm).

Similar to inputs, the interface block has two types of outputs: user outputs (table 2.145) and drive outputs (table 2.15).

| Name | Data type | Meaning |
|---|---|---|
| OUT_Alarm | Bool | Presence of alarm |
| OUT_Blocked | Bool | Presence of fault |
| OUT_Reference_Setted | Bool | Homing executed |
| OUT_Speed_Reached | Bool | Speed setpoint reached |
| OUT_Position_Reached | Bool | Position setpoint reached |
| OUT_PositiveDir | Bool | Positive direction is defined |
| OUT_NegativeDir | Bool | Negative direction is defined |
| OUT_Alarm_Code | Word | Alarm code |
| OUT_Warning_Code | Word | Warning/Fault code |
| OUT_DriveActualPosition | DInt | Current position value |
| OUT_DriveActualSpeed | DInt | Current speed value |
| OUT_DriveAxisEnabled | Bool | Axis is enabled |
| OUT_ActualPosition_rpm | Real | Current position value [revolutions] |
| OUT_ActualSpeed_rpm | Real | Current speed value [rpm] |
| OUT_RockActSpeed_rpm | Real | Current speed value [Hz→rpm] |
| OUT_LU_ActSpeed_rpm | Real | Current speed value [LU→rpm] |

(Table 2.14 User output signals of R101_Control_Interface” block)

The goal of the user outputs is the communication of the drive states to the user. In other words, the user can read the main states of the drive by means of these outputs.

The outputs dedicated to the drive are the drive commands. Videlicet, the block controls drive operation through these outputs.

| Name | Data type | Meaning |
|---|---|---|
| Control_Start | Bool | Start or OFF 1 |
| Control_JobStart | Bool | Drive starts to work |
| Control_noJOB_STOP | Bool | Job is blocked |
| Control_NOSTOP | Bool | Drive no stops working |
| Control_ControlFromPLC | Bool | Control from PLC enabling |
| Control_RUN | Bool | Running mode |
| Control_GoalPosition | DInt | Setpoint position |
| Control_SpeedPosition | DInt | Setpoint speed for position control |
| Control_OverrideV | Int | Wanted velocity |
| Control_OverridePosV | Int | Override velocity for position control |

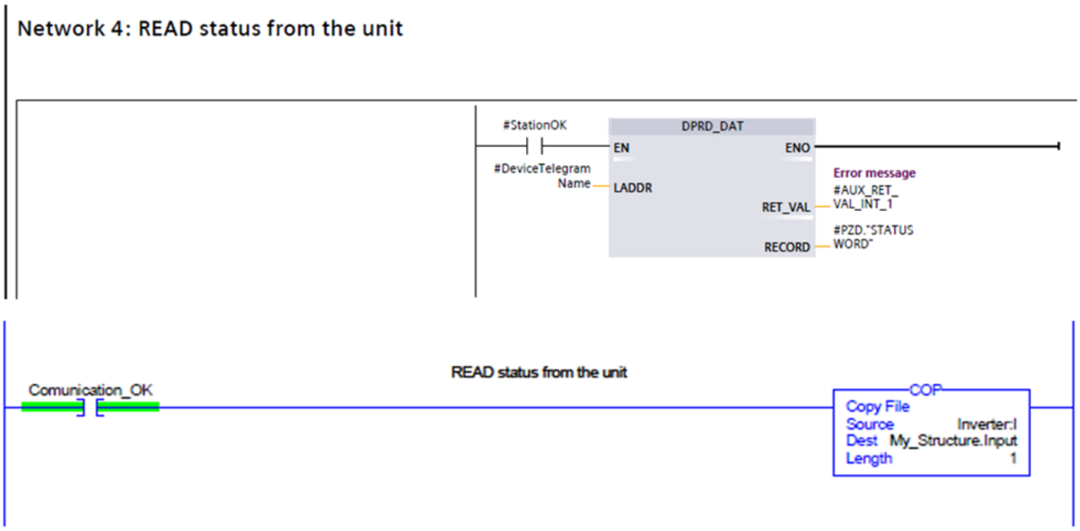
| | | |
|---|---|---|
| Control_Acceleration | Int | Setpoint acceleration |
| Control_Deceleration | Int | Setpoint deceleration |
| Control_GoalSpeed_LU | Int | Setpoint speed [LU] |
| Control_GoalSpeed_Hz | Int | Setpoint speed [ Hz] |
| Control_GoalSpeed_rpm | Int | Setpoint speed [rpm] |
| Control_PositioningMode | Int | Positioning mode activation |
| Control_ReferenceSearchStart | Int | Start of reference search mode |
| Control_PositiveDir | Int | Positive direction definition |
| Control_NegativeDir | Int | Negative direction definition |
| Control_RefType | Bool | Reference type |
| Control_ReferenceSearchDir | Bool | Direction of reference search mode |
| Control_Jog1 | Bool | Jog 1 mode activation |
| Control_Jog2 | Bool | Jog 2 mode activation |
| Control_AlarmReset | Bool | Clear faults |
| Control_SetReferencePoint | Bool | Homing mode activation |
| Control_SpeedDirection | Bool | Direction of speed control (0=Positive direction, 1= negative direction) |
| Control_EnableSpeed | Bool | Speed enabling |
| Control_OFF2 | Bool | OFF2 activation |
| Control_OFF3 | Bool | OFF3 activation |
| Control_EnableOperation | Bool | Operation enabling |
| Control_EnableRamp | Bool | Ramp enabling |
| Control_ContinueRamp | Bool | Ramp continue correctly |
| Control_EnablePos | Bool | Position enabling |
| Control_RockMOPIn | Bool | MOP increment command |
| Control_RockMOPDec | Bool | MOP decrement command |
| Control_RockJOG | Bool | Jog mode activation for Rockwell |
| Control_RockNegative | Bool | Negative direction definition for Rockwell |
| Control_RockPositive | Bool | Positive direction definition for Rockwell |
| Control_Stop | Bool | Drive stops working |

(Table 2.15 Drive output signals of R101_Control_Interface” block)

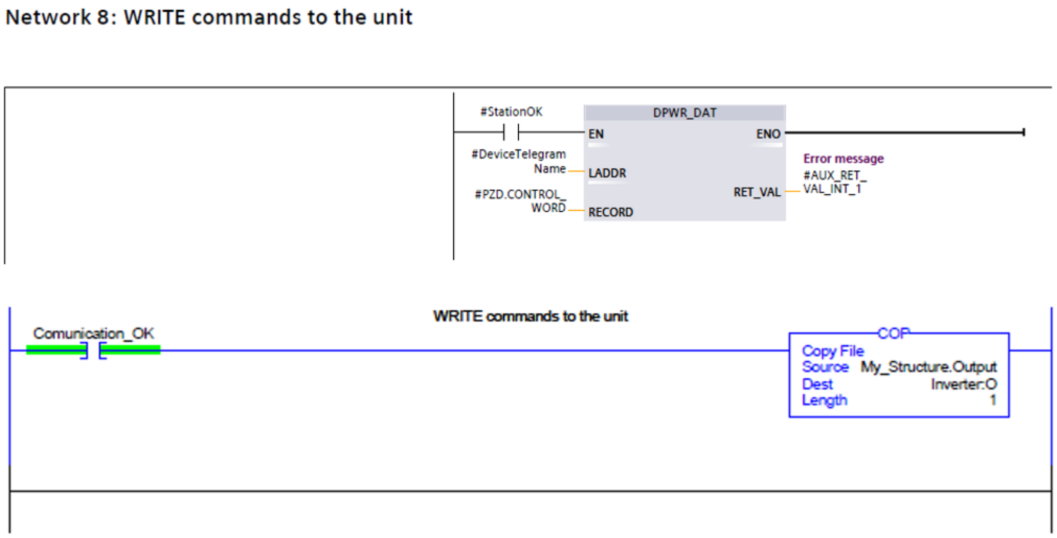
All the main states for activating the drives have been entered. The choice of commands was made according to the command word of Siemens telegrams and data type of Rockwell. The choice of these brands is made because they are antipodes. Generally, nowadays drive structures are similar to the Siemens telegrams or similar to the Rockwell data types. The main difference is the bit position within the word. So, by working on a single bit programmer can connect each bit to the type of communication he wants to have regardless of its position. In other words, if the programmer is able to control the drives of both companies by means of this block, it would be able to command most of the drives that are on the market.

To manage the communication structures by means of the individual bits the programmer has to create an internal structure to the program that is equal to the communication structure of the drive building.

Subsequently, the structure built by the programmer must be associated with the drive outputs of the block for the commands management and the drive inputs of the block for the states reading. The assignment of values must be made bit per bit according to the function that the single bit performs. In this way, internal structure will be built. For writing commands, the command word of the internal structure is copied into the drive structure. For the reading of states, the same but inverse path occurs: in this case the communication structure of drive is copied to the internal structure. In this way, the drive states can be read from the command-processing block. These two phases are called reading data and writing data.

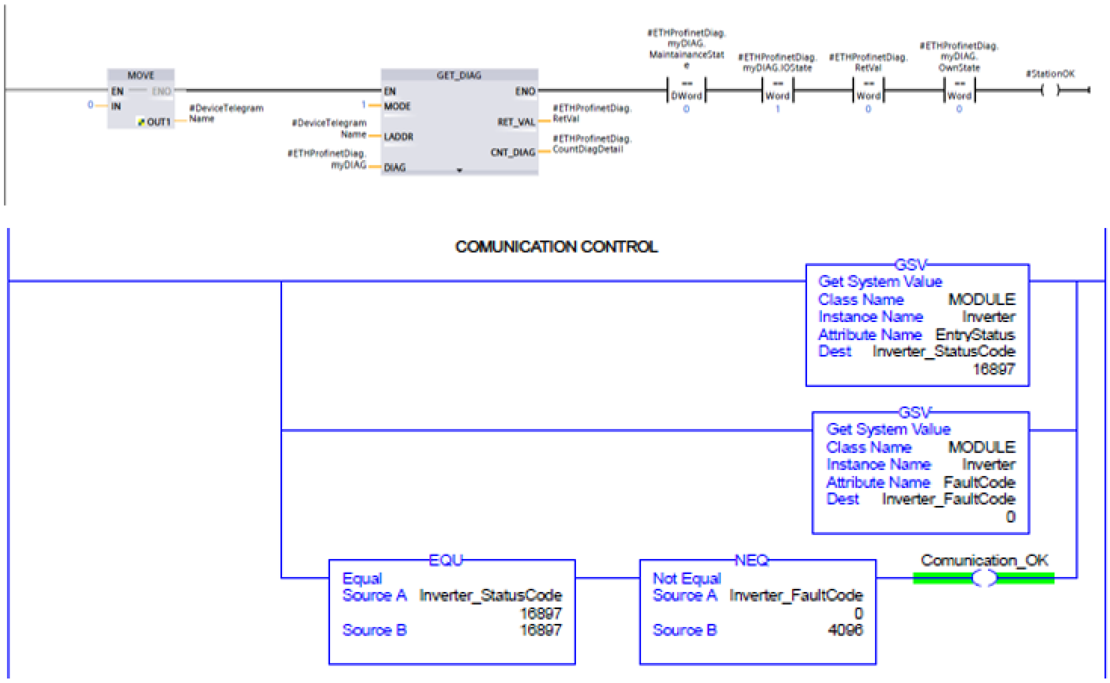


(Fig. 2.28 Ladder diagram block for reading function)



(Fig. 2.29 Ladder diagram block for writing function)

Reading and writing data are executed if the communication between CPU and drive is successful. Sure enough, a control over the communication has been entered (figure 2.30).



(Fig. 2.30 Ladder diagram block for control function)

If the communication is interrupted, the new commands will not be applied and the drive will be blocked.

Interface block "R101 Control_Interface" allows the activation of all the necessary bits to manage the drive.

"R101 Control_Interface" function block performs all operations to enable and disable bits. In particular, the second network manages the activation of all the modes that the block is able to handle.

Modes that the block is able to handle are:

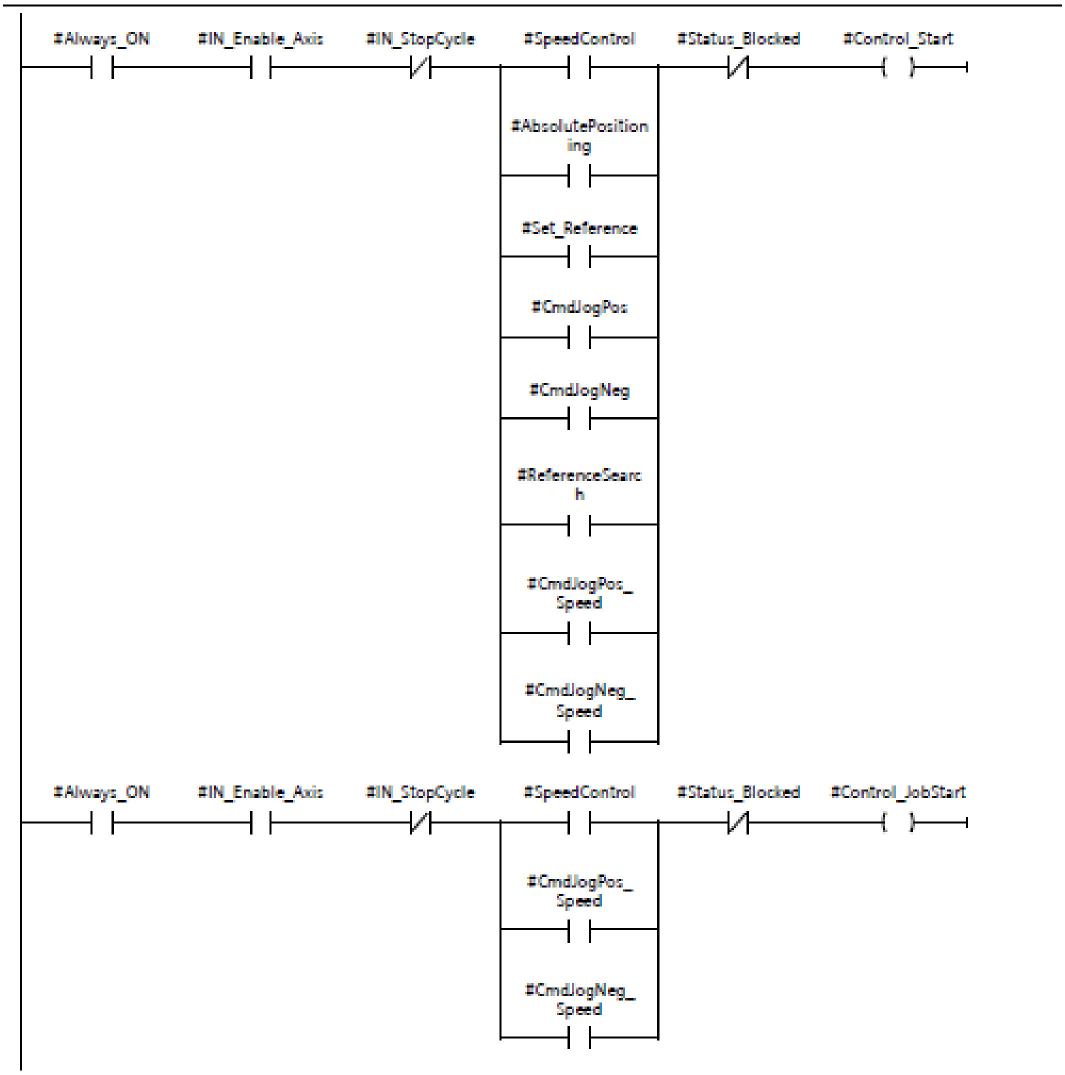
- "Speed mode command";
- "Positive jog speed mode command";
- "Negative jog speed mode command";
- "Absolute positioning mode command";
- "Home Res mode command";
- "Home Set mode command";
- "Positive jog mode command";
- "Negative jog mode command".

Specifically, a management is performed to avoid contradictory commands. Indeed, if the programmer or user entered two commands at the same time, the block would not send any command. This has been done because the drives are not able to perform two tasks in the same time: for example the drive S110 Siemens is not able to execute a homing command and positioning one at the same time, because for the execution of the homing the axis must remain standstill to allow the zero position to be recorded while positioning must start from a reference position (the zero position) to reach the desired position. Similarly, it happens for inverters: they cannot run a speed control and at the same time a jog mode.

So, network 2 avoids the origin of a contrast inside the drive thus avoids the generation of faults (appendix C).

The network 5 manages the command "Control_Start" and the "Control_JobStart" (figure 2.31). The conditions for activating these commands are:

- user has enabled the axis;
- user has not voluntarily blocked the cycle;
- user has activated a mode of operation;
- drive is not in fault state.

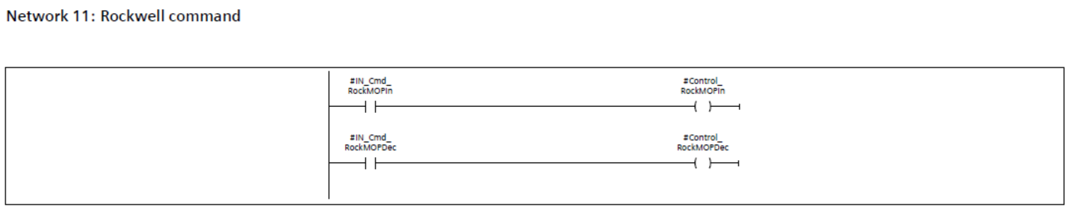


(Fig. 2.31 Ladder diagram network for “start” function)

Another important network is the network 9 "Always_ON Siemens". As specified previously, "Configurationepos" (SINA_POS block) and the "Configurationaxis" (SINA_SPEED block) are two vectors that have been set correctly. The programmer is able to manage the main values inside these structures freely through the network 9 (appendix C). Also, in this case “ConfigurationEPos” and “ConfigurationAxis” values have been activated by default for the execution of the control, but they are easily manageable also through the HMI. For example, if a user wanted to handle the "OFF2" bit of the Standard Telegram 111 or Standard Telegram 1, the programmer would be able to provide this control by attaching the variable "Always_OFF2" to an address on the user's HMI.

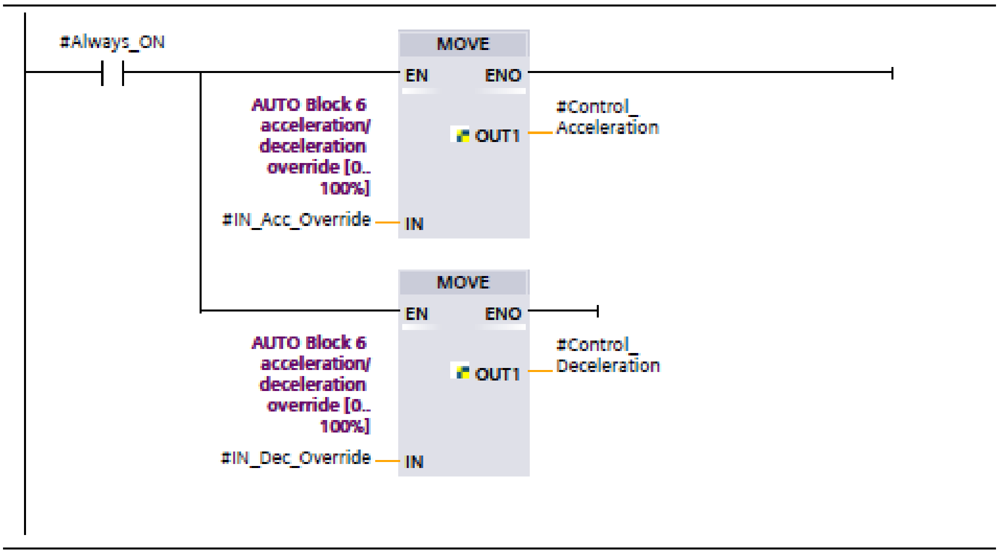
This network is important because it allows the management of all the drives that have a similar structure to Siemens as ones of the Sew-eurodrives company.

The network 11 is studied for Rockwell to expand the possibility of managing its drives. Indeed, “MOP increment” and “MOP decrement” functions were created for the execution of a control similar to Keypad. The network 11 allows the management of these functionalities also through a PLC and HMI (figure 2.32).



(Fig. 2.32 Ladder diagram network for Rockwell Automation commands)

Two of the main parameters for the drive management are the acceleration and the deceleration. Particularly, the program let to set a wanted override of acceleration and wanted override of deceleration. The maximum acceleration and maximum deceleration are automatically set by the drive based on the engine nameplate data, but the programmer or directly the user can manage acceleration and deceleration by entering a percentage value.

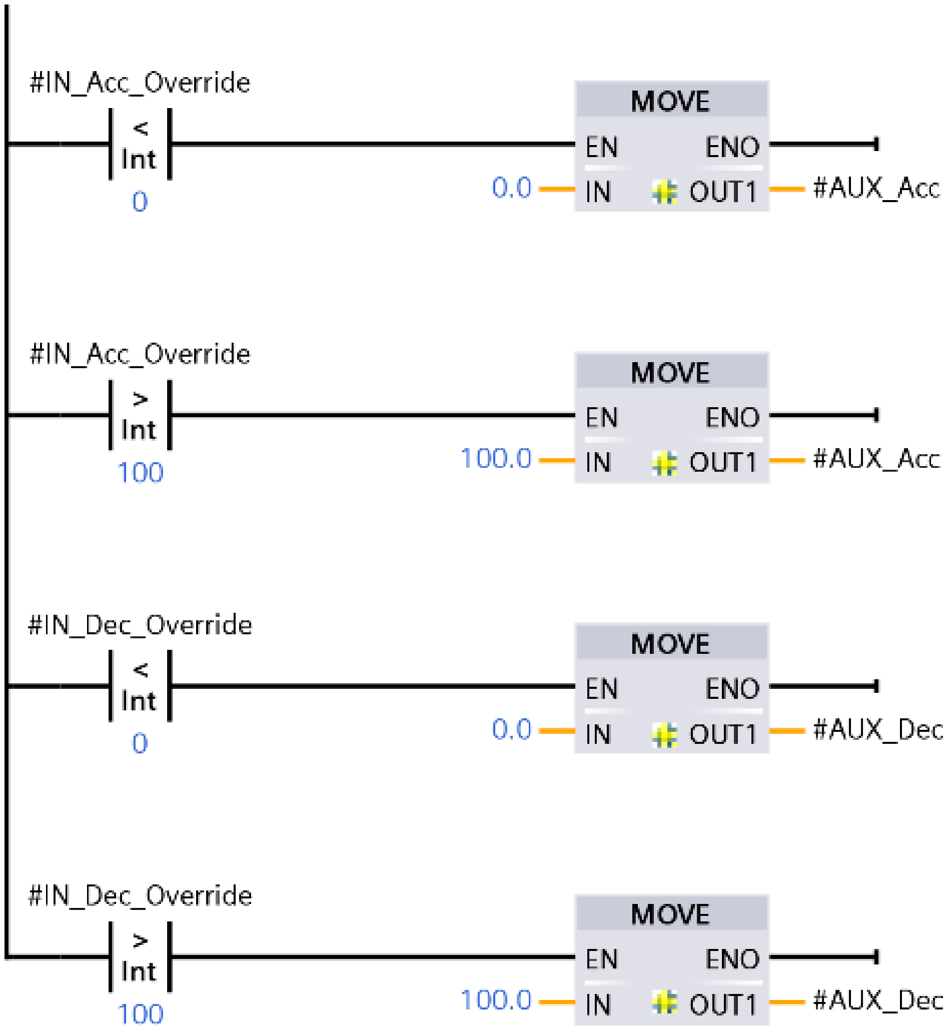

#Always_ON
MOVE
EN
ENO
AUTO Block 6 acceleration/ deceleration override [0.. 100%]
#IN_Acc_Override
IN
OUT1
#Control_
Acceleration
MOVE
EN
ENO
AUTO Block 6 acceleration/ deceleration override [0.. 100%]
#IN_Dec_Override
IN
OUT1
#Control_
Deceleration

(Fig. 2.33 Ladder diagram network for Acceleration & deceleration management)

For example, if the user wants to always have maximum acceleration, the programmer sets the variable "IN_Acc_Override" with a value of 100. While if user want a deceleration equal to half of the maximum, the programmer must set 50 to the variable "IN_Dec_Override".

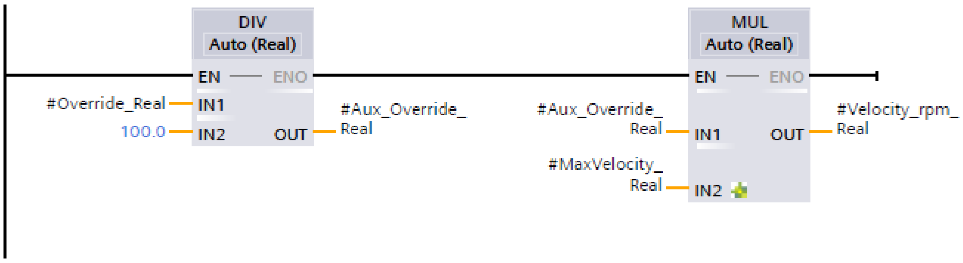
All numeric parameters entered by the user or programmer are controlled by the network 19 (appendix C).

For example, if the acceleration override value of is greater than 100, it will be automatically set to 100 to avoid damage the engine and the drive; conversely, if the acceleration is less than 0, it will automatically be set to 0.



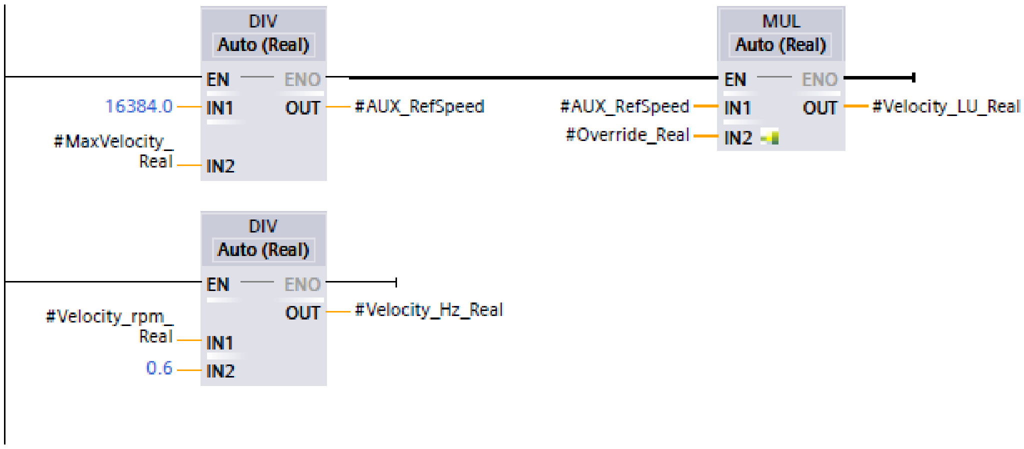
(Fig. 2.34 Ladder diagram network to control acceleration and deceleration values)

It should be noted the drive can read and manage the speed by means of different units of measurement. Indeed, if the unit of measurement of the speed desired by the drive is "rpm", the calculation of the desired speed is performed by the network 22 "Compute of Velocity in rpm".



(Fig. 2.35 Ladder diagram network to compute of wanted velocity [rpm])

If, however, the unit of measurement of the drive is the hundredth of Hz, a conversion is required: "Velocity_rpm_Real" must be divided by 0.6 (figure 2.36).



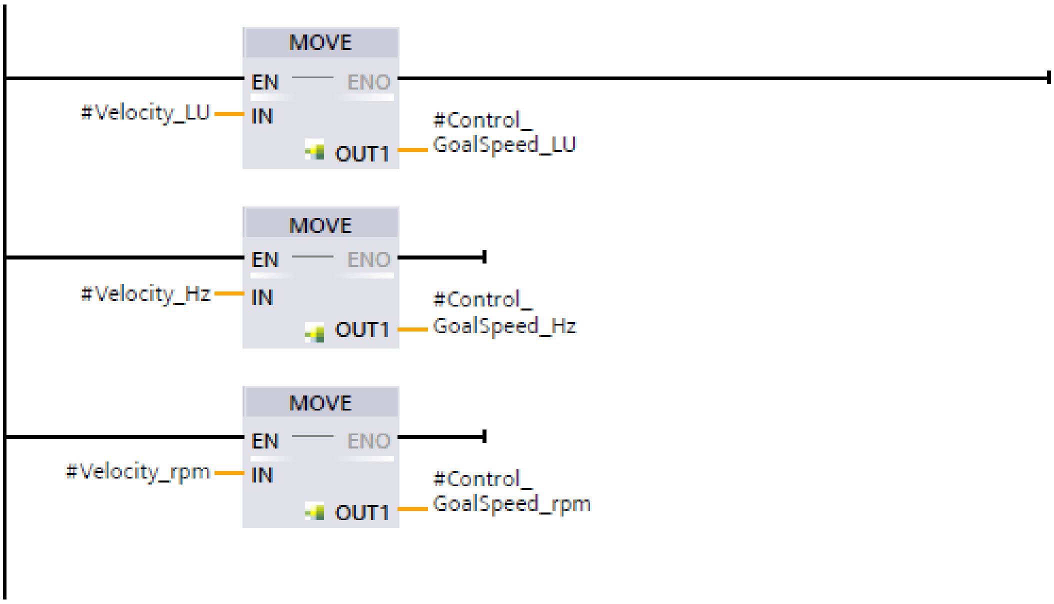
(Fig. 2.36 Ladder diagram network to compute of wanted velocity [LU/Hz])

In addition, Siemens drives calculate the desired speed by taking into account the “p2000” parameter. This parameter is in the expert list and is set when the programmer performs the drive commissioning. A Siemens drive reads the “p2000” parameter and applies the desired speed (figures 2.36).

Note that the speed management in LU is particular: the maximum value is divided by the constant "16384.0". This is because Siemens drive reads word by inverting bytes: the first byte is read after the second byte. In other words, Siemens reverses the priority of bytes. Therefore, the "16384.0" value must be divided by maximum value to make the reversal of the imposed parameter (figure 2.36).

The new speed values are checked again to prevent faults in the drive. These controls are performed in the network “Max speed writable in LU”, “Max speed writable in Hz” and “Max speed writable in rpm” (appendix C).

At this point, the network 29 "Speed SetPoint" imposes the value of the speeds obtained to the various outputs of the drive. In particular there are 3 types of variables because speeds have been calculated with 3 different units of measure (figure 2.37).



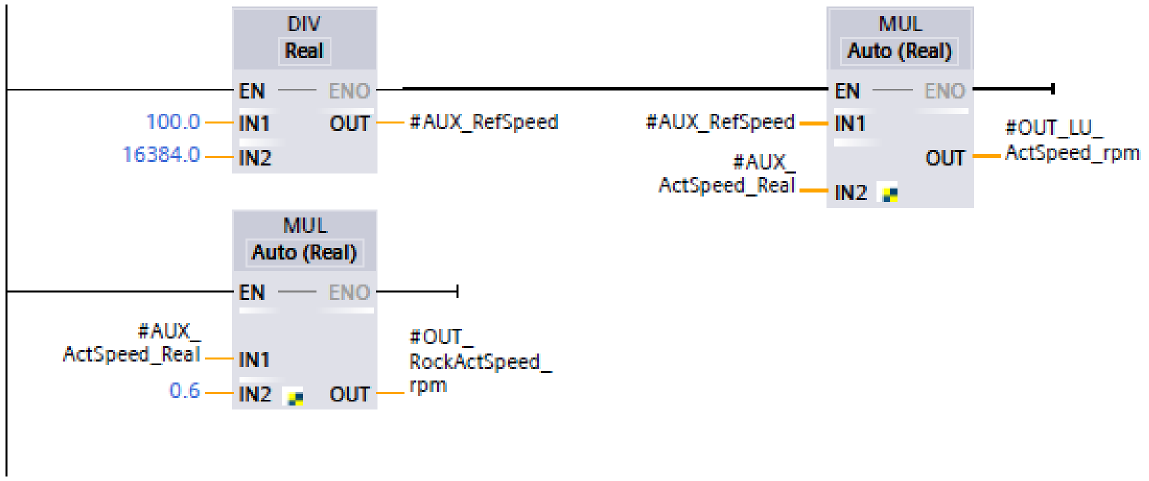
(Fig. 2.37 Ladder diagram network to write the speed setpoint [rpm/LU/Hz])

From the network 31 "Direction definition" (appendix C) management phase of the direction begins. Direction is managed according to the mode of operation that the user has set. Indeed, each mode actives several bits, such as the direction management bits for absolute positioning mode are different from those of the reference search mode (Standard Telegram 111, appendix A).

Jog function for the speed control is very important, because it allows to test the movement of a motor in safety. This function is performed by the "Movement JOG – for speed control" and "Movement JOG + for Speed control" networks. These networks allow an effective manual control at low speed in the case of a speed control. They are an extension of the drive's functionality, because in standardized Siemens blocks for the speed control there is not Jog mode. Specifically, if the user wants to run the speed control jog mode, the speed will be set at 20% of the total speed regardless of the initially inserted override (IN_Override). In addition, the direction will also be automatically set: it does not depend on the input "IN_Direction", but it depends on the inputs "CmdJogNeg_Speed" and "CmdJogPos_Speed". For example, if the user sets the input "CmdJogNeg_Speed" the rotor will move at a speed of 20% compared to the maximum in the negative direction. While if the user sets the input "CmdJogPos_Speed", the rotor will move with a speed of 20% compared to that maximum in positive direction.

This first part manages commands to be sent to the drive, while the second part reads drive states and sends the states to the user.

Networks 51 and 57 perform user output compilation as a function of the drive states. Particularly, the 57 network provides the user with the current speed of the drive and thus a conversion from “LU” to “rpm” and a conversion from hundredths of Hz and rpm.



(Fig. 2.38 Ladder diagram network to convert actual speed to rpm)

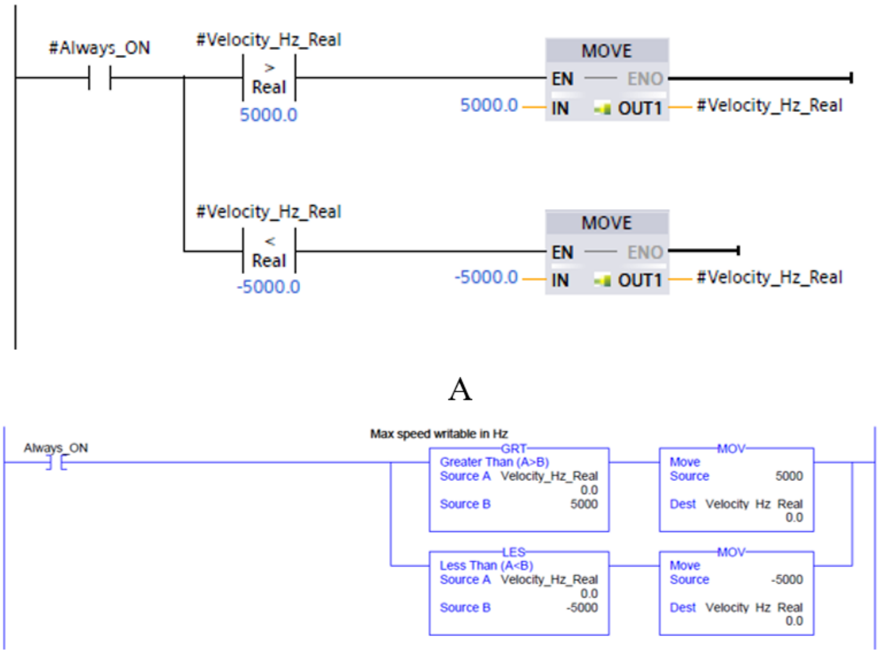
The advantages of this program are: flexibility, cost and reduction of the programming time.

Flexibility is a very important factor in the programming field because it allows to adapt a program to different needs. In particular, this program can be adapted to the communication structures of different types of drives thanks to the single bit manipulation. For example, a Siemens CPU cannot communicate with a Rockwell drive via the SINA_POS and SINA_SEED blocks, but this program is capable of communicating a Siemens CPU with a Rockwell drive and vice versa. In addition, the program allows the management of the complete communication structure. For example, the programmer can control the individual bits of the vector "ConfigurationEPos" (SINA_POS block) and the vector "ConfigurationAxis" (SINA_SPEED block). The interface is intuitive and easily manageable. This creates a reduction in programming time and lower cost. Also, within the "R101 Control_Interface" block, the Jog function for a speed Control has been implemented. The implementation of the Jog speed control reduces programming time and increases the user's ease of handling.

On the contrary, the disadvantages of this program are: the number of operations and the lack of an automatic converter.

The first disadvantage is not a very serious problem because the frequency of the PLC's processors is high. As a result, the cycle time is very small respect to number of operations.

As can be seen from the appendix C, despite the functions performed are equal (figure 2.39), but graphical interface and compilation of software T.I.A. Portal (Siemens) and RSLogix5000 (Rockwell) are different.



A

B

(Fig. 2.39 Example of same functions, but in T.I.A. Portal (Siemens) and Rslogix5000 (Rockwell) interface)

As a result, the lack of a converter generates the need to recopy the same program in different environments based on CPU. For example, the programmer must copy this program in Rockwell software environment to manage a drive with a Rockwell CPU or in Siemens software to manage a drive with a Siemens CPU.

In conclusion, the program allows word management taking into account the single bit of an internal structure that must be identical to the drive structure. By applying this technique, the programmer is able to manipulate bits and associate them correctly according to their function regardless of the order in the drive structure.

# 3. The communication interface application: test results

## INTRODUCTION

In this chapter, the results of the tests performed will be reported. Tests were performed using Siemens and Rockwell Automation PLCs and drives.

The first paragraph will explain the preliminary tasks that are required for the test to run; also, in this chapter there will be a list of communication structures tested.

In the paragraph 2, the test of the communication between Siemens inverter and Siemens CPU through the Standard Telegram 1 will be analyzed. Particularly, all the operating modes of the inverter will be studied. Indeed, the first phase of the test will focus on positive speed control; the second phase of the test will analyze negative speed control; finally, positive jog mode and negative jog mode will be examined. The analysis describes in detail all the necessary steps for a correct operation of the block and a correct communication between the PLC and the drive.

Paragraph 3 will study the communication test between Siemens inverter and Siemens CPU using the Siemens Telegram 352. Specifically, all operating modes of the inverter will be analyzed and the difference between the Standard Telegram 1 and the Siemens Telegram 352 will be applied for the reading of the drive states. Sure enough, initially there will be a study about the performance of the positive speed control, then the test will analyze negative speed control and finally positive jog mode and negative jog mode functions will be examined. These accurate analyses will describe in detail all the necessary steps for a correct operation of the block and a correct communication between the PLC and the drive.

In paragraph 4, the test about communication between Siemens servo drive and Siemens CPU using the Standard Telegram 111 will be analyzed. Specifically, the servo drive main modes will be studied. Indeed, the first phase of the test will examine positive jog mode; the second phase of test will analyze negative jog mode; in the third phase of the homing mode test will be tested; then there will be the positive absolute positioning analysis and the negative absolute positioning study and finally positive reference search point and the negative reference search point will be analyzed. The study of these modalities will describe in detail all the necessary steps for a correct functioning of block and a correct communication between PLC and drive.

Finally, the last paragraph will accurately focus on the communication between Rockwell Automation inverter and Rockwell Automation CPU. In this case, the PowerFlex 525 inverter is chosen for communication analysis. In a manner similar to Siemens inverter test, all operating modes of inverter will be analyzed. Sure enough, the first phase of the test will study positive speed control; the second phase of the test will analyze negative speed control; the third phase will examine the positive jog mode and finally the last phase of the test will describe negative jog mode. The analyses will describe in detail the necessary steps for a correct operation of the block and a correct communication between the PLC and the drive.

## *3.1 Program application*

The program constitutes of 2 blocks: "R100_Standard_NAZARI [FB1]" and "R101_Control_Interface [FB2]".

The slave block "R101_Control_Interface [FB2]" is inside the master block "R100_Standard_NAZARI [FB1]". In this way, the programmer must recall a single block in the main one. The master block is recalled through the recall function as for the standard SINA_POS and SINA_SPEED blocks.

After the programmer has invoked the block, he must open the master block and must fill out all the internal fields. The first variable to bind to the block is "Devicetelegramname" (appendix C, network 2).

Then the programmer must complete the interface, associating user inputs variables and user outputs variables according to the user's request and connecting drive inputs variables and drive outputs variables according to the chosen telegram.

In other words, each bit of the drive's communication structure must be assigned to each bit of the standard block's drive output signals. An example of the interface application is the test on the Standard Telegram 1 (tables 3.1 and 3.2).

Generally, this structure can be applied regardless of the drive on the market. Specifically, the tests were carried out for:

- “Standard Telegram 1”;
- “Siemens Telegram 352”;
- “Standard Telegram 111”;
- “PowerFlex 520 data type”.

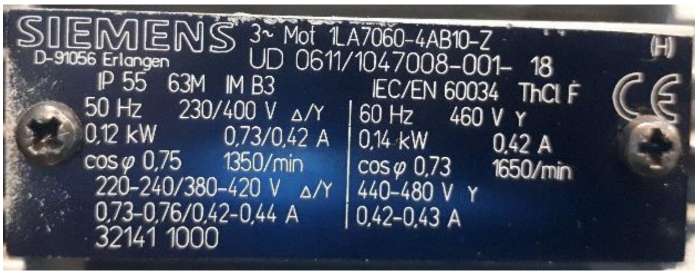
## 3.2 *Test about Standard Telegram 1*

The test on the communication interface with the Siemens drive was performed using the Siemens PLC 1511 F (figure 3.1), an electric motor and drive G120 without external encoder (figure 3.2).

(Fig. 3.1 Siemens PLC 1511F)

(Fig. 3.2 Siemens inverter drive G120)

The motor characteristics are described in the following figure (figure 3.3).



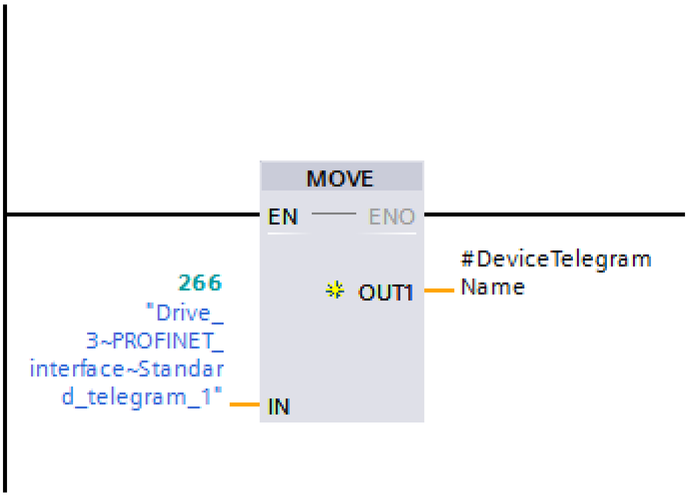
(Fig. 3.3 Electric motor plate)

The G120 drive is an inverter and it is able to control the engine speed. The communication structure between CPU and drive G120 is the Standard Telegram 1 and Siemens Telegram 352 (appendix A).

The objective of the following test is the correct functioning of the Standard Telegram 1 for a correct communication between the inverter G120 and the PLC.

Initially the speed control running with a positive direction at a speed equal to half the maximum speed will be tested.

The first two activities are the addressing (figure 3.4) and the variables assignment with each communication bit of the interface. In this case, the interface used is represented in table 3.1 and table 3.2.



(Fig. 3.4 Addressing)

| Control_Interface | Siemens interface control word |
|---|---|
| Control_Start | #PZD.CONTROL_WORD."STW1 1.0" |
| Control_JobStart | |
| Control_noJOB_STOP | |
| Control_NOSTOP | |
| Control_ControlFromPLC | #PZD.CONTROL_WORD."STW1 0.2" |
| Control_RUN | |
| Control_GoalPosition | |
| Control_SpeedPosition | |
| Control_OverrideV | |
| Control_OverridePosV | |
| Control_Acceleration | |

| | |
|---|---|
| Control_Deceleration | |
| Control_GoalSpeed_LU | #PZD.CONTROL_WORD.STW2 |
| Control_GoalSpeed_Hz | |
| Control_GoalSpeed_rpm | |
| Control_PositioningMode | |
| Control_ReferenceSearchStart | |
| Control_PositiveDir | |
| Control_NegativeDir | |
| Control_RefType | |
| Control_ReferenceSearchDir | |
| Control_Jog1 | |
| Control_Jog2 | |
| Control_AlarmReset | #PZD.CONTROL_WORD."STW1 1.7" |
| Control_SetReferencePoint | |
| Control_SpeedDirection | #PZD.CONTROL_WORD."STW1 0.3" |
| Control_EnableSpeed | #PZD.CONTROL_WORD."STW1 1.6" |
| Control_OFF2 | #PZD.CONTROL_WORD."STW1 1.1" |
| Control_OFF3 | #PZD.CONTROL_WORD."STW1 1.2" |
| Control_EnableOperation | #PZD.CONTROL_WORD."STW1 1.3" |
| Control_EnableRamp | #PZD.CONTROL_WORD."STW1 1.4" |
| Control_ContinueRamp | #PZD.CONTROL_WORD."STW1 1.5" |
| Control_EnablePos | |
| Control_RockMOPIn | |
| Control_RockMOPDec | |
| Control_RockJOG | |
| Control_RockNegative | |
| Control_RockPositive | |
| Control_Stop | |

(Table 3.1 Interface between Drive output signals of R101_Control_Interface” block and Standard Telegram 1 control words)

| Status_Interface | Siemens interface status word |
|---|---|
| Status_Alarm | #PZD.STATUS_WORD."ZSW1 1.7" |
| Status_Ready | #PZD.STATUS_WORD."ZSW1 1.2" |
| Status_Blocked | #PZD.STATUS_WORD."ZSW1 1.3" |
| Status_AxisEnabled | #PZD.STATUS_WORD."ZSW1 1.0" |
| Status_PositiveDir | |
| Status_NegativeDir | |
| Status_Direction | |
| Status_ReferenceDone | |
| Status_SpeedReached | #PZD.STATUS_WORD."ZSW1 0.2" |
| Status_PositionReached | |
| Status_AlarmCode | |
| Status_WarningCode | |
| Status_ActualSpeed | #PZD.STATUS_WORD.ZSW2 |
| Status_ActualPosition | |

(Table 3.2 Interface between Drive input signals of R101_Control_Interface" block and Standard Telegram 1 status words)

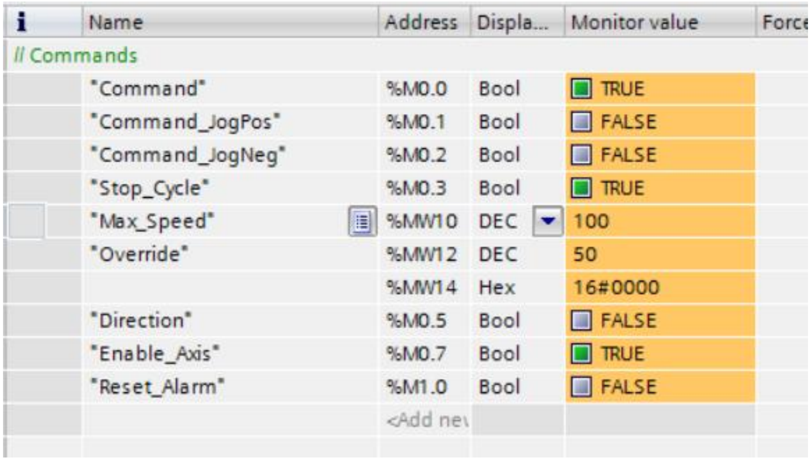
The third task is the setting of the type of control that must be executed and the setting of its mode.

In this case the parameters to be set in the block "R101_Control_Interface" are:

- "IN_CmdSpeed" = 1;
- "IN_StopCycle" = 1;
- "IN_Direction" = 0;
- "IN_MaxSpeed" = 100 [%];
- "IN_Override" = 50 [%];
- "IN_ReseAlarm" = 0;
- "IN_Enable_Axis" = 1.

It should be noted that the "IN_MaxSpeed" value is different from the maximum engine speed, but it is equal to 100. This peculiarity is caused by fact that the inverter calculates the wanted speed according to the parameter "p2000" of the list of experts. In the specific case of the test, the speed will be 50% of the value of the parameter "p2000".

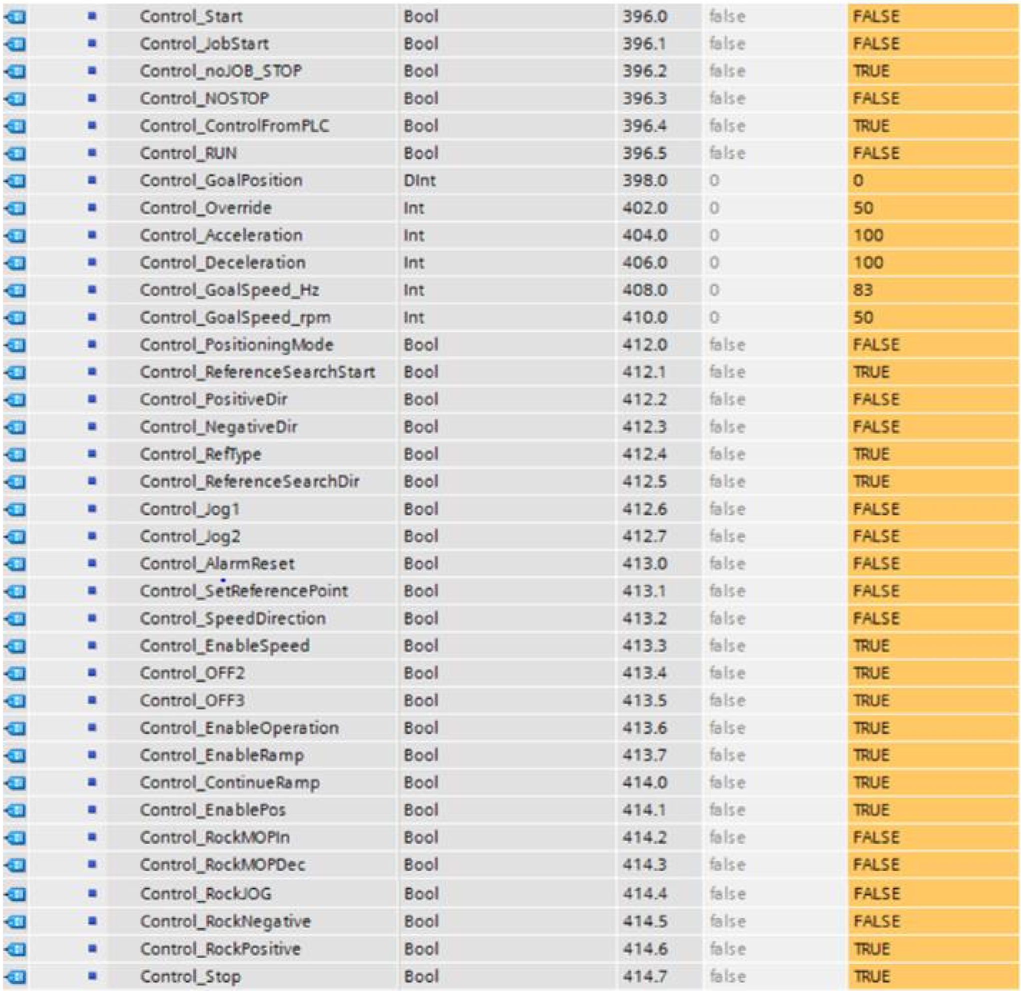
Figure 3.5 represents the commands that the user has applied.



| i | Name | Address | Displa... | Monitor value | Force |
|---|---|---|---|---|---|
| // Commands | | | | | |
| | "Command" | %M0.0 | Bool | 🟩 TRUE | |
| | "Command_JogPos" | %M0.1 | Bool | □ FALSE | |
| | "Command_JogNeg" | %M0.2 | Bool | □ FALSE | |
| | "Stop_Cycle" | %M0.3 | Bool | 🟩 TRUE | |
| | "Max_Speed" | %MW10 | DEC | 100 | |
| | "Override" | %MW12 | DEC | 50 | |
| | | %MW14 | Hex | 16#0000 | |
| | "Direction" | %M0.5 | Bool | □ FALSE | |
| | "Enable_Axis" | %M0.7 | Bool | 🟩 TRUE | |
| | "Reset_Alarm" | %M1.0 | Bool | □ FALSE | |
| | | <Add nev | | | |

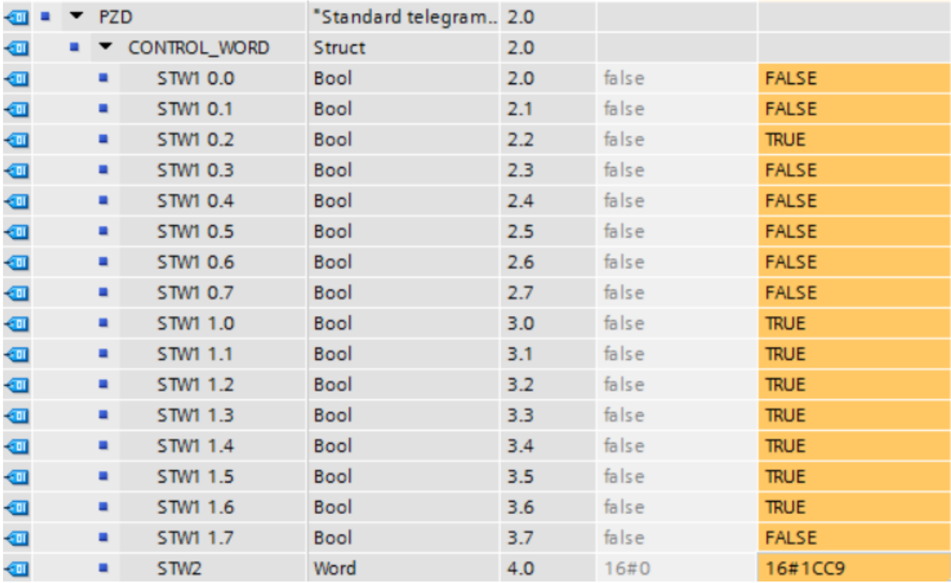
(Fig. 3.5 User-setted commands)

The block processes the user's commands and calculates the drive output variables (figure 3.6). Each Drive output variable is connected to internal structure (figure 3.7).



| | | | | |
|---|---|---|---|---|
| Control_Start | Bool | 396.0 | false | FALSE |
| Control_JobStart | Bool | 396.1 | false | FALSE |
| Control_noJOB_STOP | Bool | 396.2 | false | TRUE |
| Control_NOSTOP | Bool | 396.3 | false | FALSE |
| Control_ControlFromPLC | Bool | 396.4 | false | TRUE |
| Control_RUN | Bool | 396.5 | false | FALSE |
| Control_GoalPosition | DInt | 398.0 | 0 | 0 |
| Control_Override | Int | 402.0 | 0 | 50 |
| Control_Acceleration | Int | 404.0 | 0 | 100 |
| Control_Deceleration | Int | 406.0 | 0 | 100 |
| Control_GoalSpeed_Hz | Int | 408.0 | 0 | 83 |
| Control_GoalSpeed_rpm | Int | 410.0 | 0 | 50 |
| Control_PositioningMode | Bool | 412.0 | false | FALSE |
| Control_ReferenceSearchStart | Bool | 412.1 | false | TRUE |
| Control_PositiveDir | Bool | 412.2 | false | FALSE |
| Control_NegativeDir | Bool | 412.3 | false | FALSE |
| Control_RefType | Bool | 412.4 | false | TRUE |
| Control_ReferenceSearchDir | Bool | 412.5 | false | TRUE |
| Control_Jog1 | Bool | 412.6 | false | FALSE |
| Control_Jog2 | Bool | 412.7 | false | FALSE |
| Control_AlarmReset | Bool | 413.0 | false | FALSE |
| Control_SetReferencePoint | Bool | 413.1 | false | FALSE |
| Control_SpeedDirection | Bool | 413.2 | false | FALSE |
| Control_EnableSpeed | Bool | 413.3 | false | TRUE |
| Control_OFF2 | Bool | 413.4 | false | TRUE |
| Control_OFF3 | Bool | 413.5 | false | TRUE |
| Control_EnableOperation | Bool | 413.6 | false | TRUE |
| Control_EnableRamp | Bool | 413.7 | false | TRUE |
| Control_ContinueRamp | Bool | 414.0 | false | TRUE |
| Control_EnablePos | Bool | 414.1 | false | TRUE |
| Control_RockMOPIn | Bool | 414.2 | false | FALSE |
| Control_RockMOPDec | Bool | 414.3 | false | FALSE |
| Control_RockJOG | Bool | 414.4 | false | FALSE |
| Control_RockNegative | Bool | 414.5 | false | FALSE |
| Control_RockPositive | Bool | 414.6 | false | TRUE |
| Control_Stop | Bool | 414.7 | false | TRUE |

(Fig. 3.6 The values of drive outputs)



| | | | | |
|---|---|---|---|---|
| PZD | "Standard telegram.. | 2.0 | | |
| CONTROL_WORD | Struct | 2.0 | | |
| STW1 0.0 | Bool | 2.0 | false | FALSE |
| STW1 0.1 | Bool | 2.1 | false | FALSE |
| STW1 0.2 | Bool | 2.2 | false | TRUE |
| STW1 0.3 | Bool | 2.3 | false | FALSE |
| STW1 0.4 | Bool | 2.4 | false | FALSE |
| STW1 0.5 | Bool | 2.5 | false | FALSE |
| STW1 0.6 | Bool | 2.6 | false | FALSE |
| STW1 0.7 | Bool | 2.7 | false | FALSE |
| STW1 1.0 | Bool | 3.0 | false | TRUE |
| STW1 1.1 | Bool | 3.1 | false | TRUE |
| STW1 1.2 | Bool | 3.2 | false | TRUE |
| STW1 1.3 | Bool | 3.3 | false | TRUE |
| STW1 1.4 | Bool | 3.4 | false | TRUE |
| STW1 1.5 | Bool | 3.5 | false | TRUE |
| STW1 1.6 | Bool | 3.6 | false | TRUE |
| STW1 1.7 | Bool | 3.7 | false | FALSE |
| STW2 | Word | 4.0 | 16#0 | 16#1CC9 |

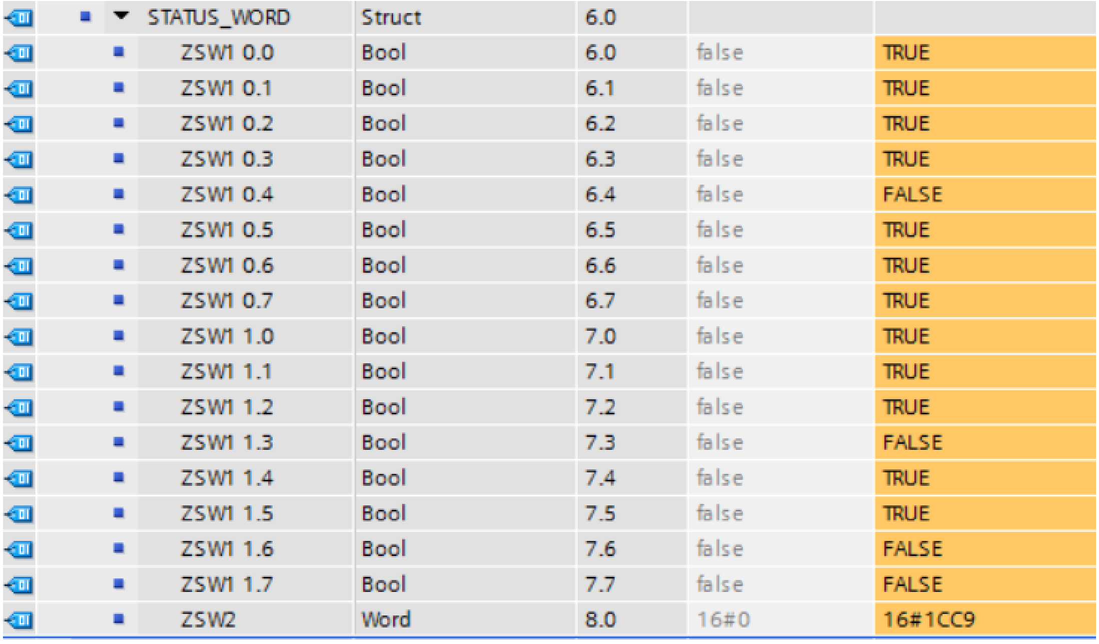
(Fig. 3.7 The values of command words of drive)

Drive output variables values are copied into the drive structure. In this way, the drive activation bits are set correctly.

It should be noted that the structure of figure 3.7 is similar to the command words of Standard Telegram 1 (appendix A). Indeed, the only difference is the bytes position that consists of the command words. This difference depends on how Siemens devices read words. Specifically, Siemens devices reverse the priority of the word bytes. In other words, firstly Siemens devices read the second byte of the word and then the first byte of the word. This difference is only present for Siemens devices.

At this point the motor starts spinning with a positive direction and at a rotational speed equal to 675 rpm.

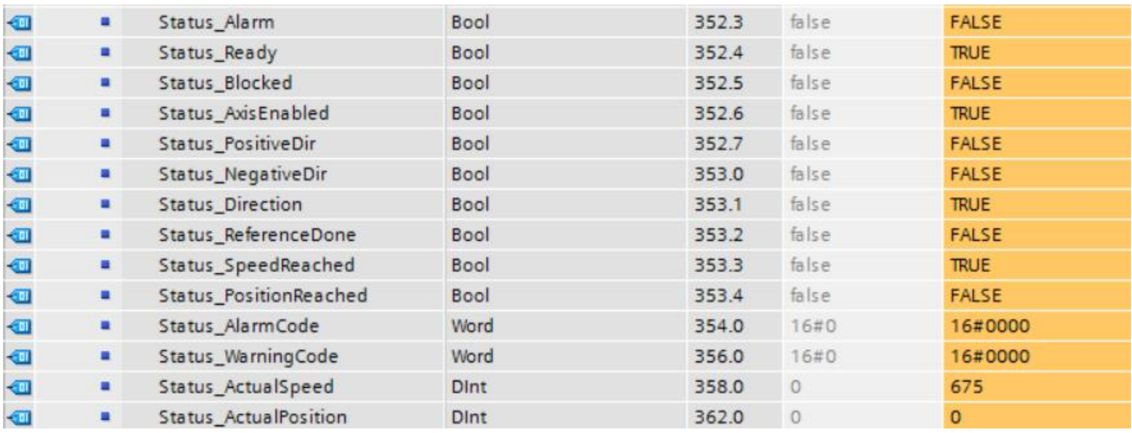
The operation of the drive can be monitored through the values of the drive status words.



| | | | | |
|---|---|---|---|---|
| STATUS_WORD | Struct | 6.0 | | |
| ZSW1 0.0 | Bool | 6.0 | false | TRUE |
| ZSW1 0.1 | Bool | 6.1 | false | TRUE |
| ZSW1 0.2 | Bool | 6.2 | false | TRUE |
| ZSW1 0.3 | Bool | 6.3 | false | TRUE |
| ZSW1 0.4 | Bool | 6.4 | false | FALSE |
| ZSW1 0.5 | Bool | 6.5 | false | TRUE |
| ZSW1 0.6 | Bool | 6.6 | false | TRUE |
| ZSW1 0.7 | Bool | 6.7 | false | TRUE |
| ZSW1 1.0 | Bool | 7.0 | false | TRUE |
| ZSW1 1.1 | Bool | 7.1 | false | TRUE |
| ZSW1 1.2 | Bool | 7.2 | false | TRUE |
| ZSW1 1.3 | Bool | 7.3 | false | FALSE |
| ZSW1 1.4 | Bool | 7.4 | false | TRUE |
| ZSW1 1.5 | Bool | 7.5 | false | TRUE |
| ZSW1 1.6 | Bool | 7.6 | false | FALSE |
| ZSW1 1.7 | Bool | 7.7 | false | FALSE |
| ZSW2 | Word | 8.0 | 16#0 | 16#1CC9 |

(Fig. 3.8 The values of status words of drive)

In a manner similar to command words, the structure of figure 3.8 is similar to the status words of Standard Telegram 1 (appendix A). These parameters provide the drive states and they must be transmitted to the interface block. As a result, drive status words are copied into the internal structure (figure 3.8). The most important bits of the internal structure status words are connected to the drive input signals (figure 3.9).



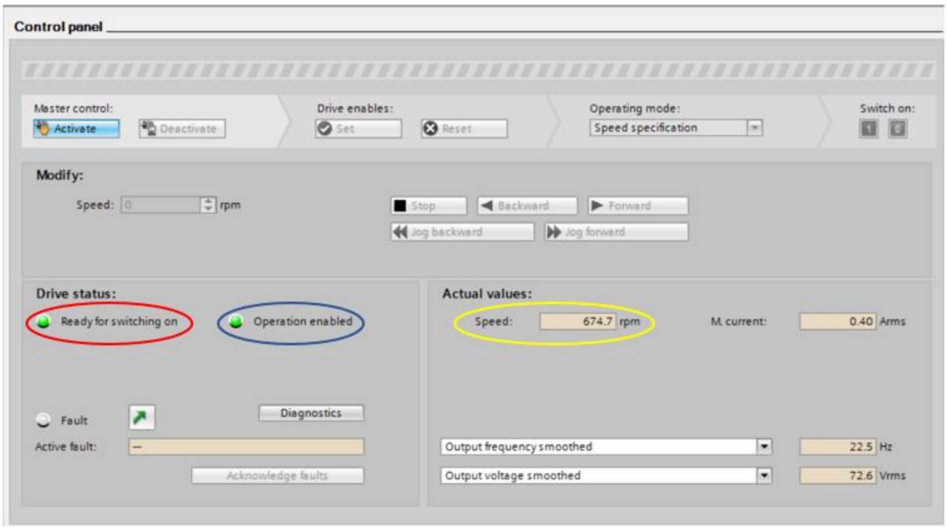
| | | | | |
|---|---|---|---|---|
| Status_Alarm | Bool | 352.3 | false | FALSE |
| Status_Ready | Bool | 352.4 | false | TRUE |
| Status_Blocked | Bool | 352.5 | false | FALSE |
| Status_AxisEnabled | Bool | 352.6 | false | TRUE |
| Status_PositiveDir | Bool | 352.7 | false | FALSE |
| Status_NegativeDir | Bool | 353.0 | false | FALSE |
| Status_Direction | Bool | 353.1 | false | TRUE |
| Status_ReferenceDone | Bool | 353.2 | false | FALSE |
| Status_SpeedReached | Bool | 353.3 | false | TRUE |
| Status_PositionReached | Bool | 353.4 | false | FALSE |
| Status_AlarmCode | Word | 354.0 | 16#0 | 16#0000 |
| Status_WarningCode | Word | 356.0 | 16#0 | 16#0000 |
| Status_ActualSpeed | DInt | 358.0 | 0 | 675 |
| Status_ActualPosition | DInt | 362.0 | 0 | 0 |

(Fig. 3.9 The values of drive input signals.)

From the image 3.9 a correct operation of the drive is evident:

- drive is ready state (Status_Ready = TRUE);
- the measured speed is 675 [rpm] (Status_ActualSpeed = 675);
- the direction is positive (Status_Direction = TRUE).

Siemens control panel demonstrates the speed of the engine (figure 3.10).



Operating mode: Speed specification

Modify:

Speed: 0 rpm

Drive status:

● Ready for switching on

● Operation enabled

○ Fault

Active fault: –

Actual values:

Speed: 674.7 rpm

M. current: 0.40 Arms

Output frequency smoothed 22.5 Hz

Output voltage smoothed 72.6 Vrms

(Fig. 3.10 Siemens control panel.)

Figure 3.10 shows that the drive is ready and it is running:

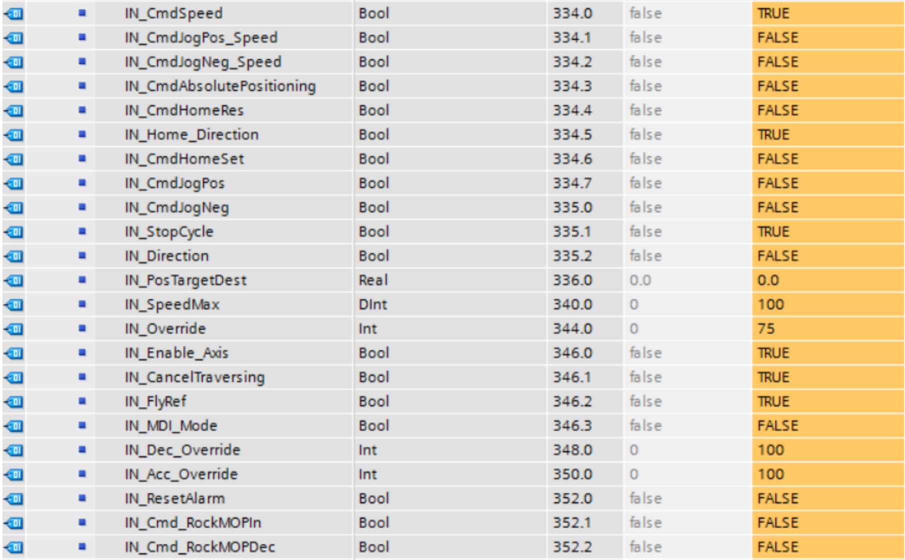
- ready for switching on LED is green (red circle);
- operation enabled LED is green (blue circle).

Speed is 675 [rpm] (equation 3.1)

$$1350 * \left(\frac{50}{100}\right) = 675\ [rpm] \qquad (3.1)$$

The value recorded by the control panel (yellow circle) is equal to about calculated value, so drive works properly.

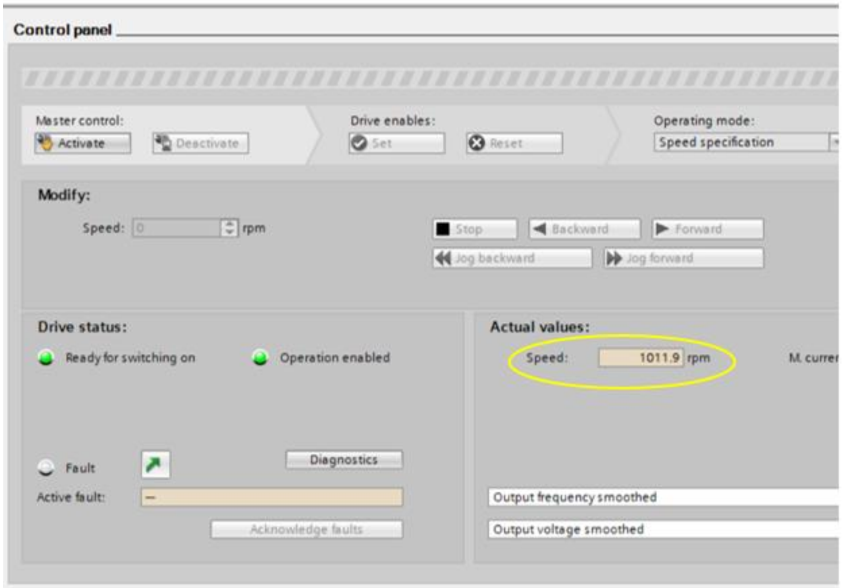
If the speed increases from 50% to 75% by means of the commands intended for the user, the speed changes to about 1012 rpm (figure 3.11).



| | | | | |
|---|---|---|---|---|
| IN_CmdSpeed | Bool | 334.0 | false | TRUE |
| IN_CmdJogPos_Speed | Bool | 334.1 | false | FALSE |
| IN_CmdJogNeg_Speed | Bool | 334.2 | false | FALSE |
| IN_CmdAbsolutePositioning | Bool | 334.3 | false | FALSE |
| IN_CmdHomeRes | Bool | 334.4 | false | FALSE |
| IN_Home_Direction | Bool | 334.5 | false | TRUE |
| IN_CmdHomeSet | Bool | 334.6 | false | FALSE |
| IN_CmdJogPos | Bool | 334.7 | false | FALSE |
| IN_CmdJogNeg | Bool | 335.0 | false | FALSE |
| IN_StopCycle | Bool | 335.1 | false | TRUE |
| IN_Direction | Bool | 335.2 | false | FALSE |
| IN_PosTargetDest | Real | 336.0 | 0.0 | 0.0 |
| IN_SpeedMax | DInt | 340.0 | 0 | 100 |
| IN_Override | Int | 344.0 | 0 | 75 |
| IN_Enable_Axis | Bool | 346.0 | false | TRUE |
| IN_CancelTraversing | Bool | 346.1 | false | TRUE |
| IN_FlyRef | Bool | 346.2 | false | TRUE |
| IN_MDI_Mode | Bool | 346.3 | false | FALSE |
| IN_Dec_Override | Int | 348.0 | 0 | 100 |
| IN_Acc_Override | Int | 350.0 | 0 | 100 |
| IN_ResetAlarm | Bool | 352.0 | false | FALSE |
| IN_Cmd_RockMOPIn | Bool | 352.1 | false | FALSE |
| IN_Cmd_RockMOPDec | Bool | 352.2 | false | FALSE |

(Fig. 3.11 New values of user inputs variables.)

Siemens control panel demonstrates the new engine speed (figure 3.12).



Control panel

Master control:

Drive enables:

Operating mode: Speed specification

Modify:

Speed: 0 rpm

Drive status:

● Ready for switching on

● Operation enabled

○ Fault

Active fault: –

Actual values:

Speed: 1011.9 rpm

Output frequency smoothed

Output voltage smoothed

(Fig. 3.12 Siemens control panel.)

It should be noted that the velocity is 1012 [rpm] (figure 3.12) for equation 3.2.

$$1350 * \left(\frac{80}{100}\right) = 1012\ [rpm] \qquad (3.2)$$

The value recorded by the control panel (yellow circle) is equal to about calculated value, so drive works properly.

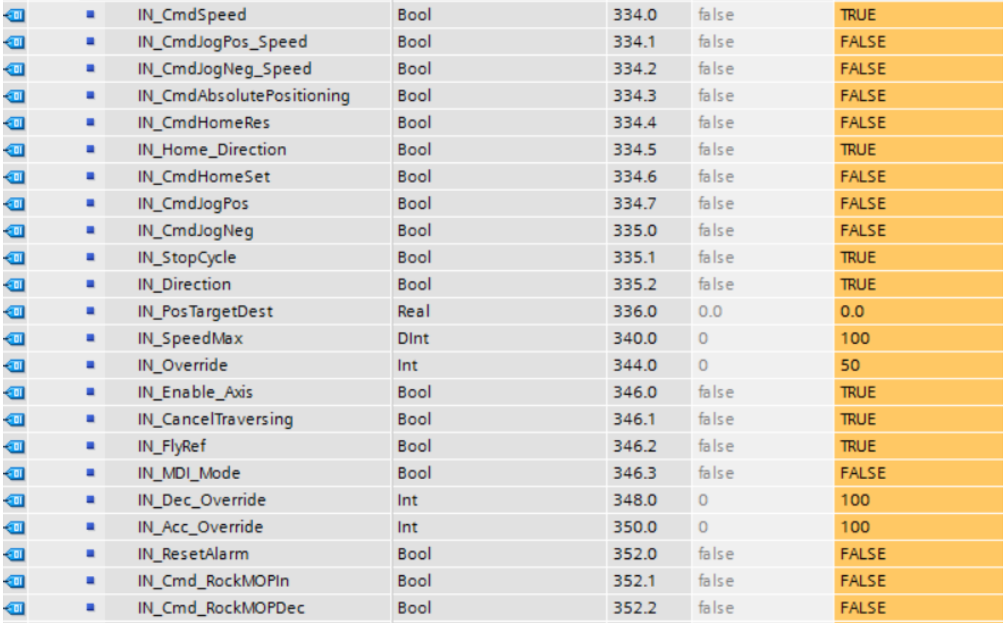
The second phase of the test concerns the speed control analysis with a negative direction at a speed equal to half the maximum speed.

Similarly to before, the programmer must set the desired control type.

The parameters to be set in the block "R101_Control_Interface" are:

- "IN_Cmdspeed" = 1;
- "IN_StopCycle" = 1;
- "IN_Direction" = 1;
- "IN_MaxSpeed"  = 100 [%];
- "IN_Override" = 50 [%];
- "IN_ReseAlarm" = 0;
- "IN_Enable_Axis" = 1.

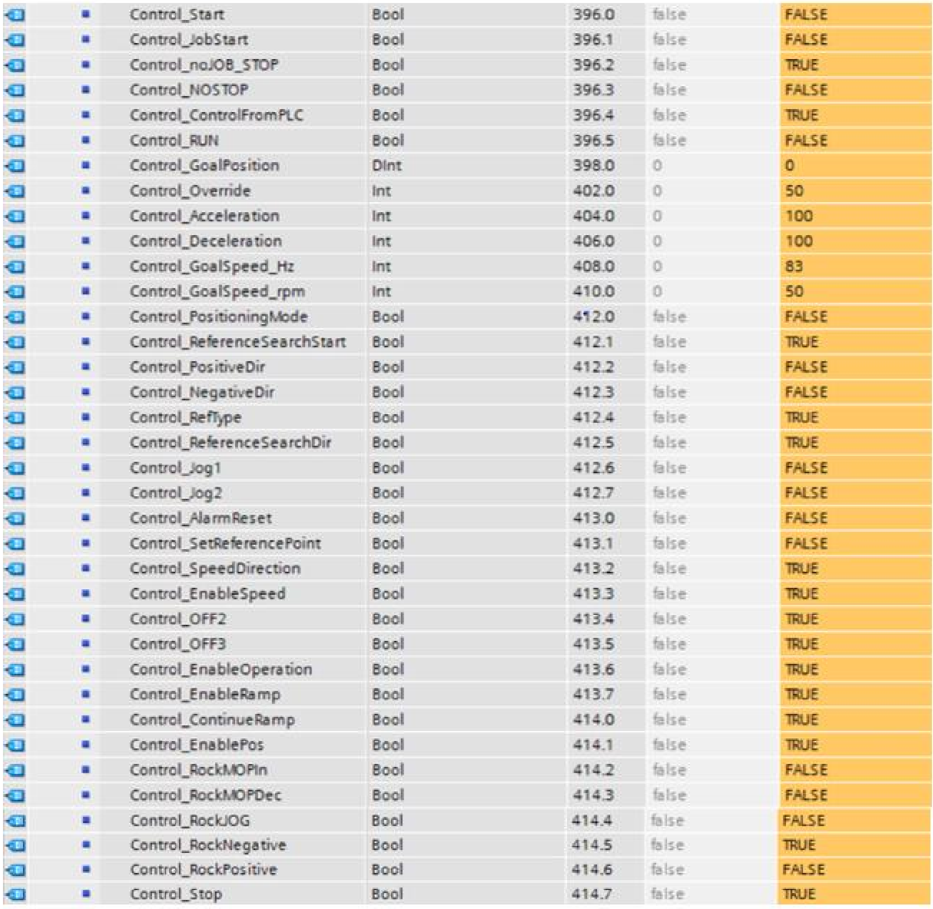
The operating mode must be set by using the user inputs. Figure 3.13 represents the correct values for user inputs to run a speed control with a negative direction at a speed equal to half the maximum speed.



| | | | | |
|---|---|---|---|---|
| IN_CmdSpeed | Bool | 334.0 | false | TRUE |
| IN_CmdJogPos_Speed | Bool | 334.1 | false | FALSE |
| IN_CmdJogNeg_Speed | Bool | 334.2 | false | FALSE |
| IN_CmdAbsolutePositioning | Bool | 334.3 | false | FALSE |
| IN_CmdHomeRes | Bool | 334.4 | false | FALSE |
| IN_Home_Direction | Bool | 334.5 | false | TRUE |
| IN_CmdHomeSet | Bool | 334.6 | false | FALSE |
| IN_CmdJogPos | Bool | 334.7 | false | FALSE |
| IN_CmdJogNeg | Bool | 335.0 | false | FALSE |
| IN_StopCycle | Bool | 335.1 | false | TRUE |
| IN_Direction | Bool | 335.2 | false | TRUE |
| IN_PosTargetDest | Real | 336.0 | 0.0 | 0.0 |
| IN_SpeedMax | DInt | 340.0 | 0 | 100 |
| IN_Override | Int | 344.0 | 0 | 50 |
| IN_Enable_Axis | Bool | 346.0 | false | TRUE |
| IN_CancelTraversing | Bool | 346.1 | false | TRUE |
| IN_FlyRef | Bool | 346.2 | false | TRUE |
| IN_MDI_Mode | Bool | 346.3 | false | FALSE |
| IN_Dec_Override | Int | 348.0 | 0 | 100 |
| IN_Acc_Override | Int | 350.0 | 0 | 100 |
| IN_ResetAlarm | Bool | 352.0 | false | FALSE |
| IN_Cmd_RockMOPIn | Bool | 352.1 | false | FALSE |
| IN_Cmd_RockMOPDec | Bool | 352.2 | false | FALSE |

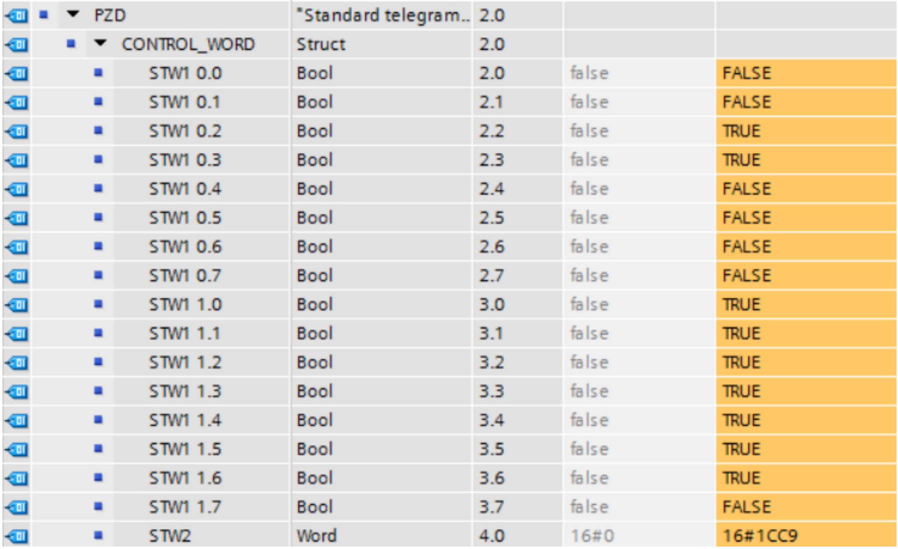
(Fig. 3.13 The values of user inputs)

The R101 Control_Interface block reads the user's commands and generates the correct drive output values (figure 3.14). Each Drive output variable is connected to the internal structure (figure 3.15).



| | | | | |
|---|---|---|---|---|
| Control_Start | Bool | 396.0 | false | FALSE |
| Control_JobStart | Bool | 396.1 | false | FALSE |
| Control_noJOB_STOP | Bool | 396.2 | false | TRUE |
| Control_NOSTOP | Bool | 396.3 | false | FALSE |
| Control_ControlFromPLC | Bool | 396.4 | false | TRUE |
| Control_RUN | Bool | 396.5 | false | FALSE |
| Control_GoalPosition | DInt | 398.0 | 0 | 0 |
| Control_Override | Int | 402.0 | 0 | 50 |
| Control_Acceleration | Int | 404.0 | 0 | 100 |
| Control_Deceleration | Int | 406.0 | 0 | 100 |
| Control_GoalSpeed_Hz | Int | 408.0 | 0 | 83 |
| Control_GoalSpeed_rpm | Int | 410.0 | 0 | 50 |
| Control_PositioningMode | Bool | 412.0 | false | FALSE |
| Control_ReferenceSearchStart | Bool | 412.1 | false | TRUE |
| Control_PositiveDir | Bool | 412.2 | false | FALSE |
| Control_NegativeDir | Bool | 412.3 | false | FALSE |
| Control_RefType | Bool | 412.4 | false | TRUE |
| Control_ReferenceSearchDir | Bool | 412.5 | false | TRUE |
| Control_Jog1 | Bool | 412.6 | false | FALSE |
| Control_Jog2 | Bool | 412.7 | false | FALSE |
| Control_AlarmReset | Bool | 413.0 | false | FALSE |
| Control_SetReferencePoint | Bool | 413.1 | false | FALSE |
| Control_SpeedDirection | Bool | 413.2 | false | TRUE |
| Control_EnableSpeed | Bool | 413.3 | false | TRUE |
| Control_OFF2 | Bool | 413.4 | false | TRUE |
| Control_OFF3 | Bool | 413.5 | false | TRUE |
| Control_EnableOperation | Bool | 413.6 | false | TRUE |
| Control_EnableRamp | Bool | 413.7 | false | TRUE |
| Control_ContinueRamp | Bool | 414.0 | false | TRUE |
| Control_EnablePos | Bool | 414.1 | false | TRUE |
| Control_RockMOPIn | Bool | 414.2 | false | FALSE |
| Control_RockMOPDec | Bool | 414.3 | false | FALSE |
| Control_RockJOG | Bool | 414.4 | false | FALSE |
| Control_RockNegative | Bool | 414.5 | false | TRUE |
| Control_RockPositive | Bool | 414.6 | false | FALSE |
| Control_Stop | Bool | 414.7 | false | TRUE |

(Fig. 3.14 The values of drive outputs)



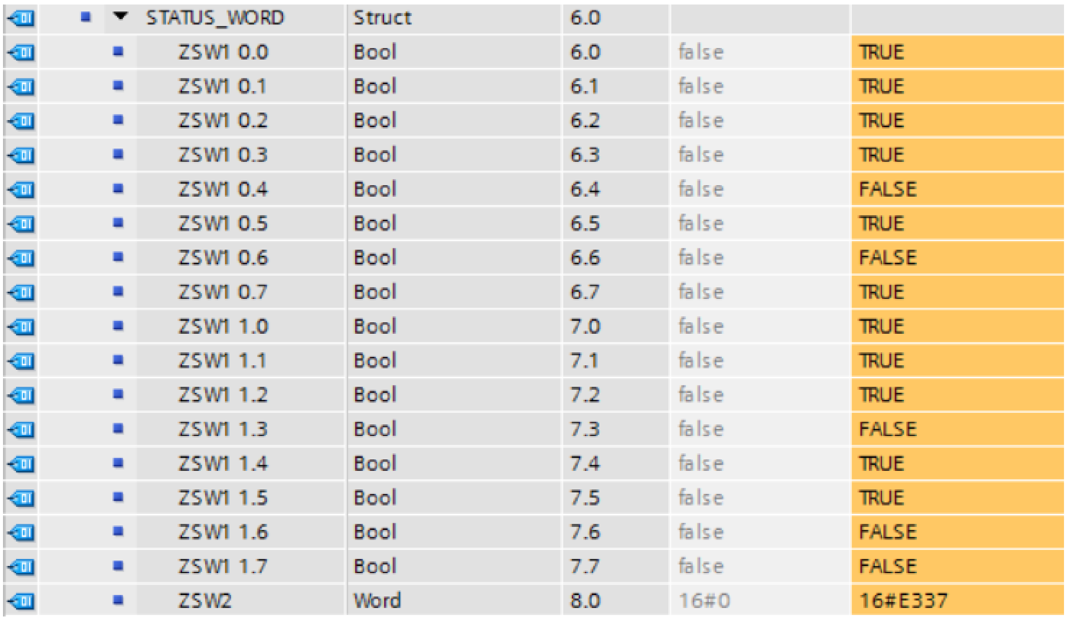
| | | | | |
|---|---|---|---|---|
| PZD | "Standard telegram.. | 2.0 | | |
| CONTROL_WORD | Struct | 2.0 | | |
| STW1 0.0 | Bool | 2.0 | false | FALSE |
| STW1 0.1 | Bool | 2.1 | false | FALSE |
| STW1 0.2 | Bool | 2.2 | false | TRUE |
| STW1 0.3 | Bool | 2.3 | false | TRUE |
| STW1 0.4 | Bool | 2.4 | false | FALSE |
| STW1 0.5 | Bool | 2.5 | false | FALSE |
| STW1 0.6 | Bool | 2.6 | false | FALSE |
| STW1 0.7 | Bool | 2.7 | false | FALSE |
| STW1 1.0 | Bool | 3.0 | false | TRUE |
| STW1 1.1 | Bool | 3.1 | false | TRUE |
| STW1 1.2 | Bool | 3.2 | false | TRUE |
| STW1 1.3 | Bool | 3.3 | false | TRUE |
| STW1 1.4 | Bool | 3.4 | false | TRUE |
| STW1 1.5 | Bool | 3.5 | false | TRUE |
| STW1 1.6 | Bool | 3.6 | false | TRUE |
| STW1 1.7 | Bool | 3.7 | false | FALSE |
| STW2 | Word | 4.0 | 16#0 | 16#1CC9 |

(Fig. 3.15 The values of command words of drive)

Drive output variables values are copied into the drive structure. In this way, the drive activation bits are set correctly.

As a result, the drive runs the speed control and the engine starts spinning with a negative direction and at a speed equal to 675 rpm.

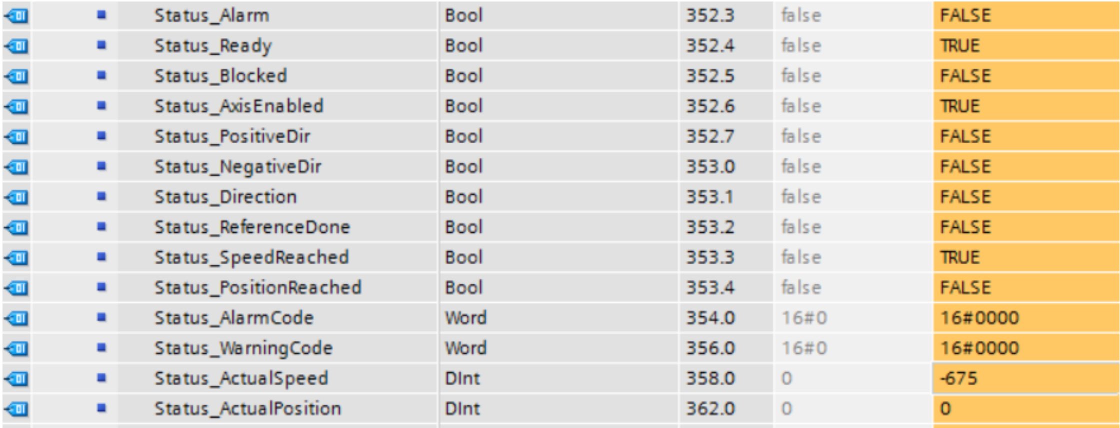
Drive status words are copied into the internal structure. Proper operation is demonstrated by the word status of the drive (figure 3.16).



| | | | | |
|---|---|---|---|---|
| ▼ STATUS_WORD | Struct | 6.0 | | |
| ZSW1 0.0 | Bool | 6.0 | false | TRUE |
| ZSW1 0.1 | Bool | 6.1 | false | TRUE |
| ZSW1 0.2 | Bool | 6.2 | false | TRUE |
| ZSW1 0.3 | Bool | 6.3 | false | TRUE |
| ZSW1 0.4 | Bool | 6.4 | false | FALSE |
| ZSW1 0.5 | Bool | 6.5 | false | TRUE |
| ZSW1 0.6 | Bool | 6.6 | false | FALSE |
| ZSW1 0.7 | Bool | 6.7 | false | TRUE |
| ZSW1 1.0 | Bool | 7.0 | false | TRUE |
| ZSW1 1.1 | Bool | 7.1 | false | TRUE |
| ZSW1 1.2 | Bool | 7.2 | false | TRUE |
| ZSW1 1.3 | Bool | 7.3 | false | FALSE |
| ZSW1 1.4 | Bool | 7.4 | false | TRUE |
| ZSW1 1.5 | Bool | 7.5 | false | TRUE |
| ZSW1 1.6 | Bool | 7.6 | false | FALSE |
| ZSW1 1.7 | Bool | 7.7 | false | FALSE |
| ZSW2 | Word | 8.0 | 16#0 | 16#E337 |

(Fig. 3.16 The values of status words of drive)

The figure 3.17 reports the most important information that block can read.



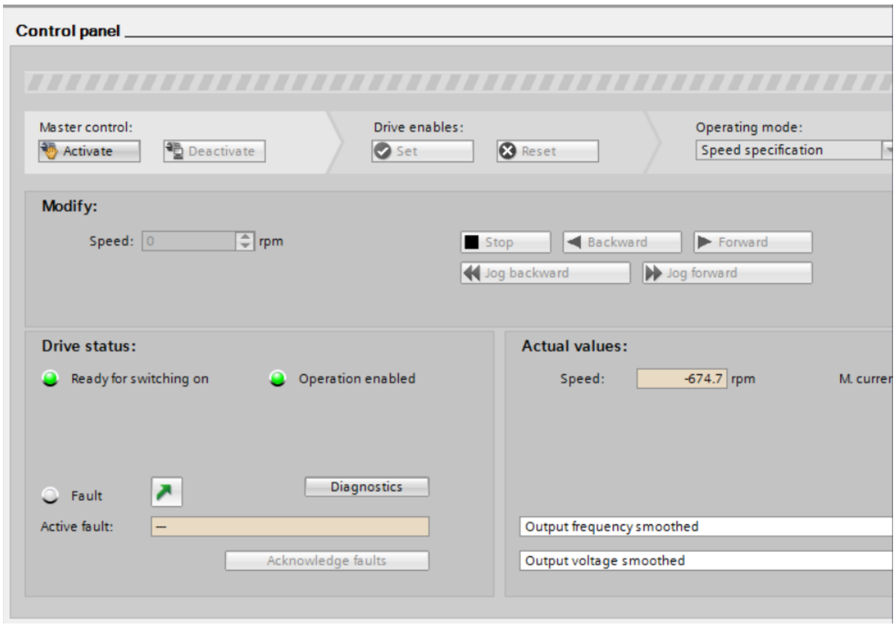
| | | | | |
|---|---|---|---|---|
| Status_Alarm | Bool | 352.3 | false | FALSE |
| Status_Ready | Bool | 352.4 | false | TRUE |
| Status_Blocked | Bool | 352.5 | false | FALSE |
| Status_AxisEnabled | Bool | 352.6 | false | TRUE |
| Status_PositiveDir | Bool | 352.7 | false | FALSE |
| Status_NegativeDir | Bool | 353.0 | false | FALSE |
| Status_Direction | Bool | 353.1 | false | FALSE |
| Status_ReferenceDone | Bool | 353.2 | false | FALSE |
| Status_SpeedReached | Bool | 353.3 | false | TRUE |
| Status_PositionReached | Bool | 353.4 | false | FALSE |
| Status_AlarmCode | Word | 354.0 | 16#0 | 16#0000 |
| Status_WarningCode | Word | 356.0 | 16#0 | 16#0000 |
| Status_ActualSpeed | DInt | 358.0 | 0 | -675 |
| Status_ActualPosition | DInt | 362.0 | 0 | 0 |

(Fig. 3.17 The values of drive input signals.)

From the image 3.17 a correct operation of the drive is evident:

- drive is ready state (Status_Ready = TRUE);
- the measured speed is -675 [rpm] (Status_ActualSpeed = -675);
- the direction is negative (Status_Direction = FALSE).

Siemens control panel demonstrates the speed at which the engine moves (figure 3.18).



Control panel

Master control:

Drive enables:

Operating mode: Speed specification

Modify:

Speed: 0 rpm

Drive status:

● Ready for switching on ● Operation enabled

○ Fault

Active fault: –

Actual values:

Speed: -674.7 rpm

M. curre

Output frequency smoothed

Output voltage smoothed

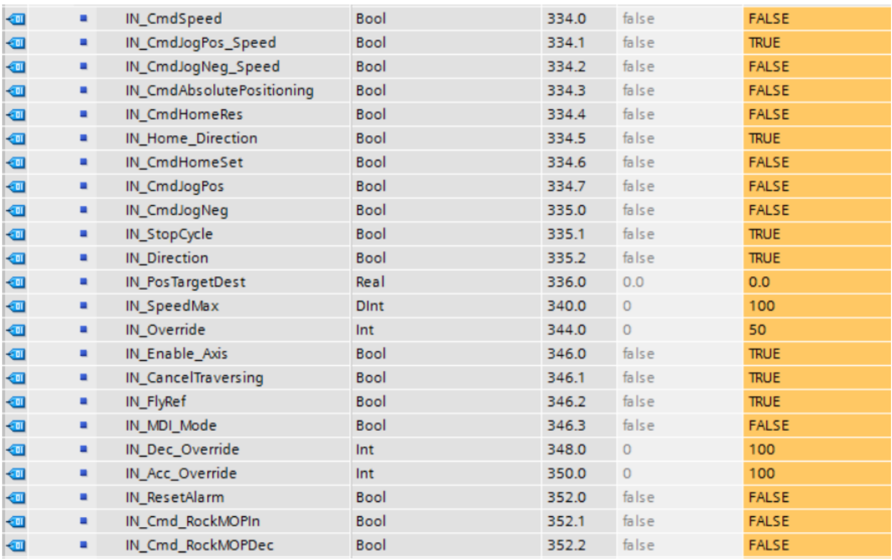
(Fig. 3.18 Siemens control panel.)

The third phase of the test is about study of the manual control with a positive direction. The speed of the jog mode is handled in the "R101_Control_Interface" block and independent of the “IN_Override” input. The jog mode speed override is equal to 20% and so the speed is 20% of the maximum speed.

The parameters to be set in the block "R101_Control_Interface" are:

- “IN_CmdJogPos_Speed” = 1;
- “IN_StopCycle” = 1;
- “IN_Enable_Axis” = 1;
- “IN_ReseAlarm” = 0.

The operating mode must be set by using the user inputs. Figure 3.19 represents the correct values for user inputs to perform positive jog mode.

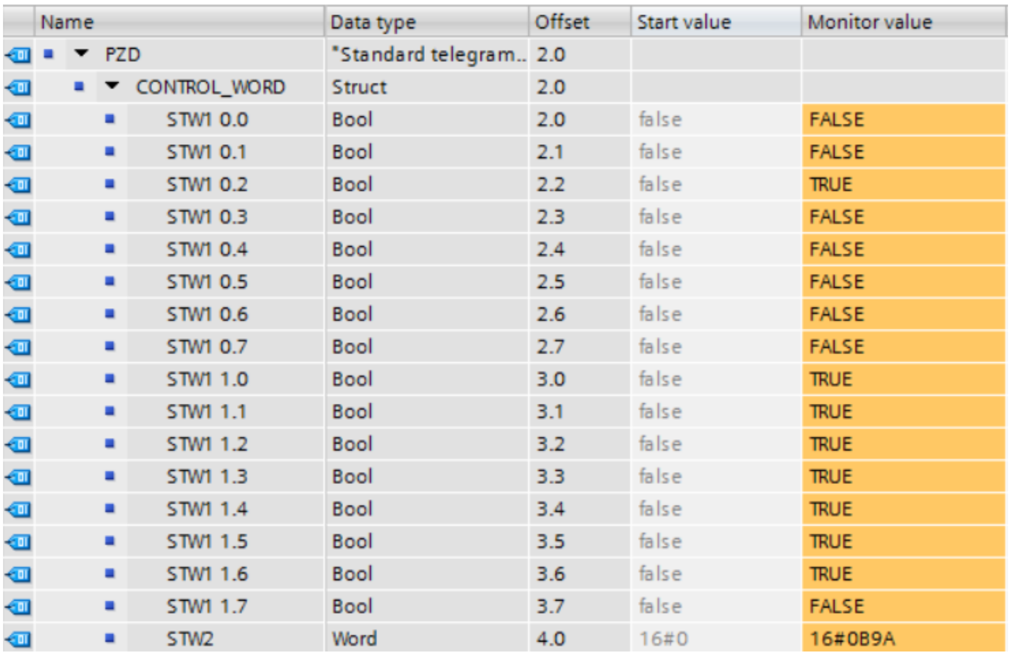


| | | | | |
|---|---|---|---|---|
| IN_CmdSpeed | Bool | 334.0 | false | FALSE |
| IN_CmdJogPos_Speed | Bool | 334.1 | false | TRUE |
| IN_CmdJogNeg_Speed | Bool | 334.2 | false | FALSE |
| IN_CmdAbsolutePositioning | Bool | 334.3 | false | FALSE |
| IN_CmdHomeRes | Bool | 334.4 | false | FALSE |
| IN_Home_Direction | Bool | 334.5 | false | TRUE |
| IN_CmdHomeSet | Bool | 334.6 | false | FALSE |
| IN_CmdJogPos | Bool | 334.7 | false | FALSE |
| IN_CmdJogNeg | Bool | 335.0 | false | FALSE |
| IN_StopCycle | Bool | 335.1 | false | TRUE |
| IN_Direction | Bool | 335.2 | false | TRUE |
| IN_PosTargetDest | Real | 336.0 | 0.0 | 0.0 |
| IN_SpeedMax | DInt | 340.0 | 0 | 100 |
| IN_Override | Int | 344.0 | 0 | 50 |
| IN_Enable_Axis | Bool | 346.0 | false | TRUE |
| IN_CancelTraversing | Bool | 346.1 | false | TRUE |
| IN_FlyRef | Bool | 346.2 | false | TRUE |
| IN_MDI_Mode | Bool | 346.3 | false | FALSE |
| IN_Dec_Override | Int | 348.0 | 0 | 100 |
| IN_Acc_Override | Int | 350.0 | 0 | 100 |
| IN_ResetAlarm | Bool | 352.0 | false | FALSE |
| IN_Cmd_RockMOPIn | Bool | 352.1 | false | FALSE |
| IN_Cmd_RockMOPDec | Bool | 352.2 | false | FALSE |

(Fig. 3.19 The values of user inputs)

It should be noted that "IN_CmdSpeed" input is equal to "FALSE", and "IN_CmdJogPos_Speed" is equal to "TRUE".

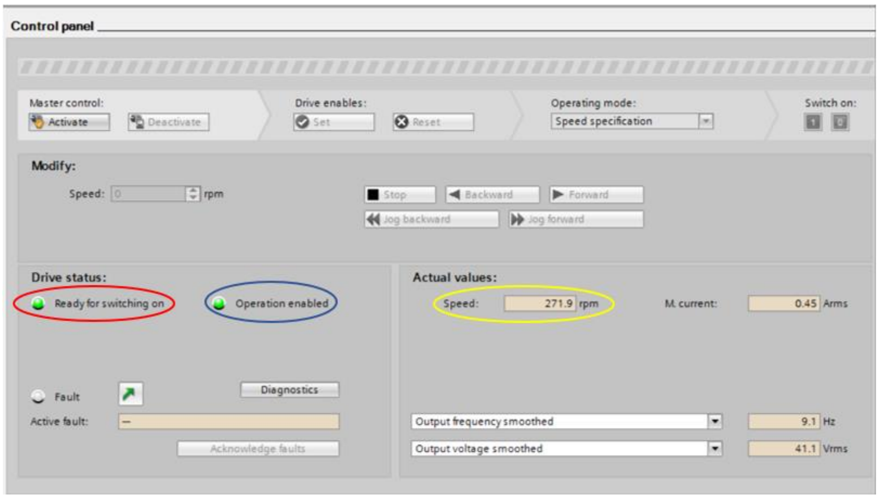
Jog mode for the speed control is an additional mode. As a result, jog mode is implemented within the "R101_Control_Interface" block by exploiting the same bits of the drive structure that have been used previously for speed control (figure 3.20). The procedure of control execution is similar to that of the speed control and the only difference is the management of the user inputs.



| Name | Data type | Offset | Start value | Monitor value |
|---|---|---|---|---|
| ▼ PZD | "Standard telegram.. | 2.0 | | |
| ▼ CONTROL_WORD | Struct | 2.0 | | |
| STW1 0.0 | Bool | 2.0 | false | FALSE |
| STW1 0.1 | Bool | 2.1 | false | FALSE |
| STW1 0.2 | Bool | 2.2 | false | TRUE |
| STW1 0.3 | Bool | 2.3 | false | FALSE |
| STW1 0.4 | Bool | 2.4 | false | FALSE |
| STW1 0.5 | Bool | 2.5 | false | FALSE |
| STW1 0.6 | Bool | 2.6 | false | FALSE |
| STW1 0.7 | Bool | 2.7 | false | FALSE |
| STW1 1.0 | Bool | 3.0 | false | TRUE |
| STW1 1.1 | Bool | 3.1 | false | TRUE |
| STW1 1.2 | Bool | 3.2 | false | TRUE |
| STW1 1.3 | Bool | 3.3 | false | TRUE |
| STW1 1.4 | Bool | 3.4 | false | TRUE |
| STW1 1.5 | Bool | 3.5 | false | TRUE |
| STW1 1.6 | Bool | 3.6 | false | TRUE |
| STW1 1.7 | Bool | 3.7 | false | FALSE |
| STW2 | Word | 4.0 | 16#0 | 16#0B9A |

(Fig. 3.20 The values of command words of drive)

Correct operation of the jog mode is shown by the following figure.



Operating mode: Speed specification

Speed: 0 rpm

● Ready for switching on

● Operation enabled

○ Fault

Active fault: –

Speed: 271.9 rpm

M. current: 0.45 Arms

Output frequency smoothed: 9.1 Hz

Output voltage smoothed: 41.1 Vrms

(Fig. 3.21 Siemens control panel.)

Figure 3.21 shows that the drive is ready and it is running:

- ready for switching on LED is green (red circle);
- operation enabled LED is green (blue circle).

The speed of the jog mode must be 270 [rpm] (equation 3.3).

$$1350 * \left(\frac{20}{100}\right) = 270\ [rpm] \qquad (3.3)$$

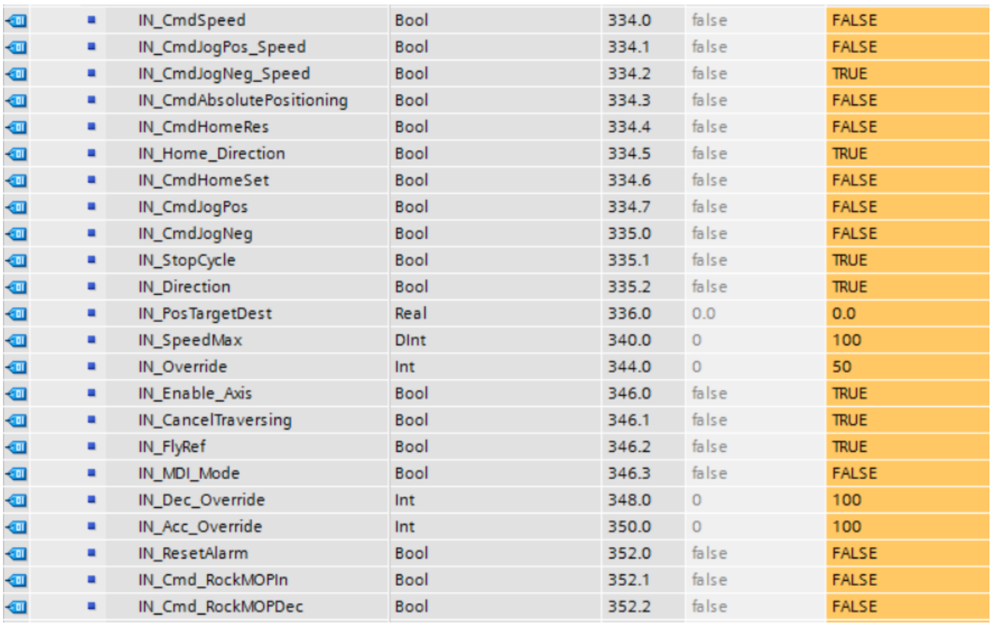
The control panel provides an actual value (yellow circle) that is about the calculated value. In addition, it should be noted that actual values (yellow circle) does not depend on the "IN_Override" value (figure 3.19), but it depends on the value inside the block. As a result, the drive works properly.

The fourth phase of the test consists in the analysis of the jog mode with a negative direction. Here again, the speed of the jog mode is set at 20% of the maximum speed by the "R101_Control_Interface" block and it is independent of the input override.

The parameters to be set in the block "R101_Control_Interface" are:

- “IN_CmdJogNeg_Speed” = 1;
- “IN_StopCycle” = 1;
- “IN_Enable_Axis” = 1;
- “IN_ReseAlarm” = 0.

The operating mode must be set by using the user inputs. Figure 3.22 represents the correct values for user inputs to perform positive jog mode.

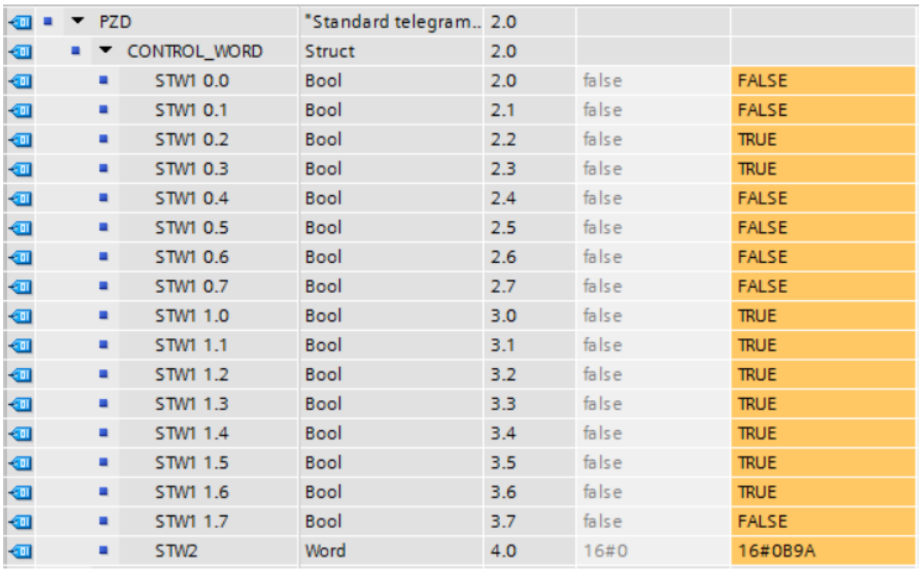


| | | | | |
|---|---|---|---|---|
| IN_CmdSpeed | Bool | 334.0 | false | FALSE |
| IN_CmdJogPos_Speed | Bool | 334.1 | false | FALSE |
| IN_CmdJogNeg_Speed | Bool | 334.2 | false | TRUE |
| IN_CmdAbsolutePositioning | Bool | 334.3 | false | FALSE |
| IN_CmdHomeRes | Bool | 334.4 | false | FALSE |
| IN_Home_Direction | Bool | 334.5 | false | TRUE |
| IN_CmdHomeSet | Bool | 334.6 | false | FALSE |
| IN_CmdJogPos | Bool | 334.7 | false | FALSE |
| IN_CmdJogNeg | Bool | 335.0 | false | FALSE |
| IN_StopCycle | Bool | 335.1 | false | TRUE |
| IN_Direction | Bool | 335.2 | false | TRUE |
| IN_PosTargetDest | Real | 336.0 | 0.0 | 0.0 |
| IN_SpeedMax | DInt | 340.0 | 0 | 100 |
| IN_Override | Int | 344.0 | 0 | 50 |
| IN_Enable_Axis | Bool | 346.0 | false | TRUE |
| IN_CancelTraversing | Bool | 346.1 | false | TRUE |
| IN_FlyRef | Bool | 346.2 | false | TRUE |
| IN_MDI_Mode | Bool | 346.3 | false | FALSE |
| IN_Dec_Override | Int | 348.0 | 0 | 100 |
| IN_Acc_Override | Int | 350.0 | 0 | 100 |
| IN_ResetAlarm | Bool | 352.0 | false | FALSE |
| IN_Cmd_RockMOPIn | Bool | 352.1 | false | FALSE |
| IN_Cmd_RockMOPDec | Bool | 352.2 | false | FALSE |

(Fig. 3.22 The values of user inputs)

It should be noted that "IN_CmdSpeed" inputs is equal to "FALSE", and "IN_CmdJogNeg_Speed" is equal to "TRUE".

The procedure of control execution is similar to control process of the positive jog mode, indeed the only difference is the management of user inputs (figure 3.23).



| | | | | |
|---|---|---|---|---|
| ▾ PZD | "Standard telegram.. | 2.0 | | |
| ▾ CONTROL_WORD | Struct | 2.0 | | |
| STW1 0.0 | Bool | 2.0 | false | FALSE |
| STW1 0.1 | Bool | 2.1 | false | FALSE |
| STW1 0.2 | Bool | 2.2 | false | TRUE |
| STW1 0.3 | Bool | 2.3 | false | TRUE |
| STW1 0.4 | Bool | 2.4 | false | FALSE |
| STW1 0.5 | Bool | 2.5 | false | FALSE |
| STW1 0.6 | Bool | 2.6 | false | FALSE |
| STW1 0.7 | Bool | 2.7 | false | FALSE |
| STW1 1.0 | Bool | 3.0 | false | TRUE |
| STW1 1.1 | Bool | 3.1 | false | TRUE |
| STW1 1.2 | Bool | 3.2 | false | TRUE |
| STW1 1.3 | Bool | 3.3 | false | TRUE |
| STW1 1.4 | Bool | 3.4 | false | TRUE |
| STW1 1.5 | Bool | 3.5 | false | TRUE |
| STW1 1.6 | Bool | 3.6 | false | TRUE |
| STW1 1.7 | Bool | 3.7 | false | FALSE |
| STW2 | Word | 4.0 | 16#0 | 16#0B9A |

(Fig. 3.23 The values of command words of drive)

Control panel in figure 3.24 shows that the drive is ready and it is running:

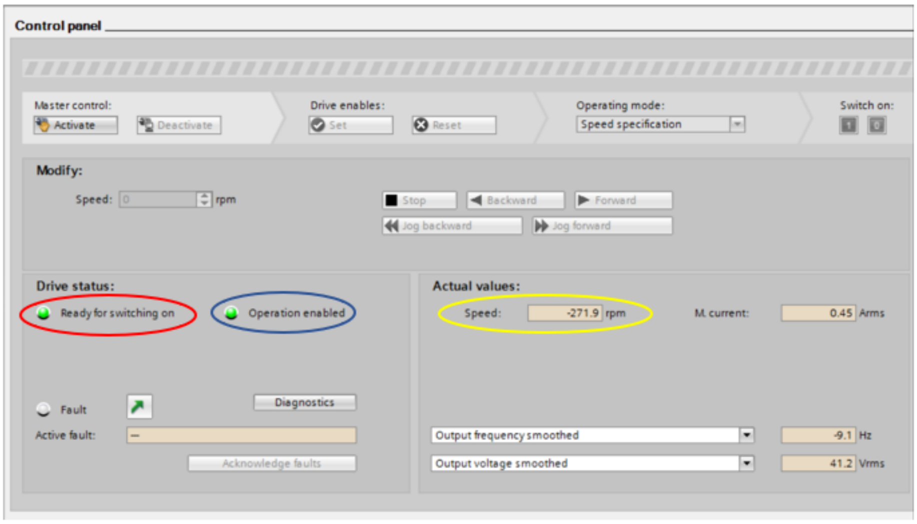
• ready for switching on LED is green (red circle);

• operation enabled LED is green (blue circle).

The speed of the jog mode must be -270 [rpm] (equation 3.4) because the direction is negative.

$$-1350 * \left(\frac{20}{100}\right) = -270\ [rpm] \qquad (3.4)$$

The control panel provides a value equal to about the calculated value, so drive works properly. In addition, it should be noted that actual values (yellow circle) is independent of the "IN_Override" value (figure 3.22) and it depends only on the internal value of the block "R101_Control_Interface". As a result, the drive works rightly.



Operating mode: Speed specification

Speed: 0 rpm

Active fault: –

Speed: -271.9 rpm

M. current: 0.45 Arms

Output frequency smoothed: -9.1 Hz

Output voltage smoothed: 41.2 Vrms

(Fig. 3.24 Siemens control panel.)

In conclusion, the test showed that the communication between Siemens PLC and Siemens inverter drive by means of the Standard Telegram 1 was performed correctly. Particularly, all the modes of operation required for the program have been executed completely.

## 3.3 *Test about Siemens Telegram 352*

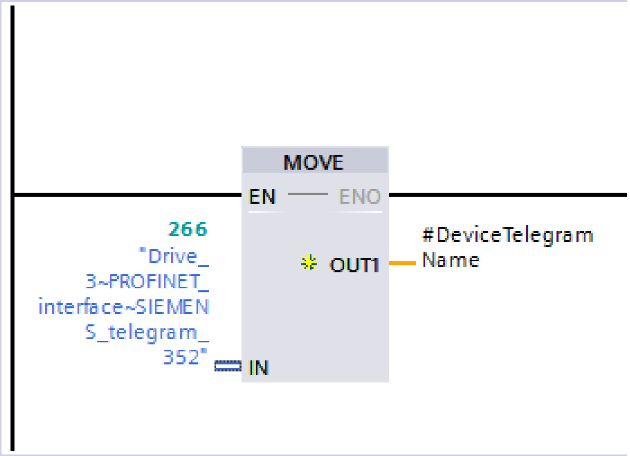
The test of the Siemens Telegram 352 was performed by using the Siemens PLC 1511 F (figure 3.1), an electric motor and drive G120 without external encoder (figure 3.2).

The characteristics of the motor are described in figure 3.3.

The objective of the following test is the right functioning of the Standard Telegram 352 (appendix A) for a correct communication between the inverter G120 and the PLC.

The first phase of the test consists of a speed control with a positive direction at a speed equal to 80% of the maximum speed.

In a manner similar to test about the Standard Telegram 1, the first two tasks are the addressing (figure 3.25) and assigning variables to each communication bit of the interface. In this case, the interface used is represented in table 3.3 and table 3.4.



(Fig. 3.25 Addressing)

| Control_Interface | Siemens interface control word |
|---|---|
| Control_Start | #PZD.CONTROL_WORD."STW1 1.0" |
| Control_JobStart | |
| Control_noJOB_STOP | |
| Control_NOSTOP | |
| Control_ControlFromPLC | #PZD.CONTROL_WORD."STW1 0.2" |
| Control_RUN | |
| Control_GoalPosition | |
| Control_SpeedPosition | |
| Control_OverrideV | |
| Control_OverridePosV | |
| Control_Acceleration | |
| Control_Deceleration | |
| Control_GoalSpeed_LU | #PZD.CONTROL_WORD. NSOLL_A |
| Control_GoalSpeed_Hz | |

| | |
|---|---|
| Control_GoalSpeed_rpm | |
| Control_PositioningMode | |
| Control_ReferenceSearchStart | |
| Control_PositiveDir | |
| Control_NegativeDir | |
| Control_RefType | |
| Control_ReferenceSearchDir | |
| Control_Jog1 | |
| Control_Jog2 | |
| Control_AlarmReset | #PZD.CONTROL_WORD."STW1 1.7" |
| Control_SetReferencePoint | |
| Control_SpeedDirection | #PZD.CONTROL_WORD."STW1 0.3" |
| Control_EnableSpeed | #PZD.CONTROL_WORD."STW1 1.6" |
| Control_OFF2 | #PZD.CONTROL_WORD."STW1 1.1" |
| Control_OFF3 | #PZD.CONTROL_WORD."STW1 1.2" |
| Control_EnableOperation | #PZD.CONTROL_WORD."STW1 1.3" |
| Control_EnableRamp | #PZD.CONTROL_WORD."STW1 1.4" |
| Control_ContinueRamp | #PZD.CONTROL_WORD."STW1 1.5" |
| Control_EnablePos | |
| Control_RockMOPIn | |
| Control_RockMOPDec | |
| Control_RockJOG | |
| Control_RockNegative | |
| Control_RockPositive | |
| Control_Stop | |

(Table 3.3 Interface between Drive output signals of R101_Control_Interface” block and Siemens Telegram 352 control words)

| Status_Interface | Siemens interface status word |
|---|---|
| Status_Alarm | #PZD.STATUS_WORD."ZSW1 1.7" |
| Status_Ready | #PZD.STATUS_WORD."ZSW1 1.2" |
| Status_Blocked | #PZD.STATUS_WORD."ZSW1 1.3" |
| Status_AxisEnabled | #PZD.STATUS_WORD."ZSW1 1.0" |
| Status_PositiveDir | |
| Status_NegativeDir | |
| Status_Direction | |

| | |
|---|---|
| Status_ReferenceDone | |
| Status_SpeedReached | #PZD.STATUS_WORD."ZSW1 0.2" |
| Status_PositionReached | |
| Status_AlarmCode | #PZD.STATUS_WORD.ZSW5 |
| Status_WarningCode | #PZD.STATUS_WORD.ZSW6 |
| Status_ActualSpeed | #PZD.STATUS_WORD.ZSW2 |
| Status_ActualPosition | |

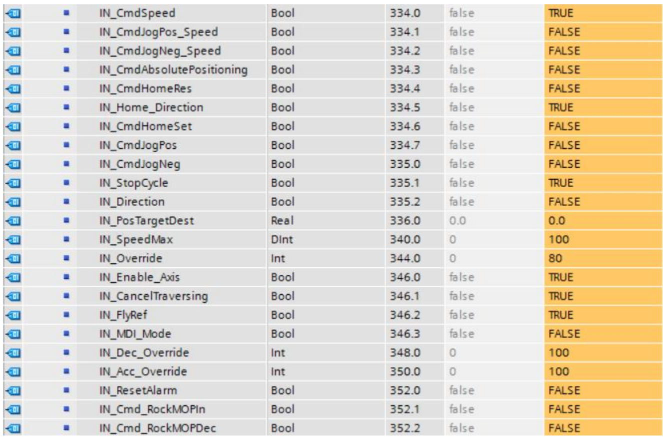
(Table 3.4 Interface between Drive input signals of R101_Control_Interface" block and Siemens Telegram 352 status words)

The third task is the setting of the type of control that must be executed and its mode. In this case the parameters to be set in the block "R101_Control_Interface" are:

- "IN_CmdSpeed" = 1;
- "IN_StopCycle" = 1;
- "IN_Direction" = 0;
- "IN_MaxSpeed" = 100 [%];
- "IN_Override" = 80 [%];
- "IN_ReseAlarm" = 0;
- "IN_Enable_Axis" = 1.

Also in this case, the inverter calculates the desired speed according to the parameter "p2000" of the expert list.

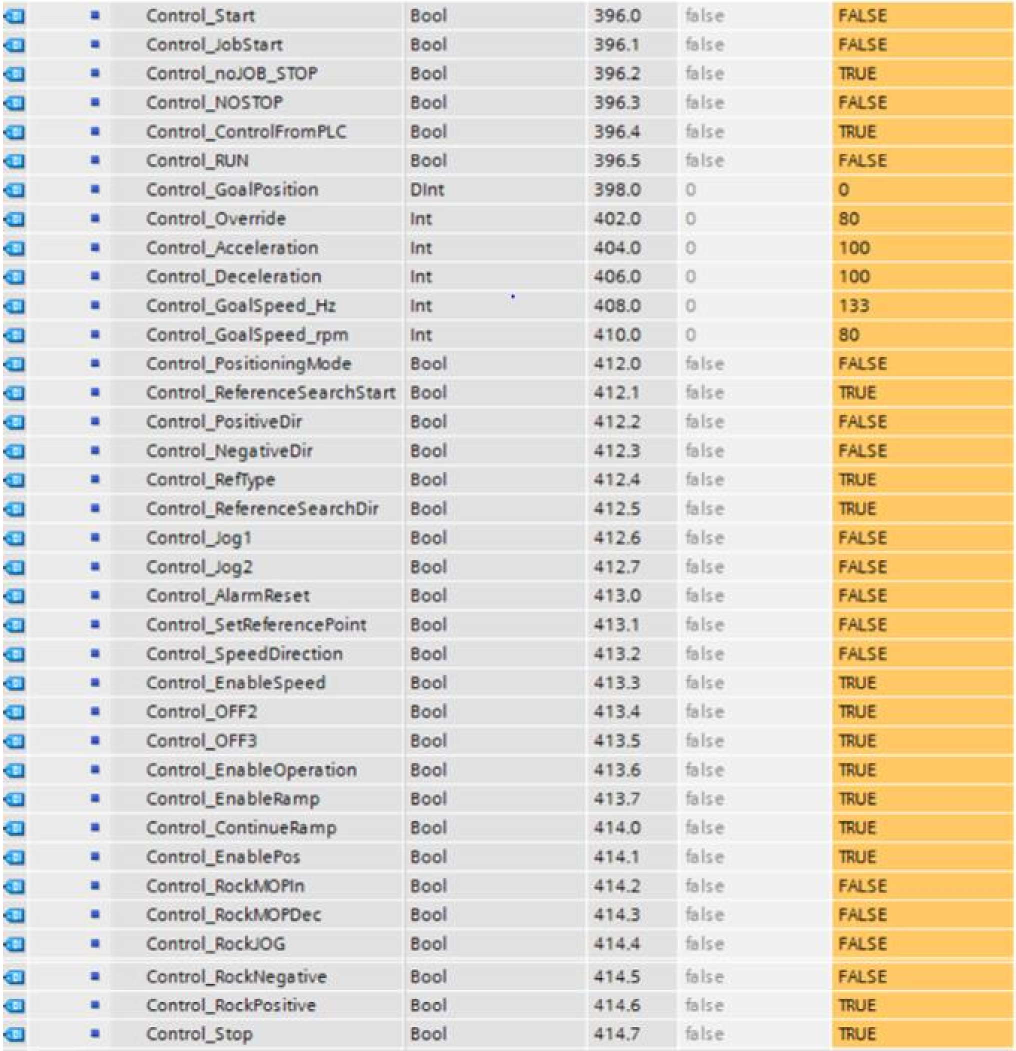
The operating mode must be set by using the user inputs. Figure 3.26 represents the correct values for user inputs to perform a speed control with a positive direction at a speed equal to 80% of the maximum speed.



| | | | | |
|---|---|---|---|---|
| IN_CmdSpeed | Bool | 334.0 | false | TRUE |
| IN_CmdJogPos_Speed | Bool | 334.1 | false | FALSE |
| IN_CmdJogNeg_Speed | Bool | 334.2 | false | FALSE |
| IN_CmdAbsolutePositioning | Bool | 334.3 | false | FALSE |
| IN_CmdHomeRes | Bool | 334.4 | false | FALSE |
| IN_Home_Direction | Bool | 334.5 | false | TRUE |
| IN_CmdHomeSet | Bool | 334.6 | false | FALSE |
| IN_CmdJogPos | Bool | 334.7 | false | FALSE |
| IN_CmdJogNeg | Bool | 335.0 | false | FALSE |
| IN_StopCycle | Bool | 335.1 | false | TRUE |
| IN_Direction | Bool | 335.2 | false | FALSE |
| IN_PosTargetDest | Real | 336.0 | 0.0 | 0.0 |
| IN_SpeedMax | DInt | 340.0 | 0 | 100 |
| IN_Override | Int | 344.0 | 0 | 80 |
| IN_Enable_Axis | Bool | 346.0 | false | TRUE |
| IN_CancelTraversing | Bool | 346.1 | false | TRUE |
| IN_FlyRef | Bool | 346.2 | false | TRUE |
| IN_MDI_Mode | Bool | 346.3 | false | FALSE |
| IN_Dec_Override | Int | 348.0 | 0 | 100 |
| IN_Acc_Override | Int | 350.0 | 0 | 100 |
| IN_ResetAlarm | Bool | 352.0 | false | FALSE |
| IN_Cmd_RockMOPIn | Bool | 352.1 | false | FALSE |
| IN_Cmd_RockMOPDec | Bool | 352.2 | false | FALSE |

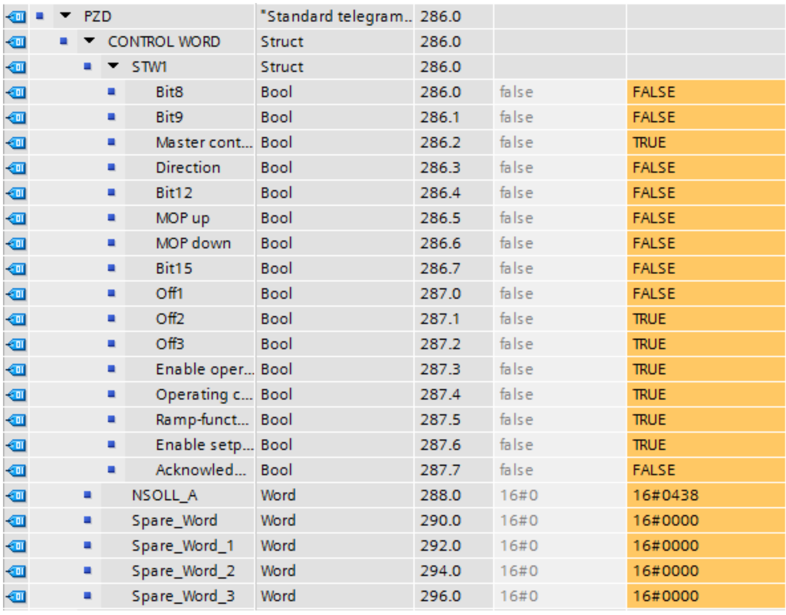
(Fig. 3.26 Values of user inputs variables)

The block processes the user's commands and calculates the drive output variables (figure 3.27). Each Drive output variable is connected to internal structure (figure 3.28).



| | | | | |
|---|---|---|---|---|
| Control_Start | Bool | 396.0 | false | FALSE |
| Control_JobStart | Bool | 396.1 | false | FALSE |
| Control_noJOB_STOP | Bool | 396.2 | false | TRUE |
| Control_NOSTOP | Bool | 396.3 | false | FALSE |
| Control_ControlFromPLC | Bool | 396.4 | false | TRUE |
| Control_RUN | Bool | 396.5 | false | FALSE |
| Control_GoalPosition | DInt | 398.0 | 0 | 0 |
| Control_Override | Int | 402.0 | 0 | 80 |
| Control_Acceleration | Int | 404.0 | 0 | 100 |
| Control_Deceleration | Int | 406.0 | 0 | 100 |
| Control_GoalSpeed_Hz | Int | 408.0 | 0 | 133 |
| Control_GoalSpeed_rpm | Int | 410.0 | 0 | 80 |
| Control_PositioningMode | Bool | 412.0 | false | FALSE |
| Control_ReferenceSearchStart | Bool | 412.1 | false | TRUE |
| Control_PositiveDir | Bool | 412.2 | false | FALSE |
| Control_NegativeDir | Bool | 412.3 | false | FALSE |
| Control_RefType | Bool | 412.4 | false | TRUE |
| Control_ReferenceSearchDir | Bool | 412.5 | false | TRUE |
| Control_Jog1 | Bool | 412.6 | false | FALSE |
| Control_Jog2 | Bool | 412.7 | false | FALSE |
| Control_AlarmReset | Bool | 413.0 | false | FALSE |
| Control_SetReferencePoint | Bool | 413.1 | false | FALSE |
| Control_SpeedDirection | Bool | 413.2 | false | FALSE |
| Control_EnableSpeed | Bool | 413.3 | false | TRUE |
| Control_OFF2 | Bool | 413.4 | false | TRUE |
| Control_OFF3 | Bool | 413.5 | false | TRUE |
| Control_EnableOperation | Bool | 413.6 | false | TRUE |
| Control_EnableRamp | Bool | 413.7 | false | TRUE |
| Control_ContinueRamp | Bool | 414.0 | false | TRUE |
| Control_EnablePos | Bool | 414.1 | false | TRUE |
| Control_RockMOPIn | Bool | 414.2 | false | FALSE |
| Control_RockMOPDec | Bool | 414.3 | false | FALSE |
| Control_RockJOG | Bool | 414.4 | false | FALSE |
| Control_RockNegative | Bool | 414.5 | false | FALSE |
| Control_RockPositive | Bool | 414.6 | false | TRUE |
| Control_Stop | Bool | 414.7 | false | TRUE |

(Fig. 3.27 The values of drive outputs)



| | | | | |
|---|---|---|---|---|
| PZD | "Standard telegram... | 286.0 | | |
| CONTROL WORD | Struct | 286.0 | | |
| STW1 | Struct | 286.0 | | |
| Bit8 | Bool | 286.0 | false | FALSE |
| Bit9 | Bool | 286.1 | false | FALSE |
| Master cont... | Bool | 286.2 | false | TRUE |
| Direction | Bool | 286.3 | false | FALSE |
| Bit12 | Bool | 286.4 | false | FALSE |
| MOP up | Bool | 286.5 | false | FALSE |
| MOP down | Bool | 286.6 | false | FALSE |
| Bit15 | Bool | 286.7 | false | FALSE |
| Off1 | Bool | 287.0 | false | FALSE |
| Off2 | Bool | 287.1 | false | TRUE |
| Off3 | Bool | 287.2 | false | TRUE |
| Enable oper... | Bool | 287.3 | false | TRUE |
| Operating c... | Bool | 287.4 | false | TRUE |
| Ramp-funct... | Bool | 287.5 | false | TRUE |
| Enable setp... | Bool | 287.6 | false | TRUE |
| Acknowled... | Bool | 287.7 | false | FALSE |
| NSOLL_A | Word | 288.0 | 16#0 | 16#0438 |
| Spare_Word | Word | 290.0 | 16#0 | 16#0000 |
| Spare_Word_1 | Word | 292.0 | 16#0 | 16#0000 |
| Spare_Word_2 | Word | 294.0 | 16#0 | 16#0000 |
| Spare_Word_3 | Word | 296.0 | 16#0 | 16#0000 |

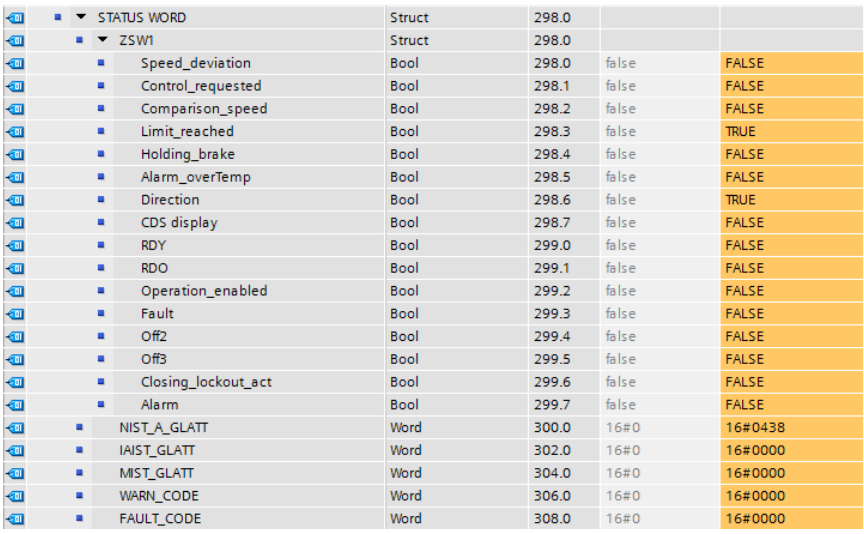
(Fig. 3.28 The values of command words of drive)

It should be noted that the structure of figure 3.28 has the same words of command structure of Siemens Telegram 352 (appendix A).

Drive output variables values are copied into the drive structure. In this way, the drive activation bits are set correctly.

At this point the motor starts spinning with a positive direction and at a rotational speed equal to 1080 rpm.

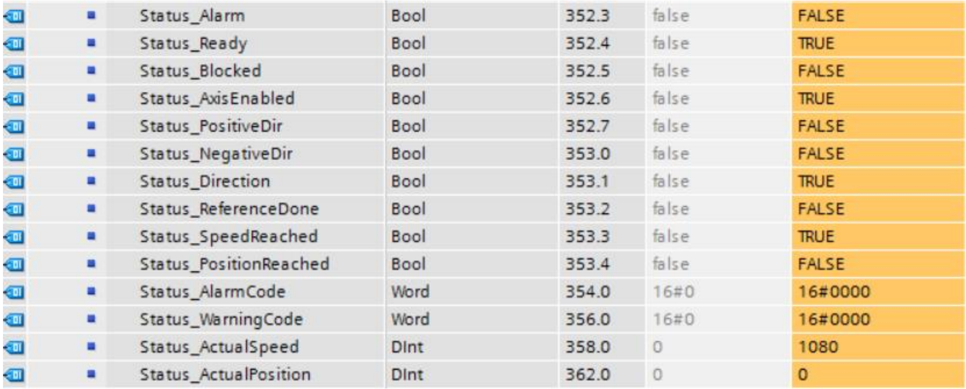
The operation of the drive can be monitored through the values of the drive status words.



| | | | | |
|---|---|---|---|---|
| STATUS WORD | Struct | 298.0 | | |
| ZSW1 | Struct | 298.0 | | |
| Speed_deviation | Bool | 298.0 | false | FALSE |
| Control_requested | Bool | 298.1 | false | FALSE |
| Comparison_speed | Bool | 298.2 | false | FALSE |
| Limit_reached | Bool | 298.3 | false | TRUE |
| Holding_brake | Bool | 298.4 | false | FALSE |
| Alarm_overTemp | Bool | 298.5 | false | FALSE |
| Direction | Bool | 298.6 | false | TRUE |
| CDS display | Bool | 298.7 | false | FALSE |
| RDY | Bool | 299.0 | false | FALSE |
| RDO | Bool | 299.1 | false | FALSE |
| Operation_enabled | Bool | 299.2 | false | FALSE |
| Fault | Bool | 299.3 | false | FALSE |
| Off2 | Bool | 299.4 | false | FALSE |
| Off3 | Bool | 299.5 | false | FALSE |
| Closing_lockout_act | Bool | 299.6 | false | FALSE |
| Alarm | Bool | 299.7 | false | FALSE |
| NIST_A_GLATT | Word | 300.0 | 16#0 | 16#0438 |
| IAIST_GLATT | Word | 302.0 | 16#0 | 16#0000 |
| MIST_GLATT | Word | 304.0 | 16#0 | 16#0000 |
| WARN_CODE | Word | 306.0 | 16#0 | 16#0000 |
| FAULT_CODE | Word | 308.0 | 16#0 | 16#0000 |

(Fig. 3.29 The values of status words of drive)

It should be noted that the structure of figure 3.29 is similar to the status words of the Siemens Telegram 352 (appendix A). These parameters provide the drive states and they must be transmitted to the interface block. As a result, drive status words are copied into the internal structure (figure 3.29). The most important bits of the internal structure status words are connected to the drive input signals (figure 3.30).

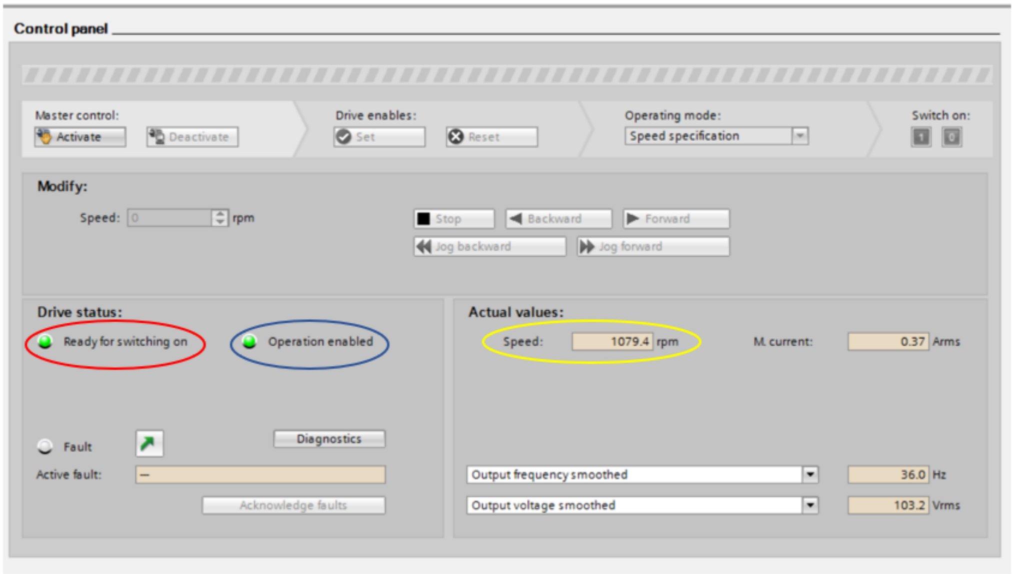


| | | | | |
|---|---|---|---|---|
| Status_Alarm | Bool | 352.3 | false | FALSE |
| Status_Ready | Bool | 352.4 | false | TRUE |
| Status_Blocked | Bool | 352.5 | false | FALSE |
| Status_AxisEnabled | Bool | 352.6 | false | TRUE |
| Status_PositiveDir | Bool | 352.7 | false | FALSE |
| Status_NegativeDir | Bool | 353.0 | false | FALSE |
| Status_Direction | Bool | 353.1 | false | TRUE |
| Status_ReferenceDone | Bool | 353.2 | false | FALSE |
| Status_SpeedReached | Bool | 353.3 | false | TRUE |
| Status_PositionReached | Bool | 353.4 | false | FALSE |
| Status_AlarmCode | Word | 354.0 | 16#0 | 16#0000 |
| Status_WarningCode | Word | 356.0 | 16#0 | 16#0000 |
| Status_ActualSpeed | DInt | 358.0 | 0 | 1080 |
| Status_ActualPosition | DInt | 362.0 | 0 | 0 |

(Fig. 3.30 The values of drive input signals.)

From the image 3.30 a correct functioning of the drive is evident. Specifically, drive is ready (Status_Ready = TRUE), there are not alarms (Status_AlarmCode = 16 # 0000) and there are not faults (Status_WarningCode = 16 # 0000). In addition, the measured speed is 1080 [rpm] (Status_ActualSpeed = 1080) and the direction is positive (Status_Direction = TRUE).

Siemens control panel demonstrates the speed of the engine (figure3.31).



Operating mode: Speed specification

Speed: 0 rpm

Speed: 1079.4 rpm

M. current: 0.37 Arms

Active fault: –

Output frequency smoothed 36.0 Hz

Output voltage smoothed 103.2 Vrms

(Fig. 3.31 Siemens control panel.)

Figure 3.31 shows that the drive is ready and it is running:

- ready for switching on LED is green (red circle);
- operation enabled LED is green (blue circle).

Speed is 1080 [rpm] (equation 3.5)

$$1350 * \left(\frac{80}{100}\right) = 1080\ [rpm] \tag{3.5}$$

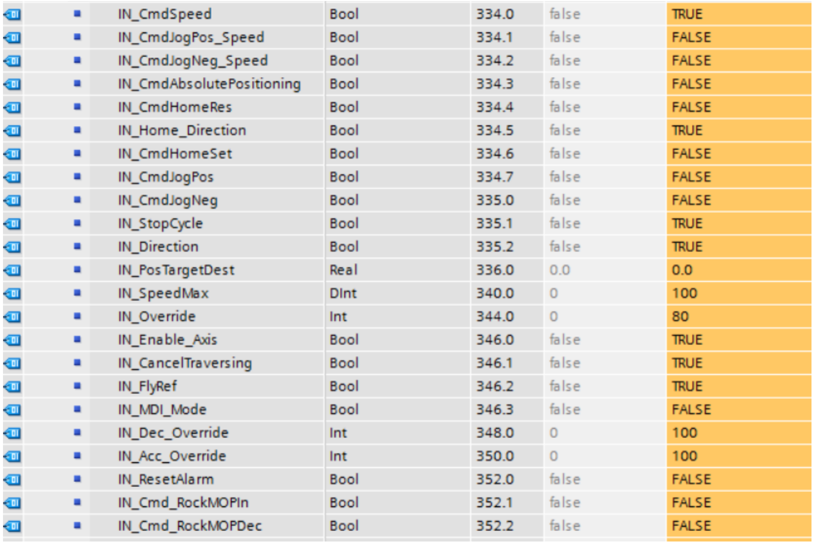
Value recorded by the control panel (yellow circle) is about calculated value, so drive is working properly.

The second phase of the test is the analysis of the speed control with a negative direction at a speed equal to 80% of the maximum speed.

The parameters to be set in the block "R101_Control_Interface" are:

- “IN_CmdSpeed” = 1;
- “IN_StopCycle” = 1;
- “IN_Direction”  = 1;
- “IN_MaxSpeed” = 100 [%];
- “IN_Override” = 80 [%];
- “IN_ReseAlarm”  = 0;
- “IN_Enable_Axis” = 1.

The following figure shows the set of values for performing this mode.



| | | | | |
|---|---|---|---|---|
| IN_CmdSpeed | Bool | 334.0 | false | TRUE |
| IN_CmdJogPos_Speed | Bool | 334.1 | false | FALSE |
| IN_CmdJogNeg_Speed | Bool | 334.2 | false | FALSE |
| IN_CmdAbsolutePositioning | Bool | 334.3 | false | FALSE |
| IN_CmdHomeRes | Bool | 334.4 | false | FALSE |
| IN_Home_Direction | Bool | 334.5 | false | TRUE |
| IN_CmdHomeSet | Bool | 334.6 | false | FALSE |
| IN_CmdJogPos | Bool | 334.7 | false | FALSE |
| IN_CmdJogNeg | Bool | 335.0 | false | FALSE |
| IN_StopCycle | Bool | 335.1 | false | TRUE |
| IN_Direction | Bool | 335.2 | false | TRUE |
| IN_PosTargetDest | Real | 336.0 | 0.0 | 0.0 |
| IN_SpeedMax | DInt | 340.0 | 0 | 100 |
| IN_Override | Int | 344.0 | 0 | 80 |
| IN_Enable_Axis | Bool | 346.0 | false | TRUE |
| IN_CancelTraversing | Bool | 346.1 | false | TRUE |
| IN_FlyRef | Bool | 346.2 | false | TRUE |
| IN_MDI_Mode | Bool | 346.3 | false | FALSE |
| IN_Dec_Override | Int | 348.0 | 0 | 100 |
| IN_Acc_Override | Int | 350.0 | 0 | 100 |
| IN_ResetAlarm | Bool | 352.0 | false | FALSE |
| IN_Cmd_RockMOPIn | Bool | 352.1 | false | FALSE |
| IN_Cmd_RockMOPDec | Bool | 352.2 | false | FALSE |

(Fig. 3.32 The values of user inputs)

In a manner similar to speed control with a positive direction at a speed equal to 80% of the maximum speed, the execution phases of the control are:

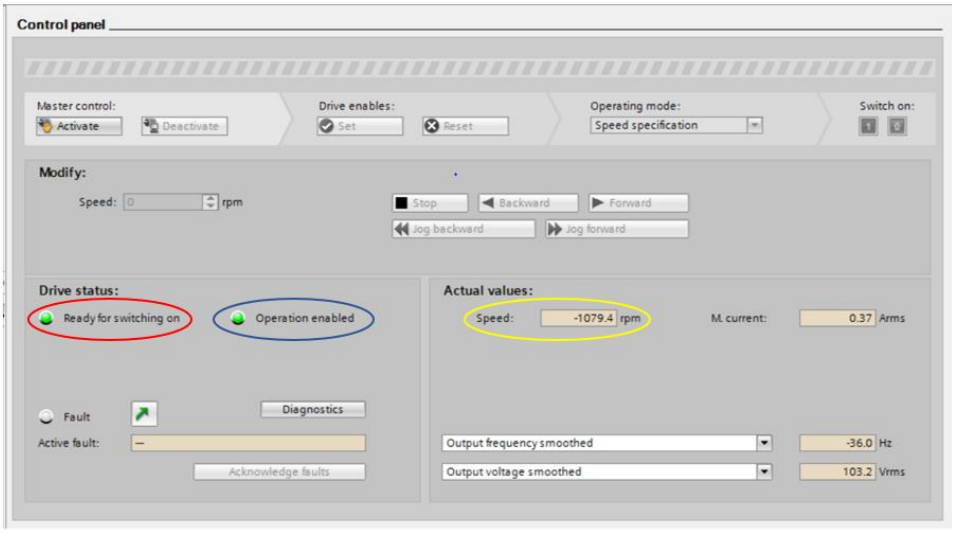
1. the R101 Control_Interface block reads the user's commands;

2. the R101 Control_Interface block generates the correct drive output values;

3. drive output variables values are copied into the drive structure;

4. the drive will run the speed control and the motor starts spinning with a negative direction and at a speed equal to -1080 rpm.

The correct operation of the drive is highlighted by the Siemens control panel (figure 3.33).



Operating mode: Speed specification

Speed: 0 rpm

Active fault: –

Speed: -1079.4 rpm

M. current: 0.37 Arms

Output frequency smoothed: -36.0 Hz

Output voltage smoothed: 103.2 Vrms

(Fig 3.33 Siemens control panel.)

Figure 3.33 shows that the drive is ready and it is running:

• ready for switching on LED is green (red circle);

• operation enabled LED is green (blue circle).

The speed of the jog mode is -1080 [rpm] (equation 3.6)

$$-1350 * \left(\frac{80}{100}\right) = -1080\ [rpm] \qquad (3.6)$$

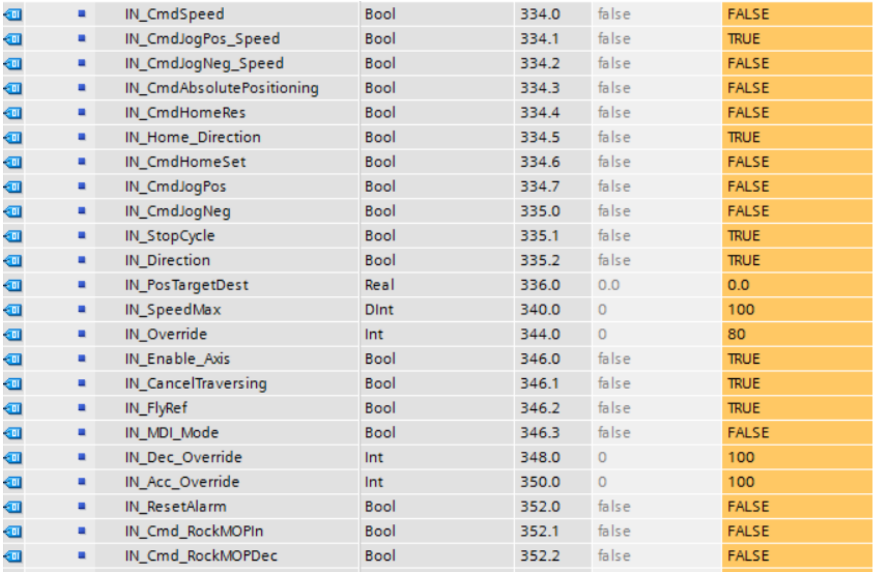
In other words, the value measured by the control panel (yellow circle) is equal to around the calculated value.

The third phase of the test is the study of the manual control with a positive direction. In a manner similar to “Jog mode” of Standard Telegram 1, speed of the jog mode is handled in the "R101_Control_Interface" block and it is independent of the input speed override. The jog mode override is set to 20% and therefore the speed is 20% of the maximum speed.

The parameters to be set in the block "Control_Interface" are:

- “IN_CmdJogPos_Speed” = 1;
- “IN_StopCycle” = 1;
- “IN_ReseAlarm”  = 0;
- “IN_Enable_Axis” = 1.

The following figure depicts the set of values for performing this mode.

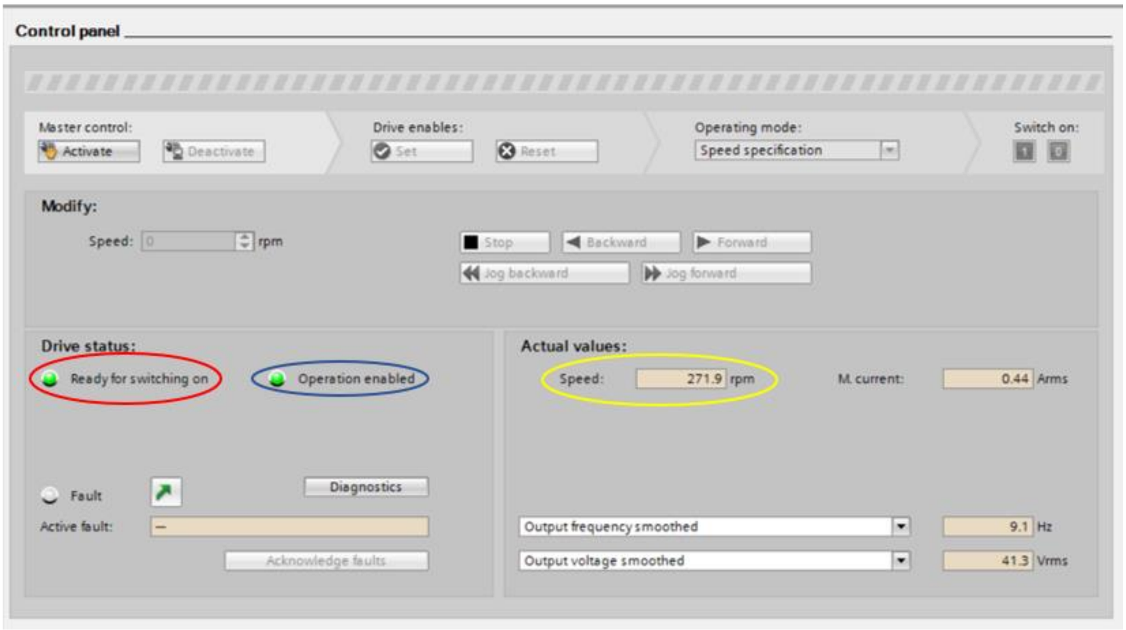


| | | | | |
|---|---|---|---|---|
| IN_CmdSpeed | Bool | 334.0 | false | FALSE |
| IN_CmdJogPos_Speed | Bool | 334.1 | false | TRUE |
| IN_CmdJogNeg_Speed | Bool | 334.2 | false | FALSE |
| IN_CmdAbsolutePositioning | Bool | 334.3 | false | FALSE |
| IN_CmdHomeRes | Bool | 334.4 | false | FALSE |
| IN_Home_Direction | Bool | 334.5 | false | TRUE |
| IN_CmdHomeSet | Bool | 334.6 | false | FALSE |
| IN_CmdJogPos | Bool | 334.7 | false | FALSE |
| IN_CmdJogNeg | Bool | 335.0 | false | FALSE |
| IN_StopCycle | Bool | 335.1 | false | TRUE |
| IN_Direction | Bool | 335.2 | false | TRUE |
| IN_PosTargetDest | Real | 336.0 | 0.0 | 0.0 |
| IN_SpeedMax | DInt | 340.0 | 0 | 100 |
| IN_Override | Int | 344.0 | 0 | 80 |
| IN_Enable_Axis | Bool | 346.0 | false | TRUE |
| IN_CancelTraversing | Bool | 346.1 | false | TRUE |
| IN_FlyRef | Bool | 346.2 | false | TRUE |
| IN_MDI_Mode | Bool | 346.3 | false | FALSE |
| IN_Dec_Override | Int | 348.0 | 0 | 100 |
| IN_Acc_Override | Int | 350.0 | 0 | 100 |
| IN_ResetAlarm | Bool | 352.0 | false | FALSE |
| IN_Cmd_RockMOPIn | Bool | 352.1 | false | FALSE |
| IN_Cmd_RockMOPDec | Bool | 352.2 | false | FALSE |

(Fig. 3.34 The values of user inputs)

After imposing the user inputs parameters, the block processes the commands and sends them to the drive as it happened for the second phase of the test.

At this point the motor starts spinning and the drive controls the engine speed (figure 3.35).



Control panel

Operating mode: Speed specification

Speed: 0 rpm

Speed: 271.9 rpm

M. current: 0.44 Arms

Active fault: –

Output frequency smoothed 9.1 Hz

Output voltage smoothed 41.3 Vrms

(Fig 3.35 Siemens control panel.)

Figure 3.35 shows that the drive is ready and is running:

- ready for switching on LED is green (red circle);
- operation enabled LED is green (blue circle).

Speed of the jog mode must be 270 [rpm] (equation 3.7)

$$1350 * \left(\frac{20}{100}\right) = 270\ [rpm] \qquad (3.7)$$

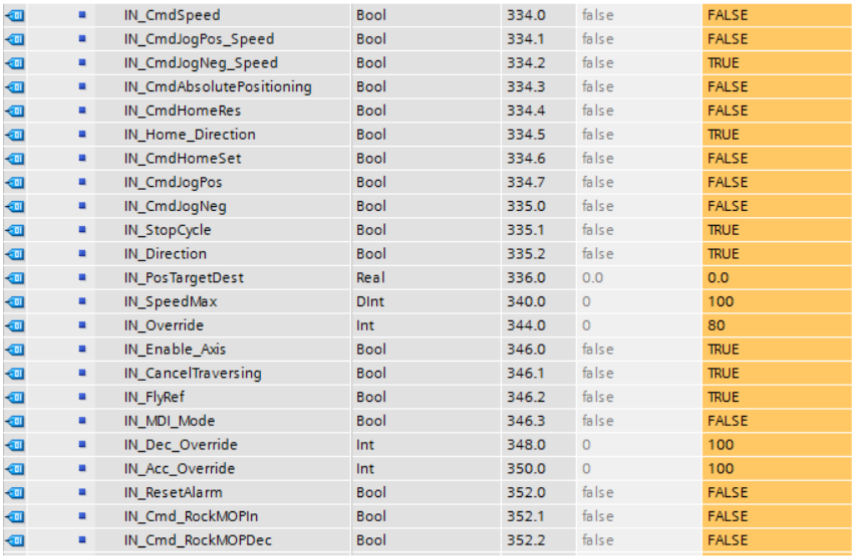
The control panel provides an actual value (yellow circle) equal to about the calculated value. In addition, it should be noted that actual values (yellow circle) depend on the internal value of the block and it is independent of the value "IN_Override" (figure 3.34). As a result, the drive works correctly.

The last phase of the test is the jog mode analysis with a negative direction. Again, the override of jog mode is set to 20% automatically by the "R101 Control_Interface block" and then the speed is 20% of the maximum speed.

The parameters to set in the block "R101 Control_Interface" are:

- “IN_CmdJogNeg_Speed” = 1;
- “IN_StopCycle” = 1;
- “IN_ReseAlarm” = 0;
- “IN_Enable_Axis” = 1.

This way the user sets the user inputs (figure 3.36).

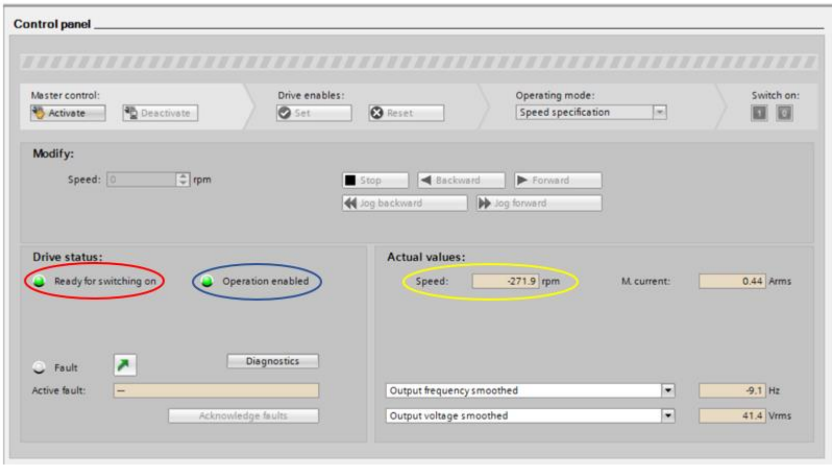


| | | | | |
|---|---|---|---|---|
| IN_CmdSpeed | Bool | 334.0 | false | FALSE |
| IN_CmdJogPos_Speed | Bool | 334.1 | false | FALSE |
| IN_CmdJogNeg_Speed | Bool | 334.2 | false | TRUE |
| IN_CmdAbsolutePositioning | Bool | 334.3 | false | FALSE |
| IN_CmdHomeRes | Bool | 334.4 | false | FALSE |
| IN_Home_Direction | Bool | 334.5 | false | TRUE |
| IN_CmdHomeSet | Bool | 334.6 | false | FALSE |
| IN_CmdJogPos | Bool | 334.7 | false | FALSE |
| IN_CmdJogNeg | Bool | 335.0 | false | FALSE |
| IN_StopCycle | Bool | 335.1 | false | TRUE |
| IN_Direction | Bool | 335.2 | false | TRUE |
| IN_PosTargetDest | Real | 336.0 | 0.0 | 0.0 |
| IN_SpeedMax | DInt | 340.0 | 0 | 100 |
| IN_Override | Int | 344.0 | 0 | 80 |
| IN_Enable_Axis | Bool | 346.0 | false | TRUE |
| IN_CancelTraversing | Bool | 346.1 | false | TRUE |
| IN_FlyRef | Bool | 346.2 | false | TRUE |
| IN_MDI_Mode | Bool | 346.3 | false | FALSE |
| IN_Dec_Override | Int | 348.0 | 0 | 100 |
| IN_Acc_Override | Int | 350.0 | 0 | 100 |
| IN_ResetAlarm | Bool | 352.0 | false | FALSE |
| IN_Cmd_RockMOPIn | Bool | 352.1 | false | FALSE |
| IN_Cmd_RockMOPDec | Bool | 352.2 | false | FALSE |

(Fig. 3.36 The values of user inputs)

As a result, "R101 Control_Interface" block produces the drive commands and the engine starts spinning at approximate -270 rpm (figure 3.37).

It should be noted that actual values (yellow circle) is independent of the value "IN_Override" (figure 3.36) because it depends only on the internal value of the block "R101_Control_Interface".



Control panel

Operating mode: Speed specification

Modify: Speed: 0 rpm

Drive status: Ready for switching on, Operation enabled

Active fault: –

Actual values:

Speed: -271.9 rpm

M. current: 0.44 Arms

Output frequency smoothed: -9.1 Hz

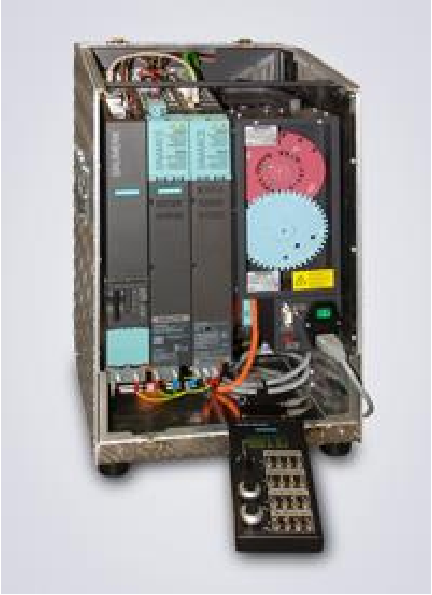
Output voltage smoothed: 41.4 Vrms

(Fig 3.37 Siemens control panel.)

In conclusion, the test showed that the communication between PLC and Siemens inverter drive via the Siemens Telegram 352 was performed correctly. Specifically, all the functions of the program have been performed correctly and completely.

## 3.4 Test about Standard Telegram 111

The test on the communication interface with the Siemens Servo drive was performed using the PLC Siemens 1511 F (figure 3.1), Servo drive S120 with a “Double motor module” (figure 3.38) and an electric motor.



(Fig. 3.38 Siemens servo drive S120)

The motor characteristics are described in the following table (table 3.5).

| Parameters | Value | Unit |
|---|---|---|
| Rated motor voltage | 186 | Veff |
| Rated motor current | 1.40 | Aeff |
| Number of revolutions rated motor | 6000.0 | 1/min |
| Number of pairs of motor poles | 3 | |
| Motor torque constant | 0.46 | Nm/A |
| Maximum motor speed | 10000.0 | 1/min |
| Maximum motor current | 7.50 | Aeff |
| Number of maximum laps | 10000.000 | 1/min |

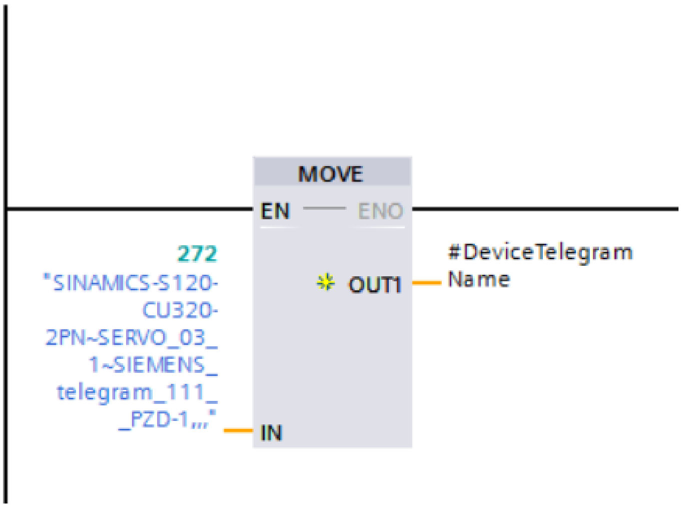
(Table 3.5 Motor data)

The servo drive S120 is able to control the position of an engine. The communication structure between CPU and servo drive S120 is the Standard Telegram 111 (appendix A).

The objective of the following test is the correct functioning of the Standard Telegram 111 for the correct communication between Siemens PLC and Siemens servo drives.

Initially the positive jog mode will be tested.

The first two activities are the addressing (figure 3.39) and the assigning variables to each communication bit of the interface. In this case, the interface used is represented in table 3.6 and table 3.7.


MOVE
EN
ENO
272
"SINAMICS-S120-CU320-2PN~SERVO_03_1~SIEMENS_telegram_111__PZD-1,,,"
IN
OUT1
#DeviceTelegramName

(Fig. 3.39 Addressing)

| Control_Interface | Siemens interface control word |
|---|---|
| Control_Start | #PZD.CONTROL_WORD.Control_Word_1.Off1 |
| Control_JobStart | #PZD.CONTROL_WORD.Control_Word_1.TrvStart |
| Control_noJOB_STOP | #PZD.CONTROL_WORD.Control_Word_1.RejTrvTsk |
| Control_NOSTOP | #PZD.CONTROL_WORD.Control_Word_1.IntMStop |
| Control_ControlFromPLC | #PZD.CONTROL_WORD.Control_Word_1.LB |
| Control_RUN | #PZD.CONTROL_WORD."EPosSTW 1".MdiStart |
| Control_GoalPosition | #PZD.CONTROL_WORD.Position |
| Control_SpeedPosition | #PZD.CONTROL_WORD.Velocity |
| Control_OverrideV | |
| Control_OverridePosV | #PZD.CONTROL_WORD.OverrideV |
| Control_Acceleration | #PZD.CONTROL_WORD.OverrideA |
| Control_Deceleration | #PZD.CONTROL_WORD.OverrideD |
| Control_GoalSpeed_LU | |
| Control_GoalSpeed_Hz | |
| Control_GoalSpeed_rpm | |

| | |
|---|---|
| Control_PositioningMode | #PZD.CONTROL_WORD."EPosSTW 1".MdiTyp |
| Control_ReferenceSearchStart | #PZD.CONTROL_WORD.Control_Word_1.RefStart |
| Control_PositiveDir | #PZD.CONTROL_WORD."EPosSTW 1".MdiPos |
| Control_NegativeDir | #PZD.CONTROL_WORD."EPosSTW 1".MdiNeg |
| Control_RefType | #PZD.CONTROL_WORD."EPosSTW 2".RefTyp |
| Control_ReferenceSearchDir | #PZD.CONTROL_WORD."EPosSTW 2".RefStDi |
| Control_Jog1 | #PZD.CONTROL_WORD.Control_Word_1.Jog1 |
| Control_Jog2 | #PZD.CONTROL_WORD.Control_Word_1.Jog2 |
| Control_AlarmReset | #PZD.CONTROL_WORD.Control_Word_1.AckFault |
| Control_SetReferencePoint | #PZD.CONTROL_WORD."EPosSTW 2".SetRefPt |
| Control_SpeedDirection | |
| Control_EnableSpeed | |
| Control_OFF2 | #PZD.CONTROL_WORD.Control_Word_1.Off2 |
| Control_OFF3 | #PZD.CONTROL_WORD.Control_Word_1.Off3 |
| Control_EnableOperation | |
| Control_EnableRamp | |
| Control_ContinueRamp | |
| Control_EnablePos | #PZD.CONTROL_WORD.Control_Word_1.Enc |
| Control_RockMOPIn | |
| Control_RockMOPDec | |
| Control_RockJOG | |
| Control_RockNegative | |
| Control_RockPositive | |
| Control_Stop | |

(Table 3.6 Interface between Drive output signals of R101_Control_Interface” block and Standard Telegram 111 control words)

| Status_Interface | Siemens interface status word |
|---|---|
| Status_Alarm | #PZD."STATUS WORD".Status_Word_1.Alarm |
| Status_Ready | #PZD."STATUS WORD".Status_Word_1.IOp |
| Status_Blocked | #PZD."STATUS WORD".Status_Word_1.Fault |
| Status_AxisEnabled | |
| Status_PositiveDir | #PZD."STATUS WORD"."EPosZSW 2".FWD |
| Status_NegativeDir | #PZD."STATUS WORD"."EPosZSW 2".BWD |
| Status_Direction | #PZD."STATUS WORD".Status_Word_1.RTS |

| | |
|---|---|
| Status_ReferenceDone | #PZD."STATUS WORD".Status_Word_1.RefPset |
| Status_SpeedReached | |
| Status_PositionReached | #PZD."STATUS WORD".Status_Word_1.TargPos |
| Status_AlarmCode | #PZD."STATUS WORD".ErrNr |
| Status_WarningCode | #PZD."STATUS WORD".WarnNr |
| Status_ActualSpeed | #PZD."STATUS WORD".Velocity |
| Status_ActualPosition | #PZD."STATUS WORD".Position |

(Table 3.7 Interface between Drive input signals of R101_Control_Interface” block and Standard Telegram 111 status words)

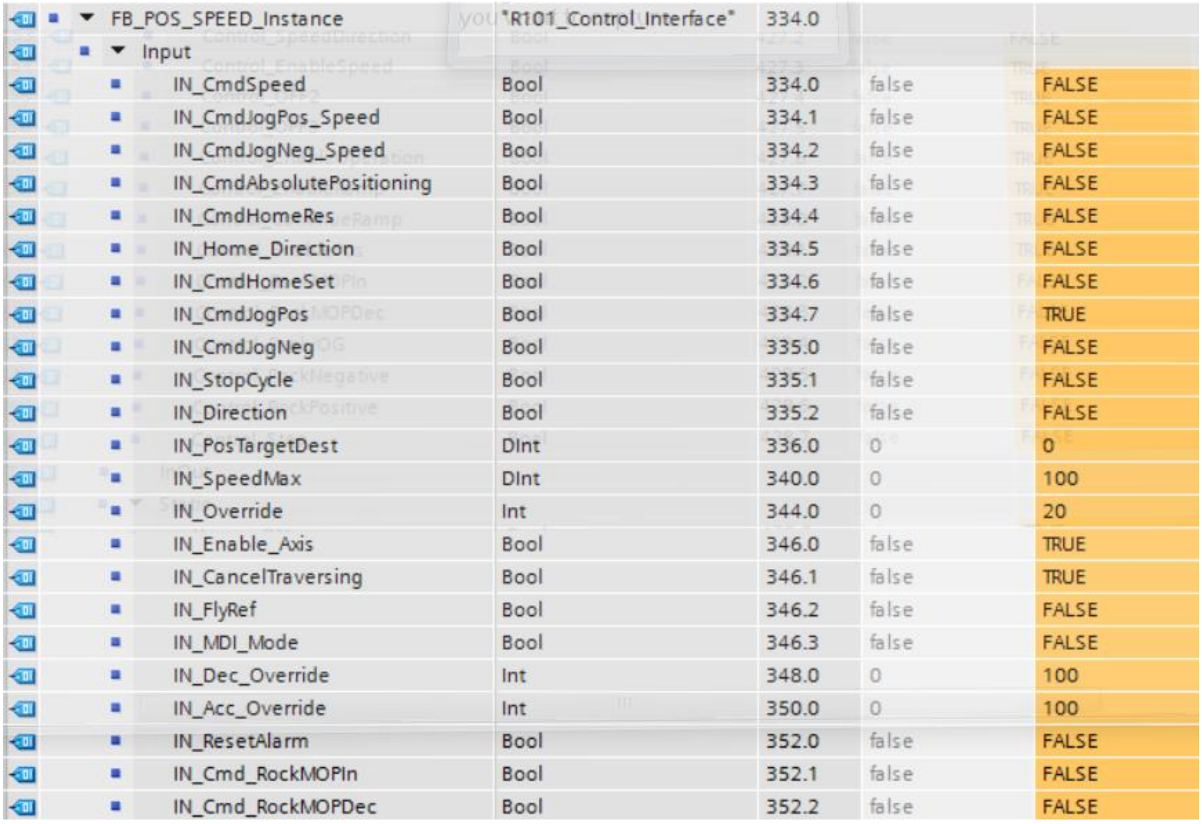
The second task is setting the type of control that must be executed and its mode.

In this case the parameters to be set in the block "R101_Control_Interface" are:

- “IN_CmdJogPos” = 1;
- “IN_StopCycle” = 0;
- “IN_SpeedMax” = 100 [%];
- “IN_Enable_Axis” = 1;
- “IN_CancelTraversing” =1.

It should be noted that the "IN_SpeedMax" value is different from the maximum engine speed, but it is equal to 100. This peculiarity is caused by how Siemens Servo drive calculates the wanted speed. It calculates the desired speed according to the "p2000" parameter of the expert list, like the Siemens inverter. In the specific case of the test, the speed will be set by servo drive automatically. In other words, the speed does not depend on “IN_Override” input.

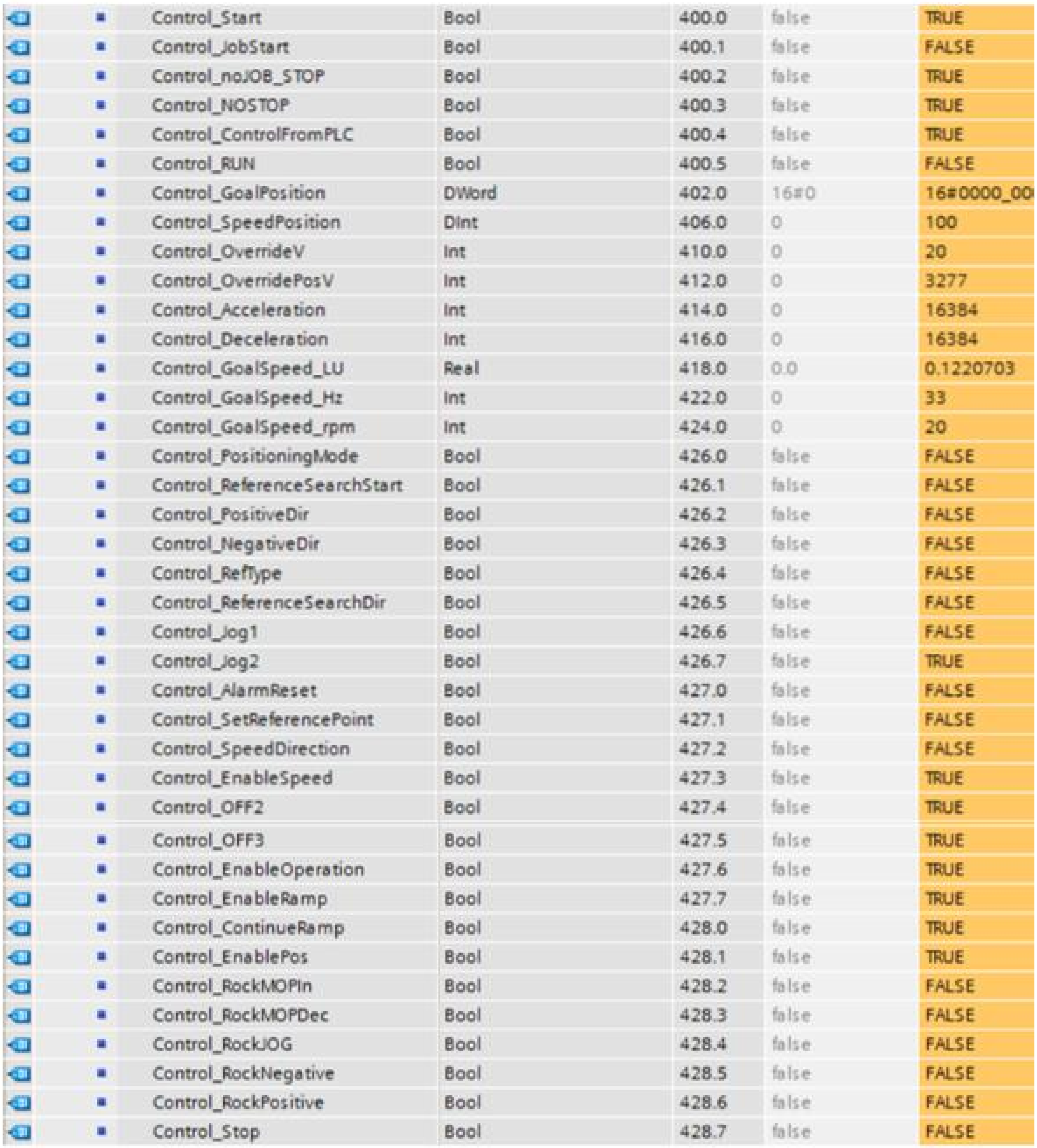
Figure 3.40 represents the commands that the user has applied.



| | | | | |
|---|---|---|---|---|
| FB_POS_SPEED_Instance | "R101_Control_Interface" | 334.0 | | |
| Input | | | | |
| IN_CmdSpeed | Bool | 334.0 | false | FALSE |
| IN_CmdJogPos_Speed | Bool | 334.1 | false | FALSE |
| IN_CmdJogNeg_Speed | Bool | 334.2 | false | FALSE |
| IN_CmdAbsolutePositioning | Bool | 334.3 | false | FALSE |
| IN_CmdHomeRes | Bool | 334.4 | false | FALSE |
| IN_Home_Direction | Bool | 334.5 | false | FALSE |
| IN_CmdHomeSet | Bool | 334.6 | false | FALSE |
| IN_CmdJogPos | Bool | 334.7 | false | TRUE |
| IN_CmdJogNeg | Bool | 335.0 | false | FALSE |
| IN_StopCycle | Bool | 335.1 | false | FALSE |
| IN_Direction | Bool | 335.2 | false | FALSE |
| IN_PosTargetDest | DInt | 336.0 | 0 | 0 |
| IN_SpeedMax | DInt | 340.0 | 0 | 100 |
| IN_Override | Int | 344.0 | 0 | 20 |
| IN_Enable_Axis | Bool | 346.0 | false | TRUE |
| IN_CancelTraversing | Bool | 346.1 | false | TRUE |
| IN_FlyRef | Bool | 346.2 | false | FALSE |
| IN_MDI_Mode | Bool | 346.3 | false | FALSE |
| IN_Dec_Override | Int | 348.0 | 0 | 100 |
| IN_Acc_Override | Int | 350.0 | 0 | 100 |
| IN_ResetAlarm | Bool | 352.0 | false | FALSE |
| IN_Cmd_RockMOPIn | Bool | 352.1 | false | FALSE |
| IN_Cmd_RockMOPDec | Bool | 352.2 | false | FALSE |

(Fig. 3.40 User inputs)

The "R101_Control_Interface" block calculates the commands that must be sent to the servo drive according to the user inputs (figure 3.41).

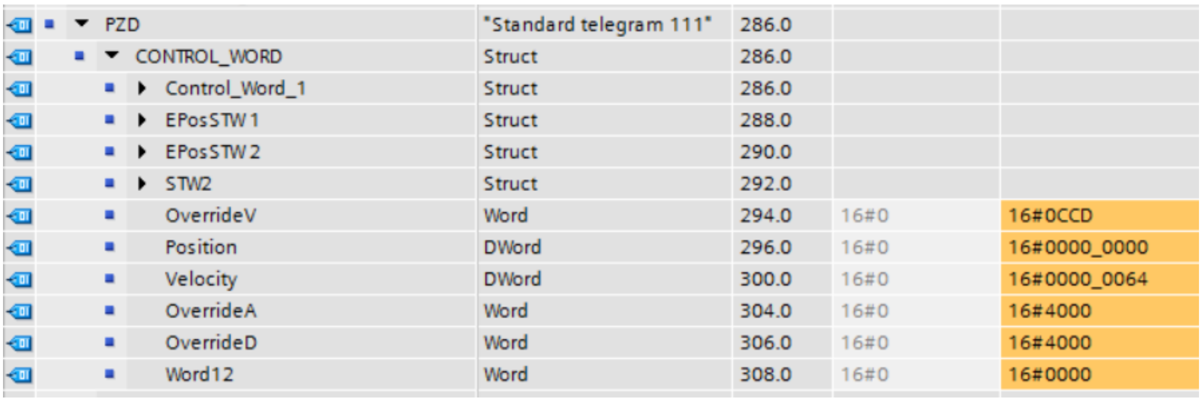


| | | | | |
|---|---|---|---|---|
| Control_Start | Bool | 400.0 | false | TRUE |
| Control_JobStart | Bool | 400.1 | false | FALSE |
| Control_noJOB_STOP | Bool | 400.2 | false | TRUE |
| Control_NOSTOP | Bool | 400.3 | false | TRUE |
| Control_ControlFromPLC | Bool | 400.4 | false | TRUE |
| Control_RUN | Bool | 400.5 | false | FALSE |
| Control_GoalPosition | DWord | 402.0 | 16#0 | 16#0000_00 |
| Control_SpeedPosition | DInt | 406.0 | 0 | 100 |
| Control_OverrideV | Int | 410.0 | 0 | 20 |
| Control_OverridePosV | Int | 412.0 | 0 | 3277 |
| Control_Acceleration | Int | 414.0 | 0 | 16384 |
| Control_Deceleration | Int | 416.0 | 0 | 16384 |
| Control_GoalSpeed_LU | Real | 418.0 | 0.0 | 0.1220703 |
| Control_GoalSpeed_Hz | Int | 422.0 | 0 | 33 |
| Control_GoalSpeed_rpm | Int | 424.0 | 0 | 20 |
| Control_PositioningMode | Bool | 426.0 | false | FALSE |
| Control_ReferenceSearchStart | Bool | 426.1 | false | FALSE |
| Control_PositiveDir | Bool | 426.2 | false | FALSE |
| Control_NegativeDir | Bool | 426.3 | false | FALSE |
| Control_RefType | Bool | 426.4 | false | FALSE |
| Control_ReferenceSearchDir | Bool | 426.5 | false | FALSE |
| Control_Jog1 | Bool | 426.6 | false | FALSE |
| Control_Jog2 | Bool | 426.7 | false | TRUE |
| Control_AlarmReset | Bool | 427.0 | false | FALSE |
| Control_SetReferencePoint | Bool | 427.1 | false | FALSE |
| Control_SpeedDirection | Bool | 427.2 | false | FALSE |
| Control_EnableSpeed | Bool | 427.3 | false | TRUE |
| Control_OFF2 | Bool | 427.4 | false | TRUE |
| Control_OFF3 | Bool | 427.5 | false | TRUE |
| Control_EnableOperation | Bool | 427.6 | false | TRUE |
| Control_EnableRamp | Bool | 427.7 | false | TRUE |
| Control_ContinueRamp | Bool | 428.0 | false | TRUE |
| Control_EnablePos | Bool | 428.1 | false | TRUE |
| Control_RockMOPIn | Bool | 428.2 | false | FALSE |
| Control_RockMOPDec | Bool | 428.3 | false | FALSE |
| Control_RockJOG | Bool | 428.4 | false | FALSE |
| Control_RockNegative | Bool | 428.5 | false | FALSE |
| Control_RockPositive | Bool | 428.6 | false | FALSE |
| Control_Stop | Bool | 428.7 | false | FALSE |

(Fig. 3.41 The values of drive outputs)

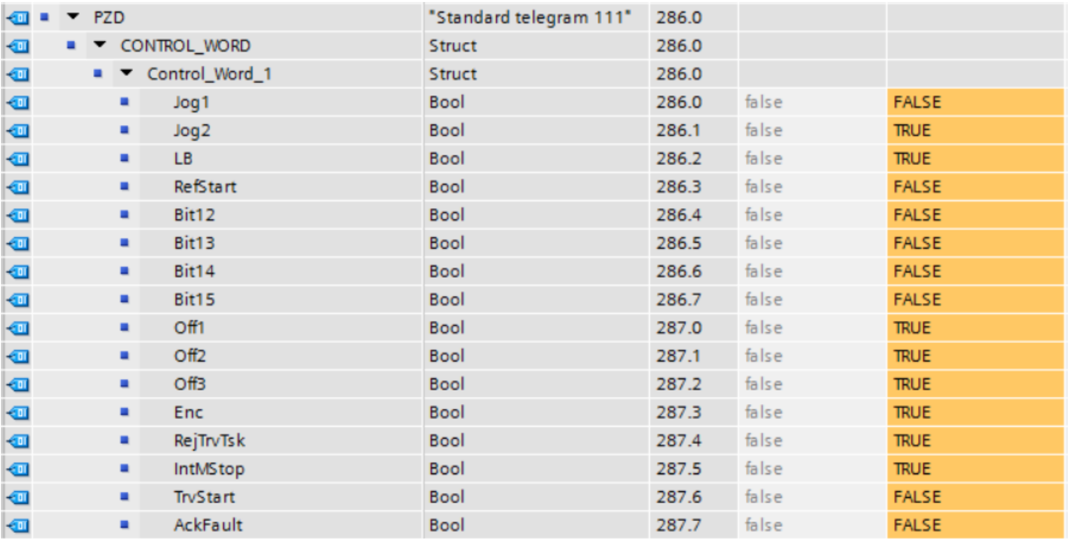
Drive output variables are associated with bits of internal structure (figure 3.42).

It should be noted that the structure of figure 3.42 is similar to the command words of the Standard Telegram 111 (appendix A). Indeed, the only difference is the bytes position that consists of the command words. This difference depends on how Siemens devices read words. Specifically, Siemens devices reverse the priority of the word bytes. In other words, firstly Siemens devices read the second byte of the word and then the first byte of the word. This difference is only present for Siemens devices.



| | | | | |
|---|---|---|---|---|
| PZD | "Standard telegram 111" | 286.0 | | |
| CONTROL_WORD | Struct | 286.0 | | |
| Control_Word_1 | Struct | 286.0 | | |
| EPosSTW1 | Struct | 288.0 | | |
| EPosSTW2 | Struct | 290.0 | | |
| STW2 | Struct | 292.0 | | |
| OverrideV | Word | 294.0 | 16#0 | 16#0CCD |
| Position | DWord | 296.0 | 16#0 | 16#0000_0000 |
| Velocity | DWord | 300.0 | 16#0 | 16#0000_0064 |
| OverrideA | Word | 304.0 | 16#0 | 16#4000 |
| OverrideD | Word | 306.0 | 16#0 | 16#4000 |
| Word12 | Word | 308.0 | 16#0 | 16#0000 |

(Fig. 3.42 A The values of command words of drive)



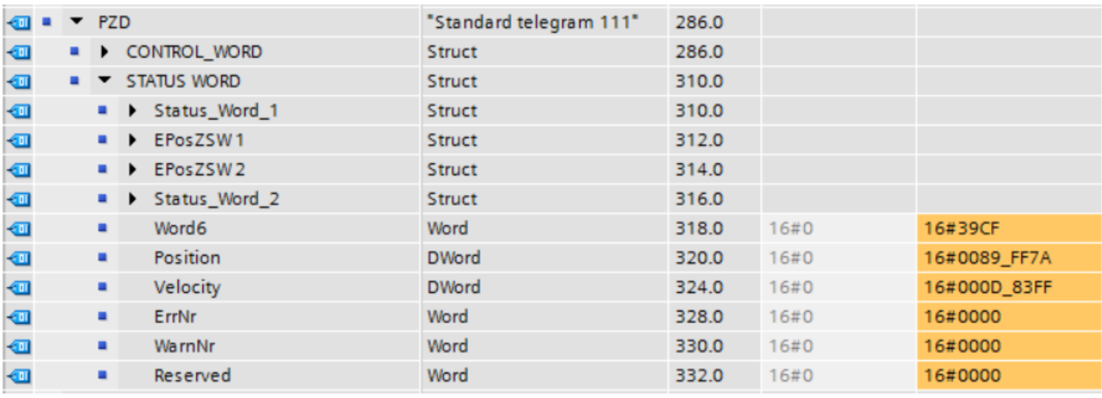
| | | | | |
|---|---|---|---|---|
| PZD | "Standard telegram 111" | 286.0 | | |
| CONTROL_WORD | Struct | 286.0 | | |
| Control_Word_1 | Struct | 286.0 | | |
| Jog1 | Bool | 286.0 | false | FALSE |
| Jog2 | Bool | 286.1 | false | TRUE |
| LB | Bool | 286.2 | false | TRUE |
| RefStart | Bool | 286.3 | false | FALSE |
| Bit12 | Bool | 286.4 | false | FALSE |
| Bit13 | Bool | 286.5 | false | FALSE |
| Bit14 | Bool | 286.6 | false | FALSE |
| Bit15 | Bool | 286.7 | false | FALSE |
| Off1 | Bool | 287.0 | false | TRUE |
| Off2 | Bool | 287.1 | false | TRUE |
| Off3 | Bool | 287.2 | false | TRUE |
| Enc | Bool | 287.3 | false | TRUE |
| RejTrvTsk | Bool | 287.4 | false | TRUE |
| IntMStop | Bool | 287.5 | false | TRUE |
| TrvStart | Bool | 287.6 | false | FALSE |
| AckFault | Bool | 287.7 | false | FALSE |

(Fig. 3.42 B The values of command words of drive)

The internal structure is copied into the drive structure to activate a series of bits and consequently the start of the drive (figure 3.42).

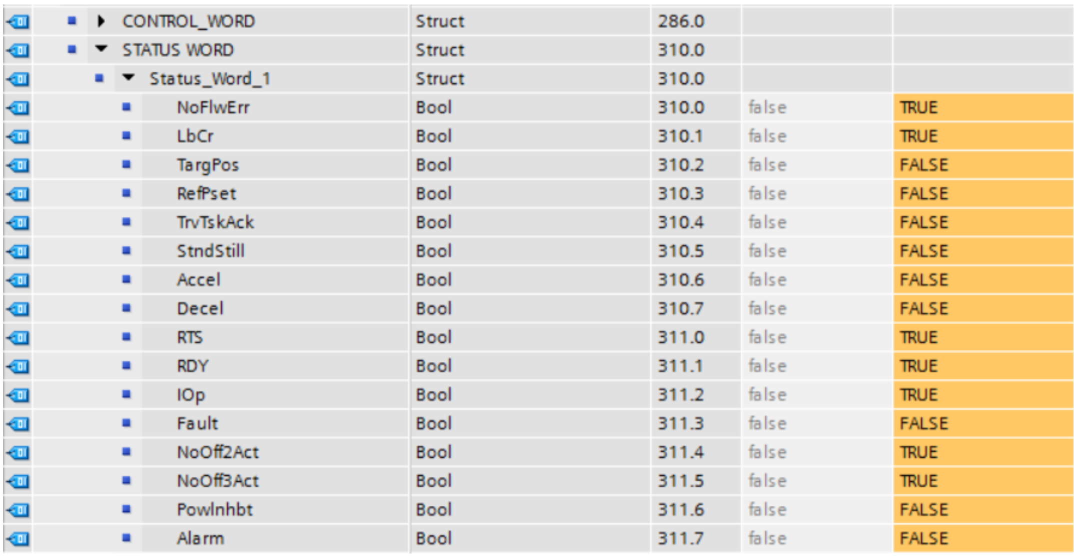
As a result, the motor starts spinning.

The reading values of the drive status words shows the operation of the drive (figure 3.43).



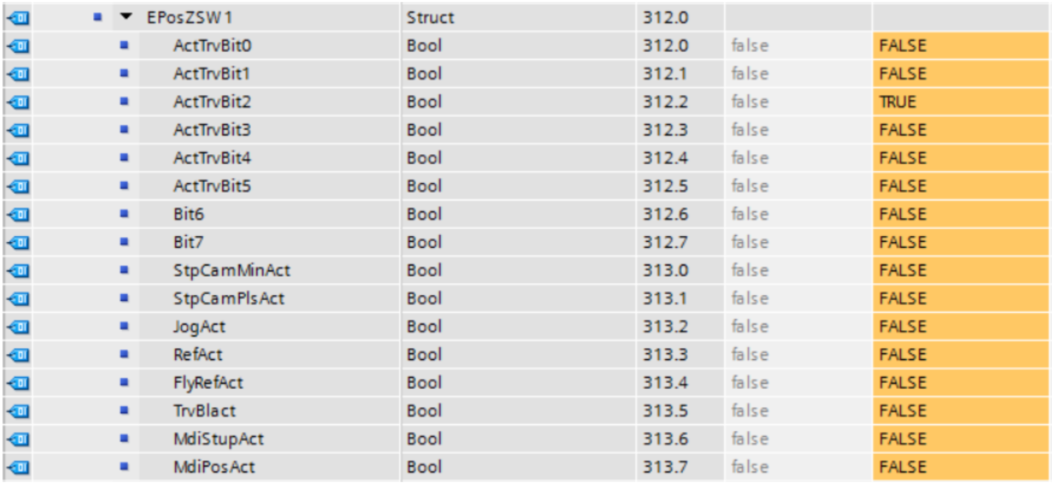
| | | | | |
|---|---|---|---|---|
| PZD | "Standard telegram 111" | 286.0 | | |
| CONTROL_WORD | Struct | 286.0 | | |
| STATUS WORD | Struct | 310.0 | | |
| Status_Word_1 | Struct | 310.0 | | |
| EPosZSW1 | Struct | 312.0 | | |
| EPosZSW2 | Struct | 314.0 | | |
| Status_Word_2 | Struct | 316.0 | | |
| Word6 | Word | 318.0 | 16#0 | 16#39CF |
| Position | DWord | 320.0 | 16#0 | 16#0089_FF7A |
| Velocity | DWord | 324.0 | 16#0 | 16#000D_83FF |
| ErrNr | Word | 328.0 | 16#0 | 16#0000 |
| WarnNr | Word | 330.0 | 16#0 | 16#0000 |
| Reserved | Word | 332.0 | 16#0 | 16#0000 |

(Fig. 3.43 A The values of status words of drive)



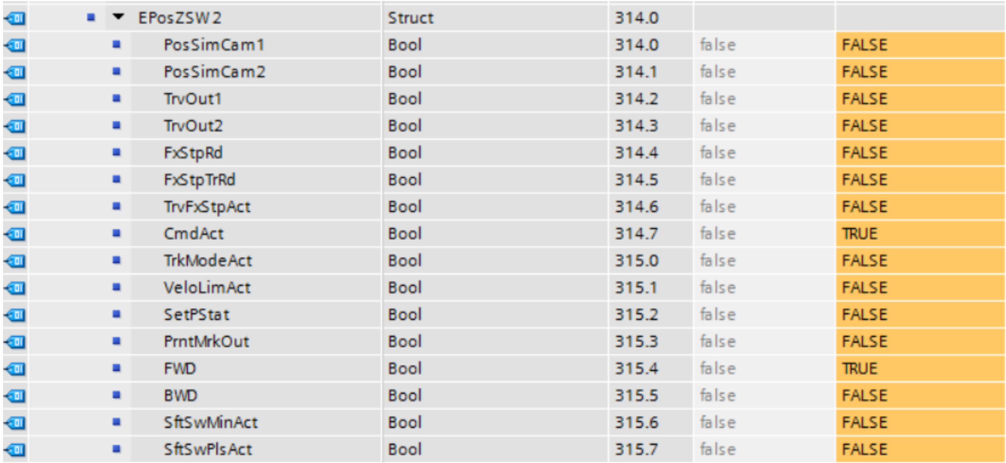
| | | | | |
|---|---|---|---|---|
| CONTROL_WORD | Struct | 286.0 | | |
| STATUS WORD | Struct | 310.0 | | |
| Status_Word_1 | Struct | 310.0 | | |
| NoFlwErr | Bool | 310.0 | false | TRUE |
| LbCr | Bool | 310.1 | false | TRUE |
| TargPos | Bool | 310.2 | false | FALSE |
| RefPset | Bool | 310.3 | false | FALSE |
| TrvTskAck | Bool | 310.4 | false | FALSE |
| StndStill | Bool | 310.5 | false | FALSE |
| Accel | Bool | 310.6 | false | FALSE |
| Decel | Bool | 310.7 | false | FALSE |
| RTS | Bool | 311.0 | false | TRUE |
| RDY | Bool | 311.1 | false | TRUE |
| IOp | Bool | 311.2 | false | TRUE |
| Fault | Bool | 311.3 | false | FALSE |
| NoOff2Act | Bool | 311.4 | false | TRUE |
| NoOff3Act | Bool | 311.5 | false | TRUE |
| PowInhbt | Bool | 311.6 | false | FALSE |
| Alarm | Bool | 311.7 | false | FALSE |

(Fig. 3.43 B The values of status words of drive)



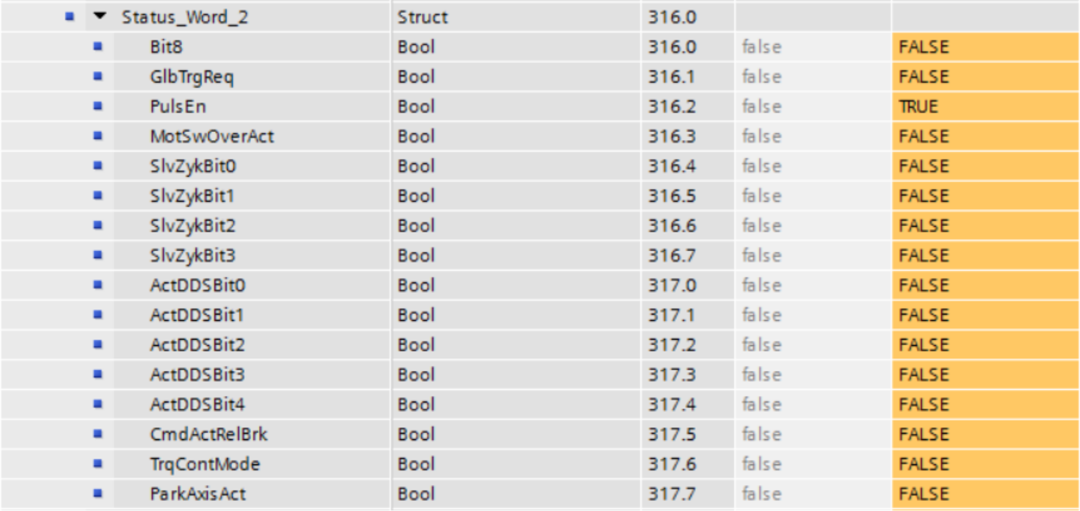
| | | | | |
|---|---|---|---|---|
| EPosZSW1 | Struct | 312.0 | | |
| ActTrvBit0 | Bool | 312.0 | false | FALSE |
| ActTrvBit1 | Bool | 312.1 | false | FALSE |
| ActTrvBit2 | Bool | 312.2 | false | TRUE |
| ActTrvBit3 | Bool | 312.3 | false | FALSE |
| ActTrvBit4 | Bool | 312.4 | false | FALSE |
| ActTrvBit5 | Bool | 312.5 | false | FALSE |
| Bit6 | Bool | 312.6 | false | FALSE |
| Bit7 | Bool | 312.7 | false | FALSE |
| StpCamMinAct | Bool | 313.0 | false | FALSE |
| StpCamPlsAct | Bool | 313.1 | false | FALSE |
| JogAct | Bool | 313.2 | false | FALSE |
| RefAct | Bool | 313.3 | false | FALSE |
| FlyRefAct | Bool | 313.4 | false | FALSE |
| TrvBlact | Bool | 313.5 | false | FALSE |
| MdiStupAct | Bool | 313.6 | false | FALSE |
| MdiPosAct | Bool | 313.7 | false | FALSE |

(Fig. 3.43 C The values of status words of drive)



| | | | | |
|---|---|---|---|---|
| EPosZSW2 | Struct | 314.0 | | |
| PosSimCam1 | Bool | 314.0 | false | FALSE |
| PosSimCam2 | Bool | 314.1 | false | FALSE |
| TrvOut1 | Bool | 314.2 | false | FALSE |
| TrvOut2 | Bool | 314.3 | false | FALSE |
| FxStpRd | Bool | 314.4 | false | FALSE |
| FxStpTrRd | Bool | 314.5 | false | FALSE |
| TrvFxStpAct | Bool | 314.6 | false | FALSE |
| CmdAct | Bool | 314.7 | false | TRUE |
| TrkModeAct | Bool | 315.0 | false | FALSE |
| VeloLimAct | Bool | 315.1 | false | FALSE |
| SetPStat | Bool | 315.2 | false | FALSE |
| PrntMrkOut | Bool | 315.3 | false | FALSE |
| FWD | Bool | 315.4 | false | TRUE |
| BWD | Bool | 315.5 | false | FALSE |
| SftSwMinAct | Bool | 315.6 | false | FALSE |
| SftSwPlsAct | Bool | 315.7 | false | FALSE |

(Fig. 3.43 D The values of status words of drive)

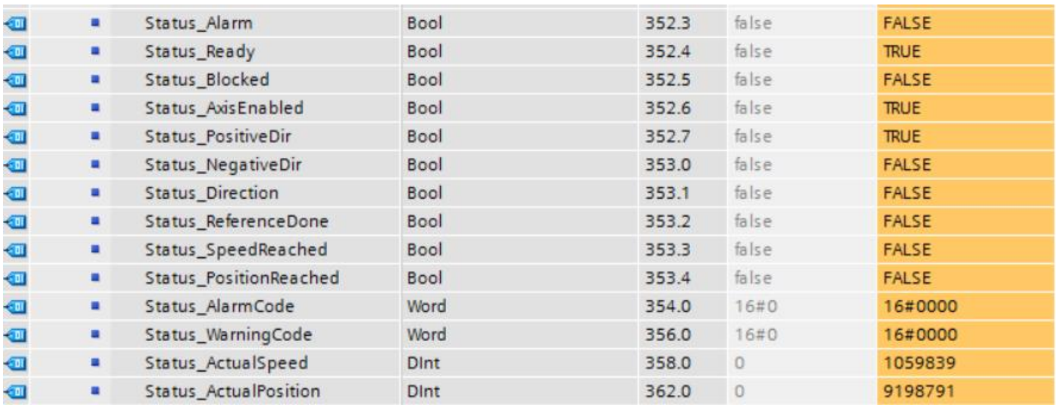


| | | | | |
|---|---|---|---|---|
| Status_Word_2 | Struct | 316.0 | | |
| Bit8 | Bool | 316.0 | false | FALSE |
| GlbTrgReq | Bool | 316.1 | false | FALSE |
| PulsEn | Bool | 316.2 | false | TRUE |
| MotSwOverAct | Bool | 316.3 | false | FALSE |
| SlvZykBit0 | Bool | 316.4 | false | FALSE |
| SlvZykBit1 | Bool | 316.5 | false | FALSE |
| SlvZykBit2 | Bool | 316.6 | false | FALSE |
| SlvZykBit3 | Bool | 316.7 | false | FALSE |
| ActDDSBit0 | Bool | 317.0 | false | FALSE |
| ActDDSBit1 | Bool | 317.1 | false | FALSE |
| ActDDSBit2 | Bool | 317.2 | false | FALSE |
| ActDDSBit3 | Bool | 317.3 | false | FALSE |
| ActDDSBit4 | Bool | 317.4 | false | FALSE |
| CmdActRelBrk | Bool | 317.5 | false | FALSE |
| TrqContMode | Bool | 317.6 | false | FALSE |
| ParkAxisAct | Bool | 317.7 | false | FALSE |

(Fig. 3.43 E The values of status words of drive)

It should be noted that the structure of figure 3.43 is similar to the status words of the Standard Telegram 111 (appendix A). The "jog2" bit in the word "Status_Word_1" is equal to "TRUE"; sure enough, this bit performs the positive jog mode.

As command words structure, the only difference is the position of command words bytes. These parameters provide the drive states and they must be sent to the interface block to transmit them to the end user and for the interface block normal function (figure 3.45). As a result, drive status words are copied into the internal structure (figure 3.43) and the most important bits are connected to the drive input signals (figure 3.44).

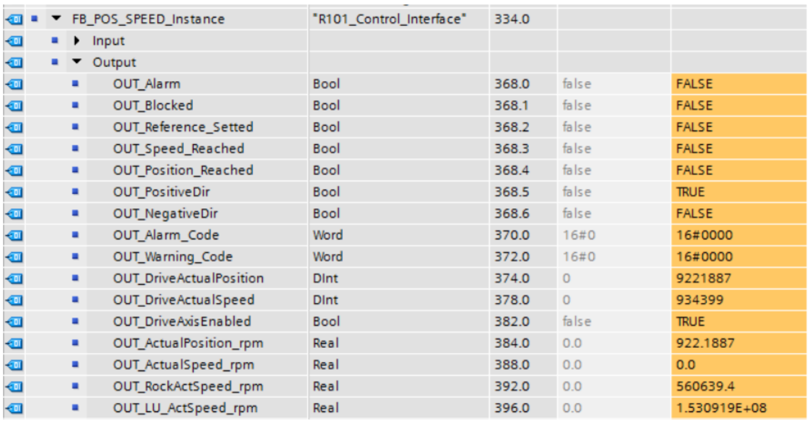


| | | | | |
|---|---|---|---|---|
| Status_Alarm | Bool | 352.3 | false | FALSE |
| Status_Ready | Bool | 352.4 | false | TRUE |
| Status_Blocked | Bool | 352.5 | false | FALSE |
| Status_AxisEnabled | Bool | 352.6 | false | TRUE |
| Status_PositiveDir | Bool | 352.7 | false | TRUE |
| Status_NegativeDir | Bool | 353.0 | false | FALSE |
| Status_Direction | Bool | 353.1 | false | FALSE |
| Status_ReferenceDone | Bool | 353.2 | false | FALSE |
| Status_SpeedReached | Bool | 353.3 | false | FALSE |
| Status_PositionReached | Bool | 353.4 | false | FALSE |
| Status_AlarmCode | Word | 354.0 | 16#0 | 16#0000 |
| Status_WarningCode | Word | 356.0 | 16#0 | 16#0000 |
| Status_ActualSpeed | DInt | 358.0 | 0 | 1059839 |
| Status_ActualPosition | DInt | 362.0 | 0 | 9198791 |

(Fig. 3.44 The values of drive input signals.)

From the image 3.44 a correct operation of the drive is evident:

- drive is ready state (Status_Ready = TRUE);
- the direction is positive (Status_PositiveDir = TRUE);
- there are not alarms (Status_Alarm = FALSE and Status_Warning = FALSE).



| | | | | |
|---|---|---|---|---|
| FB_POS_SPEED_Instance | "R101_Control_Interface" | 334.0 | | |
| Input | | | | |
| Output | | | | |
| OUT_Alarm | Bool | 368.0 | false | FALSE |
| OUT_Blocked | Bool | 368.1 | false | FALSE |
| OUT_Reference_Setted | Bool | 368.2 | false | FALSE |
| OUT_Speed_Reached | Bool | 368.3 | false | FALSE |
| OUT_Position_Reached | Bool | 368.4 | false | FALSE |
| OUT_PositiveDir | Bool | 368.5 | false | TRUE |
| OUT_NegativeDir | Bool | 368.6 | false | FALSE |
| OUT_Alarm_Code | Word | 370.0 | 16#0 | 16#0000 |
| OUT_Warning_Code | Word | 372.0 | 16#0 | 16#0000 |
| OUT_DriveActualPosition | DInt | 374.0 | 0 | 9221887 |
| OUT_DriveActualSpeed | DInt | 378.0 | 0 | 934399 |
| OUT_DriveAxisEnabled | Bool | 382.0 | false | TRUE |
| OUT_ActualPosition_rpm | Real | 384.0 | 0.0 | 922.1887 |
| OUT_ActualSpeed_rpm | Real | 388.0 | 0.0 | 0.0 |
| OUT_RockActSpeed_rpm | Real | 392.0 | 0.0 | 560639.4 |
| OUT_LU_ActSpeed_rpm | Real | 396.0 | 0.0 | 1.530919E+08 |

(Fig. 3.45 The values of user outputs)

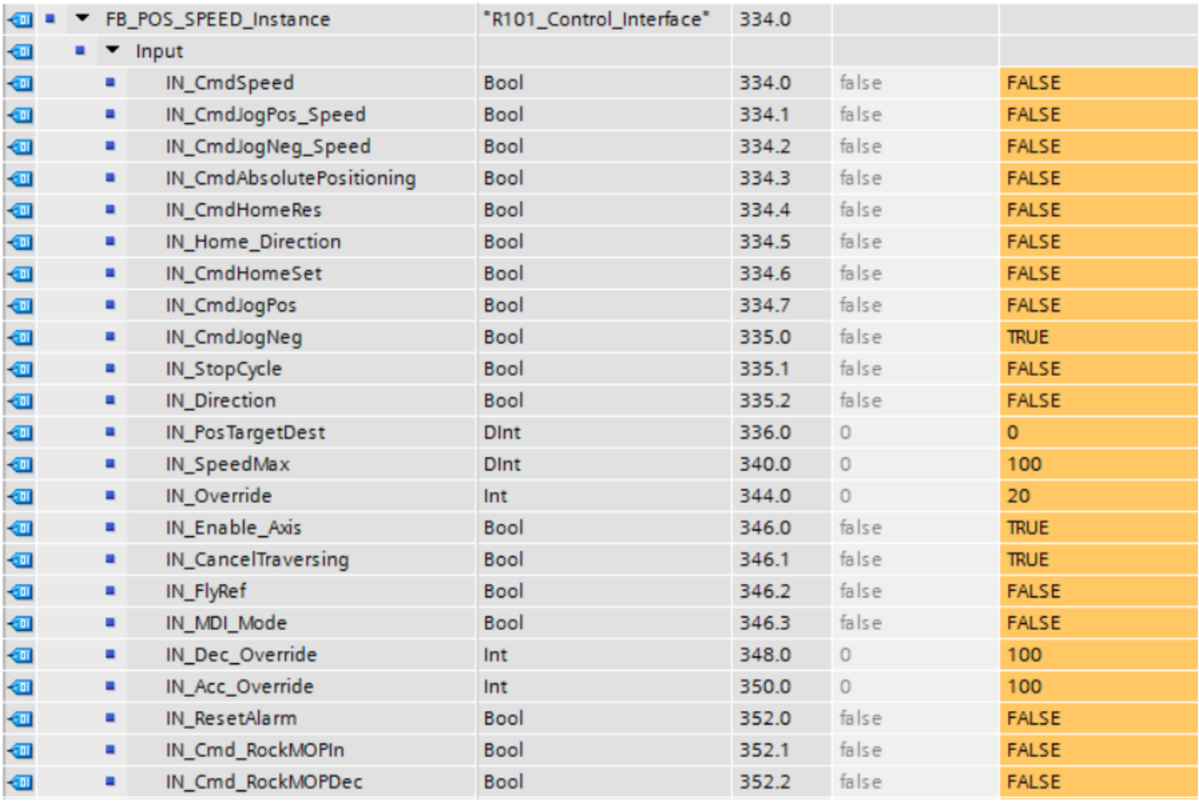
The second phase of the test is the analysis of the negative jog mode. In this case, the desired motor behavior is a manual movement with a negative movement direction.

In a manner similar to positive jog mode, the programmer must set the desired control type.

The parameters to be set in the block "R101_Control_Interface" are:

- “IN_CmdJogNeg” = 1;
- “IN_StopCycle” = 0;
- “IN_SpeedMax” = 100 [%];
- “IN_Enable_Axis” = 1;
- “IN_CancelTraversing” =1.

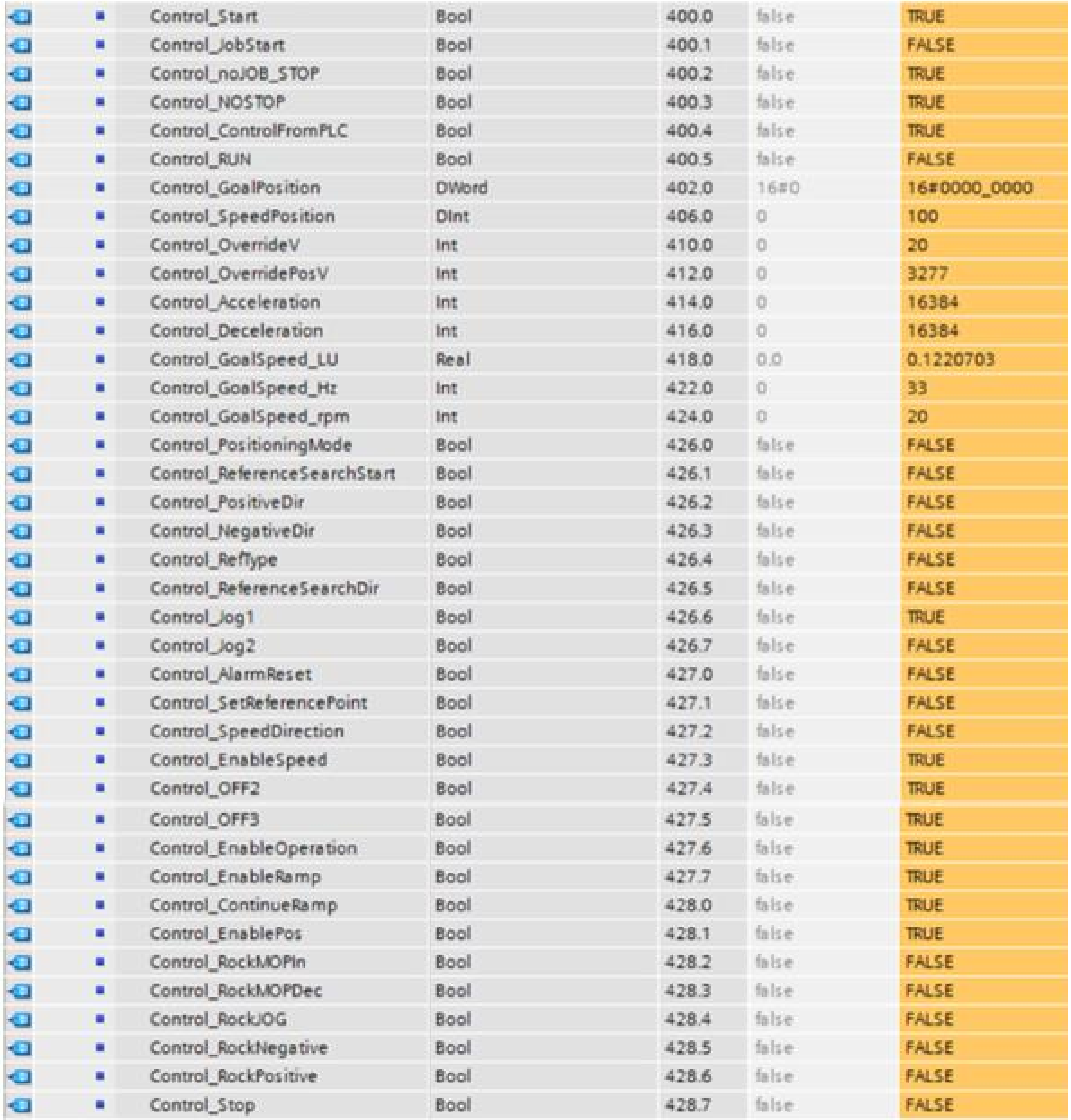
The operating mode must be set by using user inputs. Figure 3.46 represents the correct user inputs values for running the negative jog mode.



| | | | | |
|---|---|---|---|---|
| FB_POS_SPEED_Instance | "R101_Control_Interface" | 334.0 | | |
| Input | | | | |
| IN_CmdSpeed | Bool | 334.0 | false | FALSE |
| IN_CmdJogPos_Speed | Bool | 334.1 | false | FALSE |
| IN_CmdJogNeg_Speed | Bool | 334.2 | false | FALSE |
| IN_CmdAbsolutePositioning | Bool | 334.3 | false | FALSE |
| IN_CmdHomeRes | Bool | 334.4 | false | FALSE |
| IN_Home_Direction | Bool | 334.5 | false | FALSE |
| IN_CmdHomeSet | Bool | 334.6 | false | FALSE |
| IN_CmdJogPos | Bool | 334.7 | false | FALSE |
| IN_CmdJogNeg | Bool | 335.0 | false | TRUE |
| IN_StopCycle | Bool | 335.1 | false | FALSE |
| IN_Direction | Bool | 335.2 | false | FALSE |
| IN_PosTargetDest | DInt | 336.0 | 0 | 0 |
| IN_SpeedMax | DInt | 340.0 | 0 | 100 |
| IN_Override | Int | 344.0 | 0 | 20 |
| IN_Enable_Axis | Bool | 346.0 | false | TRUE |
| IN_CancelTraversing | Bool | 346.1 | false | TRUE |
| IN_FlyRef | Bool | 346.2 | false | FALSE |
| IN_MDI_Mode | Bool | 346.3 | false | FALSE |
| IN_Dec_Override | Int | 348.0 | 0 | 100 |
| IN_Acc_Override | Int | 350.0 | 0 | 100 |
| IN_ResetAlarm | Bool | 352.0 | false | FALSE |
| IN_Cmd_RockMOPIn | Bool | 352.1 | false | FALSE |
| IN_Cmd_RockMOPDec | Bool | 352.2 | false | FALSE |

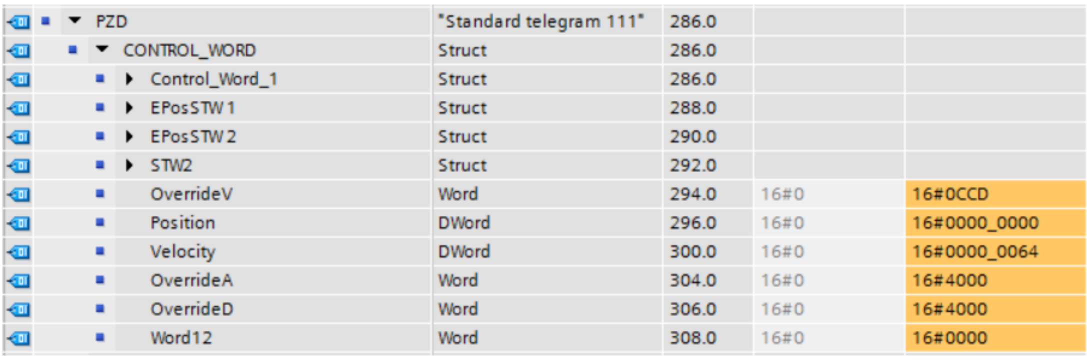
(Fig. 3.46 The values of user inputs)

The R101 Control_Interface block reads the user's commands and generates the correct drive output values (figure 3.47). Each Drive output variable is connected to internal structure (figure 3.48).



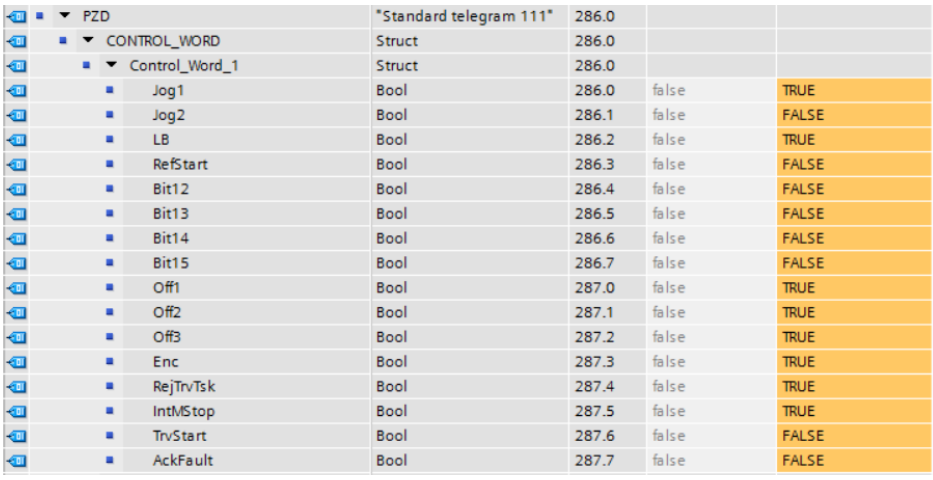
| | | | | |
|---|---|---|---|---|
| Control_Start | Bool | 400.0 | false | TRUE |
| Control_JobStart | Bool | 400.1 | false | FALSE |
| Control_noJOB_STOP | Bool | 400.2 | false | TRUE |
| Control_NOSTOP | Bool | 400.3 | false | TRUE |
| Control_ControlFromPLC | Bool | 400.4 | false | TRUE |
| Control_RUN | Bool | 400.5 | false | FALSE |
| Control_GoalPosition | DWord | 402.0 | 16#0 | 16#0000_0000 |
| Control_SpeedPosition | DInt | 406.0 | 0 | 100 |
| Control_OverrideV | Int | 410.0 | 0 | 20 |
| Control_OverridePosV | Int | 412.0 | 0 | 3277 |
| Control_Acceleration | Int | 414.0 | 0 | 16384 |
| Control_Deceleration | Int | 416.0 | 0 | 16384 |
| Control_GoalSpeed_LU | Real | 418.0 | 0.0 | 0.1220703 |
| Control_GoalSpeed_Hz | Int | 422.0 | 0 | 33 |
| Control_GoalSpeed_rpm | Int | 424.0 | 0 | 20 |
| Control_PositioningMode | Bool | 426.0 | false | FALSE |
| Control_ReferenceSearchStart | Bool | 426.1 | false | FALSE |
| Control_PositiveDir | Bool | 426.2 | false | FALSE |
| Control_NegativeDir | Bool | 426.3 | false | FALSE |
| Control_RefType | Bool | 426.4 | false | FALSE |
| Control_ReferenceSearchDir | Bool | 426.5 | false | FALSE |
| Control_Jog1 | Bool | 426.6 | false | TRUE |
| Control_Jog2 | Bool | 426.7 | false | FALSE |
| Control_AlarmReset | Bool | 427.0 | false | FALSE |
| Control_SetReferencePoint | Bool | 427.1 | false | FALSE |
| Control_SpeedDirection | Bool | 427.2 | false | FALSE |
| Control_EnableSpeed | Bool | 427.3 | false | TRUE |
| Control_OFF2 | Bool | 427.4 | false | TRUE |
| Control_OFF3 | Bool | 427.5 | false | TRUE |
| Control_EnableOperation | Bool | 427.6 | false | TRUE |
| Control_EnableRamp | Bool | 427.7 | false | TRUE |
| Control_ContinueRamp | Bool | 428.0 | false | TRUE |
| Control_EnablePos | Bool | 428.1 | false | TRUE |
| Control_RockMOPIn | Bool | 428.2 | false | FALSE |
| Control_RockMOPDec | Bool | 428.3 | false | FALSE |
| Control_RockJOG | Bool | 428.4 | false | FALSE |
| Control_RockNegative | Bool | 428.5 | false | FALSE |
| Control_RockPositive | Bool | 428.6 | false | FALSE |
| Control_Stop | Bool | 428.7 | false | FALSE |

(Fig. 3.47 The values of drive outputs)



| | | | | |
|---|---|---|---|---|
| ▼ PZD | "Standard telegram 111" | 286.0 | | |
| ▼ CONTROL_WORD | Struct | 286.0 | | |
| ▶ Control_Word_1 | Struct | 286.0 | | |
| ▶ EPosSTW1 | Struct | 288.0 | | |
| ▶ EPosSTW2 | Struct | 290.0 | | |
| ▶ STW2 | Struct | 292.0 | | |
| OverrideV | Word | 294.0 | 16#0 | 16#0CCD |
| Position | DWord | 296.0 | 16#0 | 16#0000_0000 |
| Velocity | DWord | 300.0 | 16#0 | 16#0000_0064 |
| OverrideA | Word | 304.0 | 16#0 | 16#4000 |
| OverrideD | Word | 306.0 | 16#0 | 16#4000 |
| Word12 | Word | 308.0 | 16#0 | 16#0000 |

(Fig. 3.48 A The values of command words of drive)



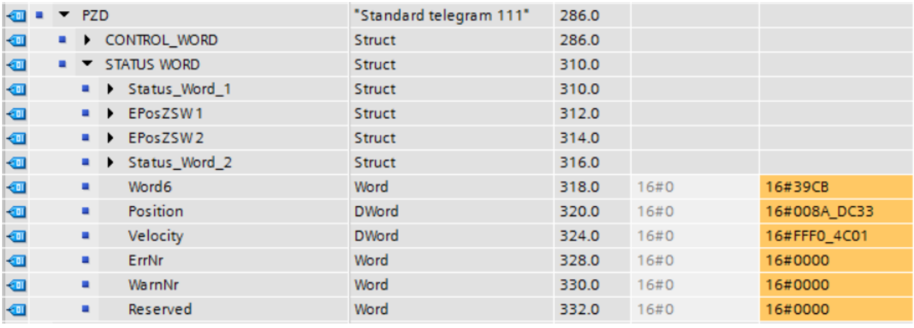
| | | | | |
|---|---|---|---|---|
| PZD | "Standard telegram 111" | 286.0 | | |
| CONTROL_WORD | Struct | 286.0 | | |
| Control_Word_1 | Struct | 286.0 | | |
| Jog1 | Bool | 286.0 | false | TRUE |
| Jog2 | Bool | 286.1 | false | FALSE |
| LB | Bool | 286.2 | false | TRUE |
| RefStart | Bool | 286.3 | false | FALSE |
| Bit12 | Bool | 286.4 | false | FALSE |
| Bit13 | Bool | 286.5 | false | FALSE |
| Bit14 | Bool | 286.6 | false | FALSE |
| Bit15 | Bool | 286.7 | false | FALSE |
| Off1 | Bool | 287.0 | false | TRUE |
| Off2 | Bool | 287.1 | false | TRUE |
| Off3 | Bool | 287.2 | false | TRUE |
| Enc | Bool | 287.3 | false | TRUE |
| RejTrvTsk | Bool | 287.4 | false | TRUE |
| IntMStop | Bool | 287.5 | false | TRUE |
| TrvStart | Bool | 287.6 | false | FALSE |
| AckFault | Bool | 287.7 | false | FALSE |

(Fig. 3.48 B The values of command words of drive)

Drive output variables values are copied into the drive structure. In this way, the drive activation bits are set.

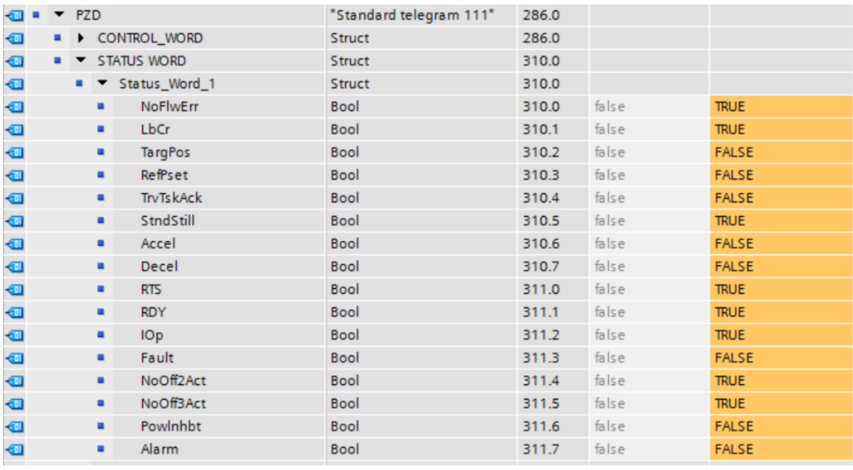
As a result, the drive performs the negative jog mode.

Proper operation is demonstrated by the word status of the drive (figure 3.49).



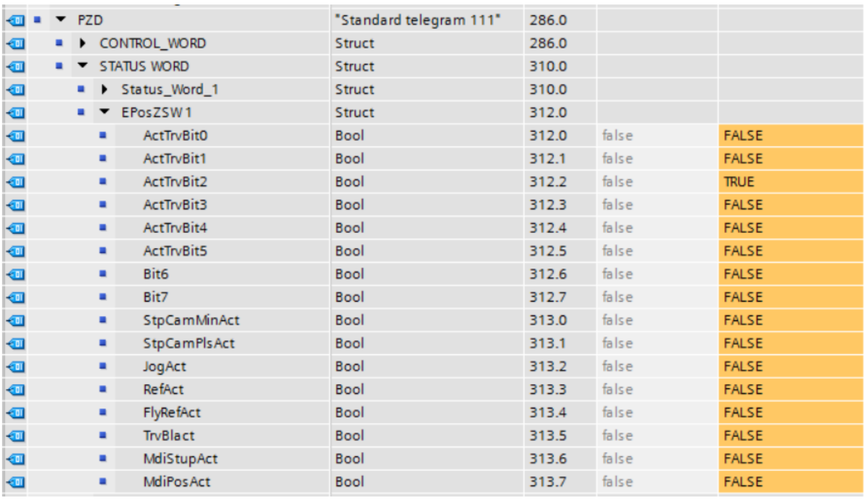
| | | | | |
|---|---|---|---|---|
| PZD | "Standard telegram 111" | 286.0 | | |
| CONTROL_WORD | Struct | 286.0 | | |
| STATUS WORD | Struct | 310.0 | | |
| Status_Word_1 | Struct | 310.0 | | |
| EPosZSW1 | Struct | 312.0 | | |
| EPosZSW2 | Struct | 314.0 | | |
| Status_Word_2 | Struct | 316.0 | | |
| Word6 | Word | 318.0 | 16#0 | 16#39CB |
| Position | DWord | 320.0 | 16#0 | 16#008A_DC33 |
| Velocity | DWord | 324.0 | 16#0 | 16#FFF0_4C01 |
| ErrNr | Word | 328.0 | 16#0 | 16#0000 |
| WarnNr | Word | 330.0 | 16#0 | 16#0000 |
| Reserved | Word | 332.0 | 16#0 | 16#0000 |

(Fig. 3.49 A The values of status words of drive)



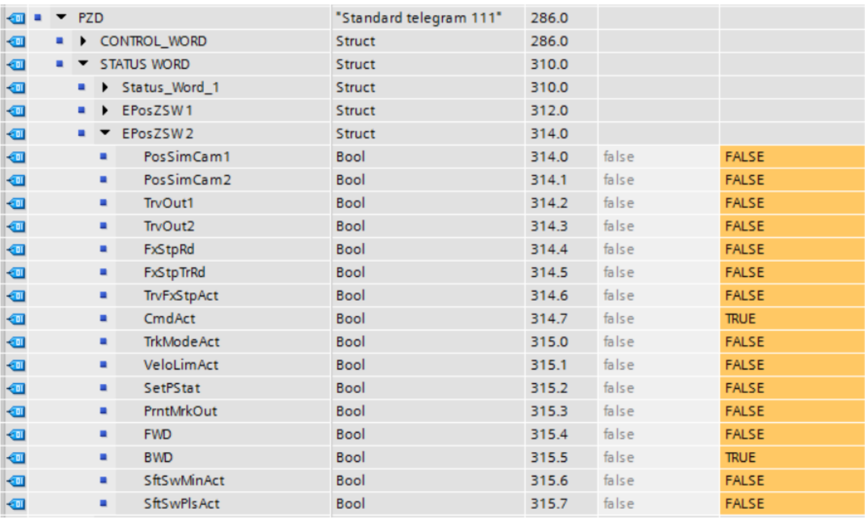
| | | | | |
|---|---|---|---|---|
| PZD | "Standard telegram 111" | 286.0 | | |
| CONTROL_WORD | Struct | 286.0 | | |
| STATUS WORD | Struct | 310.0 | | |
| Status_Word_1 | Struct | 310.0 | | |
| NoFlwErr | Bool | 310.0 | false | TRUE |
| LbCr | Bool | 310.1 | false | TRUE |
| TargPos | Bool | 310.2 | false | FALSE |
| RefPset | Bool | 310.3 | false | FALSE |
| TrvTskAck | Bool | 310.4 | false | FALSE |
| StndStill | Bool | 310.5 | false | TRUE |
| Accel | Bool | 310.6 | false | FALSE |
| Decel | Bool | 310.7 | false | FALSE |
| RTS | Bool | 311.0 | false | TRUE |
| RDY | Bool | 311.1 | false | TRUE |
| IOp | Bool | 311.2 | false | TRUE |
| Fault | Bool | 311.3 | false | FALSE |
| NoOff2Act | Bool | 311.4 | false | TRUE |
| NoOff3Act | Bool | 311.5 | false | TRUE |
| PowInhbt | Bool | 311.6 | false | FALSE |
| Alarm | Bool | 311.7 | false | FALSE |

(Fig. 3.49 B The values of status words of drive)



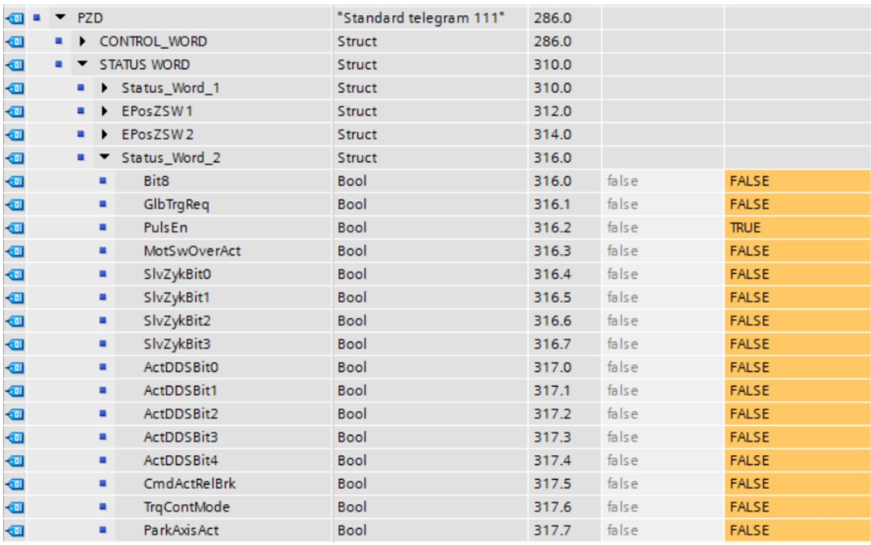
| | | | | |
|---|---|---|---|---|
| PZD | "Standard telegram 111" | 286.0 | | |
| CONTROL_WORD | Struct | 286.0 | | |
| STATUS WORD | Struct | 310.0 | | |
| Status_Word_1 | Struct | 310.0 | | |
| EPosZSW1 | Struct | 312.0 | | |
| ActTrvBit0 | Bool | 312.0 | false | FALSE |
| ActTrvBit1 | Bool | 312.1 | false | FALSE |
| ActTrvBit2 | Bool | 312.2 | false | TRUE |
| ActTrvBit3 | Bool | 312.3 | false | FALSE |
| ActTrvBit4 | Bool | 312.4 | false | FALSE |
| ActTrvBit5 | Bool | 312.5 | false | FALSE |
| Bit6 | Bool | 312.6 | false | FALSE |
| Bit7 | Bool | 312.7 | false | FALSE |
| StpCamMinAct | Bool | 313.0 | false | FALSE |
| StpCamPlsAct | Bool | 313.1 | false | FALSE |
| JogAct | Bool | 313.2 | false | FALSE |
| RefAct | Bool | 313.3 | false | FALSE |
| FlyRefAct | Bool | 313.4 | false | FALSE |
| TrvBlact | Bool | 313.5 | false | FALSE |
| MdiStupAct | Bool | 313.6 | false | FALSE |
| MdiPosAct | Bool | 313.7 | false | FALSE |

(Fig. 3.49 C The values of status words of drive)



| | | | | |
|---|---|---|---|---|
| PZD | "Standard telegram 111" | 286.0 | | |
| CONTROL_WORD | Struct | 286.0 | | |
| STATUS WORD | Struct | 310.0 | | |
| Status_Word_1 | Struct | 310.0 | | |
| EPosZSW1 | Struct | 312.0 | | |
| EPosZSW2 | Struct | 314.0 | | |
| PosSimCam1 | Bool | 314.0 | false | FALSE |
| PosSimCam2 | Bool | 314.1 | false | FALSE |
| TrvOut1 | Bool | 314.2 | false | FALSE |
| TrvOut2 | Bool | 314.3 | false | FALSE |
| FxStpRd | Bool | 314.4 | false | FALSE |
| FxStpTrRd | Bool | 314.5 | false | FALSE |
| TrvFxStpAct | Bool | 314.6 | false | FALSE |
| CmdAct | Bool | 314.7 | false | TRUE |
| TrkModeAct | Bool | 315.0 | false | FALSE |
| VeloLimAct | Bool | 315.1 | false | FALSE |
| SetPStat | Bool | 315.2 | false | FALSE |
| PrntMrkOut | Bool | 315.3 | false | FALSE |
| FWD | Bool | 315.4 | false | FALSE |
| BWD | Bool | 315.5 | false | TRUE |
| SftSwMinAct | Bool | 315.6 | false | FALSE |
| SftSwPlsAct | Bool | 315.7 | false | FALSE |

(Fig. 3.49 D The values of status words of drive)

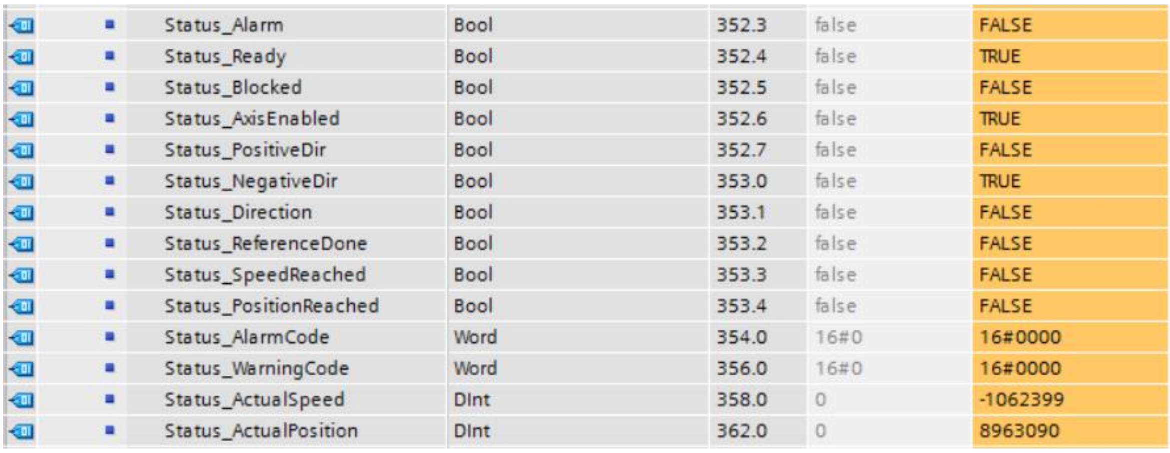


| | | | | |
|---|---|---|---|---|
| PZD | "Standard telegram 111" | 286.0 | | |
| CONTROL_WORD | Struct | 286.0 | | |
| STATUS WORD | Struct | 310.0 | | |
| Status_Word_1 | Struct | 310.0 | | |
| EPosZSW1 | Struct | 312.0 | | |
| EPosZSW2 | Struct | 314.0 | | |
| Status_Word_2 | Struct | 316.0 | | |
| Bit8 | Bool | 316.0 | false | FALSE |
| GlbTrgReq | Bool | 316.1 | false | FALSE |
| PulsEn | Bool | 316.2 | false | TRUE |
| MotSwOverAct | Bool | 316.3 | false | FALSE |
| SlvZykBit0 | Bool | 316.4 | false | FALSE |
| SlvZykBit1 | Bool | 316.5 | false | FALSE |
| SlvZykBit2 | Bool | 316.6 | false | FALSE |
| SlvZykBit3 | Bool | 316.7 | false | FALSE |
| ActDDSBit0 | Bool | 317.0 | false | FALSE |
| ActDDSBit1 | Bool | 317.1 | false | FALSE |
| ActDDSBit2 | Bool | 317.2 | false | FALSE |
| ActDDSBit3 | Bool | 317.3 | false | FALSE |
| ActDDSBit4 | Bool | 317.4 | false | FALSE |
| CmdActRelBrk | Bool | 317.5 | false | FALSE |
| TrqContMode | Bool | 317.6 | false | FALSE |
| ParkAxisAct | Bool | 317.7 | false | FALSE |

(Fig. 3.49 E The values of status words of drive)

It should be noted that the "jog1" bit of the word "Status_word_1" is equal to "TRUE"; indeed, this bit executes the negative jog mode.

In a manner similar to positive jog mode, drive status words are copied to the internal structure (figure 3.49). Figure 3.50 shows the most important information.



| | | | | |
|---|---|---|---|---|
| Status_Alarm | Bool | 352.3 | false | FALSE |
| Status_Ready | Bool | 352.4 | false | TRUE |
| Status_Blocked | Bool | 352.5 | false | FALSE |
| Status_AxisEnabled | Bool | 352.6 | false | TRUE |
| Status_PositiveDir | Bool | 352.7 | false | FALSE |
| Status_NegativeDir | Bool | 353.0 | false | TRUE |
| Status_Direction | Bool | 353.1 | false | FALSE |
| Status_ReferenceDone | Bool | 353.2 | false | FALSE |
| Status_SpeedReached | Bool | 353.3 | false | FALSE |
| Status_PositionReached | Bool | 353.4 | false | FALSE |
| Status_AlarmCode | Word | 354.0 | 16#0 | 16#0000 |
| Status_WarningCode | Word | 356.0 | 16#0 | 16#0000 |
| Status_ActualSpeed | DInt | 358.0 | 0 | -1062399 |
| Status_ActualPosition | DInt | 362.0 | 0 | 8963090 |

(Fig. 3.50 The values of drive input signals.)

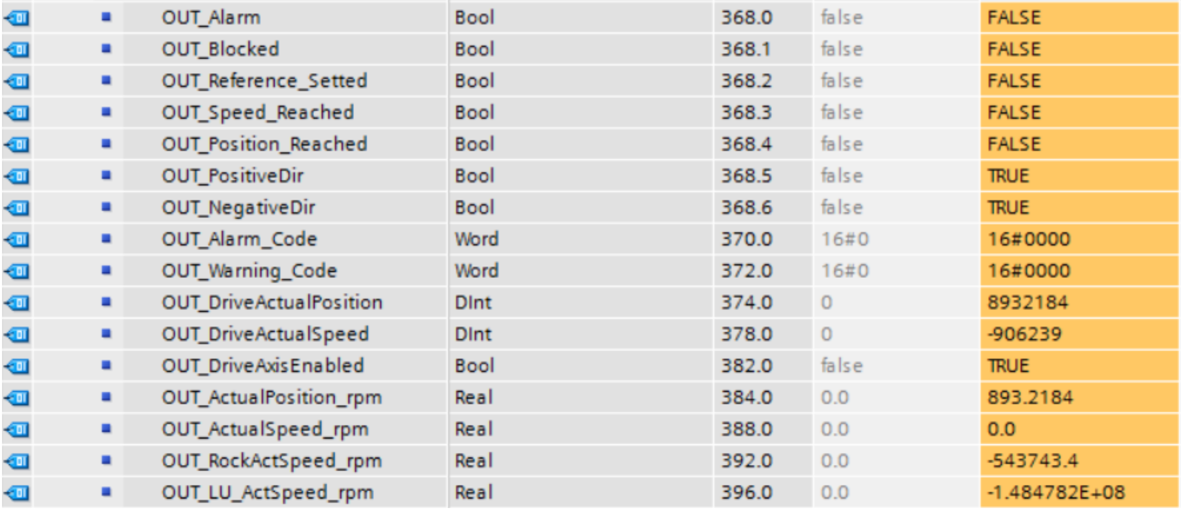
The correct functioning is highlighted by the following statuses of the figure 3.50:

• drive is ready state (Status_Ready = TRUE);

• the direction is negative (Status_NegativeDir = TRUE);

• there are not alarms (Status_Alarm = FALSE and Status_Warning = FALSE).

Finally, the figure 3.51 shows the parameters transmitted to the user.



| | | | | |
|---|---|---|---|---|
| OUT_Alarm | Bool | 368.0 | false | FALSE |
| OUT_Blocked | Bool | 368.1 | false | FALSE |
| OUT_Reference_Setted | Bool | 368.2 | false | FALSE |
| OUT_Speed_Reached | Bool | 368.3 | false | FALSE |
| OUT_Position_Reached | Bool | 368.4 | false | FALSE |
| OUT_PositiveDir | Bool | 368.5 | false | TRUE |
| OUT_NegativeDir | Bool | 368.6 | false | TRUE |
| OUT_Alarm_Code | Word | 370.0 | 16#0 | 16#0000 |
| OUT_Warning_Code | Word | 372.0 | 16#0 | 16#0000 |
| OUT_DriveActualPosition | DInt | 374.0 | 0 | 8932184 |
| OUT_DriveActualSpeed | DInt | 378.0 | 0 | -906239 |
| OUT_DriveAxisEnabled | Bool | 382.0 | false | TRUE |
| OUT_ActualPosition_rpm | Real | 384.0 | 0.0 | 893.2184 |
| OUT_ActualSpeed_rpm | Real | 388.0 | 0.0 | 0.0 |
| OUT_RockActSpeed_rpm | Real | 392.0 | 0.0 | -543743.4 |
| OUT_LU_ActSpeed_rpm | Real | 396.0 | 0.0 | -1.484782E+08 |

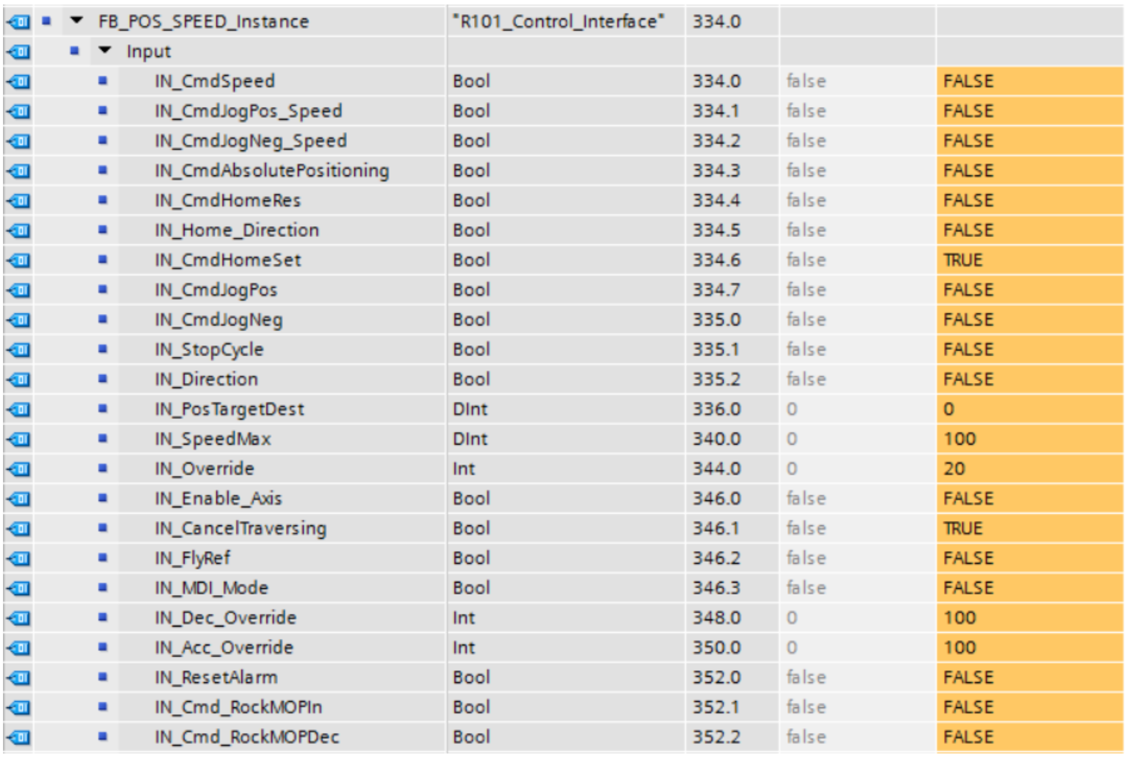
(Fig. 3.51 The values of user outputs)

In the third phase of the "Home Set" test mode is studied in detail. This mode is used to set the zero position, i.e. the reference position for the subsequent positions calculation.

The programmer must set the following parameters of the "R101_Control_Interface" block to execute the desired function:

- "IN_CmdHomeSet" = 1;
- "IN_StopCycle" = 0;
- "IN_CancelTraversing" =1.

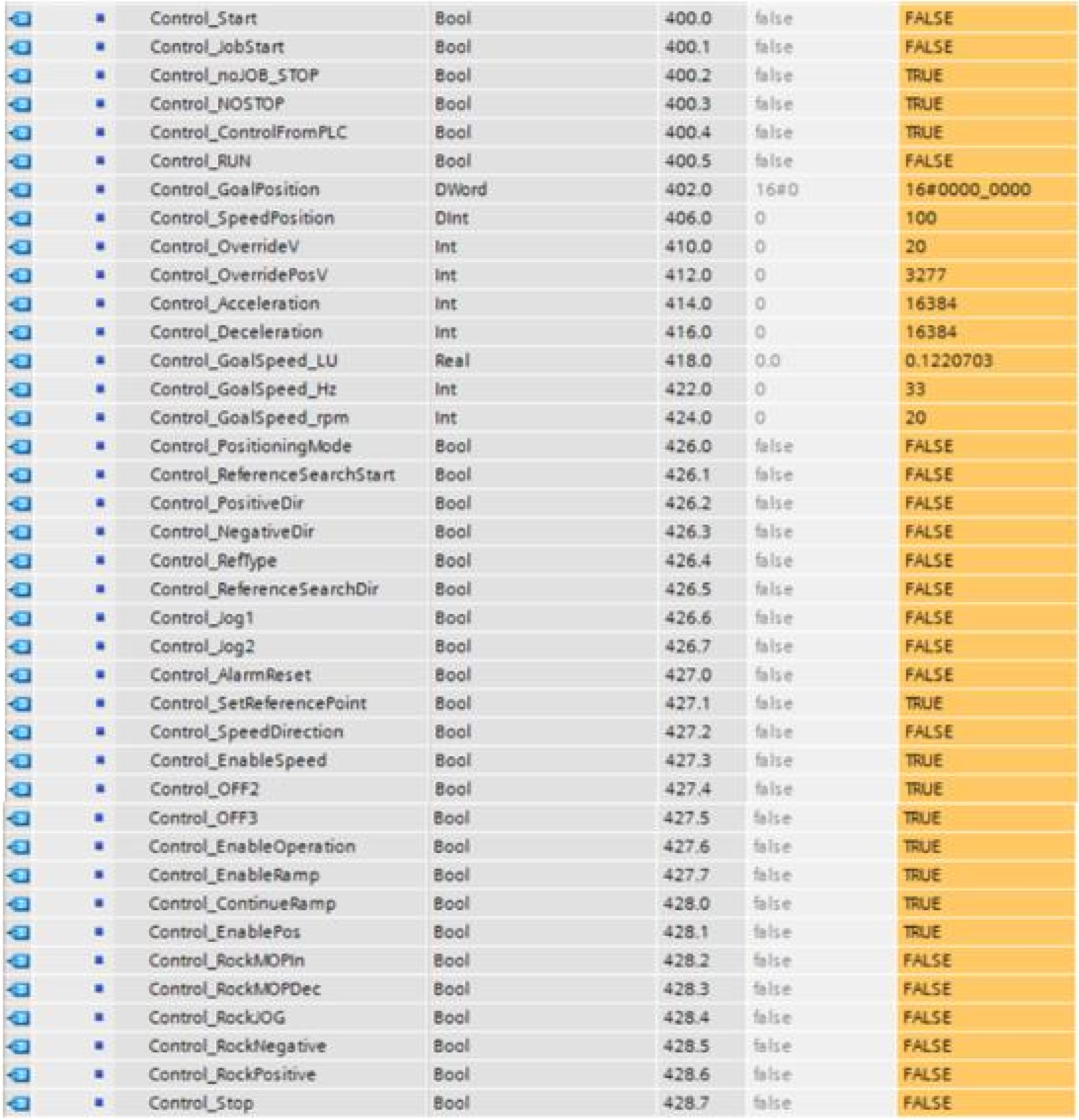
The correct user inputs to be set are shown in figure 3.52.



| | | | | |
|---|---|---|---|---|
| FB_POS_SPEED_Instance | "R101_Control_Interface" | 334.0 | | |
| Input | | | | |
| IN_CmdSpeed | Bool | 334.0 | false | FALSE |
| IN_CmdJogPos_Speed | Bool | 334.1 | false | FALSE |
| IN_CmdJogNeg_Speed | Bool | 334.2 | false | FALSE |
| IN_CmdAbsolutePositioning | Bool | 334.3 | false | FALSE |
| IN_CmdHomeRes | Bool | 334.4 | false | FALSE |
| IN_Home_Direction | Bool | 334.5 | false | FALSE |
| IN_CmdHomeSet | Bool | 334.6 | false | TRUE |
| IN_CmdJogPos | Bool | 334.7 | false | FALSE |
| IN_CmdJogNeg | Bool | 335.0 | false | FALSE |
| IN_StopCycle | Bool | 335.1 | false | FALSE |
| IN_Direction | Bool | 335.2 | false | FALSE |
| IN_PosTargetDest | DInt | 336.0 | 0 | 0 |
| IN_SpeedMax | DInt | 340.0 | 0 | 100 |
| IN_Override | Int | 344.0 | 0 | 20 |
| IN_Enable_Axis | Bool | 346.0 | false | FALSE |
| IN_CancelTraversing | Bool | 346.1 | false | TRUE |
| IN_FlyRef | Bool | 346.2 | false | FALSE |
| IN_MDI_Mode | Bool | 346.3 | false | FALSE |
| IN_Dec_Override | Int | 348.0 | 0 | 100 |
| IN_Acc_Override | Int | 350.0 | 0 | 100 |
| IN_ResetAlarm | Bool | 352.0 | false | FALSE |
| IN_Cmd_RockMOPIn | Bool | 352.1 | false | FALSE |
| IN_Cmd_RockMOPDec | Bool | 352.2 | false | FALSE |

(Fig. 3.52 The values of user inputs)

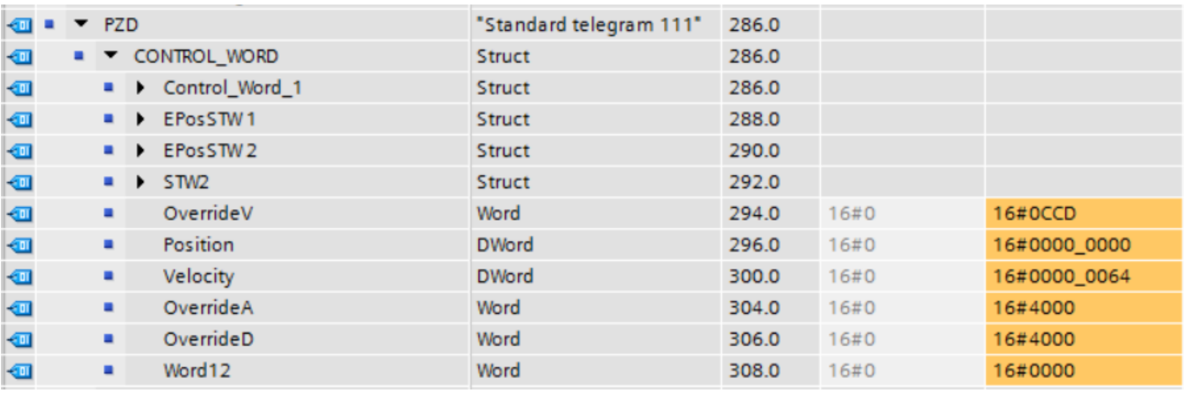
The R101 Control_Interface block reads user commands and generates the correct drive output values (figure 3.53).



| | | | | |
|---|---|---|---|---|
| Control_Start | Bool | 400.0 | false | FALSE |
| Control_JobStart | Bool | 400.1 | false | FALSE |
| Control_noJOB_STOP | Bool | 400.2 | false | TRUE |
| Control_NOSTOP | Bool | 400.3 | false | TRUE |
| Control_ControlFromPLC | Bool | 400.4 | false | TRUE |
| Control_RUN | Bool | 400.5 | false | FALSE |
| Control_GoalPosition | DWord | 402.0 | 16#0 | 16#0000_0000 |
| Control_SpeedPosition | DInt | 406.0 | 0 | 100 |
| Control_OverrideV | Int | 410.0 | 0 | 20 |
| Control_OverridePosV | Int | 412.0 | 0 | 3277 |
| Control_Acceleration | Int | 414.0 | 0 | 16384 |
| Control_Deceleration | Int | 416.0 | 0 | 16384 |
| Control_GoalSpeed_LU | Real | 418.0 | 0.0 | 0.1220703 |
| Control_GoalSpeed_Hz | Int | 422.0 | 0 | 33 |
| Control_GoalSpeed_rpm | Int | 424.0 | 0 | 20 |
| Control_PositioningMode | Bool | 426.0 | false | FALSE |
| Control_ReferenceSearchStart | Bool | 426.1 | false | FALSE |
| Control_PositiveDir | Bool | 426.2 | false | FALSE |
| Control_NegativeDir | Bool | 426.3 | false | FALSE |
| Control_RefType | Bool | 426.4 | false | FALSE |
| Control_ReferenceSearchDir | Bool | 426.5 | false | FALSE |
| Control_Jog1 | Bool | 426.6 | false | FALSE |
| Control_Jog2 | Bool | 426.7 | false | FALSE |
| Control_AlarmReset | Bool | 427.0 | false | FALSE |
| Control_SetReferencePoint | Bool | 427.1 | false | TRUE |
| Control_SpeedDirection | Bool | 427.2 | false | FALSE |
| Control_EnableSpeed | Bool | 427.3 | false | TRUE |
| Control_OFF2 | Bool | 427.4 | false | TRUE |
| Control_OFF3 | Bool | 427.5 | false | TRUE |
| Control_EnableOperation | Bool | 427.6 | false | TRUE |
| Control_EnableRamp | Bool | 427.7 | false | TRUE |
| Control_ContinueRamp | Bool | 428.0 | false | TRUE |
| Control_EnablePos | Bool | 428.1 | false | TRUE |
| Control_RockMOPIn | Bool | 428.2 | false | FALSE |
| Control_RockMOPDec | Bool | 428.3 | false | FALSE |
| Control_RockJOG | Bool | 428.4 | false | FALSE |
| Control_RockNegative | Bool | 428.5 | false | FALSE |
| Control_RockPositive | Bool | 428.6 | false | FALSE |
| Control_Stop | Bool | 428.7 | false | FALSE |

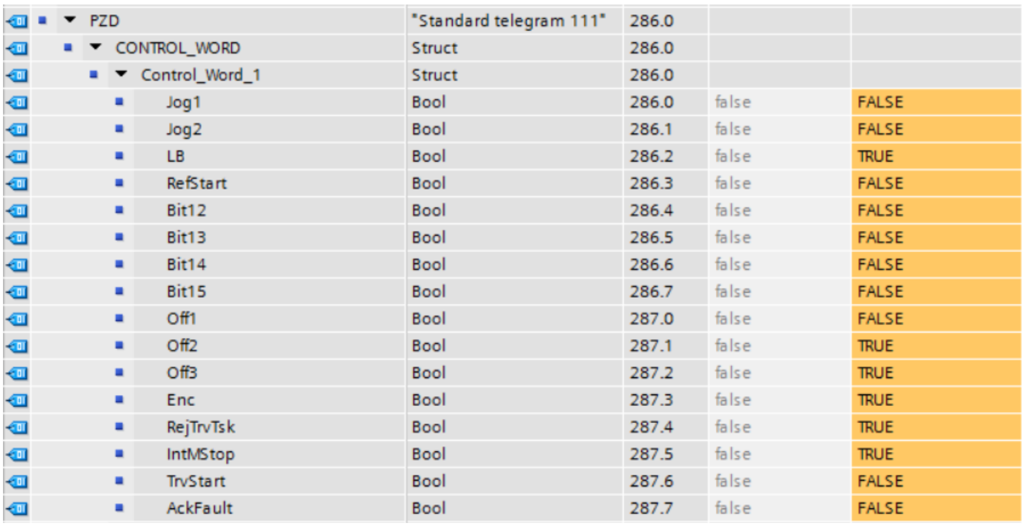
(Fig. 3.53 The values of drive outputs)

Each drive output is connected to internal structure (figure 3.54).



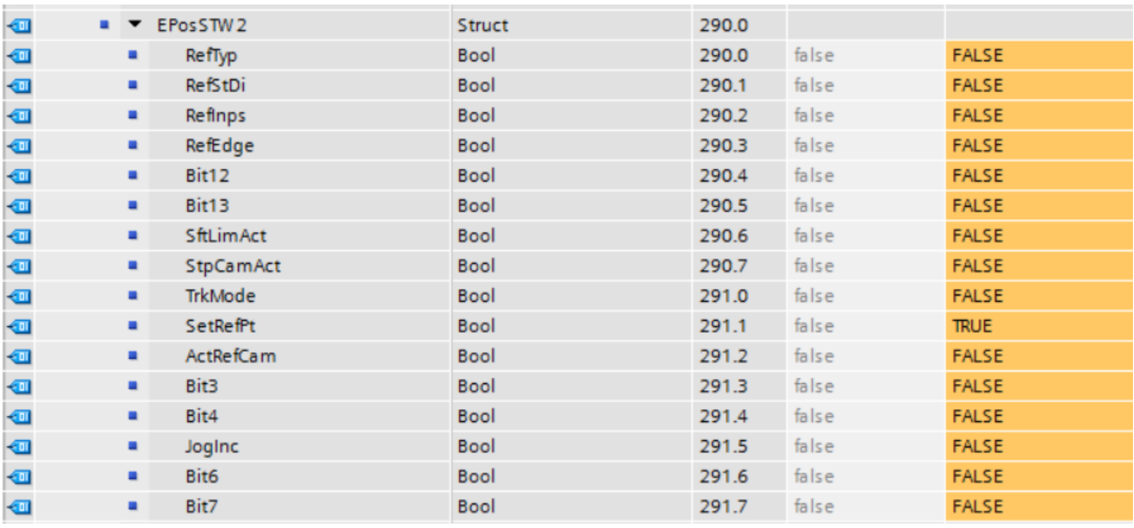
| | | | | |
|---|---|---|---|---|
| ▼ PZD | "Standard telegram 111" | 286.0 | | |
| ▼ CONTROL_WORD | Struct | 286.0 | | |
| ▶ Control_Word_1 | Struct | 286.0 | | |
| ▶ EPosSTW1 | Struct | 288.0 | | |
| ▶ EPosSTW2 | Struct | 290.0 | | |
| ▶ STW2 | Struct | 292.0 | | |
| OverrideV | Word | 294.0 | 16#0 | 16#0CCD |
| Position | DWord | 296.0 | 16#0 | 16#0000_0000 |
| Velocity | DWord | 300.0 | 16#0 | 16#0000_0064 |
| OverrideA | Word | 304.0 | 16#0 | 16#4000 |
| OverrideD | Word | 306.0 | 16#0 | 16#4000 |
| Word12 | Word | 308.0 | 16#0 | 16#0000 |

(Fig. 3.54 A The values of command words of drive)



| | | | | |
|---|---|---|---|---|
| PZD | "Standard telegram 111" | 286.0 | | |
| CONTROL_WORD | Struct | 286.0 | | |
| Control_Word_1 | Struct | 286.0 | | |
| Jog1 | Bool | 286.0 | false | FALSE |
| Jog2 | Bool | 286.1 | false | FALSE |
| LB | Bool | 286.2 | false | TRUE |
| RefStart | Bool | 286.3 | false | FALSE |
| Bit12 | Bool | 286.4 | false | FALSE |
| Bit13 | Bool | 286.5 | false | FALSE |
| Bit14 | Bool | 286.6 | false | FALSE |
| Bit15 | Bool | 286.7 | false | FALSE |
| Off1 | Bool | 287.0 | false | FALSE |
| Off2 | Bool | 287.1 | false | TRUE |
| Off3 | Bool | 287.2 | false | TRUE |
| Enc | Bool | 287.3 | false | TRUE |
| RejTrvTsk | Bool | 287.4 | false | TRUE |
| IntMStop | Bool | 287.5 | false | TRUE |
| TrvStart | Bool | 287.6 | false | FALSE |
| AckFault | Bool | 287.7 | false | FALSE |

(Fig. 3.54 B The values of command words of drive)



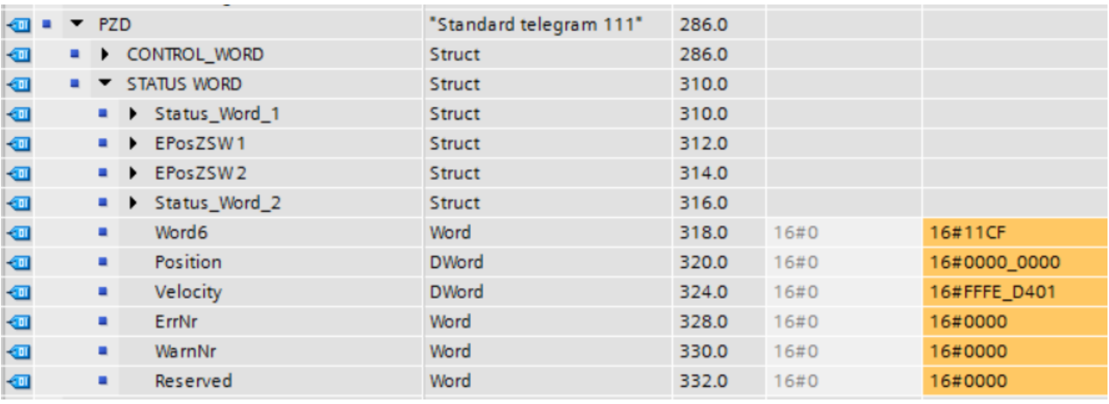
| | | | | |
|---|---|---|---|---|
| EPosSTW2 | Struct | 290.0 | | |
| RefTyp | Bool | 290.0 | false | FALSE |
| RefStDi | Bool | 290.1 | false | FALSE |
| RefInps | Bool | 290.2 | false | FALSE |
| RefEdge | Bool | 290.3 | false | FALSE |
| Bit12 | Bool | 290.4 | false | FALSE |
| Bit13 | Bool | 290.5 | false | FALSE |
| SftLimAct | Bool | 290.6 | false | FALSE |
| StpCamAct | Bool | 290.7 | false | FALSE |
| TrkMode | Bool | 291.0 | false | FALSE |
| SetRefPt | Bool | 291.1 | false | TRUE |
| ActRefCam | Bool | 291.2 | false | FALSE |
| Bit3 | Bool | 291.3 | false | FALSE |
| Bit4 | Bool | 291.4 | false | FALSE |
| JogInc | Bool | 291.5 | false | FALSE |
| Bit6 | Bool | 291.6 | false | FALSE |
| Bit7 | Bool | 291.7 | false | FALSE |

(Fig. 3.54 C The values of command words of drive)

Drive structure is compiled through the copy of the drive output variables values. This procedure let the execution of the “Home Set” function.

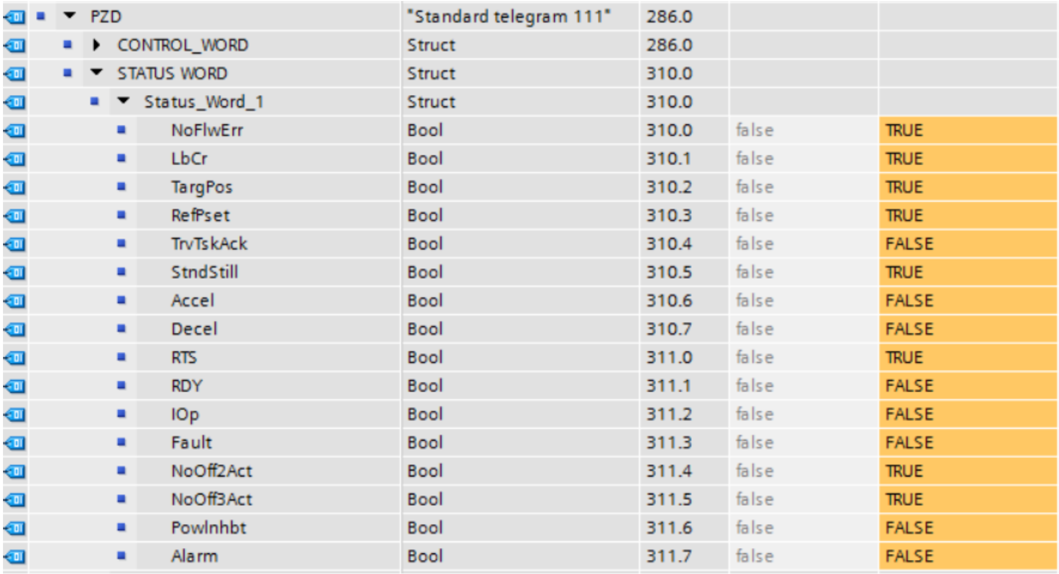
The "Home Set" command was transmitted to the drive by means of "SetRefPt" bit of the command word "EPosSTW2".

At this point the current position becomes the zero position (figure 3.55).



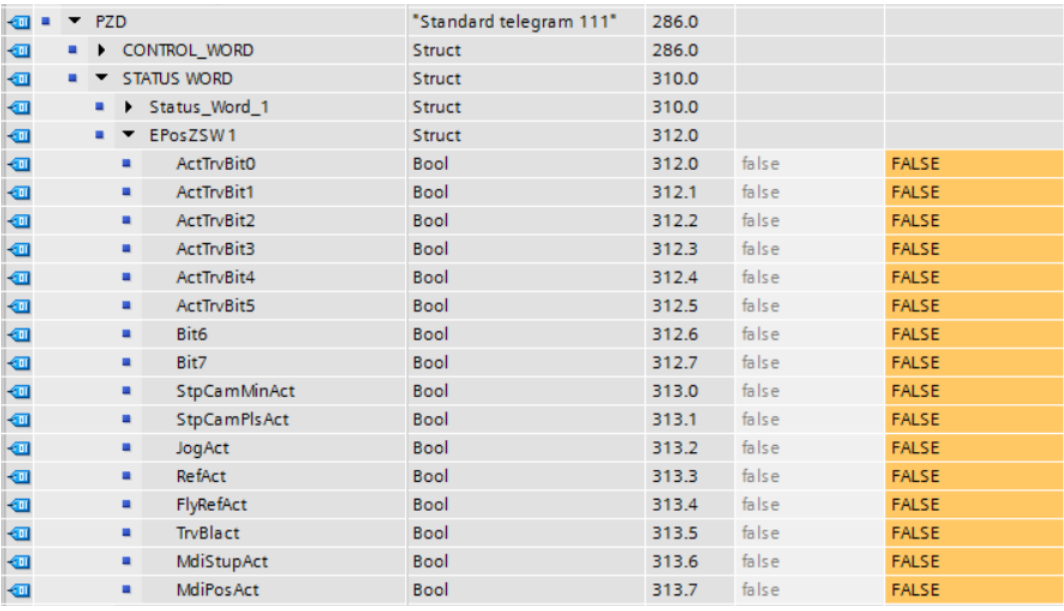
| | | | | |
|---|---|---|---|---|
| PZD | "Standard telegram 111" | 286.0 | | |
| CONTROL_WORD | Struct | 286.0 | | |
| STATUS WORD | Struct | 310.0 | | |
| Status_Word_1 | Struct | 310.0 | | |
| EPosZSW1 | Struct | 312.0 | | |
| EPosZSW2 | Struct | 314.0 | | |
| Status_Word_2 | Struct | 316.0 | | |
| Word6 | Word | 318.0 | 16#0 | 16#11CF |
| Position | DWord | 320.0 | 16#0 | 16#0000_0000 |
| Velocity | DWord | 324.0 | 16#0 | 16#FFFE_D401 |
| ErrNr | Word | 328.0 | 16#0 | 16#0000 |
| WarnNr | Word | 330.0 | 16#0 | 16#0000 |
| Reserved | Word | 332.0 | 16#0 | 16#0000 |

(Fig. 3.55 A The values of status words of drive)



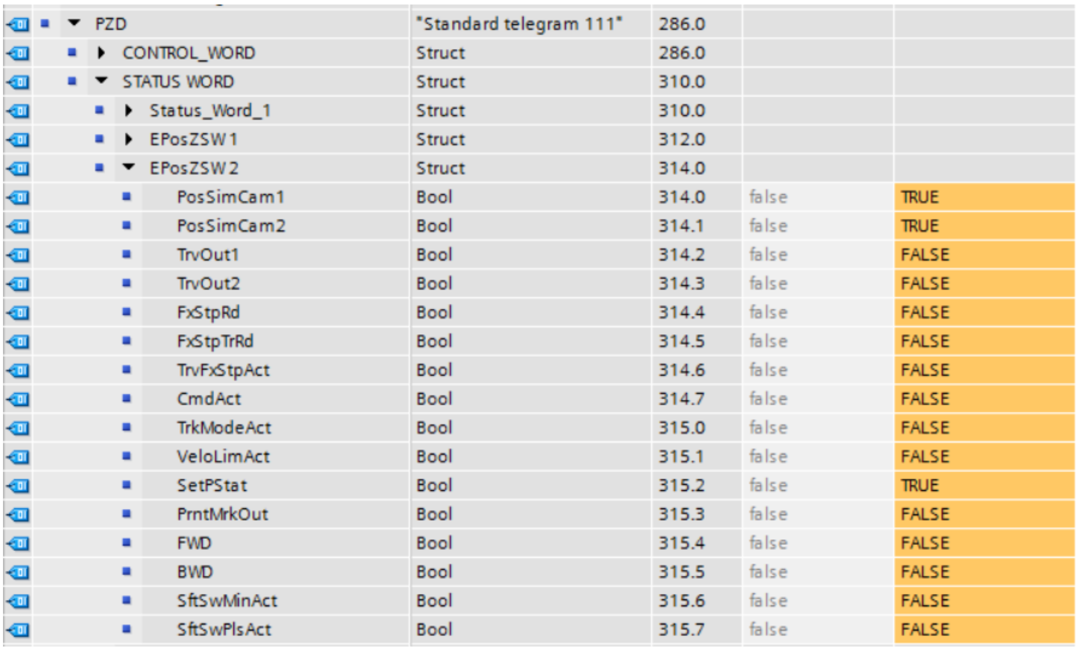
| | | | | |
|---|---|---|---|---|
| PZD | "Standard telegram 111" | 286.0 | | |
| CONTROL_WORD | Struct | 286.0 | | |
| STATUS WORD | Struct | 310.0 | | |
| Status_Word_1 | Struct | 310.0 | | |
| NoFlwErr | Bool | 310.0 | false | TRUE |
| LbCr | Bool | 310.1 | false | TRUE |
| TargPos | Bool | 310.2 | false | TRUE |
| RefPset | Bool | 310.3 | false | TRUE |
| TrvTskAck | Bool | 310.4 | false | FALSE |
| StndStill | Bool | 310.5 | false | TRUE |
| Accel | Bool | 310.6 | false | FALSE |
| Decel | Bool | 310.7 | false | FALSE |
| RTS | Bool | 311.0 | false | TRUE |
| RDY | Bool | 311.1 | false | FALSE |
| IOp | Bool | 311.2 | false | FALSE |
| Fault | Bool | 311.3 | false | FALSE |
| NoOff2Act | Bool | 311.4 | false | TRUE |
| NoOff3Act | Bool | 311.5 | false | TRUE |
| PowInhbt | Bool | 311.6 | false | FALSE |
| Alarm | Bool | 311.7 | false | FALSE |

(Fig. 3.55 B The values of status words of drive)



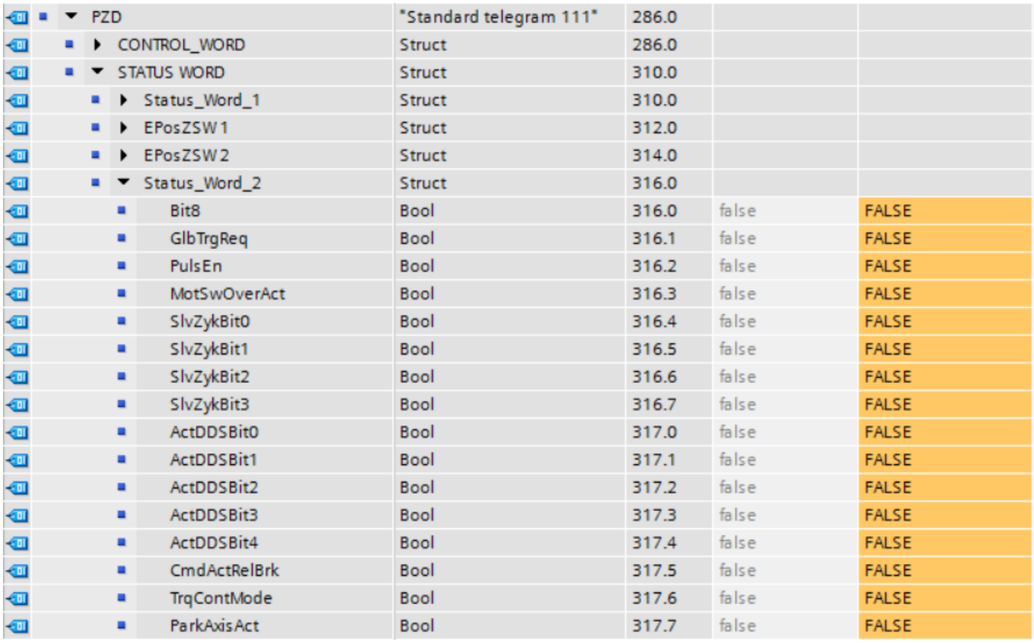
| | | | | |
|---|---|---|---|---|
| PZD | "Standard telegram 111" | 286.0 | | |
| CONTROL_WORD | Struct | 286.0 | | |
| STATUS WORD | Struct | 310.0 | | |
| Status_Word_1 | Struct | 310.0 | | |
| EPosZSW1 | Struct | 312.0 | | |
| ActTrvBit0 | Bool | 312.0 | false | FALSE |
| ActTrvBit1 | Bool | 312.1 | false | FALSE |
| ActTrvBit2 | Bool | 312.2 | false | FALSE |
| ActTrvBit3 | Bool | 312.3 | false | FALSE |
| ActTrvBit4 | Bool | 312.4 | false | FALSE |
| ActTrvBit5 | Bool | 312.5 | false | FALSE |
| Bit6 | Bool | 312.6 | false | FALSE |
| Bit7 | Bool | 312.7 | false | FALSE |
| StpCamMinAct | Bool | 313.0 | false | FALSE |
| StpCamPlsAct | Bool | 313.1 | false | FALSE |
| JogAct | Bool | 313.2 | false | FALSE |
| RefAct | Bool | 313.3 | false | FALSE |
| FlyRefAct | Bool | 313.4 | false | FALSE |
| TrvBlact | Bool | 313.5 | false | FALSE |
| MdiStupAct | Bool | 313.6 | false | FALSE |
| MdiPosAct | Bool | 313.7 | false | FALSE |

(Fig. 3.55 C The values of status words of drive)



| | | | | |
|---|---|---|---|---|
| PZD | "Standard telegram 111" | 286.0 | | |
| CONTROL_WORD | Struct | 286.0 | | |
| STATUS WORD | Struct | 310.0 | | |
| Status_Word_1 | Struct | 310.0 | | |
| EPosZSW1 | Struct | 312.0 | | |
| EPosZSW2 | Struct | 314.0 | | |
| PosSimCam1 | Bool | 314.0 | false | TRUE |
| PosSimCam2 | Bool | 314.1 | false | TRUE |
| TrvOut1 | Bool | 314.2 | false | FALSE |
| TrvOut2 | Bool | 314.3 | false | FALSE |
| FxStpRd | Bool | 314.4 | false | FALSE |
| FxStpTrRd | Bool | 314.5 | false | FALSE |
| TrvFxStpAct | Bool | 314.6 | false | FALSE |
| CmdAct | Bool | 314.7 | false | FALSE |
| TrkModeAct | Bool | 315.0 | false | FALSE |
| VeloLimAct | Bool | 315.1 | false | FALSE |
| SetPStat | Bool | 315.2 | false | TRUE |
| PrntMrkOut | Bool | 315.3 | false | FALSE |
| FWD | Bool | 315.4 | false | FALSE |
| BWD | Bool | 315.5 | false | FALSE |
| SftSwMinAct | Bool | 315.6 | false | FALSE |
| SftSwPlsAct | Bool | 315.7 | false | FALSE |

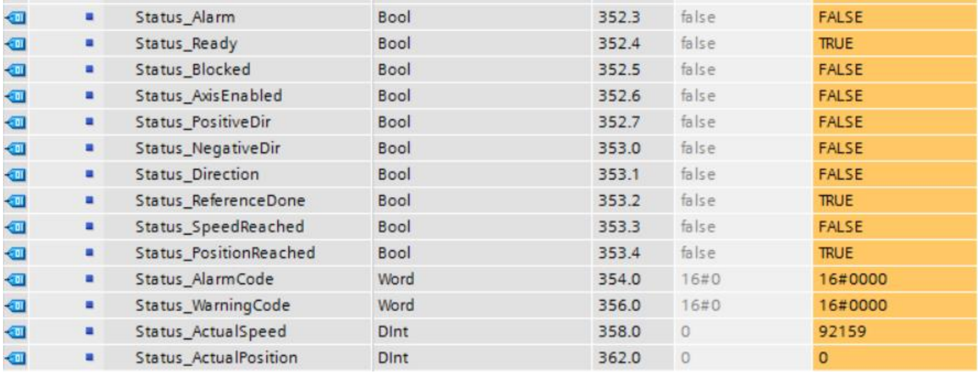
(Fig. 3.55 D The values of status words of drive)



| | | | | |
|---|---|---|---|---|
| PZD | "Standard telegram 111" | 286.0 | | |
| CONTROL_WORD | Struct | 286.0 | | |
| STATUS WORD | Struct | 310.0 | | |
| Status_Word_1 | Struct | 310.0 | | |
| EPosZSW1 | Struct | 312.0 | | |
| EPosZSW2 | Struct | 314.0 | | |
| Status_Word_2 | Struct | 316.0 | | |
| Bit8 | Bool | 316.0 | false | FALSE |
| GlbTrgReq | Bool | 316.1 | false | FALSE |
| PulsEn | Bool | 316.2 | false | FALSE |
| MotSwOverAct | Bool | 316.3 | false | FALSE |
| SlvZykBit0 | Bool | 316.4 | false | FALSE |
| SlvZykBit1 | Bool | 316.5 | false | FALSE |
| SlvZykBit2 | Bool | 316.6 | false | FALSE |
| SlvZykBit3 | Bool | 316.7 | false | FALSE |
| ActDDSBit0 | Bool | 317.0 | false | FALSE |
| ActDDSBit1 | Bool | 317.1 | false | FALSE |
| ActDDSBit2 | Bool | 317.2 | false | FALSE |
| ActDDSBit3 | Bool | 317.3 | false | FALSE |
| ActDDSBit4 | Bool | 317.4 | false | FALSE |
| CmdActRelBrk | Bool | 317.5 | false | FALSE |
| TrqContMode | Bool | 317.6 | false | FALSE |
| ParkAxisAct | Bool | 317.7 | false | FALSE |

(Fig. 3.55 E The values of status words of drive)

In a manner similar to previous tests, drive status words are copied to the internal structure (figure 3.55). Figure 3.56 shows the most important information.

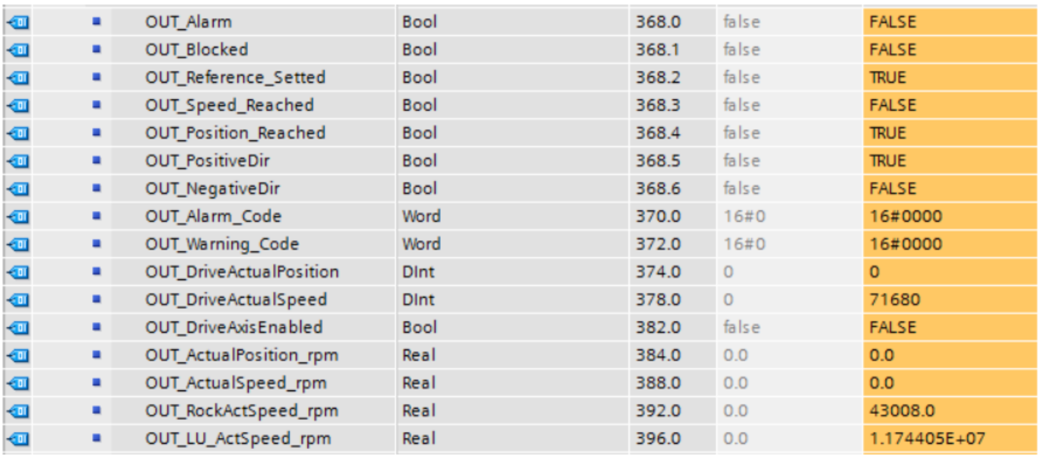


| | | | | |
|---|---|---|---|---|
| Status_Alarm | Bool | 352.3 | false | FALSE |
| Status_Ready | Bool | 352.4 | false | TRUE |
| Status_Blocked | Bool | 352.5 | false | FALSE |
| Status_AxisEnabled | Bool | 352.6 | false | FALSE |
| Status_PositiveDir | Bool | 352.7 | false | FALSE |
| Status_NegativeDir | Bool | 353.0 | false | FALSE |
| Status_Direction | Bool | 353.1 | false | FALSE |
| Status_ReferenceDone | Bool | 353.2 | false | TRUE |
| Status_SpeedReached | Bool | 353.3 | false | FALSE |
| Status_PositionReached | Bool | 353.4 | false | TRUE |
| Status_AlarmCode | Word | 354.0 | 16#0 | 16#0000 |
| Status_WarningCode | Word | 356.0 | 16#0 | 16#0000 |
| Status_ActualSpeed | DInt | 358.0 | 0 | 92159 |
| Status_ActualPosition | DInt | 362.0 | 0 | 0 |

(Fig. 3.56 The values of drive input signals)

It should be noted that the actual position is zero, indeed Status_ActualPosition = 0 (figure 3.56) and Position = 16#0000_0000 (figure 3.55 A), while previous value of position is different from zero (figure 3.49 A). In addition, the value "Status_ReferenceDone" = TRUE. This value demonstrates the correct functioning of the drive.

The transmission of drive states to the user is performed by user outputs (figure 3.57).



| | | | | |
|---|---|---|---|---|
| OUT_Alarm | Bool | 368.0 | false | FALSE |
| OUT_Blocked | Bool | 368.1 | false | FALSE |
| OUT_Reference_Setted | Bool | 368.2 | false | TRUE |
| OUT_Speed_Reached | Bool | 368.3 | false | FALSE |
| OUT_Position_Reached | Bool | 368.4 | false | TRUE |
| OUT_PositiveDir | Bool | 368.5 | false | TRUE |
| OUT_NegativeDir | Bool | 368.6 | false | FALSE |
| OUT_Alarm_Code | Word | 370.0 | 16#0 | 16#0000 |
| OUT_Warning_Code | Word | 372.0 | 16#0 | 16#0000 |
| OUT_DriveActualPosition | DInt | 374.0 | 0 | 0 |
| OUT_DriveActualSpeed | DInt | 378.0 | 0 | 71680 |
| OUT_DriveAxisEnabled | Bool | 382.0 | false | FALSE |
| OUT_ActualPosition_rpm | Real | 384.0 | 0.0 | 0.0 |
| OUT_ActualSpeed_rpm | Real | 388.0 | 0.0 | 0.0 |
| OUT_RockActSpeed_rpm | Real | 392.0 | 0.0 | 43008.0 |
| OUT_LU_ActSpeed_rpm | Real | 396.0 | 0.0 | 1.174405E+07 |

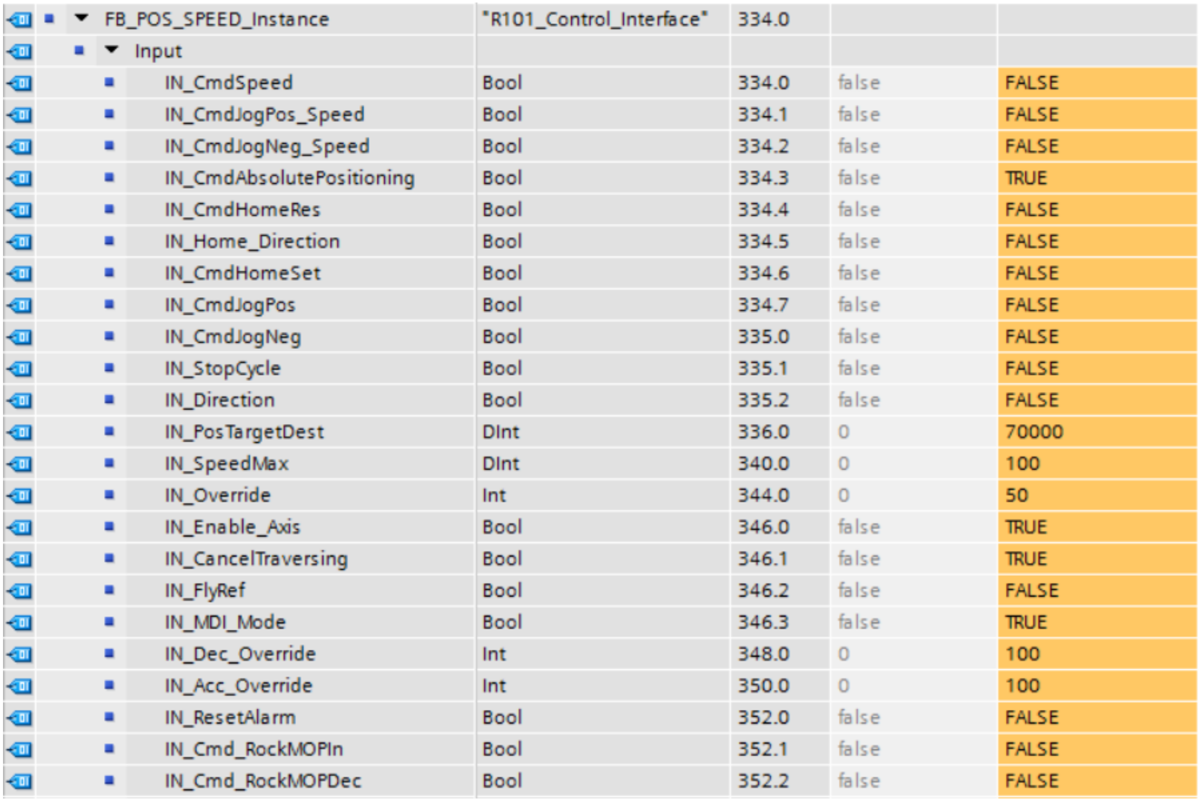
(Fig. 3.57 The values of user outputs)

The fourth phase of the test is the execution of the positive absolute position mode. To perform this mode, the user must impose the following parameters:

- “IN_CmdAbsolutePositioning” = 1;
- “IN_StopCycle” = 0;
- “IN_Direction” = 0;
- “IN_PosTargetDest” = 70000;
- “IN_SpeedMax” = 100 [%];
- “IN_Override” = 50 [%];
- “IN_Enable_Axis” = 1;

- “IN_CancelTraversing” =1;
- “IN_MDI_Mode” = 1.

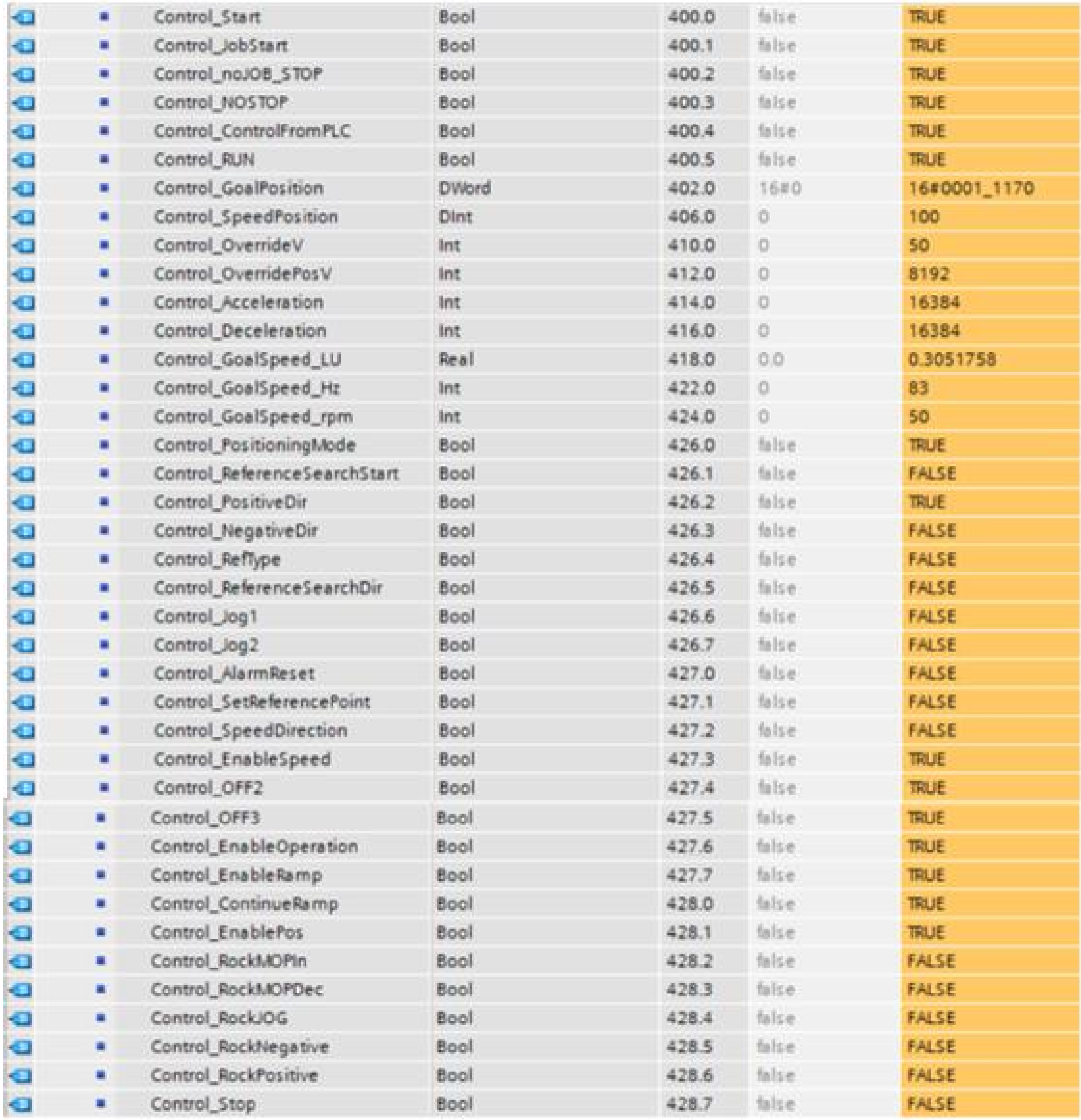
These parameters are set by means of user inputs (figure 3.58).



| | | | | |
|---|---|---|---|---|
| ▾ FB_POS_SPEED_Instance | "R101_Control_Interface" | 334.0 | | |
| ▾ Input | | | | |
| IN_CmdSpeed | Bool | 334.0 | false | FALSE |
| IN_CmdJogPos_Speed | Bool | 334.1 | false | FALSE |
| IN_CmdJogNeg_Speed | Bool | 334.2 | false | FALSE |
| IN_CmdAbsolutePositioning | Bool | 334.3 | false | TRUE |
| IN_CmdHomeRes | Bool | 334.4 | false | FALSE |
| IN_Home_Direction | Bool | 334.5 | false | FALSE |
| IN_CmdHomeSet | Bool | 334.6 | false | FALSE |
| IN_CmdJogPos | Bool | 334.7 | false | FALSE |
| IN_CmdJogNeg | Bool | 335.0 | false | FALSE |
| IN_StopCycle | Bool | 335.1 | false | FALSE |
| IN_Direction | Bool | 335.2 | false | FALSE |
| IN_PosTargetDest | DInt | 336.0 | 0 | 70000 |
| IN_SpeedMax | DInt | 340.0 | 0 | 100 |
| IN_Override | Int | 344.0 | 0 | 50 |
| IN_Enable_Axis | Bool | 346.0 | false | TRUE |
| IN_CancelTraversing | Bool | 346.1 | false | TRUE |
| IN_FlyRef | Bool | 346.2 | false | FALSE |
| IN_MDI_Mode | Bool | 346.3 | false | TRUE |
| IN_Dec_Override | Int | 348.0 | 0 | 100 |
| IN_Acc_Override | Int | 350.0 | 0 | 100 |
| IN_ResetAlarm | Bool | 352.0 | false | FALSE |
| IN_Cmd_RockMOPIn | Bool | 352.1 | false | FALSE |
| IN_Cmd_RockMOPDec | Bool | 352.2 | false | FALSE |

(Fig. 3.58 User inputs)

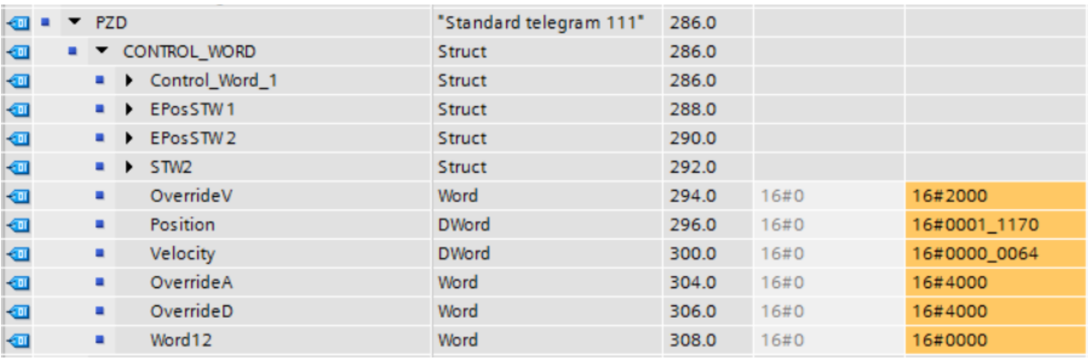
The "R101_Control_Interface" block processes the commands that must be sent to the servo drive according to user inputs (figure 3.59).



| | | | | |
|---|---|---|---|---|
| Control_Start | Bool | 400.0 | false | TRUE |
| Control_JobStart | Bool | 400.1 | false | TRUE |
| Control_noJOB_STOP | Bool | 400.2 | false | TRUE |
| Control_NOSTOP | Bool | 400.3 | false | TRUE |
| Control_ControlFromPLC | Bool | 400.4 | false | TRUE |
| Control_RUN | Bool | 400.5 | false | TRUE |
| Control_GoalPosition | DWord | 402.0 | 16#0 | 16#0001_1170 |
| Control_SpeedPosition | DInt | 406.0 | 0 | 100 |
| Control_OverrideV | Int | 410.0 | 0 | 50 |
| Control_OverridePosV | Int | 412.0 | 0 | 8192 |
| Control_Acceleration | Int | 414.0 | 0 | 16384 |
| Control_Deceleration | Int | 416.0 | 0 | 16384 |
| Control_GoalSpeed_LU | Real | 418.0 | 0.0 | 0.3051758 |
| Control_GoalSpeed_Hz | Int | 422.0 | 0 | 83 |
| Control_GoalSpeed_rpm | Int | 424.0 | 0 | 50 |
| Control_PositioningMode | Bool | 426.0 | false | TRUE |
| Control_ReferenceSearchStart | Bool | 426.1 | false | FALSE |
| Control_PositiveDir | Bool | 426.2 | false | TRUE |
| Control_NegativeDir | Bool | 426.3 | false | FALSE |
| Control_RefType | Bool | 426.4 | false | FALSE |
| Control_ReferenceSearchDir | Bool | 426.5 | false | FALSE |
| Control_Jog1 | Bool | 426.6 | false | FALSE |
| Control_Jog2 | Bool | 426.7 | false | FALSE |
| Control_AlarmReset | Bool | 427.0 | false | FALSE |
| Control_SetReferencePoint | Bool | 427.1 | false | FALSE |
| Control_SpeedDirection | Bool | 427.2 | false | FALSE |
| Control_EnableSpeed | Bool | 427.3 | false | TRUE |
| Control_OFF2 | Bool | 427.4 | false | TRUE |
| Control_OFF3 | Bool | 427.5 | false | TRUE |
| Control_EnableOperation | Bool | 427.6 | false | TRUE |
| Control_EnableRamp | Bool | 427.7 | false | TRUE |
| Control_ContinueRamp | Bool | 428.0 | false | TRUE |
| Control_EnablePos | Bool | 428.1 | false | TRUE |
| Control_RockMOPIn | Bool | 428.2 | false | FALSE |
| Control_RockMOPDec | Bool | 428.3 | false | FALSE |
| Control_RockJOG | Bool | 428.4 | false | FALSE |
| Control_RockNegative | Bool | 428.5 | false | FALSE |
| Control_RockPositive | Bool | 428.6 | false | FALSE |
| Control_Stop | Bool | 428.7 | false | FALSE |

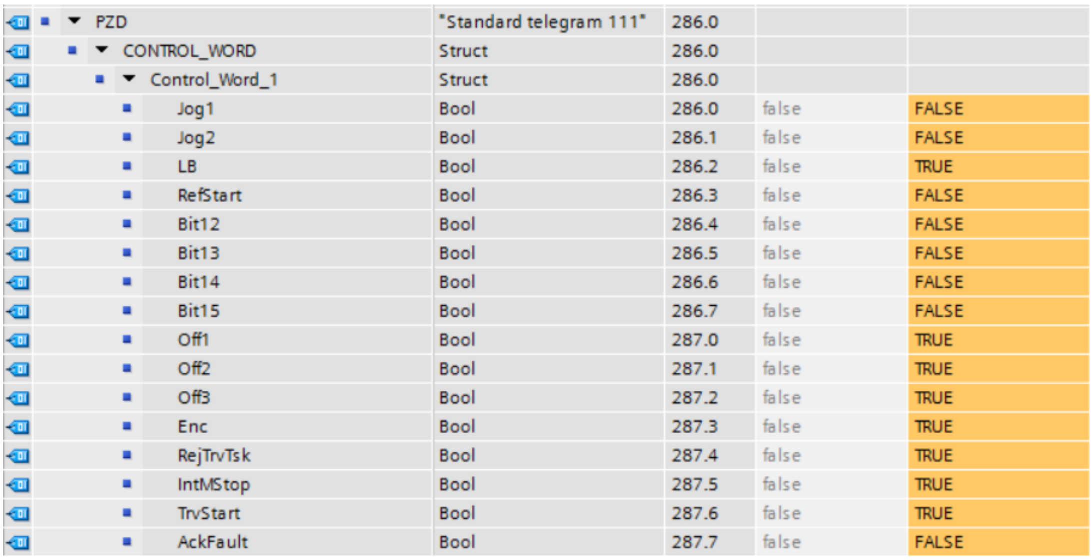
(Fig. 3.59 The values of drive outputs)

The search for the final position is performed if a series of drive bits are active (figure 3.60).



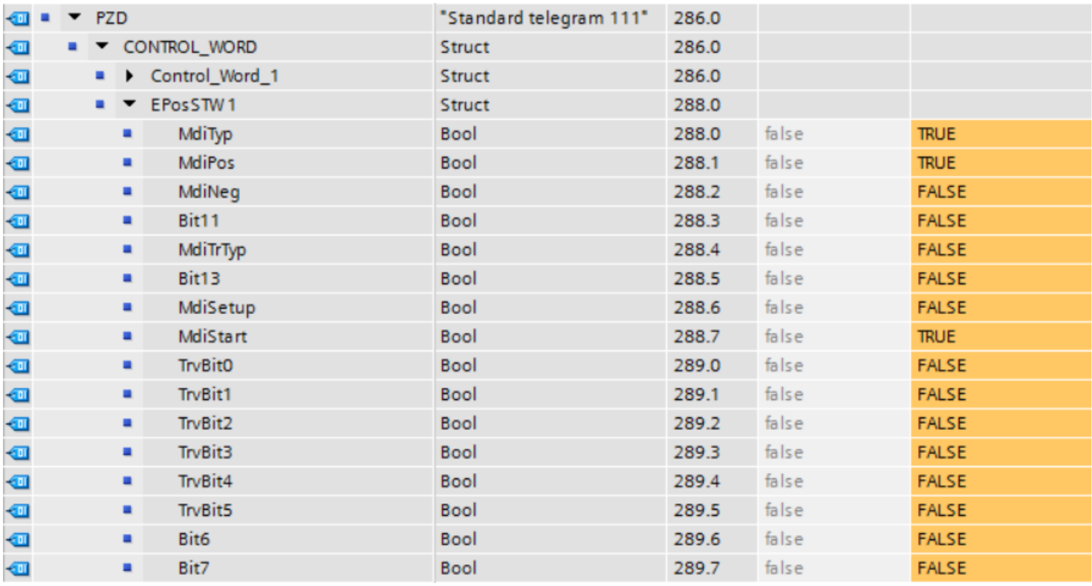
| | | | | |
|---|---|---|---|---|
| ▼ PZD | "Standard telegram 111" | 286.0 | | |
| ▼ CONTROL_WORD | Struct | 286.0 | | |
| ▶ Control_Word_1 | Struct | 286.0 | | |
| ▶ EPosSTW1 | Struct | 288.0 | | |
| ▶ EPosSTW2 | Struct | 290.0 | | |
| ▶ STW2 | Struct | 292.0 | | |
| OverrideV | Word | 294.0 | 16#0 | 16#2000 |
| Position | DWord | 296.0 | 16#0 | 16#0001_1170 |
| Velocity | DWord | 300.0 | 16#0 | 16#0000_0064 |
| OverrideA | Word | 304.0 | 16#0 | 16#4000 |
| OverrideD | Word | 306.0 | 16#0 | 16#4000 |
| Word12 | Word | 308.0 | 16#0 | 16#0000 |

(Fig. 3.60 A The values of command words of drive)



| | | | | |
|---|---|---|---|---|
| PZD | "Standard telegram 111" | 286.0 | | |
| CONTROL_WORD | Struct | 286.0 | | |
| Control_Word_1 | Struct | 286.0 | | |
| Jog1 | Bool | 286.0 | false | FALSE |
| Jog2 | Bool | 286.1 | false | FALSE |
| LB | Bool | 286.2 | false | TRUE |
| RefStart | Bool | 286.3 | false | FALSE |
| Bit12 | Bool | 286.4 | false | FALSE |
| Bit13 | Bool | 286.5 | false | FALSE |
| Bit14 | Bool | 286.6 | false | FALSE |
| Bit15 | Bool | 286.7 | false | FALSE |
| Off1 | Bool | 287.0 | false | TRUE |
| Off2 | Bool | 287.1 | false | TRUE |
| Off3 | Bool | 287.2 | false | TRUE |
| Enc | Bool | 287.3 | false | TRUE |
| RejTrvTsk | Bool | 287.4 | false | TRUE |
| IntMStop | Bool | 287.5 | false | TRUE |
| TrvStart | Bool | 287.6 | false | TRUE |
| AckFault | Bool | 287.7 | false | FALSE |

(Fig. 3.60 B The values of command words of drive)

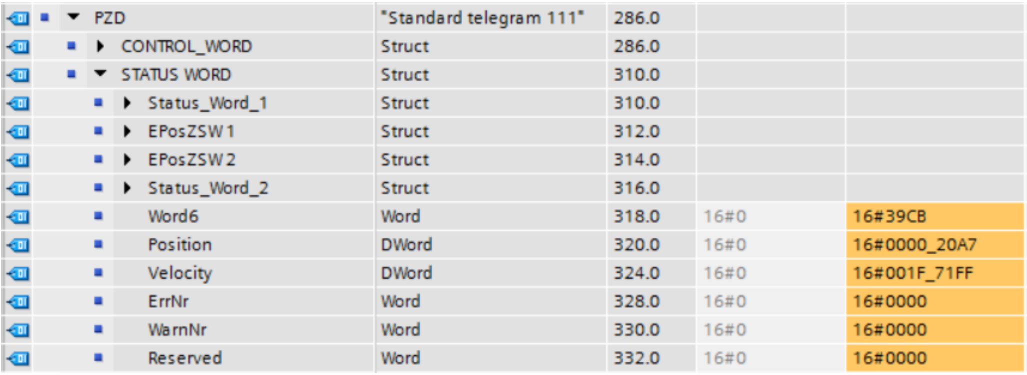


| | | | | |
|---|---|---|---|---|
| PZD | "Standard telegram 111" | 286.0 | | |
| CONTROL_WORD | Struct | 286.0 | | |
| Control_Word_1 | Struct | 286.0 | | |
| EPosSTW1 | Struct | 288.0 | | |
| MdiTyp | Bool | 288.0 | false | TRUE |
| MdiPos | Bool | 288.1 | false | TRUE |
| MdiNeg | Bool | 288.2 | false | FALSE |
| Bit11 | Bool | 288.3 | false | FALSE |
| MdiTrTyp | Bool | 288.4 | false | FALSE |
| Bit13 | Bool | 288.5 | false | FALSE |
| MdiSetup | Bool | 288.6 | false | FALSE |
| MdiStart | Bool | 288.7 | false | TRUE |
| TrvBit0 | Bool | 289.0 | false | FALSE |
| TrvBit1 | Bool | 289.1 | false | FALSE |
| TrvBit2 | Bool | 289.2 | false | FALSE |
| TrvBit3 | Bool | 289.3 | false | FALSE |
| TrvBit4 | Bool | 289.4 | false | FALSE |
| TrvBit5 | Bool | 289.5 | false | FALSE |
| Bit6 | Bool | 289.6 | false | FALSE |
| Bit7 | Bool | 289.7 | false | FALSE |

(Fig. 3.60 C The values of command words of drive)

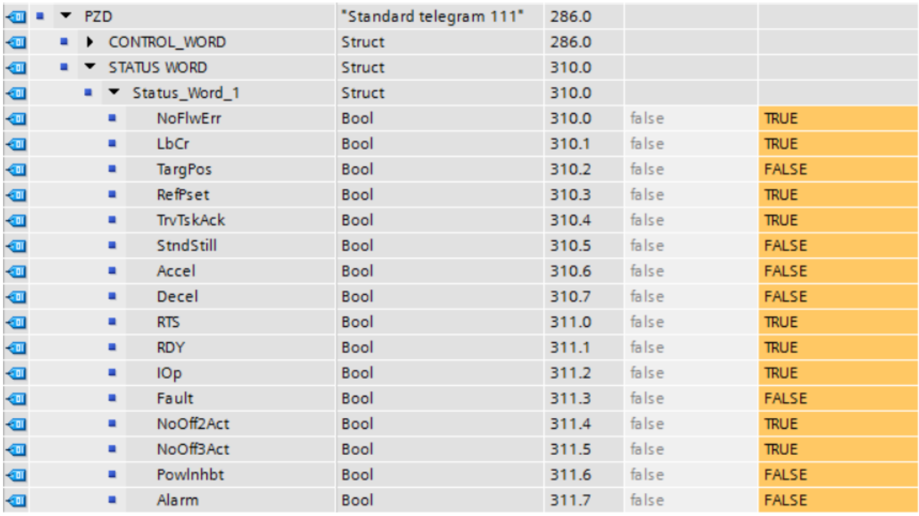
These bits are activated by copying the drive output variables values into the drive structure. At this point the engine starts spinning in a positive direction until it reaches the desired position.

The drive status words are copied to the internal structure (figure 3.61) and the most important bits are connected to the drive input signals (figure 3.62).



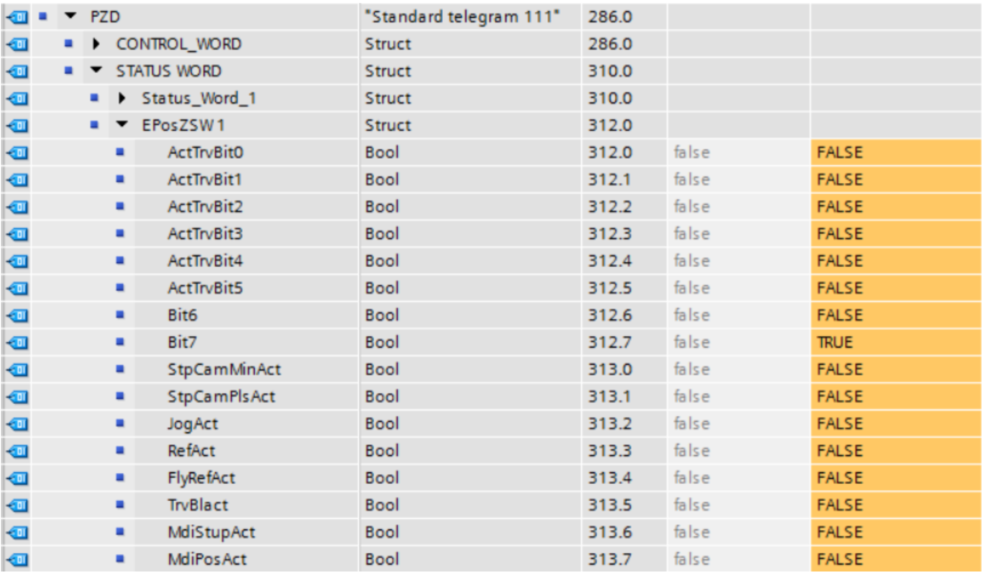
| | | | | |
|---|---|---|---|---|
| PZD | "Standard telegram 111" | 286.0 | | |
| CONTROL_WORD | Struct | 286.0 | | |
| STATUS WORD | Struct | 310.0 | | |
| Status_Word_1 | Struct | 310.0 | | |
| EPosZSW1 | Struct | 312.0 | | |
| EPosZSW2 | Struct | 314.0 | | |
| Status_Word_2 | Struct | 316.0 | | |
| Word6 | Word | 318.0 | 16#0 | 16#39CB |
| Position | DWord | 320.0 | 16#0 | 16#0000_20A7 |
| Velocity | DWord | 324.0 | 16#0 | 16#001F_71FF |
| ErrNr | Word | 328.0 | 16#0 | 16#0000 |
| WarnNr | Word | 330.0 | 16#0 | 16#0000 |
| Reserved | Word | 332.0 | 16#0 | 16#0000 |

(Fig. 3.61 A The values of status words of drive)



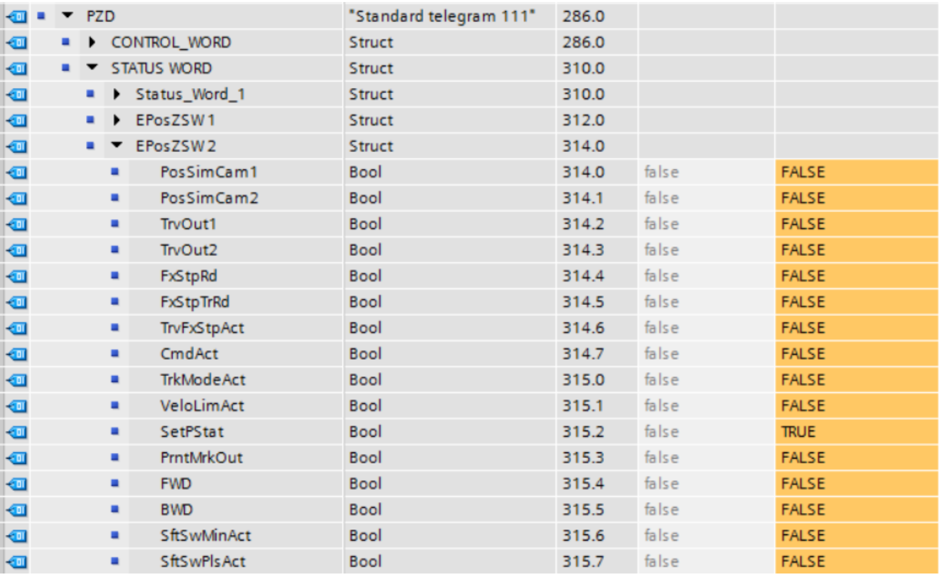
| | | | | |
|---|---|---|---|---|
| PZD | "Standard telegram 111" | 286.0 | | |
| CONTROL_WORD | Struct | 286.0 | | |
| STATUS WORD | Struct | 310.0 | | |
| Status_Word_1 | Struct | 310.0 | | |
| NoFlwErr | Bool | 310.0 | false | TRUE |
| LbCr | Bool | 310.1 | false | TRUE |
| TargPos | Bool | 310.2 | false | FALSE |
| RefPset | Bool | 310.3 | false | TRUE |
| TrvTskAck | Bool | 310.4 | false | TRUE |
| StndStill | Bool | 310.5 | false | FALSE |
| Accel | Bool | 310.6 | false | FALSE |
| Decel | Bool | 310.7 | false | FALSE |
| RTS | Bool | 311.0 | false | TRUE |
| RDY | Bool | 311.1 | false | TRUE |
| IOp | Bool | 311.2 | false | TRUE |
| Fault | Bool | 311.3 | false | FALSE |
| NoOff2Act | Bool | 311.4 | false | TRUE |
| NoOff3Act | Bool | 311.5 | false | TRUE |
| PowInhbt | Bool | 311.6 | false | FALSE |
| Alarm | Bool | 311.7 | false | FALSE |

(Fig. 3.61 B The values of status words of drive)



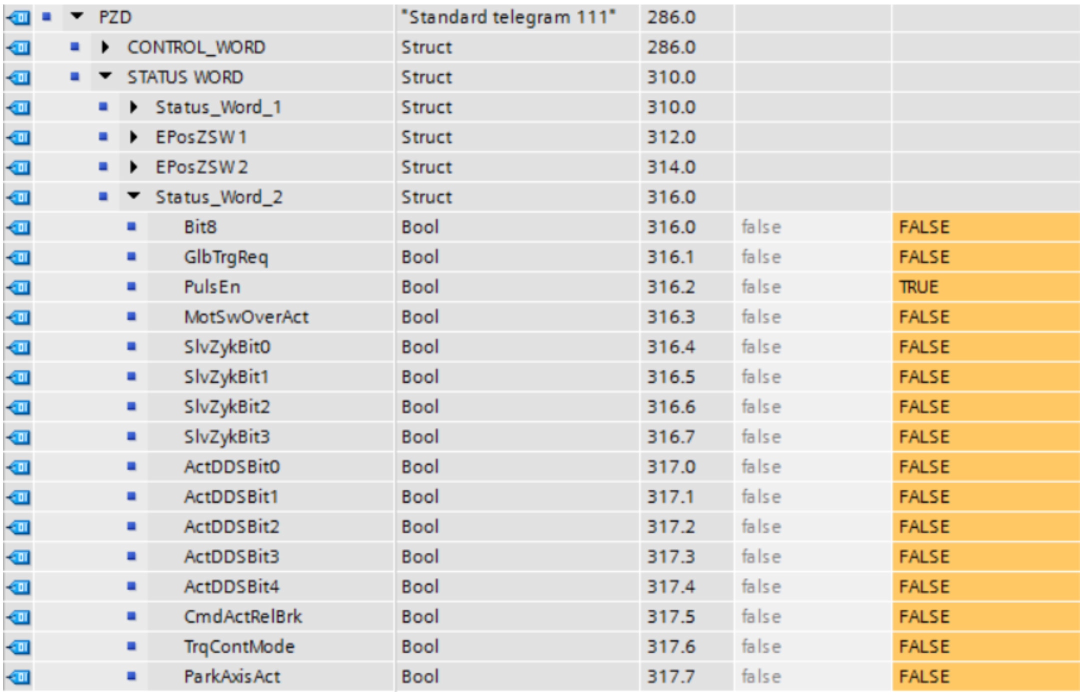
| | | | | |
|---|---|---|---|---|
| PZD | "Standard telegram 111" | 286.0 | | |
| CONTROL_WORD | Struct | 286.0 | | |
| STATUS WORD | Struct | 310.0 | | |
| Status_Word_1 | Struct | 310.0 | | |
| EPosZSW1 | Struct | 312.0 | | |
| ActTrvBit0 | Bool | 312.0 | false | FALSE |
| ActTrvBit1 | Bool | 312.1 | false | FALSE |
| ActTrvBit2 | Bool | 312.2 | false | FALSE |
| ActTrvBit3 | Bool | 312.3 | false | FALSE |
| ActTrvBit4 | Bool | 312.4 | false | FALSE |
| ActTrvBit5 | Bool | 312.5 | false | FALSE |
| Bit6 | Bool | 312.6 | false | FALSE |
| Bit7 | Bool | 312.7 | false | TRUE |
| StpCamMinAct | Bool | 313.0 | false | FALSE |
| StpCamPlsAct | Bool | 313.1 | false | FALSE |
| JogAct | Bool | 313.2 | false | FALSE |
| RefAct | Bool | 313.3 | false | FALSE |
| FlyRefAct | Bool | 313.4 | false | FALSE |
| TrvBlact | Bool | 313.5 | false | FALSE |
| MdiStupAct | Bool | 313.6 | false | FALSE |
| MdiPosAct | Bool | 313.7 | false | FALSE |

(Fig. 3.61 C The values of status words of drive)



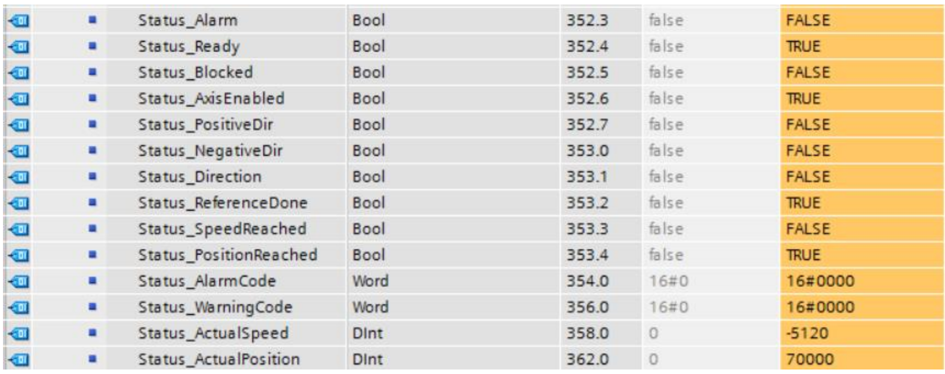
| | | | | |
|---|---|---|---|---|
| PZD | "Standard telegram 111" | 286.0 | | |
| CONTROL_WORD | Struct | 286.0 | | |
| STATUS WORD | Struct | 310.0 | | |
| Status_Word_1 | Struct | 310.0 | | |
| EPosZSW1 | Struct | 312.0 | | |
| EPosZSW2 | Struct | 314.0 | | |
| PosSimCam1 | Bool | 314.0 | false | FALSE |
| PosSimCam2 | Bool | 314.1 | false | FALSE |
| TrvOut1 | Bool | 314.2 | false | FALSE |
| TrvOut2 | Bool | 314.3 | false | FALSE |
| FxStpRd | Bool | 314.4 | false | FALSE |
| FxStpTrRd | Bool | 314.5 | false | FALSE |
| TrvFxStpAct | Bool | 314.6 | false | FALSE |
| CmdAct | Bool | 314.7 | false | FALSE |
| TrkModeAct | Bool | 315.0 | false | FALSE |
| VeloLimAct | Bool | 315.1 | false | FALSE |
| SetPStat | Bool | 315.2 | false | TRUE |
| PrntMrkOut | Bool | 315.3 | false | FALSE |
| FWD | Bool | 315.4 | false | FALSE |
| BWD | Bool | 315.5 | false | FALSE |
| SftSwMinAct | Bool | 315.6 | false | FALSE |
| SftSwPlsAct | Bool | 315.7 | false | FALSE |

(Fig. 3.61 D The values of status words of drive)



| | | | | |
|---|---|---|---|---|
| PZD | "Standard telegram 111" | 286.0 | | |
| CONTROL_WORD | Struct | 286.0 | | |
| STATUS WORD | Struct | 310.0 | | |
| Status_Word_1 | Struct | 310.0 | | |
| EPosZSW1 | Struct | 312.0 | | |
| EPosZSW2 | Struct | 314.0 | | |
| Status_Word_2 | Struct | 316.0 | | |
| Bit8 | Bool | 316.0 | false | FALSE |
| GlbTrgReq | Bool | 316.1 | false | FALSE |
| PulsEn | Bool | 316.2 | false | TRUE |
| MotSwOverAct | Bool | 316.3 | false | FALSE |
| SlvZykBit0 | Bool | 316.4 | false | FALSE |
| SlvZykBit1 | Bool | 316.5 | false | FALSE |
| SlvZykBit2 | Bool | 316.6 | false | FALSE |
| SlvZykBit3 | Bool | 316.7 | false | FALSE |
| ActDDSBit0 | Bool | 317.0 | false | FALSE |
| ActDDSBit1 | Bool | 317.1 | false | FALSE |
| ActDDSBit2 | Bool | 317.2 | false | FALSE |
| ActDDSBit3 | Bool | 317.3 | false | FALSE |
| ActDDSBit4 | Bool | 317.4 | false | FALSE |
| CmdActRelBrk | Bool | 317.5 | false | FALSE |
| TrqContMode | Bool | 317.6 | false | FALSE |
| ParkAxisAct | Bool | 317.7 | false | FALSE |

(Fig. 3.61 E The values of status words of drive)



| | | | | |
|---|---|---|---|---|
| Status_Alarm | Bool | 352.3 | false | FALSE |
| Status_Ready | Bool | 352.4 | false | TRUE |
| Status_Blocked | Bool | 352.5 | false | FALSE |
| Status_AxisEnabled | Bool | 352.6 | false | TRUE |
| Status_PositiveDir | Bool | 352.7 | false | FALSE |
| Status_NegativeDir | Bool | 353.0 | false | FALSE |
| Status_Direction | Bool | 353.1 | false | FALSE |
| Status_ReferenceDone | Bool | 353.2 | false | TRUE |
| Status_SpeedReached | Bool | 353.3 | false | FALSE |
| Status_PositionReached | Bool | 353.4 | false | TRUE |
| Status_AlarmCode | Word | 354.0 | 16#0 | 16#0000 |
| Status_WarningCode | Word | 356.0 | 16#0 | 16#0000 |
| Status_ActualSpeed | DInt | 358.0 | 0 | -5120 |
| Status_ActualPosition | DInt | 362.0 | 0 | 70000 |

(Fig. 3.62 The values of drive input signals.)

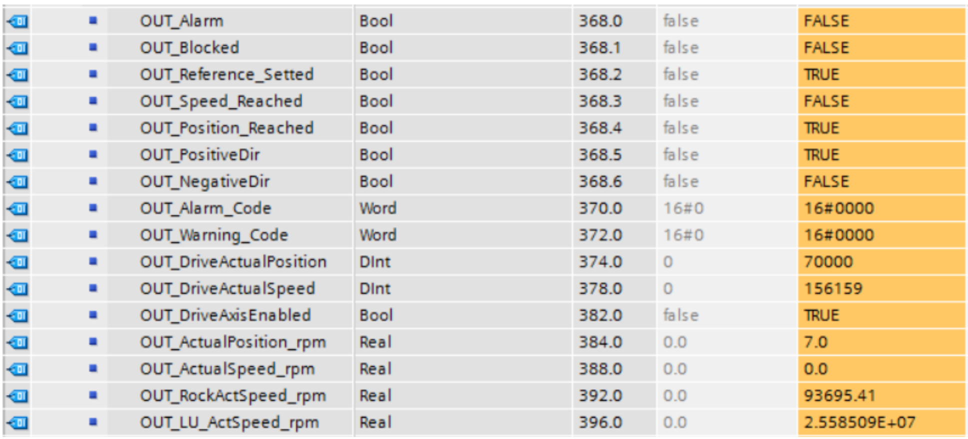
The following bits demonstrate the correct operation of the drive (figure 3.62):

• drive is ready state (Status_Ready=1);

• current position is correct (Status_Position=70000);

• wanted position is reached (Position reached = 1).

The user can read this information by means of user outputs (figure 3.63).



| | | | | |
|---|---|---|---|---|
| OUT_Alarm | Bool | 368.0 | false | FALSE |
| OUT_Blocked | Bool | 368.1 | false | FALSE |
| OUT_Reference_Setted | Bool | 368.2 | false | TRUE |
| OUT_Speed_Reached | Bool | 368.3 | false | FALSE |
| OUT_Position_Reached | Bool | 368.4 | false | TRUE |
| OUT_PositiveDir | Bool | 368.5 | false | TRUE |
| OUT_NegativeDir | Bool | 368.6 | false | FALSE |
| OUT_Alarm_Code | Word | 370.0 | 16#0 | 16#0000 |
| OUT_Warning_Code | Word | 372.0 | 16#0 | 16#0000 |
| OUT_DriveActualPosition | DInt | 374.0 | 0 | 70000 |
| OUT_DriveActualSpeed | DInt | 378.0 | 0 | 156159 |
| OUT_DriveAxisEnabled | Bool | 382.0 | false | TRUE |
| OUT_ActualPosition_rpm | Real | 384.0 | 0.0 | 7.0 |
| OUT_ActualSpeed_rpm | Real | 388.0 | 0.0 | 0.0 |
| OUT_RockActSpeed_rpm | Real | 392.0 | 0.0 | 93695.41 |
| OUT_LU_ActSpeed_rpm | Real | 396.0 | 0.0 | 2.558509E+07 |

(Fig. 3.63 The values of user outputs)

The fifth phase of the test is the execution of the negative absolute position mode. To perform this mode, the user must impose the following parameters:

• “IN_CmdAbsolutePositioning” = 1;

• “IN_StopCycle” = 0;

• “IN_Direction” = 1;

• “IN_PosTargetDest” = 10000;

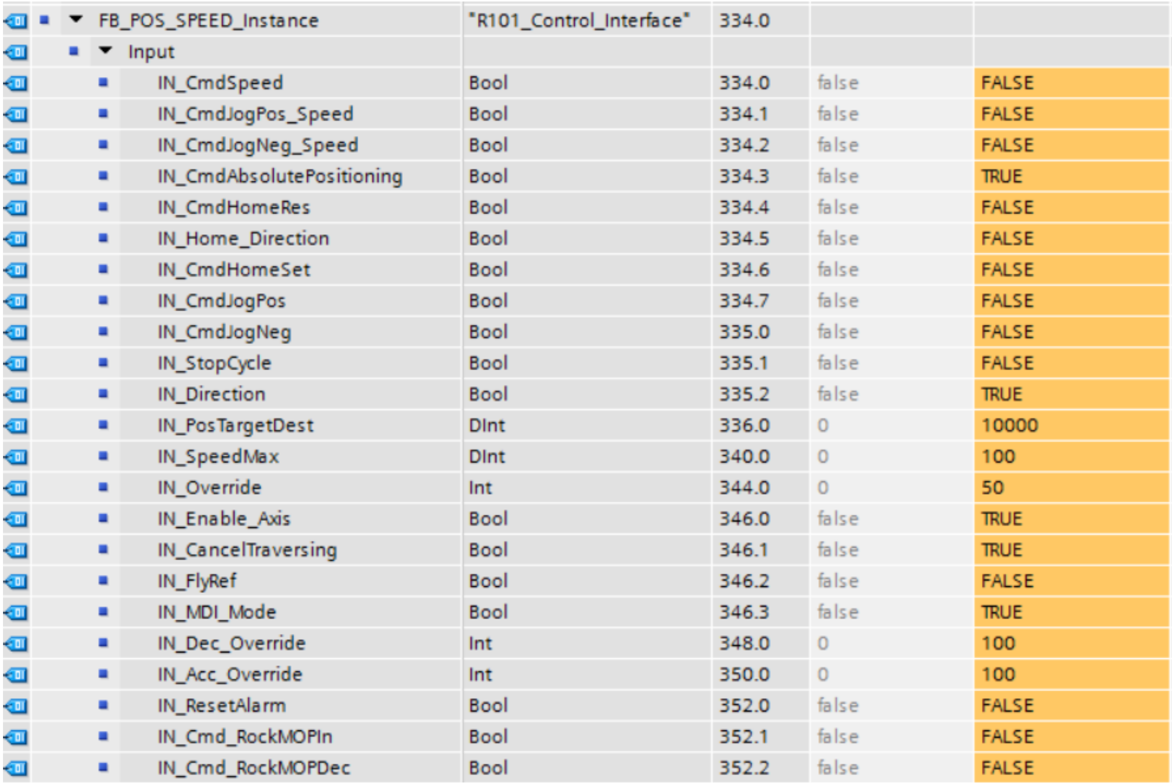
• “IN_SpeedMax” = 100 [%];

• “IN_Override” = 50 [%];

• “IN_Enable_Axis” = 1;

- “IN_CancelTraversing” =1;
- “IN_MDI_Mode” = 1.

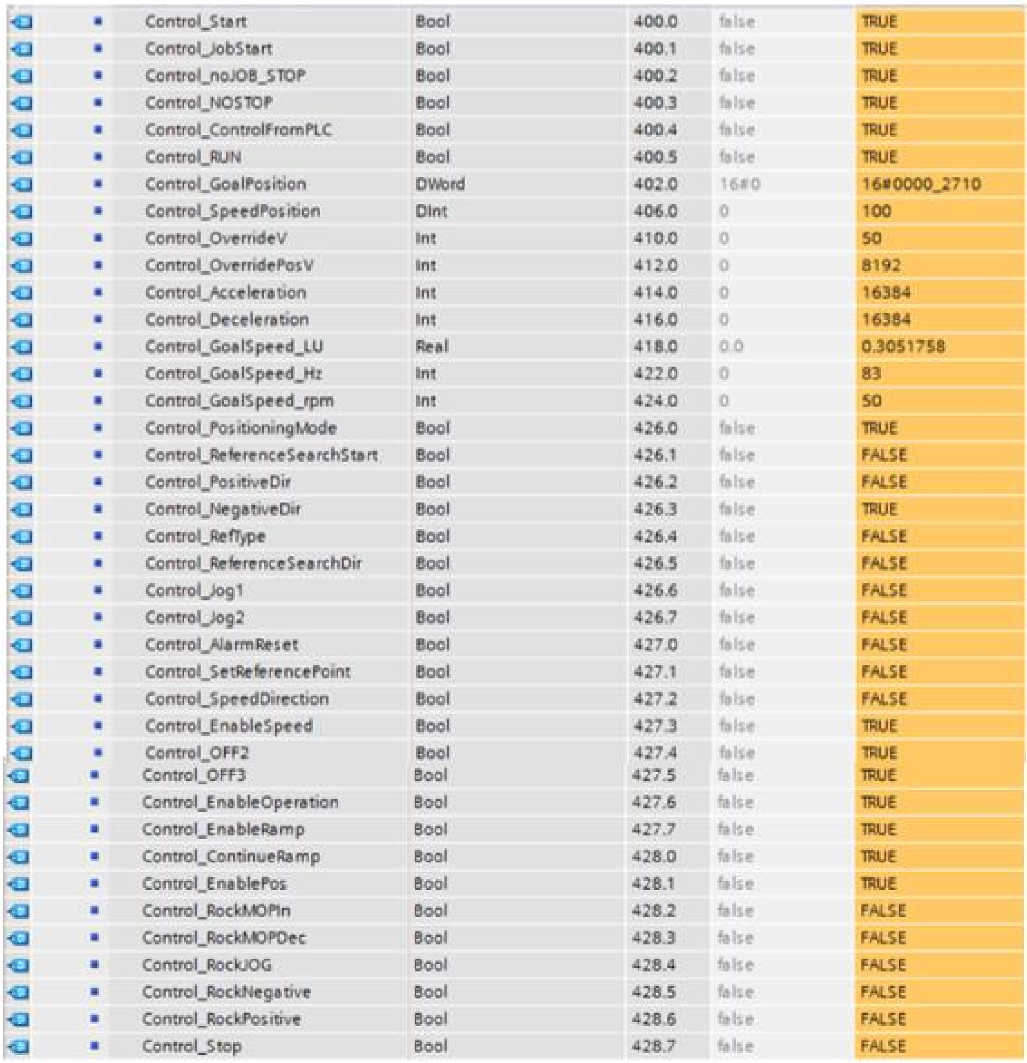
Figure 3.64 shows the correct user inputs that are necessary for the execution of the negative absolute position mode.



| | | | | |
|---|---|---|---|---|
| FB_POS_SPEED_Instance | "R101_Control_Interface" | 334.0 | | |
| Input | | | | |
| IN_CmdSpeed | Bool | 334.0 | false | FALSE |
| IN_CmdJogPos_Speed | Bool | 334.1 | false | FALSE |
| IN_CmdJogNeg_Speed | Bool | 334.2 | false | FALSE |
| IN_CmdAbsolutePositioning | Bool | 334.3 | false | TRUE |
| IN_CmdHomeRes | Bool | 334.4 | false | FALSE |
| IN_Home_Direction | Bool | 334.5 | false | FALSE |
| IN_CmdHomeSet | Bool | 334.6 | false | FALSE |
| IN_CmdJogPos | Bool | 334.7 | false | FALSE |
| IN_CmdJogNeg | Bool | 335.0 | false | FALSE |
| IN_StopCycle | Bool | 335.1 | false | FALSE |
| IN_Direction | Bool | 335.2 | false | TRUE |
| IN_PosTargetDest | DInt | 336.0 | 0 | 10000 |
| IN_SpeedMax | DInt | 340.0 | 0 | 100 |
| IN_Override | Int | 344.0 | 0 | 50 |
| IN_Enable_Axis | Bool | 346.0 | false | TRUE |
| IN_CancelTraversing | Bool | 346.1 | false | TRUE |
| IN_FlyRef | Bool | 346.2 | false | FALSE |
| IN_MDI_Mode | Bool | 346.3 | false | TRUE |
| IN_Dec_Override | Int | 348.0 | 0 | 100 |
| IN_Acc_Override | Int | 350.0 | 0 | 100 |
| IN_ResetAlarm | Bool | 352.0 | false | FALSE |
| IN_Cmd_RockMOPIn | Bool | 352.1 | false | FALSE |
| IN_Cmd_RockMOPDec | Bool | 352.2 | false | FALSE |

(Fig. 3.64 User inputs)

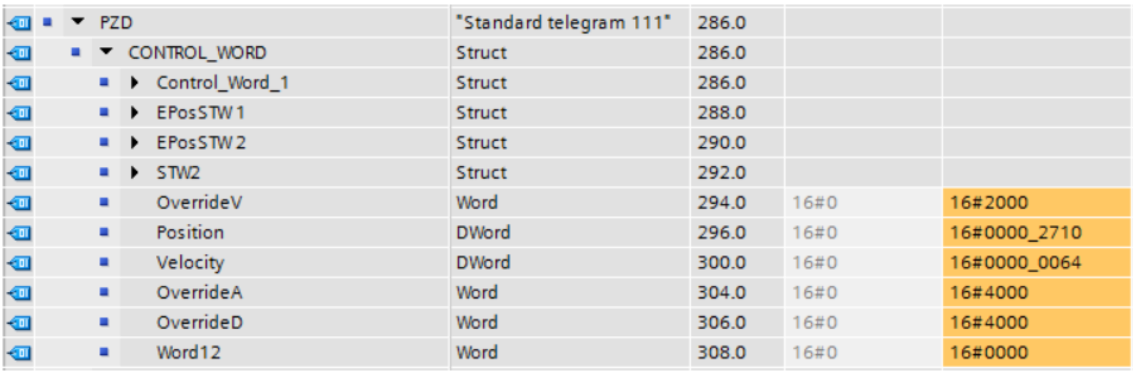
Subsequently, the "R101_Control_Interface" block calculates the commands and sends them to the servo drive (figure 3.65).



| | | | | |
|---|---|---|---|---|
| Control_Start | Bool | 400.0 | false | TRUE |
| Control_JobStart | Bool | 400.1 | false | TRUE |
| Control_noJOB_STOP | Bool | 400.2 | false | TRUE |
| Control_NOSTOP | Bool | 400.3 | false | TRUE |
| Control_ControlFromPLC | Bool | 400.4 | false | TRUE |
| Control_RUN | Bool | 400.5 | false | TRUE |
| Control_GoalPosition | DWord | 402.0 | 16#0 | 16#0000_2710 |
| Control_SpeedPosition | DInt | 406.0 | 0 | 100 |
| Control_OverrideV | Int | 410.0 | 0 | 50 |
| Control_OverridePosV | Int | 412.0 | 0 | 8192 |
| Control_Acceleration | Int | 414.0 | 0 | 16384 |
| Control_Deceleration | Int | 416.0 | 0 | 16384 |
| Control_GoalSpeed_LU | Real | 418.0 | 0.0 | 0.3051758 |
| Control_GoalSpeed_Hz | Int | 422.0 | 0 | 83 |
| Control_GoalSpeed_rpm | Int | 424.0 | 0 | 50 |
| Control_PositioningMode | Bool | 426.0 | false | TRUE |
| Control_ReferenceSearchStart | Bool | 426.1 | false | FALSE |
| Control_PositiveDir | Bool | 426.2 | false | FALSE |
| Control_NegativeDir | Bool | 426.3 | false | TRUE |
| Control_RefType | Bool | 426.4 | false | FALSE |
| Control_ReferenceSearchDir | Bool | 426.5 | false | FALSE |
| Control_Jog1 | Bool | 426.6 | false | FALSE |
| Control_Jog2 | Bool | 426.7 | false | FALSE |
| Control_AlarmReset | Bool | 427.0 | false | FALSE |
| Control_SetReferencePoint | Bool | 427.1 | false | FALSE |
| Control_SpeedDirection | Bool | 427.2 | false | FALSE |
| Control_EnableSpeed | Bool | 427.3 | false | TRUE |
| Control_OFF2 | Bool | 427.4 | false | TRUE |
| Control_OFF3 | Bool | 427.5 | false | TRUE |
| Control_EnableOperation | Bool | 427.6 | false | TRUE |
| Control_EnableRamp | Bool | 427.7 | false | TRUE |
| Control_ContinueRamp | Bool | 428.0 | false | TRUE |
| Control_EnablePos | Bool | 428.1 | false | TRUE |
| Control_RockMOPIn | Bool | 428.2 | false | FALSE |
| Control_RockMOPDec | Bool | 428.3 | false | FALSE |
| Control_RockJOG | Bool | 428.4 | false | FALSE |
| Control_RockNegative | Bool | 428.5 | false | FALSE |
| Control_RockPositive | Bool | 428.6 | false | FALSE |
| Control_Stop | Bool | 428.7 | false | FALSE |

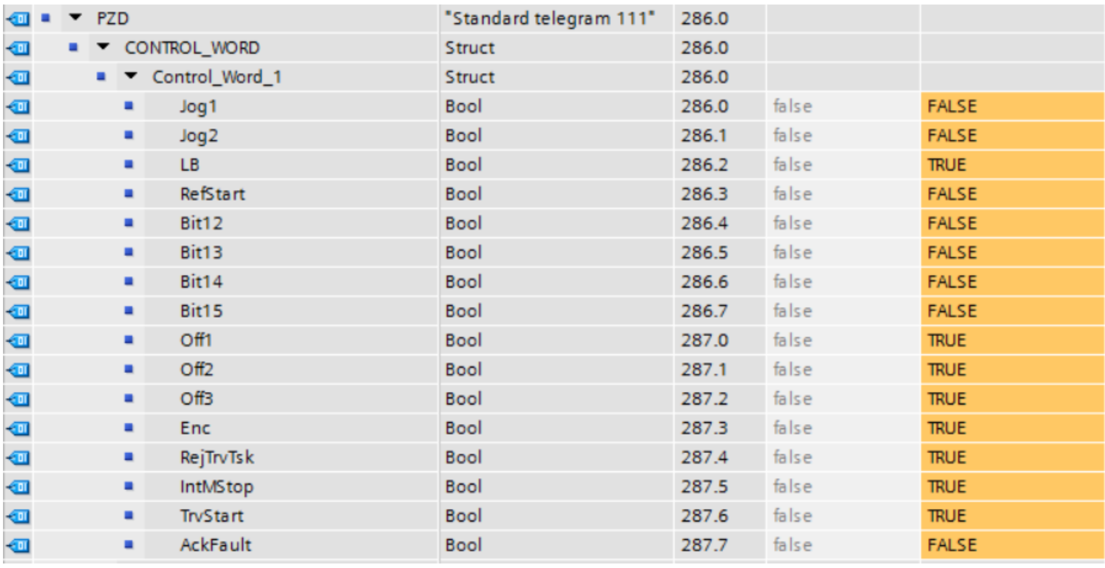
(Fig. 3.65 The values of drive outputs)

Each bit is connected to internal structure (figure 3.66).



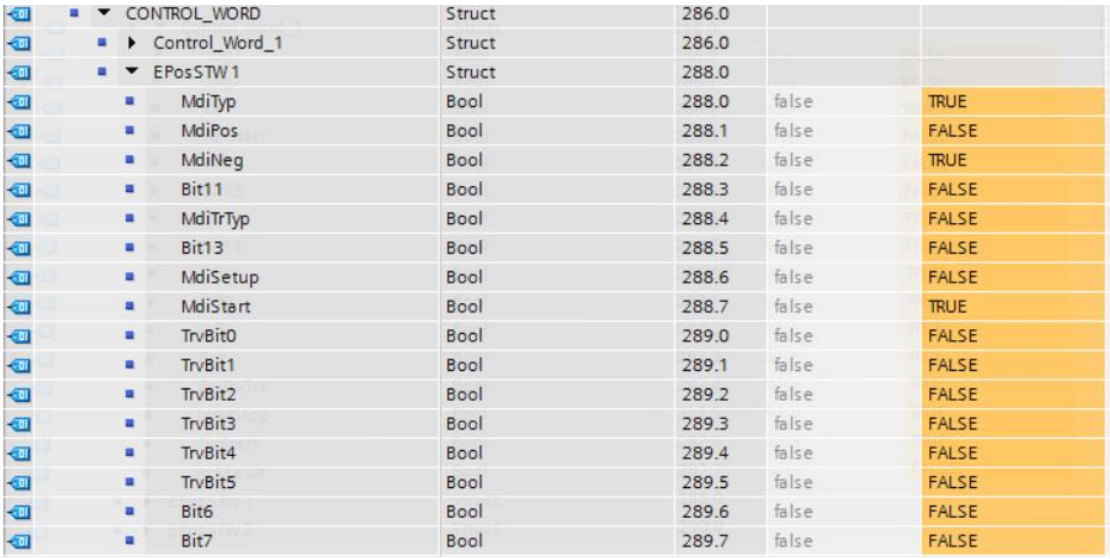
| | | | | |
|---|---|---|---|---|
| ▼ PZD | "Standard telegram 111" | 286.0 | | |
| ▼ CONTROL_WORD | Struct | 286.0 | | |
| ▶ Control_Word_1 | Struct | 286.0 | | |
| ▶ EPosSTW1 | Struct | 288.0 | | |
| ▶ EPosSTW2 | Struct | 290.0 | | |
| ▶ STW2 | Struct | 292.0 | | |
| OverrideV | Word | 294.0 | 16#0 | 16#2000 |
| Position | DWord | 296.0 | 16#0 | 16#0000_2710 |
| Velocity | DWord | 300.0 | 16#0 | 16#0000_0064 |
| OverrideA | Word | 304.0 | 16#0 | 16#4000 |
| OverrideD | Word | 306.0 | 16#0 | 16#4000 |
| Word12 | Word | 308.0 | 16#0 | 16#0000 |

(Fig. 3.66 A The values of command words of drive)



| | | | | |
|---|---|---|---|---|
| PZD | "Standard telegram 111" | 286.0 | | |
| CONTROL_WORD | Struct | 286.0 | | |
| Control_Word_1 | Struct | 286.0 | | |
| Jog1 | Bool | 286.0 | false | FALSE |
| Jog2 | Bool | 286.1 | false | FALSE |
| LB | Bool | 286.2 | false | TRUE |
| RefStart | Bool | 286.3 | false | FALSE |
| Bit12 | Bool | 286.4 | false | FALSE |
| Bit13 | Bool | 286.5 | false | FALSE |
| Bit14 | Bool | 286.6 | false | FALSE |
| Bit15 | Bool | 286.7 | false | FALSE |
| Off1 | Bool | 287.0 | false | TRUE |
| Off2 | Bool | 287.1 | false | TRUE |
| Off3 | Bool | 287.2 | false | TRUE |
| Enc | Bool | 287.3 | false | TRUE |
| RejTrvTsk | Bool | 287.4 | false | TRUE |
| IntMStop | Bool | 287.5 | false | TRUE |
| TrvStart | Bool | 287.6 | false | TRUE |
| AckFault | Bool | 287.7 | false | FALSE |

(Fig. 3.66 B The values of command words of drive)



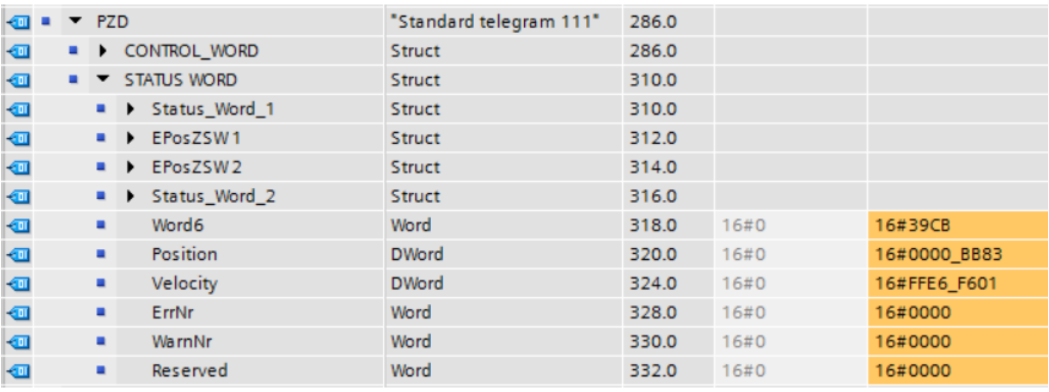
| | | | | |
|---|---|---|---|---|
| CONTROL_WORD | Struct | 286.0 | | |
| Control_Word_1 | Struct | 286.0 | | |
| EPosSTW1 | Struct | 288.0 | | |
| MdiTyp | Bool | 288.0 | false | TRUE |
| MdiPos | Bool | 288.1 | false | FALSE |
| MdiNeg | Bool | 288.2 | false | TRUE |
| Bit11 | Bool | 288.3 | false | FALSE |
| MdiTrTyp | Bool | 288.4 | false | FALSE |
| Bit13 | Bool | 288.5 | false | FALSE |
| MdiSetup | Bool | 288.6 | false | FALSE |
| MdiStart | Bool | 288.7 | false | TRUE |
| TrvBit0 | Bool | 289.0 | false | FALSE |
| TrvBit1 | Bool | 289.1 | false | FALSE |
| TrvBit2 | Bool | 289.2 | false | FALSE |
| TrvBit3 | Bool | 289.3 | false | FALSE |
| TrvBit4 | Bool | 289.4 | false | FALSE |
| TrvBit5 | Bool | 289.5 | false | FALSE |
| Bit6 | Bool | 289.6 | false | FALSE |
| Bit7 | Bool | 289.7 | false | FALSE |

(Fig. 3.66 C The values of command words of drive)

Drive output values are copied to the command drive structure to allow desired position research.

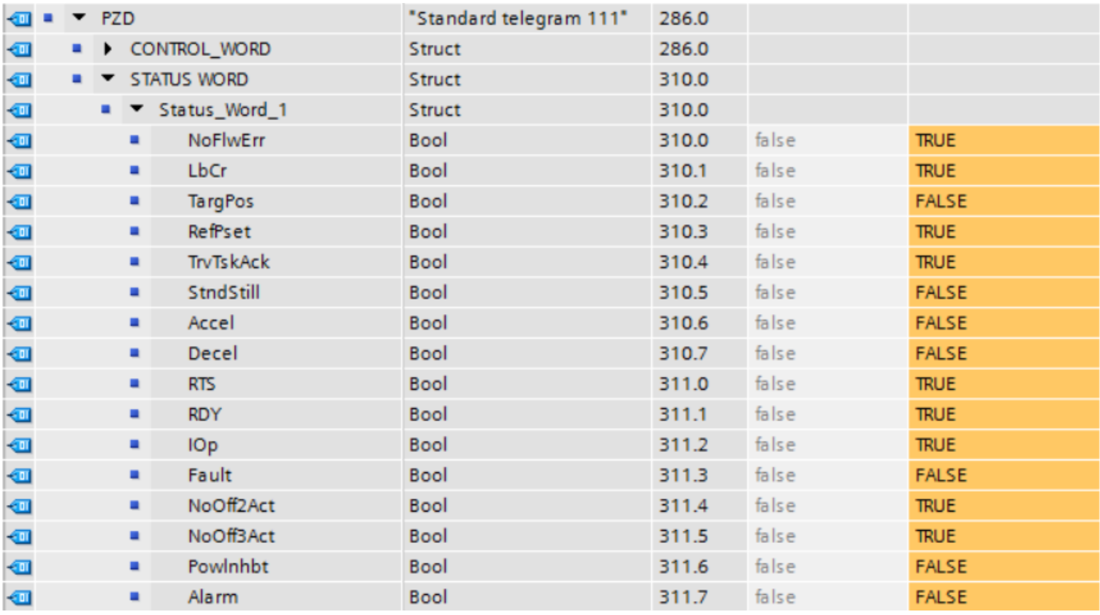
At this point the engine starts spinning in a negative direction until it reaches the desired position.

All drive states are copied to the internal structure (figure 3.67) and the most important bits are connected to the drive input signals (figure 3.68).



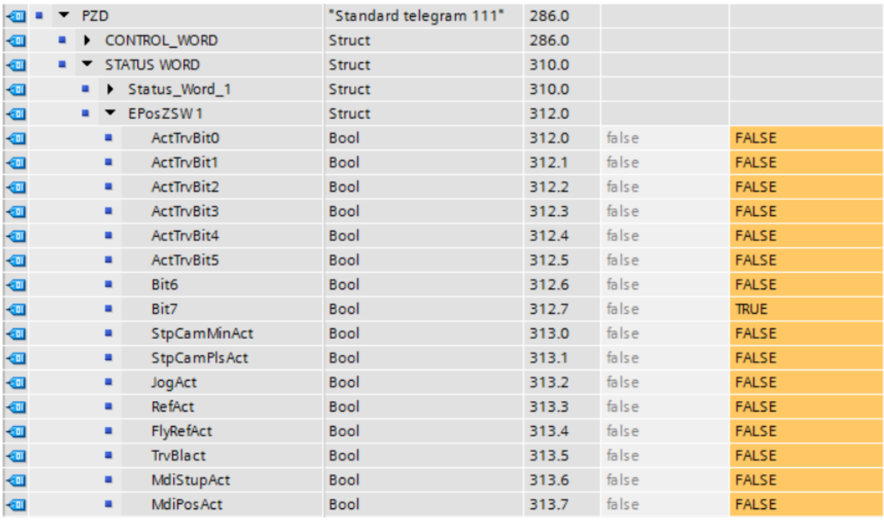
| | | | | |
|---|---|---|---|---|
| ▼ PZD | "Standard telegram 111" | 286.0 | | |
| ▶ CONTROL_WORD | Struct | 286.0 | | |
| ▼ STATUS WORD | Struct | 310.0 | | |
| ▶ Status_Word_1 | Struct | 310.0 | | |
| ▶ EPosZSW1 | Struct | 312.0 | | |
| ▶ EPosZSW2 | Struct | 314.0 | | |
| ▶ Status_Word_2 | Struct | 316.0 | | |
| Word6 | Word | 318.0 | 16#0 | 16#39CB |
| Position | DWord | 320.0 | 16#0 | 16#0000_BB83 |
| Velocity | DWord | 324.0 | 16#0 | 16#FFE6_F601 |
| ErrNr | Word | 328.0 | 16#0 | 16#0000 |
| WarnNr | Word | 330.0 | 16#0 | 16#0000 |
| Reserved | Word | 332.0 | 16#0 | 16#0000 |

(Fig. 3.67 A The values of status words of drive)



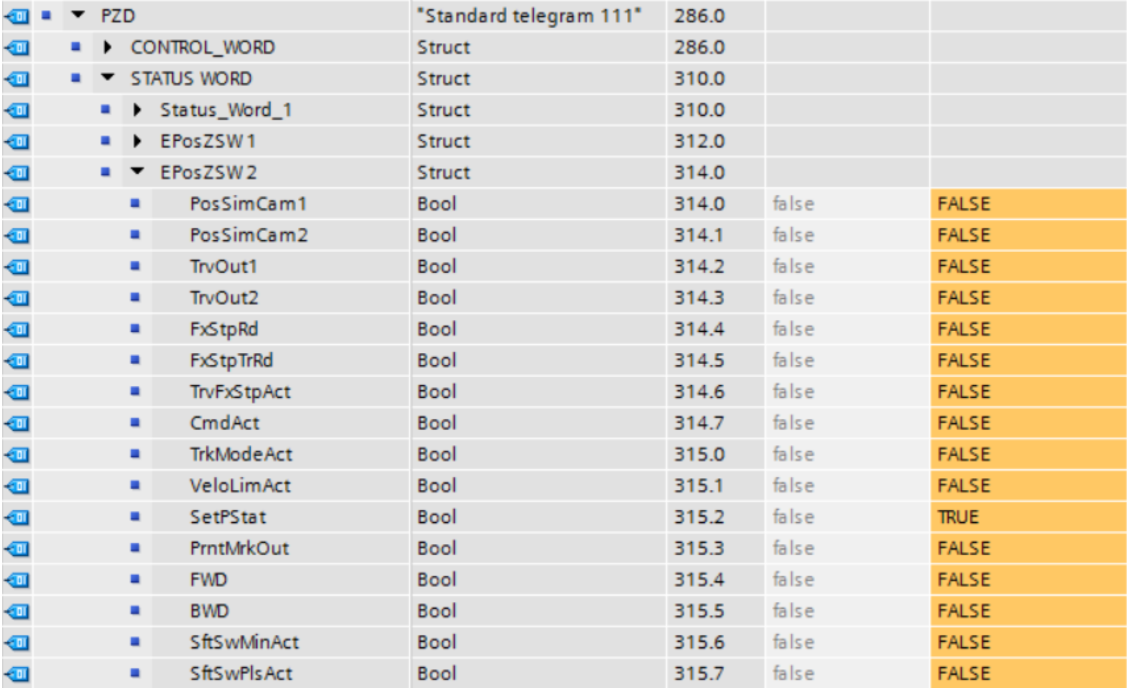
| | | | | |
|---|---|---|---|---|
| ▼ PZD | "Standard telegram 111" | 286.0 | | |
| ▶ CONTROL_WORD | Struct | 286.0 | | |
| ▼ STATUS WORD | Struct | 310.0 | | |
| ▼ Status_Word_1 | Struct | 310.0 | | |
| NoFlwErr | Bool | 310.0 | false | TRUE |
| LbCr | Bool | 310.1 | false | TRUE |
| TargPos | Bool | 310.2 | false | FALSE |
| RefPset | Bool | 310.3 | false | TRUE |
| TrvTskAck | Bool | 310.4 | false | TRUE |
| StndStill | Bool | 310.5 | false | FALSE |
| Accel | Bool | 310.6 | false | FALSE |
| Decel | Bool | 310.7 | false | FALSE |
| RTS | Bool | 311.0 | false | TRUE |
| RDY | Bool | 311.1 | false | TRUE |
| IOp | Bool | 311.2 | false | TRUE |
| Fault | Bool | 311.3 | false | FALSE |
| NoOff2Act | Bool | 311.4 | false | TRUE |
| NoOff3Act | Bool | 311.5 | false | TRUE |
| PowInhbt | Bool | 311.6 | false | FALSE |
| Alarm | Bool | 311.7 | false | FALSE |

(Fig. 3.67 B The values of status words of drive)



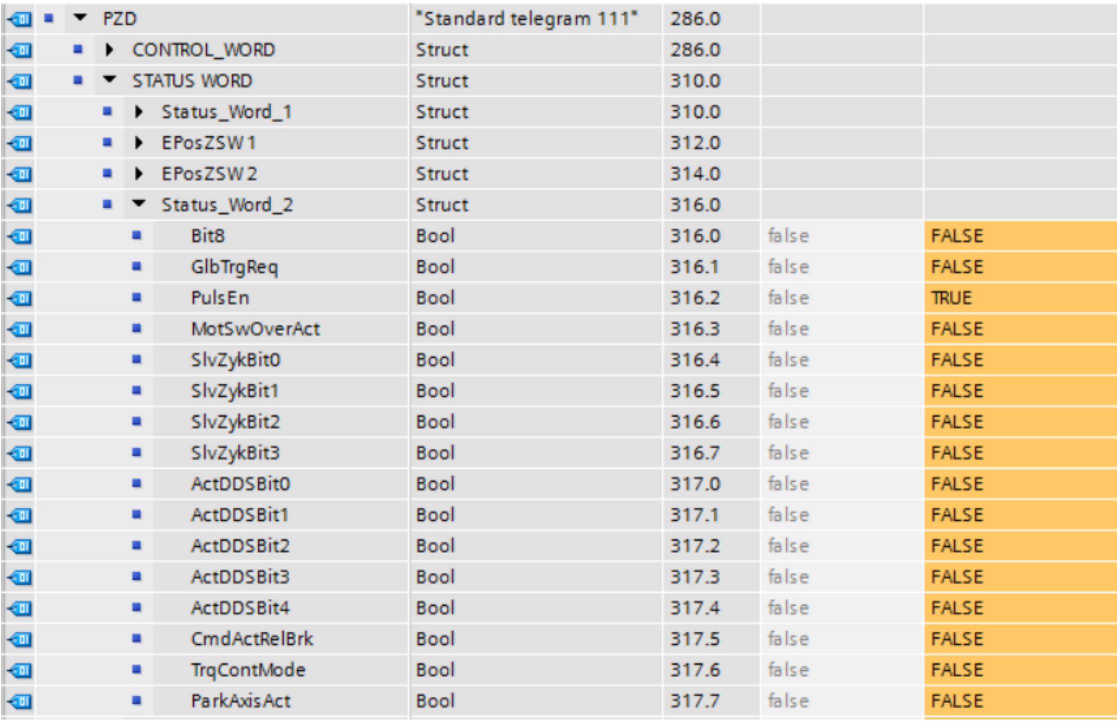
| | | | | |
|---|---|---|---|---|
| ▼ PZD | "Standard telegram 111" | 286.0 | | |
| ▶ CONTROL_WORD | Struct | 286.0 | | |
| ▼ STATUS WORD | Struct | 310.0 | | |
| ▶ Status_Word_1 | Struct | 310.0 | | |
| ▼ EPosZSW1 | Struct | 312.0 | | |
| ActTrvBit0 | Bool | 312.0 | false | FALSE |
| ActTrvBit1 | Bool | 312.1 | false | FALSE |
| ActTrvBit2 | Bool | 312.2 | false | FALSE |
| ActTrvBit3 | Bool | 312.3 | false | FALSE |
| ActTrvBit4 | Bool | 312.4 | false | FALSE |
| ActTrvBit5 | Bool | 312.5 | false | FALSE |
| Bit6 | Bool | 312.6 | false | FALSE |
| Bit7 | Bool | 312.7 | false | TRUE |
| StpCamMinAct | Bool | 313.0 | false | FALSE |
| StpCamPlsAct | Bool | 313.1 | false | FALSE |
| JogAct | Bool | 313.2 | false | FALSE |
| RefAct | Bool | 313.3 | false | FALSE |
| FlyRefAct | Bool | 313.4 | false | FALSE |
| TrvBlact | Bool | 313.5 | false | FALSE |
| MdiStupAct | Bool | 313.6 | false | FALSE |
| MdiPosAct | Bool | 313.7 | false | FALSE |

(Fig. 3.67 C The values of status words of drive)



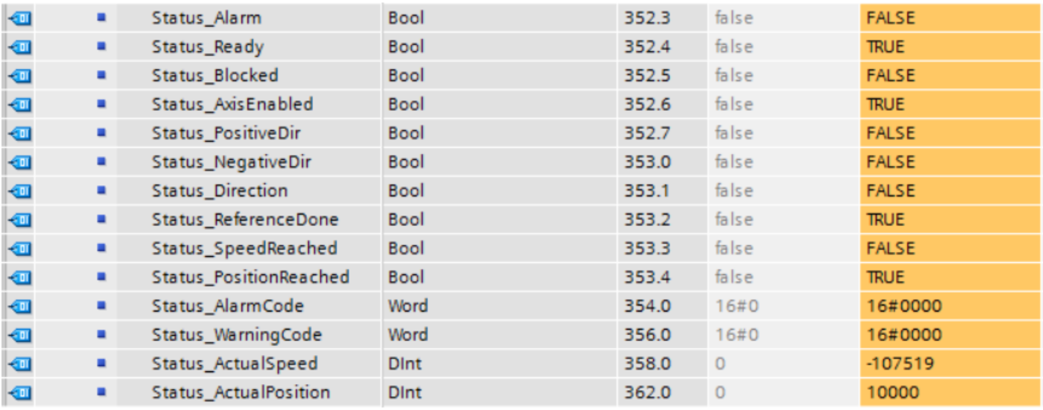
| | | | | |
|---|---|---|---|---|
| PZD | "Standard telegram 111" | 286.0 | | |
| CONTROL_WORD | Struct | 286.0 | | |
| STATUS WORD | Struct | 310.0 | | |
| Status_Word_1 | Struct | 310.0 | | |
| EPosZSW1 | Struct | 312.0 | | |
| EPosZSW2 | Struct | 314.0 | | |
| PosSimCam1 | Bool | 314.0 | false | FALSE |
| PosSimCam2 | Bool | 314.1 | false | FALSE |
| TrvOut1 | Bool | 314.2 | false | FALSE |
| TrvOut2 | Bool | 314.3 | false | FALSE |
| FxStpRd | Bool | 314.4 | false | FALSE |
| FxStpTrRd | Bool | 314.5 | false | FALSE |
| TrvFxStpAct | Bool | 314.6 | false | FALSE |
| CmdAct | Bool | 314.7 | false | FALSE |
| TrkModeAct | Bool | 315.0 | false | FALSE |
| VeloLimAct | Bool | 315.1 | false | FALSE |
| SetPStat | Bool | 315.2 | false | TRUE |
| PrntMrkOut | Bool | 315.3 | false | FALSE |
| FWD | Bool | 315.4 | false | FALSE |
| BWD | Bool | 315.5 | false | FALSE |
| SftSwMinAct | Bool | 315.6 | false | FALSE |
| SftSwPlsAct | Bool | 315.7 | false | FALSE |

(Fig. 3.67 D The values of status words of drive)



| | | | | |
|---|---|---|---|---|
| PZD | "Standard telegram 111" | 286.0 | | |
| CONTROL_WORD | Struct | 286.0 | | |
| STATUS WORD | Struct | 310.0 | | |
| Status_Word_1 | Struct | 310.0 | | |
| EPosZSW1 | Struct | 312.0 | | |
| EPosZSW2 | Struct | 314.0 | | |
| Status_Word_2 | Struct | 316.0 | | |
| Bit8 | Bool | 316.0 | false | FALSE |
| GlbTrgReq | Bool | 316.1 | false | FALSE |
| PulsEn | Bool | 316.2 | false | TRUE |
| MotSwOverAct | Bool | 316.3 | false | FALSE |
| SlvZykBit0 | Bool | 316.4 | false | FALSE |
| SlvZykBit1 | Bool | 316.5 | false | FALSE |
| SlvZykBit2 | Bool | 316.6 | false | FALSE |
| SlvZykBit3 | Bool | 316.7 | false | FALSE |
| ActDDSBit0 | Bool | 317.0 | false | FALSE |
| ActDDSBit1 | Bool | 317.1 | false | FALSE |
| ActDDSBit2 | Bool | 317.2 | false | FALSE |
| ActDDSBit3 | Bool | 317.3 | false | FALSE |
| ActDDSBit4 | Bool | 317.4 | false | FALSE |
| CmdActRelBrk | Bool | 317.5 | false | FALSE |
| TrqContMode | Bool | 317.6 | false | FALSE |
| ParkAxisAct | Bool | 317.7 | false | FALSE |

(Fig. 3.67 E The values of status words of drive)



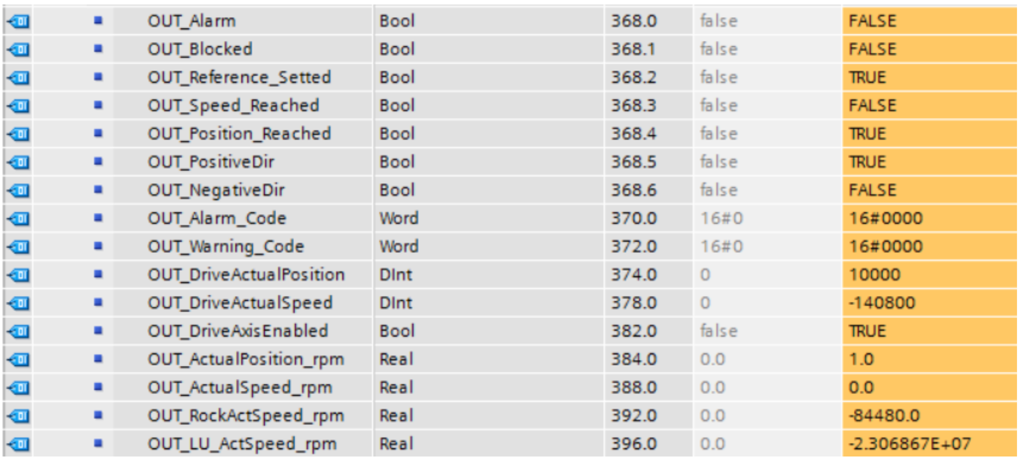
| | | | | |
|---|---|---|---|---|
| Status_Alarm | Bool | 352.3 | false | FALSE |
| Status_Ready | Bool | 352.4 | false | TRUE |
| Status_Blocked | Bool | 352.5 | false | FALSE |
| Status_AxisEnabled | Bool | 352.6 | false | TRUE |
| Status_PositiveDir | Bool | 352.7 | false | FALSE |
| Status_NegativeDir | Bool | 353.0 | false | FALSE |
| Status_Direction | Bool | 353.1 | false | FALSE |
| Status_ReferenceDone | Bool | 353.2 | false | TRUE |
| Status_SpeedReached | Bool | 353.3 | false | FALSE |
| Status_PositionReached | Bool | 353.4 | false | TRUE |
| Status_AlarmCode | Word | 354.0 | 16#0 | 16#0000 |
| Status_WarningCode | Word | 356.0 | 16#0 | 16#0000 |
| Status_ActualSpeed | DInt | 358.0 | 0 | -107519 |
| Status_ActualPosition | DInt | 362.0 | 0 | 10000 |

(Fig. 3.68 The values of drive input signals.)

The following bits demonstrate the correct operation of the drive (figure 3.68):

- drive is ready state (Status_Ready=1);
- current position is correct (Status_Position=10000);
- wanted position is reached (Status_Position reached = 1).

The user can read this information by means of user outputs (figure 3.69).



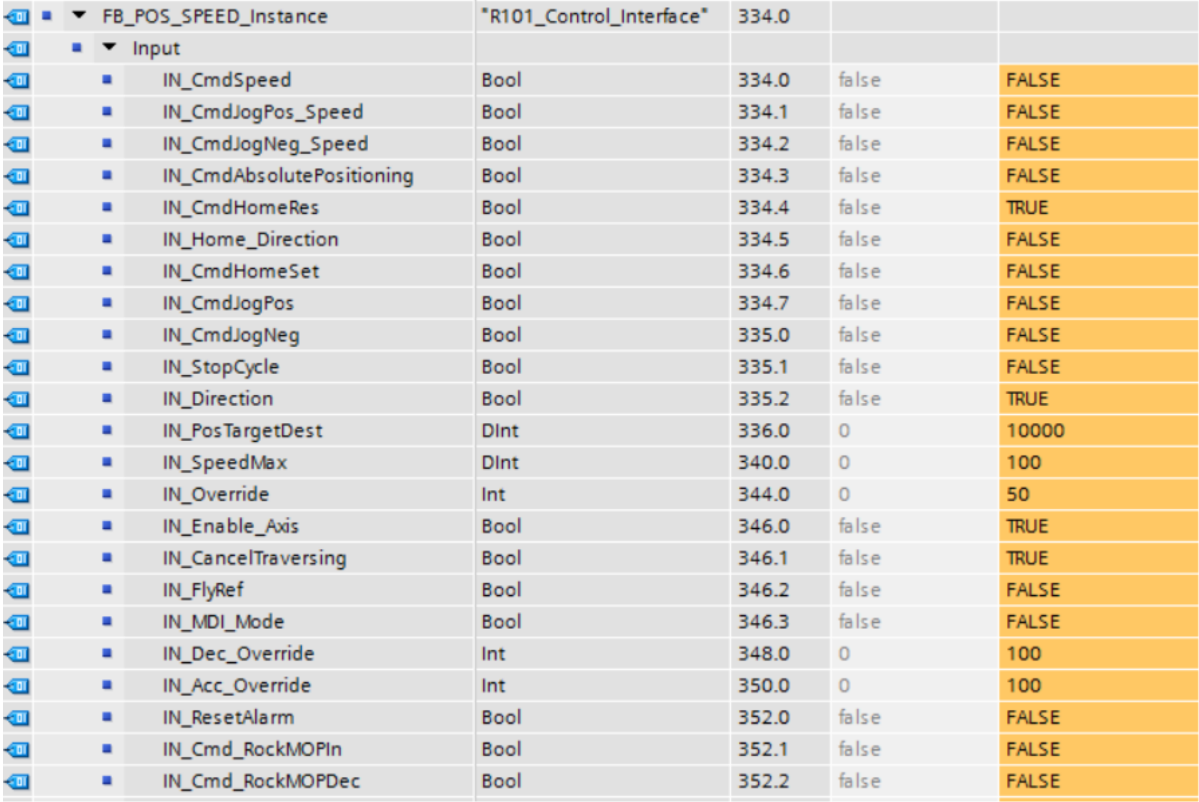
| | | | | |
|---|---|---|---|---|
| OUT_Alarm | Bool | 368.0 | false | FALSE |
| OUT_Blocked | Bool | 368.1 | false | FALSE |
| OUT_Reference_Setted | Bool | 368.2 | false | TRUE |
| OUT_Speed_Reached | Bool | 368.3 | false | FALSE |
| OUT_Position_Reached | Bool | 368.4 | false | TRUE |
| OUT_PositiveDir | Bool | 368.5 | false | TRUE |
| OUT_NegativeDir | Bool | 368.6 | false | FALSE |
| OUT_Alarm_Code | Word | 370.0 | 16#0 | 16#0000 |
| OUT_Warning_Code | Word | 372.0 | 16#0 | 16#0000 |
| OUT_DriveActualPosition | DInt | 374.0 | 0 | 10000 |
| OUT_DriveActualSpeed | DInt | 378.0 | 0 | -140800 |
| OUT_DriveAxisEnabled | Bool | 382.0 | false | TRUE |
| OUT_ActualPosition_rpm | Real | 384.0 | 0.0 | 1.0 |
| OUT_ActualSpeed_rpm | Real | 388.0 | 0.0 | 0.0 |
| OUT_RockActSpeed_rpm | Real | 392.0 | 0.0 | -84480.0 |
| OUT_LU_ActSpeed_rpm | Real | 396.0 | 0.0 | -2.306867E+07 |

(Fig. 3.69 The values of user outputs)

In the sixth phase of the test the execution of the positive reference search mode is studied. This mode is executed when the user imposes the following parameters:

- “IN_CmdHomeRes” = 1;
- “IN_Home_Direction” = 0;
- “IN_StopCycle” = 0;
- “IN_CancelTraversing” =1;
- “IN_Enable_Axis” = 1;
- “IN_MDI_Mode” = 1.

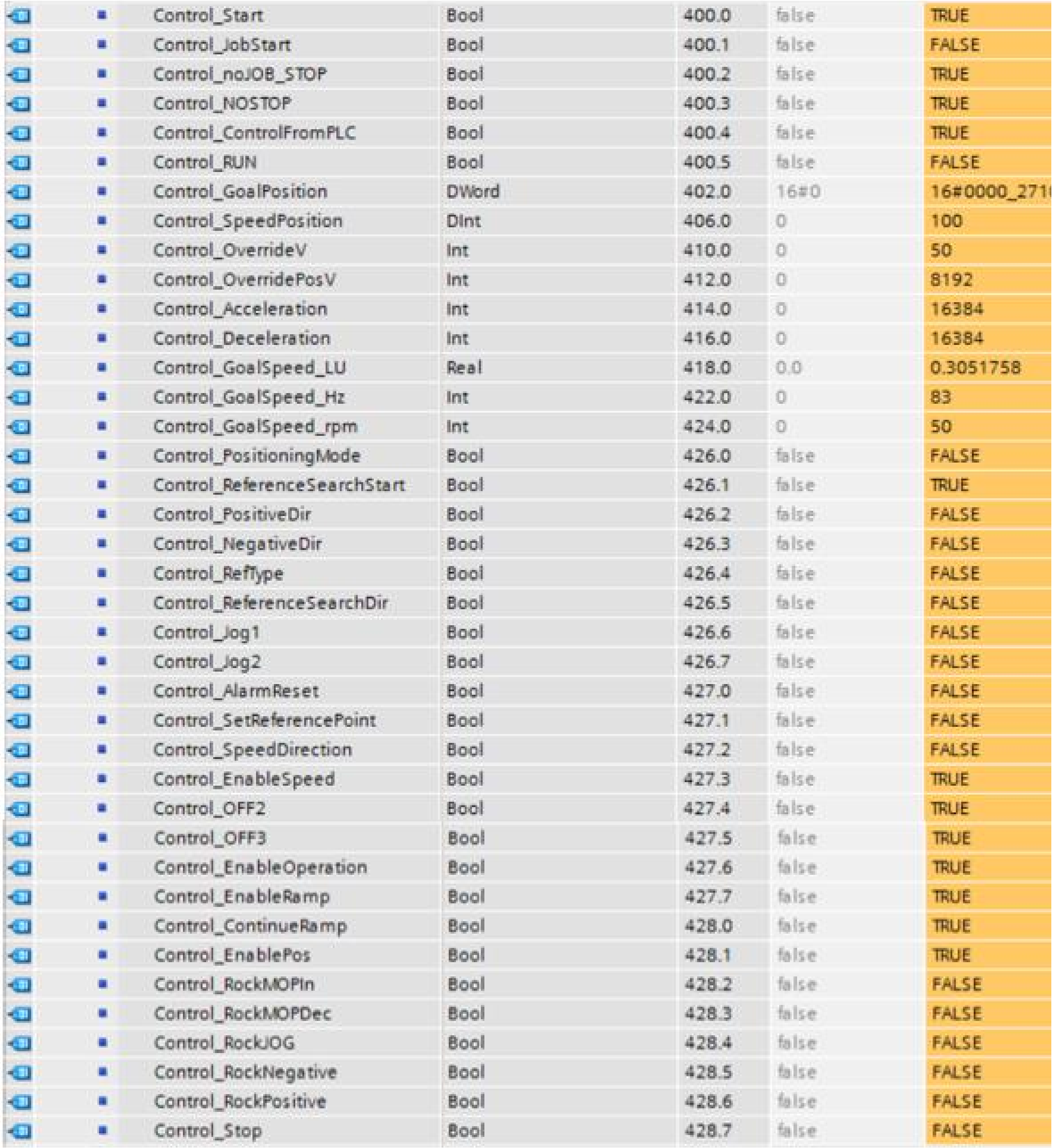
The right user inputs values are represented in figure 3.70.



| | | | | |
|---|---|---|---|---|
| ▼ FB_POS_SPEED_Instance | "R101_Control_Interface" | 334.0 | | |
| ▼ Input | | | | |
| IN_CmdSpeed | Bool | 334.0 | false | FALSE |
| IN_CmdJogPos_Speed | Bool | 334.1 | false | FALSE |
| IN_CmdJogNeg_Speed | Bool | 334.2 | false | FALSE |
| IN_CmdAbsolutePositioning | Bool | 334.3 | false | FALSE |
| IN_CmdHomeRes | Bool | 334.4 | false | TRUE |
| IN_Home_Direction | Bool | 334.5 | false | FALSE |
| IN_CmdHomeSet | Bool | 334.6 | false | FALSE |
| IN_CmdJogPos | Bool | 334.7 | false | FALSE |
| IN_CmdJogNeg | Bool | 335.0 | false | FALSE |
| IN_StopCycle | Bool | 335.1 | false | FALSE |
| IN_Direction | Bool | 335.2 | false | TRUE |
| IN_PosTargetDest | DInt | 336.0 | 0 | 10000 |
| IN_SpeedMax | DInt | 340.0 | 0 | 100 |
| IN_Override | Int | 344.0 | 0 | 50 |
| IN_Enable_Axis | Bool | 346.0 | false | TRUE |
| IN_CancelTraversing | Bool | 346.1 | false | TRUE |
| IN_FlyRef | Bool | 346.2 | false | FALSE |
| IN_MDI_Mode | Bool | 346.3 | false | FALSE |
| IN_Dec_Override | Int | 348.0 | 0 | 100 |
| IN_Acc_Override | Int | 350.0 | 0 | 100 |
| IN_ResetAlarm | Bool | 352.0 | false | FALSE |
| IN_Cmd_RockMOPIn | Bool | 352.1 | false | FALSE |
| IN_Cmd_RockMOPDec | Bool | 352.2 | false | FALSE |

(Fig. 3.70 User inputs)

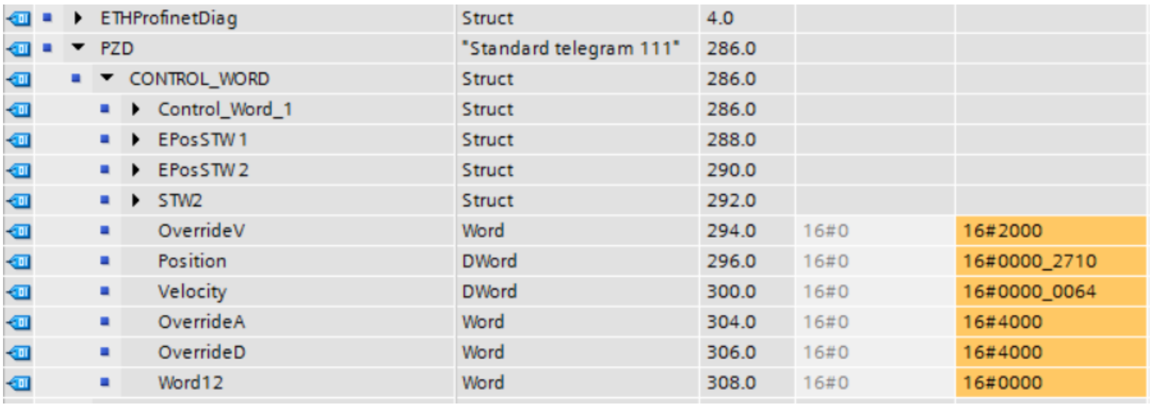
The interface block processes all the inputs and generates the servo drive commands (figure 3.71).



| | | | | |
|---|---|---|---|---|
| Control_Start | Bool | 400.0 | false | TRUE |
| Control_JobStart | Bool | 400.1 | false | FALSE |
| Control_noJOB_STOP | Bool | 400.2 | false | TRUE |
| Control_NOSTOP | Bool | 400.3 | false | TRUE |
| Control_ControlFromPLC | Bool | 400.4 | false | TRUE |
| Control_RUN | Bool | 400.5 | false | FALSE |
| Control_GoalPosition | DWord | 402.0 | 16#0 | 16#0000_271 |
| Control_SpeedPosition | DInt | 406.0 | 0 | 100 |
| Control_OverrideV | Int | 410.0 | 0 | 50 |
| Control_OverridePosV | Int | 412.0 | 0 | 8192 |
| Control_Acceleration | Int | 414.0 | 0 | 16384 |
| Control_Deceleration | Int | 416.0 | 0 | 16384 |
| Control_GoalSpeed_LU | Real | 418.0 | 0.0 | 0.3051758 |
| Control_GoalSpeed_Hz | Int | 422.0 | 0 | 83 |
| Control_GoalSpeed_rpm | Int | 424.0 | 0 | 50 |
| Control_PositioningMode | Bool | 426.0 | false | FALSE |
| Control_ReferenceSearchStart | Bool | 426.1 | false | TRUE |
| Control_PositiveDir | Bool | 426.2 | false | FALSE |
| Control_NegativeDir | Bool | 426.3 | false | FALSE |
| Control_RefType | Bool | 426.4 | false | FALSE |
| Control_ReferenceSearchDir | Bool | 426.5 | false | FALSE |
| Control_Jog1 | Bool | 426.6 | false | FALSE |
| Control_Jog2 | Bool | 426.7 | false | FALSE |
| Control_AlarmReset | Bool | 427.0 | false | FALSE |
| Control_SetReferencePoint | Bool | 427.1 | false | FALSE |
| Control_SpeedDirection | Bool | 427.2 | false | FALSE |
| Control_EnableSpeed | Bool | 427.3 | false | TRUE |
| Control_OFF2 | Bool | 427.4 | false | TRUE |
| Control_OFF3 | Bool | 427.5 | false | TRUE |
| Control_EnableOperation | Bool | 427.6 | false | TRUE |
| Control_EnableRamp | Bool | 427.7 | false | TRUE |
| Control_ContinueRamp | Bool | 428.0 | false | TRUE |
| Control_EnablePos | Bool | 428.1 | false | TRUE |
| Control_RockMOPIn | Bool | 428.2 | false | FALSE |
| Control_RockMOPDec | Bool | 428.3 | false | FALSE |
| Control_RockJOG | Bool | 428.4 | false | FALSE |
| Control_RockNegative | Bool | 428.5 | false | FALSE |
| Control_RockPositive | Bool | 428.6 | false | FALSE |
| Control_Stop | Bool | 428.7 | false | FALSE |

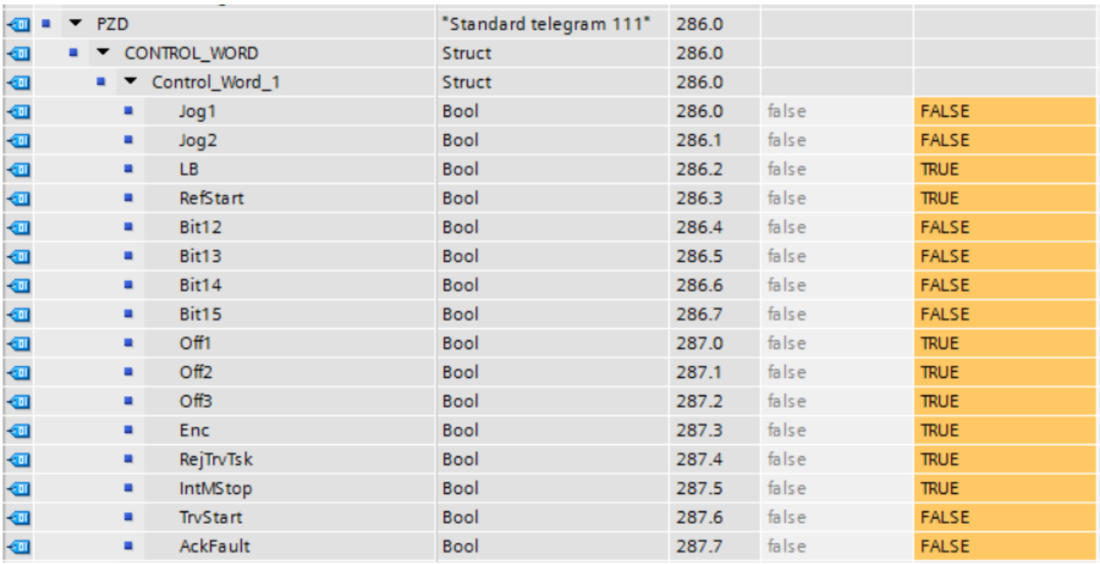
(Fig. 3.71 The values of drive outputs)

Drive outputs are associated with the internal structure (figure 3.72).



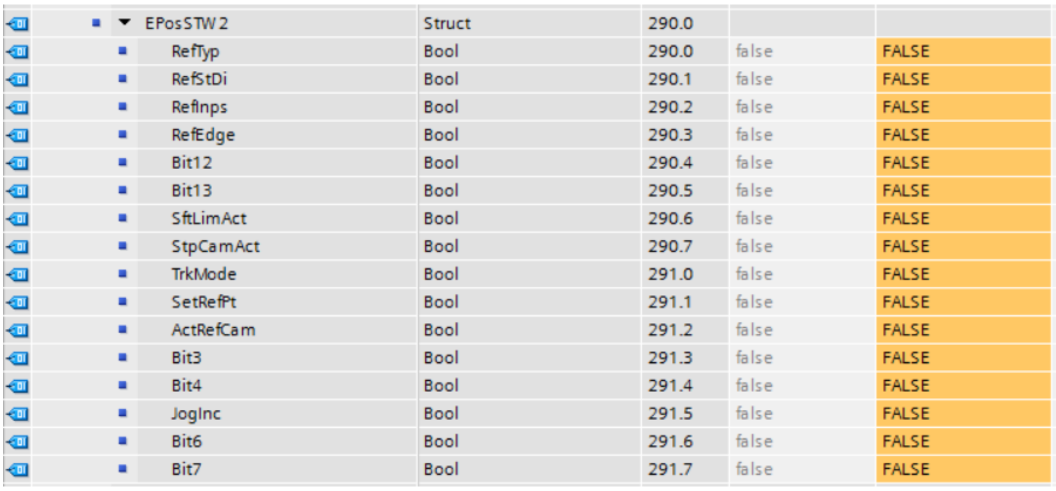
| | | | | |
|---|---|---|---|---|
| ETHProfinetDiag | Struct | 4.0 | | |
| PZD | "Standard telegram 111" | 286.0 | | |
| CONTROL_WORD | Struct | 286.0 | | |
| Control_Word_1 | Struct | 286.0 | | |
| EPosSTW1 | Struct | 288.0 | | |
| EPosSTW2 | Struct | 290.0 | | |
| STW2 | Struct | 292.0 | | |
| OverrideV | Word | 294.0 | 16#0 | 16#2000 |
| Position | DWord | 296.0 | 16#0 | 16#0000_2710 |
| Velocity | DWord | 300.0 | 16#0 | 16#0000_0064 |
| OverrideA | Word | 304.0 | 16#0 | 16#4000 |
| OverrideD | Word | 306.0 | 16#0 | 16#4000 |
| Word12 | Word | 308.0 | 16#0 | 16#0000 |

(Fig. 3.72 A The values of command words of drive)



| | | | | |
|---|---|---|---|---|
| PZD | "Standard telegram 111" | 286.0 | | |
| CONTROL_WORD | Struct | 286.0 | | |
| Control_Word_1 | Struct | 286.0 | | |
| Jog1 | Bool | 286.0 | false | FALSE |
| Jog2 | Bool | 286.1 | false | FALSE |
| LB | Bool | 286.2 | false | TRUE |
| RefStart | Bool | 286.3 | false | TRUE |
| Bit12 | Bool | 286.4 | false | FALSE |
| Bit13 | Bool | 286.5 | false | FALSE |
| Bit14 | Bool | 286.6 | false | FALSE |
| Bit15 | Bool | 286.7 | false | FALSE |
| Off1 | Bool | 287.0 | false | TRUE |
| Off2 | Bool | 287.1 | false | TRUE |
| Off3 | Bool | 287.2 | false | TRUE |
| Enc | Bool | 287.3 | false | TRUE |
| RejTrvTsk | Bool | 287.4 | false | TRUE |
| IntMStop | Bool | 287.5 | false | TRUE |
| TrvStart | Bool | 287.6 | false | FALSE |
| AckFault | Bool | 287.7 | false | FALSE |

(Fig. 3.72 B The values of command words of drive)



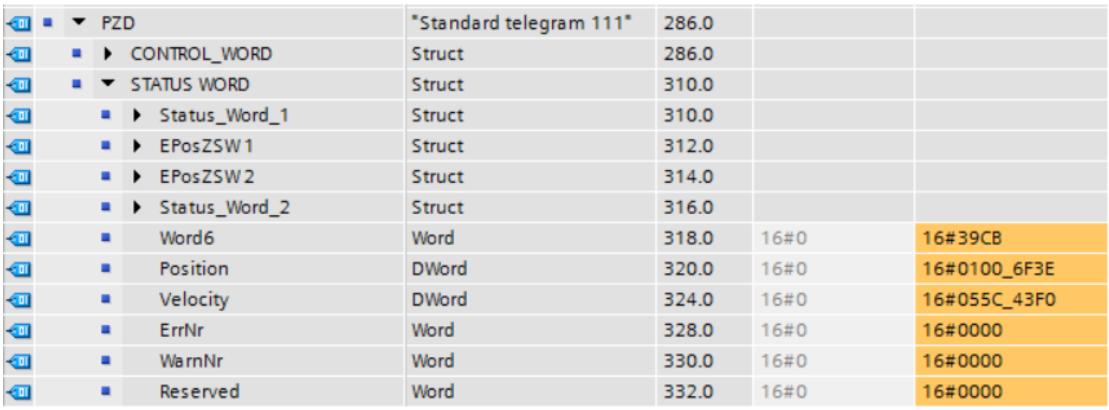
| | | | | |
|---|---|---|---|---|
| EPosSTW2 | Struct | 290.0 | | |
| RefTyp | Bool | 290.0 | false | FALSE |
| RefStDi | Bool | 290.1 | false | FALSE |
| RefInps | Bool | 290.2 | false | FALSE |
| RefEdge | Bool | 290.3 | false | FALSE |
| Bit12 | Bool | 290.4 | false | FALSE |
| Bit13 | Bool | 290.5 | false | FALSE |
| SftLimAct | Bool | 290.6 | false | FALSE |
| StpCamAct | Bool | 290.7 | false | FALSE |
| TrkMode | Bool | 291.0 | false | FALSE |
| SetRefPt | Bool | 291.1 | false | FALSE |
| ActRefCam | Bool | 291.2 | false | FALSE |
| Bit3 | Bool | 291.3 | false | FALSE |
| Bit4 | Bool | 291.4 | false | FALSE |
| JogInc | Bool | 291.5 | false | FALSE |
| Bit6 | Bool | 291.6 | false | FALSE |
| Bit7 | Bool | 291.7 | false | FALSE |

(Fig. 3.72 C The values of command words of drive)

The internal structure is copied into the command drive structure to allow the search of the initial position.

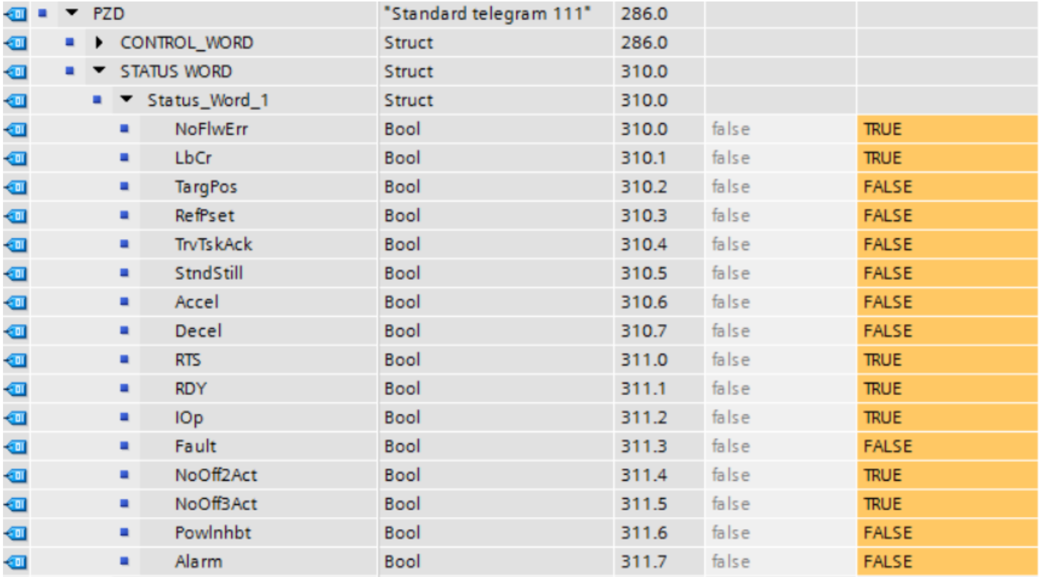
At this point the motor starts spinning in a positive direction until the physical zero position coincides with the virtual zero position.

The drive activities are monitored by means of the drive status words (figure 3.73).



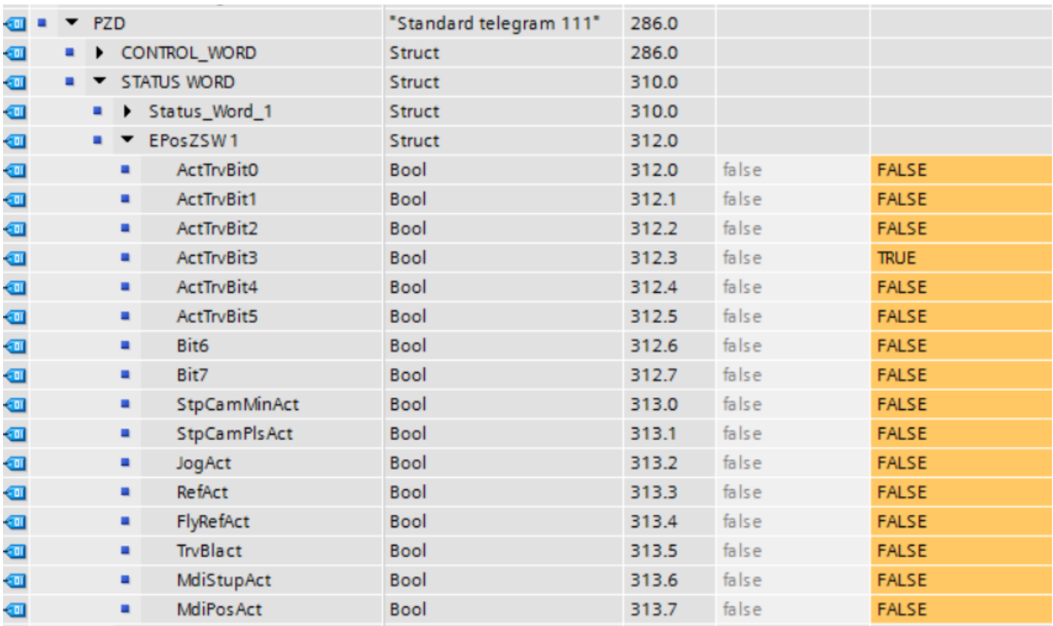
| | | | | |
|---|---|---|---|---|
| PZD | "Standard telegram 111" | 286.0 | | |
| CONTROL_WORD | Struct | 286.0 | | |
| STATUS WORD | Struct | 310.0 | | |
| Status_Word_1 | Struct | 310.0 | | |
| EPosZSW1 | Struct | 312.0 | | |
| EPosZSW2 | Struct | 314.0 | | |
| Status_Word_2 | Struct | 316.0 | | |
| Word6 | Word | 318.0 | 16#0 | 16#39CB |
| Position | DWord | 320.0 | 16#0 | 16#0100_6F3E |
| Velocity | DWord | 324.0 | 16#0 | 16#055C_43F0 |
| ErrNr | Word | 328.0 | 16#0 | 16#0000 |
| WarnNr | Word | 330.0 | 16#0 | 16#0000 |
| Reserved | Word | 332.0 | 16#0 | 16#0000 |

(Fig. 3.73 A The values of status words of drive)



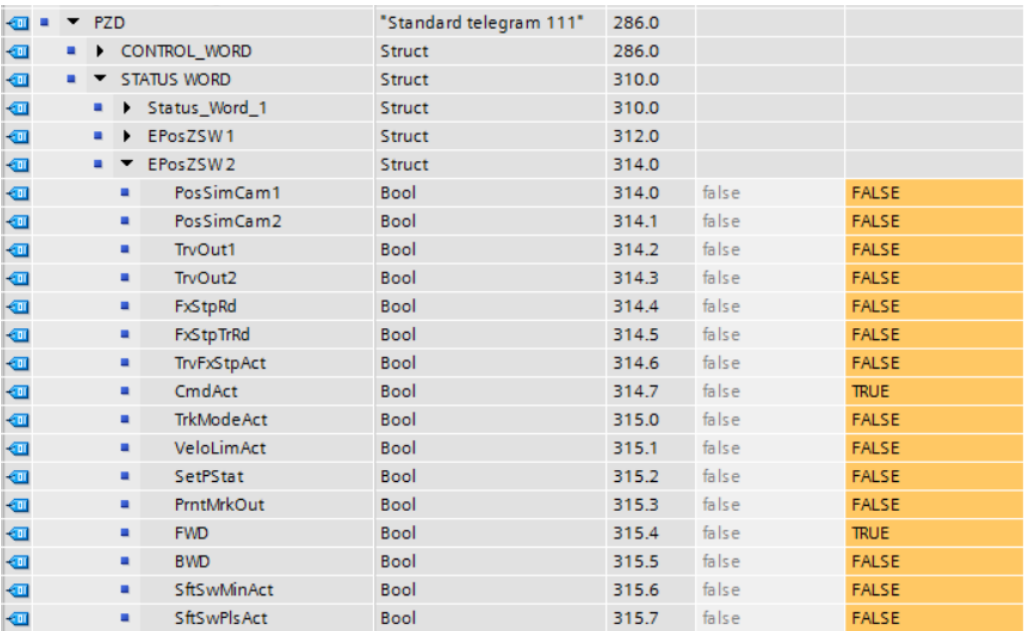
| | | | | |
|---|---|---|---|---|
| PZD | "Standard telegram 111" | 286.0 | | |
| CONTROL_WORD | Struct | 286.0 | | |
| STATUS WORD | Struct | 310.0 | | |
| Status_Word_1 | Struct | 310.0 | | |
| NoFlwErr | Bool | 310.0 | false | TRUE |
| LbCr | Bool | 310.1 | false | TRUE |
| TargPos | Bool | 310.2 | false | FALSE |
| RefPset | Bool | 310.3 | false | FALSE |
| TrvTskAck | Bool | 310.4 | false | FALSE |
| StndStill | Bool | 310.5 | false | FALSE |
| Accel | Bool | 310.6 | false | FALSE |
| Decel | Bool | 310.7 | false | FALSE |
| RTS | Bool | 311.0 | false | TRUE |
| RDY | Bool | 311.1 | false | TRUE |
| IOp | Bool | 311.2 | false | TRUE |
| Fault | Bool | 311.3 | false | FALSE |
| NoOff2Act | Bool | 311.4 | false | TRUE |
| NoOff3Act | Bool | 311.5 | false | TRUE |
| PowInhbt | Bool | 311.6 | false | FALSE |
| Alarm | Bool | 311.7 | false | FALSE |

(Fig. 3.73 B The values of status words of drive)



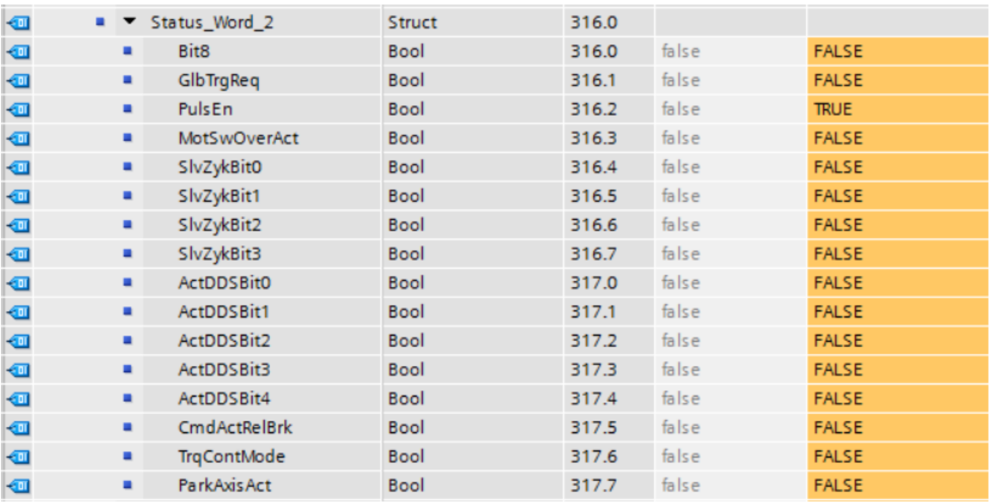
| | | | | |
|---|---|---|---|---|
| ▼ PZD | "Standard telegram 111" | 286.0 | | |
| ▶ CONTROL_WORD | Struct | 286.0 | | |
| ▼ STATUS WORD | Struct | 310.0 | | |
| ▶ Status_Word_1 | Struct | 310.0 | | |
| ▼ EPosZSW1 | Struct | 312.0 | | |
| ActTrvBit0 | Bool | 312.0 | false | FALSE |
| ActTrvBit1 | Bool | 312.1 | false | FALSE |
| ActTrvBit2 | Bool | 312.2 | false | FALSE |
| ActTrvBit3 | Bool | 312.3 | false | TRUE |
| ActTrvBit4 | Bool | 312.4 | false | FALSE |
| ActTrvBit5 | Bool | 312.5 | false | FALSE |
| Bit6 | Bool | 312.6 | false | FALSE |
| Bit7 | Bool | 312.7 | false | FALSE |
| StpCamMinAct | Bool | 313.0 | false | FALSE |
| StpCamPlsAct | Bool | 313.1 | false | FALSE |
| JogAct | Bool | 313.2 | false | FALSE |
| RefAct | Bool | 313.3 | false | FALSE |
| FlyRefAct | Bool | 313.4 | false | FALSE |
| TrvBlact | Bool | 313.5 | false | FALSE |
| MdiStupAct | Bool | 313.6 | false | FALSE |
| MdiPosAct | Bool | 313.7 | false | FALSE |

(Fig. 3.73 C The values of status words of drive)



| | | | | |
|---|---|---|---|---|
| ▼ PZD | "Standard telegram 111" | 286.0 | | |
| ▶ CONTROL_WORD | Struct | 286.0 | | |
| ▼ STATUS WORD | Struct | 310.0 | | |
| ▶ Status_Word_1 | Struct | 310.0 | | |
| ▶ EPosZSW1 | Struct | 312.0 | | |
| ▼ EPosZSW2 | Struct | 314.0 | | |
| PosSimCam1 | Bool | 314.0 | false | FALSE |
| PosSimCam2 | Bool | 314.1 | false | FALSE |
| TrvOut1 | Bool | 314.2 | false | FALSE |
| TrvOut2 | Bool | 314.3 | false | FALSE |
| FxStpRd | Bool | 314.4 | false | FALSE |
| FxStpTrRd | Bool | 314.5 | false | FALSE |
| TrvFxStpAct | Bool | 314.6 | false | FALSE |
| CmdAct | Bool | 314.7 | false | TRUE |
| TrkModeAct | Bool | 315.0 | false | FALSE |
| VeloLimAct | Bool | 315.1 | false | FALSE |
| SetPStat | Bool | 315.2 | false | FALSE |
| PrntMrkOut | Bool | 315.3 | false | FALSE |
| FWD | Bool | 315.4 | false | TRUE |
| BWD | Bool | 315.5 | false | FALSE |
| SftSwMinAct | Bool | 315.6 | false | FALSE |
| SftSwPlsAct | Bool | 315.7 | false | FALSE |

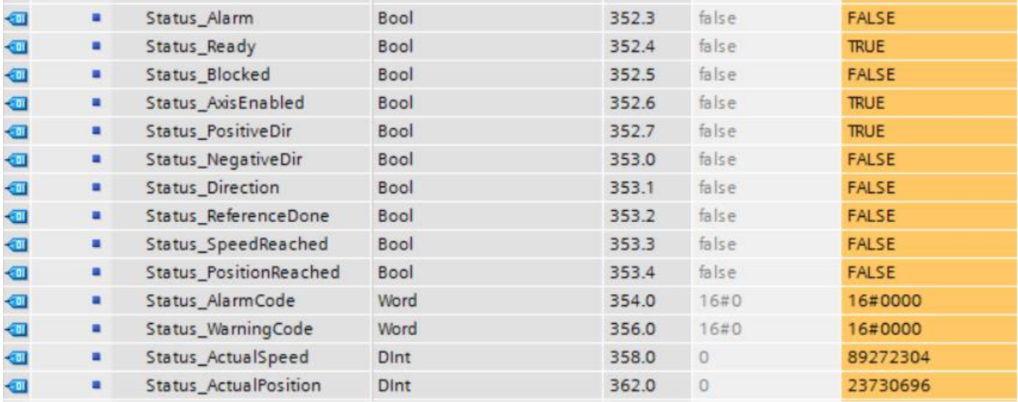
(Fig. 3.73 D The values of status words of drive)



| | | | | |
|---|---|---|---|---|
| Status_Word_2 | Struct | 316.0 | | |
| Bit8 | Bool | 316.0 | false | FALSE |
| GlbTrgReq | Bool | 316.1 | false | FALSE |
| PulsEn | Bool | 316.2 | false | TRUE |
| MotSwOverAct | Bool | 316.3 | false | FALSE |
| SlvZykBit0 | Bool | 316.4 | false | FALSE |
| SlvZykBit1 | Bool | 316.5 | false | FALSE |
| SlvZykBit2 | Bool | 316.6 | false | FALSE |
| SlvZykBit3 | Bool | 316.7 | false | FALSE |
| ActDDSBit0 | Bool | 317.0 | false | FALSE |
| ActDDSBit1 | Bool | 317.1 | false | FALSE |
| ActDDSBit2 | Bool | 317.2 | false | FALSE |
| ActDDSBit3 | Bool | 317.3 | false | FALSE |
| ActDDSBit4 | Bool | 317.4 | false | FALSE |
| CmdActRelBrk | Bool | 317.5 | false | FALSE |
| TrqContMode | Bool | 317.6 | false | FALSE |
| ParkAxisAct | Bool | 317.7 | false | FALSE |

(Fig. 3.73 E The values of status words of drive)

In a manner similar to previous tests, all drive states are copied to the internal structure (figures 3.73) and the most important bits are connected to the drive input signals (figure 3.74).

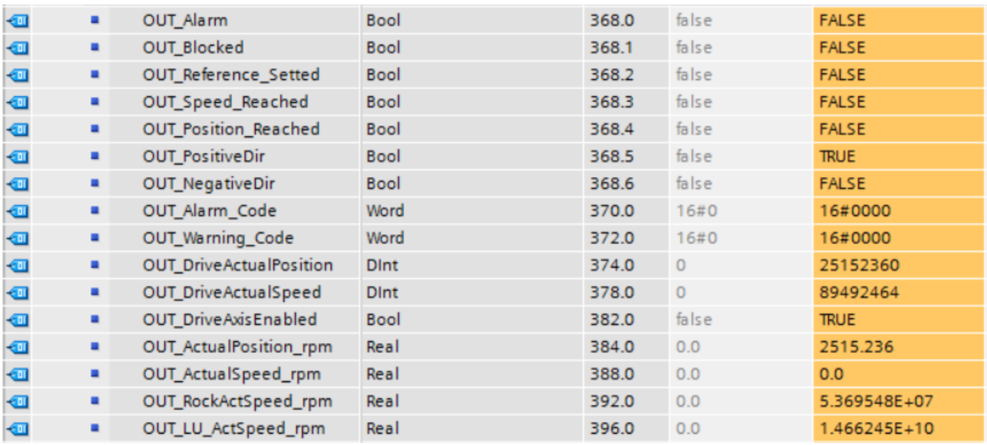


| | | | | |
|---|---|---|---|---|
| Status_Alarm | Bool | 352.3 | false | FALSE |
| Status_Ready | Bool | 352.4 | false | TRUE |
| Status_Blocked | Bool | 352.5 | false | FALSE |
| Status_AxisEnabled | Bool | 352.6 | false | TRUE |
| Status_PositiveDir | Bool | 352.7 | false | TRUE |
| Status_NegativeDir | Bool | 353.0 | false | FALSE |
| Status_Direction | Bool | 353.1 | false | FALSE |
| Status_ReferenceDone | Bool | 353.2 | false | FALSE |
| Status_SpeedReached | Bool | 353.3 | false | FALSE |
| Status_PositionReached | Bool | 353.4 | false | FALSE |
| Status_AlarmCode | Word | 354.0 | 16#0 | 16#0000 |
| Status_WarningCode | Word | 356.0 | 16#0 | 16#0000 |
| Status_ActualSpeed | DInt | 358.0 | 0 | 89272304 |
| Status_ActualPosition | DInt | 362.0 | 0 | 23730696 |

(Fig. 3.74 The values of drive input signals.)

It should be noted that regardless of the value of the input "IN_Direction" which is equal to 1, the rotation of the rotor is positive (Status_PositiveDir = TRUE). This happens because in the case of a reference search mode the direction is managed only by "IN_Home_Direction" input.

The user can read the drive states through user outputs (figure 3.75).



| | | | | |
|---|---|---|---|---|
| OUT_Alarm | Bool | 368.0 | false | FALSE |
| OUT_Blocked | Bool | 368.1 | false | FALSE |
| OUT_Reference_Setted | Bool | 368.2 | false | FALSE |
| OUT_Speed_Reached | Bool | 368.3 | false | FALSE |
| OUT_Position_Reached | Bool | 368.4 | false | FALSE |
| OUT_PositiveDir | Bool | 368.5 | false | TRUE |
| OUT_NegativeDir | Bool | 368.6 | false | FALSE |
| OUT_Alarm_Code | Word | 370.0 | 16#0 | 16#0000 |
| OUT_Warning_Code | Word | 372.0 | 16#0 | 16#0000 |
| OUT_DriveActualPosition | DInt | 374.0 | 0 | 25152360 |
| OUT_DriveActualSpeed | DInt | 378.0 | 0 | 89492464 |
| OUT_DriveAxisEnabled | Bool | 382.0 | false | TRUE |
| OUT_ActualPosition_rpm | Real | 384.0 | 0.0 | 2515.236 |
| OUT_ActualSpeed_rpm | Real | 388.0 | 0.0 | 0.0 |
| OUT_RockActSpeed_rpm | Real | 392.0 | 0.0 | 5.369548E+07 |
| OUT_LU_ActSpeed_rpm | Real | 396.0 | 0.0 | 1.466245E+10 |

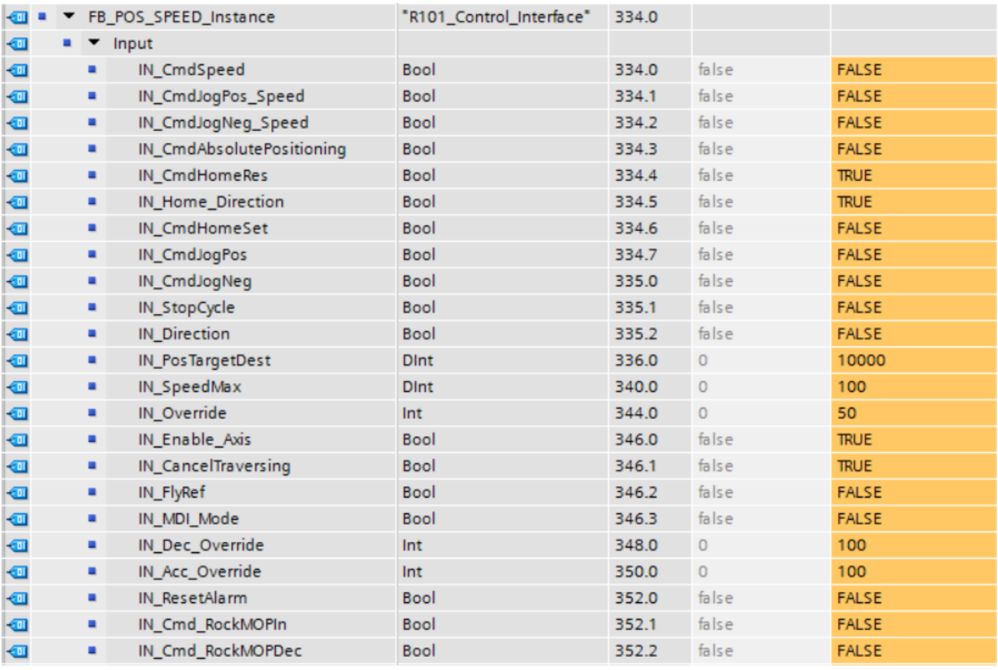
(Fig. 3.75 The values of user outputs)

The last phase of the test analyzes the negative reference search mode.

This mode is executed when the user imposes the following parameters:

- "IN_CmdHomeRes" = 1;
- "IN_Home_Direction" = 1;
- "IN_StopCycle" = 0;
- "IN_CancelTraversing" =1;
- "IN_Enable_Axis" = 1;
- "IN_MDI_Mode" = 1.

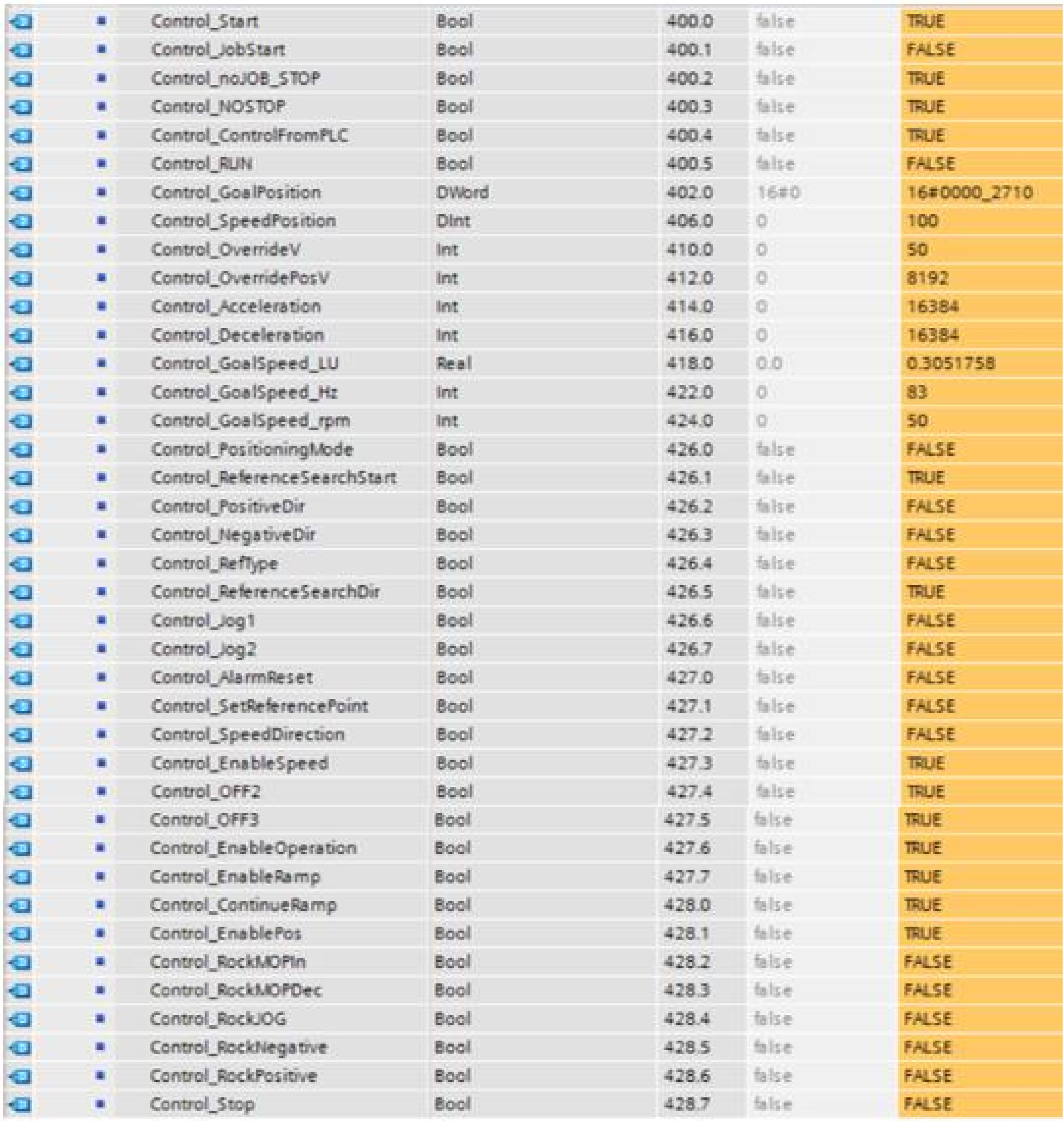
The parameters correspond to user inputs (figure 3.76).



| | | | | |
|---|---|---|---|---|
| FB_POS_SPEED_Instance | "R101_Control_Interface" | 334.0 | | |
| Input | | | | |
| IN_CmdSpeed | Bool | 334.0 | false | FALSE |
| IN_CmdJogPos_Speed | Bool | 334.1 | false | FALSE |
| IN_CmdJogNeg_Speed | Bool | 334.2 | false | FALSE |
| IN_CmdAbsolutePositioning | Bool | 334.3 | false | FALSE |
| IN_CmdHomeRes | Bool | 334.4 | false | TRUE |
| IN_Home_Direction | Bool | 334.5 | false | TRUE |
| IN_CmdHomeSet | Bool | 334.6 | false | FALSE |
| IN_CmdJogPos | Bool | 334.7 | false | FALSE |
| IN_CmdJogNeg | Bool | 335.0 | false | FALSE |
| IN_StopCycle | Bool | 335.1 | false | FALSE |
| IN_Direction | Bool | 335.2 | false | FALSE |
| IN_PosTargetDest | DInt | 336.0 | 0 | 10000 |
| IN_SpeedMax | DInt | 340.0 | 0 | 100 |
| IN_Override | Int | 344.0 | 0 | 50 |
| IN_Enable_Axis | Bool | 346.0 | false | TRUE |
| IN_CancelTraversing | Bool | 346.1 | false | TRUE |
| IN_FlyRef | Bool | 346.2 | false | FALSE |
| IN_MDI_Mode | Bool | 346.3 | false | FALSE |
| IN_Dec_Override | Int | 348.0 | 0 | 100 |
| IN_Acc_Override | Int | 350.0 | 0 | 100 |
| IN_ResetAlarm | Bool | 352.0 | false | FALSE |
| IN_Cmd_RockMOPIn | Bool | 352.1 | false | FALSE |
| IN_Cmd_RockMOPDec | Bool | 352.2 | false | FALSE |

(Fig. 3.76 User inputs)

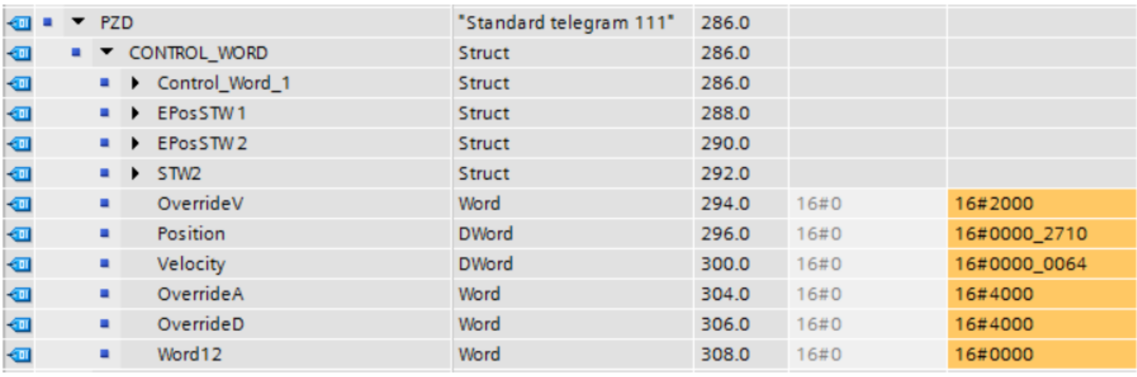
Drive outputs of the "R101_Control_Interface" block are shown in figure 3.77).



| | | | | |
|---|---|---|---|---|
| Control_Start | Bool | 400.0 | false | TRUE |
| Control_JobStart | Bool | 400.1 | false | FALSE |
| Control_noJOB_STOP | Bool | 400.2 | false | TRUE |
| Control_NOSTOP | Bool | 400.3 | false | TRUE |
| Control_ControlFromPLC | Bool | 400.4 | false | TRUE |
| Control_RUN | Bool | 400.5 | false | FALSE |
| Control_GoalPosition | DWord | 402.0 | 16#0 | 16#0000_2710 |
| Control_SpeedPosition | DInt | 406.0 | 0 | 100 |
| Control_OverrideV | Int | 410.0 | 0 | 50 |
| Control_OverridePosV | Int | 412.0 | 0 | 8192 |
| Control_Acceleration | Int | 414.0 | 0 | 16384 |
| Control_Deceleration | Int | 416.0 | 0 | 16384 |
| Control_GoalSpeed_LU | Real | 418.0 | 0.0 | 0.3051758 |
| Control_GoalSpeed_Hz | Int | 422.0 | 0 | 83 |
| Control_GoalSpeed_rpm | Int | 424.0 | 0 | 50 |
| Control_PositioningMode | Bool | 426.0 | false | FALSE |
| Control_ReferenceSearchStart | Bool | 426.1 | false | TRUE |
| Control_PositiveDir | Bool | 426.2 | false | FALSE |
| Control_NegativeDir | Bool | 426.3 | false | FALSE |
| Control_RefType | Bool | 426.4 | false | FALSE |
| Control_ReferenceSearchDir | Bool | 426.5 | false | TRUE |
| Control_Jog1 | Bool | 426.6 | false | FALSE |
| Control_Jog2 | Bool | 426.7 | false | FALSE |
| Control_AlarmReset | Bool | 427.0 | false | FALSE |
| Control_SetReferencePoint | Bool | 427.1 | false | FALSE |
| Control_SpeedDirection | Bool | 427.2 | false | FALSE |
| Control_EnableSpeed | Bool | 427.3 | false | TRUE |
| Control_OFF2 | Bool | 427.4 | false | TRUE |
| Control_OFF3 | Bool | 427.5 | false | TRUE |
| Control_EnableOperation | Bool | 427.6 | false | TRUE |
| Control_EnableRamp | Bool | 427.7 | false | TRUE |
| Control_ContinueRamp | Bool | 428.0 | false | TRUE |
| Control_EnablePos | Bool | 428.1 | false | TRUE |
| Control_RockMOPIn | Bool | 428.2 | false | FALSE |
| Control_RockMOPDec | Bool | 428.3 | false | FALSE |
| Control_RockJOG | Bool | 428.4 | false | FALSE |
| Control_RockNegative | Bool | 428.5 | false | FALSE |
| Control_RockPositive | Bool | 428.6 | false | FALSE |
| Control_Stop | Bool | 428.7 | false | FALSE |

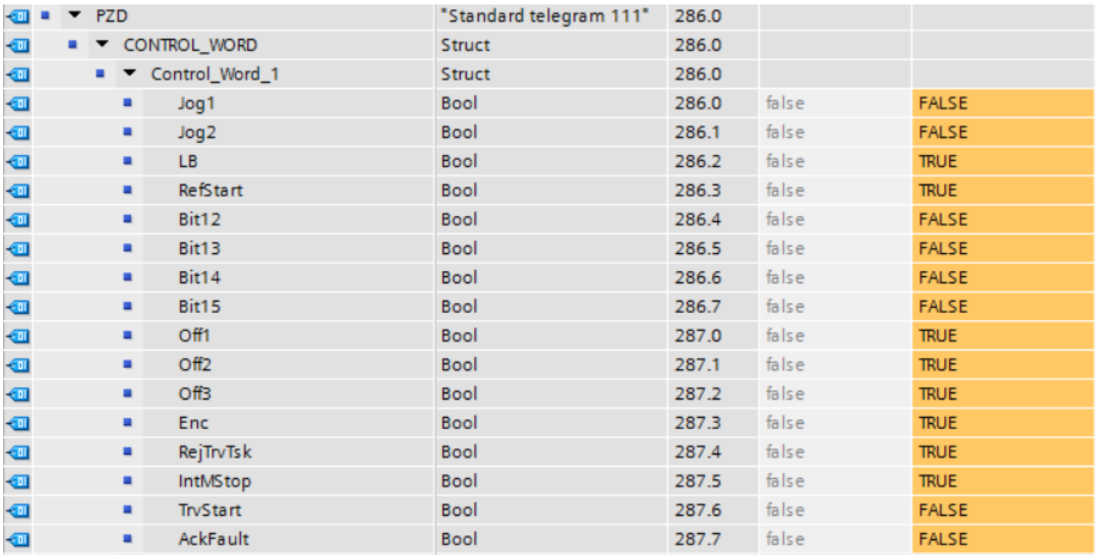
(Fig. 3.77 The values of drive outputs)

Subsequently, drive outputs are associated with the internal structure (figure 3.78) that is copied to the command drive structure.



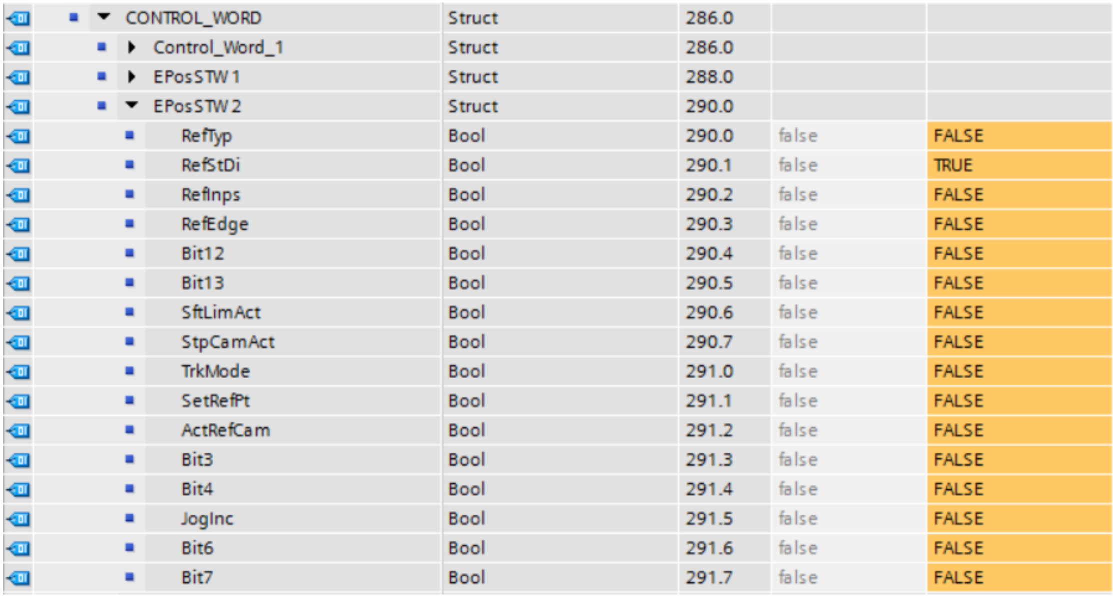
| | | | | |
|---|---|---|---|---|
| PZD | "Standard telegram 111" | 286.0 | | |
| CONTROL_WORD | Struct | 286.0 | | |
| Control_Word_1 | Struct | 286.0 | | |
| EPosSTW1 | Struct | 288.0 | | |
| EPosSTW2 | Struct | 290.0 | | |
| STW2 | Struct | 292.0 | | |
| OverrideV | Word | 294.0 | 16#0 | 16#2000 |
| Position | DWord | 296.0 | 16#0 | 16#0000_2710 |
| Velocity | DWord | 300.0 | 16#0 | 16#0000_0064 |
| OverrideA | Word | 304.0 | 16#0 | 16#4000 |
| OverrideD | Word | 306.0 | 16#0 | 16#4000 |
| Word12 | Word | 308.0 | 16#0 | 16#0000 |

(Fig. 3.78 A The values of command words of drive)



| | | | | |
|---|---|---|---|---|
| PZD | "Standard telegram 111" | 286.0 | | |
| CONTROL_WORD | Struct | 286.0 | | |
| Control_Word_1 | Struct | 286.0 | | |
| Jog1 | Bool | 286.0 | false | FALSE |
| Jog2 | Bool | 286.1 | false | FALSE |
| LB | Bool | 286.2 | false | TRUE |
| RefStart | Bool | 286.3 | false | TRUE |
| Bit12 | Bool | 286.4 | false | FALSE |
| Bit13 | Bool | 286.5 | false | FALSE |
| Bit14 | Bool | 286.6 | false | FALSE |
| Bit15 | Bool | 286.7 | false | FALSE |
| Off1 | Bool | 287.0 | false | TRUE |
| Off2 | Bool | 287.1 | false | TRUE |
| Off3 | Bool | 287.2 | false | TRUE |
| Enc | Bool | 287.3 | false | TRUE |
| RejTrvTsk | Bool | 287.4 | false | TRUE |
| IntMStop | Bool | 287.5 | false | TRUE |
| TrvStart | Bool | 287.6 | false | FALSE |
| AckFault | Bool | 287.7 | false | FALSE |

(Fig. 3.78 B The values of command words of drive)

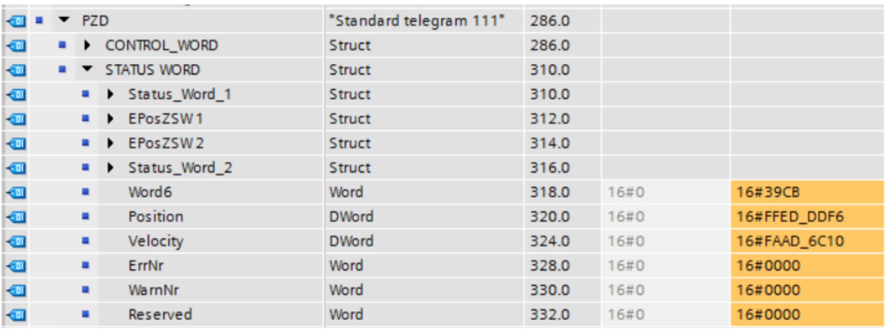


| | | | | |
|---|---|---|---|---|
| CONTROL_WORD | Struct | 286.0 | | |
| Control_Word_1 | Struct | 286.0 | | |
| EPosSTW1 | Struct | 288.0 | | |
| EPosSTW2 | Struct | 290.0 | | |
| RefTyp | Bool | 290.0 | false | FALSE |
| RefStDi | Bool | 290.1 | false | TRUE |
| RefInps | Bool | 290.2 | false | FALSE |
| RefEdge | Bool | 290.3 | false | FALSE |
| Bit12 | Bool | 290.4 | false | FALSE |
| Bit13 | Bool | 290.5 | false | FALSE |
| SftLimAct | Bool | 290.6 | false | FALSE |
| StpCamAct | Bool | 290.7 | false | FALSE |
| TrkMode | Bool | 291.0 | false | FALSE |
| SetRefPt | Bool | 291.1 | false | FALSE |
| ActRefCam | Bool | 291.2 | false | FALSE |
| Bit3 | Bool | 291.3 | false | FALSE |
| Bit4 | Bool | 291.4 | false | FALSE |
| JogInc | Bool | 291.5 | false | FALSE |
| Bit6 | Bool | 291.6 | false | FALSE |
| Bit7 | Bool | 291.7 | false | FALSE |

(Fig. 3.78 C The values of command words of drive)

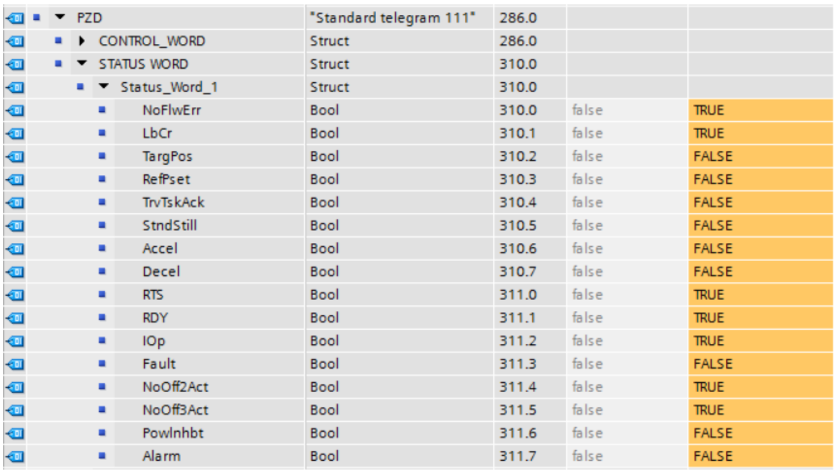
At this point the motor starts running in a negative direction until the physical zero position coincides with the virtual zero position.

The drive states are shown in the drive status words (figure 3.79).



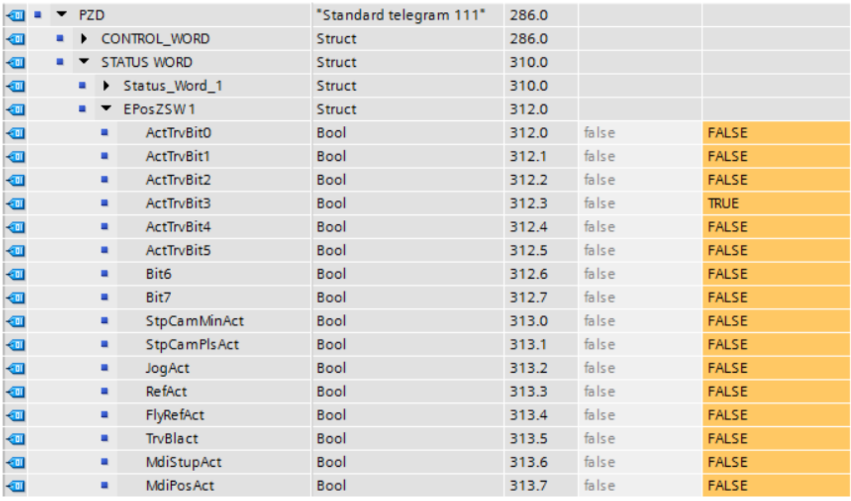
| | | | | |
|---|---|---|---|---|
| PZD | "Standard telegram 111" | 286.0 | | |
| CONTROL_WORD | Struct | 286.0 | | |
| STATUS WORD | Struct | 310.0 | | |
| Status_Word_1 | Struct | 310.0 | | |
| EPosZSW1 | Struct | 312.0 | | |
| EPosZSW2 | Struct | 314.0 | | |
| Status_Word_2 | Struct | 316.0 | | |
| Word6 | Word | 318.0 | 16#0 | 16#39CB |
| Position | DWord | 320.0 | 16#0 | 16#FFED_DDF6 |
| Velocity | DWord | 324.0 | 16#0 | 16#FAAD_6C10 |
| ErrNr | Word | 328.0 | 16#0 | 16#0000 |
| WarnNr | Word | 330.0 | 16#0 | 16#0000 |
| Reserved | Word | 332.0 | 16#0 | 16#0000 |

(Fig. 3.79 A The values of status words of drive)



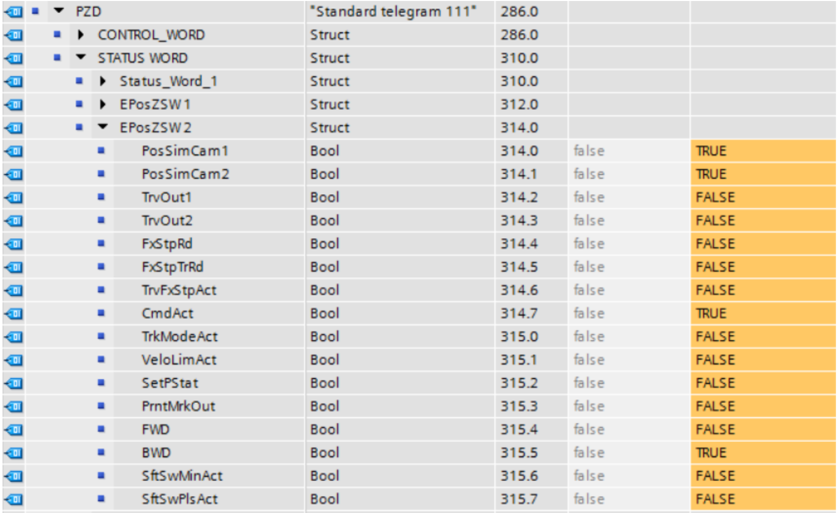
| | | | | |
|---|---|---|---|---|
| PZD | "Standard telegram 111" | 286.0 | | |
| CONTROL_WORD | Struct | 286.0 | | |
| STATUS WORD | Struct | 310.0 | | |
| Status_Word_1 | Struct | 310.0 | | |
| NoFlwErr | Bool | 310.0 | false | TRUE |
| LbCr | Bool | 310.1 | false | TRUE |
| TargPos | Bool | 310.2 | false | FALSE |
| RefPset | Bool | 310.3 | false | FALSE |
| TrvTskAck | Bool | 310.4 | false | FALSE |
| StndStill | Bool | 310.5 | false | FALSE |
| Accel | Bool | 310.6 | false | FALSE |
| Decel | Bool | 310.7 | false | FALSE |
| RTS | Bool | 311.0 | false | TRUE |
| RDY | Bool | 311.1 | false | TRUE |
| IOp | Bool | 311.2 | false | TRUE |
| Fault | Bool | 311.3 | false | FALSE |
| NoOff2Act | Bool | 311.4 | false | TRUE |
| NoOff3Act | Bool | 311.5 | false | TRUE |
| PowInhbt | Bool | 311.6 | false | FALSE |
| Alarm | Bool | 311.7 | false | FALSE |

(Fig. 3.79 B The values of status words of drive)



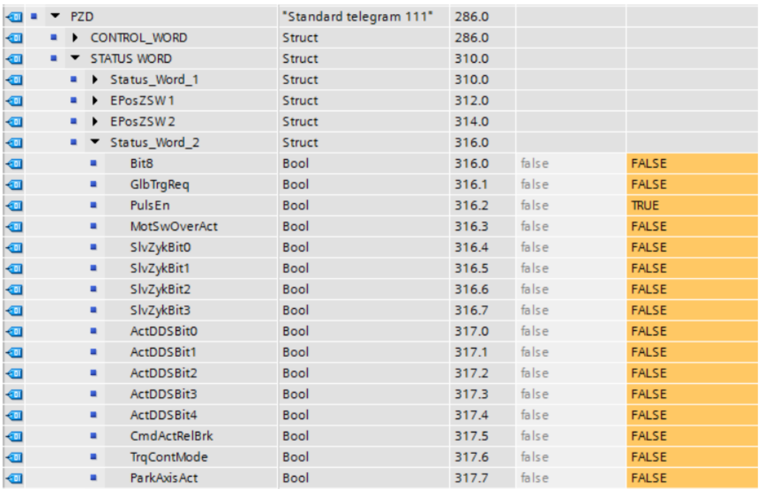
| | | | | |
|---|---|---|---|---|
| PZD | "Standard telegram 111" | 286.0 | | |
| CONTROL_WORD | Struct | 286.0 | | |
| STATUS WORD | Struct | 310.0 | | |
| Status_Word_1 | Struct | 310.0 | | |
| EPosZSW1 | Struct | 312.0 | | |
| ActTrvBit0 | Bool | 312.0 | false | FALSE |
| ActTrvBit1 | Bool | 312.1 | false | FALSE |
| ActTrvBit2 | Bool | 312.2 | false | FALSE |
| ActTrvBit3 | Bool | 312.3 | false | TRUE |
| ActTrvBit4 | Bool | 312.4 | false | FALSE |
| ActTrvBit5 | Bool | 312.5 | false | FALSE |
| Bit6 | Bool | 312.6 | false | FALSE |
| Bit7 | Bool | 312.7 | false | FALSE |
| StpCamMinAct | Bool | 313.0 | false | FALSE |
| StpCamPlsAct | Bool | 313.1 | false | FALSE |
| JogAct | Bool | 313.2 | false | FALSE |
| RefAct | Bool | 313.3 | false | FALSE |
| FlyRefAct | Bool | 313.4 | false | FALSE |
| TrvBlact | Bool | 313.5 | false | FALSE |
| MdiStupAct | Bool | 313.6 | false | FALSE |
| MdiPosAct | Bool | 313.7 | false | FALSE |

(Fig. 3.79 C The values of status words of drive)



| | | | | |
|---|---|---|---|---|
| PZD | "Standard telegram 111" | 286.0 | | |
| CONTROL_WORD | Struct | 286.0 | | |
| STATUS WORD | Struct | 310.0 | | |
| Status_Word_1 | Struct | 310.0 | | |
| EPosZSW1 | Struct | 312.0 | | |
| EPosZSW2 | Struct | 314.0 | | |
| PosSimCam1 | Bool | 314.0 | false | TRUE |
| PosSimCam2 | Bool | 314.1 | false | TRUE |
| TrvOut1 | Bool | 314.2 | false | FALSE |
| TrvOut2 | Bool | 314.3 | false | FALSE |
| FxStpRd | Bool | 314.4 | false | FALSE |
| FxStpTrRd | Bool | 314.5 | false | FALSE |
| TrvFxStpAct | Bool | 314.6 | false | FALSE |
| CmdAct | Bool | 314.7 | false | TRUE |
| TrkModeAct | Bool | 315.0 | false | FALSE |
| VeloLimAct | Bool | 315.1 | false | FALSE |
| SetPStat | Bool | 315.2 | false | FALSE |
| PrntMrkOut | Bool | 315.3 | false | FALSE |
| FWD | Bool | 315.4 | false | FALSE |
| BWD | Bool | 315.5 | false | TRUE |
| SftSwMinAct | Bool | 315.6 | false | FALSE |
| SftSwPlsAct | Bool | 315.7 | false | FALSE |

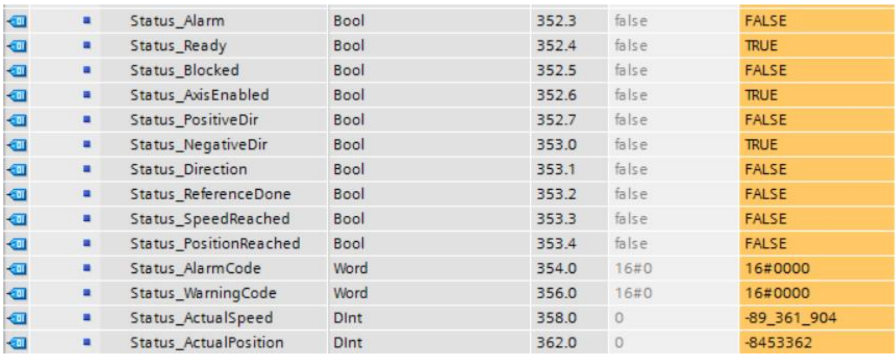
(Fig. 3.79 D The values of status words of drive)



| | | | | |
|---|---|---|---|---|
| PZD | "Standard telegram 111" | 286.0 | | |
| CONTROL_WORD | Struct | 286.0 | | |
| STATUS WORD | Struct | 310.0 | | |
| Status_Word_1 | Struct | 310.0 | | |
| EPosZSW1 | Struct | 312.0 | | |
| EPosZSW2 | Struct | 314.0 | | |
| Status_Word_2 | Struct | 316.0 | | |
| Bit8 | Bool | 316.0 | false | FALSE |
| GlbTrgReq | Bool | 316.1 | false | FALSE |
| PulsEn | Bool | 316.2 | false | TRUE |
| MotSwOverAct | Bool | 316.3 | false | FALSE |
| SlvZykBit0 | Bool | 316.4 | false | FALSE |
| SlvZykBit1 | Bool | 316.5 | false | FALSE |
| SlvZykBit2 | Bool | 316.6 | false | FALSE |
| SlvZykBit3 | Bool | 316.7 | false | FALSE |
| ActDDSBit0 | Bool | 317.0 | false | FALSE |
| ActDDSBit1 | Bool | 317.1 | false | FALSE |
| ActDDSBit2 | Bool | 317.2 | false | FALSE |
| ActDDSBit3 | Bool | 317.3 | false | FALSE |
| ActDDSBit4 | Bool | 317.4 | false | FALSE |
| CmdActRelBrk | Bool | 317.5 | false | FALSE |
| TrqContMode | Bool | 317.6 | false | FALSE |
| ParkAxisAct | Bool | 317.7 | false | FALSE |

(Fig. 3.79 E The values of status words of drive)

After the status drive structure has been copied to internal structure (figures 3.79), the most important bits are connected to the drive input signals (figure 3.80).

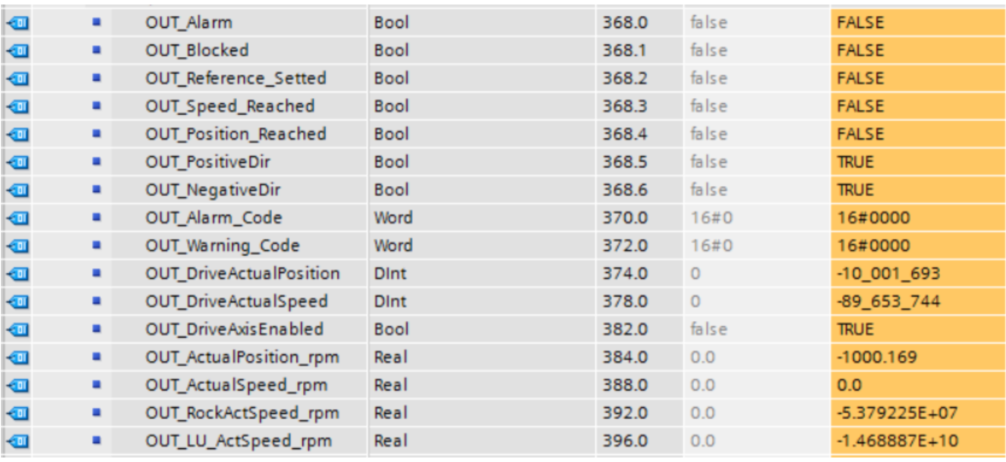


| | | | | |
|---|---|---|---|---|
| Status_Alarm | Bool | 352.3 | false | FALSE |
| Status_Ready | Bool | 352.4 | false | TRUE |
| Status_Blocked | Bool | 352.5 | false | FALSE |
| Status_AxisEnabled | Bool | 352.6 | false | TRUE |
| Status_PositiveDir | Bool | 352.7 | false | FALSE |
| Status_NegativeDir | Bool | 353.0 | false | TRUE |
| Status_Direction | Bool | 353.1 | false | FALSE |
| Status_ReferenceDone | Bool | 353.2 | false | FALSE |
| Status_SpeedReached | Bool | 353.3 | false | FALSE |
| Status_PositionReached | Bool | 353.4 | false | FALSE |
| Status_AlarmCode | Word | 354.0 | 16#0 | 16#0000 |
| Status_WarningCode | Word | 356.0 | 16#0 | 16#0000 |
| Status_ActualSpeed | DInt | 358.0 | 0 | -89_361_904 |
| Status_ActualPosition | DInt | 362.0 | 0 | -8453362 |

(Fig. 3.80 The values of drive input signals.)

In a manner similar to positive reference search mode, the rotation of the rotor is negative (Status_NegativeDir = TRUE) even if input "IN_Direction" is equal to 0.

The user can read the drive states through user outputs (figure 3.81).



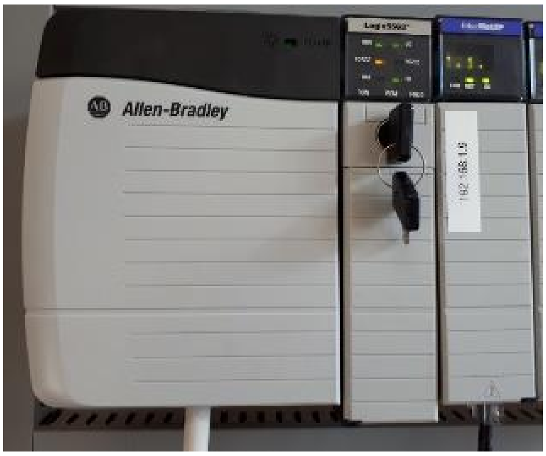
| | | | | |
|---|---|---|---|---|
| OUT_Alarm | Bool | 368.0 | false | FALSE |
| OUT_Blocked | Bool | 368.1 | false | FALSE |
| OUT_Reference_Setted | Bool | 368.2 | false | FALSE |
| OUT_Speed_Reached | Bool | 368.3 | false | FALSE |
| OUT_Position_Reached | Bool | 368.4 | false | FALSE |
| OUT_PositiveDir | Bool | 368.5 | false | TRUE |
| OUT_NegativeDir | Bool | 368.6 | false | TRUE |
| OUT_Alarm_Code | Word | 370.0 | 16#0 | 16#0000 |
| OUT_Warning_Code | Word | 372.0 | 16#0 | 16#0000 |
| OUT_DriveActualPosition | DInt | 374.0 | 0 | -10_001_693 |
| OUT_DriveActualSpeed | DInt | 378.0 | 0 | -89_653_744 |
| OUT_DriveAxisEnabled | Bool | 382.0 | false | TRUE |
| OUT_ActualPosition_rpm | Real | 384.0 | 0.0 | -1000.169 |
| OUT_ActualSpeed_rpm | Real | 388.0 | 0.0 | 0.0 |
| OUT_RockActSpeed_rpm | Real | 392.0 | 0.0 | -5.379225E+07 |
| OUT_LU_ActSpeed_rpm | Real | 396.0 | 0.0 | -1.468887E+10 |

(Fig. 3.81 The values of user outputs)

In conclusion, the test showed that the communication between Siemens PLC and Siemens S120 servo drive by means of the Standard Telegram 111 has been carried out correctly and all the functions of the program have been completely executed.

## *3.5 Test about Rockwell Automation inverter drive*

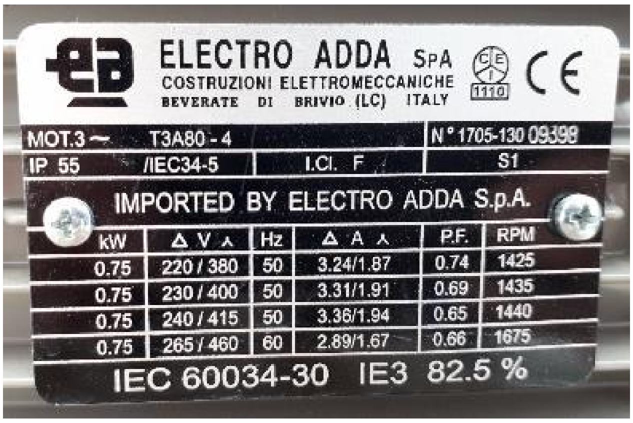
The test on the interface of communication with Rockwell Automation drive was executed by means of the "Logix5563” CPU (figure 3.82), an asynchronous electric motor and the PowerFlex 525 drive without external encoder (figure 3.83).



(Fig. 3.82 Rockwell Automation Logix5563 CPU)

(Fig. 3.83 Rockwell Automation PowerFlex 525 inverter drive)

The characteristics of the engine are in the motor plate (figure 3.84).



ELECTRO ADDA SpA
COSTRUZIONI ELETTROMECCANICHE
BEVERATE DI BRIVIO (LC) ITALY
CE

| | | | |
|---|---|---|---|
| MOT.3 ~ | T3A80 - 4 | | N° 1705-130 09398 |
| IP 55 | /IEC34-5 | I.Cl. F | S1 |

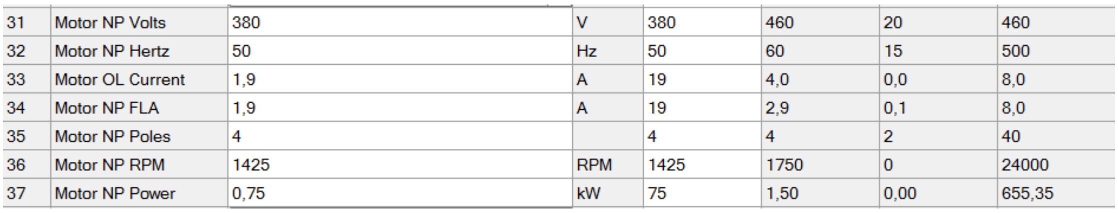
IMPORTED BY ELECTRO ADDA S.p.A.

| kW | Δ V ⅄ | Hz | Δ A ⅄ | P.F. | RPM |
|---|---|---|---|---|---|
| 0.75 | 220 / 380 | 50 | 3.24/1.87 | 0.74 | 1425 |
| 0.75 | 230 / 400 | 50 | 3.31/1.91 | 0.69 | 1435 |
| 0.75 | 240 / 415 | 50 | 3.36/1.94 | 0.65 | 1440 |
| 0.75 | 265 / 460 | 60 | 2.89/1.67 | 0.66 | 1675 |

IEC 60034-30 IE3 82.5 %

(Fig. 3.84 Electric motor plate)

This data was insert into the drive's expert list to impose engine characteristics on the drive.



| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 31 | Motor NP Volts | 380 | V | 380 | 460 | 20 | 460 |
| 32 | Motor NP Hertz | 50 | Hz | 50 | 60 | 15 | 500 |
| 33 | Motor OL Current | 1,9 | A | 19 | 4,0 | 0,0 | 8,0 |
| 34 | Motor NP FLA | 1,9 | A | 19 | 2,9 | 0,1 | 8,0 |
| 35 | Motor NP Poles | 4 | | 4 | 4 | 2 | 40 |
| 36 | Motor NP RPM | 1425 | RPM | 1425 | 1750 | 0 | 24000 |
| 37 | Motor NP Power | 0,75 | kW | 75 | 1,50 | 0,00 | 655,35 |

(Fig. 3.85 Electric motor plate)

The PowerFlex 525 drive is an inverter and it is able to control the engine speed. The communication structure of the drives belonging to the 52x family is described in appendix B.

The objective of the following test is the exact functioning of a Rockwell inverter by means of the PowerFlex 525 communication structure.

The first test involves running the speed control with a positive direction at a speed equal to half the maximum speed.

The first two activities are the addressing and the variables assignment with each communication bit of the interface. In this case, the interface used is represented in table 3.8 and table 3.9.

| Control_Interface | Rockwell Automation interface control word |
|---|---|
| Control_Start | My_Structure.Output.Start |
| Control_JobStart | |
| Control_noJOB_STOP | |
| Control_NOSTOP | |
| Control_ControlFromPLC | |
| Control_RUN | |
| Control_GoalPosition | |
| Control_SpeedPosition | |
| Control_OverrideV | |
| Control_OverridePosV | |
| Control_Acceleration | |
| Control_Deceleration | |
| Control_GoalSpeed_LU | |
| Control_GoalSpeed_Hz | My_Structure.Output.FreqCommand |
| Control_GoalSpeed_rpm | |
| Control_PositioningMode | |
| Control_ReferenceSearchStart | |
| Control_PositiveDir | |
| Control_NegativeDir | |
| Control_RefType | |
| Control_ReferenceSearchDir | |
| Control_Jog1 | |
| Control_Jog2 | |
| Control_AlarmReset | My_Structure.Output.ClearFaults |

| | |
|---|---|
| Control_SetReferencePoint | |
| Control_SpeedDirection | |
| Control_EnableSpeed | |
| Control_OFF2 | |
| Control_OFF3 | |
| Control_EnableOperation | |
| Control_EnableRamp | |
| Control_ContinueRamp | |
| Control_EnablePos | |
| Control_RockMOPIn | My_Structure.Output.MOPIncrement |
| Control_RockMOPDec | My_Structure.Output.MOPDecrement |
| Control_RockJOG | My_Structure.Output.Jog |
| Control_RockNegative | My_Structure.Output.Reverse |
| Control_RockPositive | My_Structure.Output.Forward |
| Control_Stop | My_Structure.Output.Stop |

(Table 3.8 Interface between Drive output signals of R101_Control_Interface” block and PowerFlex525 drive structure control words)

| Status_Interface | Rockwell Automation interface status word |
|---|---|
| Status_Alarm | |
| Status_Ready | My_Structure.Input.Ready |
| Status_Blocked | My_Structure.Input.Faulted |
| Status_AxisEnabled | My_Structure.Input.Active |
| Status_PositiveDir | |
| Status_NegativeDir | |
| Status_Direction | My_Structure.Input.ActualDir |
| Status_ReferenceDone | |
| Status_SpeedReached | My_Structure.Input.AtReference |
| Status_PositionReached | |
| Status_AlarmCode | |
| Status_WarningCode | |
| Status_ActualSpeed | My_Structure.Input.OutputFreq |
| Status_ActualPosition | |

(Table 3.9 Interface between Drive input signals of R101_Control_Interface” block and PowerFlex525 drive structure status words)

The procedure for activating this drive type depends on a specific sequence of commands. The sequence is given by the following bits:

1. “ClearFaults” inputs activation to reset the drive and alarms;

2. “Start” inputs activation to start the movement;

3. “Stop” inputs activation to lock the movement;

4. “Start” and “Stop” inputs reset;

5. the starting procedure again from step 1.

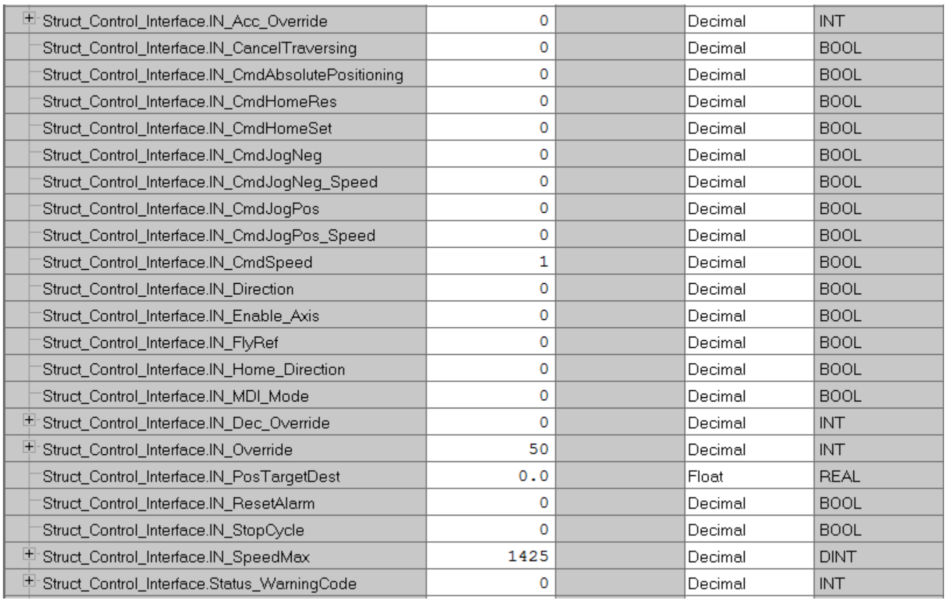
The second task is setting of the type of control that must be executed and its mode.

The parameters to set in the block "R101_Control_Interface" are:

- “IN_CmdSpeed” = 1;
- “IN_StopCycle” = 0;
- “IN_Direction” = 0;
- “IN_MaxSpeed” = 1425 [rpm];
- “IN_Override” = 50 [%].

It should be noted that the "IN_MaxSpeed" value is equal to the maximum engine speed. Indeed, unlike Siemens, Rockwell inverter does not calculate the desired speed according to expert list parameter, but the inverter imposes on the motor the transmitted speed.

The operating mode must be set using the user inputs. Figure 3.86 represents the correct values for user inputs to run a speed control with a positive direction at a speed equal to half the maximum speed.

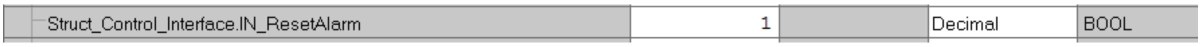


| | | | | |
|---|---|---|---|---|
| ⊞ Struct_Control_Interface.IN_Acc_Override | 0 | | Decimal | INT |
| Struct_Control_Interface.IN_CancelTraversing | 0 | | Decimal | BOOL |
| Struct_Control_Interface.IN_CmdAbsolutePositioning | 0 | | Decimal | BOOL |
| Struct_Control_Interface.IN_CmdHomeRes | 0 | | Decimal | BOOL |
| Struct_Control_Interface.IN_CmdHomeSet | 0 | | Decimal | BOOL |
| Struct_Control_Interface.IN_CmdJogNeg | 0 | | Decimal | BOOL |
| Struct_Control_Interface.IN_CmdJogNeg_Speed | 0 | | Decimal | BOOL |
| Struct_Control_Interface.IN_CmdJogPos | 0 | | Decimal | BOOL |
| Struct_Control_Interface.IN_CmdJogPos_Speed | 0 | | Decimal | BOOL |
| Struct_Control_Interface.IN_CmdSpeed | 1 | | Decimal | BOOL |
| Struct_Control_Interface.IN_Direction | 0 | | Decimal | BOOL |
| Struct_Control_Interface.IN_Enable_Axis | 0 | | Decimal | BOOL |
| Struct_Control_Interface.IN_FlyRef | 0 | | Decimal | BOOL |
| Struct_Control_Interface.IN_Home_Direction | 0 | | Decimal | BOOL |
| Struct_Control_Interface.IN_MDI_Mode | 0 | | Decimal | BOOL |
| ⊞ Struct_Control_Interface.IN_Dec_Override | 0 | | Decimal | INT |
| ⊞ Struct_Control_Interface.IN_Override | 50 | | Decimal | INT |
| Struct_Control_Interface.IN_PosTargetDest | 0.0 | | Float | REAL |
| Struct_Control_Interface.IN_ResetAlarm | 0 | | Decimal | BOOL |
| Struct_Control_Interface.IN_StopCycle | 0 | | Decimal | BOOL |
| ⊞ Struct_Control_Interface.IN_SpeedMax | 1425 | | Decimal | DINT |
| ⊞ Struct_Control_Interface.Status_WarningCode | 0 | | Decimal | INT |

(Fig. 3.86 The values of user inputs)

To activate the motion the Rockwell sequence must be performed.

The first activity is "IN_ReseAlarm" activation (figure 3.87).



| | | | | |
|---|---|---|---|---|
| Struct_Control_Interface.IN_ResetAlarm | 1 | | Decimal | BOOL |

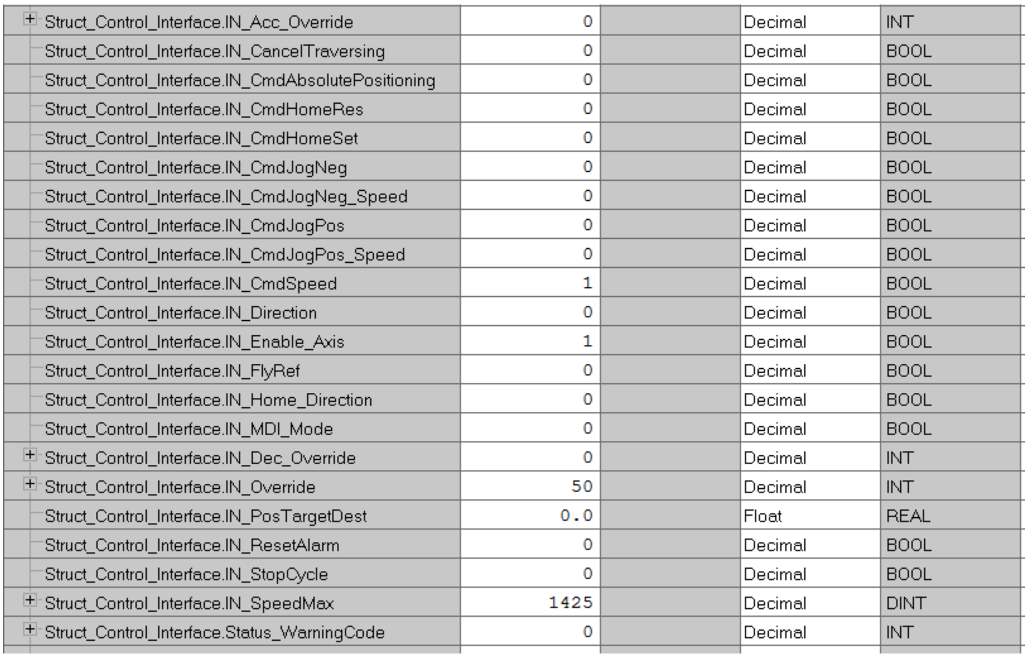
(Fig. 3.87 Set of IN_ResetAlarm)

The following operations are:

• reset of “IN_ResetAlarm” input (IN_ResetAlarm = 0);

• set of “IN_EnableAxis” input (IN_EnableAxis = 1).

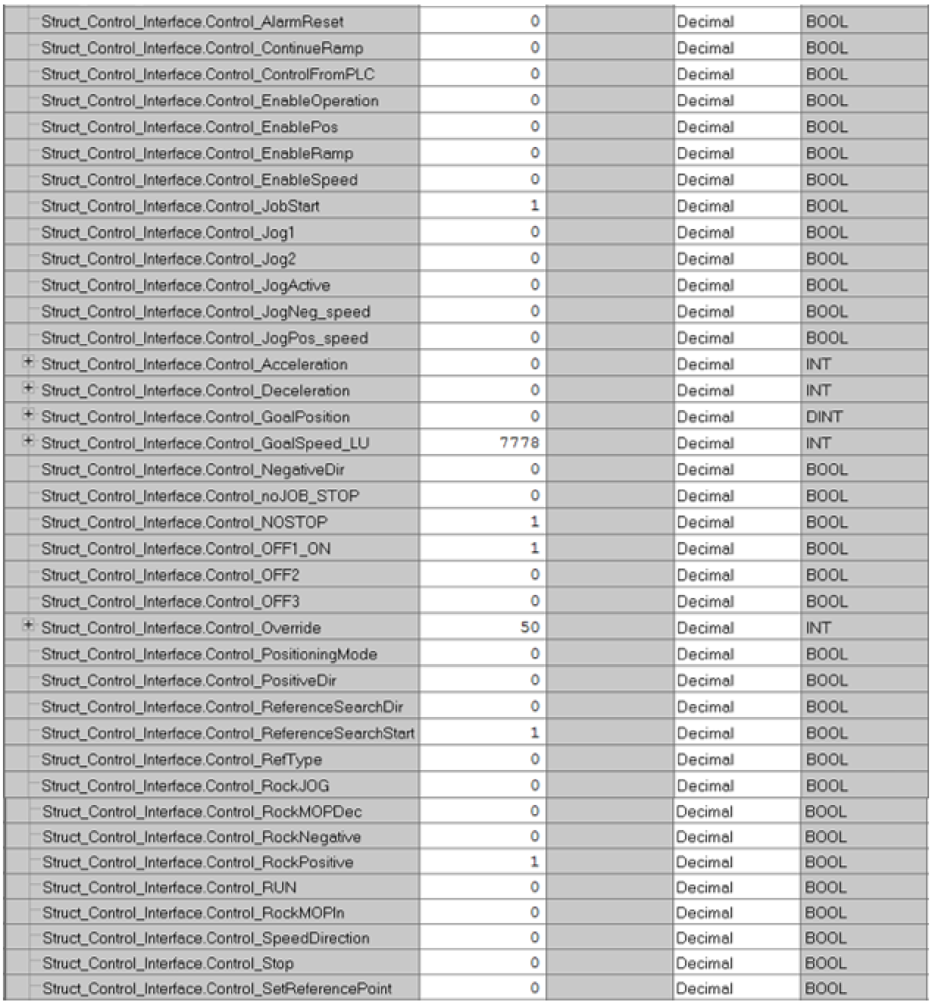
These operations update user inputs values (figure 3.88).



| | | | | |
|---|---|---|---|---|
| + Struct_Control_Interface.IN_Acc_Override | 0 | | Decimal | INT |
| Struct_Control_Interface.IN_CancelTraversing | 0 | | Decimal | BOOL |
| Struct_Control_Interface.IN_CmdAbsolutePositioning | 0 | | Decimal | BOOL |
| Struct_Control_Interface.IN_CmdHomeRes | 0 | | Decimal | BOOL |
| Struct_Control_Interface.IN_CmdHomeSet | 0 | | Decimal | BOOL |
| Struct_Control_Interface.IN_CmdJogNeg | 0 | | Decimal | BOOL |
| Struct_Control_Interface.IN_CmdJogNeg_Speed | 0 | | Decimal | BOOL |
| Struct_Control_Interface.IN_CmdJogPos | 0 | | Decimal | BOOL |
| Struct_Control_Interface.IN_CmdJogPos_Speed | 0 | | Decimal | BOOL |
| Struct_Control_Interface.IN_CmdSpeed | 1 | | Decimal | BOOL |
| Struct_Control_Interface.IN_Direction | 0 | | Decimal | BOOL |
| Struct_Control_Interface.IN_Enable_Axis | 1 | | Decimal | BOOL |
| Struct_Control_Interface.IN_FlyRef | 0 | | Decimal | BOOL |
| Struct_Control_Interface.IN_Home_Direction | 0 | | Decimal | BOOL |
| Struct_Control_Interface.IN_MDI_Mode | 0 | | Decimal | BOOL |
| + Struct_Control_Interface.IN_Dec_Override | 0 | | Decimal | INT |
| + Struct_Control_Interface.IN_Override | 50 | | Decimal | INT |
| Struct_Control_Interface.IN_PosTargetDest | 0.0 | | Float | REAL |
| Struct_Control_Interface.IN_ResetAlarm | 0 | | Decimal | BOOL |
| Struct_Control_Interface.IN_StopCycle | 0 | | Decimal | BOOL |
| + Struct_Control_Interface.IN_SpeedMax | 1425 | | Decimal | DINT |
| + Struct_Control_Interface.Status_WarningCode | 0 | | Decimal | INT |

(Fig. 3.88 New values of user inputs)

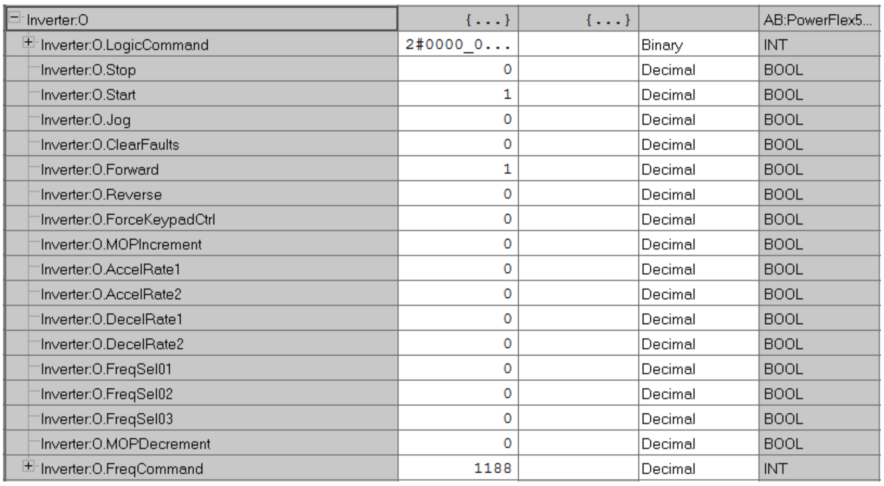
The R101 Control_Interface block reads the user's commands and generates the correct drive output values (figure 3.89). Each Drive output variable is connected to internal structure.



| | | | | |
|---|---|---|---|---|
| Struct_Control_Interface.Control_AlarmReset | 0 | | Decimal | BOOL |
| Struct_Control_Interface.Control_ContinueRamp | 0 | | Decimal | BOOL |
| Struct_Control_Interface.Control_ControlFromPLC | 0 | | Decimal | BOOL |
| Struct_Control_Interface.Control_EnableOperation | 0 | | Decimal | BOOL |
| Struct_Control_Interface.Control_EnablePos | 0 | | Decimal | BOOL |
| Struct_Control_Interface.Control_EnableRamp | 0 | | Decimal | BOOL |
| Struct_Control_Interface.Control_EnableSpeed | 0 | | Decimal | BOOL |
| Struct_Control_Interface.Control_JobStart | 1 | | Decimal | BOOL |
| Struct_Control_Interface.Control_Jog1 | 0 | | Decimal | BOOL |
| Struct_Control_Interface.Control_Jog2 | 0 | | Decimal | BOOL |
| Struct_Control_Interface.Control_JogActive | 0 | | Decimal | BOOL |
| Struct_Control_Interface.Control_JogNeg_speed | 0 | | Decimal | BOOL |
| Struct_Control_Interface.Control_JogPos_speed | 0 | | Decimal | BOOL |
| + Struct_Control_Interface.Control_Acceleration | 0 | | Decimal | INT |
| + Struct_Control_Interface.Control_Deceleration | 0 | | Decimal | INT |
| + Struct_Control_Interface.Control_GoalPosition | 0 | | Decimal | DINT |
| + Struct_Control_Interface.Control_GoalSpeed_LU | 7778 | | Decimal | INT |
| Struct_Control_Interface.Control_NegativeDir | 0 | | Decimal | BOOL |
| Struct_Control_Interface.Control_noJOB_STOP | 0 | | Decimal | BOOL |
| Struct_Control_Interface.Control_NOSTOP | 1 | | Decimal | BOOL |
| Struct_Control_Interface.Control_OFF1_ON | 1 | | Decimal | BOOL |
| Struct_Control_Interface.Control_OFF2 | 0 | | Decimal | BOOL |
| Struct_Control_Interface.Control_OFF3 | 0 | | Decimal | BOOL |
| + Struct_Control_Interface.Control_Override | 50 | | Decimal | INT |
| Struct_Control_Interface.Control_PositioningMode | 0 | | Decimal | BOOL |
| Struct_Control_Interface.Control_PositiveDir | 0 | | Decimal | BOOL |
| Struct_Control_Interface.Control_ReferenceSearchDir | 0 | | Decimal | BOOL |
| Struct_Control_Interface.Control_ReferenceSearchStart | 1 | | Decimal | BOOL |
| Struct_Control_Interface.Control_RefType | 0 | | Decimal | BOOL |
| Struct_Control_Interface.Control_RockJOG | 0 | | Decimal | BOOL |
| Struct_Control_Interface.Control_RockMOPDec | 0 | | Decimal | BOOL |
| Struct_Control_Interface.Control_RockNegative | 0 | | Decimal | BOOL |
| Struct_Control_Interface.Control_RockPositive | 1 | | Decimal | BOOL |
| Struct_Control_Interface.Control_RUN | 0 | | Decimal | BOOL |
| Struct_Control_Interface.Control_RockMOPIn | 0 | | Decimal | BOOL |
| Struct_Control_Interface.Control_SpeedDirection | 0 | | Decimal | BOOL |
| Struct_Control_Interface.Control_Stop | 0 | | Decimal | BOOL |
| Struct_Control_Interface.Control_SetReferencePoint | 0 | | Decimal | BOOL |

(Fig. 3.89 The values of drive outputs)

Drive output variables values are copied into the drive structure. In this way, the drive activation bits are set correctly (figure 3.90).

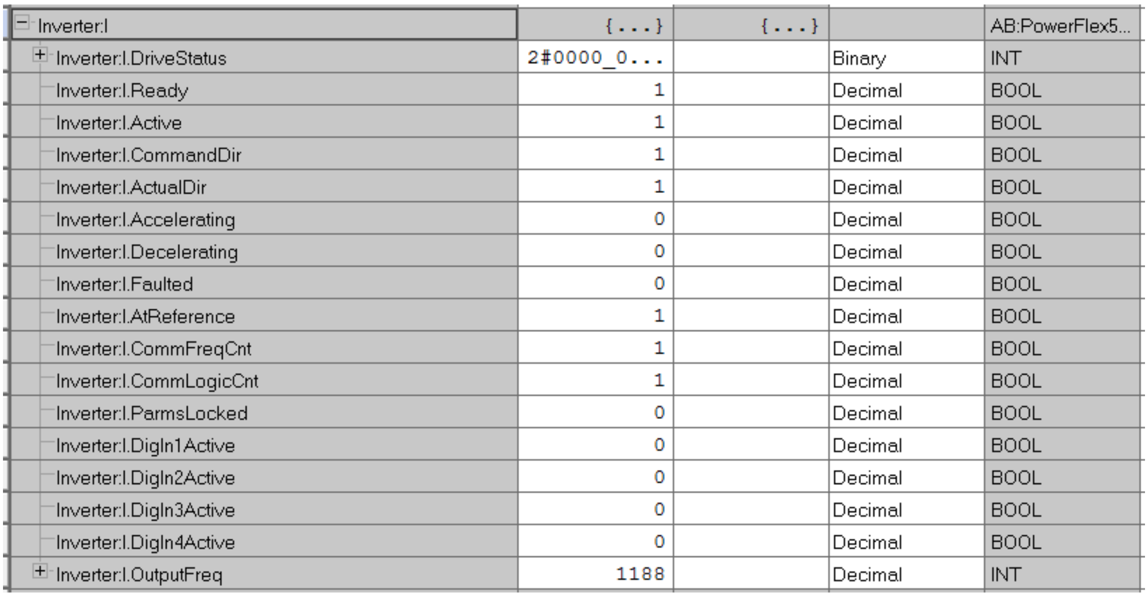


| | | | | |
|---|---|---|---|---|
| − Inverter:O | {...} | {...} | | AB:PowerFlex5... |
| + Inverter:O.LogicCommand | 2#0000_0... | | Binary | INT |
| Inverter:O.Stop | 0 | | Decimal | BOOL |
| Inverter:O.Start | 1 | | Decimal | BOOL |
| Inverter:O.Jog | 0 | | Decimal | BOOL |
| Inverter:O.ClearFaults | 0 | | Decimal | BOOL |
| Inverter:O.Forward | 1 | | Decimal | BOOL |
| Inverter:O.Reverse | 0 | | Decimal | BOOL |
| Inverter:O.ForceKeypadCtrl | 0 | | Decimal | BOOL |
| Inverter:O.MOPIncrement | 0 | | Decimal | BOOL |
| Inverter:O.AccelRate1 | 0 | | Decimal | BOOL |
| Inverter:O.AccelRate2 | 0 | | Decimal | BOOL |
| Inverter:O.DecelRate1 | 0 | | Decimal | BOOL |
| Inverter:O.DecelRate2 | 0 | | Decimal | BOOL |
| Inverter:O.FreqSel01 | 0 | | Decimal | BOOL |
| Inverter:O.FreqSel02 | 0 | | Decimal | BOOL |
| Inverter:O.FreqSel03 | 0 | | Decimal | BOOL |
| Inverter:O.MOPDecrement | 0 | | Decimal | BOOL |
| + Inverter:O.FreqCommand | 1188 | | Decimal | INT |

(Fig. 3.90 The values of command words of drive)

As a result, the engine starts rotating at the desired speed (1188 Hz = 712.8 rpm) with a positive direction, because the drive is running the speed control.

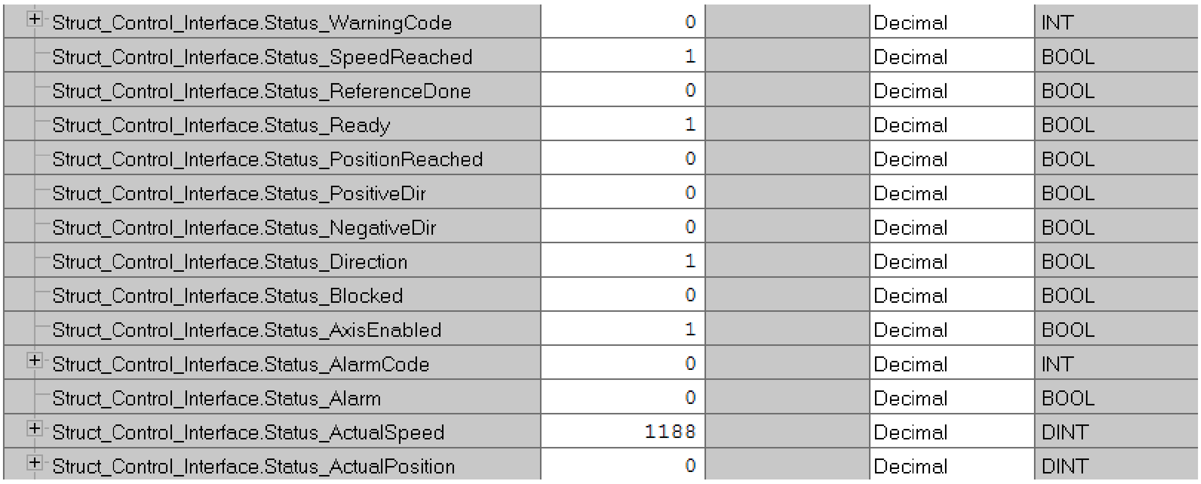
Proper operation is demonstrated by the word status of the drive (figure 3.91).



| ⊟ Inverter:I | {...} | {...} | | AB:PowerFlex5... |
|---|---|---|---|---|
| ⊞ Inverter:I.DriveStatus | 2#0000_0... | | Binary | INT |
| Inverter:I.Ready | 1 | | Decimal | BOOL |
| Inverter:I.Active | 1 | | Decimal | BOOL |
| Inverter:I.CommandDir | 1 | | Decimal | BOOL |
| Inverter:I.ActualDir | 1 | | Decimal | BOOL |
| Inverter:I.Accelerating | 0 | | Decimal | BOOL |
| Inverter:I.Decelerating | 0 | | Decimal | BOOL |
| Inverter:I.Faulted | 0 | | Decimal | BOOL |
| Inverter:I.AtReference | 1 | | Decimal | BOOL |
| Inverter:I.CommFreqCnt | 1 | | Decimal | BOOL |
| Inverter:I.CommLogicCnt | 1 | | Decimal | BOOL |
| Inverter:I.ParmsLocked | 0 | | Decimal | BOOL |
| Inverter:I.DigIn1Active | 0 | | Decimal | BOOL |
| Inverter:I.DigIn2Active | 0 | | Decimal | BOOL |
| Inverter:I.DigIn3Active | 0 | | Decimal | BOOL |
| Inverter:I.DigIn4Active | 0 | | Decimal | BOOL |
| ⊞ Inverter:I.OutputFreq | 1188 | | Decimal | INT |

(Fig. 3.91 The values of status words of drive)

It should be noted that the structure of figure 3.91 is equal to the status words of PowerFlex 525 data type (appendix B). These parameters provide the drive states and they must be transmitted to the interface block. As a result, drive status words are copied into the internal structure. The most important bits of the internal structure status words are connected to the drive input signals (figure 3.92).



| | | | | |
|---|---|---|---|---|
| ⊞ Struct_Control_Interface.Status_WarningCode | 0 | | Decimal | INT |
| Struct_Control_Interface.Status_SpeedReached | 1 | | Decimal | BOOL |
| Struct_Control_Interface.Status_ReferenceDone | 0 | | Decimal | BOOL |
| Struct_Control_Interface.Status_Ready | 1 | | Decimal | BOOL |
| Struct_Control_Interface.Status_PositionReached | 0 | | Decimal | BOOL |
| Struct_Control_Interface.Status_PositiveDir | 0 | | Decimal | BOOL |
| Struct_Control_Interface.Status_NegativeDir | 0 | | Decimal | BOOL |
| Struct_Control_Interface.Status_Direction | 1 | | Decimal | BOOL |
| Struct_Control_Interface.Status_Blocked | 0 | | Decimal | BOOL |
| Struct_Control_Interface.Status_AxisEnabled | 1 | | Decimal | BOOL |
| ⊞ Struct_Control_Interface.Status_AlarmCode | 0 | | Decimal | INT |
| Struct_Control_Interface.Status_Alarm | 0 | | Decimal | BOOL |
| ⊞ Struct_Control_Interface.Status_ActualSpeed | 1188 | | Decimal | DINT |
| ⊞ Struct_Control_Interface.Status_ActualPosition | 0 | | Decimal | DINT |

(Fig. 3.92 The values of drive input signals.)

Figure 3.92 shows that the drive works correctly:

- “Struct_Control_Interface.Status_Ready” = 1;
- “Struct_Control_Interface.Status_Direction” =1 (positive direction);
- “Struct_Control_Interface.Status_Blocked” = 0;
- “Struct_Control_Interface.Status_SpeedReached” =1;
- “Struct_Control_Interface.Status_ActualSpeed” = 1188.

It should be noted that the speed supplied is different from 712,5 [rpm] (equation 3.8).

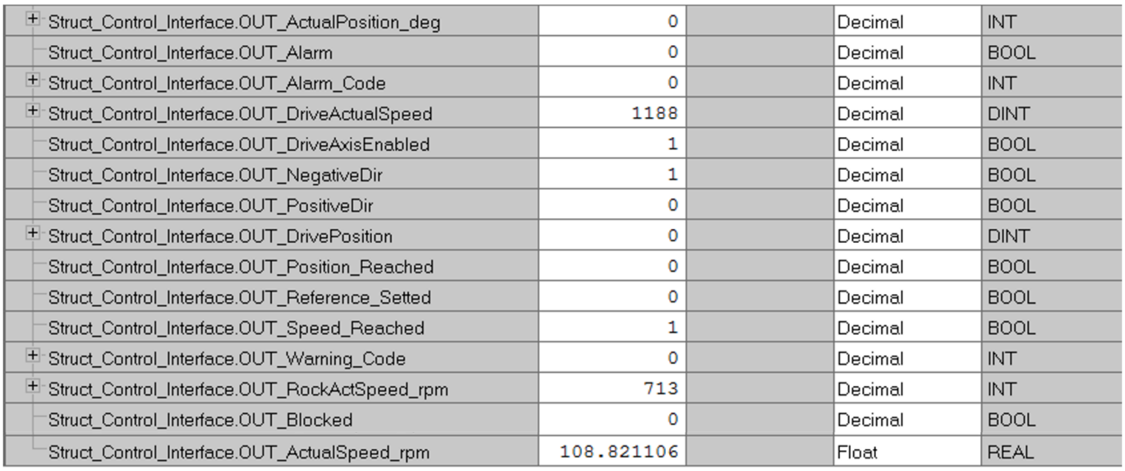
$$1350 * \left(\frac{50}{100}\right) = 712{,}5\ [rpm] \tag{3.8}$$

This difference is caused by the unit of measure: as has been observed in chapter 2, the Rockwell Automation speed unit is the Hz cent (equation 3.9).

$$1188 = 11{,}88\ [Hz] = 712{,}8\ [rpm] \tag{3.9}$$

As a result, the drive's response value is equivalent to the desired speed value. In other words, the drive is working properly.

Finally, figure 3.93 expresses the parameters transmitted to the user.



| | | | | |
|---|---|---|---|---|
| Struct_Control_Interface.OUT_ActualPosition_deg | 0 | | Decimal | INT |
| Struct_Control_Interface.OUT_Alarm | 0 | | Decimal | BOOL |
| Struct_Control_Interface.OUT_Alarm_Code | 0 | | Decimal | INT |
| Struct_Control_Interface.OUT_DriveActualSpeed | 1188 | | Decimal | DINT |
| Struct_Control_Interface.OUT_DriveAxisEnabled | 1 | | Decimal | BOOL |
| Struct_Control_Interface.OUT_NegativeDir | 1 | | Decimal | BOOL |
| Struct_Control_Interface.OUT_PositiveDir | 0 | | Decimal | BOOL |
| Struct_Control_Interface.OUT_DrivePosition | 0 | | Decimal | DINT |
| Struct_Control_Interface.OUT_Position_Reached | 0 | | Decimal | BOOL |
| Struct_Control_Interface.OUT_Reference_Setted | 0 | | Decimal | BOOL |
| Struct_Control_Interface.OUT_Speed_Reached | 1 | | Decimal | BOOL |
| Struct_Control_Interface.OUT_Warning_Code | 0 | | Decimal | INT |
| Struct_Control_Interface.OUT_RockActSpeed_rpm | 713 | | Decimal | INT |
| Struct_Control_Interface.OUT_Blocked | 0 | | Decimal | BOOL |
| Struct_Control_Interface.OUT_ActualSpeed_rpm | 108.821106 | | Float | REAL |

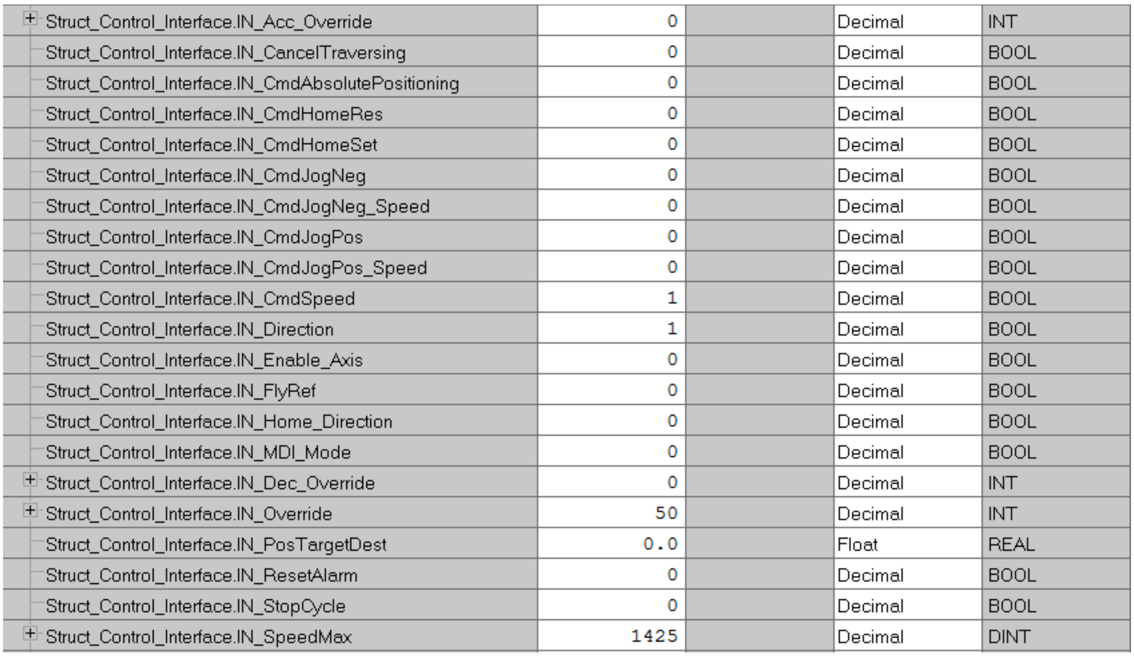
(Fig. 3.93 The values of user outputs)

The second phase of the test is the study of the speed control with a negative direction at a speed equal to half the maximum speed.

In this case, the parameters to set in the block "R101_Control_Interface" are:

- "IN_CmdSpeed" = 1;
- "IN_StopCycle" = 0;
- "IN_Direction" = 1;
- "IN_MaxSpeed" = 1425 [rpm];
- "IN_Override" = 50 [%];

The operating mode must be set using the user inputs. Figure 3.94 represents the correct values for user inputs to run a speed control with a negative direction at a speed equal to half the maximum speed.



| | | | | |
|---|---|---|---|---|
| + Struct_Control_Interface.IN_Acc_Override | 0 | | Decimal | INT |
| Struct_Control_Interface.IN_CancelTraversing | 0 | | Decimal | BOOL |
| Struct_Control_Interface.IN_CmdAbsolutePositioning | 0 | | Decimal | BOOL |
| Struct_Control_Interface.IN_CmdHomeRes | 0 | | Decimal | BOOL |
| Struct_Control_Interface.IN_CmdHomeSet | 0 | | Decimal | BOOL |
| Struct_Control_Interface.IN_CmdJogNeg | 0 | | Decimal | BOOL |
| Struct_Control_Interface.IN_CmdJogNeg_Speed | 0 | | Decimal | BOOL |
| Struct_Control_Interface.IN_CmdJogPos | 0 | | Decimal | BOOL |
| Struct_Control_Interface.IN_CmdJogPos_Speed | 0 | | Decimal | BOOL |
| Struct_Control_Interface.IN_CmdSpeed | 1 | | Decimal | BOOL |
| Struct_Control_Interface.IN_Direction | 1 | | Decimal | BOOL |
| Struct_Control_Interface.IN_Enable_Axis | 0 | | Decimal | BOOL |
| Struct_Control_Interface.IN_FlyRef | 0 | | Decimal | BOOL |
| Struct_Control_Interface.IN_Home_Direction | 0 | | Decimal | BOOL |
| Struct_Control_Interface.IN_MDI_Mode | 0 | | Decimal | BOOL |
| + Struct_Control_Interface.IN_Dec_Override | 0 | | Decimal | INT |
| + Struct_Control_Interface.IN_Override | 50 | | Decimal | INT |
| Struct_Control_Interface.IN_PosTargetDest | 0.0 | | Float | REAL |
| Struct_Control_Interface.IN_ResetAlarm | 0 | | Decimal | BOOL |
| Struct_Control_Interface.IN_StopCycle | 0 | | Decimal | BOOL |
| + Struct_Control_Interface.IN_SpeedMax | 1425 | | Decimal | DINT |

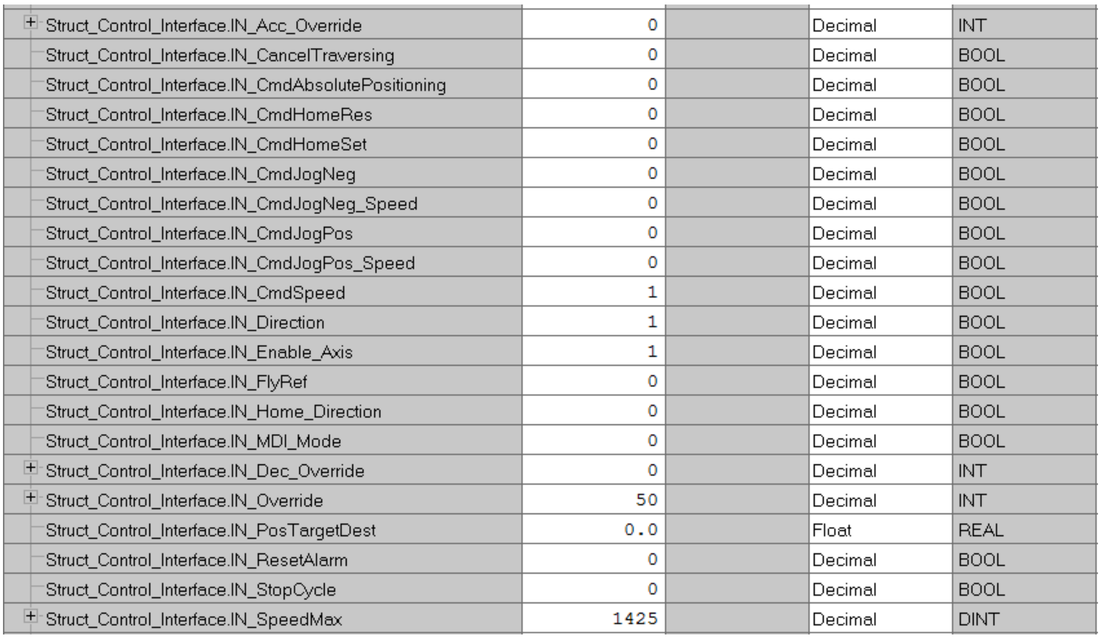
(Fig. 3.94 The values of user inputs)

In a manner similar to positive speed control, the procedure of activation must be performed.

The following operations are:

1. set of "IN_ResetAlarm" input (IN_ResetAlarm = 1);
2. reset of "IN_ResetAlarm" input (IN_ResetAlarm = 0);
3. set of "IN_EnableAxis" input (IN_EnableAxis = 1).

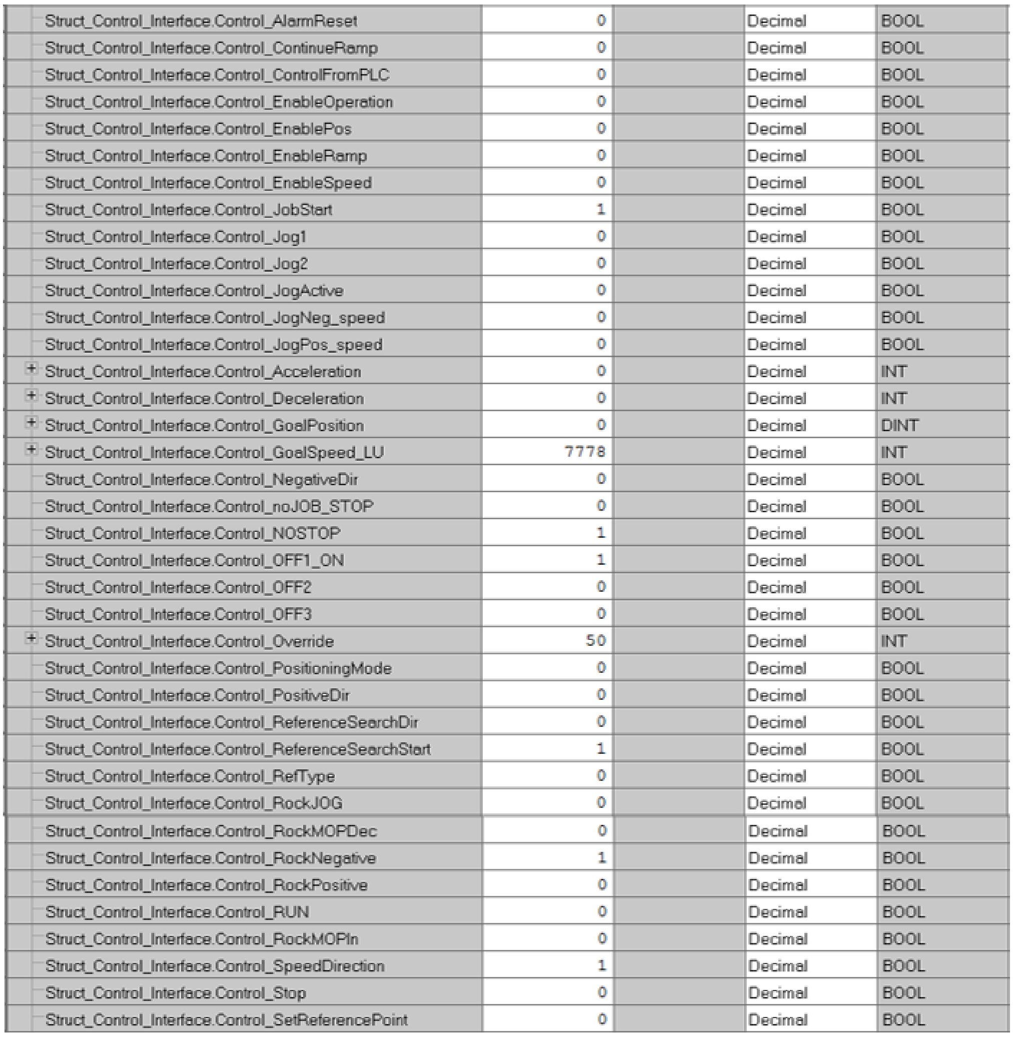
As a consequence, user input values change (figure 3.95).



| | | | | |
|---|---|---|---|---|
| + Struct_Control_Interface.IN_Acc_Override | 0 | | Decimal | INT |
| Struct_Control_Interface.IN_CancelTraversing | 0 | | Decimal | BOOL |
| Struct_Control_Interface.IN_CmdAbsolutePositioning | 0 | | Decimal | BOOL |
| Struct_Control_Interface.IN_CmdHomeRes | 0 | | Decimal | BOOL |
| Struct_Control_Interface.IN_CmdHomeSet | 0 | | Decimal | BOOL |
| Struct_Control_Interface.IN_CmdJogNeg | 0 | | Decimal | BOOL |
| Struct_Control_Interface.IN_CmdJogNeg_Speed | 0 | | Decimal | BOOL |
| Struct_Control_Interface.IN_CmdJogPos | 0 | | Decimal | BOOL |
| Struct_Control_Interface.IN_CmdJogPos_Speed | 0 | | Decimal | BOOL |
| Struct_Control_Interface.IN_CmdSpeed | 1 | | Decimal | BOOL |
| Struct_Control_Interface.IN_Direction | 1 | | Decimal | BOOL |
| Struct_Control_Interface.IN_Enable_Axis | 1 | | Decimal | BOOL |
| Struct_Control_Interface.IN_FlyRef | 0 | | Decimal | BOOL |
| Struct_Control_Interface.IN_Home_Direction | 0 | | Decimal | BOOL |
| Struct_Control_Interface.IN_MDI_Mode | 0 | | Decimal | BOOL |
| + Struct_Control_Interface.IN_Dec_Override | 0 | | Decimal | INT |
| + Struct_Control_Interface.IN_Override | 50 | | Decimal | INT |
| Struct_Control_Interface.IN_PosTargetDest | 0.0 | | Float | REAL |
| Struct_Control_Interface.IN_ResetAlarm | 0 | | Decimal | BOOL |
| Struct_Control_Interface.IN_StopCycle | 0 | | Decimal | BOOL |
| + Struct_Control_Interface.IN_SpeedMax | 1425 | | Decimal | DINT |

(Fig. 3.95 New values of user inputs)

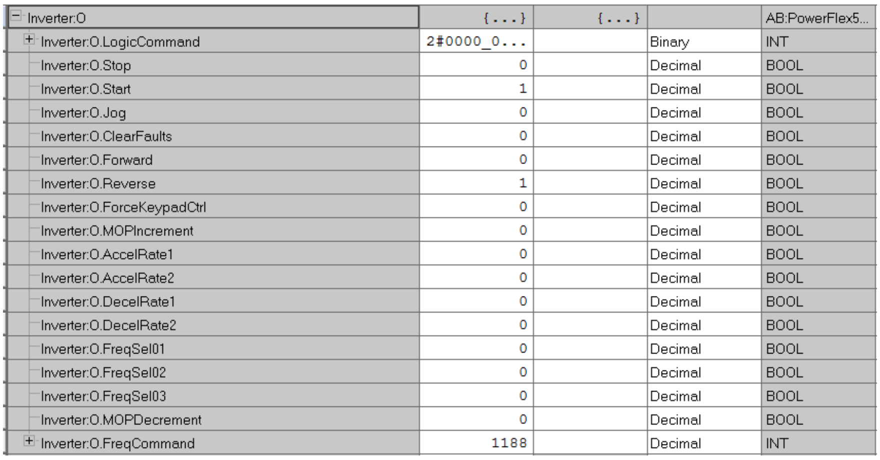
The R101 Control_Interface block generates the exact drive output values according to the user commands (figure 3.96). Each drive output is connected to internal structure through the R101 Control_Interface block interface.



| | | | | |
|---|---|---|---|---|
| Struct_Control_Interface.Control_AlarmReset | 0 | | Decimal | BOOL |
| Struct_Control_Interface.Control_ContinueRamp | 0 | | Decimal | BOOL |
| Struct_Control_Interface.Control_ControlFromPLC | 0 | | Decimal | BOOL |
| Struct_Control_Interface.Control_EnableOperation | 0 | | Decimal | BOOL |
| Struct_Control_Interface.Control_EnablePos | 0 | | Decimal | BOOL |
| Struct_Control_Interface.Control_EnableRamp | 0 | | Decimal | BOOL |
| Struct_Control_Interface.Control_EnableSpeed | 0 | | Decimal | BOOL |
| Struct_Control_Interface.Control_JobStart | 1 | | Decimal | BOOL |
| Struct_Control_Interface.Control_Jog1 | 0 | | Decimal | BOOL |
| Struct_Control_Interface.Control_Jog2 | 0 | | Decimal | BOOL |
| Struct_Control_Interface.Control_JogActive | 0 | | Decimal | BOOL |
| Struct_Control_Interface.Control_JogNeg_speed | 0 | | Decimal | BOOL |
| Struct_Control_Interface.Control_JogPos_speed | 0 | | Decimal | BOOL |
| + Struct_Control_Interface.Control_Acceleration | 0 | | Decimal | INT |
| + Struct_Control_Interface.Control_Deceleration | 0 | | Decimal | INT |
| + Struct_Control_Interface.Control_GoalPosition | 0 | | Decimal | DINT |
| + Struct_Control_Interface.Control_GoalSpeed_LU | 7778 | | Decimal | INT |
| Struct_Control_Interface.Control_NegativeDir | 0 | | Decimal | BOOL |
| Struct_Control_Interface.Control_noJOB_STOP | 0 | | Decimal | BOOL |
| Struct_Control_Interface.Control_NOSTOP | 1 | | Decimal | BOOL |
| Struct_Control_Interface.Control_OFF1_ON | 1 | | Decimal | BOOL |
| Struct_Control_Interface.Control_OFF2 | 0 | | Decimal | BOOL |
| Struct_Control_Interface.Control_OFF3 | 0 | | Decimal | BOOL |
| + Struct_Control_Interface.Control_Override | 50 | | Decimal | INT |
| Struct_Control_Interface.Control_PositioningMode | 0 | | Decimal | BOOL |
| Struct_Control_Interface.Control_PositiveDir | 0 | | Decimal | BOOL |
| Struct_Control_Interface.Control_ReferenceSearchDir | 0 | | Decimal | BOOL |
| Struct_Control_Interface.Control_ReferenceSearchStart | 1 | | Decimal | BOOL |
| Struct_Control_Interface.Control_RefType | 0 | | Decimal | BOOL |
| Struct_Control_Interface.Control_RockJOG | 0 | | Decimal | BOOL |
| Struct_Control_Interface.Control_RockMOPDec | 0 | | Decimal | BOOL |
| Struct_Control_Interface.Control_RockNegative | 1 | | Decimal | BOOL |
| Struct_Control_Interface.Control_RockPositive | 0 | | Decimal | BOOL |
| Struct_Control_Interface.Control_RUN | 0 | | Decimal | BOOL |
| Struct_Control_Interface.Control_RockMOPIn | 0 | | Decimal | BOOL |
| Struct_Control_Interface.Control_SpeedDirection | 1 | | Decimal | BOOL |
| Struct_Control_Interface.Control_Stop | 0 | | Decimal | BOOL |
| Struct_Control_Interface.Control_SetReferencePoint | 0 | | Decimal | BOOL |

(Fig. 3.96 The values of drive outputs)

The block copies drive output structure into the drive structure and then the drive activation bits are set (figure 3.97).

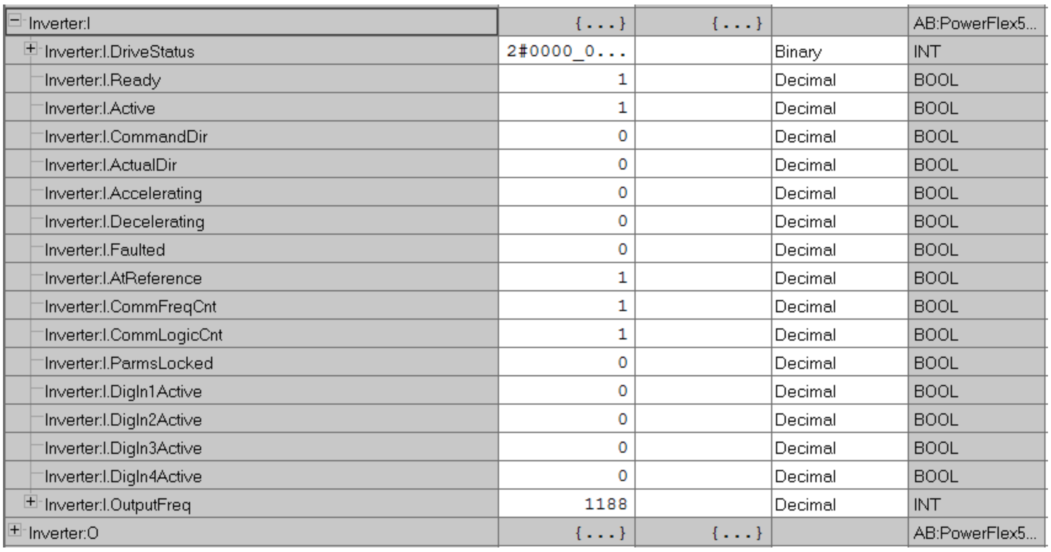


| − Inverter:O | {...} | {...} | | AB:PowerFlex5... |
|---|---|---|---|---|
| + Inverter:O.LogicCommand | 2#0000_0... | | Binary | INT |
| Inverter:O.Stop | 0 | | Decimal | BOOL |
| Inverter:O.Start | 1 | | Decimal | BOOL |
| Inverter:O.Jog | 0 | | Decimal | BOOL |
| Inverter:O.ClearFaults | 0 | | Decimal | BOOL |
| Inverter:O.Forward | 0 | | Decimal | BOOL |
| Inverter:O.Reverse | 1 | | Decimal | BOOL |
| Inverter:O.ForceKeypadCtrl | 0 | | Decimal | BOOL |
| Inverter:O.MOPIncrement | 0 | | Decimal | BOOL |
| Inverter:O.AccelRate1 | 0 | | Decimal | BOOL |
| Inverter:O.AccelRate2 | 0 | | Decimal | BOOL |
| Inverter:O.DecelRate1 | 0 | | Decimal | BOOL |
| Inverter:O.DecelRate2 | 0 | | Decimal | BOOL |
| Inverter:O.FreqSel01 | 0 | | Decimal | BOOL |
| Inverter:O.FreqSel02 | 0 | | Decimal | BOOL |
| Inverter:O.FreqSel03 | 0 | | Decimal | BOOL |
| Inverter:O.MOPDecrement | 0 | | Decimal | BOOL |
| + Inverter:O.FreqCommand | 1188 | | Decimal | INT |

(Fig. 3.97 The values of command words of drive)

As a result, the engine starts spinning at the desired speed in a negative direction because the drive is performing speed control.

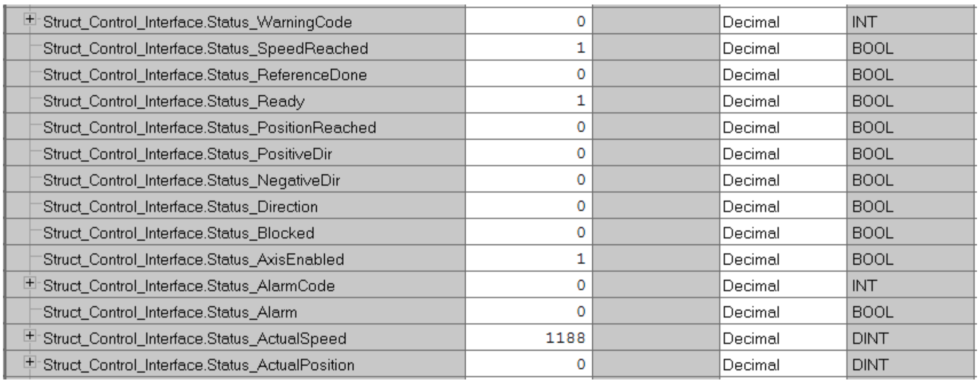
The correct functioning is demonstrated by the status words of the drive (figure 3.98).



| | | | | |
|---|---|---|---|---|
| ⊟ Inverter:I | {...} | {...} | | AB:PowerFlex5... |
| ⊞ Inverter:I.DriveStatus | 2#0000_0... | | Binary | INT |
| Inverter:I.Ready | 1 | | Decimal | BOOL |
| Inverter:I.Active | 1 | | Decimal | BOOL |
| Inverter:I.CommandDir | 0 | | Decimal | BOOL |
| Inverter:I.ActualDir | 0 | | Decimal | BOOL |
| Inverter:I.Accelerating | 0 | | Decimal | BOOL |
| Inverter:I.Decelerating | 0 | | Decimal | BOOL |
| Inverter:I.Faulted | 0 | | Decimal | BOOL |
| Inverter:I.AtReference | 1 | | Decimal | BOOL |
| Inverter:I.CommFreqCnt | 1 | | Decimal | BOOL |
| Inverter:I.CommLogicCnt | 1 | | Decimal | BOOL |
| Inverter:I.ParmsLocked | 0 | | Decimal | BOOL |
| Inverter:I.DigIn1Active | 0 | | Decimal | BOOL |
| Inverter:I.DigIn2Active | 0 | | Decimal | BOOL |
| Inverter:I.DigIn3Active | 0 | | Decimal | BOOL |
| Inverter:I.DigIn4Active | 0 | | Decimal | BOOL |
| ⊞ Inverter:I.OutputFreq | 1188 | | Decimal | INT |
| ⊞ Inverter:O | {...} | {...} | | AB:PowerFlex5... |

(Fig. 3.98 The values of status words of drive)

Drive status words must be copied to the internal structure so that the user can read the states and the block can correctly process the commands. The most important bits of the status words of the internal structure are connected to the drive input signals (figure 3.99).



| | | | | |
|---|---|---|---|---|
| ⊞ Struct_Control_Interface.Status_WarningCode | 0 | | Decimal | INT |
| Struct_Control_Interface.Status_SpeedReached | 1 | | Decimal | BOOL |
| Struct_Control_Interface.Status_ReferenceDone | 0 | | Decimal | BOOL |
| Struct_Control_Interface.Status_Ready | 1 | | Decimal | BOOL |
| Struct_Control_Interface.Status_PositionReached | 0 | | Decimal | BOOL |
| Struct_Control_Interface.Status_PositiveDir | 0 | | Decimal | BOOL |
| Struct_Control_Interface.Status_NegativeDir | 0 | | Decimal | BOOL |
| Struct_Control_Interface.Status_Direction | 0 | | Decimal | BOOL |
| Struct_Control_Interface.Status_Blocked | 0 | | Decimal | BOOL |
| Struct_Control_Interface.Status_AxisEnabled | 1 | | Decimal | BOOL |
| ⊞ Struct_Control_Interface.Status_AlarmCode | 0 | | Decimal | INT |
| Struct_Control_Interface.Status_Alarm | 0 | | Decimal | BOOL |
| ⊞ Struct_Control_Interface.Status_ActualSpeed | 1188 | | Decimal | DINT |
| ⊞ Struct_Control_Interface.Status_ActualPosition | 0 | | Decimal | DINT |

(Fig. 3.99 The values of drive input signals.)

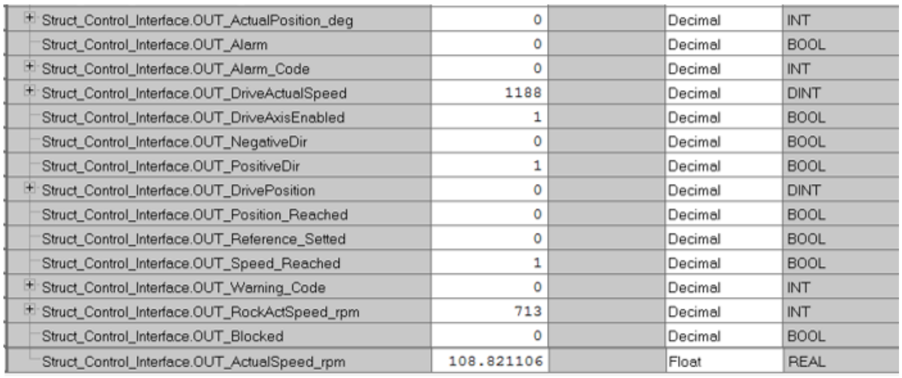
Figure 3.99 shows that the drive works correctly:

- “Struct_Control_Interface.Status_Ready” = 1;
- “Struct_Control_Interface.Status_Direction” =0 (negative direction);
- “Struct_Control_Interface.Status_Blocked” = 0;
- “Struct_Control_Interface.Status_SpeedReached” =1;

• "Struct_Control_Interface.Status_ActualSpeed" = 1188 (equation 3.10).

$$1425 * \left(\frac{50}{100}\right) = 712{,}5\ \ [rpm] \cong 11{,}88\ [Hz] = 1188\ [hundreath\ of\ Hz] \quad (3.10)$$

Finally, some drive states are transmitted to the user, because the user must be able to read the states of the main drive (figure 3.100).



| | | | | |
|---|---|---|---|---|
| + Struct_Control_Interface.OUT_ActualPosition_deg | 0 | | Decimal | INT |
| Struct_Control_Interface.OUT_Alarm | 0 | | Decimal | BOOL |
| + Struct_Control_Interface.OUT_Alarm_Code | 0 | | Decimal | INT |
| + Struct_Control_Interface.OUT_DriveActualSpeed | 1188 | | Decimal | DINT |
| Struct_Control_Interface.OUT_DriveAxisEnabled | 1 | | Decimal | BOOL |
| Struct_Control_Interface.OUT_NegativeDir | 0 | | Decimal | BOOL |
| Struct_Control_Interface.OUT_PositiveDir | 1 | | Decimal | BOOL |
| + Struct_Control_Interface.OUT_DrivePosition | 0 | | Decimal | DINT |
| Struct_Control_Interface.OUT_Position_Reached | 0 | | Decimal | BOOL |
| Struct_Control_Interface.OUT_Reference_Setted | 0 | | Decimal | BOOL |
| Struct_Control_Interface.OUT_Speed_Reached | 1 | | Decimal | BOOL |
| + Struct_Control_Interface.OUT_Warning_Code | 0 | | Decimal | INT |
| + Struct_Control_Interface.OUT_RockActSpeed_rpm | 713 | | Decimal | INT |
| Struct_Control_Interface.OUT_Blocked | 0 | | Decimal | BOOL |
| Struct_Control_Interface.OUT_ActualSpeed_rpm | 108.821106 | | Float | REAL |

(Fig. 3.100 The values of user outputs)

The third phase of the test examines the positive jog speed control, in other words the manual speed control with a positive direction. In a manner similar to Siemens tests, the speed of the jog mode is set to 20% of the maximum speed by the "R101_Control_Interface" block and is independent of the input override.

The parameters to be set in the "R101_Control_Interface" block are:

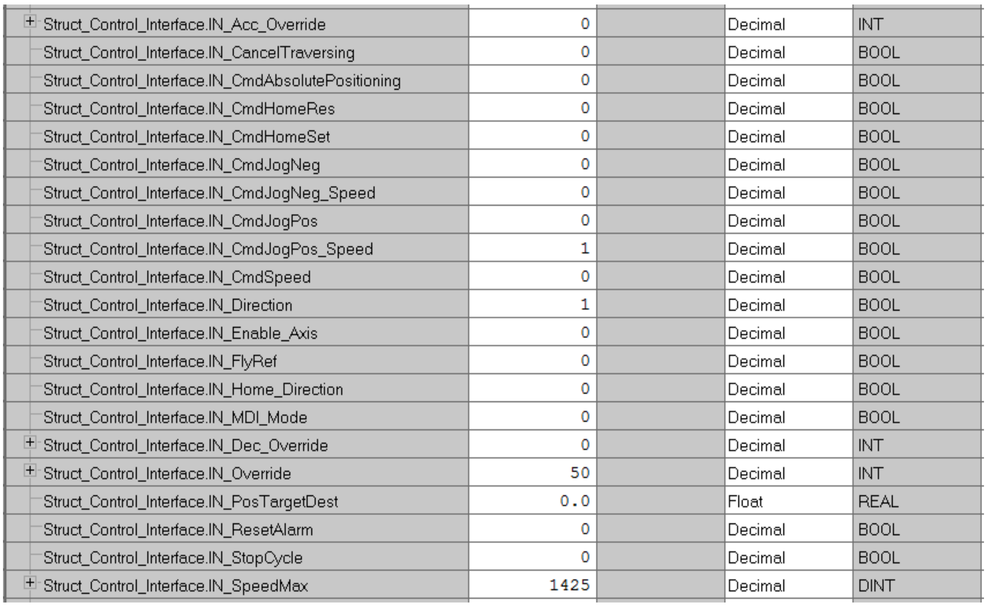
• IN_CmdSpeed = 0;

• IN_CmdJogPos_Speed = 1;

• IN_StopCycle = 0;

• IN_MaxSpeed = 1425 [rpm].

These values set the operating mode by means of user inputs. Figure 3.101 represents the current values of user inputs to perform a jog speed control with a positive direction at a speed equal to 20% of the maximum speed.



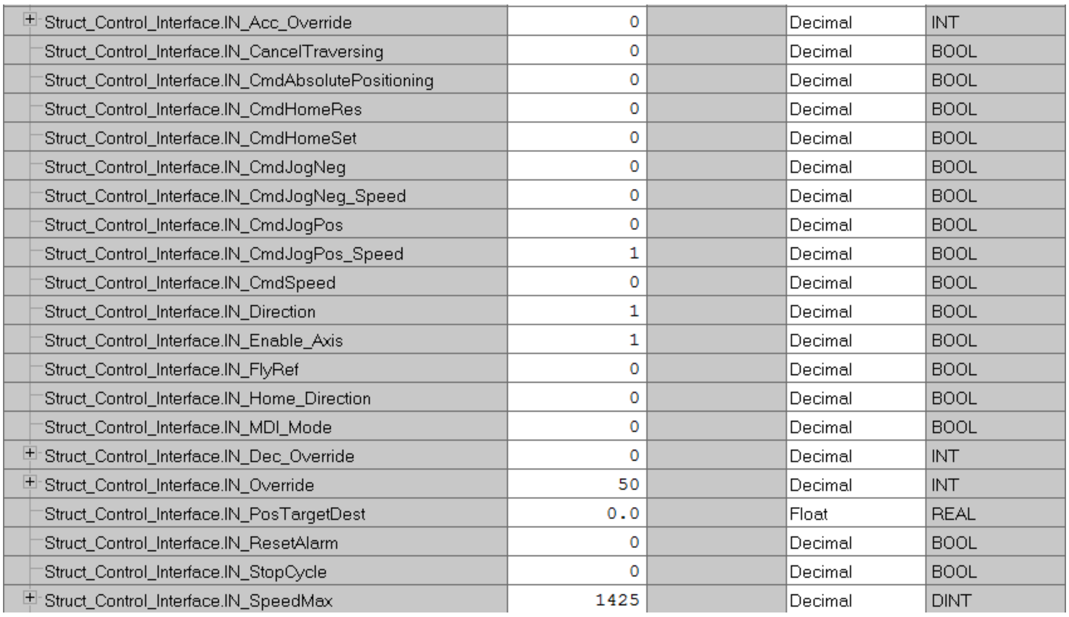
| | | | | |
|---|---|---|---|---|
| ⊞ Struct_Control_Interface.IN_Acc_Override | 0 | | Decimal | INT |
| Struct_Control_Interface.IN_CancelTraversing | 0 | | Decimal | BOOL |
| Struct_Control_Interface.IN_CmdAbsolutePositioning | 0 | | Decimal | BOOL |
| Struct_Control_Interface.IN_CmdHomeRes | 0 | | Decimal | BOOL |
| Struct_Control_Interface.IN_CmdHomeSet | 0 | | Decimal | BOOL |
| Struct_Control_Interface.IN_CmdJogNeg | 0 | | Decimal | BOOL |
| Struct_Control_Interface.IN_CmdJogNeg_Speed | 0 | | Decimal | BOOL |
| Struct_Control_Interface.IN_CmdJogPos | 0 | | Decimal | BOOL |
| Struct_Control_Interface.IN_CmdJogPos_Speed | 1 | | Decimal | BOOL |
| Struct_Control_Interface.IN_CmdSpeed | 0 | | Decimal | BOOL |
| Struct_Control_Interface.IN_Direction | 1 | | Decimal | BOOL |
| Struct_Control_Interface.IN_Enable_Axis | 0 | | Decimal | BOOL |
| Struct_Control_Interface.IN_FlyRef | 0 | | Decimal | BOOL |
| Struct_Control_Interface.IN_Home_Direction | 0 | | Decimal | BOOL |
| Struct_Control_Interface.IN_MDI_Mode | 0 | | Decimal | BOOL |
| ⊞ Struct_Control_Interface.IN_Dec_Override | 0 | | Decimal | INT |
| ⊞ Struct_Control_Interface.IN_Override | 50 | | Decimal | INT |
| Struct_Control_Interface.IN_PosTargetDest | 0.0 | | Float | REAL |
| Struct_Control_Interface.IN_ResetAlarm | 0 | | Decimal | BOOL |
| Struct_Control_Interface.IN_StopCycle | 0 | | Decimal | BOOL |
| ⊞ Struct_Control_Interface.IN_SpeedMax | 1425 | | Decimal | DINT |

(Fig. 3.101 The values of user inputs)

Also, in this case, the activation procedure is:

1. set of “IN_ResetAlarm” input (IN_ResetAlarm = 1);
2. reset of “IN_ResetAlarm” input (IN_ResetAlarm = 0);
3. set of “IN_EnableAxis” input (IN_EnableAxis = 1).

As a result, the user input values change (figure 3.102).

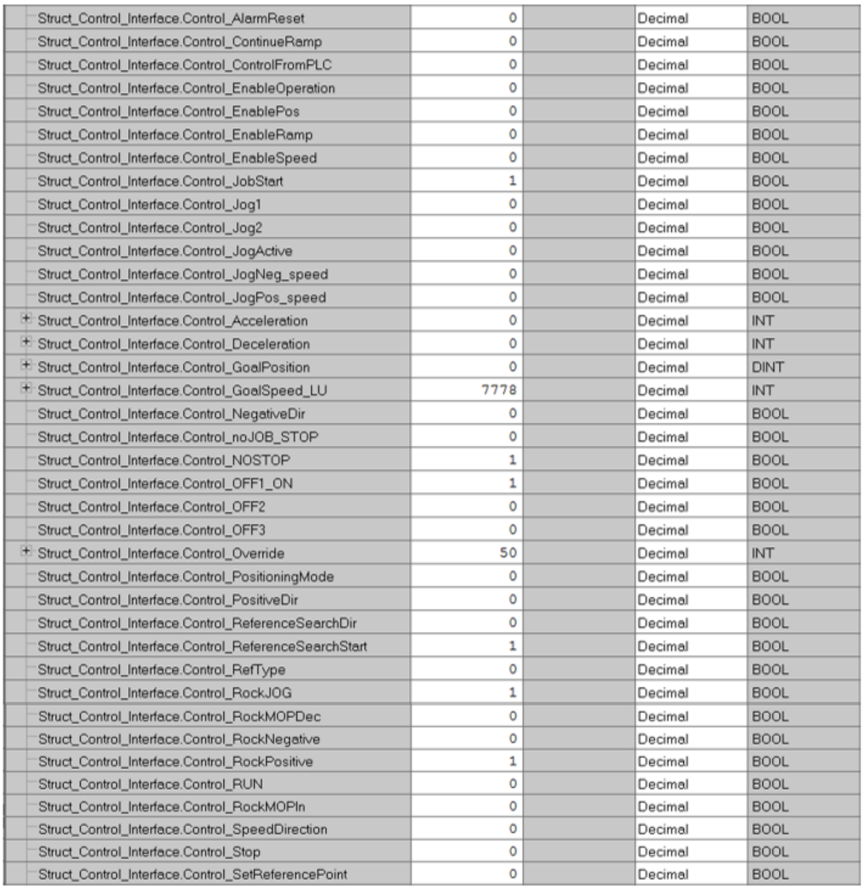


| | | | | |
|---|---|---|---|---|
| ⊞ Struct_Control_Interface.IN_Acc_Override | 0 | | Decimal | INT |
| Struct_Control_Interface.IN_CancelTraversing | 0 | | Decimal | BOOL |
| Struct_Control_Interface.IN_CmdAbsolutePositioning | 0 | | Decimal | BOOL |
| Struct_Control_Interface.IN_CmdHomeRes | 0 | | Decimal | BOOL |
| Struct_Control_Interface.IN_CmdHomeSet | 0 | | Decimal | BOOL |
| Struct_Control_Interface.IN_CmdJogNeg | 0 | | Decimal | BOOL |
| Struct_Control_Interface.IN_CmdJogNeg_Speed | 0 | | Decimal | BOOL |
| Struct_Control_Interface.IN_CmdJogPos | 0 | | Decimal | BOOL |
| Struct_Control_Interface.IN_CmdJogPos_Speed | 1 | | Decimal | BOOL |
| Struct_Control_Interface.IN_CmdSpeed | 0 | | Decimal | BOOL |
| Struct_Control_Interface.IN_Direction | 1 | | Decimal | BOOL |
| Struct_Control_Interface.IN_Enable_Axis | 1 | | Decimal | BOOL |
| Struct_Control_Interface.IN_FlyRef | 0 | | Decimal | BOOL |
| Struct_Control_Interface.IN_Home_Direction | 0 | | Decimal | BOOL |
| Struct_Control_Interface.IN_MDI_Mode | 0 | | Decimal | BOOL |
| ⊞ Struct_Control_Interface.IN_Dec_Override | 0 | | Decimal | INT |
| ⊞ Struct_Control_Interface.IN_Override | 50 | | Decimal | INT |
| Struct_Control_Interface.IN_PosTargetDest | 0.0 | | Float | REAL |
| Struct_Control_Interface.IN_ResetAlarm | 0 | | Decimal | BOOL |
| Struct_Control_Interface.IN_StopCycle | 0 | | Decimal | BOOL |
| ⊞ Struct_Control_Interface.IN_SpeedMax | 1425 | | Decimal | DINT |

(Fig. 3.102 New values of user inputs)

The R101 Control_Interface block generates the correct drive output values according to the user's commands (figure 3.103).

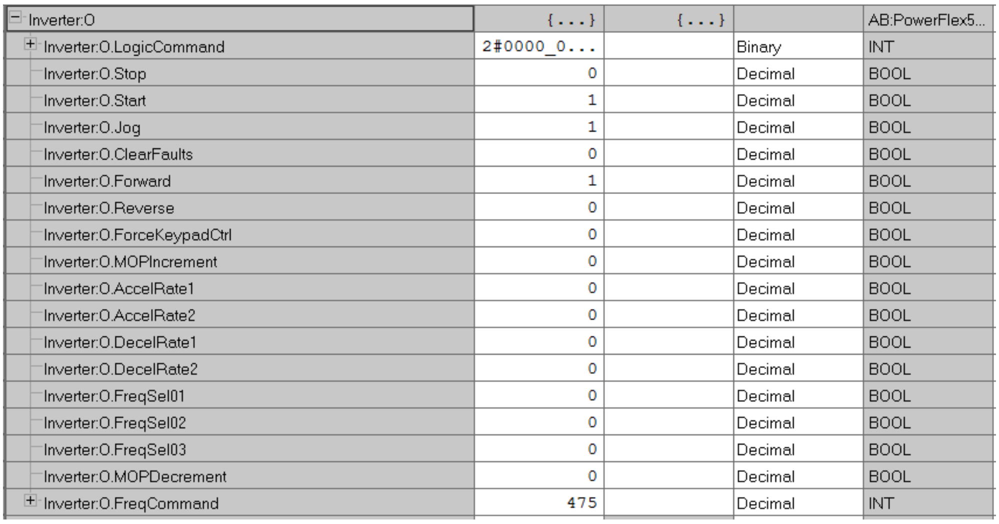
Each drive output is connected to the internal structure through the R101 Control_Interface block interface.



| | | | | |
|---|---|---|---|---|
| Struct_Control_Interface.Control_AlarmReset | 0 | | Decimal | BOOL |
| Struct_Control_Interface.Control_ContinueRamp | 0 | | Decimal | BOOL |
| Struct_Control_Interface.Control_ControlFromPLC | 0 | | Decimal | BOOL |
| Struct_Control_Interface.Control_EnableOperation | 0 | | Decimal | BOOL |
| Struct_Control_Interface.Control_EnablePos | 0 | | Decimal | BOOL |
| Struct_Control_Interface.Control_EnableRamp | 0 | | Decimal | BOOL |
| Struct_Control_Interface.Control_EnableSpeed | 0 | | Decimal | BOOL |
| Struct_Control_Interface.Control_JobStart | 1 | | Decimal | BOOL |
| Struct_Control_Interface.Control_Jog1 | 0 | | Decimal | BOOL |
| Struct_Control_Interface.Control_Jog2 | 0 | | Decimal | BOOL |
| Struct_Control_Interface.Control_JogActive | 0 | | Decimal | BOOL |
| Struct_Control_Interface.Control_JogNeg_speed | 0 | | Decimal | BOOL |
| Struct_Control_Interface.Control_JogPos_speed | 0 | | Decimal | BOOL |
| ⊞ Struct_Control_Interface.Control_Acceleration | 0 | | Decimal | INT |
| ⊞ Struct_Control_Interface.Control_Deceleration | 0 | | Decimal | INT |
| ⊞ Struct_Control_Interface.Control_GoalPosition | 0 | | Decimal | DINT |
| ⊞ Struct_Control_Interface.Control_GoalSpeed_LU | 7778 | | Decimal | INT |
| Struct_Control_Interface.Control_NegativeDir | 0 | | Decimal | BOOL |
| Struct_Control_Interface.Control_noJOB_STOP | 0 | | Decimal | BOOL |
| Struct_Control_Interface.Control_NOSTOP | 1 | | Decimal | BOOL |
| Struct_Control_Interface.Control_OFF1_ON | 1 | | Decimal | BOOL |
| Struct_Control_Interface.Control_OFF2 | 0 | | Decimal | BOOL |
| Struct_Control_Interface.Control_OFF3 | 0 | | Decimal | BOOL |
| ⊞ Struct_Control_Interface.Control_Override | 50 | | Decimal | INT |
| Struct_Control_Interface.Control_PositioningMode | 0 | | Decimal | BOOL |
| Struct_Control_Interface.Control_PositiveDir | 0 | | Decimal | BOOL |
| Struct_Control_Interface.Control_ReferenceSearchDir | 0 | | Decimal | BOOL |
| Struct_Control_Interface.Control_ReferenceSearchStart | 1 | | Decimal | BOOL |
| Struct_Control_Interface.Control_RefType | 0 | | Decimal | BOOL |
| Struct_Control_Interface.Control_RockJOG | 1 | | Decimal | BOOL |
| Struct_Control_Interface.Control_RockMOPDec | 0 | | Decimal | BOOL |
| Struct_Control_Interface.Control_RockNegative | 0 | | Decimal | BOOL |
| Struct_Control_Interface.Control_RockPositive | 1 | | Decimal | BOOL |
| Struct_Control_Interface.Control_RUN | 0 | | Decimal | BOOL |
| Struct_Control_Interface.Control_RockMOPIn | 0 | | Decimal | BOOL |
| Struct_Control_Interface.Control_SpeedDirection | 0 | | Decimal | BOOL |
| Struct_Control_Interface.Control_Stop | 0 | | Decimal | BOOL |
| Struct_Control_Interface.Control_SetReferencePoint | 0 | | Decimal | BOOL |

(Fig. 3.103 The values of drive outputs)

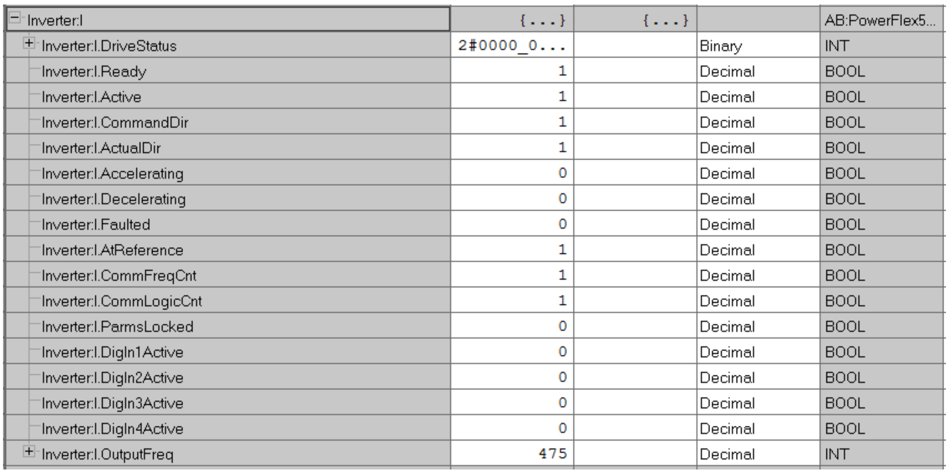
The values of the internal structure are copied into drive structure (figure 3.104).



| ⊟ Inverter:O | {...} | {...} | | AB:PowerFlex5... |
|---|---|---|---|---|
| ⊞ Inverter:O.LogicCommand | 2#0000_0... | | Binary | INT |
| Inverter:O.Stop | 0 | | Decimal | BOOL |
| Inverter:O.Start | 1 | | Decimal | BOOL |
| Inverter:O.Jog | 1 | | Decimal | BOOL |
| Inverter:O.ClearFaults | 0 | | Decimal | BOOL |
| Inverter:O.Forward | 1 | | Decimal | BOOL |
| Inverter:O.Reverse | 0 | | Decimal | BOOL |
| Inverter:O.ForceKeypadCtrl | 0 | | Decimal | BOOL |
| Inverter:O.MOPIncrement | 0 | | Decimal | BOOL |
| Inverter:O.AccelRate1 | 0 | | Decimal | BOOL |
| Inverter:O.AccelRate2 | 0 | | Decimal | BOOL |
| Inverter:O.DecelRate1 | 0 | | Decimal | BOOL |
| Inverter:O.DecelRate2 | 0 | | Decimal | BOOL |
| Inverter:O.FreqSel01 | 0 | | Decimal | BOOL |
| Inverter:O.FreqSel02 | 0 | | Decimal | BOOL |
| Inverter:O.FreqSel03 | 0 | | Decimal | BOOL |
| Inverter:O.MOPDecrement | 0 | | Decimal | BOOL |
| ⊞ Inverter:O.FreqCommand | 475 | | Decimal | INT |

(Fig. 3.104 The values of command words of drive)

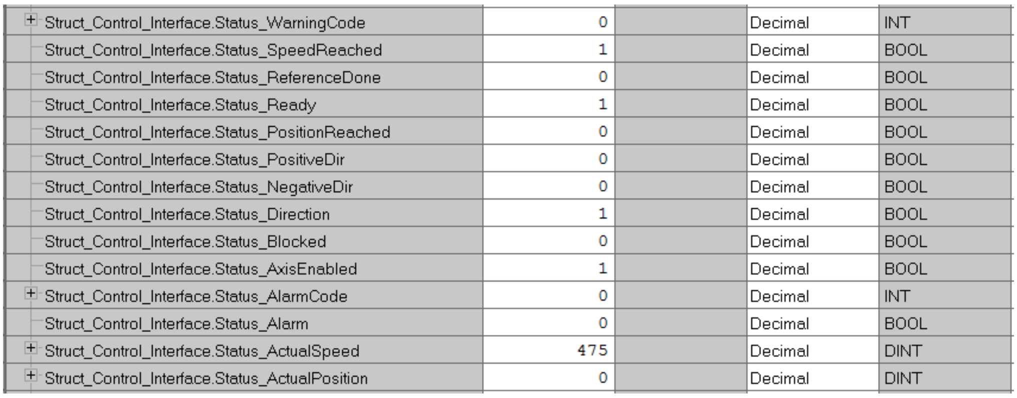
As a result, the engine starts spinning at a speed of 285 [rpm] in a positive direction. The correct functioning is shown by the drive status words (figure 3.105).



| ⊟ Inverter:I | {...} | {...} | | AB:PowerFlex5... |
|---|---|---|---|---|
| ⊞ Inverter:I.DriveStatus | 2#0000_0... | | Binary | INT |
| Inverter:I.Ready | 1 | | Decimal | BOOL |
| Inverter:I.Active | 1 | | Decimal | BOOL |
| Inverter:I.CommandDir | 1 | | Decimal | BOOL |
| Inverter:I.ActualDir | 1 | | Decimal | BOOL |
| Inverter:I.Accelerating | 0 | | Decimal | BOOL |
| Inverter:I.Decelerating | 0 | | Decimal | BOOL |
| Inverter:I.Faulted | 0 | | Decimal | BOOL |
| Inverter:I.AtReference | 1 | | Decimal | BOOL |
| Inverter:I.CommFreqCnt | 1 | | Decimal | BOOL |
| Inverter:I.CommLogicCnt | 1 | | Decimal | BOOL |
| Inverter:I.ParmsLocked | 0 | | Decimal | BOOL |
| Inverter:I.DigIn1Active | 0 | | Decimal | BOOL |
| Inverter:I.DigIn2Active | 0 | | Decimal | BOOL |
| Inverter:I.DigIn3Active | 0 | | Decimal | BOOL |
| Inverter:I.DigIn4Active | 0 | | Decimal | BOOL |
| ⊞ Inverter:I.OutputFreq | 475 | | Decimal | INT |

(Fig. 3.105 The values of status words of drive)

Also in this case, the most important bits of the status words of the internal structure are transmitted to the drive input signals (figure 3.106).



| | | | | |
|---|---|---|---|---|
| ⊞ Struct_Control_Interface.Status_WarningCode | 0 | | Decimal | INT |
| Struct_Control_Interface.Status_SpeedReached | 1 | | Decimal | BOOL |
| Struct_Control_Interface.Status_ReferenceDone | 0 | | Decimal | BOOL |
| Struct_Control_Interface.Status_Ready | 1 | | Decimal | BOOL |
| Struct_Control_Interface.Status_PositionReached | 0 | | Decimal | BOOL |
| Struct_Control_Interface.Status_PositiveDir | 0 | | Decimal | BOOL |
| Struct_Control_Interface.Status_NegativeDir | 0 | | Decimal | BOOL |
| Struct_Control_Interface.Status_Direction | 1 | | Decimal | BOOL |
| Struct_Control_Interface.Status_Blocked | 0 | | Decimal | BOOL |
| Struct_Control_Interface.Status_AxisEnabled | 1 | | Decimal | BOOL |
| ⊞ Struct_Control_Interface.Status_AlarmCode | 0 | | Decimal | INT |
| Struct_Control_Interface.Status_Alarm | 0 | | Decimal | BOOL |
| ⊞ Struct_Control_Interface.Status_ActualSpeed | 475 | | Decimal | DINT |
| ⊞ Struct_Control_Interface.Status_ActualPosition | 0 | | Decimal | DINT |

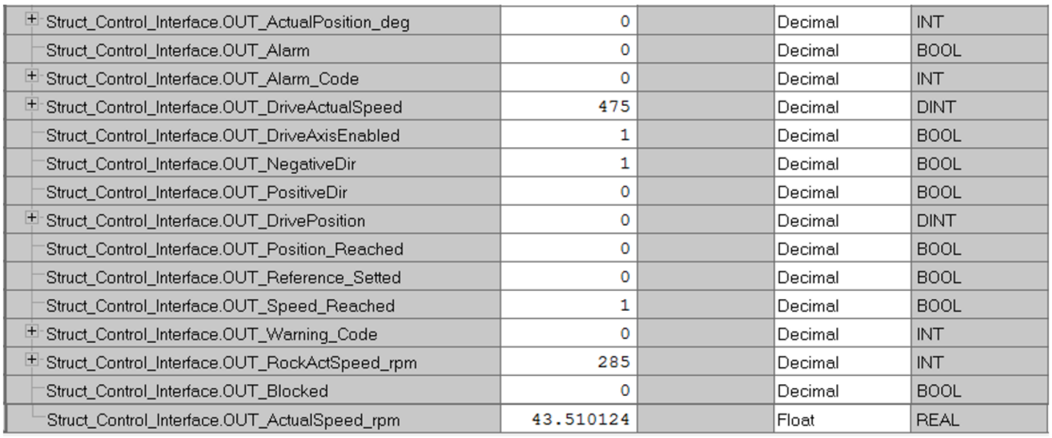
(Fig. 3.106 The values of drive input signals.)

The drive works correctly (figure 3.106):

- “Struct_Control_Interface.Status_Ready” = 1;
- “Struct_Control_Interface.Status_Direction” =1 (positive direction);
- “Struct_Control_Interface.Status_Blocked” = 0;
- “Struct_Control_Interface.Status_SpeedReached” =1;
- “Struct_Control_Interface.Status_ActualSpeed” = 475 (equation 3.11).

$$1425 * \left(\frac{20}{100}\right) = 285\ \ [rpm] = 4{,}75\ [Hz] = 475\ [hundreath\ of\ Hz] \qquad (3.11)$$

Finally, the main drive states are transmitted to the user by means of user output variables (figure 3.107).



| | | | | |
|---|---|---|---|---|
| Struct_Control_Interface.OUT_ActualPosition_deg | 0 | | Decimal | INT |
| Struct_Control_Interface.OUT_Alarm | 0 | | Decimal | BOOL |
| Struct_Control_Interface.OUT_Alarm_Code | 0 | | Decimal | INT |
| Struct_Control_Interface.OUT_DriveActualSpeed | 475 | | Decimal | DINT |
| Struct_Control_Interface.OUT_DriveAxisEnabled | 1 | | Decimal | BOOL |
| Struct_Control_Interface.OUT_NegativeDir | 1 | | Decimal | BOOL |
| Struct_Control_Interface.OUT_PositiveDir | 0 | | Decimal | BOOL |
| Struct_Control_Interface.OUT_DrivePosition | 0 | | Decimal | DINT |
| Struct_Control_Interface.OUT_Position_Reached | 0 | | Decimal | BOOL |
| Struct_Control_Interface.OUT_Reference_Setted | 0 | | Decimal | BOOL |
| Struct_Control_Interface.OUT_Speed_Reached | 1 | | Decimal | BOOL |
| Struct_Control_Interface.OUT_Warning_Code | 0 | | Decimal | INT |
| Struct_Control_Interface.OUT_RockActSpeed_rpm | 285 | | Decimal | INT |
| Struct_Control_Interface.OUT_Blocked | 0 | | Decimal | BOOL |
| Struct_Control_Interface.OUT_ActualSpeed_rpm | 43.510124 | | Float | REAL |

(Fig. 3.107 The values of user outputs)

The fourth phase of the test consist of the analysis of the negative jog speed control. Negative jog speed mode consists of a manual command with a negative direction at a speed equal to 20% of the maximum speed.

The parameters to be set in the "Control_Interface" block are:

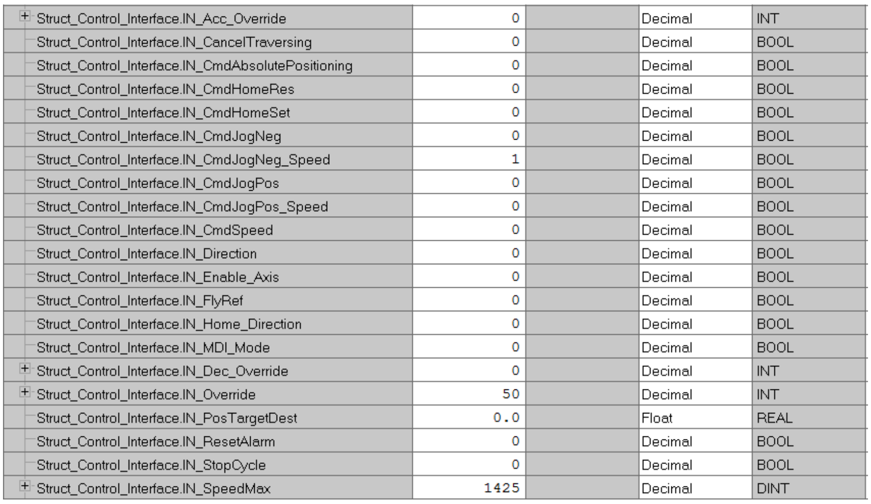
• “IN_CmdSpeed = 0”;

• “IN_CmdJogNeg_Speed = 1”;

• “IN_StopCycle = 0”;

• “IN_MaxSpeed” = 1425 [rpm]”;

As a result, initial user inputs values are the values shown in the figure 3.108.



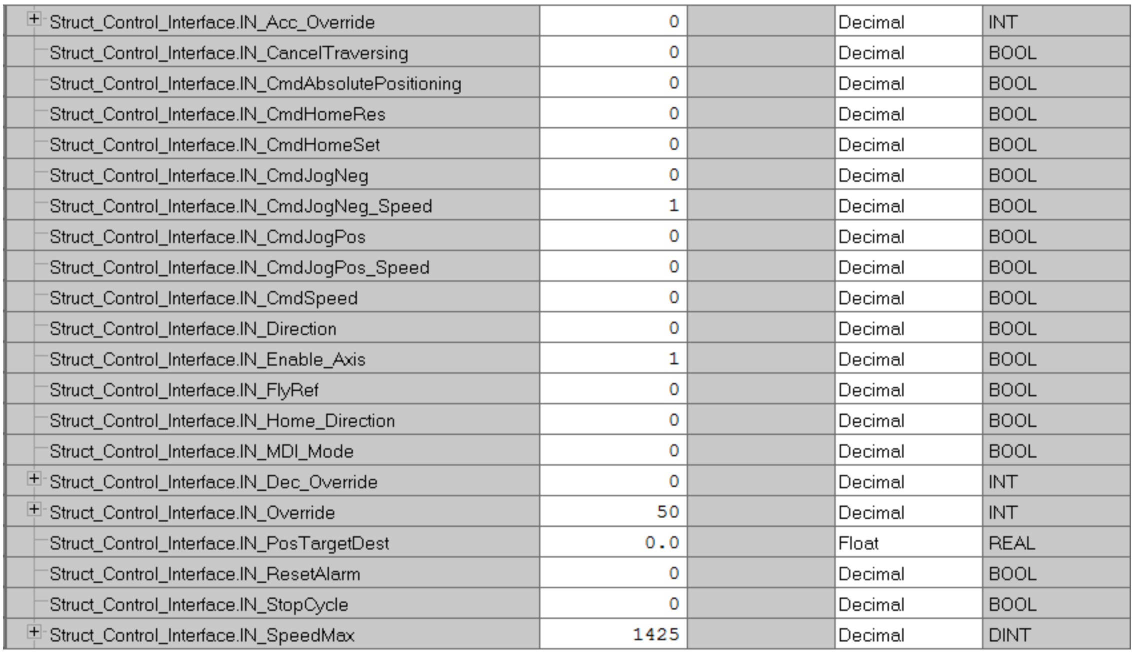
| | | | | |
|---|---|---|---|---|
| Struct_Control_Interface.IN_Acc_Override | 0 | | Decimal | INT |
| Struct_Control_Interface.IN_CancelTraversing | 0 | | Decimal | BOOL |
| Struct_Control_Interface.IN_CmdAbsolutePositioning | 0 | | Decimal | BOOL |
| Struct_Control_Interface.IN_CmdHomeRes | 0 | | Decimal | BOOL |
| Struct_Control_Interface.IN_CmdHomeSet | 0 | | Decimal | BOOL |
| Struct_Control_Interface.IN_CmdJogNeg | 0 | | Decimal | BOOL |
| Struct_Control_Interface.IN_CmdJogNeg_Speed | 1 | | Decimal | BOOL |
| Struct_Control_Interface.IN_CmdJogPos | 0 | | Decimal | BOOL |
| Struct_Control_Interface.IN_CmdJogPos_Speed | 0 | | Decimal | BOOL |
| Struct_Control_Interface.IN_CmdSpeed | 0 | | Decimal | BOOL |
| Struct_Control_Interface.IN_Direction | 0 | | Decimal | BOOL |
| Struct_Control_Interface.IN_Enable_Axis | 0 | | Decimal | BOOL |
| Struct_Control_Interface.IN_FlyRef | 0 | | Decimal | BOOL |
| Struct_Control_Interface.IN_Home_Direction | 0 | | Decimal | BOOL |
| Struct_Control_Interface.IN_MDI_Mode | 0 | | Decimal | BOOL |
| Struct_Control_Interface.IN_Dec_Override | 0 | | Decimal | INT |
| Struct_Control_Interface.IN_Override | 50 | | Decimal | INT |
| Struct_Control_Interface.IN_PosTargetDest | 0.0 | | Float | REAL |
| Struct_Control_Interface.IN_ResetAlarm | 0 | | Decimal | BOOL |
| Struct_Control_Interface.IN_StopCycle | 0 | | Decimal | BOOL |
| Struct_Control_Interface.IN_SpeedMax | 1425 | | Decimal | DINT |

(Fig. 3.108 The values of user inputs)

The activation procedure is repeated:

1. set of “IN_ResetAlarm” input (IN_ResetAlarm = 1);
2. reset of “IN_ResetAlarm” input (IN_ResetAlarm = 0);
3. set of “IN_EnableAxis” input (IN_EnableAxis = 1).

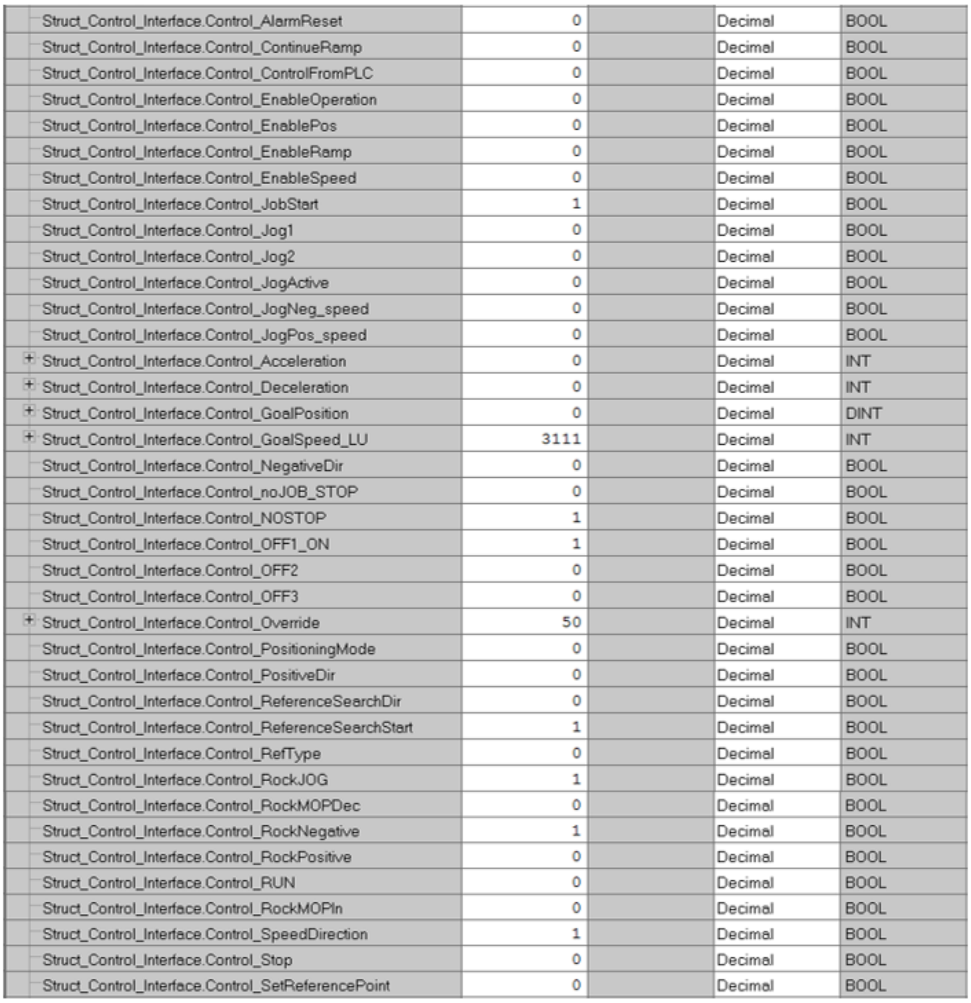
As a result, user input values change (figure 3.109).



| | | | | |
|---|---|---|---|---|
| Struct_Control_Interface.IN_Acc_Override | 0 | | Decimal | INT |
| Struct_Control_Interface.IN_CancelTraversing | 0 | | Decimal | BOOL |
| Struct_Control_Interface.IN_CmdAbsolutePositioning | 0 | | Decimal | BOOL |
| Struct_Control_Interface.IN_CmdHomeRes | 0 | | Decimal | BOOL |
| Struct_Control_Interface.IN_CmdHomeSet | 0 | | Decimal | BOOL |
| Struct_Control_Interface.IN_CmdJogNeg | 0 | | Decimal | BOOL |
| Struct_Control_Interface.IN_CmdJogNeg_Speed | 1 | | Decimal | BOOL |
| Struct_Control_Interface.IN_CmdJogPos | 0 | | Decimal | BOOL |
| Struct_Control_Interface.IN_CmdJogPos_Speed | 0 | | Decimal | BOOL |
| Struct_Control_Interface.IN_CmdSpeed | 0 | | Decimal | BOOL |
| Struct_Control_Interface.IN_Direction | 0 | | Decimal | BOOL |
| Struct_Control_Interface.IN_Enable_Axis | 1 | | Decimal | BOOL |
| Struct_Control_Interface.IN_FlyRef | 0 | | Decimal | BOOL |
| Struct_Control_Interface.IN_Home_Direction | 0 | | Decimal | BOOL |
| Struct_Control_Interface.IN_MDI_Mode | 0 | | Decimal | BOOL |
| Struct_Control_Interface.IN_Dec_Override | 0 | | Decimal | INT |
| Struct_Control_Interface.IN_Override | 50 | | Decimal | INT |
| Struct_Control_Interface.IN_PosTargetDest | 0.0 | | Float | REAL |
| Struct_Control_Interface.IN_ResetAlarm | 0 | | Decimal | BOOL |
| Struct_Control_Interface.IN_StopCycle | 0 | | Decimal | BOOL |
| Struct_Control_Interface.IN_SpeedMax | 1425 | | Decimal | DINT |

(Fig. 3.109 New values of user inputs)

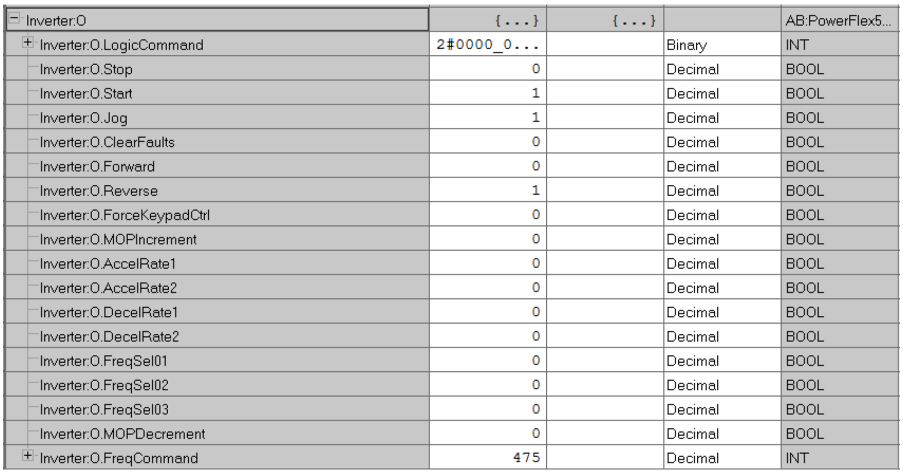
The calculation of the drive output values is performed by the R101 Control_Interface block and these values are transmitted to the internal structure (figure 3.110).



| | | | | |
|---|---|---|---|---|
| Struct_Control_Interface.Control_AlarmReset | 0 | | Decimal | BOOL |
| Struct_Control_Interface.Control_ContinueRamp | 0 | | Decimal | BOOL |
| Struct_Control_Interface.Control_ControlFromPLC | 0 | | Decimal | BOOL |
| Struct_Control_Interface.Control_EnableOperation | 0 | | Decimal | BOOL |
| Struct_Control_Interface.Control_EnablePos | 0 | | Decimal | BOOL |
| Struct_Control_Interface.Control_EnableRamp | 0 | | Decimal | BOOL |
| Struct_Control_Interface.Control_EnableSpeed | 0 | | Decimal | BOOL |
| Struct_Control_Interface.Control_JobStart | 1 | | Decimal | BOOL |
| Struct_Control_Interface.Control_Jog1 | 0 | | Decimal | BOOL |
| Struct_Control_Interface.Control_Jog2 | 0 | | Decimal | BOOL |
| Struct_Control_Interface.Control_JogActive | 0 | | Decimal | BOOL |
| Struct_Control_Interface.Control_JogNeg_speed | 0 | | Decimal | BOOL |
| Struct_Control_Interface.Control_JogPos_speed | 0 | | Decimal | BOOL |
| + Struct_Control_Interface.Control_Acceleration | 0 | | Decimal | INT |
| + Struct_Control_Interface.Control_Deceleration | 0 | | Decimal | INT |
| + Struct_Control_Interface.Control_GoalPosition | 0 | | Decimal | DINT |
| + Struct_Control_Interface.Control_GoalSpeed_LU | 3111 | | Decimal | INT |
| Struct_Control_Interface.Control_NegativeDir | 0 | | Decimal | BOOL |
| Struct_Control_Interface.Control_noJOB_STOP | 0 | | Decimal | BOOL |
| Struct_Control_Interface.Control_NOSTOP | 1 | | Decimal | BOOL |
| Struct_Control_Interface.Control_OFF1_ON | 1 | | Decimal | BOOL |
| Struct_Control_Interface.Control_OFF2 | 0 | | Decimal | BOOL |
| Struct_Control_Interface.Control_OFF3 | 0 | | Decimal | BOOL |
| + Struct_Control_Interface.Control_Override | 50 | | Decimal | INT |
| Struct_Control_Interface.Control_PositioningMode | 0 | | Decimal | BOOL |
| Struct_Control_Interface.Control_PositiveDir | 0 | | Decimal | BOOL |
| Struct_Control_Interface.Control_ReferenceSearchDir | 0 | | Decimal | BOOL |
| Struct_Control_Interface.Control_ReferenceSearchStart | 1 | | Decimal | BOOL |
| Struct_Control_Interface.Control_RefType | 0 | | Decimal | BOOL |
| Struct_Control_Interface.Control_RockJOG | 1 | | Decimal | BOOL |
| Struct_Control_Interface.Control_RockMOPDec | 0 | | Decimal | BOOL |
| Struct_Control_Interface.Control_RockNegative | 1 | | Decimal | BOOL |
| Struct_Control_Interface.Control_RockPositive | 0 | | Decimal | BOOL |
| Struct_Control_Interface.Control_RUN | 0 | | Decimal | BOOL |
| Struct_Control_Interface.Control_RockMOPIn | 0 | | Decimal | BOOL |
| Struct_Control_Interface.Control_SpeedDirection | 1 | | Decimal | BOOL |
| Struct_Control_Interface.Control_Stop | 0 | | Decimal | BOOL |
| Struct_Control_Interface.Control_SetReferencePoint | 0 | | Decimal | BOOL |

(Fig. 3.110 The values of drive outputs)

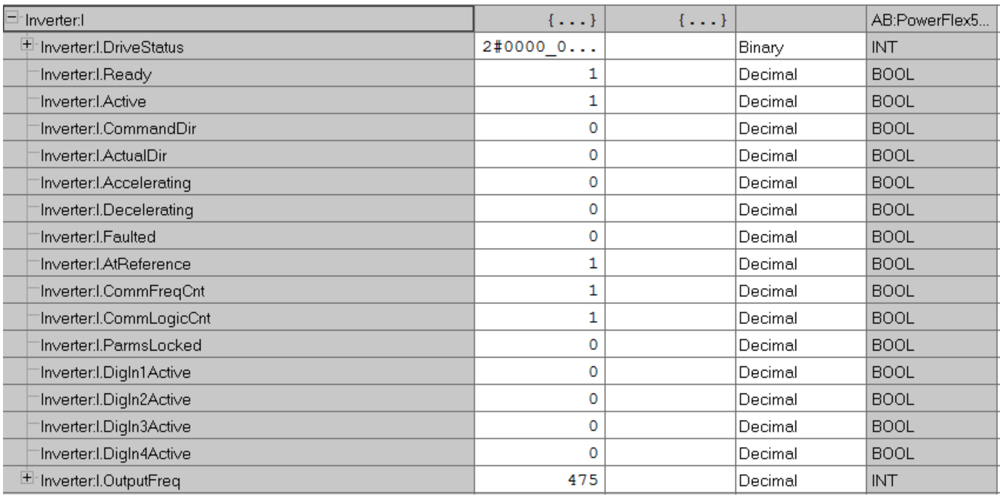
The values of the internal structure update the values of the drive structure by means of the writing function (figure 3.111).



| | | | | |
|---|---|---|---|---|
| − Inverter:O | {...} | {...} | | AB:PowerFlex5... |
| + Inverter:O.LogicCommand | 2#0000_0... | | Binary | INT |
| Inverter:O.Stop | 0 | | Decimal | BOOL |
| Inverter:O.Start | 1 | | Decimal | BOOL |
| Inverter:O.Jog | 1 | | Decimal | BOOL |
| Inverter:O.ClearFaults | 0 | | Decimal | BOOL |
| Inverter:O.Forward | 0 | | Decimal | BOOL |
| Inverter:O.Reverse | 1 | | Decimal | BOOL |
| Inverter:O.ForceKeypadCtrl | 0 | | Decimal | BOOL |
| Inverter:O.MOPIncrement | 0 | | Decimal | BOOL |
| Inverter:O.AccelRate1 | 0 | | Decimal | BOOL |
| Inverter:O.AccelRate2 | 0 | | Decimal | BOOL |
| Inverter:O.DecelRate1 | 0 | | Decimal | BOOL |
| Inverter:O.DecelRate2 | 0 | | Decimal | BOOL |
| Inverter:O.FreqSel01 | 0 | | Decimal | BOOL |
| Inverter:O.FreqSel02 | 0 | | Decimal | BOOL |
| Inverter:O.FreqSel03 | 0 | | Decimal | BOOL |
| Inverter:O.MOPDecrement | 0 | | Decimal | BOOL |
| + Inverter:O.FreqCommand | 475 | | Decimal | INT |

(Fig. 3.111 The values of command words of drive)

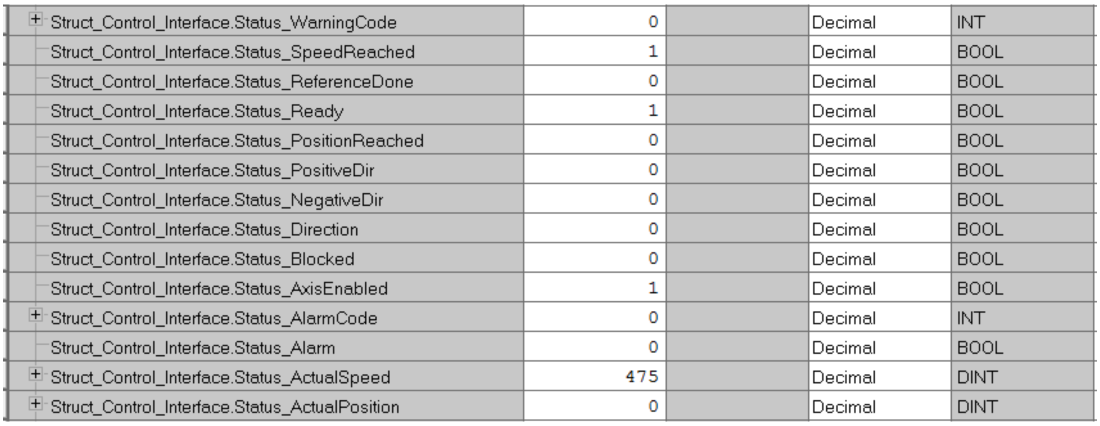
As a result, the engine starts spinning at -285 [rpm] because it has a negative direction. The correct functioning is demonstrated by the status words of the drive (figure 3.112).



| ⊟ Inverter:I | {...} | {...} | | AB:PowerFlex5... |
|---|---|---|---|---|
| ⊞ Inverter:I.DriveStatus | 2#0000_0... | | Binary | INT |
| Inverter:I.Ready | 1 | | Decimal | BOOL |
| Inverter:I.Active | 1 | | Decimal | BOOL |
| Inverter:I.CommandDir | 0 | | Decimal | BOOL |
| Inverter:I.ActualDir | 0 | | Decimal | BOOL |
| Inverter:I.Accelerating | 0 | | Decimal | BOOL |
| Inverter:I.Decelerating | 0 | | Decimal | BOOL |
| Inverter:I.Faulted | 0 | | Decimal | BOOL |
| Inverter:I.AtReference | 1 | | Decimal | BOOL |
| Inverter:I.CommFreqCnt | 1 | | Decimal | BOOL |
| Inverter:I.CommLogicCnt | 1 | | Decimal | BOOL |
| Inverter:I.ParmsLocked | 0 | | Decimal | BOOL |
| Inverter:I.DigIn1Active | 0 | | Decimal | BOOL |
| Inverter:I.DigIn2Active | 0 | | Decimal | BOOL |
| Inverter:I.DigIn3Active | 0 | | Decimal | BOOL |
| Inverter:I.DigIn4Active | 0 | | Decimal | BOOL |
| ⊞ Inverter:I.OutputFreq | 475 | | Decimal | INT |

(Fig. 3.112 The values of status words of drive)

The main states of the drive are copied to the internal structure by means of the read function (figure 3.113).



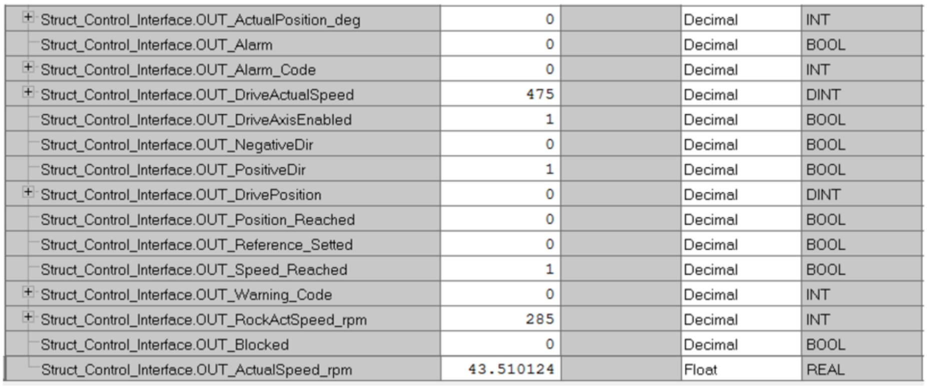
| | | | | |
|---|---|---|---|---|
| ⊞ Struct_Control_Interface.Status_WarningCode | 0 | | Decimal | INT |
| Struct_Control_Interface.Status_SpeedReached | 1 | | Decimal | BOOL |
| Struct_Control_Interface.Status_ReferenceDone | 0 | | Decimal | BOOL |
| Struct_Control_Interface.Status_Ready | 1 | | Decimal | BOOL |
| Struct_Control_Interface.Status_PositionReached | 0 | | Decimal | BOOL |
| Struct_Control_Interface.Status_PositiveDir | 0 | | Decimal | BOOL |
| Struct_Control_Interface.Status_NegativeDir | 0 | | Decimal | BOOL |
| Struct_Control_Interface.Status_Direction | 0 | | Decimal | BOOL |
| Struct_Control_Interface.Status_Blocked | 0 | | Decimal | BOOL |
| Struct_Control_Interface.Status_AxisEnabled | 1 | | Decimal | BOOL |
| ⊞ Struct_Control_Interface.Status_AlarmCode | 0 | | Decimal | INT |
| Struct_Control_Interface.Status_Alarm | 0 | | Decimal | BOOL |
| ⊞ Struct_Control_Interface.Status_ActualSpeed | 475 | | Decimal | DINT |
| ⊞ Struct_Control_Interface.Status_ActualPosition | 0 | | Decimal | DINT |

(Fig. 3.113 The values of drive input signals.)

The operation of the drive is correct (figure 3.113):

- "Struct_Control_Interface.Status_Ready" = 1;
- "Struct_Control_Interface.Status_Direction" =0 (negative direction);
- "Struct_Control_Interface.Status_Blocked" = 0;
- "Struct_Control_Interface.Status_SpeedReached" =1;
- "Struct_Control_Interface.Status_ActualSpeed" = 475.

Finally, some drive states are transmitted to the user by means of user output variables (figure 3.114).



| | | | | |
|---|---|---|---|---|
| Struct_Control_Interface.OUT_ActualPosition_deg | 0 | | Decimal | INT |
| Struct_Control_Interface.OUT_Alarm | 0 | | Decimal | BOOL |
| Struct_Control_Interface.OUT_Alarm_Code | 0 | | Decimal | INT |
| Struct_Control_Interface.OUT_DriveActualSpeed | 475 | | Decimal | DINT |
| Struct_Control_Interface.OUT_DriveAxisEnabled | 1 | | Decimal | BOOL |
| Struct_Control_Interface.OUT_NegativeDir | 0 | | Decimal | BOOL |
| Struct_Control_Interface.OUT_PositiveDir | 1 | | Decimal | BOOL |
| Struct_Control_Interface.OUT_DrivePosition | 0 | | Decimal | DINT |
| Struct_Control_Interface.OUT_Position_Reached | 0 | | Decimal | BOOL |
| Struct_Control_Interface.OUT_Reference_Setted | 0 | | Decimal | BOOL |
| Struct_Control_Interface.OUT_Speed_Reached | 1 | | Decimal | BOOL |
| Struct_Control_Interface.OUT_Warning_Code | 0 | | Decimal | INT |
| Struct_Control_Interface.OUT_RockActSpeed_rpm | 285 | | Decimal | INT |
| Struct_Control_Interface.OUT_Blocked | 0 | | Decimal | BOOL |
| Struct_Control_Interface.OUT_ActualSpeed_rpm | 43.510124 | | Float | REAL |

(Fig. 3.114 The values of user outputs)

In conclusion, the test showed that the communication between PLC and Rockwell Automation inverter drive by means of the drive structure of PowerFlex 525 was performed correctly and all functions of the program were executed correctly and completely.

# CONCLUSION

Current industrial evolution is characterized by the transition from Industry 3.0 to Industry 4.0. Programmable logic controllers are the main devices that have allowed this evolution, because they permit a complete control of the industrial plants. The programmable logic controllers perform a total control of an industrial system by using communication interfaces. Specifically, the communication interface between programmable logic controller and drive is extremely important for electric motors control.

This thesis has analyzed the communication between PLC and drive, and it has implemented an interface that is able to communicate programmable logic controllers and drives independently of type and manufacturer.

The experimental work has been articulated in 4 sections.

The first section has been the analysis of the communication between a Siemens inverter and a Siemens PLC by means of the Standard Telegram 1. This telegram is the communication structure that allows the information exchange between Siemens drive and Siemens programmable logic controllers. During this test, the tested modes were positive speed control, negative speed control, positive jog mode and negative jog mode. The test results were right, because all the modes worked properly. Specifically, the positive speed control and the negative speed control have been performed correctly. In the first case the rotor rotated at the wanted speed with a positive direction, while in the second case the rotor rotation was with a negative direction and at the wanted speed. Therefore, in both cases the communication interface worked correctly. Furthermore, during the positive jog mode test the motor rotor rotated at a speed of 20% of the maximum speed with a positive direction confirming the expectations. The information exchange between inverter drive and PLC also functioned during the negative jog mode test. Indeed, the engine rotated at a speed of 20% of the maximum speed with a negative direction.

The second section has been an analysis of communication between a Siemens inverter and Siemens PLC using the Siemens Telegram 352. In a manner similar to Standard Telegram 1, this telegram is the communication structure that allows the information exchange between Siemens drive and Siemens programmable logic controllers. The difference between the Standard Telegram 1 and the Siemens Telegram 352 is the amount of information that are exchanged. Indeed, the number of information exchanged through the Siemens Telegram 352 is greater than the number of data that are exchanged with Standard Telegram 1. The tested modes were the same as those tested for the Standard Telegram 1: positive speed control, negative speed control, positive jog mode and negative jog mode. Particularly, the largest number of transmissible information has been verified. Again, the test results were great, because all modes worked correctly and the possible error code was read through a special interface variable.

The third section has analyzed the communication between a Siemens servo drive and a Siemens PLC by means of the Standard Telegram 111. This telegram is the communication structure that allows information exchange between Siemens servo drive and Siemens PLC. At this stage, all required operating modes have been tested. The tested modes are positive jog mode, negative jog mode, homing mode, positive absolute positioning, negative absolute positioning, positive reference search mode and negative reference search mode. The results have showed a correct communication exchange. Particularly, during positive jog mode test, the servo drive performed this mode correctly. Indeed, the electric motor turned in manual mode with a positive direction. During negative jog mode test, a manual rotation with a negative direction was performed by the electric motor, confirming the expectations. The homing mode test has been performed correctly and the servo drive has set the current position of the rotor as its starting position. The main activity of a servo drive is the positioning. Tests about positive absolute positioning and negative absolute positioning are fundamental. Both tests have been completed and they showed proper functioning of the servo drive. Particularly, during the positive absolute positioning the servo drive rotated the rotor in a positive direction until it obtained the desired position, while during the negative absolute positioning the servo drive led the rotor to wanted position by rotating it in negative direction. The last tests performed about servo drive were positive reference search mode and negative reference search mode. Both tests have been performed correctly because during the first test the rotor started turning in a positive direction to search a suitable reference point, while in the second test the rotor started to turn in a negative direction in search of a reference point.

The last section has been the analysis of the communication between a Rockwell Automation inverter and a Rockwell Automation PLC by means of the PowerFlex 525 data type. This structure allows the information exchange between PowerFlex 525 inverter and programmable logic controllers. During this test, the tested modes were positive speed control, negative speed control, positive jog mode and negative jog mode. During the test about positive speed control, the rotor rotation was with a positive direction and at the desired speed, while during the test about negative speed control the electric motor rotor rotated at the wanted speed with negative direction. Therefore, in both cases the communication interface worked and modes were correct. In a manner similar to Siemens test, the positive jog mode test tested the manual mode with positive direction. Also in this case, the information exchange between inverter drive and PLC has worked perfectly. Indeed, the engine response was a positive rotation at a speed of 20% of the maximum speed. The last test performed was negative jog mode. During this activity, the engine rotated at a speed of 20% of the maximum speed with a negative direction, confirming the expectations.

In conclusion, the communication interface block that I have implemented is able to adapt to various types of communication structures, independently of the drive type and the drive manufacturers.

The possible future development of my research could be the converter implementation that can easily convert the interface block from one programming environment to another. Therefore, if programmer wants to switch from one company's programming software to another company's programming software, he should not copy it. This converter would increase the flexibility of the interface block.

# APPENDIX A

Siemens structure of Telegram 111.

Control words.

| PZD | Assignment of the process data |
|---|---|
| PZD 1 | Control word 1 |
| PZD 2 | EPosSTW 1 |
| PZD 3 | EPosSTW 2 |
| PZD 4 | Control word 2 |
| PZD 5 | Velocity override for all operating modes (4000HEX = 100%) |
| PZD 6 | Position setpoint in [LU] for direct setpoint specification / MDI mode |
| PZD 7 | |
| PZD 8 | Velocity setpoint in the MDI mode |
| PZD 9 | |
| PZD 10 | Acceleration override for direct setpoint input / MDI mode |
| PZD 11 | Deceleration override for direct setpoint input / MDI mode |
| PZD 12 | Reserved |

## Assignment of control word 1.

| Bit | Abbr. | Designation |
|---|---|---|
| 0 | Off1 | ON command: 0 = OFF1 active, 1 = ON |
| 1 | Off2 | 0 =: OFF2 active<br>1 = signal: Operating condition<br>No coasting down active |
| 2 | Off3 | 0 = OFF3 active<br>1 = operating condition no rapid stop active |
| 3 | Enc | Enable inverter |

| | | |
|---|---|---|
| 4 | RejTrvTsk | Traversing blocks and direct setpoint input / MDI<br>Reject traversing task 0 = active traversing command is rejected / axis brakes with 100% deceleration override<br>1 = do not reject traversing task (axis can be traversed) |
| 5 | IntMStop | Intermediate STOP traversing blocks and MDI/direct setpoint input – intermediate stop 0 = active traversing command is interrupted / axis brakes with specified deceleration override<br>1 = no intermediate stop (axis can be traversed) |
| 6 | TrvStart | Activate traversing task Setpoint acceptance edge if MdiTyp = 0 |
| 7 | AckFault | Acknowledge fault |
| 8 | Jog1 | Jog signal source 1 |
| 9 | Jog2 | Jog signal source 2 |
| 10 | LB | Life bit (control requested from PLC) |
| 11 | RefStart | Start referencing |
| 12 | Bit12 | Reserved |
| 13 | Bit13 | External block change (0 →1) |
| 14 | Bit14 | Reserved |
| 15 | Bit15 | Reserved |

Assignment of EPosSTW 1

| Bit | Abbr | Designation |
|---|---|---|
| 0 | TrvBit0 | Block selection, bit 0 |
| 1 | TrvBit1 | Block selection, bit 1 |
| 2 | TrvBit2 | Block selection, bit 2 |
| 3 | TrvBit3 | Block selection, bit 3 |
| 4 | TrvBit4 | Block selection, bit 4 |
| 5 | TrvBit5 | Block selection, bit 5 |
| 6 | Bit6 | Reserved |
| 7 | Bit7 | Reserved |

| | | |
|---|---|---|
| 8 | MdiTyp | Positioning type 0 = relative positioning<br>1 = absolute positioning |
| 9 | MdiPos | Direction selection for the setup, or absolute positioning of rotary axes, in positive direction |
| 10 | MdiNeg | Direction selection for the setup, or absolute positioning of rotary axes, in negative direction |
| 11 | Bit11 | Reserved |
| 12 | MdiTrTyp | Transfer type 0 = value acceptance through 0 □ 1 edge at MdiEdge 1 signal: continuous setpoint acceptance |
| 13 | Bit13 | Reserved |
| 14 | MdiSetup | Direct setpoint input/MDI – setup selection<br>Select MDI mode setup 0 = positioning<br>1 = setup |
| 15 | MdiStart | MDI / direct setpoint input mode |

Assignment of EPosSTW 2

| Bit | Abbr. | Designation |
|---|---|---|
| 0 | TrkMode | Start follow-up mode |
| 1 | SetRefPt | Set reference point |
| 2 | ActRefCam | Activate reference cam |
| 3 | Bit3 | Activate fixed stop |
| 4 | Bit4 | Reserved |
| 5 | JogInc | Jogging:<br>0 = endless traversing<br>1 = traverse by parameterized distance |
| 6 | Bit6 | Reserved |
| 7 | Bit7 | Reserved |

| | | |
|---|---|---|
| 8 | RefTyp | Referencing type selection<br>0 = reference point approach 1 = flying referencing |
| 9 | RefStDi | Reference point approach, start direction<br>0 = positive start direction<br>1 = negative start direction |
| 10 | RefInpS | Sets the signal source for the selection of the probe for flying (passive) referencing<br>0 = probe 1 is activated<br>1 = probe 2 is activated |
| 11 | RefEdge | Passive referencing: Setting the edge evaluation<br>0: positive edge<br>1: negative edge |
| 12 | Bit12 | Reserved |
| 13 | Bit13 | Reserved |
| 14 | SftLimAct | Activation of the software limit switches |
| 15 | StpCamAct | Activation of the stop cams |

Assignment of STW2

| Bit | Abbr. | Designation |
|---|---|---|
| 0 | DDSBit0 | Drive data set, bit 0 |
| 1 | DDSBit1 | Drive data set, bit 1 |
| 2 | DDSBit2 | Drive data set, bit 2 |
| 3 | DDSBit3 | Drive data set, bit 3 |
| 4 | DDSBit4 | Drive data set, bit 4 |
| 5 | GlbStart | Global start |
| 6 | ResIComp | Reset I-component of speed controller |

| | | |
|---|---|---|
| 7 | ActPrkAxis | Activate parking axis |
| 8 | TrvFixedStp | Travel to fixed stop |
| 9 | GlbTrgCom | Global trigger command |
| 10 | Bit10 | Reserved |
| 11 | MotSwOver | Motor switchover completed (0 → 1) |
| 12 | MsZykBit0 | Master sign-of-life, bit 0 |
| 13 | MsZykBit1 | Master sign-of-life, bit 1 |
| 14 | MsZykBit2 | Master sign-of-life, bit 2 |
| 15 | MsZykBit3 | Master sign-of-life, bit 3 |

Setpoint overview

| PZD | Abbr. | Setpoint |
|---|---|---|
| 5 | OverrideV | Velocity override |
| 6+7 | Position | Position setpoint |
| 8+9 | Velocity | Velocity setpoint |
| 10 | OverrideA | Acceleration override |
| 11 | OverrideD | Deceleration override |
| 12 | Word 12 | Reserved |

Status words.

| PZD | Assignment of the process data |
|---|---|
| PZD 1 | Status word 1 |
| PZD 2 | EPosZSW 1 |
| PZD 3 | EPosZSW 2 |
| PZD 4 | Status word 2 |
| PZD 5 | MELDW |

| | |
|---|---|
| PZD 6 | Position actual value [LU] |
| PZD 7 | |
| PZD 8 | Velocity actual value (refers to the reference speed p2000)<br>Note: 40000000HEX = 100% |
| PZD 9 | |
| PZD 10 | Fault (transfer of the active fault number) |
| PZD 11 | Alarm (transfer of the active alarm number) |
| PZD 12 | Reserved |

Assignment of status word 1

| Bit | Abbr. | Designation |
|---|---|---|
| 0 | RTS | Ready to start |
| 1 | RDY | Ready to operate |
| 2 | IOp | Drive is switched on (condition for the mode selection of the EPos) |
| 3 | Fault | Fault active |
| 4 | NoOff2Act | OFF2 not activated (partial condition for switching on) |
| 5 | NoOff3Act | OFF3 not activated (partial condition for switching on) |
| 6 | PowInhbt | Switching on inhibited active |
| 7 | Alarm | Alarm/warning active |
| 8 | NoFlwErr | Following error within tolerance |
| 9 | LbCr | Control requested |
| 10 | TargPos | Target position reached |
| 11 | RefPSet | Reference point set |
| 12 | TrvTskAck | Acknowledgment, traversing block activated |
| 13 | StndStill | \|n_act\| < speed threshold value 3 [p2161]<br>This bit is used for the standstill detection |
| 14 | Accel | Axis accelerates |
| 15 | Decel | Axis decelerates |

Assignment of EPosZSW 1

| Bit | Abbr. | Designation |
|---|---|---|
| 0 | ActTrvBit0 | Active traversing block, bit 0 |
| 1 | ActTrvBit1 | Active traversing block, bit 1 |
| 2 | ActTrvBit2 | Active traversing block, bit 2 |
| 3 | ActTrvBit3 | Active traversing block, bit 3 |
| 4 | ActTrvBit4 | Active traversing block, bit 4 |
| 5 | ActTrvBit5 | Active traversing block, bit 5 |
| 6 | Bit6 | Reserved |
| 7 | Bit7 | Reserved |
| 8 | StpCamMinAct | STOP cam minus active |
| 9 | StpCamPlsAct | STOP cam plus active |
| 10 | JogAct | Jog mode is active |
| 11 | RefAct | Reference point approach mode active |
| 12 | FlyRefAct | Flying referencing active |
| 13 | TrvBlAct | Traversing blocks mode active |
| 14 | MdiStupAct | In the direct setpoint input / MDI mode, setup is active |
| 15 | MdiPosAct | In the direct setpoint input / MDI mode, positioning is active |

Assignment of EPosZSW 2

| Bit | Abbr. | Designation |
|---|---|---|
| 0 | TrkModeAct | Follow-up/tracking mode active |
| 1 | VeloLimAct | Velocity limitation active |
| 2 | SetPStat | Setpoint static |

| | | |
|---|---|---|
| 3 | PrntMrkOut | Print mark outside outer window |
| 4 | FWD | Axis moves forward |
| 5 | BWD | Axis moves backward |
| 6 | SftSwMinAct | Minus software limit switch actuated |
| 7 | SftSwPlsAct | Plus software limit switch actuated |
| 8 | PosSmCam1 | Position actual value ← cam switching position 1 |
| 9 | PosSmCam2 | Position actual value ← cam switching position 2 |
| 10 | TrvOut1 | Direct output 1 via the traversing block |
| 11 | TrvOut2 | Direct output 2 via the traversing block |
| 12 | FxStpRd | Fixed stop reached |
| 13 | FxStpTrRd | Fixed stop clamping torque reached |
| 14 | TrvFxStpAct | Travel to fixed stop active |
| 15 | CmdAct | Traversing active |

Assignment of status word 2

| Bit | Abbr | Designation |
|---|---|---|
| 0 | ActDDSBit0 | Drive data set, bit 0 |
| 1 | ActDDSBit1 | Drive data set, bit 1 |
| 2 | ActDDSBit2 | Drive data set, bit 2 |
| 3 | ActDDSBit3 | Drive data set, bit 3 |
| 4 | ActDDSBit4 | Drive data set, bit 4 |
| 5 | CmdActRelBrk | Open holding brake active |
| 6 | T rqContMode | Torque-controlled operation |
| 7 | ParkAxisAct | Parking axis selected |
| 8 | Bit8 | Reserved |

| | | |
|---|---|---|
| 9 | GlbTrgReq | Global trigger request |
| 10 | PulsEn | Pulses enabled |
| 11 | MotSwOverAct | Motor data set switchover active |
| 12 | SlvZykBit0 | Slave sign-of-life, bit 0 |
| 13 | SlvZykBit1 | Slave sign-of-life, bit 1 |
| 14 | SlvZykBit2 | Slave sign-of-life, bit 2 |
| 15 | SlvZykBit3 | Slave sign-of-life, bit 3 |

Actual value overview

| PZD | Abbr. | Actual value |
|---|---|---|
| 5 | Word6 | Reserved |
| 6+7 | Position | Position actual value |
| 8+9 | Velocity | Velocity actual value |
| 10 | ErrNr | Error |
| 11 | WarnNr | Alarm |
| 12 | Reserved | Reserved |

Siemens structure of Standard Telegram 1.

Control words.

| S7 bit display (drive) | Meaning |
|---|---|
| STW1 1.0 (bit 0) | OFF1/ON (pulse enable possible) |
| STW1 1.1 (bit 1) | OFF2/ON (pulse enable possible) |
| STW1 1.2 (bit 2) | OFF3/ON (pulse enable possible) |
| STW1 1.3 (bit 3) | Enable or disable operation |
| STW1 1.4 (bit 4) | Enable ramp-function generator |
| STW1 1.5 (bit 5) | Continue ramp-function generator |
| STW1 1.6 (bit 6) | Enable speed setpoint |
| STW1 1.7 (bit 7) | Acknowledge fault |
| STW1 0.0 (bit 8) | Reserved |
| STW1 0.1 (bit 9) | Reserved |
| STW1 0.2 (bit 10) | Master control by PLC |
| STW1 0.3 (bit 11) | Direction of rotation |
| STW1 0.4 (bit 12) | Unconditionally open holding brake |
| STW1 0.5 (bit 13) | Motorized potentiometer, increase setpoint |
| STW1 0.6 (bit 14) | Motorized potentiometer, decrease setpoint |
| STW1 0.7 (bit 15) | Reserved |
| STW2 (bits 16 to 32) | Speed setpoint |

Status words

| S7 bit display (drive) | Meaning |
|---|---|
| ZTW1 1.0 (bit 0) | Ready to start |
| ZTW1 1.1 (bit 1) | Ready to operate |
| ZTW1 1.2 (bit 2) | Operation enabled |
| ZTW1 1.3 (bit 3) | Fault active |
| ZTW1 1.4 (bit 4) | No coast to stop active (OFF2 active) |
| ZTW1 1.5 (bit 5) | No coast to stop active (OFF3 inactive) |
| ZTW1 1.6 (bit 6) | Switching on inhibited active |
| ZTW1 1.7 (bit 7) | Alarm active |
| ZTW1 0.0 (bit 8) | Following error within the tolerance range |
| ZTW1 0.1 (bit 9) | PZD control assumed |
| ZTW1 0.2 (bit 10) | Target position reached |
| ZTW1 0.3 (bit 11) | Open holding brake |
| ZTW1 0.4 (bit 12) | Acknowledgment, traversing block activated |
| ZTW1 0.5 (bit 13) | No alarm for motor overtemperature |
| ZTW1 0.6 (bit 14) | Direction of rotation |
| ZTW1 0.7 (bit 15) | No thermal overload in power unit alarm |
| ZTW2 (bits 16 to 32) | Bits 16 – 31 □ actual speed value |

Siemens structure of Telegram 352.

Control words.

| PZD | Assigment of the process data |
|---|---|
| PZD 1 | Control word in bit |
| PZD 2 | NSOLL_A → Speed setpoint |
| PZD 3 | Spare word |
| PZD 4 | Spare word |
| PZD 5 | Spare word |
| PZD 6 | Spare word |

Control word in bit

| S7 bit display (drive) | Meaning |
|---|---|
| STW1 1.0 (bit 0) | OFF1/ON (pulse enable possible) |
| STW1 1.1 (bit 1) | OFF2/ON (pulse enable possible) |
| STW1 1.2 (bit 2) | OFF3/ON (pulse enable possible) |
| STW1 1.3 (bit 3) | Enable or disable operation |
| STW1 1.4 (bit 4) | Enable ramp-function generator |
| STW1 1.5 (bit 5) | Continue ramp-function generator |
| STW1 1.6 (bit 6) | Enable speed setpoint |
| STW1 1.7 (bit 7) | Acknowledge fault |
| STW1 0.0 (bit 8) | Reserved |
| STW1 0.1 (bit 9) | Reserved |
| STW1 0.2 (bit 10) | Master control by PLC |

| | |
|---|---|
| STW1 0.3 (bit 11) | Direction of rotation |
| STW1 0.4 (bit 12) | Unconditionally open holding brake |
| STW1 0.5 (bit 13) | Motorized potentiometer, increase setpoint |
| STW1 0.6 (bit 14) | Motorized potentiometer, decrease setpoint |
| STW1 0.7 (bit 15) | Reserved |

Setpoint overview

| PZD | Setpoint overview |
|---|---|
| PZD 2 | NSOLL_A → Speed setpoint |
| PZD 3 | Spare word |
| PZD 4 | Spare word |
| PZD 5 | Spare word |
| PZD 6 | Spare word |

Status words

| PZD | Symbol |
|---|---|
| PZD 1 | Status word |
| PZD 2 | NIST_A_GLATT → Smoothed speed actual value |
| PZD 3 | IAIST_GLATT → Smoothed actual corrent value |
| PZD 4 | MIST_GLATT → Actual torque |
| PZD 5 | WARN_CODE → Alarm number |
| PZD 6 | FAULT_CODE → Fault number |

Status word

| S7 bit display (drive) | Meaning |
|---|---|
| ZTW1 1.0 (bit 0) | Ready to start |
| ZTW1 1.1 (bit 1) | Ready to operate |
| ZTW1 1.2 (bit 2) | Operation enabled |
| ZTW1 1.3 (bit 3) | Fault active |
| ZTW1 1.4 (bit 4) | No coast to stop active (OFF2 active) |
| ZTW1 1.5 (bit 5) | No coast to stop active (OFF3 inactive) |
| ZTW1 1.6 (bit 6) | Switching on inhibited active |
| ZTW1 1.7 (bit 7) | Alarm active |
| ZTW1 0.0 (bit 8) | Following error within the tolerance range |
| ZTW1 0.1 (bit 9) | PZD control assumed |
| ZTW1 0.2 (bit 10) | Target position reached |
| ZTW1 0.3 (bit 11) | Open holding brake |
| ZTW1 0.4 (bit 12) | Acknowledgment, traversing block activated |
| ZTW1 0.5 (bit 13) | No alarm for motor overtemperature |
| ZTW1 0.6 (bit 14) | Direction of rotation |
| ZTW1 0.7 (bit 15) | No thermal overload in power unit alarm |
| ZTW2 (bits 16 to 32) | Bits 16 – 31 → actual speed value |

Actual value overview

| PZD | Actual value overview |
|---|---|
| PZD 2 | NIST_A_GLATT → Smoothed speed actual value |
| PZD 3 | IAIST_GLATT → Smoothed actual corrent value |
| PZD 4 | MIST_GLATT → Actual torque |
| PZD 5 | WARN_CODE → Alarm number |
| PZD 6 | FAULT_CODE → Fault number |

# APPENDIX B

Rockwell Automation structure of PowerFlex 520 Allen Bradley inverter for speed control.

Command words.

| Words | Designation |
|---|---|
| 1° integer | Logic command word |
| 2° integer | Frequency command word |

Logic command word.

| Name | Data type | Meaning |
|---|---|---|
| Stop | Bool | Motion  stops |
| Start | Bool | Motion starts |
| Jog | Bool | Jog mode activates |
| ClearFaults | Bool | The alarms are resetted. |
| Forward | Bool | Positive start direction |
| Reverse | Bool | Negative start direction |
| ForceKeypadCtrl | Bool | Keypad forces operations |
| MOPIncrement | Bool | Frequency command increases |
| AccelRate1 | Bool | It selects the first acceleration rate |
| AccelRate2 | Bool | It selects the second acceleration rate |
| DecelRate1 | Bool | It selects the first deceleration rate |

| | | |
|---|---|---|
| DecelRate2 | Bool | It selects the second deceleration rate |
| FreqSel01 | Bool | It selects the first frequency |
| FreqSel02 | Bool | It selects the second frequency |
| FreqSel03 | Bool | It selects the third frequency |
| MOPDecrement | Bool | Frequency command decreases |

Frequency command word.

| Name | Data type | Meaning |
|---|---|---|
| FreqCommand | INT | It sets the wanted frequency. |

Status words.

| Words | Designation |
|---|---|
| 1° integer | Drive status word |
| 2° integer | Frequency actual value |

Drive logic status word.

| Name | Data type | Meaning |
|---|---|---|
| Ready | Bool | The drive is ready |
| Active | Bool | The drive is actived |
| CommandDir | Bool | Command direction |
| ActualDir | Bool | Actual direction |
| Accelerating | Bool | Accelarating state |
| Decelerating | Bool | Decelarating state |
| Faulted | Bool | There is a fault and the drive is blocked |
| AtReference | Bool | The speed setpoint is reached |
| CommFreqCnt | Bool | Main frequency is commanded by the active command |
| CommandLogicCnt | Bool | The “cnd” operation is controlled by active command |
| ParmsLocked | Bool | Parameters are safe |
| DigIn1Active | Bool | Active state of digital input 1 |
| DigIn2Active | Bool | Active state of digital input 2 |
| DigIn3Active | Bool | Active state of digital input 3 |
| DigIn4Active | Bool | Active state of digital input 4 |

Frequency status word.

| Name | Data type | Meaning |
|---|---|---|
| OutputFreq | INT | Speed actual value |

Rockwell Automation structure of PowerFlex 750 Allen Bradley inverter for position control.

Command words.

| Words | Designation |
|---|---|
| 1° integer | Logic command word |
| 2° integer | Logic command word |
| 3° | Position command word |

Logic command word.

| Bit | Name | Data type | Meaning |
|---|---|---|---|
| 0 | Stop | Bool | Motion stops |
| 1 | Start | Bool | Motion starts |
| 2 | Jog1 | Bool | Jog signal source 1 |
| 3 | ClearFaults | Bool | The alarms are resetted. |

| | | | |
|---|---|---|---|
| 4 | Forward | Bool | Positive start direction |
| 5 | Reverse | Bool | Negative start direction |
| 6 | Manual | Bool | Manual mode is setted |
| 7 | Reserved | Bool | Reserved |
| 8 | AccelRate1 | Bool | It selects the first acceleration rate |
| 9 | AccelRate2 | Bool | It selects the second acceleration rate |
| 10 | DecelRate1 | Bool | It selects the first deceleration rate |
| 11 | DecelRate2 | Bool | It selects the second deceleration rate |
| 12 | FreqSel01 | Bool | It selects the first frequency |
| 13 | FreqSel02 | Bool | It selects the second frequency |
| 14 | FreqSel03 | Bool | It selects the third frequency |
| 15 | Reserved | Bool | Reserved |
| 16 | InertiaStop | Bool | Stop by inertia |
| 17 | CurLimStop | Bool | Current limit stop |
| 18 | March | Bool | March starts |
| 19 | Jog2 | Bool | Jog signal source 2 |
| 20 | Reserved | Bool | Reserved |
| 21 | Reserved | Bool | Reserved |
| 22 | Reserved | Bool | Reserved |
| 23 | Reserved | Bool | Reserved |
| 24 | Reserved | Bool | Reserved |
| 25 | Reserved | Bool | Reserved |
| 26 | Reserved | Bool | Reserved |
| 27 | Reserved | Bool | Reserved |
| 28 | Reserved | Bool | Reserved |
| 29 | Reserved | Bool | Reserved |
| 30 | Reserved | Bool | Reserved |
| 31 | Reserved | Bool | Reserved |

Position command word.

| Name | Data type | Meaning |
|---|---|---|
| Reference | Real | It sets the wanted position |

Status words.

| Words | Designation |
|---|---|
| 1° integer | Drive logic status words |
| 2° integer | Drive logic status words |
| 3°integer | Position status word |

Drive status words

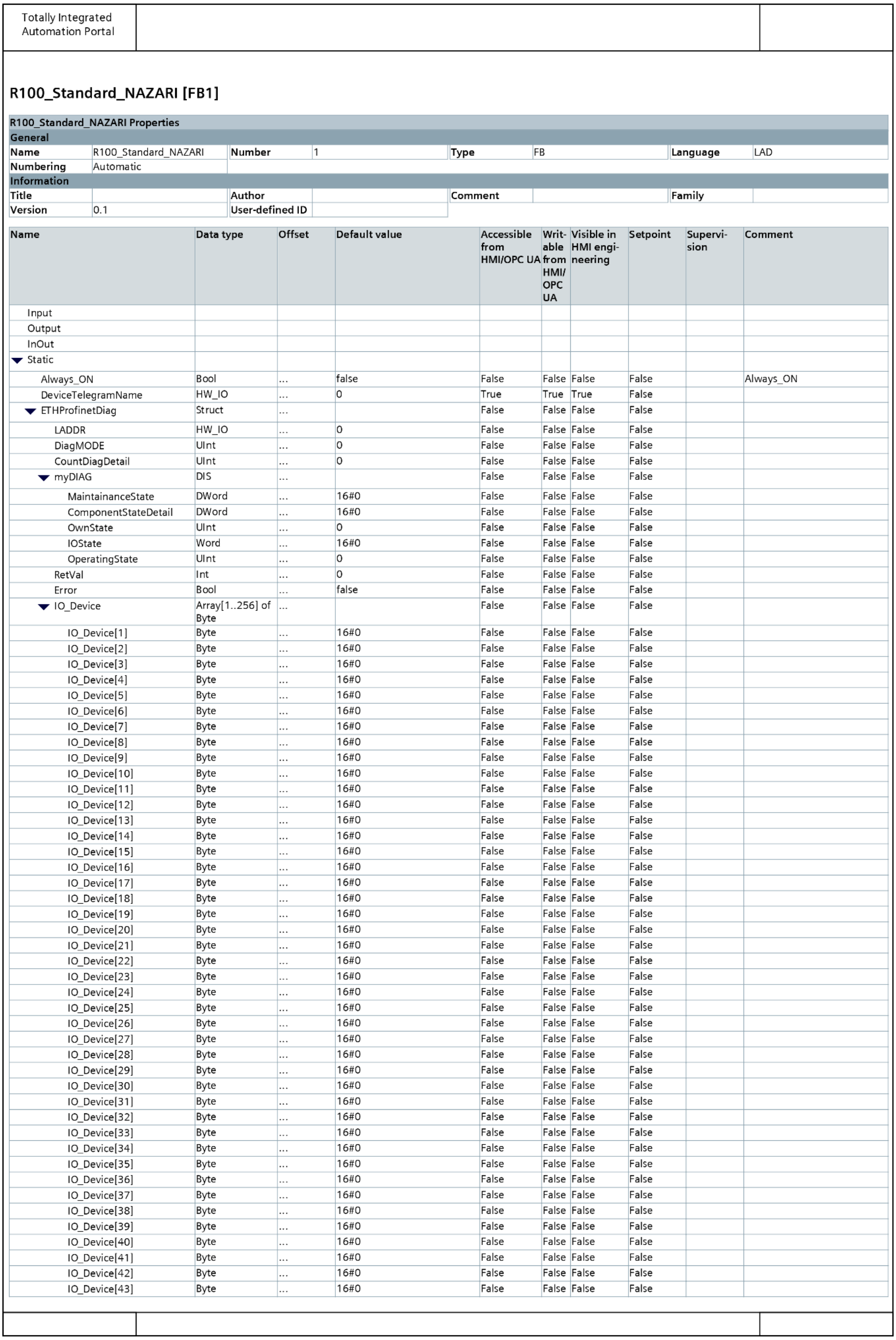
| Bit | Name | Data type | Meaning |
|---|---|---|---|
| 0 | Ready | Bool | The drive is ready |
| 1 | Active | Bool | The drive is actived |
| 2 | CommandDir | Bool | Command direction |
| 3 | ActualDir | Bool | Actual direction |
| 4 | Accelerating | Bool | Accelarating state |
| 5 | Decelerating | Bool | Decelarating state |

| | | | |
|---|---|---|---|
| 6 | Alarm | Bool | There is alarm |
| 7 | Error | Bool | There is error |
| 8 | AtReference | Bool | The speed setpoint is reached |
| 9 | Manual | Bool | Manual mode is active |
| 10 | IDRifVel0 | Bool | Preset speed 3 (Parameter 573) |
| 11 | IDRifVel1 | Bool | Preset speed 4 (Parameter 574) |
| 12 | IDRifVel2 | Bool | Preset speed 5 (Parameter 575) |
| 13 | IDRifVel3 | Bool | Preset speed 6 (Parameter 576) |
| 14 | IDRifVel4 | | Preset speed 7 (Parameter 577) |
| 15 | Reserved | Bool | Reserved |
| 16 | Running | Bool | Engine is in gear |
| 17 | InMovement | Bool | Running jog mode |
| 18 | InArrest | Bool | Engine is stopped |
| 19 | CCBrake | Bool | CC brake is active |
| 20 | DBActive | Bool | Dynamic brake is active |
| 21 | SpeedMode | Bool | Speed control is active |
| 22 | PositionMode | Bool | Position control is active |
| 23 | TorqueMode | Bool | Torque control is active |
| 24 | AtZeroVel | Bool | Velocity is zero |
| 25 | InitialPosition | Bool | In home position |
| 26 | AtLimit | Bool | At limit |
| 27 | ActualLimit | Bool | Motor has reached the current limit |
| 28 | RegFreqBus | Bool | Reg frequency bus |
| 29 | OnEnable | Bool | Enabling |
| 30 | OverloadMotor | Bool | Motor overload |
| 31 | Rigen | Bool | regeneration |

Position status words

| Name | Data type | Meaning |
|---|---|---|
| Feedback | Real | Actual value |

# APPENDIX C



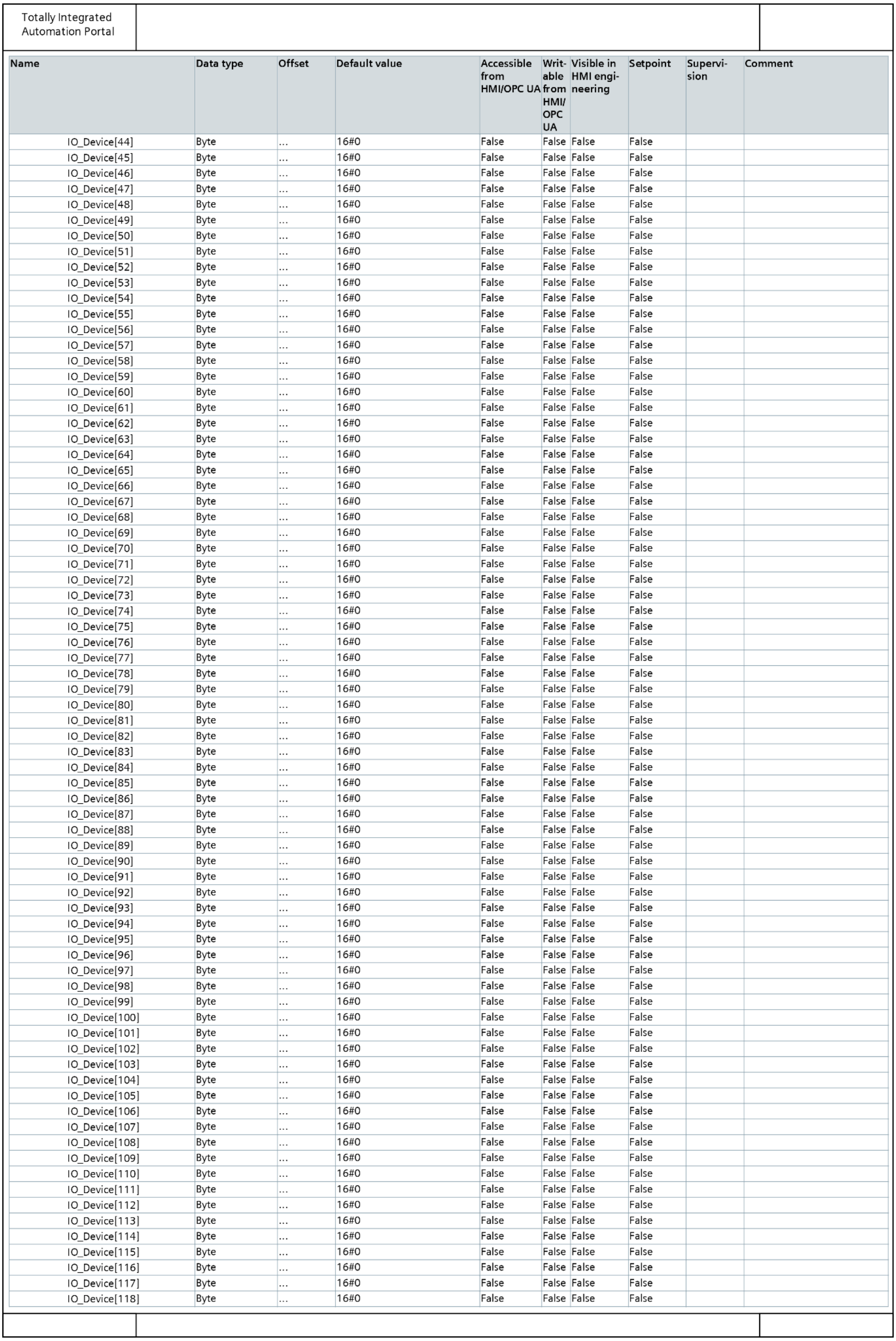
Totally Integrated
Automation Portal

## R100_Standard_NAZARI [FB1]

**R100_Standard_NAZARI Properties**

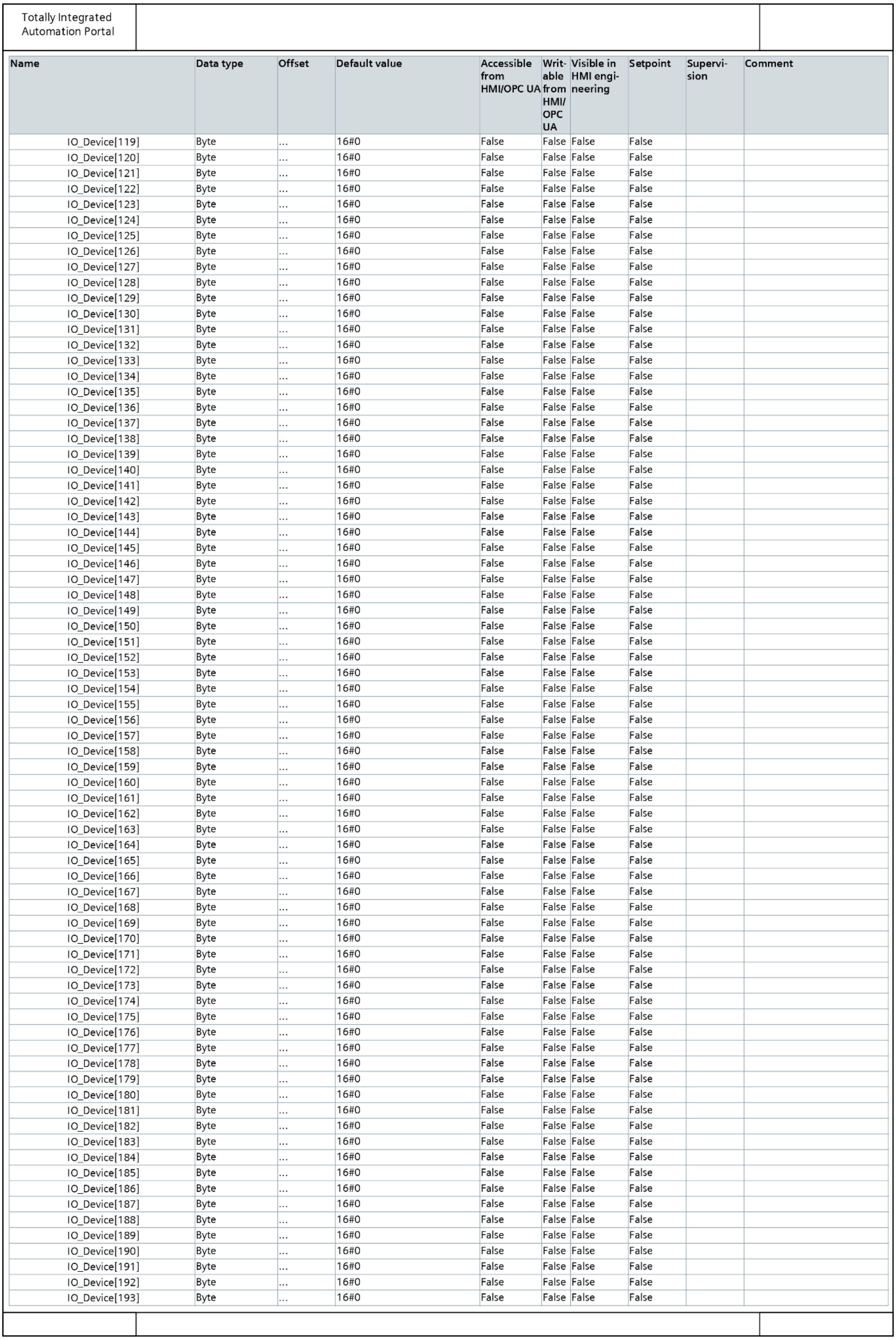
| | | | | | | | |
|---|---|---|---|---|---|---|---|
| **General** | | | | | | | |
| **Name** | R100_Standard_NAZARI | **Number** | 1 | **Type** | FB | **Language** | LAD |
| **Numbering** | Automatic | | | | | | |
| **Information** | | | | | | | |
| **Title** | | **Author** | | **Comment** | | **Family** | |
| **Version** | 0.1 | **User-defined ID** | | | | | |

| Name | Data type | Offset | Default value | Accessible from HMI/OPC UA | Writ-able from HMI/ OPC UA | Visible in HMI engi-neering | Setpoint | Supervi-sion | Comment |
|---|---|---|---|---|---|---|---|---|---|
| Input | | | | | | | | | |
| Output | | | | | | | | | |
| InOut | | | | | | | | | |
| ▼ Static | | | | | | | | | |
| Always_ON | Bool | ... | false | False | False | False | False | | Always_ON |
| DeviceTelegramName | HW_IO | ... | 0 | True | True | True | False | | |
| ▼ ETHProfinetDiag | Struct | ... | | False | False | False | False | | |
| LADDR | HW_IO | ... | 0 | False | False | False | False | | |
| DiagMODE | UInt | ... | 0 | False | False | False | False | | |
| CountDiagDetail | UInt | ... | 0 | False | False | False | False | | |
| ▼ myDIAG | DIS | ... | | False | False | False | False | | |
| MaintainanceState | DWord | ... | 16#0 | False | False | False | False | | |
| ComponentStateDetail | DWord | ... | 16#0 | False | False | False | False | | |
| OwnState | UInt | ... | 0 | False | False | False | False | | |
| IOState | Word | ... | 16#0 | False | False | False | False | | |
| OperatingState | UInt | ... | 0 | False | False | False | False | | |
| RetVal | Int | ... | 0 | False | False | False | False | | |
| Error | Bool | ... | false | False | False | False | False | | |
| ▼ IO_Device | Array[1..256] of Byte | ... | | False | False | False | False | | |
| IO_Device[1] | Byte | ... | 16#0 | False | False | False | False | | |
| IO_Device[2] | Byte | ... | 16#0 | False | False | False | False | | |
| IO_Device[3] | Byte | ... | 16#0 | False | False | False | False | | |
| IO_Device[4] | Byte | ... | 16#0 | False | False | False | False | | |
| IO_Device[5] | Byte | ... | 16#0 | False | False | False | False | | |
| IO_Device[6] | Byte | ... | 16#0 | False | False | False | False | | |
| IO_Device[7] | Byte | ... | 16#0 | False | False | False | False | | |
| IO_Device[8] | Byte | ... | 16#0 | False | False | False | False | | |
| IO_Device[9] | Byte | ... | 16#0 | False | False | False | False | | |
| IO_Device[10] | Byte | ... | 16#0 | False | False | False | False | | |
| IO_Device[11] | Byte | ... | 16#0 | False | False | False | False | | |
| IO_Device[12] | Byte | ... | 16#0 | False | False | False | False | | |
| IO_Device[13] | Byte | ... | 16#0 | False | False | False | False | | |
| IO_Device[14] | Byte | ... | 16#0 | False | False | False | False | | |
| IO_Device[15] | Byte | ... | 16#0 | False | False | False | False | | |
| IO_Device[16] | Byte | ... | 16#0 | False | False | False | False | | |
| IO_Device[17] | Byte | ... | 16#0 | False | False | False | False | | |
| IO_Device[18] | Byte | ... | 16#0 | False | False | False | False | | |
| IO_Device[19] | Byte | ... | 16#0 | False | False | False | False | | |
| IO_Device[20] | Byte | ... | 16#0 | False | False | False | False | | |
| IO_Device[21] | Byte | ... | 16#0 | False | False | False | False | | |
| IO_Device[22] | Byte | ... | 16#0 | False | False | False | False | | |
| IO_Device[23] | Byte | ... | 16#0 | False | False | False | False | | |
| IO_Device[24] | Byte | ... | 16#0 | False | False | False | False | | |
| IO_Device[25] | Byte | ... | 16#0 | False | False | False | False | | |
| IO_Device[26] | Byte | ... | 16#0 | False | False | False | False | | |
| IO_Device[27] | Byte | ... | 16#0 | False | False | False | False | | |
| IO_Device[28] | Byte | ... | 16#0 | False | False | False | False | | |
| IO_Device[29] | Byte | ... | 16#0 | False | False | False | False | | |
| IO_Device[30] | Byte | ... | 16#0 | False | False | False | False | | |
| IO_Device[31] | Byte | ... | 16#0 | False | False | False | False | | |
| IO_Device[32] | Byte | ... | 16#0 | False | False | False | False | | |
| IO_Device[33] | Byte | ... | 16#0 | False | False | False | False | | |
| IO_Device[34] | Byte | ... | 16#0 | False | False | False | False | | |
| IO_Device[35] | Byte | ... | 16#0 | False | False | False | False | | |
| IO_Device[36] | Byte | ... | 16#0 | False | False | False | False | | |
| IO_Device[37] | Byte | ... | 16#0 | False | False | False | False | | |
| IO_Device[38] | Byte | ... | 16#0 | False | False | False | False | | |
| IO_Device[39] | Byte | ... | 16#0 | False | False | False | False | | |
| IO_Device[40] | Byte | ... | 16#0 | False | False | False | False | | |
| IO_Device[41] | Byte | ... | 16#0 | False | False | False | False | | |
| IO_Device[42] | Byte | ... | 16#0 | False | False | False | False | | |
| IO_Device[43] | Byte | ... | 16#0 | False | False | False | False | | |



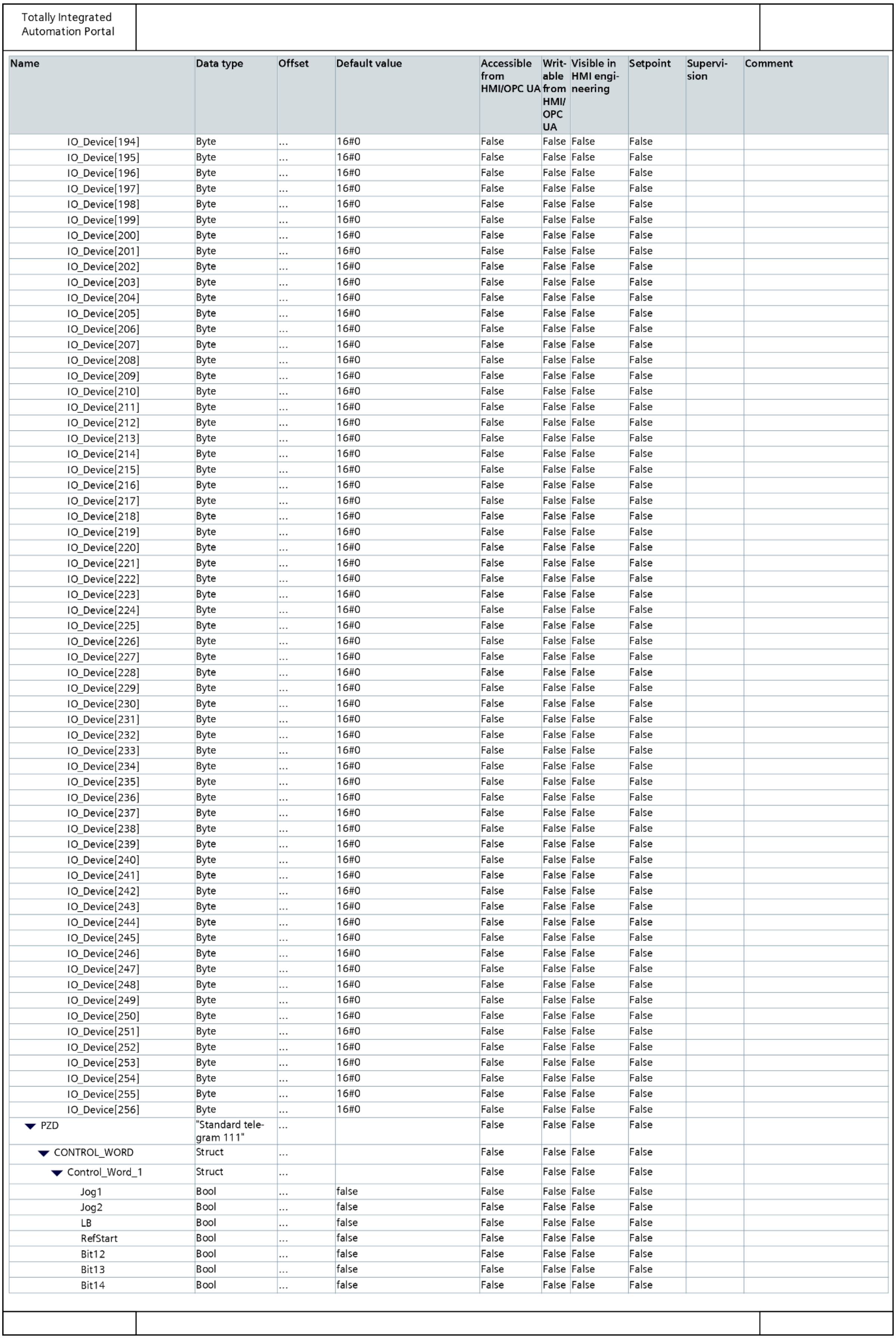
Totally Integrated Automation Portal

| Name | Data type | Offset | Default value | Accessible from HMI/OPC UA | Writable from HMI/ OPC UA | Visible in HMI engineering | Setpoint | Supervision | Comment |
|---|---|---|---|---|---|---|---|---|---|
| IO_Device[44] | Byte | ... | 16#0 | False | False | False | False | | |
| IO_Device[45] | Byte | ... | 16#0 | False | False | False | False | | |
| IO_Device[46] | Byte | ... | 16#0 | False | False | False | False | | |
| IO_Device[47] | Byte | ... | 16#0 | False | False | False | False | | |
| IO_Device[48] | Byte | ... | 16#0 | False | False | False | False | | |
| IO_Device[49] | Byte | ... | 16#0 | False | False | False | False | | |
| IO_Device[50] | Byte | ... | 16#0 | False | False | False | False | | |
| IO_Device[51] | Byte | ... | 16#0 | False | False | False | False | | |
| IO_Device[52] | Byte | ... | 16#0 | False | False | False | False | | |
| IO_Device[53] | Byte | ... | 16#0 | False | False | False | False | | |
| IO_Device[54] | Byte | ... | 16#0 | False | False | False | False | | |
| IO_Device[55] | Byte | ... | 16#0 | False | False | False | False | | |
| IO_Device[56] | Byte | ... | 16#0 | False | False | False | False | | |
| IO_Device[57] | Byte | ... | 16#0 | False | False | False | False | | |
| IO_Device[58] | Byte | ... | 16#0 | False | False | False | False | | |
| IO_Device[59] | Byte | ... | 16#0 | False | False | False | False | | |
| IO_Device[60] | Byte | ... | 16#0 | False | False | False | False | | |
| IO_Device[61] | Byte | ... | 16#0 | False | False | False | False | | |
| IO_Device[62] | Byte | ... | 16#0 | False | False | False | False | | |
| IO_Device[63] | Byte | ... | 16#0 | False | False | False | False | | |
| IO_Device[64] | Byte | ... | 16#0 | False | False | False | False | | |
| IO_Device[65] | Byte | ... | 16#0 | False | False | False | False | | |
| IO_Device[66] | Byte | ... | 16#0 | False | False | False | False | | |
| IO_Device[67] | Byte | ... | 16#0 | False | False | False | False | | |
| IO_Device[68] | Byte | ... | 16#0 | False | False | False | False | | |
| IO_Device[69] | Byte | ... | 16#0 | False | False | False | False | | |
| IO_Device[70] | Byte | ... | 16#0 | False | False | False | False | | |
| IO_Device[71] | Byte | ... | 16#0 | False | False | False | False | | |
| IO_Device[72] | Byte | ... | 16#0 | False | False | False | False | | |
| IO_Device[73] | Byte | ... | 16#0 | False | False | False | False | | |
| IO_Device[74] | Byte | ... | 16#0 | False | False | False | False | | |
| IO_Device[75] | Byte | ... | 16#0 | False | False | False | False | | |
| IO_Device[76] | Byte | ... | 16#0 | False | False | False | False | | |
| IO_Device[77] | Byte | ... | 16#0 | False | False | False | False | | |
| IO_Device[78] | Byte | ... | 16#0 | False | False | False | False | | |
| IO_Device[79] | Byte | ... | 16#0 | False | False | False | False | | |
| IO_Device[80] | Byte | ... | 16#0 | False | False | False | False | | |
| IO_Device[81] | Byte | ... | 16#0 | False | False | False | False | | |
| IO_Device[82] | Byte | ... | 16#0 | False | False | False | False | | |
| IO_Device[83] | Byte | ... | 16#0 | False | False | False | False | | |
| IO_Device[84] | Byte | ... | 16#0 | False | False | False | False | | |
| IO_Device[85] | Byte | ... | 16#0 | False | False | False | False | | |
| IO_Device[86] | Byte | ... | 16#0 | False | False | False | False | | |
| IO_Device[87] | Byte | ... | 16#0 | False | False | False | False | | |
| IO_Device[88] | Byte | ... | 16#0 | False | False | False | False | | |
| IO_Device[89] | Byte | ... | 16#0 | False | False | False | False | | |
| IO_Device[90] | Byte | ... | 16#0 | False | False | False | False | | |
| IO_Device[91] | Byte | ... | 16#0 | False | False | False | False | | |
| IO_Device[92] | Byte | ... | 16#0 | False | False | False | False | | |
| IO_Device[93] | Byte | ... | 16#0 | False | False | False | False | | |
| IO_Device[94] | Byte | ... | 16#0 | False | False | False | False | | |
| IO_Device[95] | Byte | ... | 16#0 | False | False | False | False | | |
| IO_Device[96] | Byte | ... | 16#0 | False | False | False | False | | |
| IO_Device[97] | Byte | ... | 16#0 | False | False | False | False | | |
| IO_Device[98] | Byte | ... | 16#0 | False | False | False | False | | |
| IO_Device[99] | Byte | ... | 16#0 | False | False | False | False | | |
| IO_Device[100] | Byte | ... | 16#0 | False | False | False | False | | |
| IO_Device[101] | Byte | ... | 16#0 | False | False | False | False | | |
| IO_Device[102] | Byte | ... | 16#0 | False | False | False | False | | |
| IO_Device[103] | Byte | ... | 16#0 | False | False | False | False | | |
| IO_Device[104] | Byte | ... | 16#0 | False | False | False | False | | |
| IO_Device[105] | Byte | ... | 16#0 | False | False | False | False | | |
| IO_Device[106] | Byte | ... | 16#0 | False | False | False | False | | |
| IO_Device[107] | Byte | ... | 16#0 | False | False | False | False | | |
| IO_Device[108] | Byte | ... | 16#0 | False | False | False | False | | |
| IO_Device[109] | Byte | ... | 16#0 | False | False | False | False | | |
| IO_Device[110] | Byte | ... | 16#0 | False | False | False | False | | |
| IO_Device[111] | Byte | ... | 16#0 | False | False | False | False | | |
| IO_Device[112] | Byte | ... | 16#0 | False | False | False | False | | |
| IO_Device[113] | Byte | ... | 16#0 | False | False | False | False | | |
| IO_Device[114] | Byte | ... | 16#0 | False | False | False | False | | |
| IO_Device[115] | Byte | ... | 16#0 | False | False | False | False | | |
| IO_Device[116] | Byte | ... | 16#0 | False | False | False | False | | |
| IO_Device[117] | Byte | ... | 16#0 | False | False | False | False | | |
| IO_Device[118] | Byte | ... | 16#0 | False | False | False | False | | |



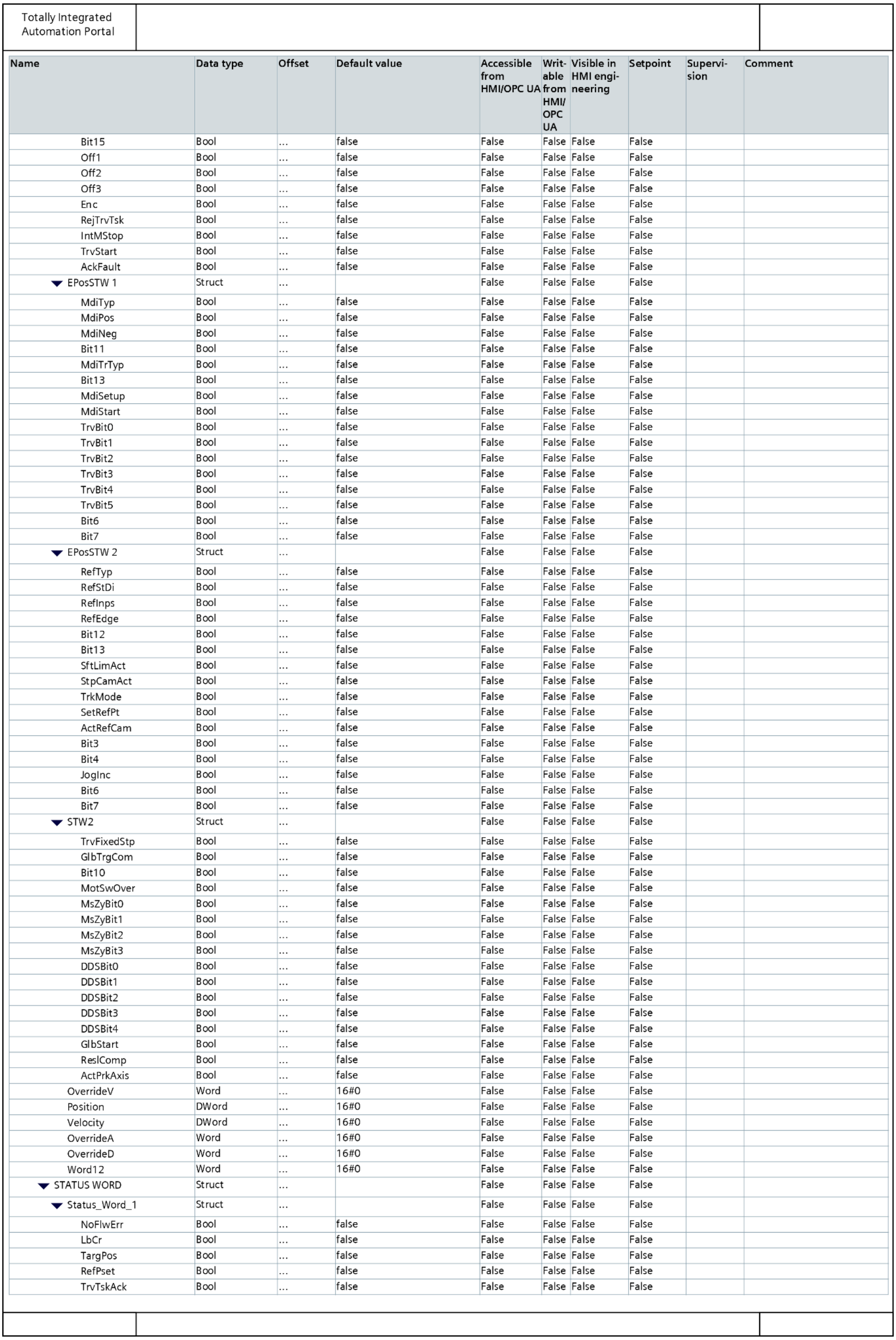
Totally Integrated Automation Portal

| Name | Data type | Offset | Default value | Accessible from HMI/OPC UA | Writable from HMI/OPC UA | Visible in HMI engineering | Setpoint | Supervision | Comment |
|---|---|---|---|---|---|---|---|---|---|
| IO_Device[119] | Byte | ... | 16#0 | False | False | False | False | | |
| IO_Device[120] | Byte | ... | 16#0 | False | False | False | False | | |
| IO_Device[121] | Byte | ... | 16#0 | False | False | False | False | | |
| IO_Device[122] | Byte | ... | 16#0 | False | False | False | False | | |
| IO_Device[123] | Byte | ... | 16#0 | False | False | False | False | | |
| IO_Device[124] | Byte | ... | 16#0 | False | False | False | False | | |
| IO_Device[125] | Byte | ... | 16#0 | False | False | False | False | | |
| IO_Device[126] | Byte | ... | 16#0 | False | False | False | False | | |
| IO_Device[127] | Byte | ... | 16#0 | False | False | False | False | | |
| IO_Device[128] | Byte | ... | 16#0 | False | False | False | False | | |
| IO_Device[129] | Byte | ... | 16#0 | False | False | False | False | | |
| IO_Device[130] | Byte | ... | 16#0 | False | False | False | False | | |
| IO_Device[131] | Byte | ... | 16#0 | False | False | False | False | | |
| IO_Device[132] | Byte | ... | 16#0 | False | False | False | False | | |
| IO_Device[133] | Byte | ... | 16#0 | False | False | False | False | | |
| IO_Device[134] | Byte | ... | 16#0 | False | False | False | False | | |
| IO_Device[135] | Byte | ... | 16#0 | False | False | False | False | | |
| IO_Device[136] | Byte | ... | 16#0 | False | False | False | False | | |
| IO_Device[137] | Byte | ... | 16#0 | False | False | False | False | | |
| IO_Device[138] | Byte | ... | 16#0 | False | False | False | False | | |
| IO_Device[139] | Byte | ... | 16#0 | False | False | False | False | | |
| IO_Device[140] | Byte | ... | 16#0 | False | False | False | False | | |
| IO_Device[141] | Byte | ... | 16#0 | False | False | False | False | | |
| IO_Device[142] | Byte | ... | 16#0 | False | False | False | False | | |
| IO_Device[143] | Byte | ... | 16#0 | False | False | False | False | | |
| IO_Device[144] | Byte | ... | 16#0 | False | False | False | False | | |
| IO_Device[145] | Byte | ... | 16#0 | False | False | False | False | | |
| IO_Device[146] | Byte | ... | 16#0 | False | False | False | False | | |
| IO_Device[147] | Byte | ... | 16#0 | False | False | False | False | | |
| IO_Device[148] | Byte | ... | 16#0 | False | False | False | False | | |
| IO_Device[149] | Byte | ... | 16#0 | False | False | False | False | | |
| IO_Device[150] | Byte | ... | 16#0 | False | False | False | False | | |
| IO_Device[151] | Byte | ... | 16#0 | False | False | False | False | | |
| IO_Device[152] | Byte | ... | 16#0 | False | False | False | False | | |
| IO_Device[153] | Byte | ... | 16#0 | False | False | False | False | | |
| IO_Device[154] | Byte | ... | 16#0 | False | False | False | False | | |
| IO_Device[155] | Byte | ... | 16#0 | False | False | False | False | | |
| IO_Device[156] | Byte | ... | 16#0 | False | False | False | False | | |
| IO_Device[157] | Byte | ... | 16#0 | False | False | False | False | | |
| IO_Device[158] | Byte | ... | 16#0 | False | False | False | False | | |
| IO_Device[159] | Byte | ... | 16#0 | False | False | False | False | | |
| IO_Device[160] | Byte | ... | 16#0 | False | False | False | False | | |
| IO_Device[161] | Byte | ... | 16#0 | False | False | False | False | | |
| IO_Device[162] | Byte | ... | 16#0 | False | False | False | False | | |
| IO_Device[163] | Byte | ... | 16#0 | False | False | False | False | | |
| IO_Device[164] | Byte | ... | 16#0 | False | False | False | False | | |
| IO_Device[165] | Byte | ... | 16#0 | False | False | False | False | | |
| IO_Device[166] | Byte | ... | 16#0 | False | False | False | False | | |
| IO_Device[167] | Byte | ... | 16#0 | False | False | False | False | | |
| IO_Device[168] | Byte | ... | 16#0 | False | False | False | False | | |
| IO_Device[169] | Byte | ... | 16#0 | False | False | False | False | | |
| IO_Device[170] | Byte | ... | 16#0 | False | False | False | False | | |
| IO_Device[171] | Byte | ... | 16#0 | False | False | False | False | | |
| IO_Device[172] | Byte | ... | 16#0 | False | False | False | False | | |
| IO_Device[173] | Byte | ... | 16#0 | False | False | False | False | | |
| IO_Device[174] | Byte | ... | 16#0 | False | False | False | False | | |
| IO_Device[175] | Byte | ... | 16#0 | False | False | False | False | | |
| IO_Device[176] | Byte | ... | 16#0 | False | False | False | False | | |
| IO_Device[177] | Byte | ... | 16#0 | False | False | False | False | | |
| IO_Device[178] | Byte | ... | 16#0 | False | False | False | False | | |
| IO_Device[179] | Byte | ... | 16#0 | False | False | False | False | | |
| IO_Device[180] | Byte | ... | 16#0 | False | False | False | False | | |
| IO_Device[181] | Byte | ... | 16#0 | False | False | False | False | | |
| IO_Device[182] | Byte | ... | 16#0 | False | False | False | False | | |
| IO_Device[183] | Byte | ... | 16#0 | False | False | False | False | | |
| IO_Device[184] | Byte | ... | 16#0 | False | False | False | False | | |
| IO_Device[185] | Byte | ... | 16#0 | False | False | False | False | | |
| IO_Device[186] | Byte | ... | 16#0 | False | False | False | False | | |
| IO_Device[187] | Byte | ... | 16#0 | False | False | False | False | | |
| IO_Device[188] | Byte | ... | 16#0 | False | False | False | False | | |
| IO_Device[189] | Byte | ... | 16#0 | False | False | False | False | | |
| IO_Device[190] | Byte | ... | 16#0 | False | False | False | False | | |
| IO_Device[191] | Byte | ... | 16#0 | False | False | False | False | | |
| IO_Device[192] | Byte | ... | 16#0 | False | False | False | False | | |
| IO_Device[193] | Byte | ... | 16#0 | False | False | False | False | | |



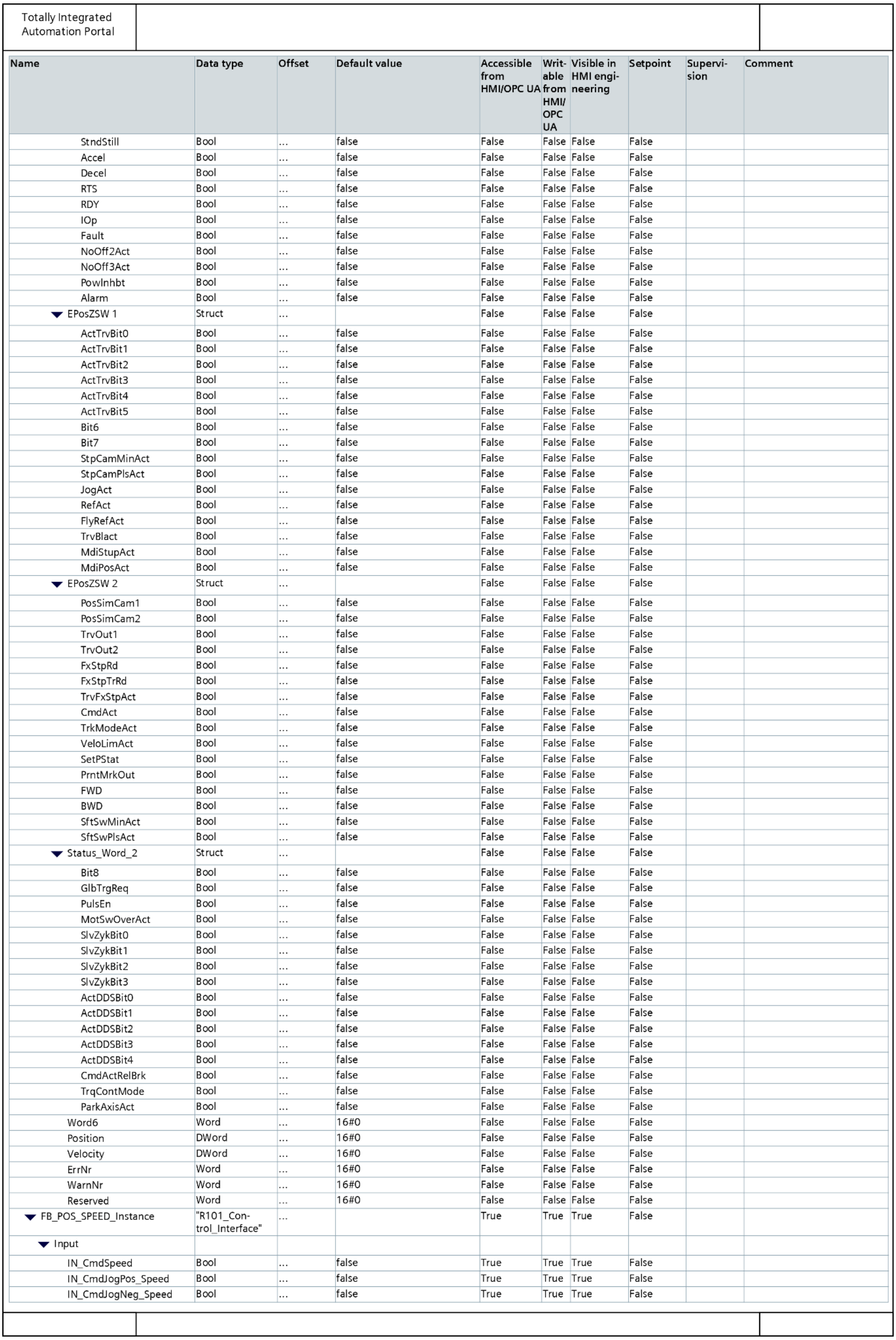
Totally Integrated Automation Portal

| Name | Data type | Offset | Default value | Accessible from HMI/OPC UA | Writable from HMI/ OPC UA | Visible in HMI engineering | Setpoint | Supervision | Comment |
|---|---|---|---|---|---|---|---|---|---|
| IO_Device[194] | Byte | ... | 16#0 | False | False | False | False | | |
| IO_Device[195] | Byte | ... | 16#0 | False | False | False | False | | |
| IO_Device[196] | Byte | ... | 16#0 | False | False | False | False | | |
| IO_Device[197] | Byte | ... | 16#0 | False | False | False | False | | |
| IO_Device[198] | Byte | ... | 16#0 | False | False | False | False | | |
| IO_Device[199] | Byte | ... | 16#0 | False | False | False | False | | |
| IO_Device[200] | Byte | ... | 16#0 | False | False | False | False | | |
| IO_Device[201] | Byte | ... | 16#0 | False | False | False | False | | |
| IO_Device[202] | Byte | ... | 16#0 | False | False | False | False | | |
| IO_Device[203] | Byte | ... | 16#0 | False | False | False | False | | |
| IO_Device[204] | Byte | ... | 16#0 | False | False | False | False | | |
| IO_Device[205] | Byte | ... | 16#0 | False | False | False | False | | |
| IO_Device[206] | Byte | ... | 16#0 | False | False | False | False | | |
| IO_Device[207] | Byte | ... | 16#0 | False | False | False | False | | |
| IO_Device[208] | Byte | ... | 16#0 | False | False | False | False | | |
| IO_Device[209] | Byte | ... | 16#0 | False | False | False | False | | |
| IO_Device[210] | Byte | ... | 16#0 | False | False | False | False | | |
| IO_Device[211] | Byte | ... | 16#0 | False | False | False | False | | |
| IO_Device[212] | Byte | ... | 16#0 | False | False | False | False | | |
| IO_Device[213] | Byte | ... | 16#0 | False | False | False | False | | |
| IO_Device[214] | Byte | ... | 16#0 | False | False | False | False | | |
| IO_Device[215] | Byte | ... | 16#0 | False | False | False | False | | |
| IO_Device[216] | Byte | ... | 16#0 | False | False | False | False | | |
| IO_Device[217] | Byte | ... | 16#0 | False | False | False | False | | |
| IO_Device[218] | Byte | ... | 16#0 | False | False | False | False | | |
| IO_Device[219] | Byte | ... | 16#0 | False | False | False | False | | |
| IO_Device[220] | Byte | ... | 16#0 | False | False | False | False | | |
| IO_Device[221] | Byte | ... | 16#0 | False | False | False | False | | |
| IO_Device[222] | Byte | ... | 16#0 | False | False | False | False | | |
| IO_Device[223] | Byte | ... | 16#0 | False | False | False | False | | |
| IO_Device[224] | Byte | ... | 16#0 | False | False | False | False | | |
| IO_Device[225] | Byte | ... | 16#0 | False | False | False | False | | |
| IO_Device[226] | Byte | ... | 16#0 | False | False | False | False | | |
| IO_Device[227] | Byte | ... | 16#0 | False | False | False | False | | |
| IO_Device[228] | Byte | ... | 16#0 | False | False | False | False | | |
| IO_Device[229] | Byte | ... | 16#0 | False | False | False | False | | |
| IO_Device[230] | Byte | ... | 16#0 | False | False | False | False | | |
| IO_Device[231] | Byte | ... | 16#0 | False | False | False | False | | |
| IO_Device[232] | Byte | ... | 16#0 | False | False | False | False | | |
| IO_Device[233] | Byte | ... | 16#0 | False | False | False | False | | |
| IO_Device[234] | Byte | ... | 16#0 | False | False | False | False | | |
| IO_Device[235] | Byte | ... | 16#0 | False | False | False | False | | |
| IO_Device[236] | Byte | ... | 16#0 | False | False | False | False | | |
| IO_Device[237] | Byte | ... | 16#0 | False | False | False | False | | |
| IO_Device[238] | Byte | ... | 16#0 | False | False | False | False | | |
| IO_Device[239] | Byte | ... | 16#0 | False | False | False | False | | |
| IO_Device[240] | Byte | ... | 16#0 | False | False | False | False | | |
| IO_Device[241] | Byte | ... | 16#0 | False | False | False | False | | |
| IO_Device[242] | Byte | ... | 16#0 | False | False | False | False | | |
| IO_Device[243] | Byte | ... | 16#0 | False | False | False | False | | |
| IO_Device[244] | Byte | ... | 16#0 | False | False | False | False | | |
| IO_Device[245] | Byte | ... | 16#0 | False | False | False | False | | |
| IO_Device[246] | Byte | ... | 16#0 | False | False | False | False | | |
| IO_Device[247] | Byte | ... | 16#0 | False | False | False | False | | |
| IO_Device[248] | Byte | ... | 16#0 | False | False | False | False | | |
| IO_Device[249] | Byte | ... | 16#0 | False | False | False | False | | |
| IO_Device[250] | Byte | ... | 16#0 | False | False | False | False | | |
| IO_Device[251] | Byte | ... | 16#0 | False | False | False | False | | |
| IO_Device[252] | Byte | ... | 16#0 | False | False | False | False | | |
| IO_Device[253] | Byte | ... | 16#0 | False | False | False | False | | |
| IO_Device[254] | Byte | ... | 16#0 | False | False | False | False | | |
| IO_Device[255] | Byte | ... | 16#0 | False | False | False | False | | |
| IO_Device[256] | Byte | ... | 16#0 | False | False | False | False | | |
| ▼ PZD | "Standard telegram 111" | ... | | False | False | False | False | | |
| ▼ CONTROL_WORD | Struct | ... | | False | False | False | False | | |
| ▼ Control_Word_1 | Struct | ... | | False | False | False | False | | |
| Jog1 | Bool | ... | false | False | False | False | False | | |
| Jog2 | Bool | ... | false | False | False | False | False | | |
| LB | Bool | ... | false | False | False | False | False | | |
| RefStart | Bool | ... | false | False | False | False | False | | |
| Bit12 | Bool | ... | false | False | False | False | False | | |
| Bit13 | Bool | ... | false | False | False | False | False | | |
| Bit14 | Bool | ... | false | False | False | False | False | | |



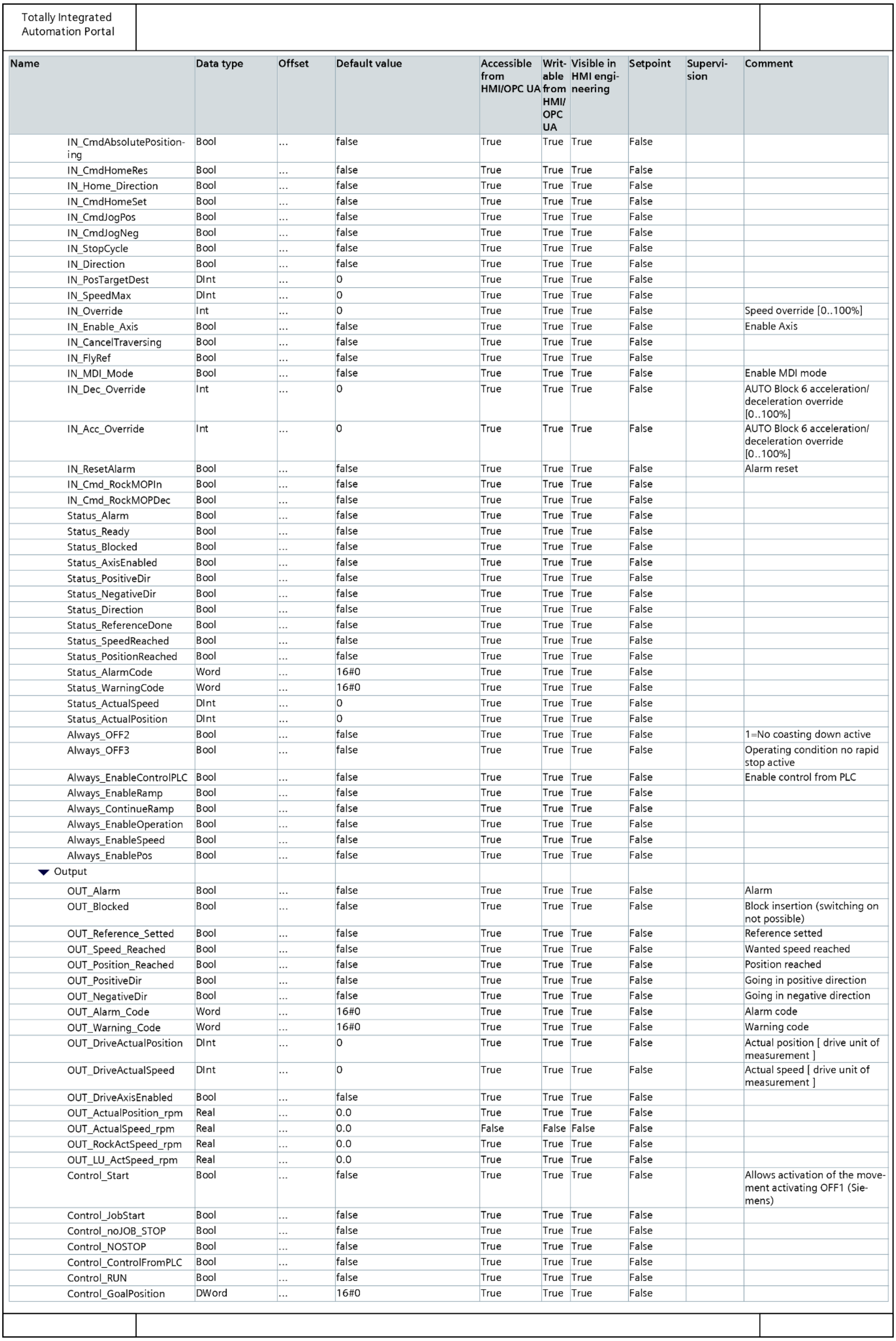
Totally Integrated Automation Portal

| Name | Data type | Offset | Default value | Accessible from HMI/OPC UA | Writ-able from HMI/OPC UA | Visible in HMI engi-neering | Setpoint | Supervi-sion | Comment |
|---|---|---|---|---|---|---|---|---|---|
| Bit15 | Bool | ... | false | False | False | False | False | | |
| Off1 | Bool | ... | false | False | False | False | False | | |
| Off2 | Bool | ... | false | False | False | False | False | | |
| Off3 | Bool | ... | false | False | False | False | False | | |
| Enc | Bool | ... | false | False | False | False | False | | |
| RejTrvTsk | Bool | ... | false | False | False | False | False | | |
| IntMStop | Bool | ... | false | False | False | False | False | | |
| TrvStart | Bool | ... | false | False | False | False | False | | |
| AckFault | Bool | ... | false | False | False | False | False | | |
| ▼ EPosSTW 1 | Struct | ... | | False | False | False | False | | |
| MdiTyp | Bool | ... | false | False | False | False | False | | |
| MdiPos | Bool | ... | false | False | False | False | False | | |
| MdiNeg | Bool | ... | false | False | False | False | False | | |
| Bit11 | Bool | ... | false | False | False | False | False | | |
| MdiTrTyp | Bool | ... | false | False | False | False | False | | |
| Bit13 | Bool | ... | false | False | False | False | False | | |
| MdiSetup | Bool | ... | false | False | False | False | False | | |
| MdiStart | Bool | ... | false | False | False | False | False | | |
| TrvBit0 | Bool | ... | false | False | False | False | False | | |
| TrvBit1 | Bool | ... | false | False | False | False | False | | |
| TrvBit2 | Bool | ... | false | False | False | False | False | | |
| TrvBit3 | Bool | ... | false | False | False | False | False | | |
| TrvBit4 | Bool | ... | false | False | False | False | False | | |
| TrvBit5 | Bool | ... | false | False | False | False | False | | |
| Bit6 | Bool | ... | false | False | False | False | False | | |
| Bit7 | Bool | ... | false | False | False | False | False | | |
| ▼ EPosSTW 2 | Struct | ... | | False | False | False | False | | |
| RefTyp | Bool | ... | false | False | False | False | False | | |
| RefStDi | Bool | ... | false | False | False | False | False | | |
| RefInps | Bool | ... | false | False | False | False | False | | |
| RefEdge | Bool | ... | false | False | False | False | False | | |
| Bit12 | Bool | ... | false | False | False | False | False | | |
| Bit13 | Bool | ... | false | False | False | False | False | | |
| SftLimAct | Bool | ... | false | False | False | False | False | | |
| StpCamAct | Bool | ... | false | False | False | False | False | | |
| TrkMode | Bool | ... | false | False | False | False | False | | |
| SetRefPt | Bool | ... | false | False | False | False | False | | |
| ActRefCam | Bool | ... | false | False | False | False | False | | |
| Bit3 | Bool | ... | false | False | False | False | False | | |
| Bit4 | Bool | ... | false | False | False | False | False | | |
| JogInc | Bool | ... | false | False | False | False | False | | |
| Bit6 | Bool | ... | false | False | False | False | False | | |
| Bit7 | Bool | ... | false | False | False | False | False | | |
| ▼ STW2 | Struct | ... | | False | False | False | False | | |
| TrvFixedStp | Bool | ... | false | False | False | False | False | | |
| GlbTrgCom | Bool | ... | false | False | False | False | False | | |
| Bit10 | Bool | ... | false | False | False | False | False | | |
| MotSwOver | Bool | ... | false | False | False | False | False | | |
| MsZyBit0 | Bool | ... | false | False | False | False | False | | |
| MsZyBit1 | Bool | ... | false | False | False | False | False | | |
| MsZyBit2 | Bool | ... | false | False | False | False | False | | |
| MsZyBit3 | Bool | ... | false | False | False | False | False | | |
| DDSBit0 | Bool | ... | false | False | False | False | False | | |
| DDSBit1 | Bool | ... | false | False | False | False | False | | |
| DDSBit2 | Bool | ... | false | False | False | False | False | | |
| DDSBit3 | Bool | ... | false | False | False | False | False | | |
| DDSBit4 | Bool | ... | false | False | False | False | False | | |
| GlbStart | Bool | ... | false | False | False | False | False | | |
| ReslComp | Bool | ... | false | False | False | False | False | | |
| ActPrkAxis | Bool | ... | false | False | False | False | False | | |
| OverrideV | Word | ... | 16#0 | False | False | False | False | | |
| Position | DWord | ... | 16#0 | False | False | False | False | | |
| Velocity | DWord | ... | 16#0 | False | False | False | False | | |
| OverrideA | Word | ... | 16#0 | False | False | False | False | | |
| OverrideD | Word | ... | 16#0 | False | False | False | False | | |
| Word12 | Word | ... | 16#0 | False | False | False | False | | |
| ▼ STATUS WORD | Struct | ... | | False | False | False | False | | |
| ▼ Status_Word_1 | Struct | ... | | False | False | False | False | | |
| NoFlwErr | Bool | ... | false | False | False | False | False | | |
| LbCr | Bool | ... | false | False | False | False | False | | |
| TargPos | Bool | ... | false | False | False | False | False | | |
| RefPset | Bool | ... | false | False | False | False | False | | |
| TrvTskAck | Bool | ... | false | False | False | False | False | | |



Totally Integrated Automation Portal

| Name | Data type | Offset | Default value | Accessible from HMI/OPC UA | Writ-able from HMI/OPC UA | Visible in HMI engi-neering | Setpoint | Supervi-sion | Comment |
|---|---|---|---|---|---|---|---|---|---|
| StndStill | Bool | ... | false | False | False | False | False | | |
| Accel | Bool | ... | false | False | False | False | False | | |
| Decel | Bool | ... | false | False | False | False | False | | |
| RTS | Bool | ... | false | False | False | False | False | | |
| RDY | Bool | ... | false | False | False | False | False | | |
| IOp | Bool | ... | false | False | False | False | False | | |
| Fault | Bool | ... | false | False | False | False | False | | |
| NoOff2Act | Bool | ... | false | False | False | False | False | | |
| NoOff3Act | Bool | ... | false | False | False | False | False | | |
| PowInhbt | Bool | ... | false | False | False | False | False | | |
| Alarm | Bool | ... | false | False | False | False | False | | |
| ▼ EPosZSW 1 | Struct | ... | | False | False | False | False | | |
| ActTrvBit0 | Bool | ... | false | False | False | False | False | | |
| ActTrvBit1 | Bool | ... | false | False | False | False | False | | |
| ActTrvBit2 | Bool | ... | false | False | False | False | False | | |
| ActTrvBit3 | Bool | ... | false | False | False | False | False | | |
| ActTrvBit4 | Bool | ... | false | False | False | False | False | | |
| ActTrvBit5 | Bool | ... | false | False | False | False | False | | |
| Bit6 | Bool | ... | false | False | False | False | False | | |
| Bit7 | Bool | ... | false | False | False | False | False | | |
| StpCamMinAct | Bool | ... | false | False | False | False | False | | |
| StpCamPlsAct | Bool | ... | false | False | False | False | False | | |
| JogAct | Bool | ... | false | False | False | False | False | | |
| RefAct | Bool | ... | false | False | False | False | False | | |
| FlyRefAct | Bool | ... | false | False | False | False | False | | |
| TrvBlact | Bool | ... | false | False | False | False | False | | |
| MdiStupAct | Bool | ... | false | False | False | False | False | | |
| MdiPosAct | Bool | ... | false | False | False | False | False | | |
| ▼ EPosZSW 2 | Struct | ... | | False | False | False | False | | |
| PosSimCam1 | Bool | ... | false | False | False | False | False | | |
| PosSimCam2 | Bool | ... | false | False | False | False | False | | |
| TrvOut1 | Bool | ... | false | False | False | False | False | | |
| TrvOut2 | Bool | ... | false | False | False | False | False | | |
| FxStpRd | Bool | ... | false | False | False | False | False | | |
| FxStpTrRd | Bool | ... | false | False | False | False | False | | |
| TrvFxStpAct | Bool | ... | false | False | False | False | False | | |
| CmdAct | Bool | ... | false | False | False | False | False | | |
| TrkModeAct | Bool | ... | false | False | False | False | False | | |
| VeloLimAct | Bool | ... | false | False | False | False | False | | |
| SetPStat | Bool | ... | false | False | False | False | False | | |
| PrntMrkOut | Bool | ... | false | False | False | False | False | | |
| FWD | Bool | ... | false | False | False | False | False | | |
| BWD | Bool | ... | false | False | False | False | False | | |
| SftSwMinAct | Bool | ... | false | False | False | False | False | | |
| SftSwPlsAct | Bool | ... | false | False | False | False | False | | |
| ▼ Status_Word_2 | Struct | ... | | False | False | False | False | | |
| Bit8 | Bool | ... | false | False | False | False | False | | |
| GlbTrgReq | Bool | ... | false | False | False | False | False | | |
| PulsEn | Bool | ... | false | False | False | False | False | | |
| MotSwOverAct | Bool | ... | false | False | False | False | False | | |
| SlvZykBit0 | Bool | ... | false | False | False | False | False | | |
| SlvZykBit1 | Bool | ... | false | False | False | False | False | | |
| SlvZykBit2 | Bool | ... | false | False | False | False | False | | |
| SlvZykBit3 | Bool | ... | false | False | False | False | False | | |
| ActDDSBit0 | Bool | ... | false | False | False | False | False | | |
| ActDDSBit1 | Bool | ... | false | False | False | False | False | | |
| ActDDSBit2 | Bool | ... | false | False | False | False | False | | |
| ActDDSBit3 | Bool | ... | false | False | False | False | False | | |
| ActDDSBit4 | Bool | ... | false | False | False | False | False | | |
| CmdActRelBrk | Bool | ... | false | False | False | False | False | | |
| TrqContMode | Bool | ... | false | False | False | False | False | | |
| ParkAxisAct | Bool | ... | false | False | False | False | False | | |
| Word6 | Word | ... | 16#0 | False | False | False | False | | |
| Position | DWord | ... | 16#0 | False | False | False | False | | |
| Velocity | DWord | ... | 16#0 | False | False | False | False | | |
| ErrNr | Word | ... | 16#0 | False | False | False | False | | |
| WarnNr | Word | ... | 16#0 | False | False | False | False | | |
| Reserved | Word | ... | 16#0 | False | False | False | False | | |
| ▼ FB_POS_SPEED_Instance | "R101_Con-trol_Interface" | ... | | True | True | True | False | | |
| ▼ Input | | | | | | | | | |
| IN_CmdSpeed | Bool | ... | false | True | True | True | False | | |
| IN_CmdJogPos_Speed | Bool | ... | false | True | True | True | False | | |
| IN_CmdJogNeg_Speed | Bool | ... | false | True | True | True | False | | |

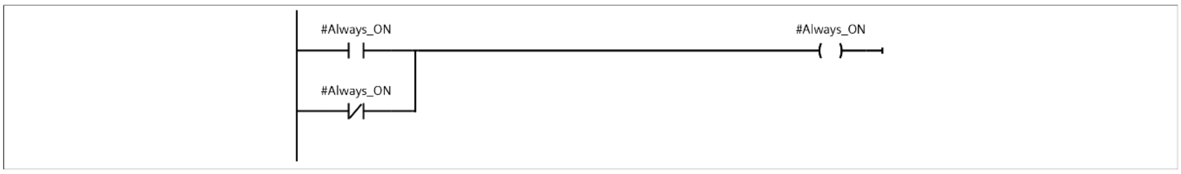


Totally Integrated Automation Portal

| Name | Data type | Offset | Default value | Accessible from HMI/OPC UA | Writable from HMI/OPC UA | Visible in HMI engineering | Setpoint | Supervision | Comment |
|---|---|---|---|---|---|---|---|---|---|
| IN_CmdAbsolutePositioning | Bool | ... | false | True | True | True | False | | |
| IN_CmdHomeRes | Bool | ... | false | True | True | True | False | | |
| IN_Home_Direction | Bool | ... | false | True | True | True | False | | |
| IN_CmdHomeSet | Bool | ... | false | True | True | True | False | | |
| IN_CmdJogPos | Bool | ... | false | True | True | True | False | | |
| IN_CmdJogNeg | Bool | ... | false | True | True | True | False | | |
| IN_StopCycle | Bool | ... | false | True | True | True | False | | |
| IN_Direction | Bool | ... | false | True | True | True | False | | |
| IN_PosTargetDest | DInt | ... | 0 | True | True | True | False | | |
| IN_SpeedMax | DInt | ... | 0 | True | True | True | False | | |
| IN_Override | Int | ... | 0 | True | True | True | False | | Speed override [0..100%] |
| IN_Enable_Axis | Bool | ... | false | True | True | True | False | | Enable Axis |
| IN_CancelTraversing | Bool | ... | false | True | True | True | False | | |
| IN_FlyRef | Bool | ... | false | True | True | True | False | | |
| IN_MDI_Mode | Bool | ... | false | True | True | True | False | | Enable MDI mode |
| IN_Dec_Override | Int | ... | 0 | True | True | True | False | | AUTO Block 6 acceleration/ deceleration override [0..100%] |
| IN_Acc_Override | Int | ... | 0 | True | True | True | False | | AUTO Block 6 acceleration/ deceleration override [0..100%] |
| IN_ResetAlarm | Bool | ... | false | True | True | True | False | | Alarm reset |
| IN_Cmd_RockMOPIn | Bool | ... | false | True | True | True | False | | |
| IN_Cmd_RockMOPDec | Bool | ... | false | True | True | True | False | | |
| Status_Alarm | Bool | ... | false | True | True | True | False | | |
| Status_Ready | Bool | ... | false | True | True | True | False | | |
| Status_Blocked | Bool | ... | false | True | True | True | False | | |
| Status_AxisEnabled | Bool | ... | false | True | True | True | False | | |
| Status_PositiveDir | Bool | ... | false | True | True | True | False | | |
| Status_NegativeDir | Bool | ... | false | True | True | True | False | | |
| Status_Direction | Bool | ... | false | True | True | True | False | | |
| Status_ReferenceDone | Bool | ... | false | True | True | True | False | | |
| Status_SpeedReached | Bool | ... | false | True | True | True | False | | |
| Status_PositionReached | Bool | ... | false | True | True | True | False | | |
| Status_AlarmCode | Word | ... | 16#0 | True | True | True | False | | |
| Status_WarningCode | Word | ... | 16#0 | True | True | True | False | | |
| Status_ActualSpeed | DInt | ... | 0 | True | True | True | False | | |
| Status_ActualPosition | DInt | ... | 0 | True | True | True | False | | |
| Always_OFF2 | Bool | ... | false | True | True | True | False | | 1=No coasting down active |
| Always_OFF3 | Bool | ... | false | True | True | True | False | | Operating condition no rapid stop active |
| Always_EnableControlPLC | Bool | ... | false | True | True | True | False | | Enable control from PLC |
| Always_EnableRamp | Bool | ... | false | True | True | True | False | | |
| Always_ContinueRamp | Bool | ... | false | True | True | True | False | | |
| Always_EnableOperation | Bool | ... | false | True | True | True | False | | |
| Always_EnableSpeed | Bool | ... | false | True | True | True | False | | |
| Always_EnablePos | Bool | ... | false | True | True | True | False | | |
| ▼ Output | | | | | | | | | |
| OUT_Alarm | Bool | ... | false | True | True | True | False | | Alarm |
| OUT_Blocked | Bool | ... | false | True | True | True | False | | Block insertion (switching on not possible) |
| OUT_Reference_Setted | Bool | ... | false | True | True | True | False | | Reference setted |
| OUT_Speed_Reached | Bool | ... | false | True | True | True | False | | Wanted speed reached |
| OUT_Position_Reached | Bool | ... | false | True | True | True | False | | Position reached |
| OUT_PositiveDir | Bool | ... | false | True | True | True | False | | Going in positive direction |
| OUT_NegativeDir | Bool | ... | false | True | True | True | False | | Going in negative direction |
| OUT_Alarm_Code | Word | ... | 16#0 | True | True | True | False | | Alarm code |
| OUT_Warning_Code | Word | ... | 16#0 | True | True | True | False | | Warning code |
| OUT_DriveActualPosition | DInt | ... | 0 | True | True | True | False | | Actual position [ drive unit of measurement ] |
| OUT_DriveActualSpeed | DInt | ... | 0 | True | True | True | False | | Actual speed [ drive unit of measurement ] |
| OUT_DriveAxisEnabled | Bool | ... | false | True | True | True | False | | |
| OUT_ActualPosition_rpm | Real | ... | 0.0 | True | True | True | False | | |
| OUT_ActualSpeed_rpm | Real | ... | 0.0 | False | False | False | False | | |
| OUT_RockActSpeed_rpm | Real | ... | 0.0 | True | True | True | False | | |
| OUT_LU_ActSpeed_rpm | Real | ... | 0.0 | True | True | True | False | | |
| Control_Start | Bool | ... | false | True | True | True | False | | Allows activation of the movement activating OFF1 (Siemens) |
| Control_JobStart | Bool | ... | false | True | True | True | False | | |
| Control_noJOB_STOP | Bool | ... | false | True | True | True | False | | |
| Control_NOSTOP | Bool | ... | false | True | True | True | False | | |
| Control_ControlFromPLC | Bool | ... | false | True | True | True | False | | |
| Control_RUN | Bool | ... | false | True | True | True | False | | |
| Control_GoalPosition | DWord | ... | 16#0 | True | True | True | False | | |

| Name | Data type | Offset | Default value | Accessible from HMI/OPC UA | Writable from HMI/OPC UA | Visible in HMI engineering | Setpoint | Supervision | Comment |
|---|---|---|---|---|---|---|---|---|---|
| Control_SpeedPosition | DInt | ... | 0 | True | True | True | False | | |
| Control_OverrideV | Int | ... | 0 | True | True | True | False | | |
| Control_OverridePosV | Int | ... | 0 | True | True | True | False | | |
| Control_Acceleration | Int | ... | 0 | True | True | True | False | | |
| Control_Deceleration | Int | ... | 0 | True | True | True | False | | |
| Control_GoalSpeed_LU | Real | ... | 0.0 | True | True | True | False | | |
| Control_GoalSpeed_Hz | Int | ... | 0 | True | True | True | False | | |
| Control_GoalSpeed_rpm | Int | ... | 0 | True | True | True | False | | |
| Control_PositioningMode | Bool | ... | false | True | True | True | False | | |
| Control_ReferenceSearchStart | Bool | ... | false | True | True | True | False | | |
| Control_PositiveDir | Bool | ... | false | True | True | True | False | | |
| Control_NegativeDir | Bool | ... | false | True | True | True | False | | |
| Control_RefType | Bool | ... | false | True | True | True | False | | |
| Control_ReferenceSearchDir | Bool | ... | false | True | True | True | False | | |
| Control_Jog1 | Bool | ... | false | True | True | True | False | | |
| Control_Jog2 | Bool | ... | false | True | True | True | False | | |
| Control_AlarmReset | Bool | ... | false | True | True | True | False | | |
| Control_SetReferencePoint | Bool | ... | false | True | True | True | False | | Set Zero position |
| Control_SpeedDirection | Bool | ... | false | True | True | True | False | | |
| Control_EnableSpeed | Bool | ... | false | True | True | True | False | | |
| Control_OFF2 | Bool | ... | false | True | True | True | False | | |
| Control_OFF3 | Bool | ... | false | True | True | True | False | | |
| Control_EnableOperation | Bool | ... | false | True | True | True | False | | |
| Control_EnableRamp | Bool | ... | false | True | True | True | False | | |
| Control_ContinueRamp | Bool | ... | false | True | True | True | False | | |
| Control_EnablePos | Bool | ... | false | True | True | True | False | | |
| Control_RockMOPIn | Bool | ... | false | True | True | True | False | | |
| Control_RockMOPDec | Bool | ... | false | True | True | True | False | | |
| Control_RockJOG | Bool | ... | false | True | True | True | False | | |
| Control_RockNegative | Bool | ... | false | True | True | True | False | | |
| Control_RockPositive | Bool | ... | false | True | True | True | False | | |
| Control_Stop | Bool | ... | false | True | True | True | False | | |
| InOut | | | | | | | | | |
| ▼ Static | | | | | | | | | |
| Always_ON | Bool | ... | false | False | False | False | False | | |
| StationOK | Bool | ... | false | True | True | True | False | | |
| ▼ Temp | | | | | | | | | |
| AUX_BYTE_1 | Byte | ... | | | | | | | Memory |
| AUX_RET_VAL_INT_1 | Int | ... | | | | | | | Error message |
| Constant | | | | | | | | | |

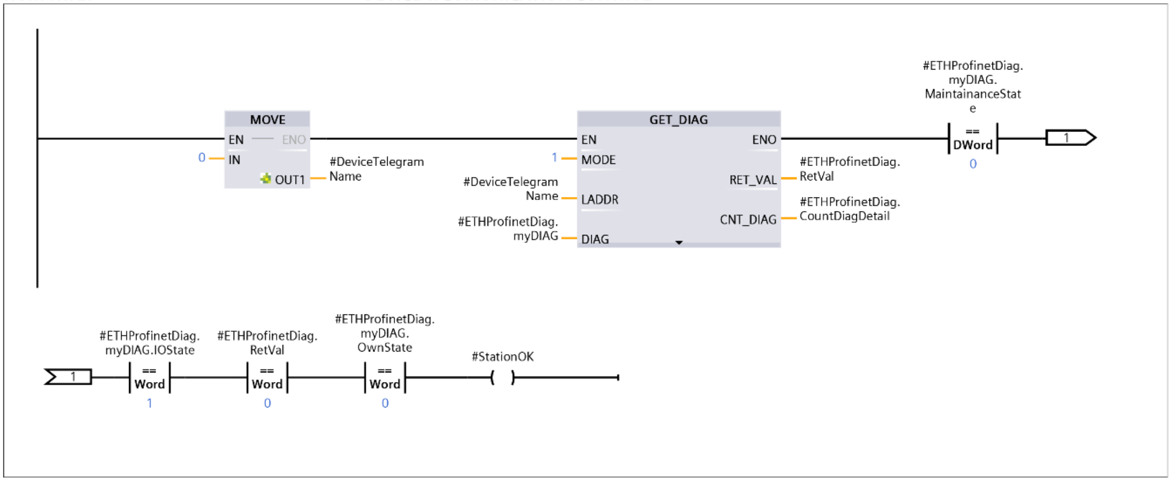
**Network 1: Always_ON**



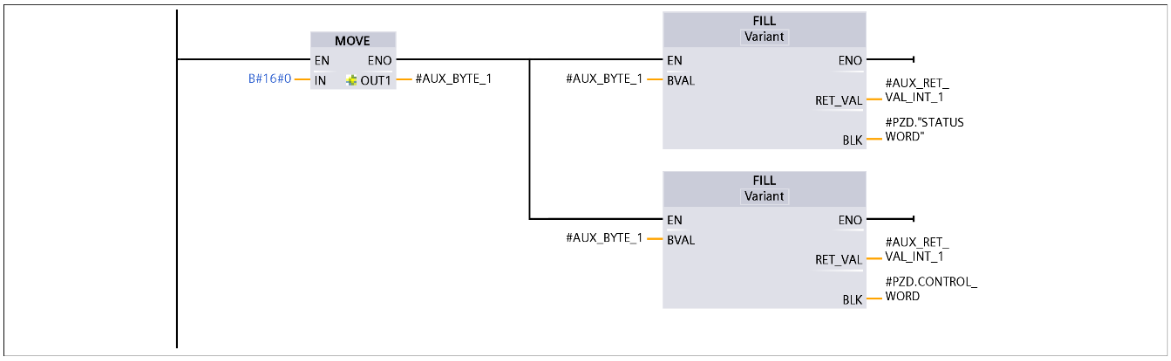
**Network 2: ****************************DEVICE & COMUNICATION CONTROL********************************

Totally Integrated
Automation Portal

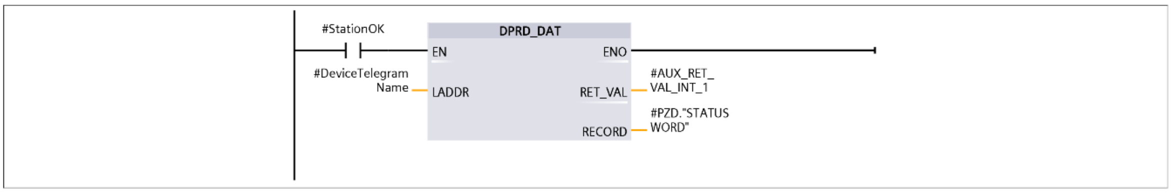
Network 2: ***************************DEVICE & COMUNICATION CONTROL****************************



Network 3: --- Reset of COMMUNICATION TELEGRAM BUFFER



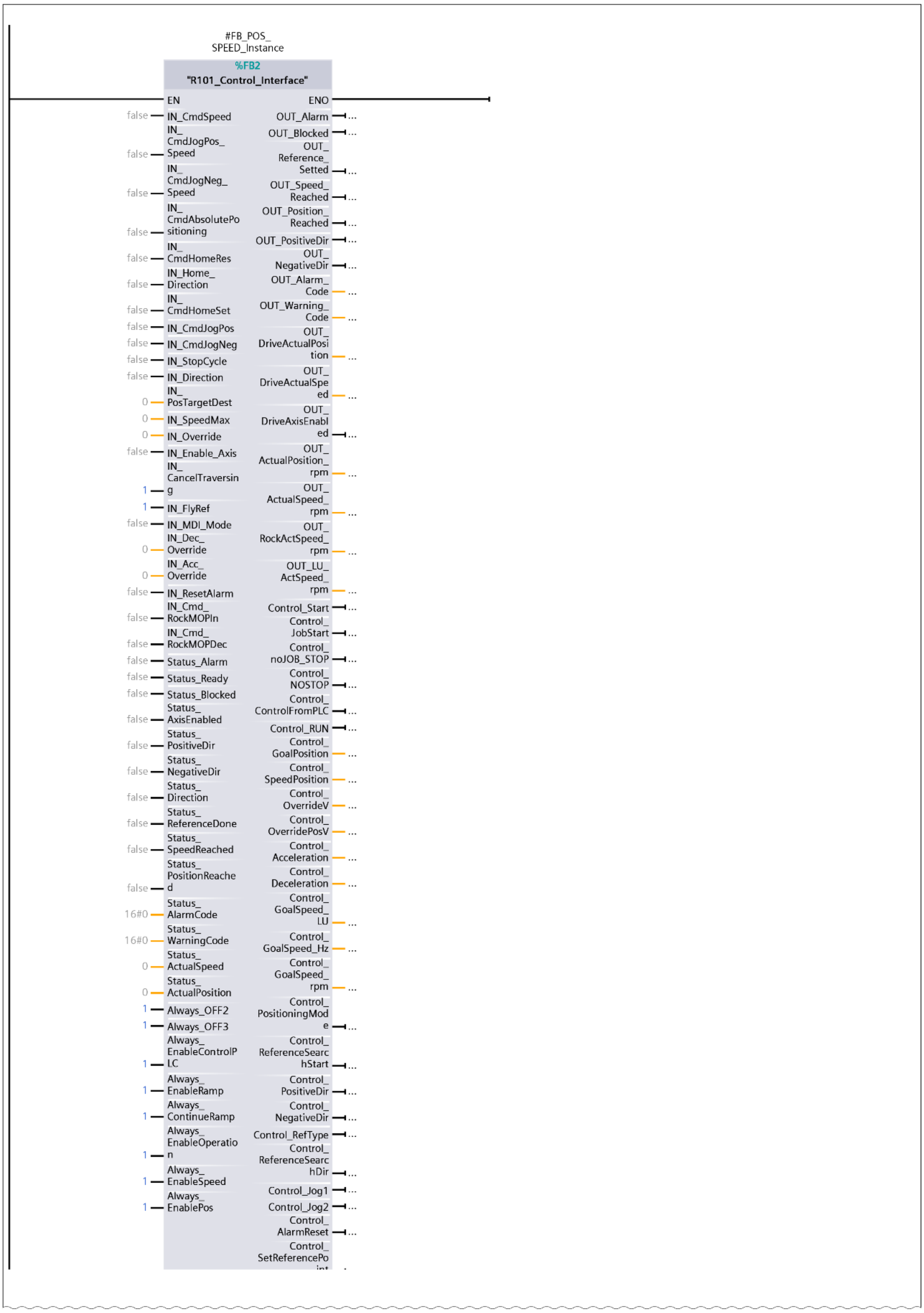
Network 4: READ status from the unit



Network 5: ------------------------------------------------------------------------------------------------------------------------------------------------------

Network 6: Standard program

Network 6: Standard program (1.1 / 2.1)

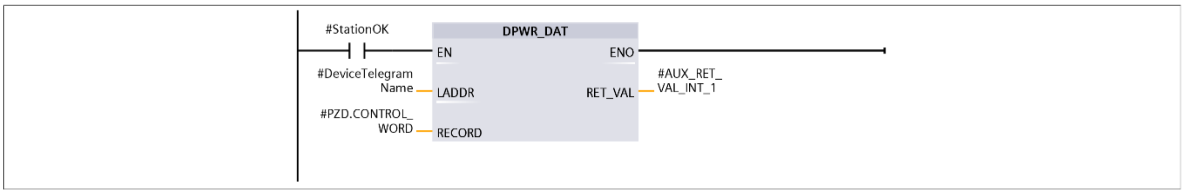


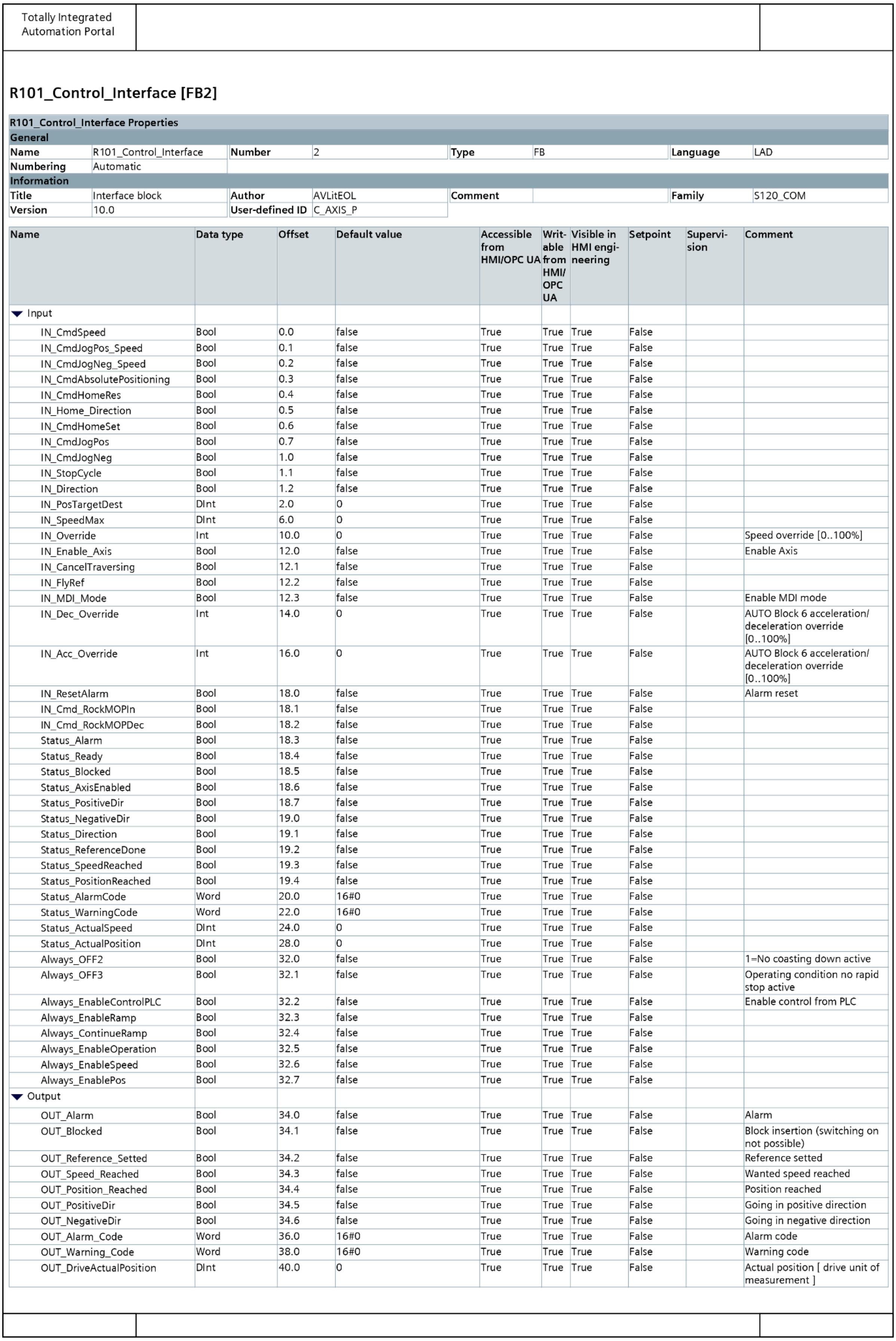
Network 6: Standard program (2.1 / 2.1)

1.1 ( Page1 - 10)
int
Control_ SpeedDirection
Control_ EnableSpeed
Control_OFF2
Control_OFF3
Control_ EnableOperation
Control_ EnableRamp
Control_ ContinueRamp
Control_ EnablePos
Control_ RockMOPIn
Control_ RockMOPDec
Control_ RockJOG
Control_ RockNegative
Control_ RockPositive
Control_Stop
...

**Network 7:** -----------------------------------------------------------------------------------------------------------------------------------------------

**Network 8: WRITE commands to the unit**



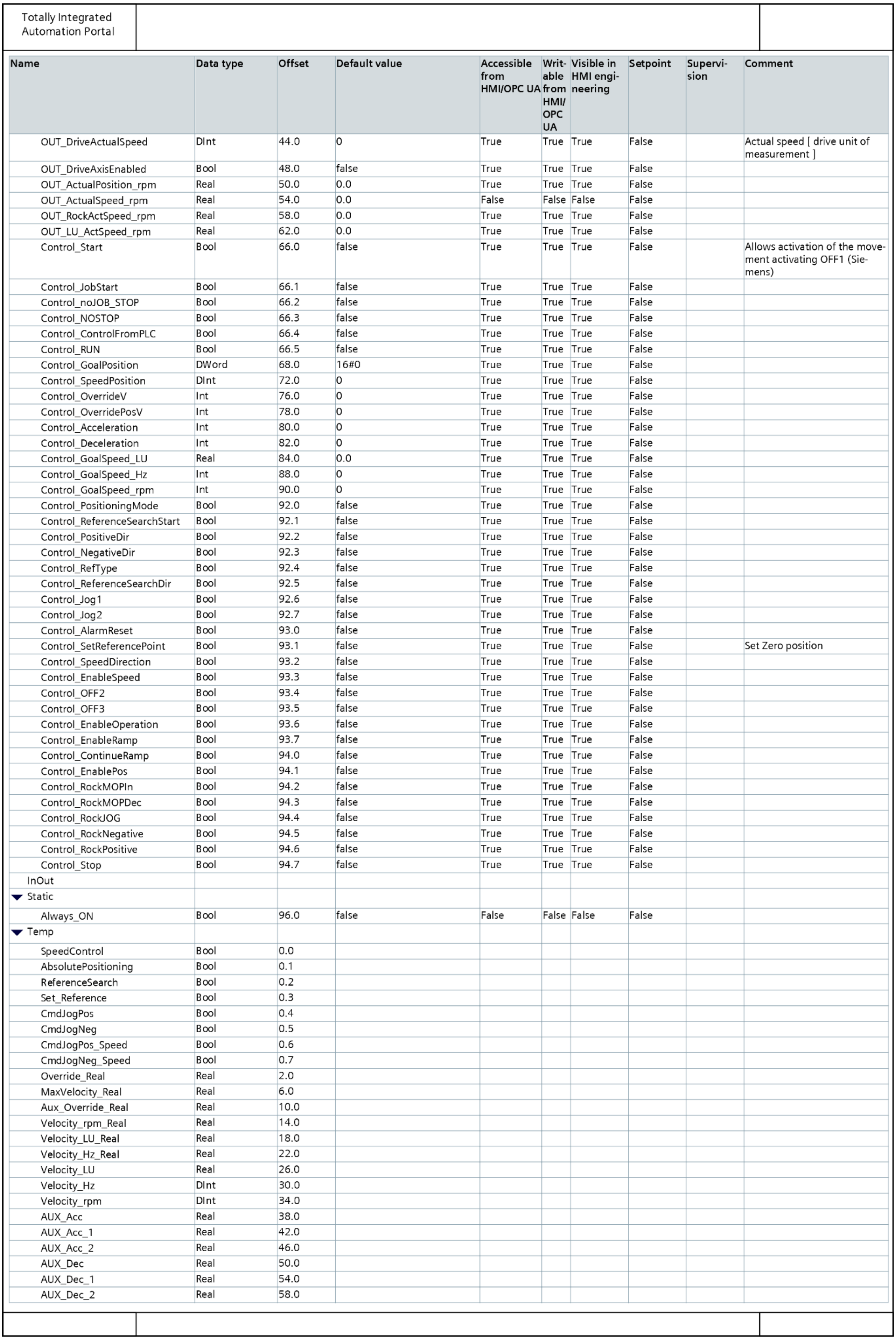


Totally Integrated Automation Portal

## R101_Control_Interface [FB2]

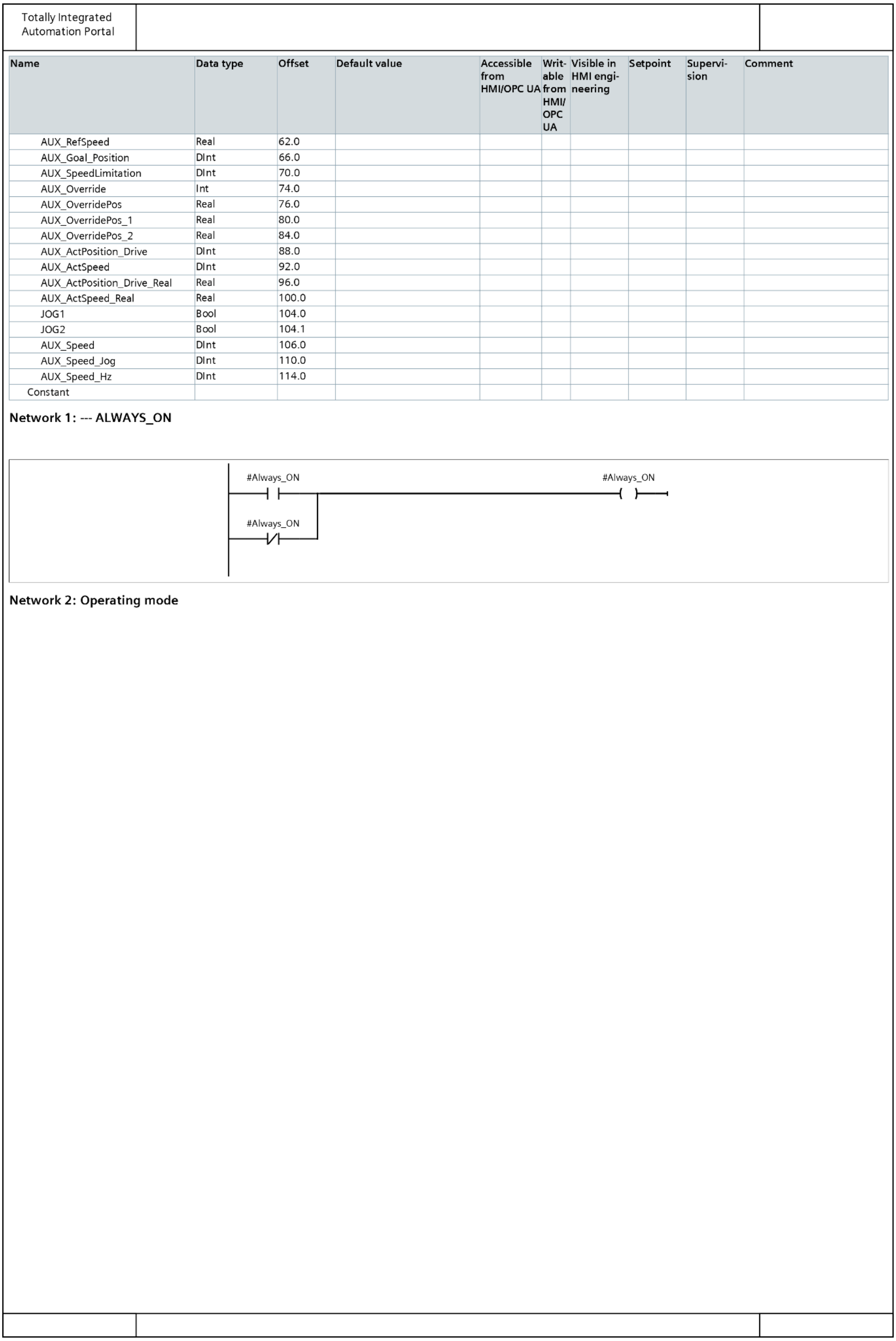
| R101_Control_Interface Properties | | | | | | | |
|---|---|---|---|---|---|---|---|
| **General** | | | | | | | |
| **Name** | R101_Control_Interface | **Number** | 2 | **Type** | FB | **Language** | LAD |
| **Numbering** | Automatic | | | | | | |
| **Information** | | | | | | | |
| **Title** | Interface block | **Author** | AVLitEOL | **Comment** | | **Family** | S120_COM |
| **Version** | 10.0 | **User-defined ID** | C_AXIS_P | | | | |

| Name | Data type | Offset | Default value | Accessible from HMI/OPC UA | Writable from HMI/ OPC UA | Visible in HMI engineering | Setpoint | Supervision | Comment |
|---|---|---|---|---|---|---|---|---|---|
| ▼ Input | | | | | | | | | |
| IN_CmdSpeed | Bool | 0.0 | false | True | True | True | False | | |
| IN_CmdJogPos_Speed | Bool | 0.1 | false | True | True | True | False | | |
| IN_CmdJogNeg_Speed | Bool | 0.2 | false | True | True | True | False | | |
| IN_CmdAbsolutePositioning | Bool | 0.3 | false | True | True | True | False | | |
| IN_CmdHomeRes | Bool | 0.4 | false | True | True | True | False | | |
| IN_Home_Direction | Bool | 0.5 | false | True | True | True | False | | |
| IN_CmdHomeSet | Bool | 0.6 | false | True | True | True | False | | |
| IN_CmdJogPos | Bool | 0.7 | false | True | True | True | False | | |
| IN_CmdJogNeg | Bool | 1.0 | false | True | True | True | False | | |
| IN_StopCycle | Bool | 1.1 | false | True | True | True | False | | |
| IN_Direction | Bool | 1.2 | false | True | True | True | False | | |
| IN_PosTargetDest | DInt | 2.0 | 0 | True | True | True | False | | |
| IN_SpeedMax | DInt | 6.0 | 0 | True | True | True | False | | |
| IN_Override | Int | 10.0 | 0 | True | True | True | False | | Speed override [0..100%] |
| IN_Enable_Axis | Bool | 12.0 | false | True | True | True | False | | Enable Axis |
| IN_CancelTraversing | Bool | 12.1 | false | True | True | True | False | | |
| IN_FlyRef | Bool | 12.2 | false | True | True | True | False | | |
| IN_MDI_Mode | Bool | 12.3 | false | True | True | True | False | | Enable MDI mode |
| IN_Dec_Override | Int | 14.0 | 0 | True | True | True | False | | AUTO Block 6 acceleration/ deceleration override [0..100%] |
| IN_Acc_Override | Int | 16.0 | 0 | True | True | True | False | | AUTO Block 6 acceleration/ deceleration override [0..100%] |
| IN_ResetAlarm | Bool | 18.0 | false | True | True | True | False | | Alarm reset |
| IN_Cmd_RockMOPIn | Bool | 18.1 | false | True | True | True | False | | |
| IN_Cmd_RockMOPDec | Bool | 18.2 | false | True | True | True | False | | |
| Status_Alarm | Bool | 18.3 | false | True | True | True | False | | |
| Status_Ready | Bool | 18.4 | false | True | True | True | False | | |
| Status_Blocked | Bool | 18.5 | false | True | True | True | False | | |
| Status_AxisEnabled | Bool | 18.6 | false | True | True | True | False | | |
| Status_PositiveDir | Bool | 18.7 | false | True | True | True | False | | |
| Status_NegativeDir | Bool | 19.0 | false | True | True | True | False | | |
| Status_Direction | Bool | 19.1 | false | True | True | True | False | | |
| Status_ReferenceDone | Bool | 19.2 | false | True | True | True | False | | |
| Status_SpeedReached | Bool | 19.3 | false | True | True | True | False | | |
| Status_PositionReached | Bool | 19.4 | false | True | True | True | False | | |
| Status_AlarmCode | Word | 20.0 | 16#0 | True | True | True | False | | |
| Status_WarningCode | Word | 22.0 | 16#0 | True | True | True | False | | |
| Status_ActualSpeed | DInt | 24.0 | 0 | True | True | True | False | | |
| Status_ActualPosition | DInt | 28.0 | 0 | True | True | True | False | | |
| Always_OFF2 | Bool | 32.0 | false | True | True | True | False | | 1=No coasting down active |
| Always_OFF3 | Bool | 32.1 | false | True | True | True | False | | Operating condition no rapid stop active |
| Always_EnableControlPLC | Bool | 32.2 | false | True | True | True | False | | Enable control from PLC |
| Always_EnableRamp | Bool | 32.3 | false | True | True | True | False | | |
| Always_ContinueRamp | Bool | 32.4 | false | True | True | True | False | | |
| Always_EnableOperation | Bool | 32.5 | false | True | True | True | False | | |
| Always_EnableSpeed | Bool | 32.6 | false | True | True | True | False | | |
| Always_EnablePos | Bool | 32.7 | false | True | True | True | False | | |
| ▼ Output | | | | | | | | | |
| OUT_Alarm | Bool | 34.0 | false | True | True | True | False | | Alarm |
| OUT_Blocked | Bool | 34.1 | false | True | True | True | False | | Block insertion (switching on not possible) |
| OUT_Reference_Setted | Bool | 34.2 | false | True | True | True | False | | Reference setted |
| OUT_Speed_Reached | Bool | 34.3 | false | True | True | True | False | | Wanted speed reached |
| OUT_Position_Reached | Bool | 34.4 | false | True | True | True | False | | Position reached |
| OUT_PositiveDir | Bool | 34.5 | false | True | True | True | False | | Going in positive direction |
| OUT_NegativeDir | Bool | 34.6 | false | True | True | True | False | | Going in negative direction |
| OUT_Alarm_Code | Word | 36.0 | 16#0 | True | True | True | False | | Alarm code |
| OUT_Warning_Code | Word | 38.0 | 16#0 | True | True | True | False | | Warning code |
| OUT_DriveActualPosition | DInt | 40.0 | 0 | True | True | True | False | | Actual position [ drive unit of measurement ] |



Totally Integrated Automation Portal

| Name | Data type | Offset | Default value | Accessible from HMI/OPC UA | Writable from HMI/OPC UA | Visible in HMI engineering | Setpoint | Supervision | Comment |
|---|---|---|---|---|---|---|---|---|---|
| OUT_DriveActualSpeed | DInt | 44.0 | 0 | True | True | True | False |  | Actual speed [ drive unit of measurement ] |
| OUT_DriveAxisEnabled | Bool | 48.0 | false | True | True | True | False |  |  |
| OUT_ActualPosition_rpm | Real | 50.0 | 0.0 | True | True | True | False |  |  |
| OUT_ActualSpeed_rpm | Real | 54.0 | 0.0 | False | False | False | False |  |  |
| OUT_RockActSpeed_rpm | Real | 58.0 | 0.0 | True | True | True | False |  |  |
| OUT_LU_ActSpeed_rpm | Real | 62.0 | 0.0 | True | True | True | False |  |  |
| Control_Start | Bool | 66.0 | false | True | True | True | False |  | Allows activation of the movement activating OFF1 (Siemens) |
| Control_JobStart | Bool | 66.1 | false | True | True | True | False |  |  |
| Control_noJOB_STOP | Bool | 66.2 | false | True | True | True | False |  |  |
| Control_NOSTOP | Bool | 66.3 | false | True | True | True | False |  |  |
| Control_ControlFromPLC | Bool | 66.4 | false | True | True | True | False |  |  |
| Control_RUN | Bool | 66.5 | false | True | True | True | False |  |  |
| Control_GoalPosition | DWord | 68.0 | 16#0 | True | True | True | False |  |  |
| Control_SpeedPosition | DInt | 72.0 | 0 | True | True | True | False |  |  |
| Control_OverrideV | Int | 76.0 | 0 | True | True | True | False |  |  |
| Control_OverridePosV | Int | 78.0 | 0 | True | True | True | False |  |  |
| Control_Acceleration | Int | 80.0 | 0 | True | True | True | False |  |  |
| Control_Deceleration | Int | 82.0 | 0 | True | True | True | False |  |  |
| Control_GoalSpeed_LU | Real | 84.0 | 0.0 | True | True | True | False |  |  |
| Control_GoalSpeed_Hz | Int | 88.0 | 0 | True | True | True | False |  |  |
| Control_GoalSpeed_rpm | Int | 90.0 | 0 | True | True | True | False |  |  |
| Control_PositioningMode | Bool | 92.0 | false | True | True | True | False |  |  |
| Control_ReferenceSearchStart | Bool | 92.1 | false | True | True | True | False |  |  |
| Control_PositiveDir | Bool | 92.2 | false | True | True | True | False |  |  |
| Control_NegativeDir | Bool | 92.3 | false | True | True | True | False |  |  |
| Control_RefType | Bool | 92.4 | false | True | True | True | False |  |  |
| Control_ReferenceSearchDir | Bool | 92.5 | false | True | True | True | False |  |  |
| Control_Jog1 | Bool | 92.6 | false | True | True | True | False |  |  |
| Control_Jog2 | Bool | 92.7 | false | True | True | True | False |  |  |
| Control_AlarmReset | Bool | 93.0 | false | True | True | True | False |  |  |
| Control_SetReferencePoint | Bool | 93.1 | false | True | True | True | False |  | Set Zero position |
| Control_SpeedDirection | Bool | 93.2 | false | True | True | True | False |  |  |
| Control_EnableSpeed | Bool | 93.3 | false | True | True | True | False |  |  |
| Control_OFF2 | Bool | 93.4 | false | True | True | True | False |  |  |
| Control_OFF3 | Bool | 93.5 | false | True | True | True | False |  |  |
| Control_EnableOperation | Bool | 93.6 | false | True | True | True | False |  |  |
| Control_EnableRamp | Bool | 93.7 | false | True | True | True | False |  |  |
| Control_ContinueRamp | Bool | 94.0 | false | True | True | True | False |  |  |
| Control_EnablePos | Bool | 94.1 | false | True | True | True | False |  |  |
| Control_RockMOPIn | Bool | 94.2 | false | True | True | True | False |  |  |
| Control_RockMOPDec | Bool | 94.3 | false | True | True | True | False |  |  |
| Control_RockJOG | Bool | 94.4 | false | True | True | True | False |  |  |
| Control_RockNegative | Bool | 94.5 | false | True | True | True | False |  |  |
| Control_RockPositive | Bool | 94.6 | false | True | True | True | False |  |  |
| Control_Stop | Bool | 94.7 | false | True | True | True | False |  |  |
| InOut |  |  |  |  |  |  |  |  |  |
| ▼ Static |  |  |  |  |  |  |  |  |  |
| Always_ON | Bool | 96.0 | false | False | False | False | False |  |  |
| ▼ Temp |  |  |  |  |  |  |  |  |  |
| SpeedControl | Bool | 0.0 |  |  |  |  |  |  |  |
| AbsolutePositioning | Bool | 0.1 |  |  |  |  |  |  |  |
| ReferenceSearch | Bool | 0.2 |  |  |  |  |  |  |  |
| Set_Reference | Bool | 0.3 |  |  |  |  |  |  |  |
| CmdJogPos | Bool | 0.4 |  |  |  |  |  |  |  |
| CmdJogNeg | Bool | 0.5 |  |  |  |  |  |  |  |
| CmdJogPos_Speed | Bool | 0.6 |  |  |  |  |  |  |  |
| CmdJogNeg_Speed | Bool | 0.7 |  |  |  |  |  |  |  |
| Override_Real | Real | 2.0 |  |  |  |  |  |  |  |
| MaxVelocity_Real | Real | 6.0 |  |  |  |  |  |  |  |
| Aux_Override_Real | Real | 10.0 |  |  |  |  |  |  |  |
| Velocity_rpm_Real | Real | 14.0 |  |  |  |  |  |  |  |
| Velocity_LU_Real | Real | 18.0 |  |  |  |  |  |  |  |
| Velocity_Hz_Real | Real | 22.0 |  |  |  |  |  |  |  |
| Velocity_LU | Real | 26.0 |  |  |  |  |  |  |  |
| Velocity_Hz | DInt | 30.0 |  |  |  |  |  |  |  |
| Velocity_rpm | DInt | 34.0 |  |  |  |  |  |  |  |
| AUX_Acc | Real | 38.0 |  |  |  |  |  |  |  |
| AUX_Acc_1 | Real | 42.0 |  |  |  |  |  |  |  |
| AUX_Acc_2 | Real | 46.0 |  |  |  |  |  |  |  |
| AUX_Dec | Real | 50.0 |  |  |  |  |  |  |  |
| AUX_Dec_1 | Real | 54.0 |  |  |  |  |  |  |  |
| AUX_Dec_2 | Real | 58.0 |  |  |  |  |  |  |  |

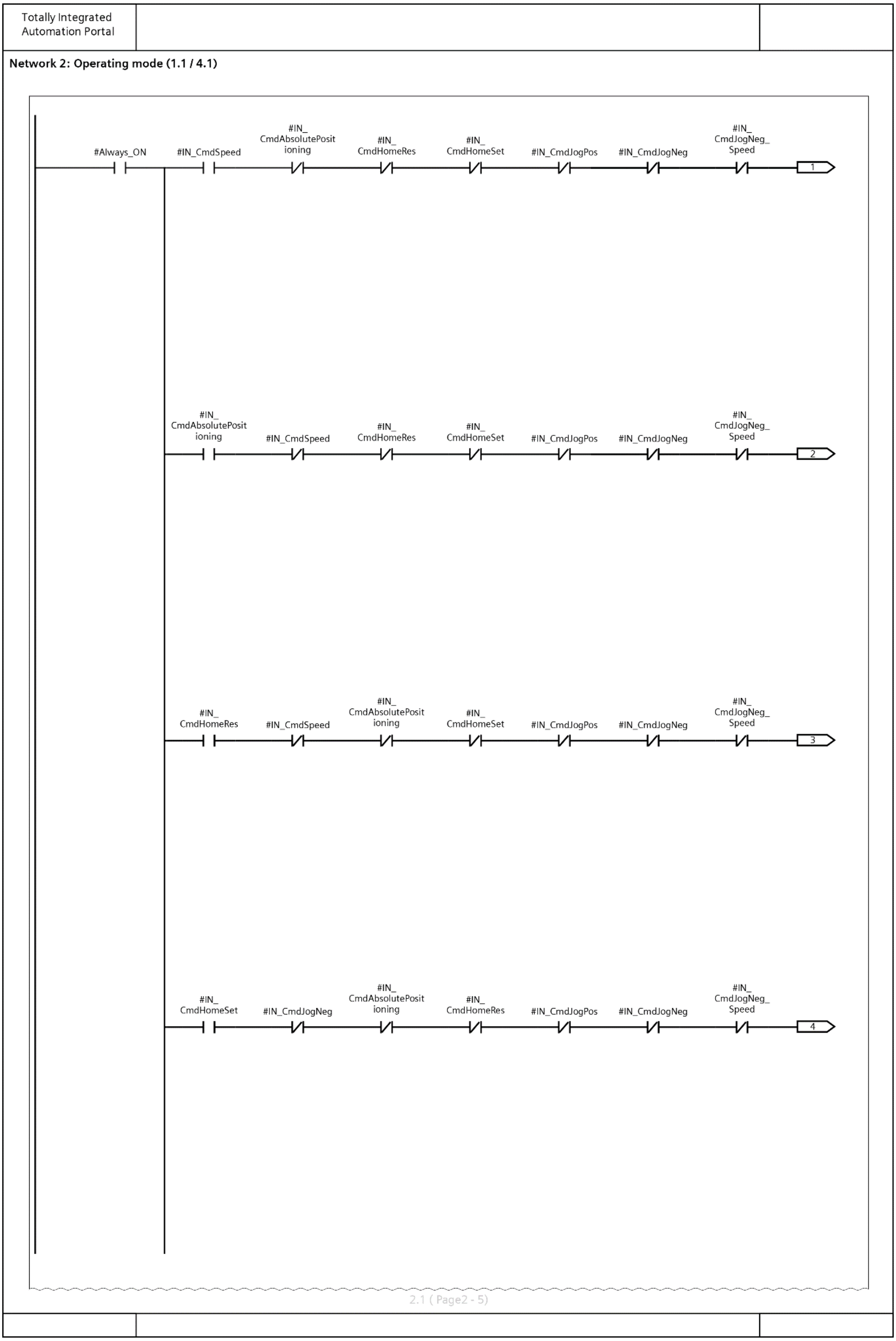


Totally Integrated Automation Portal

| Name | Data type | Offset | Default value | Accessible from HMI/OPC UA | Writ-able from HMI/OPC UA | Visible in HMI engi-neering | Setpoint | Supervi-sion | Comment |
|---|---|---|---|---|---|---|---|---|---|
| AUX_RefSpeed | Real | 62.0 | | | | | | | |
| AUX_Goal_Position | DInt | 66.0 | | | | | | | |
| AUX_SpeedLimitation | DInt | 70.0 | | | | | | | |
| AUX_Override | Int | 74.0 | | | | | | | |
| AUX_OverridePos | Real | 76.0 | | | | | | | |
| AUX_OverridePos_1 | Real | 80.0 | | | | | | | |
| AUX_OverridePos_2 | Real | 84.0 | | | | | | | |
| AUX_ActPosition_Drive | DInt | 88.0 | | | | | | | |
| AUX_ActSpeed | DInt | 92.0 | | | | | | | |
| AUX_ActPosition_Drive_Real | Real | 96.0 | | | | | | | |
| AUX_ActSpeed_Real | Real | 100.0 | | | | | | | |
| JOG1 | Bool | 104.0 | | | | | | | |
| JOG2 | Bool | 104.1 | | | | | | | |
| AUX_Speed | DInt | 106.0 | | | | | | | |
| AUX_Speed_Jog | DInt | 110.0 | | | | | | | |
| AUX_Speed_Hz | DInt | 114.0 | | | | | | | |
| Constant | | | | | | | | | |

**Network 1: --- ALWAYS_ON**

**Network 2: Operating mode**



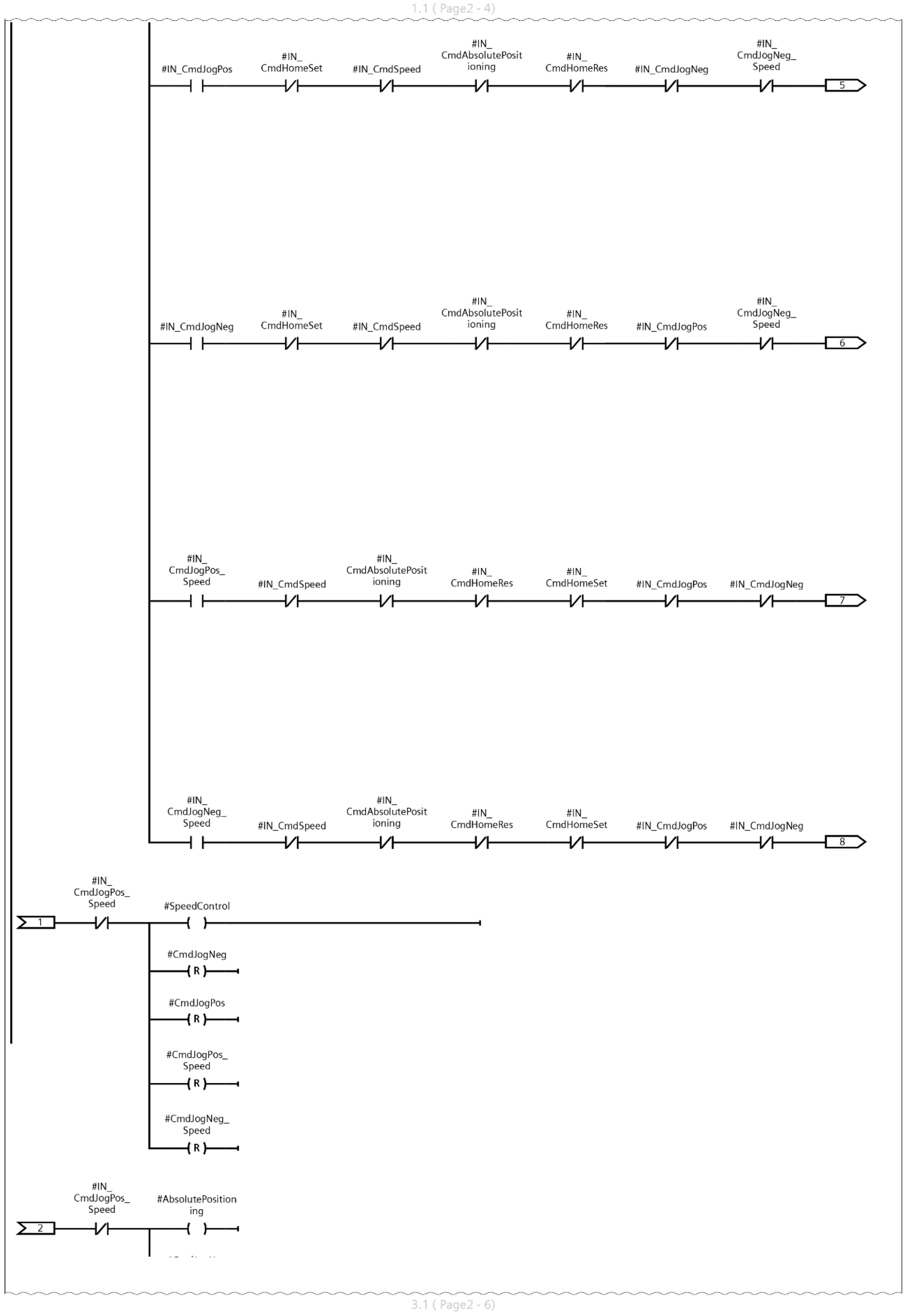
Totally Integrated Automation Portal

Network 2: Operating mode (1.1 / 4.1)

2.1 ( Page2 - 5)

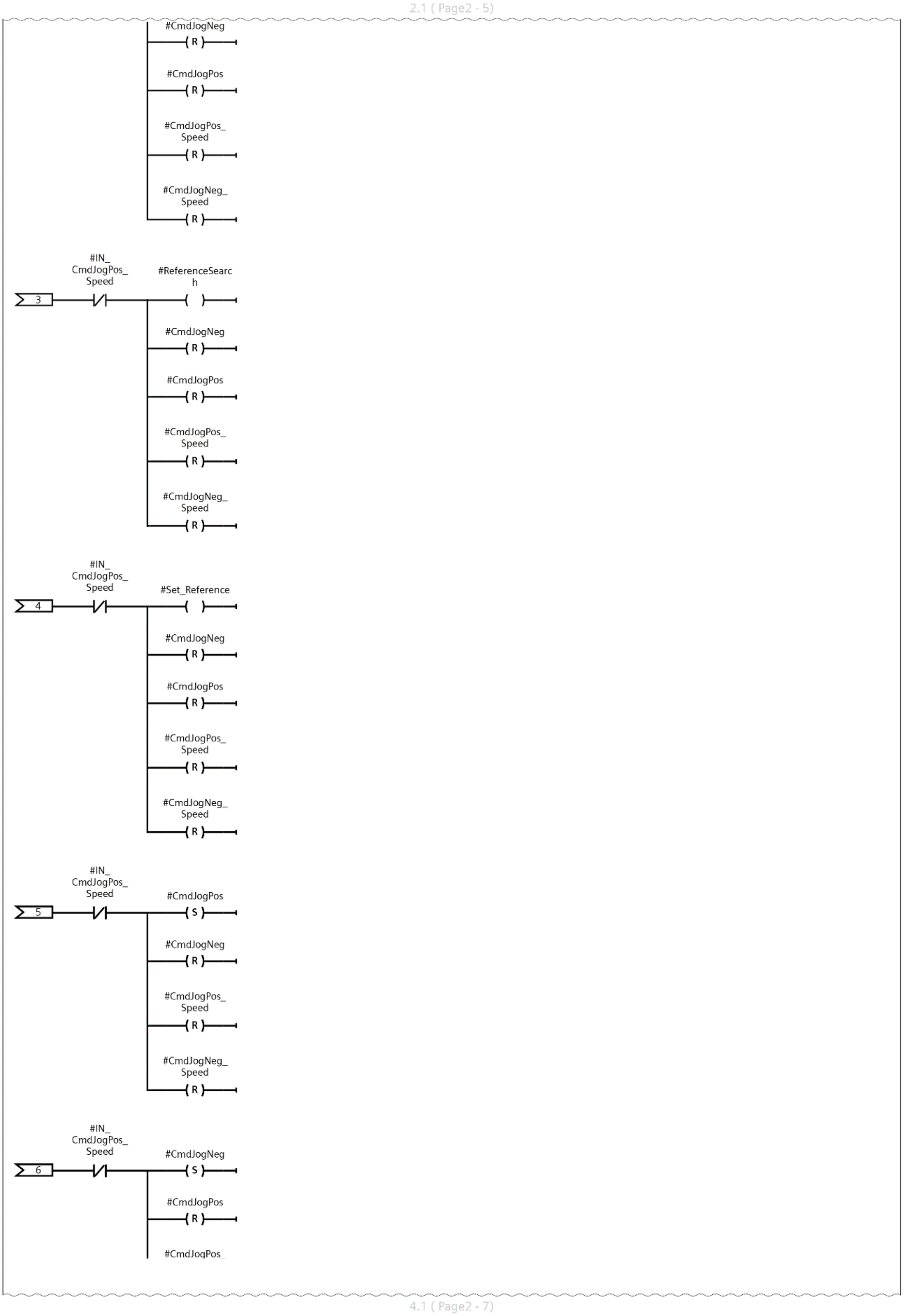
Totally Integrated
Automation Portal

**Network 2: Operating mode (2.1 / 4.1)**

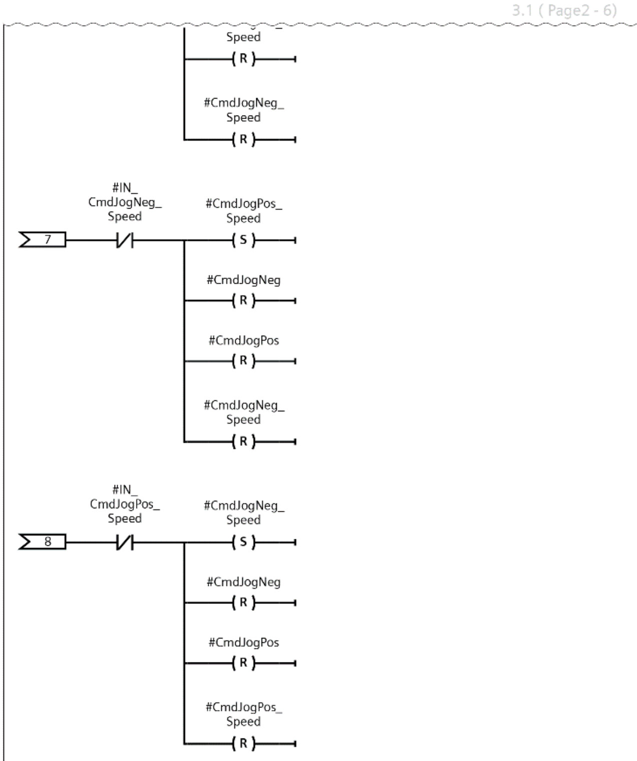


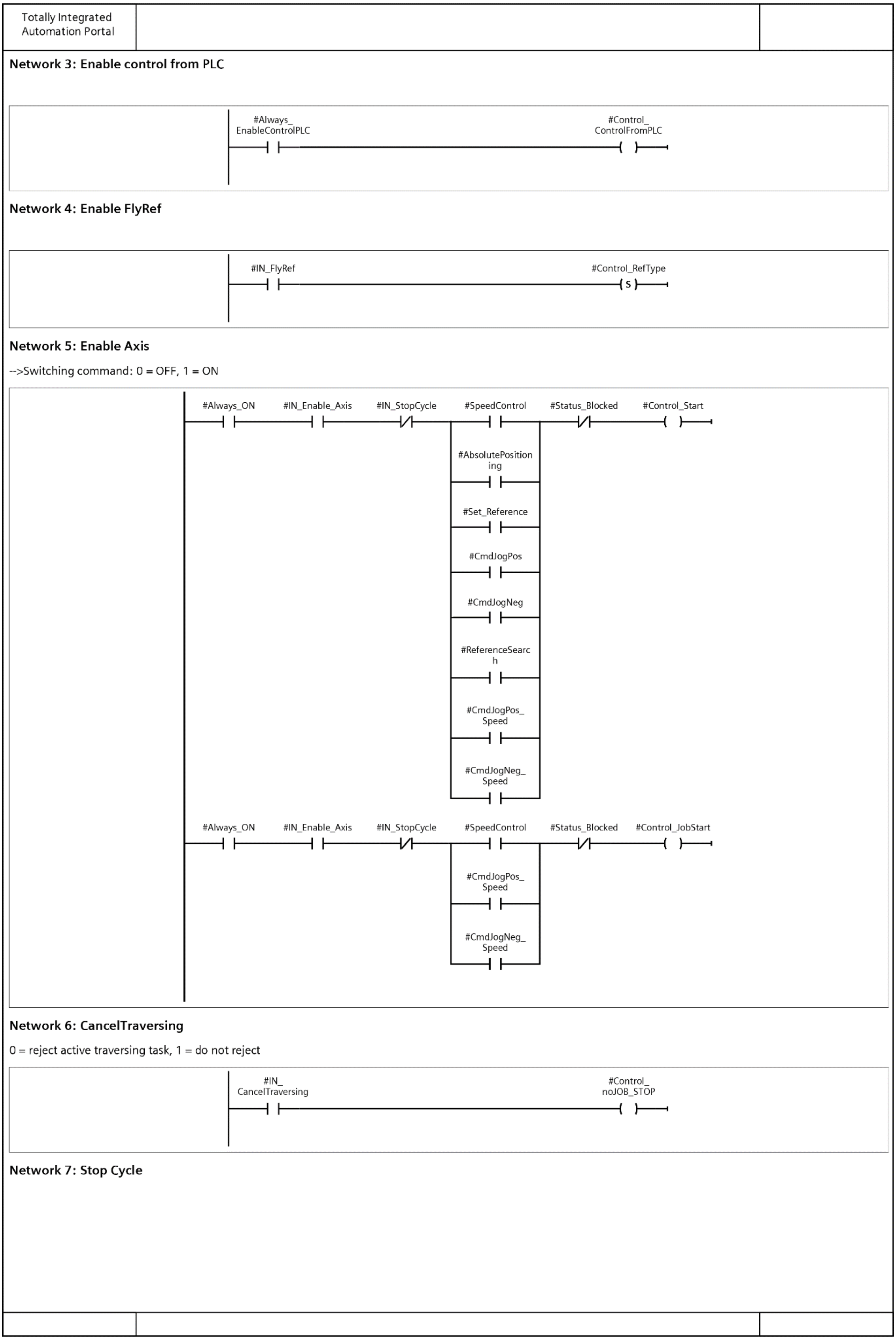
Totally Integrated Automation Portal

**Network 2: Operating mode (3.1 / 4.1)**



**Network 2: Operating mode (4.1 / 4.1)**





Totally Integrated Automation Portal

**Network 3: Enable control from PLC**

**Network 4: Enable FlyRef**

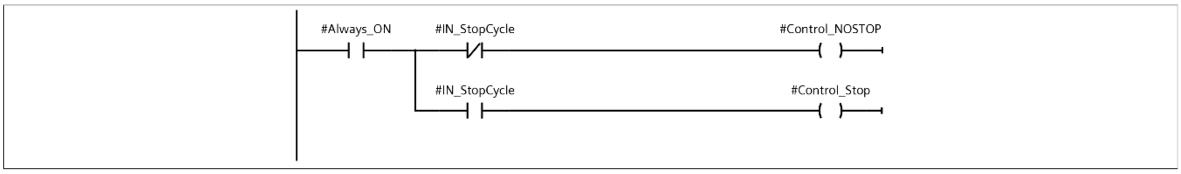
**Network 5: Enable Axis**

-->Switching command: 0 = OFF, 1 = ON

**Network 6: CancelTraversing**

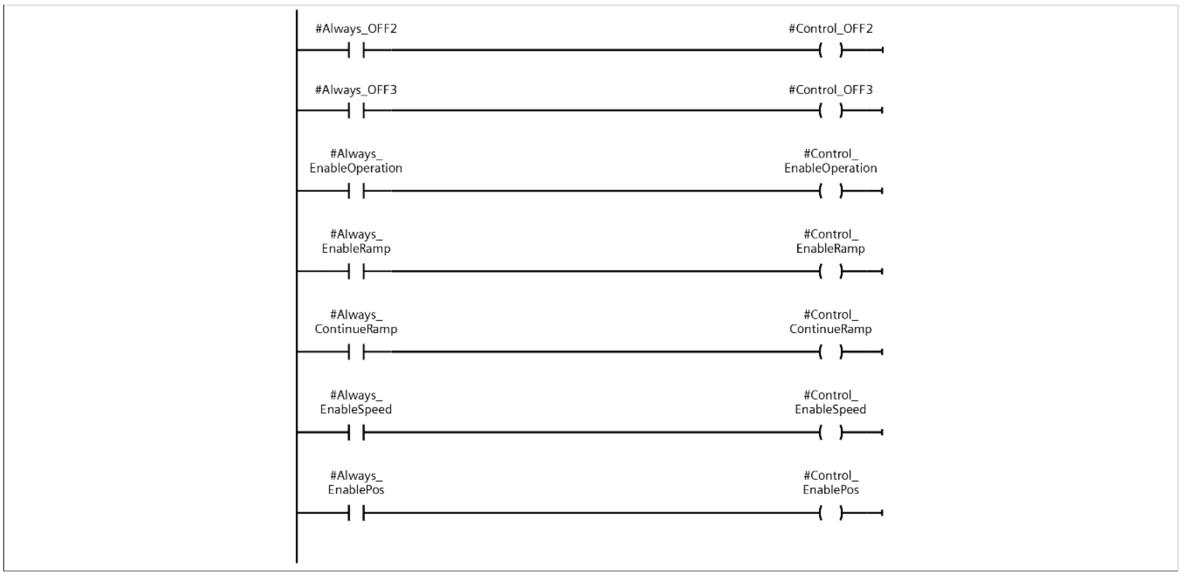
0 = reject active traversing task, 1 = do not reject

**Network 7: Stop Cycle**



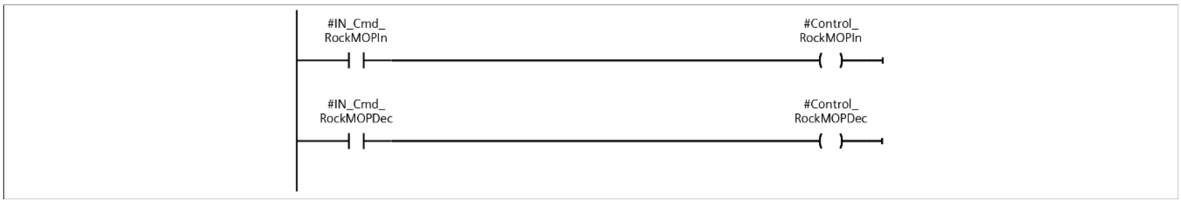
**Network 8:** ---------------------------------------------------------------------------------------------------------------------------------------------------------------

**Network 9: Always_ON Siemens**



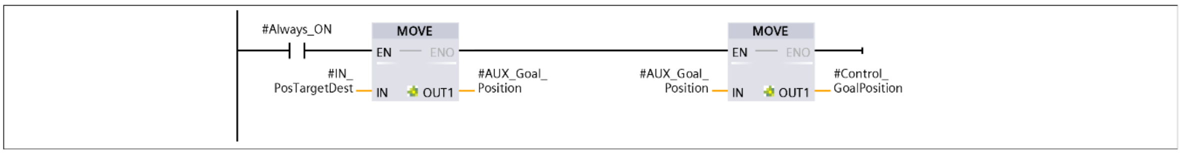
**Network 10:** ---------------------------------------------------------------------------------------------------------------------------------------------------------------

**Network 11: Rockwell command**



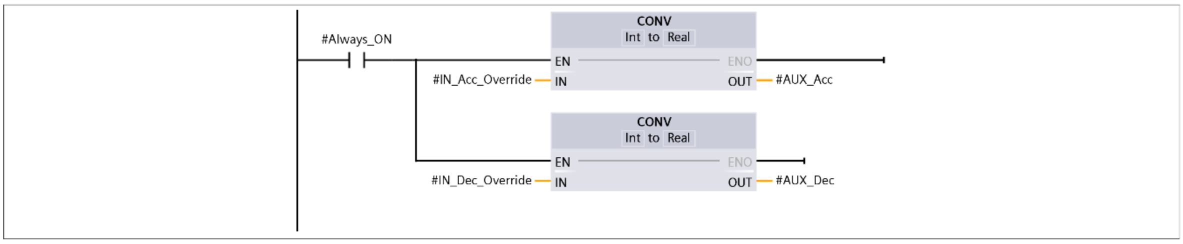
**Network 12:** ---------------------------------------------------------------------------------------------------------------------------------------------------------------

**Network 13: Position**



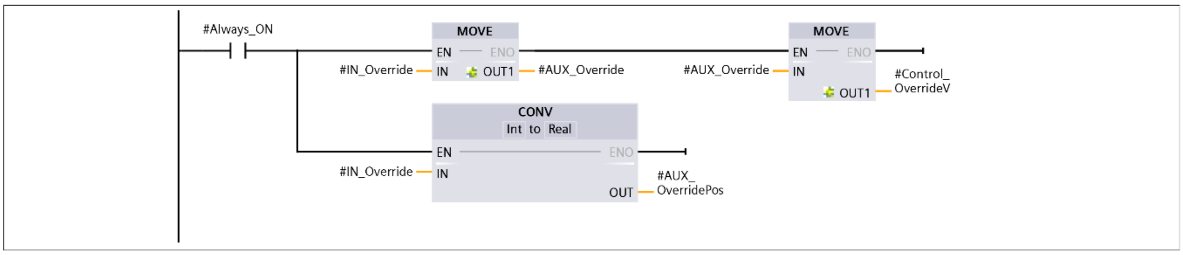
**Network 14:** ---------------------------------------------------------------------------------------------------------------------------------------------------------------

**Network 15: Acceleration & deceleration management**

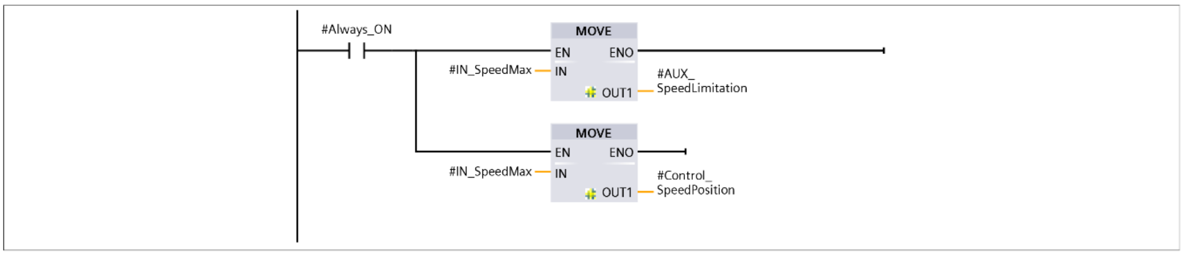


**Network 16:** -----------------------------------------------------------------------------------------------------------------------------------------------

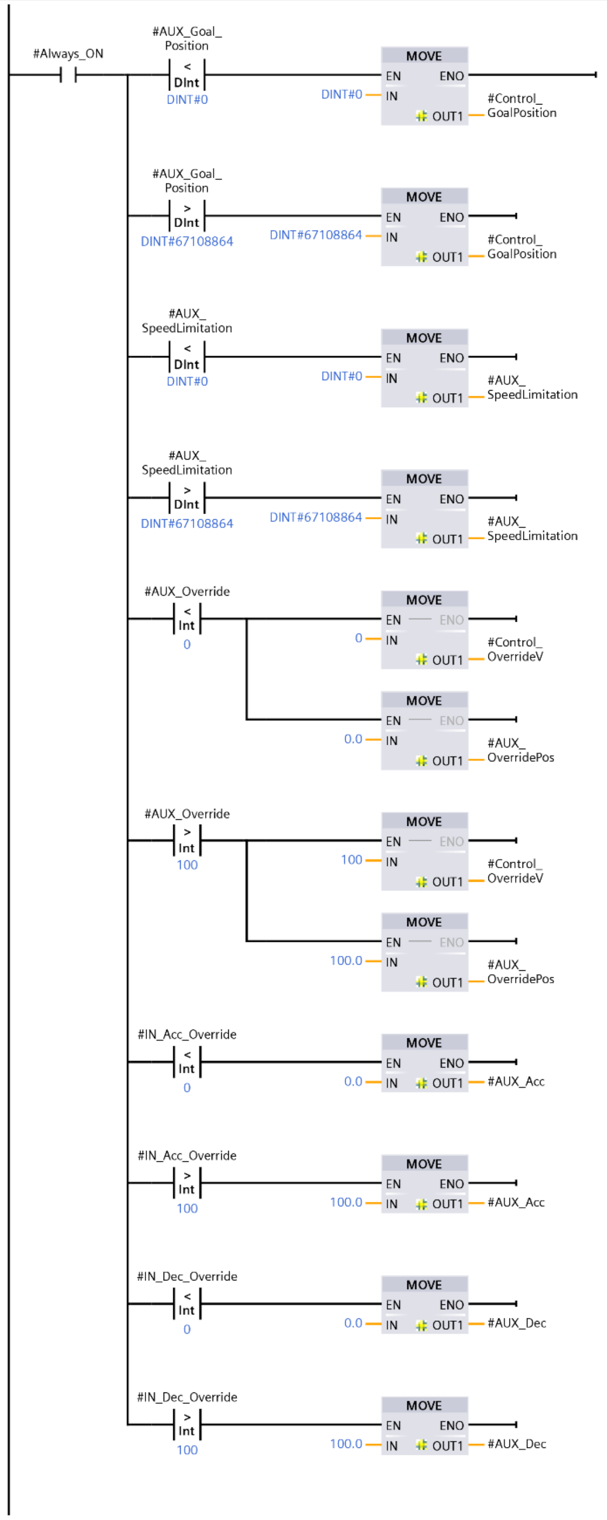
**Network 17: Velocity Override%**



**Network 18: Velocity Max**



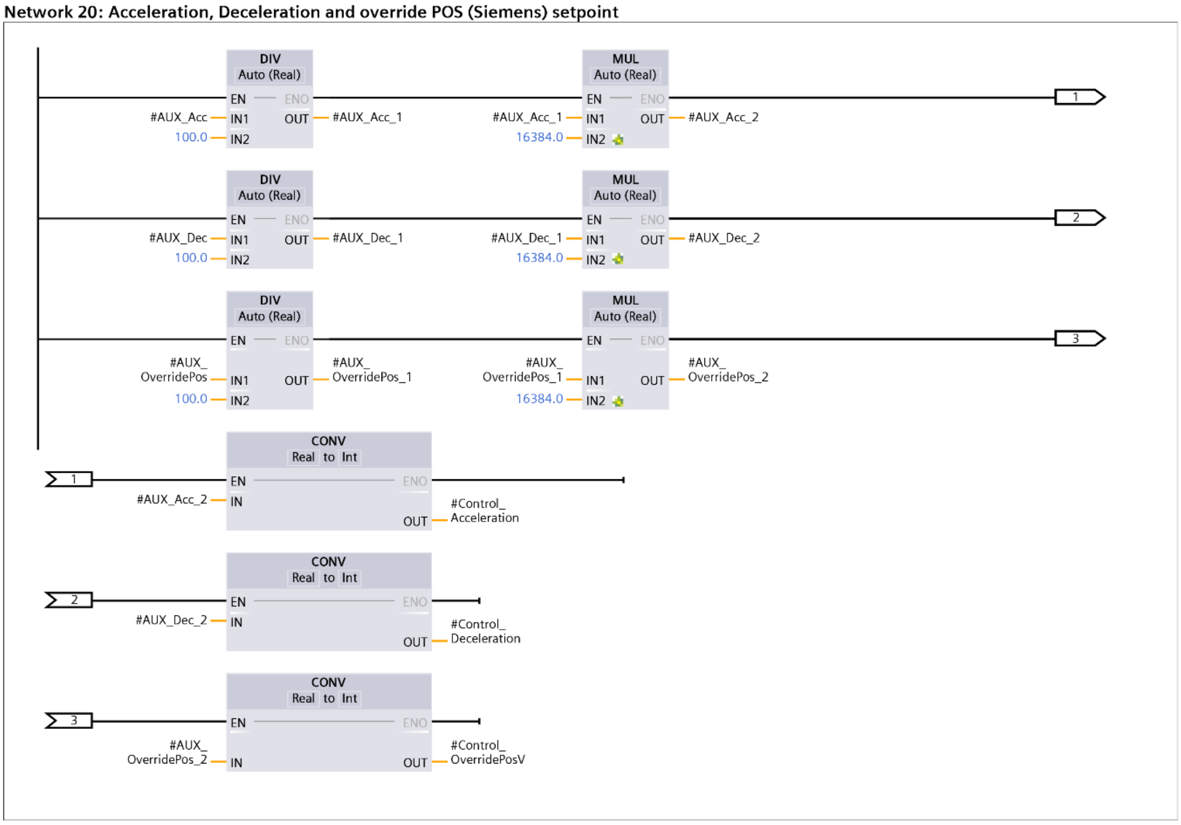
**Network 19: Limitation on "analog" setpoints for DRIVE**



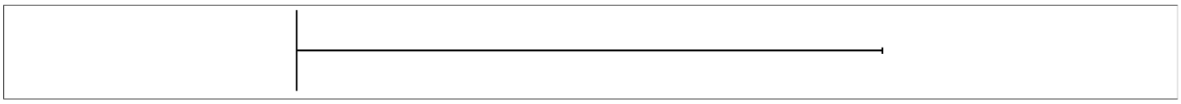
**Network 20: Acceleration, Deceleration and override POS (Siemens) setpoint**

Totally Integrated
Automation Portal

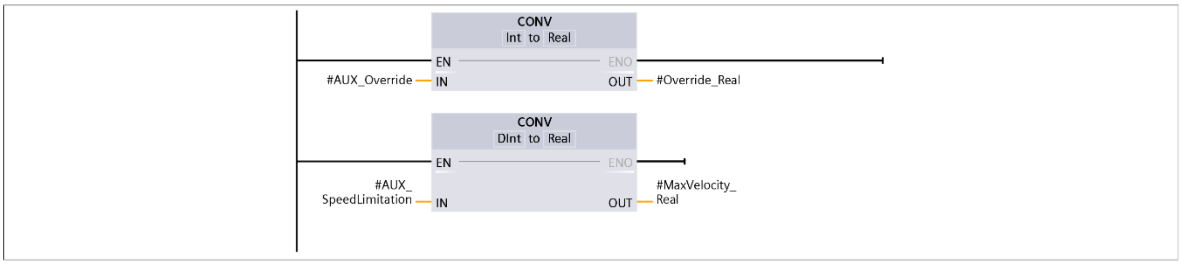
**Network 20: Acceleration, Deceleration and override POS (Siemens) setpoint**



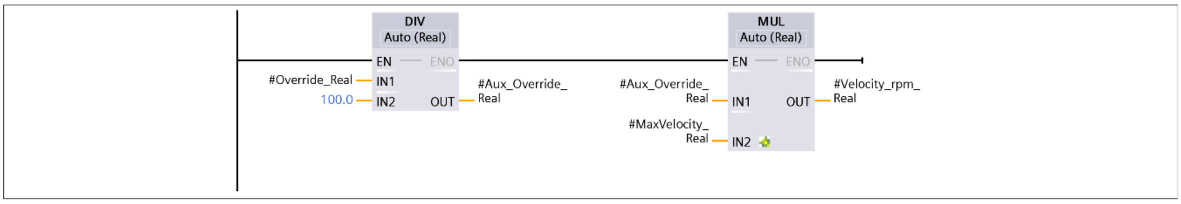
**Network 21: -----------------------SPEED DEFINITION------------------------------------**



**Network 22: Override and MaxVelocity conversion (DInt --> Real)**

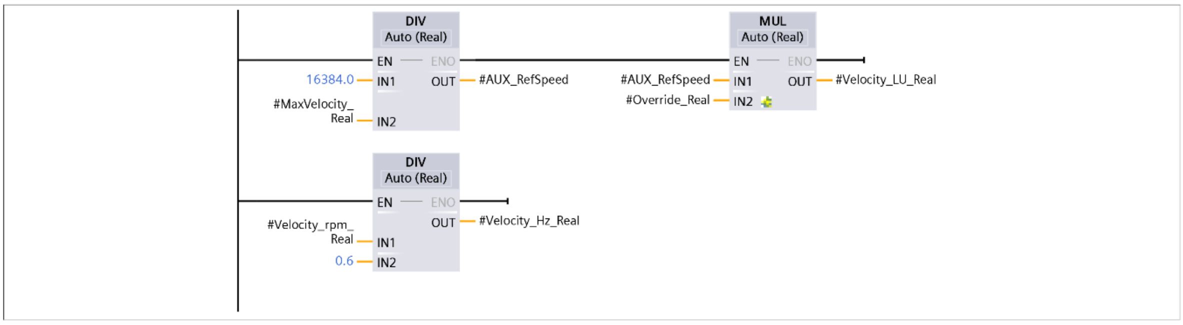


**Network 23: Compute of Velocity in rpm**

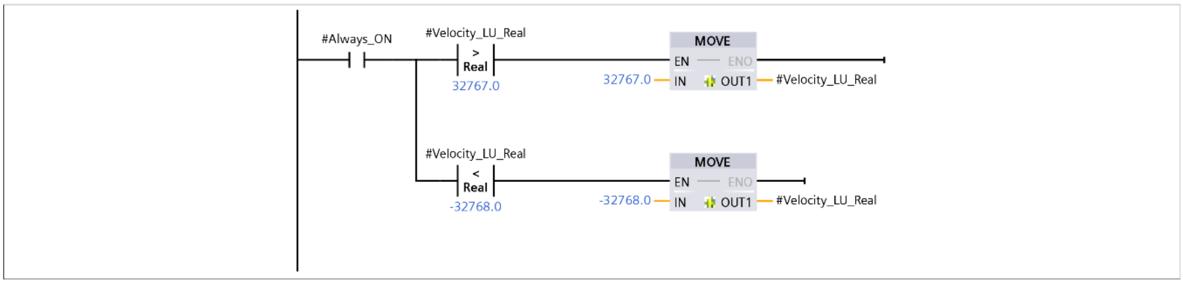


**Network 24: Speed Setpoint in LU/Hz**

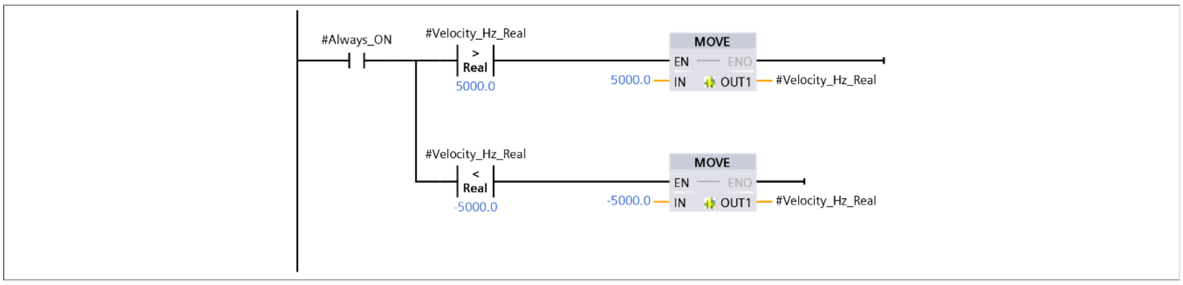
Totally Integrated
Automation Portal



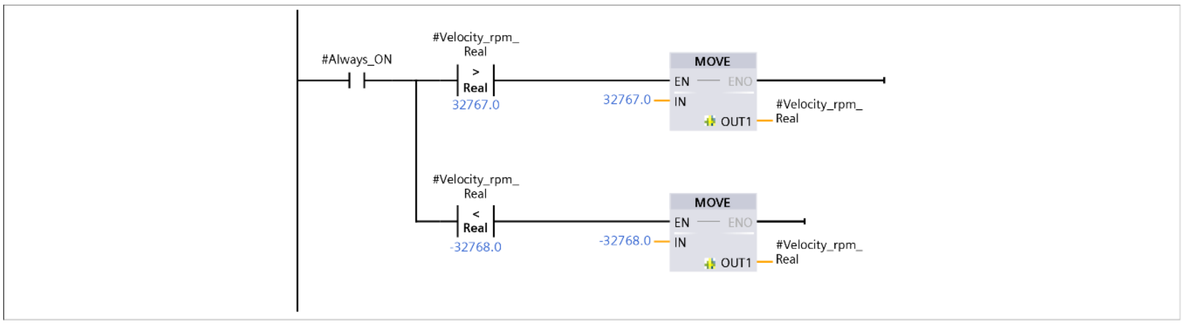
**Network 25: Max speed writable in LU**



**Network 26: Max speed writable in Hz**

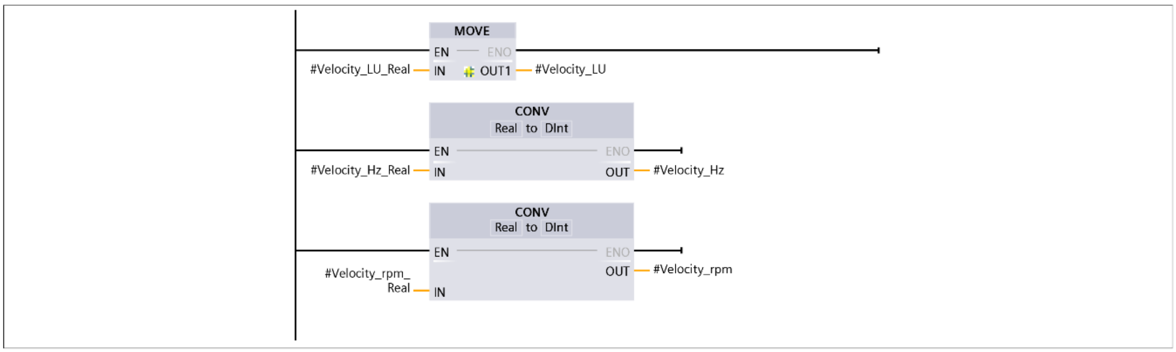


**Network 27: Max speed writable in rpm**

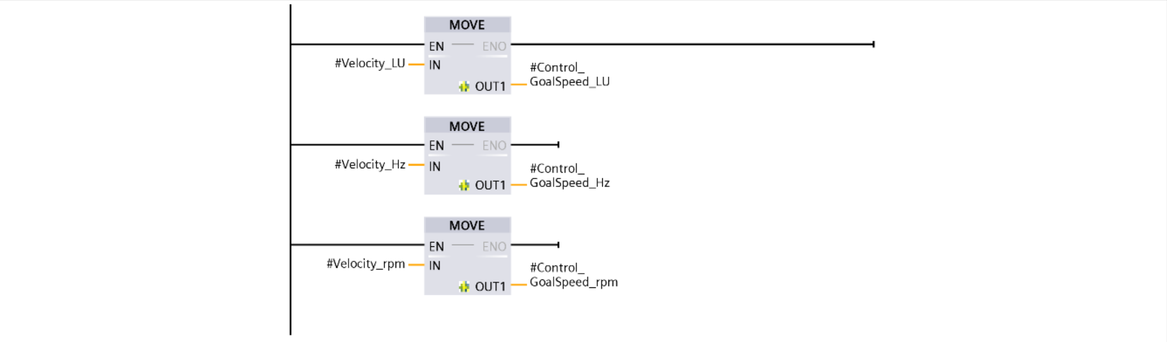


**Network 28: Conversion Real-->DInt**

Totally Integrated
Automation Portal

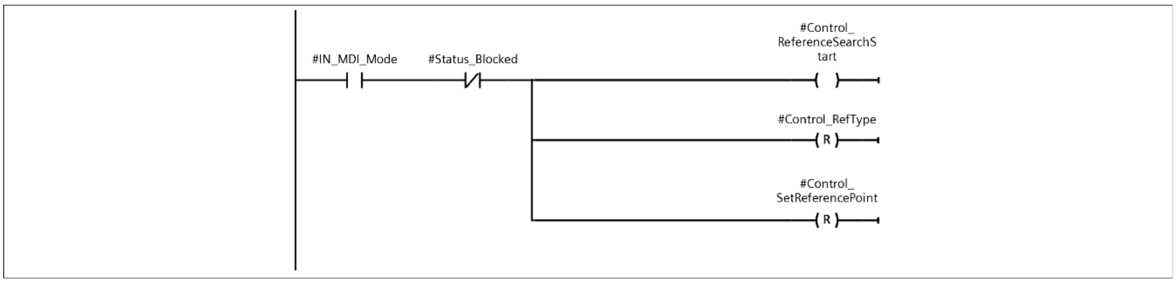


**Network 29: Speed Setpoint**



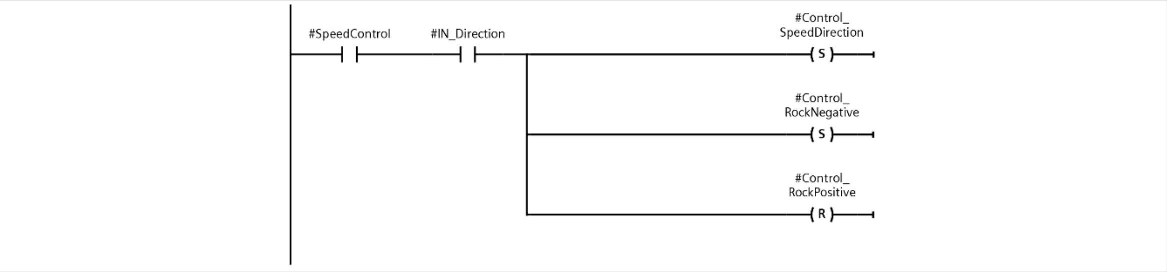
**Network 30: ----------------------------------------------------------------------------------------------------------------------------------------------------------**

**Network 31: Reference search activation**

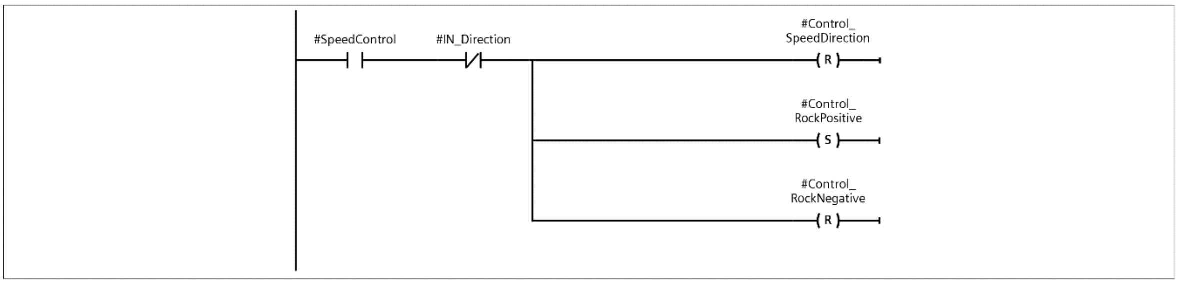


**Network 32: --------------------------------DIRECTION DEFINITION--------------------------------**

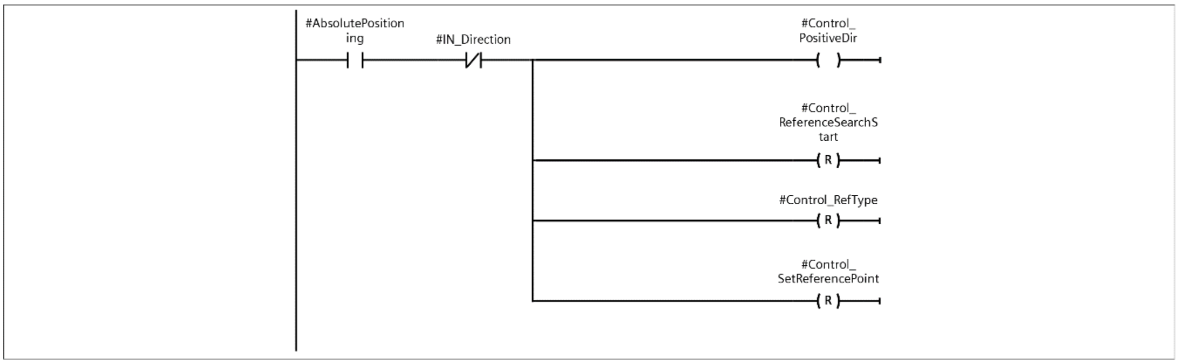
**Network 33: Negative direction of speed control**



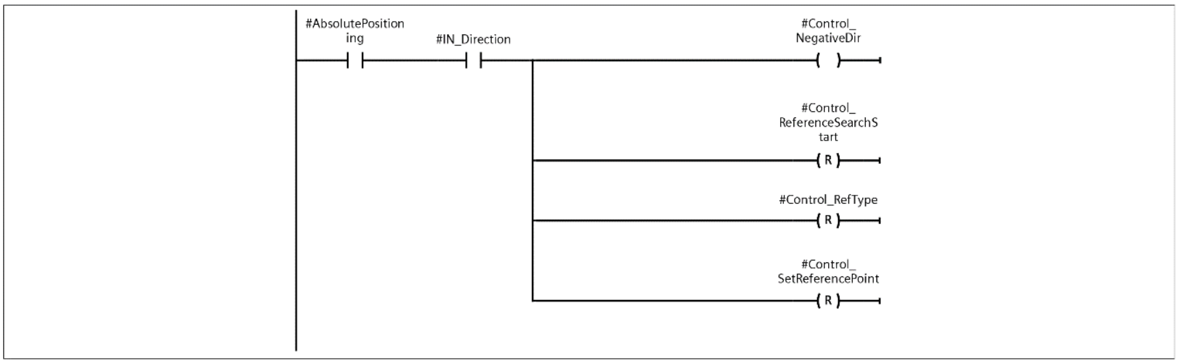
**Network 34: Positive direction of speed control**



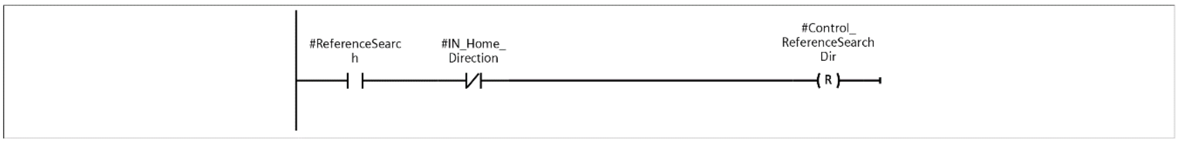
**Network 35: Positive direction of positioning**



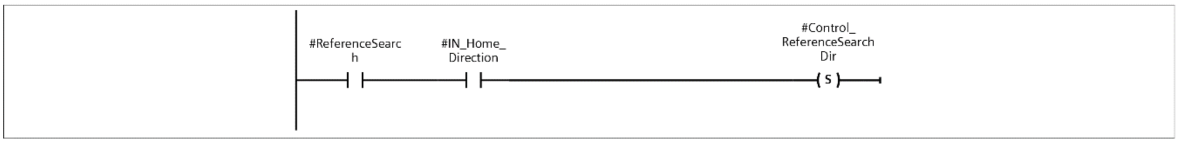
**Network 36: Negative direction of positioning**



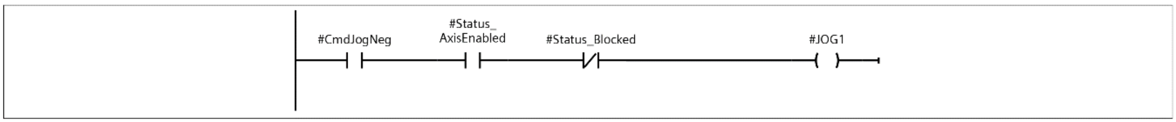
**Network 37: Positive direction of reference point approach**



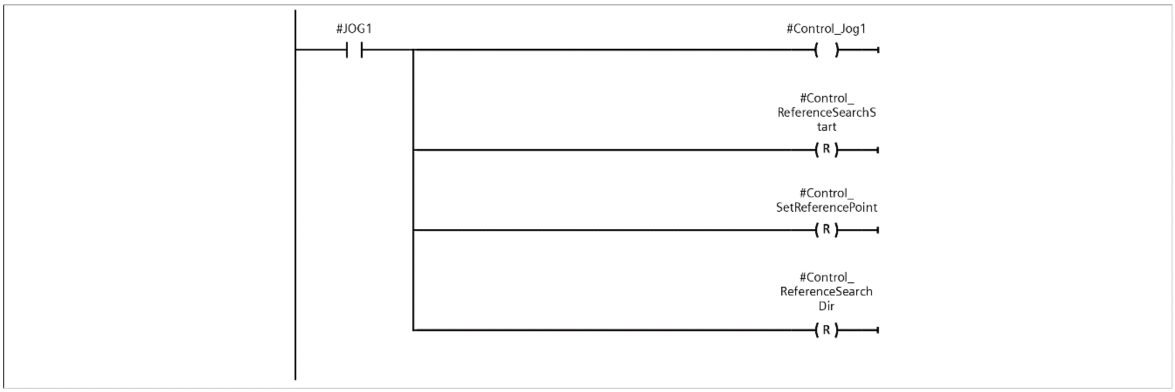
**Network 38: Negative direction of reference point approach**



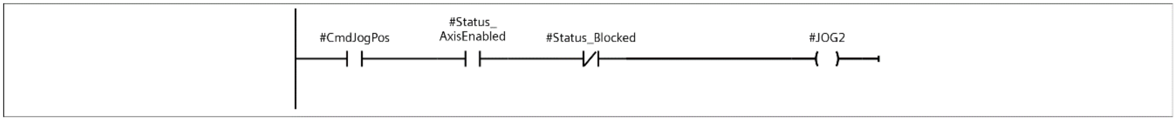
**Network 39: Movement JOG 1**



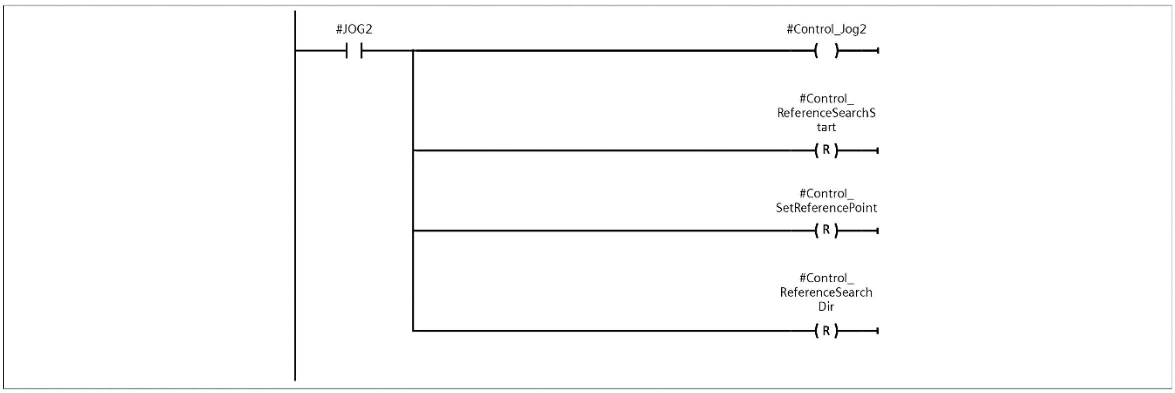
**Network 40: Writing of JOG1**



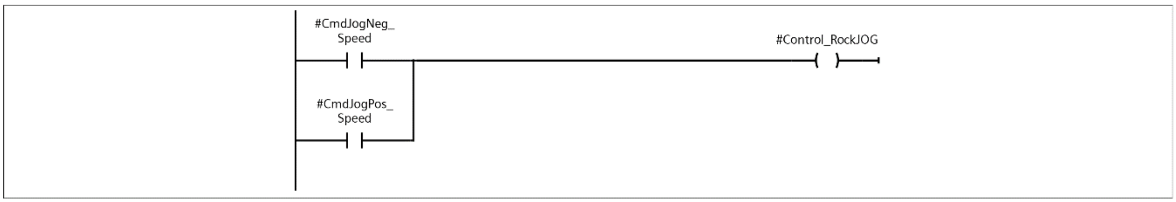
**Network 41: Movement JOG 2**



**Network 42: Writing of JOG2**

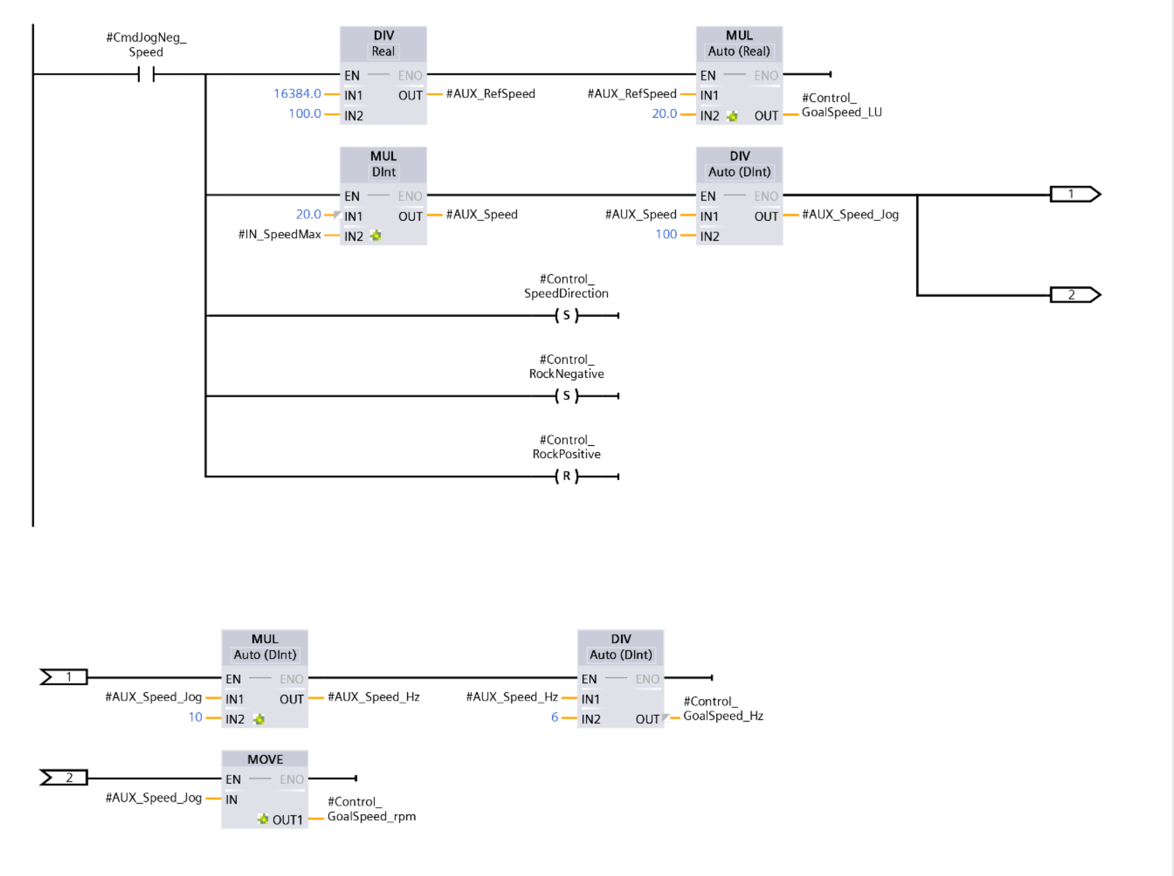


**Network 43: Movement JOG - & JOG + for speed control**



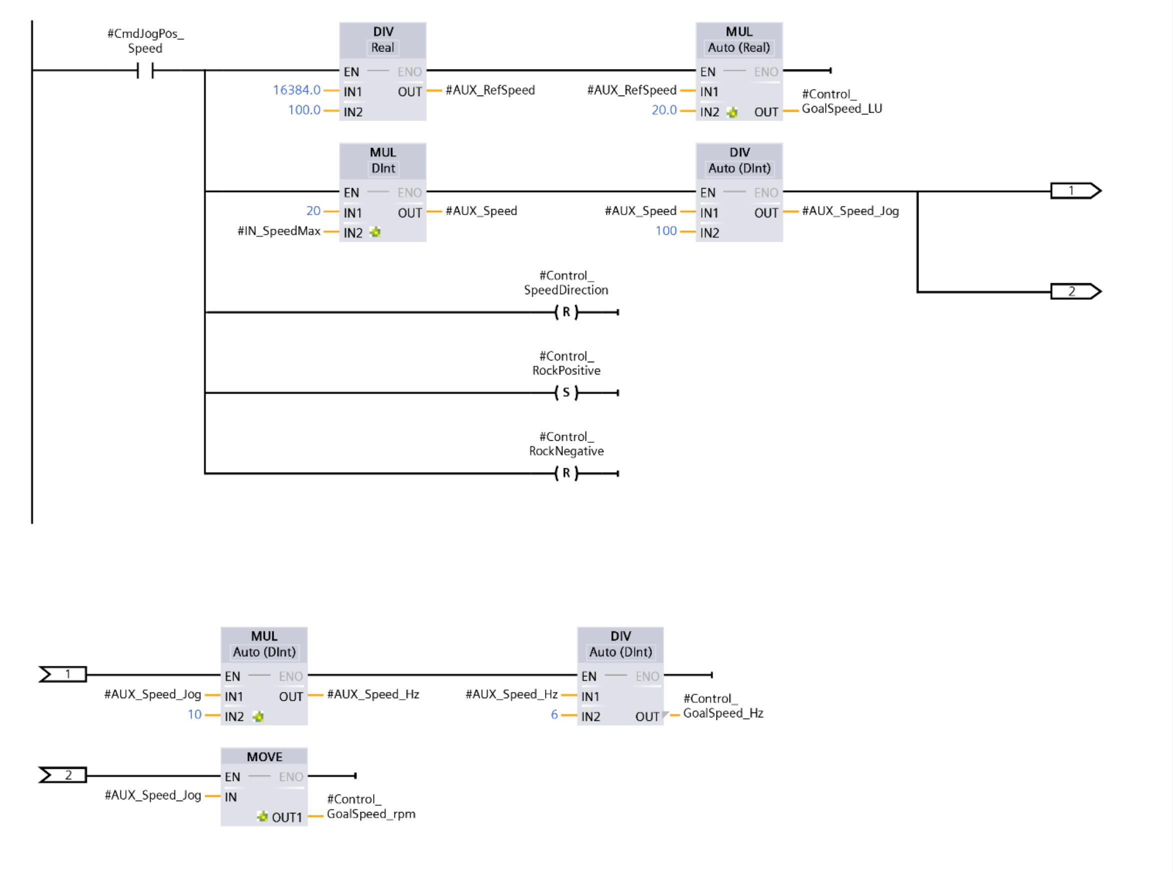
**Network 44: Movement JOG - for speed control**

Network 44: Movement JOG - for speed control



Network 45: Movement JOG + for speed control

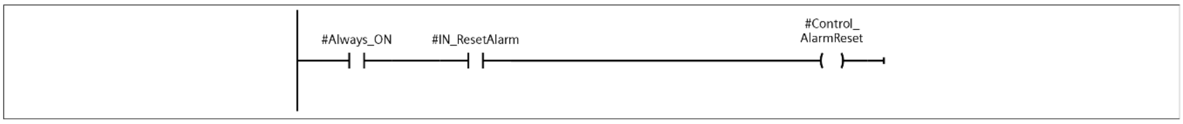
**Network 45: Movement JOG + for speed control**



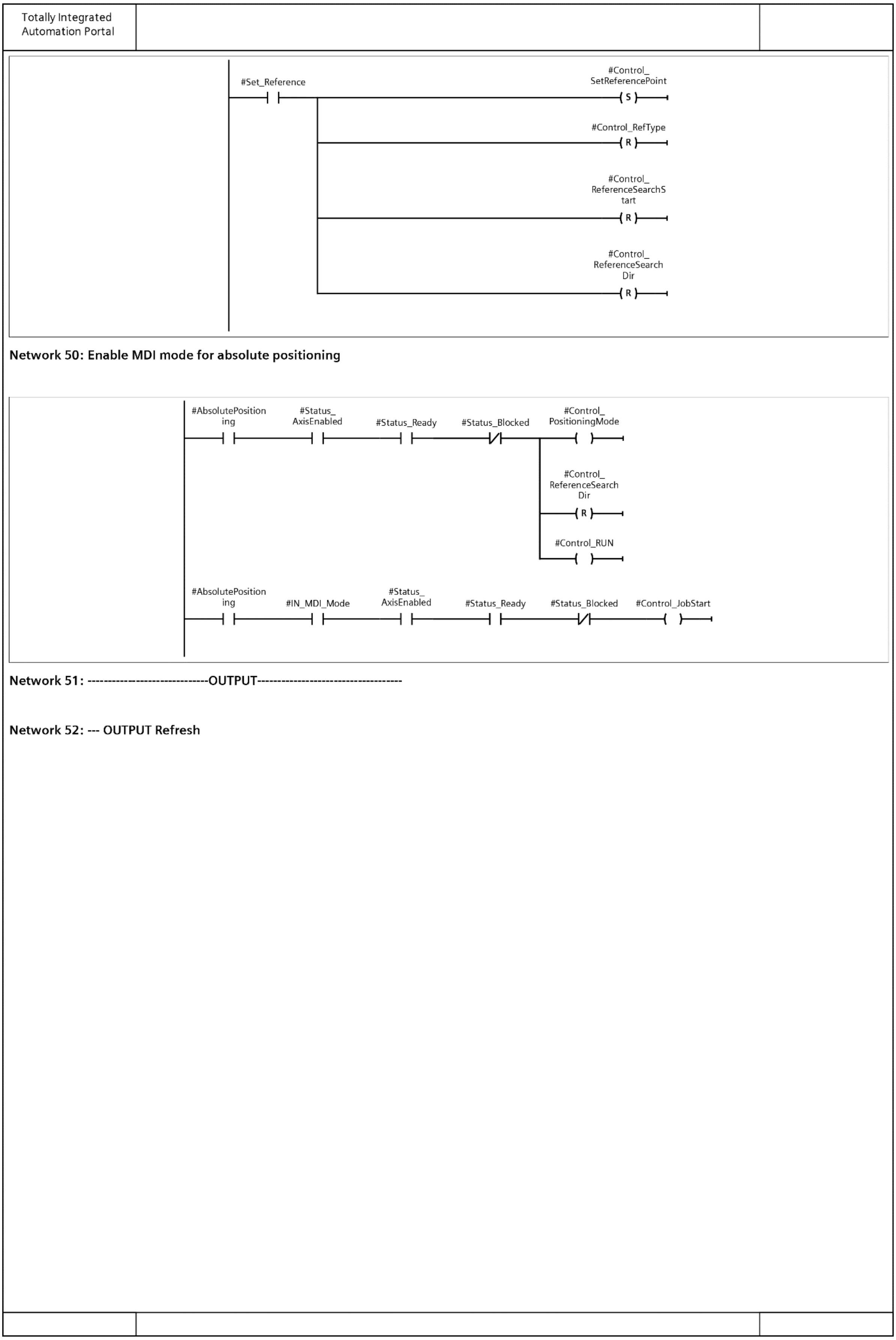
**Network 46:** ----------------------------------------------------------------------------------------------------------------------------------------------------------------------------

**Network 47: --- ALARM RESET AND CONTROL**

**Network 48: Alarm reset**



**Network 49: Set reference point function-->Homing**



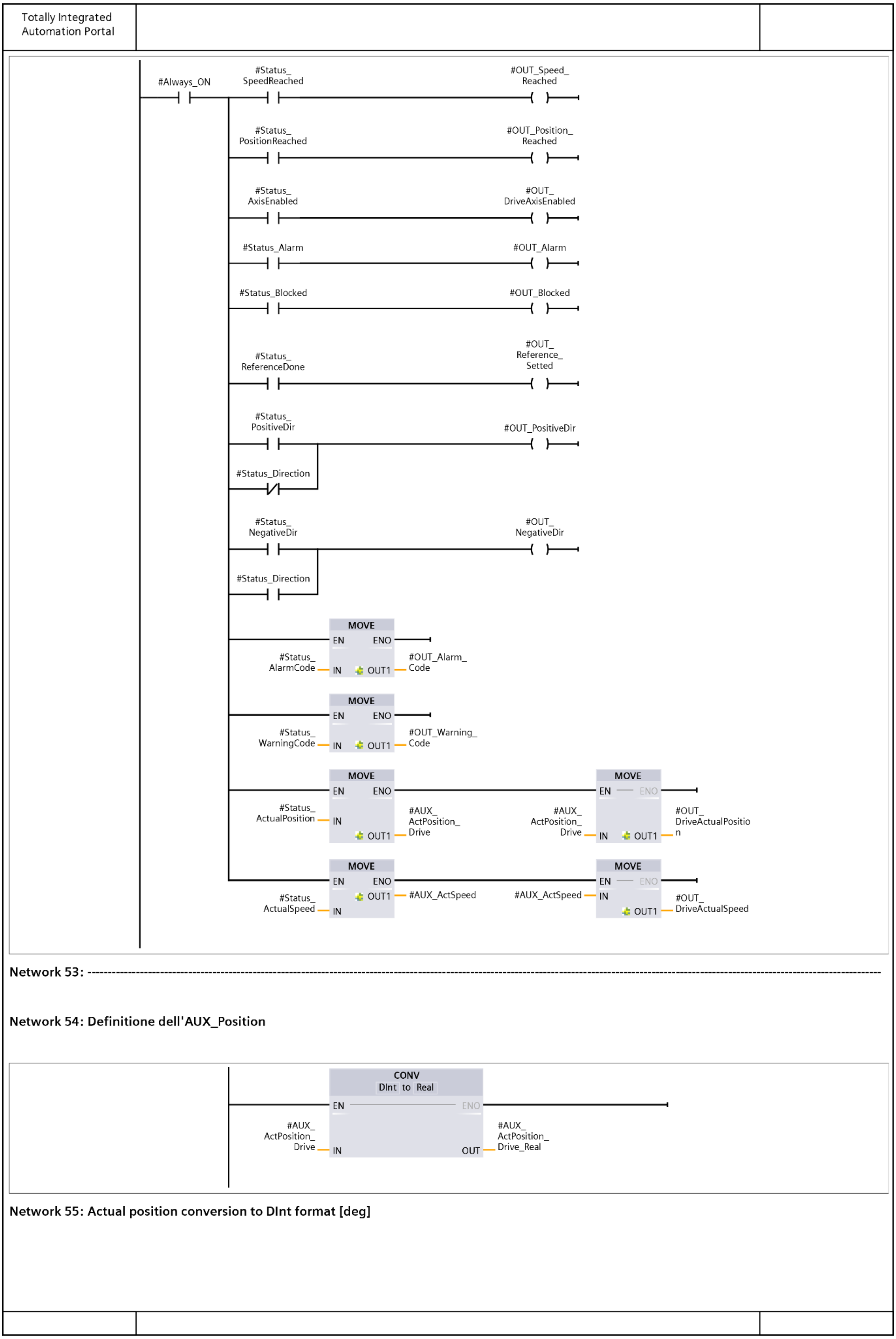
Totally Integrated Automation Portal

#Set_Reference
#Control_ SetReferencePoint
S
#Control_RefType
R
#Control_ ReferenceSearchS tart
R
#Control_ ReferenceSearch Dir
R

**Network 50: Enable MDI mode for absolute positioning**

**Network 51: -----------------------------OUTPUT-----------------------------------**

**Network 52: --- OUTPUT Refresh**



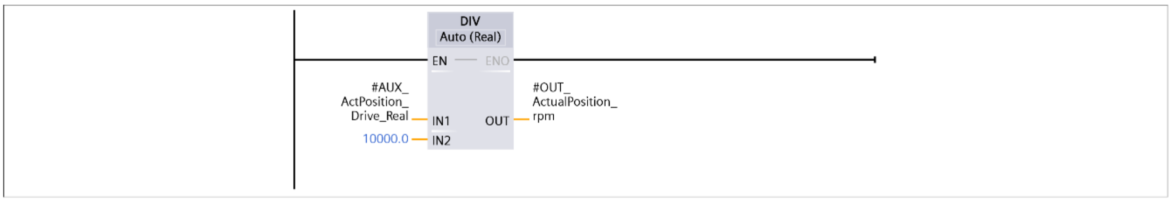
Totally Integrated Automation Portal

**Network 53: ----------------------------------------------------------------------------------------------------------------------------------------**

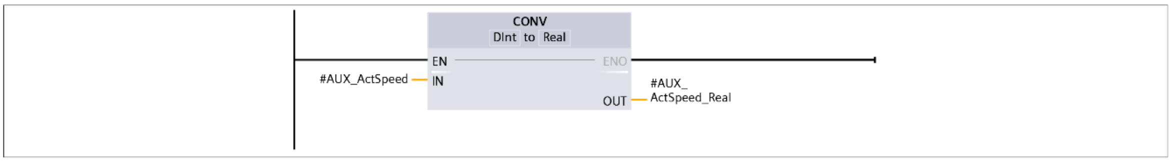
**Network 54: Definitione dell'AUX_Position**

**Network 55: Actual position conversion to DInt format [deg]**

Totally Integrated
Automation Portal

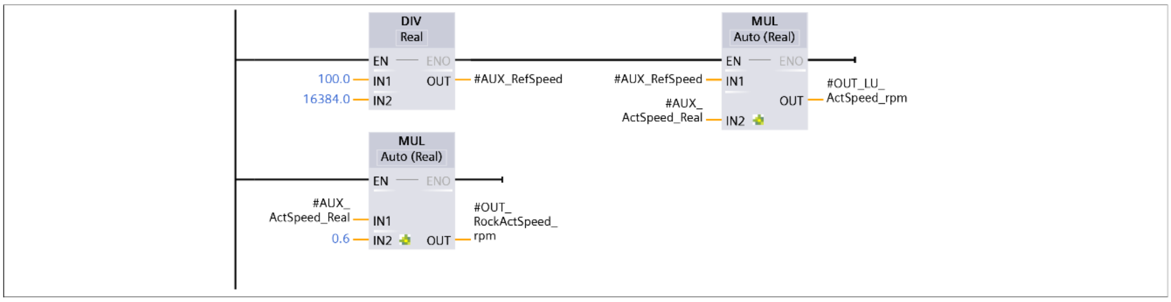


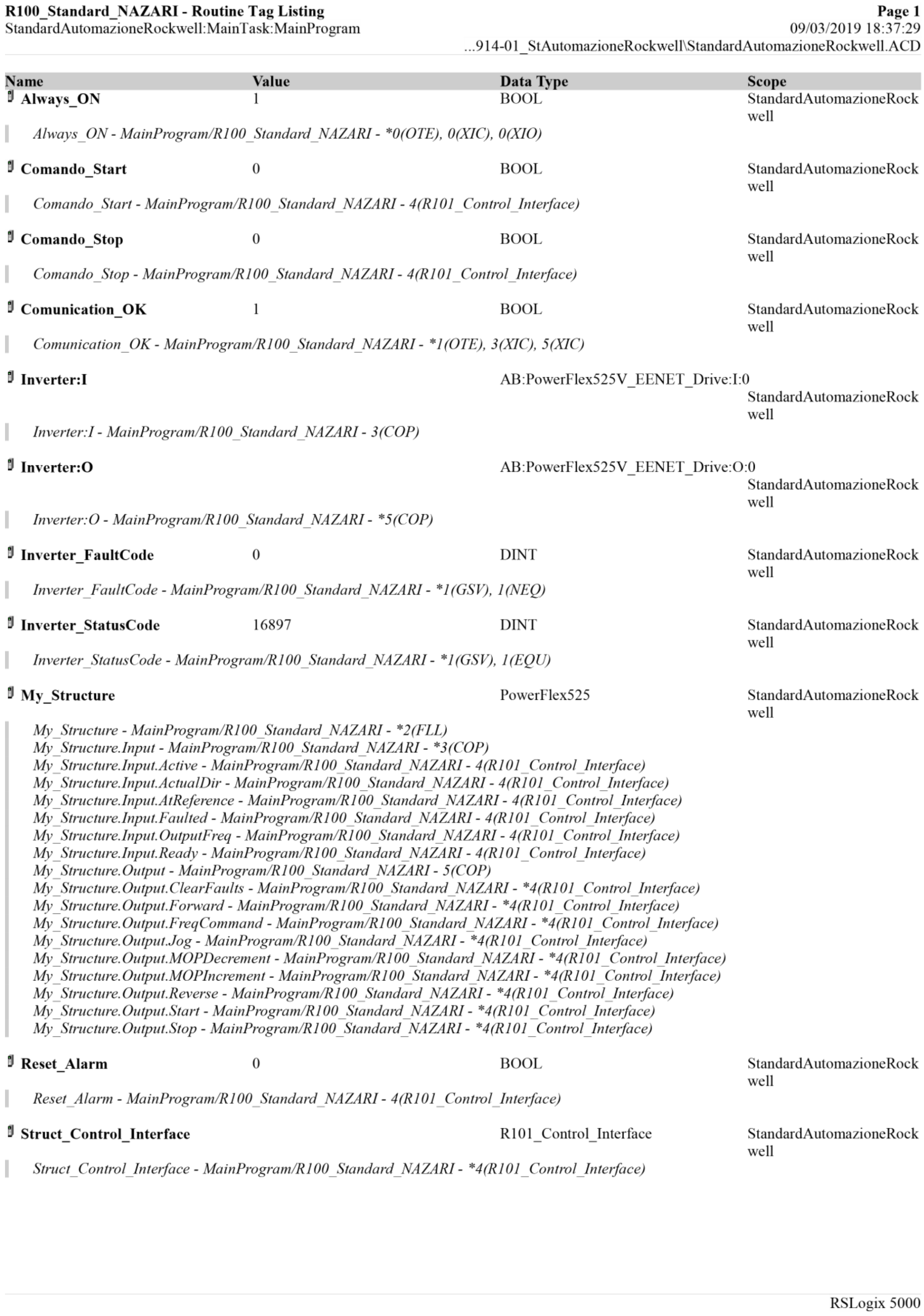
**Network 56: Definitione dell'AUX_Speed**



**Network 57: **********Actual speed conversion to REAL format [rpm]**********

**Network 58: Actual speed conversion to REAL format [rpm]**



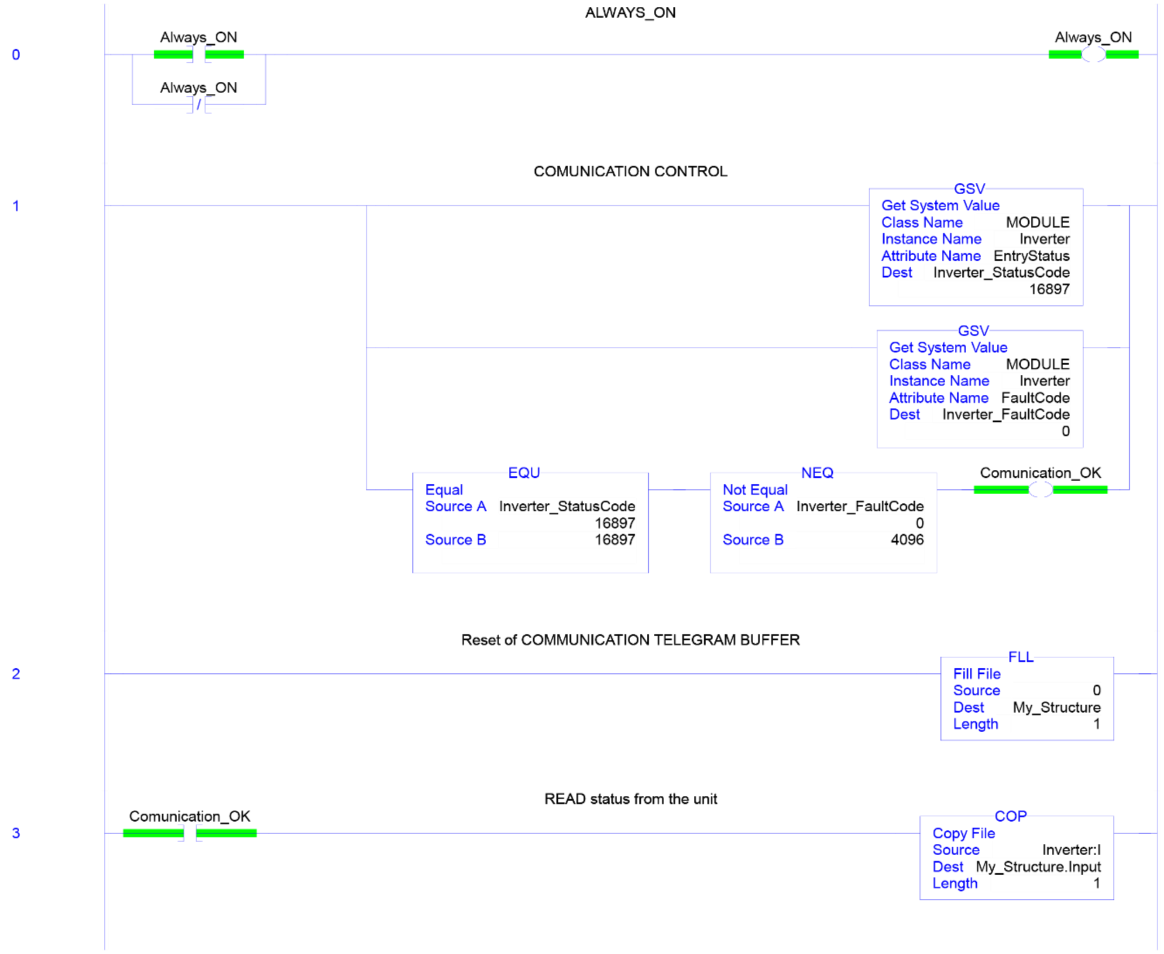


**R100_Standard_NAZARI - Routine Tag Listing** **Page 1**
StandardAutomazioneRockwell:MainTask:MainProgram 09/03/2019 18:37:29
...914-01_StAutomazioneRockwell\StandardAutomazioneRockwell.ACD

| Name | Value | Data Type | Scope |
|---|---|---|---|
| **Always_ON** | 1 | BOOL | StandardAutomazioneRockwell |
| *Always_ON - MainProgram/R100_Standard_NAZARI - *0(OTE), 0(XIC), 0(XIO)* | | | |
| **Comando_Start** | 0 | BOOL | StandardAutomazioneRockwell |
| *Comando_Start - MainProgram/R100_Standard_NAZARI - 4(R101_Control_Interface)* | | | |
| **Comando_Stop** | 0 | BOOL | StandardAutomazioneRockwell |
| *Comando_Stop - MainProgram/R100_Standard_NAZARI - 4(R101_Control_Interface)* | | | |
| **Comunication_OK** | 1 | BOOL | StandardAutomazioneRockwell |
| *Comunication_OK - MainProgram/R100_Standard_NAZARI - *1(OTE), 3(XIC), 5(XIC)* | | | |
| **Inverter:I** | | AB:PowerFlex525V_EENET_Drive:I:0 | StandardAutomazioneRockwell |
| *Inverter:I - MainProgram/R100_Standard_NAZARI - 3(COP)* | | | |
| **Inverter:O** | | AB:PowerFlex525V_EENET_Drive:O:0 | StandardAutomazioneRockwell |
| *Inverter:O - MainProgram/R100_Standard_NAZARI - *5(COP)* | | | |
| **Inverter_FaultCode** | 0 | DINT | StandardAutomazioneRockwell |
| *Inverter_FaultCode - MainProgram/R100_Standard_NAZARI - *1(GSV), 1(NEQ)* | | | |
| **Inverter_StatusCode** | 16897 | DINT | StandardAutomazioneRockwell |
| *Inverter_StatusCode - MainProgram/R100_Standard_NAZARI - *1(GSV), 1(EQU)* | | | |
| **My_Structure** | | PowerFlex525 | StandardAutomazioneRockwell |
| *My_Structure - MainProgram/R100_Standard_NAZARI - *2(FLL)*<br>*My_Structure.Input - MainProgram/R100_Standard_NAZARI - *3(COP)*<br>*My_Structure.Input.Active - MainProgram/R100_Standard_NAZARI - 4(R101_Control_Interface)*<br>*My_Structure.Input.ActualDir - MainProgram/R100_Standard_NAZARI - 4(R101_Control_Interface)*<br>*My_Structure.Input.AtReference - MainProgram/R100_Standard_NAZARI - 4(R101_Control_Interface)*<br>*My_Structure.Input.Faulted - MainProgram/R100_Standard_NAZARI - 4(R101_Control_Interface)*<br>*My_Structure.Input.OutputFreq - MainProgram/R100_Standard_NAZARI - 4(R101_Control_Interface)*<br>*My_Structure.Input.Ready - MainProgram/R100_Standard_NAZARI - 4(R101_Control_Interface)*<br>*My_Structure.Output - MainProgram/R100_Standard_NAZARI - 5(COP)*<br>*My_Structure.Output.ClearFaults - MainProgram/R100_Standard_NAZARI - *4(R101_Control_Interface)*<br>*My_Structure.Output.Forward - MainProgram/R100_Standard_NAZARI - *4(R101_Control_Interface)*<br>*My_Structure.Output.FreqCommand - MainProgram/R100_Standard_NAZARI - *4(R101_Control_Interface)*<br>*My_Structure.Output.Jog - MainProgram/R100_Standard_NAZARI - *4(R101_Control_Interface)*<br>*My_Structure.Output.MOPDecrement - MainProgram/R100_Standard_NAZARI - *4(R101_Control_Interface)*<br>*My_Structure.Output.MOPIncrement - MainProgram/R100_Standard_NAZARI - *4(R101_Control_Interface)*<br>*My_Structure.Output.Reverse - MainProgram/R100_Standard_NAZARI - *4(R101_Control_Interface)*<br>*My_Structure.Output.Start - MainProgram/R100_Standard_NAZARI - *4(R101_Control_Interface)*<br>*My_Structure.Output.Stop - MainProgram/R100_Standard_NAZARI - *4(R101_Control_Interface)* | | | |
| **Reset_Alarm** | 0 | BOOL | StandardAutomazioneRockwell |
| *Reset_Alarm - MainProgram/R100_Standard_NAZARI - 4(R101_Control_Interface)* | | | |
| **Struct_Control_Interface** | | R101_Control_Interface | StandardAutomazioneRockwell |
| *Struct_Control_Interface - MainProgram/R100_Standard_NAZARI - *4(R101_Control_Interface)* | | | |

RSLogix 5000

**R100_Standard_NAZARI - Ladder Diagram** **Page 1**
StandardAutomazioneRockwell:MainTask:MainProgram 09/03/2019 18:36:37
Total number of rungs in routine: 6 ...914-01_StAutomazioneRockwell\StandardAutomazioneRockwell.ACD



0 ALWAYS_ON
Always_ON
Always_ON
Always_ON

1 COMUNICATION CONTROL

GSV
Get System Value
Class Name MODULE
Instance Name Inverter
Attribute Name EntryStatus
Dest Inverter_StatusCode
16897

GSV
Get System Value
Class Name MODULE
Instance Name Inverter
Attribute Name FaultCode
Dest Inverter_FaultCode
0

EQU
Equal
Source A Inverter_StatusCode
16897
Source B 16897

NEQ
Not Equal
Source A Inverter_FaultCode
0
Source B 4096

Comunication_OK

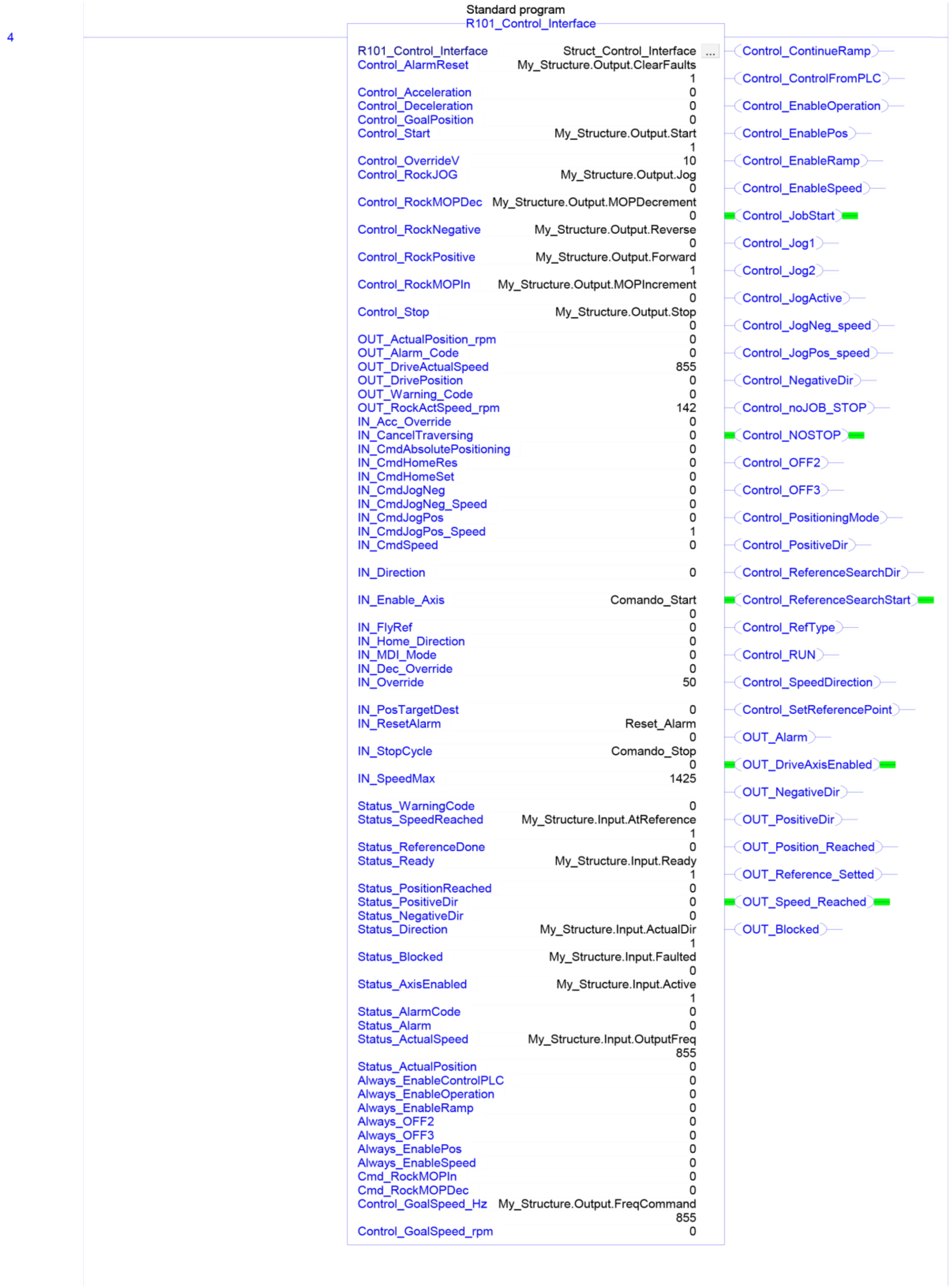
2 Reset of COMMUNICATION TELEGRAM BUFFER

FLL
Fill File
Source 0
Dest My_Structure
Length 1

3 READ status from the unit
Comunication_OK

COP
Copy File
Source Inverter:I
Dest My_Structure.Input
Length 1

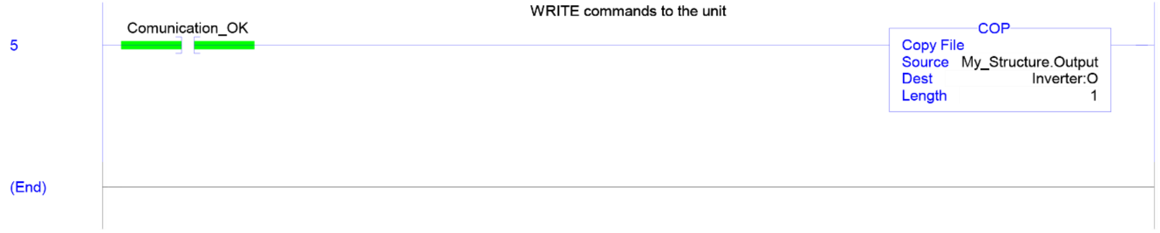
**R100_Standard_NAZARI - Ladder Diagram** **Page 2**
StandardAutomazioneRockwell:MainTask:MainProgram 09/03/2019 18:36:38
Total number of rungs in routine: 6 ...914-01_StAutomazioneRockwell\StandardAutomazioneRockwell.ACD



```
4                                  Standard program
                                   R101_Control_Interface
  R101_Control_Interface          Struct_Control_Interface ...   -( Control_ContinueRamp )-
  Control_AlarmReset          My_Structure.Output.ClearFaults    -( Control_ControlFromPLC )-
                                                            1
  Control_Acceleration                                      0    -( Control_EnableOperation )-
  Control_Deceleration                                      0
  Control_GoalPosition                                      0    -( Control_EnablePos )-
  Control_Start                     My_Structure.Output.Start
                                                            1
  Control_OverrideV                                        10    -( Control_EnableRamp )-
  Control_RockJOG                     My_Structure.Output.Jog
                                                            0    -( Control_EnableSpeed )-
  Control_RockMOPDec  My_Structure.Output.MOPDecrement           -( Control_JobStart )-
                                                            0
  Control_RockNegative            My_Structure.Output.Reverse    -( Control_Jog1 )-
                                                            0
  Control_RockPositive            My_Structure.Output.Forward    -( Control_Jog2 )-
                                                            1
  Control_RockMOPIn     My_Structure.Output.MOPIncrement         -( Control_JogActive )-
                                                            0
  Control_Stop                       My_Structure.Output.Stop    -( Control_JogNeg_speed )-
                                                            0
  OUT_ActualPosition_rpm                                    0    -( Control_JogPos_speed )-
  OUT_Alarm_Code                                            0
  OUT_DriveActualSpeed                                    855    -( Control_NegativeDir )-
  OUT_DrivePosition                                         0
  OUT_Warning_Code                                          0
  OUT_RockActSpeed_rpm                                    142    -( Control_noJOB_STOP )-
  IN_Acc_Override                                           0
  IN_CancelTraversing                                       0    -( Control_NOSTOP )-
  IN_CmdAbsolutePositioning                                 0
  IN_CmdHomeRes                                             0    -( Control_OFF2 )-
  IN_CmdHomeSet                                             0
  IN_CmdJogNeg                                              0    -( Control_OFF3 )-
  IN_CmdJogNeg_Speed                                        0
  IN_CmdJogPos                                              0    -( Control_PositioningMode )-
  IN_CmdJogPos_Speed                                        1
  IN_CmdSpeed                                               0    -( Control_PositiveDir )-

  IN_Direction                                              0    -( Control_ReferenceSearchDir )-

  IN_Enable_Axis                                Comando_Start    -( Control_ReferenceSearchStart )-
                                                            0
  IN_FlyRef                                                 0    -( Control_RefType )-
  IN_Home_Direction                                         0
  IN_MDI_Mode                                               0    -( Control_RUN )-
  IN_Dec_Override                                           0
  IN_Override                                              50    -( Control_SpeedDirection )-

  IN_PosTargetDest                                          0    -( Control_SetReferencePoint )-
  IN_ResetAlarm                                   Reset_Alarm
                                                            0    -( OUT_Alarm )-
  IN_StopCycle                                   Comando_Stop
                                                            0    -( OUT_DriveAxisEnabled )-
  IN_SpeedMax                                            1425
                                                                 -( OUT_NegativeDir )-
  Status_WarningCode                                        0
  Status_SpeedReached          My_Structure.Input.AtReference    -( OUT_PositiveDir )-
                                                            1
  Status_ReferenceDone                                      0    -( OUT_Position_Reached )-
  Status_Ready                         My_Structure.Input.Ready
                                                            1    -( OUT_Reference_Setted )-
  Status_PositionReached                                    0
  Status_PositiveDir                                        0    -( OUT_Speed_Reached )-
  Status_NegativeDir                                        0
  Status_Direction                 My_Structure.Input.ActualDir  -( OUT_Blocked )-
                                                            1
  Status_Blocked                     My_Structure.Input.Faulted
                                                            0
  Status_AxisEnabled                  My_Structure.Input.Active
                                                            1
  Status_AlarmCode                                          0
  Status_Alarm                                              0
  Status_ActualSpeed              My_Structure.Input.OutputFreq
                                                          855
  Status_ActualPosition                                     0
  Always_EnableControlPLC                                   0
  Always_EnableOperation                                    0
  Always_EnableRamp                                         0
  Always_OFF2                                               0
  Always_OFF3                                               0
  Always_EnablePos                                          0
  Always_EnableSpeed                                        0
  Cmd_RockMOPIn                                             0
  Cmd_RockMOPDec                                            0
  Control_GoalSpeed_Hz   My_Structure.Output.FreqCommand
                                                          855
  Control_GoalSpeed_rpm                                     0
```

RSLogix 5000

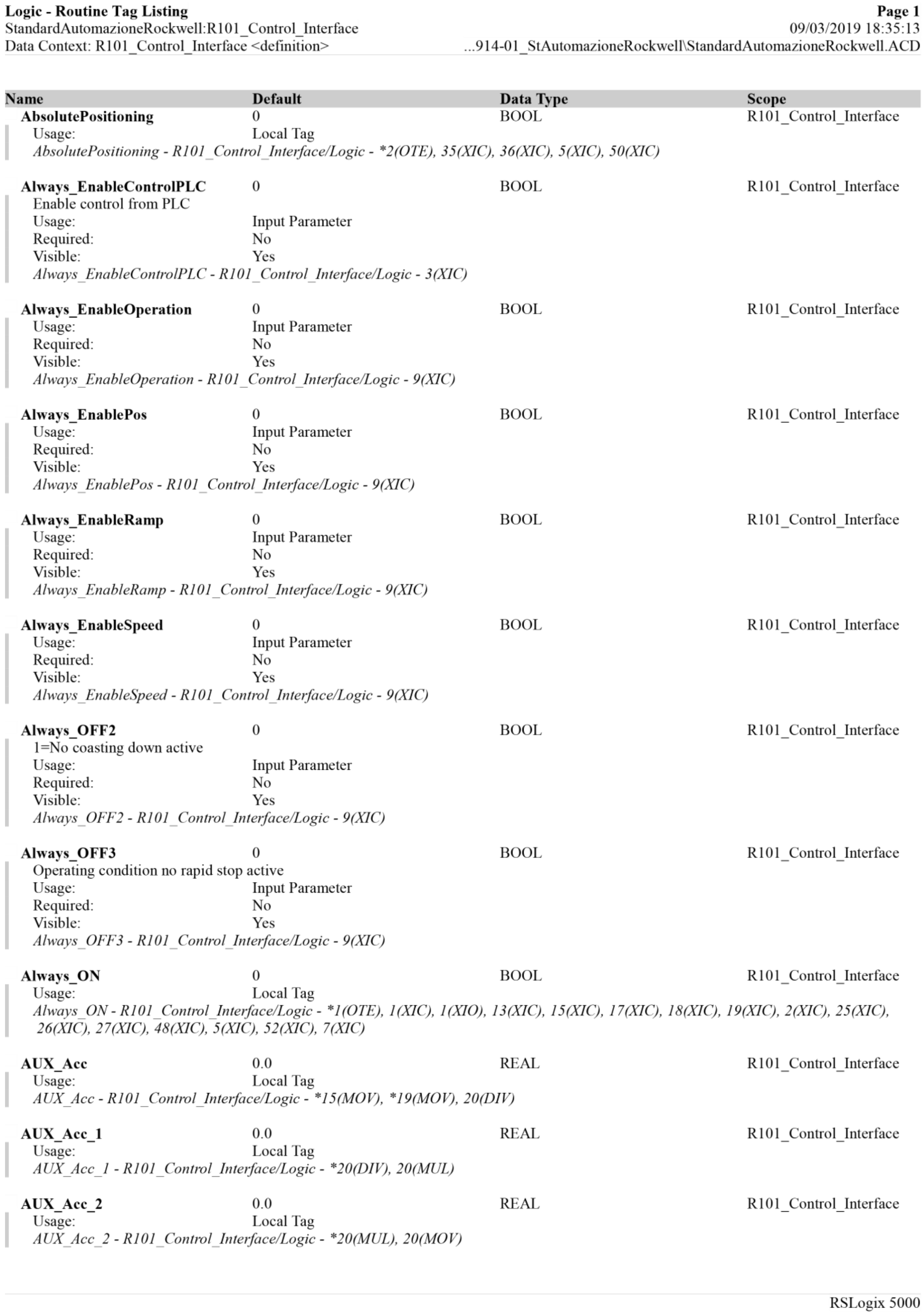
**R100_Standard_NAZARI - Ladder Diagram** **Page 3**
StandardAutomazioneRockwell:MainTask:MainProgram 09/03/2019 18:36:38
Total number of rungs in routine: 6 ...914-01_StAutomazioneRockwell\StandardAutomazioneRockwell.ACD



WRITE commands to the unit

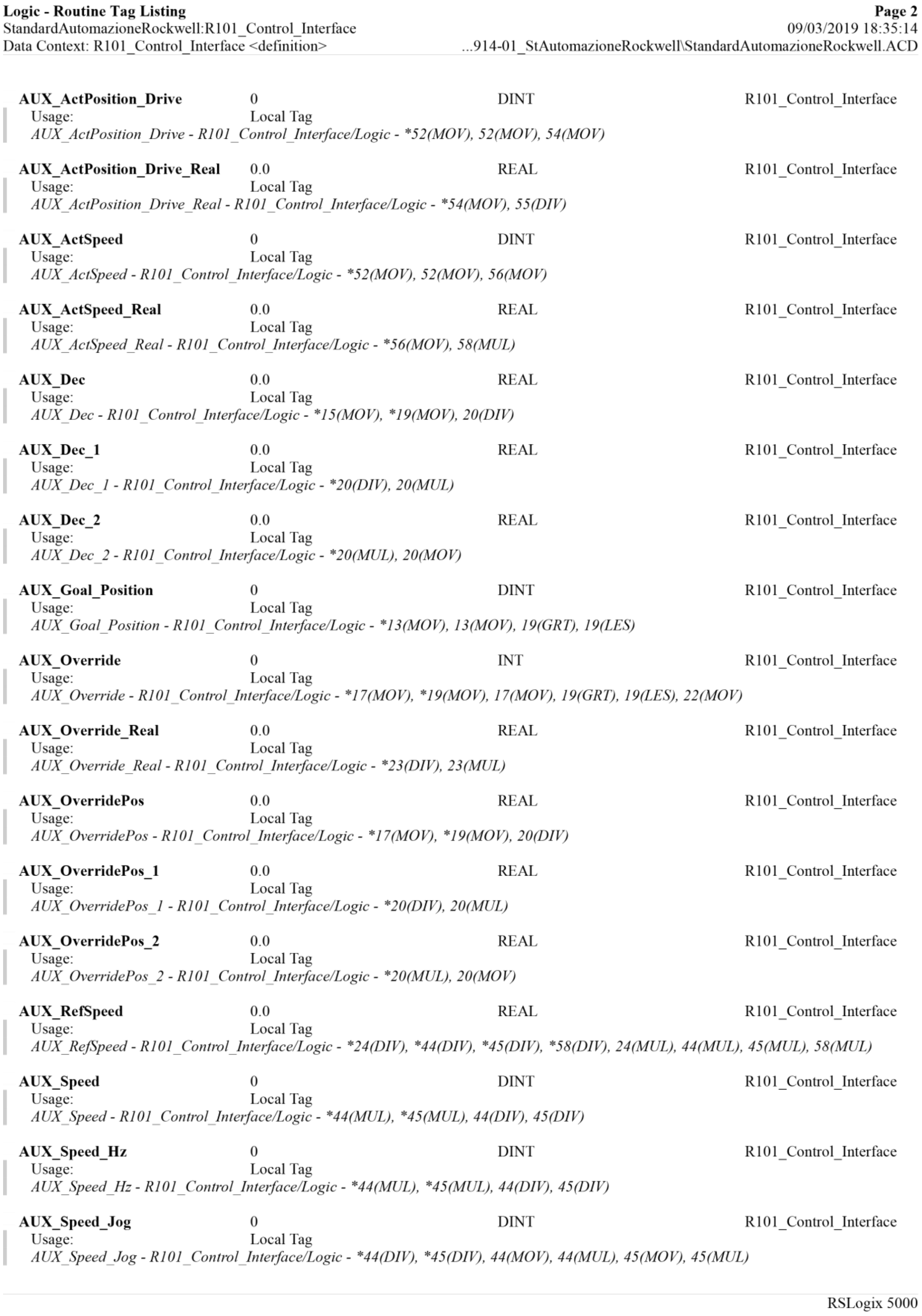
Copy File
Source My_Structure.Output
Dest Inverter:O
Length 1

RSLogix 5000

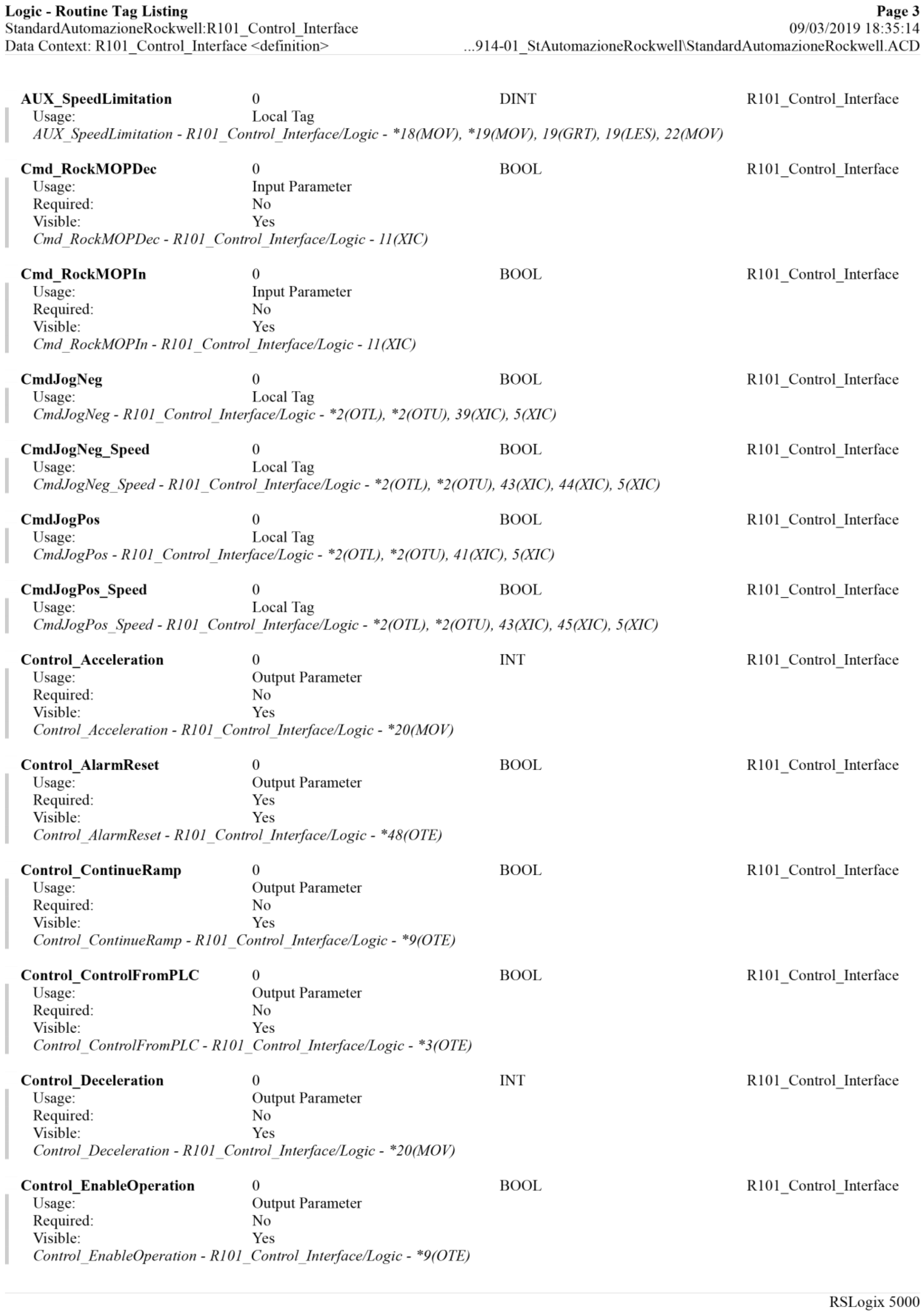


**Logic - Routine Tag Listing** **Page 1**

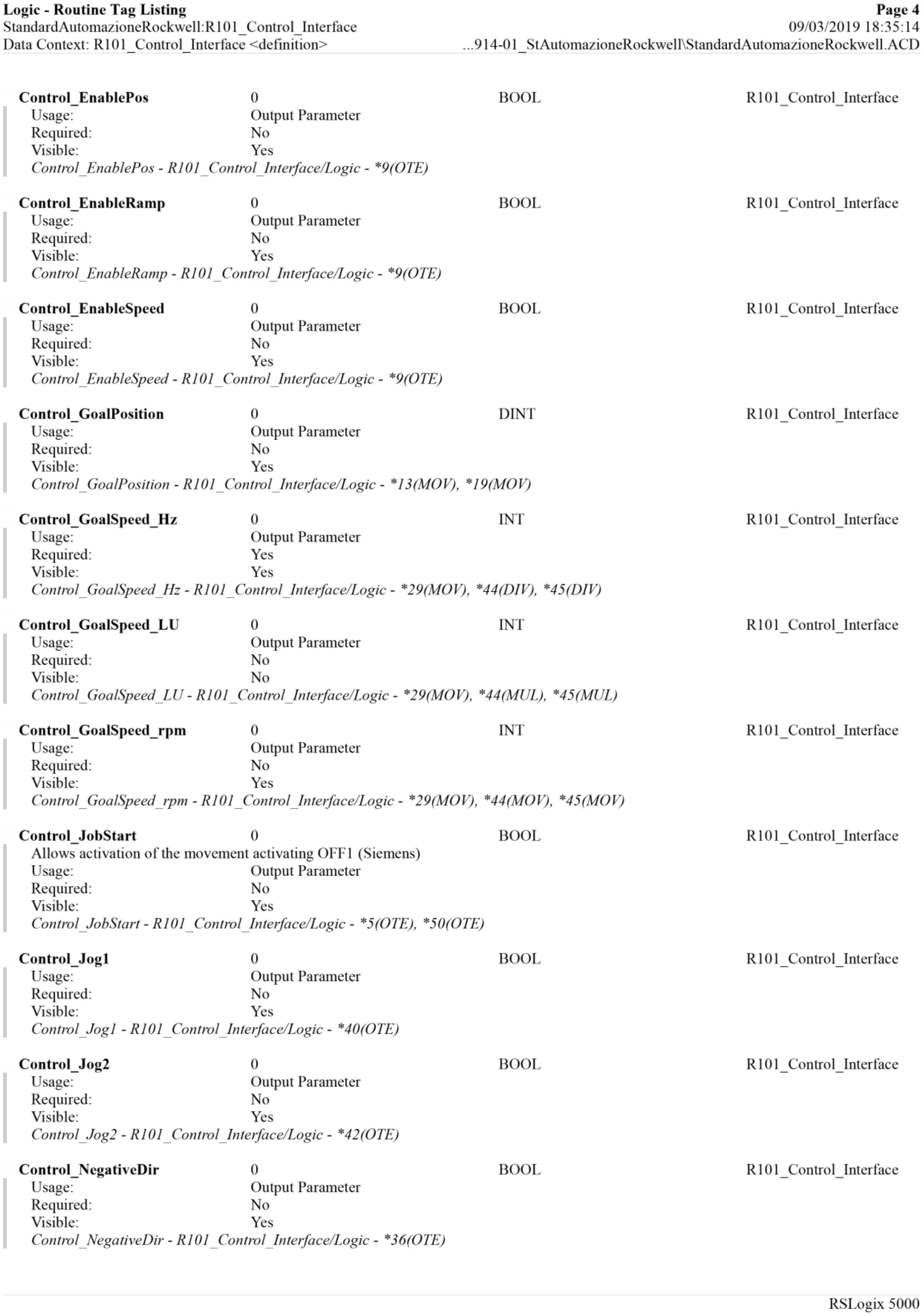
```
StandardAutomazioneRockwell:R101_Control_Interface                                          09/03/2019 18:35:13
Data Context: R101_Control_Interface <definition>              ...914-01_StAutomazioneRockwell\StandardAutomazioneRockwell.ACD

Name                          Default                   Data Type                 Scope
  AbsolutePositioning         0                         BOOL                      R101_Control_Interface
    Usage:                    Local Tag
    AbsolutePositioning - R101_Control_Interface/Logic - *2(OTE), 35(XIC), 36(XIC), 5(XIC), 50(XIC)

  Always_EnableControlPLC     0                         BOOL                      R101_Control_Interface
    Enable control from PLC
    Usage:                    Input Parameter
    Required:                 No
    Visible:                  Yes
    Always_EnableControlPLC - R101_Control_Interface/Logic - 3(XIC)

  Always_EnableOperation      0                         BOOL                      R101_Control_Interface
    Usage:                    Input Parameter
    Required:                 No
    Visible:                  Yes
    Always_EnableOperation - R101_Control_Interface/Logic - 9(XIC)

  Always_EnablePos            0                         BOOL                      R101_Control_Interface
    Usage:                    Input Parameter
    Required:                 No
    Visible:                  Yes
    Always_EnablePos - R101_Control_Interface/Logic - 9(XIC)

  Always_EnableRamp           0                         BOOL                      R101_Control_Interface
    Usage:                    Input Parameter
    Required:                 No
    Visible:                  Yes
    Always_EnableRamp - R101_Control_Interface/Logic - 9(XIC)

  Always_EnableSpeed          0                         BOOL                      R101_Control_Interface
    Usage:                    Input Parameter
    Required:                 No
    Visible:                  Yes
    Always_EnableSpeed - R101_Control_Interface/Logic - 9(XIC)

  Always_OFF2                 0                         BOOL                      R101_Control_Interface
    1=No coasting down active
    Usage:                    Input Parameter
    Required:                 No
    Visible:                  Yes
    Always_OFF2 - R101_Control_Interface/Logic - 9(XIC)

  Always_OFF3                 0                         BOOL                      R101_Control_Interface
    Operating condition no rapid stop active
    Usage:                    Input Parameter
    Required:                 No
    Visible:                  Yes
    Always_OFF3 - R101_Control_Interface/Logic - 9(XIC)

  Always_ON                   0                         BOOL                      R101_Control_Interface
    Usage:                    Local Tag
    Always_ON - R101_Control_Interface/Logic - *1(OTE), 1(XIC), 1(XIO), 13(XIC), 15(XIC), 17(XIC), 18(XIC), 19(XIC), 2(XIC), 25(XIC),
     26(XIC), 27(XIC), 48(XIC), 5(XIC), 52(XIC), 7(XIC)

  AUX_Acc                     0.0                       REAL                      R101_Control_Interface
    Usage:                    Local Tag
    AUX_Acc - R101_Control_Interface/Logic - *15(MOV), *19(MOV), 20(DIV)

  AUX_Acc_1                   0.0                       REAL                      R101_Control_Interface
    Usage:                    Local Tag
    AUX_Acc_1 - R101_Control_Interface/Logic - *20(DIV), 20(MUL)

  AUX_Acc_2                   0.0                       REAL                      R101_Control_Interface
    Usage:                    Local Tag
    AUX_Acc_2 - R101_Control_Interface/Logic - *20(MUL), 20(MOV)

                                                                                                     RSLogix 5000
```



```
Logic - Routine Tag Listing                                                                          Page 2
StandardAutomazioneRockwell:R101_Control_Interface                                      09/03/2019 18:35:14
Data Context: R101_Control_Interface <definition>   ...914-01_StAutomazioneRockwell\StandardAutomazioneRockwell.ACD

  AUX_ActPosition_Drive        0                 DINT              R101_Control_Interface
    Usage:                     Local Tag
    AUX_ActPosition_Drive - R101_Control_Interface/Logic - *52(MOV), 52(MOV), 54(MOV)

  AUX_ActPosition_Drive_Real   0.0               REAL              R101_Control_Interface
    Usage:                     Local Tag
    AUX_ActPosition_Drive_Real - R101_Control_Interface/Logic - *54(MOV), 55(DIV)

  AUX_ActSpeed                 0                 DINT              R101_Control_Interface
    Usage:                     Local Tag
    AUX_ActSpeed - R101_Control_Interface/Logic - *52(MOV), 52(MOV), 56(MOV)

  AUX_ActSpeed_Real            0.0               REAL              R101_Control_Interface
    Usage:                     Local Tag
    AUX_ActSpeed_Real - R101_Control_Interface/Logic - *56(MOV), 58(MUL)

  AUX_Dec                      0.0               REAL              R101_Control_Interface
    Usage:                     Local Tag
    AUX_Dec - R101_Control_Interface/Logic - *15(MOV), *19(MOV), 20(DIV)

  AUX_Dec_1                    0.0               REAL              R101_Control_Interface
    Usage:                     Local Tag
    AUX_Dec_1 - R101_Control_Interface/Logic - *20(DIV), 20(MUL)

  AUX_Dec_2                    0.0               REAL              R101_Control_Interface
    Usage:                     Local Tag
    AUX_Dec_2 - R101_Control_Interface/Logic - *20(MUL), 20(MOV)

  AUX_Goal_Position            0                 DINT              R101_Control_Interface
    Usage:                     Local Tag
    AUX_Goal_Position - R101_Control_Interface/Logic - *13(MOV), 13(MOV), 19(GRT), 19(LES)

  AUX_Override                 0                 INT               R101_Control_Interface
    Usage:                     Local Tag
    AUX_Override - R101_Control_Interface/Logic - *17(MOV), *19(MOV), 17(MOV), 19(GRT), 19(LES), 22(MOV)

  AUX_Override_Real            0.0               REAL              R101_Control_Interface
    Usage:                     Local Tag
    AUX_Override_Real - R101_Control_Interface/Logic - *23(DIV), 23(MUL)

  AUX_OverridePos              0.0               REAL              R101_Control_Interface
    Usage:                     Local Tag
    AUX_OverridePos - R101_Control_Interface/Logic - *17(MOV), *19(MOV), 20(DIV)

  AUX_OverridePos_1            0.0               REAL              R101_Control_Interface
    Usage:                     Local Tag
    AUX_OverridePos_1 - R101_Control_Interface/Logic - *20(DIV), 20(MUL)

  AUX_OverridePos_2            0.0               REAL              R101_Control_Interface
    Usage:                     Local Tag
    AUX_OverridePos_2 - R101_Control_Interface/Logic - *20(MUL), 20(MOV)

  AUX_RefSpeed                 0.0               REAL              R101_Control_Interface
    Usage:                     Local Tag
    AUX_RefSpeed - R101_Control_Interface/Logic - *24(DIV), *44(DIV), *45(DIV), *58(DIV), 24(MUL), 44(MUL), 45(MUL), 58(MUL)

  AUX_Speed                    0                 DINT              R101_Control_Interface
    Usage:                     Local Tag
    AUX_Speed - R101_Control_Interface/Logic - *44(MUL), *45(MUL), 44(DIV), 45(DIV)

  AUX_Speed_Hz                 0                 DINT              R101_Control_Interface
    Usage:                     Local Tag
    AUX_Speed_Hz - R101_Control_Interface/Logic - *44(MUL), *45(MUL), 44(DIV), 45(DIV)

  AUX_Speed_Jog                0                 DINT              R101_Control_Interface
    Usage:                     Local Tag
    AUX_Speed_Jog - R101_Control_Interface/Logic - *44(DIV), *45(DIV), 44(MOV), 44(MUL), 45(MOV), 45(MUL)

                                                                                          RSLogix 5000
```



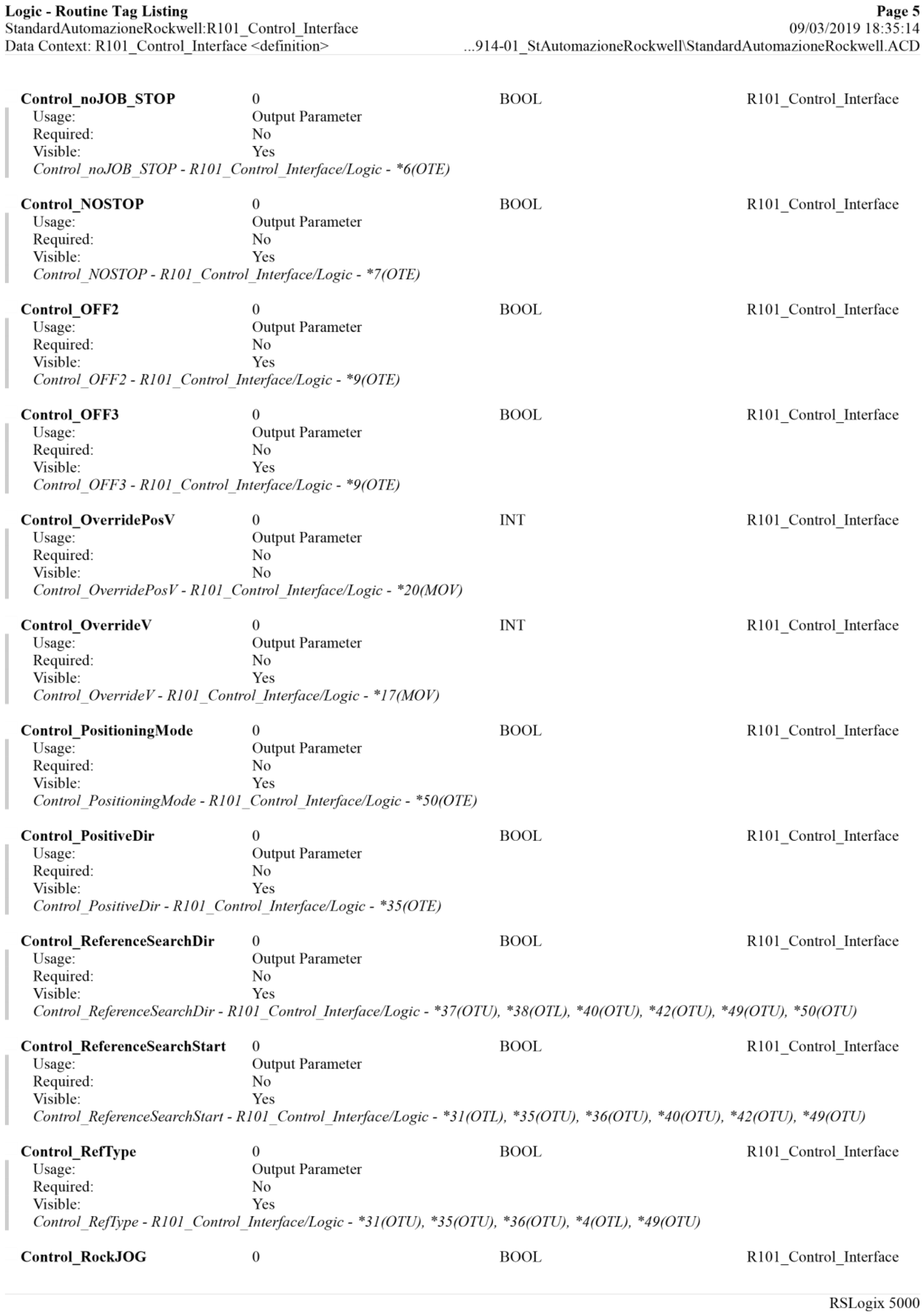
```
Logic - Routine Tag Listing                                                                          Page 3
StandardAutomazioneRockwell:R101_Control_Interface                                     09/03/2019 18:35:14
Data Context: R101_Control_Interface <definition>    ...914-01_StAutomazioneRockwell\StandardAutomazioneRockwell.ACD

  AUX_SpeedLimitation          0                        DINT                     R101_Control_Interface
    Usage:                     Local Tag
    AUX_SpeedLimitation - R101_Control_Interface/Logic - *18(MOV), *19(MOV), 19(GRT), 19(LES), 22(MOV)

  Cmd_RockMOPDec               0                        BOOL                     R101_Control_Interface
    Usage:                     Input Parameter
    Required:                  No
    Visible:                   Yes
    Cmd_RockMOPDec - R101_Control_Interface/Logic - 11(XIC)

  Cmd_RockMOPIn                0                        BOOL                     R101_Control_Interface
    Usage:                     Input Parameter
    Required:                  No
    Visible:                   Yes
    Cmd_RockMOPIn - R101_Control_Interface/Logic - 11(XIC)

  CmdJogNeg                    0                        BOOL                     R101_Control_Interface
    Usage:                     Local Tag
    CmdJogNeg - R101_Control_Interface/Logic - *2(OTL), *2(OTU), 39(XIC), 5(XIC)

  CmdJogNeg_Speed              0                        BOOL                     R101_Control_Interface
    Usage:                     Local Tag
    CmdJogNeg_Speed - R101_Control_Interface/Logic - *2(OTL), *2(OTU), 43(XIC), 44(XIC), 5(XIC)

  CmdJogPos                    0                        BOOL                     R101_Control_Interface
    Usage:                     Local Tag
    CmdJogPos - R101_Control_Interface/Logic - *2(OTL), *2(OTU), 41(XIC), 5(XIC)

  CmdJogPos_Speed              0                        BOOL                     R101_Control_Interface
    Usage:                     Local Tag
    CmdJogPos_Speed - R101_Control_Interface/Logic - *2(OTL), *2(OTU), 43(XIC), 45(XIC), 5(XIC)

  Control_Acceleration         0                        INT                      R101_Control_Interface
    Usage:                     Output Parameter
    Required:                  No
    Visible:                   Yes
    Control_Acceleration - R101_Control_Interface/Logic - *20(MOV)

  Control_AlarmReset           0                        BOOL                     R101_Control_Interface
    Usage:                     Output Parameter
    Required:                  Yes
    Visible:                   Yes
    Control_AlarmReset - R101_Control_Interface/Logic - *48(OTE)

  Control_ContinueRamp         0                        BOOL                     R101_Control_Interface
    Usage:                     Output Parameter
    Required:                  No
    Visible:                   Yes
    Control_ContinueRamp - R101_Control_Interface/Logic - *9(OTE)

  Control_ControlFromPLC       0                        BOOL                     R101_Control_Interface
    Usage:                     Output Parameter
    Required:                  No
    Visible:                   Yes
    Control_ControlFromPLC - R101_Control_Interface/Logic - *3(OTE)

  Control_Deceleration         0                        INT                      R101_Control_Interface
    Usage:                     Output Parameter
    Required:                  No
    Visible:                   Yes
    Control_Deceleration - R101_Control_Interface/Logic - *20(MOV)

  Control_EnableOperation      0                        BOOL                     R101_Control_Interface
    Usage:                     Output Parameter
    Required:                  No
    Visible:                   Yes
    Control_EnableOperation - R101_Control_Interface/Logic - *9(OTE)

                                                                                          RSLogix 5000
```



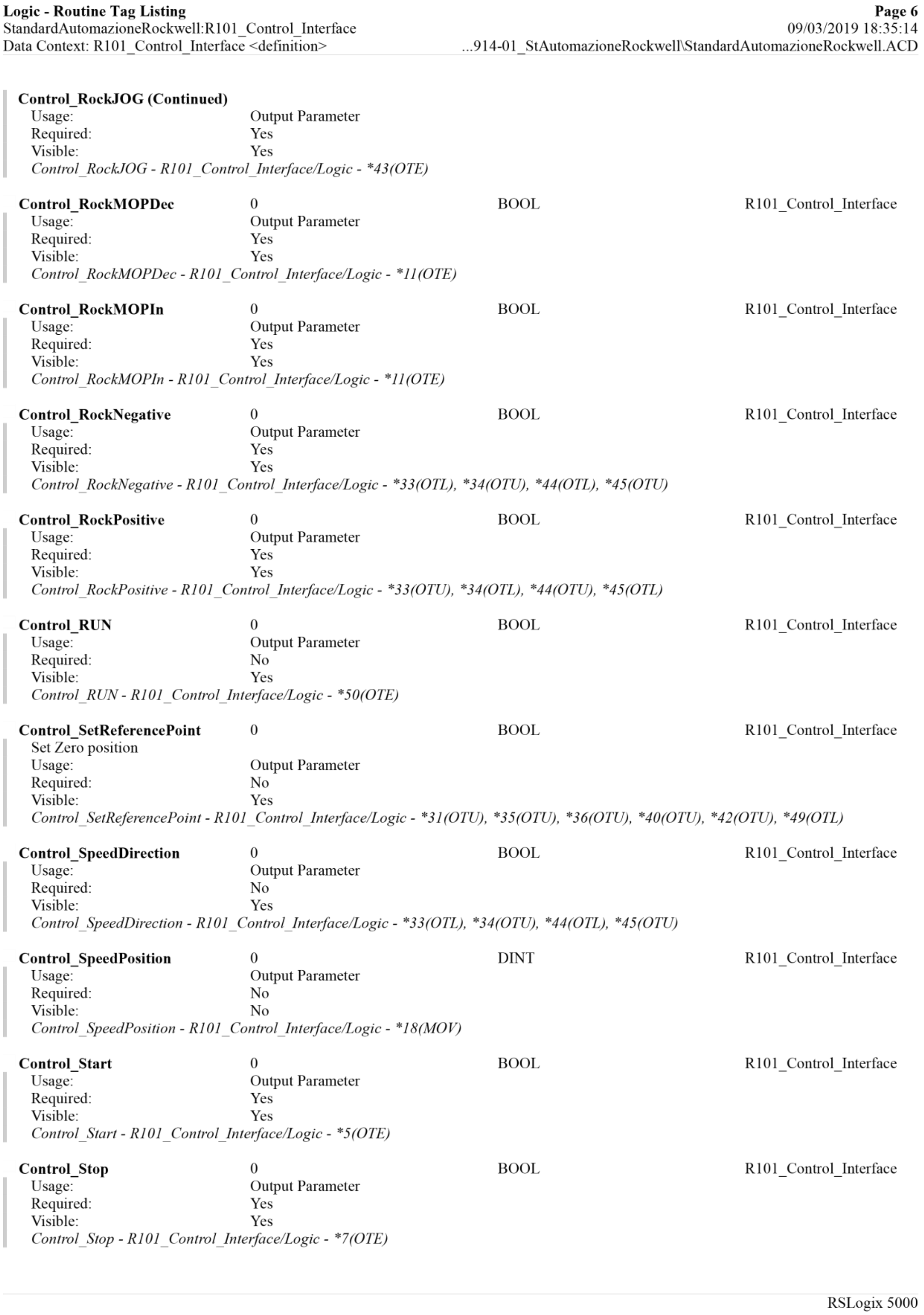
**Logic - Routine Tag Listing** **Page 4**
StandardAutomazioneRockwell:R101_Control_Interface 09/03/2019 18:35:14
Data Context: R101_Control_Interface <definition> ...914-01_StAutomazioneRockwell\StandardAutomazioneRockwell.ACD

```
Control_EnablePos               0                     BOOL              R101_Control_Interface
  Usage:                        Output Parameter
  Required:                     No
  Visible:                      Yes
  Control_EnablePos - R101_Control_Interface/Logic - *9(OTE)

Control_EnableRamp              0                     BOOL              R101_Control_Interface
  Usage:                        Output Parameter
  Required:                     No
  Visible:                      Yes
  Control_EnableRamp - R101_Control_Interface/Logic - *9(OTE)

Control_EnableSpeed             0                     BOOL              R101_Control_Interface
  Usage:                        Output Parameter
  Required:                     No
  Visible:                      Yes
  Control_EnableSpeed - R101_Control_Interface/Logic - *9(OTE)

Control_GoalPosition            0                     DINT              R101_Control_Interface
  Usage:                        Output Parameter
  Required:                     No
  Visible:                      Yes
  Control_GoalPosition - R101_Control_Interface/Logic - *13(MOV), *19(MOV)

Control_GoalSpeed_Hz            0                     INT               R101_Control_Interface
  Usage:                        Output Parameter
  Required:                     Yes
  Visible:                      Yes
  Control_GoalSpeed_Hz - R101_Control_Interface/Logic - *29(MOV), *44(DIV), *45(DIV)

Control_GoalSpeed_LU            0                     INT               R101_Control_Interface
  Usage:                        Output Parameter
  Required:                     No
  Visible:                      No
  Control_GoalSpeed_LU - R101_Control_Interface/Logic - *29(MOV), *44(MUL), *45(MUL)

Control_GoalSpeed_rpm           0                     INT               R101_Control_Interface
  Usage:                        Output Parameter
  Required:                     No
  Visible:                      Yes
  Control_GoalSpeed_rpm - R101_Control_Interface/Logic - *29(MOV), *44(MOV), *45(MOV)

Control_JobStart                0                     BOOL              R101_Control_Interface
  Allows activation of the movement activating OFF1 (Siemens)
  Usage:                        Output Parameter
  Required:                     No
  Visible:                      Yes
  Control_JobStart - R101_Control_Interface/Logic - *5(OTE), *50(OTE)

Control_Jog1                    0                     BOOL              R101_Control_Interface
  Usage:                        Output Parameter
  Required:                     No
  Visible:                      Yes
  Control_Jog1 - R101_Control_Interface/Logic - *40(OTE)

Control_Jog2                    0                     BOOL              R101_Control_Interface
  Usage:                        Output Parameter
  Required:                     No
  Visible:                      Yes
  Control_Jog2 - R101_Control_Interface/Logic - *42(OTE)

Control_NegativeDir             0                     BOOL              R101_Control_Interface
  Usage:                        Output Parameter
  Required:                     No
  Visible:                      Yes
  Control_NegativeDir - R101_Control_Interface/Logic - *36(OTE)
```

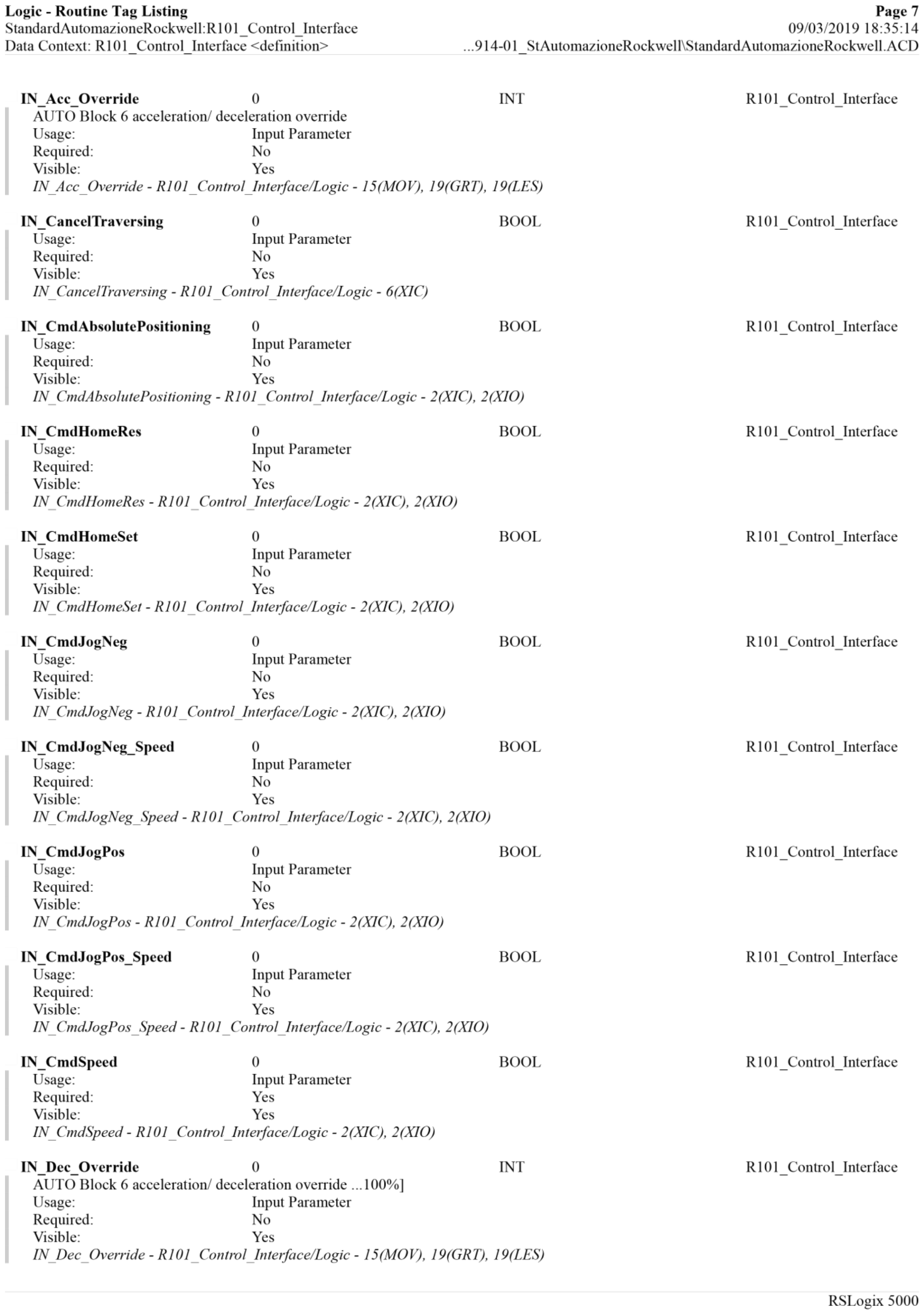
RSLogix 5000



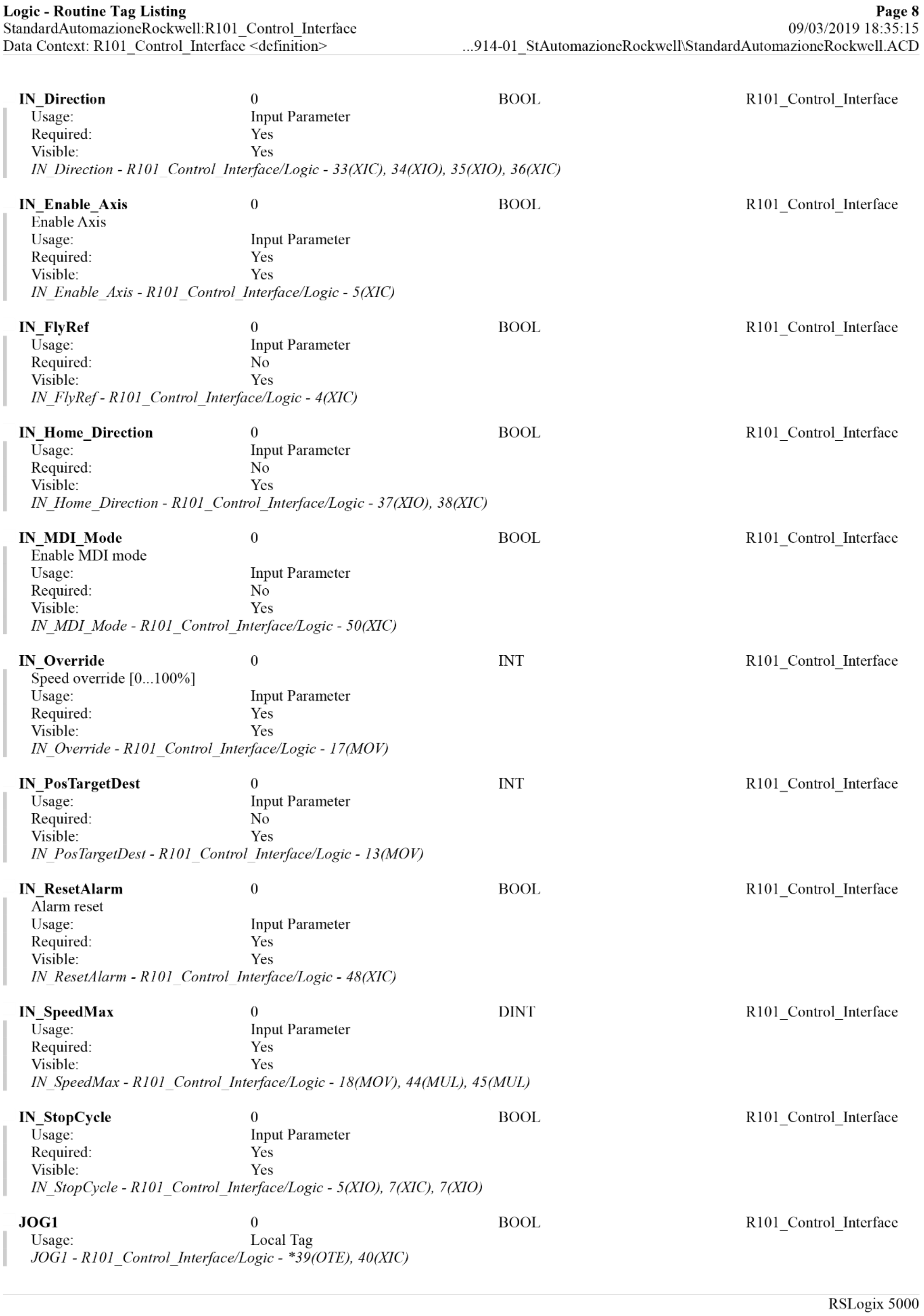
```
Logic - Routine Tag Listing                                                              Page 5
StandardAutomazioneRockwell:R101_Control_Interface                          09/03/2019 18:35:14
Data Context: R101_Control_Interface <definition>   ...914-01_StAutomazioneRockwell\StandardAutomazioneRockwell.ACD

  Control_noJOB_STOP             0                        BOOL                  R101_Control_Interface
    Usage:                       Output Parameter
    Required:                    No
    Visible:                     Yes
    Control_noJOB_STOP - R101_Control_Interface/Logic - *6(OTE)

  Control_NOSTOP                 0                        BOOL                  R101_Control_Interface
    Usage:                       Output Parameter
    Required:                    No
    Visible:                     Yes
    Control_NOSTOP - R101_Control_Interface/Logic - *7(OTE)

  Control_OFF2                   0                        BOOL                  R101_Control_Interface
    Usage:                       Output Parameter
    Required:                    No
    Visible:                     Yes
    Control_OFF2 - R101_Control_Interface/Logic - *9(OTE)

  Control_OFF3                   0                        BOOL                  R101_Control_Interface
    Usage:                       Output Parameter
    Required:                    No
    Visible:                     Yes
    Control_OFF3 - R101_Control_Interface/Logic - *9(OTE)

  Control_OverridePosV           0                        INT                   R101_Control_Interface
    Usage:                       Output Parameter
    Required:                    No
    Visible:                     No
    Control_OverridePosV - R101_Control_Interface/Logic - *20(MOV)

  Control_OverrideV              0                        INT                   R101_Control_Interface
    Usage:                       Output Parameter
    Required:                    No
    Visible:                     Yes
    Control_OverrideV - R101_Control_Interface/Logic - *17(MOV)

  Control_PositioningMode        0                        BOOL                  R101_Control_Interface
    Usage:                       Output Parameter
    Required:                    No
    Visible:                     Yes
    Control_PositioningMode - R101_Control_Interface/Logic - *50(OTE)

  Control_PositiveDir            0                        BOOL                  R101_Control_Interface
    Usage:                       Output Parameter
    Required:                    No
    Visible:                     Yes
    Control_PositiveDir - R101_Control_Interface/Logic - *35(OTE)

  Control_ReferenceSearchDir     0                        BOOL                  R101_Control_Interface
    Usage:                       Output Parameter
    Required:                    No
    Visible:                     Yes
    Control_ReferenceSearchDir - R101_Control_Interface/Logic - *37(OTU), *38(OTL), *40(OTU), *42(OTU), *49(OTU), *50(OTU)

  Control_ReferenceSearchStart   0                        BOOL                  R101_Control_Interface
    Usage:                       Output Parameter
    Required:                    No
    Visible:                     Yes
    Control_ReferenceSearchStart - R101_Control_Interface/Logic - *31(OTL), *35(OTU), *36(OTU), *40(OTU), *42(OTU), *49(OTU)

  Control_RefType                0                        BOOL                  R101_Control_Interface
    Usage:                       Output Parameter
    Required:                    No
    Visible:                     Yes
    Control_RefType - R101_Control_Interface/Logic - *31(OTU), *35(OTU), *36(OTU), *4(OTL), *49(OTU)

  Control_RockJOG                0                        BOOL                  R101_Control_Interface

                                                                                    RSLogix 5000
```



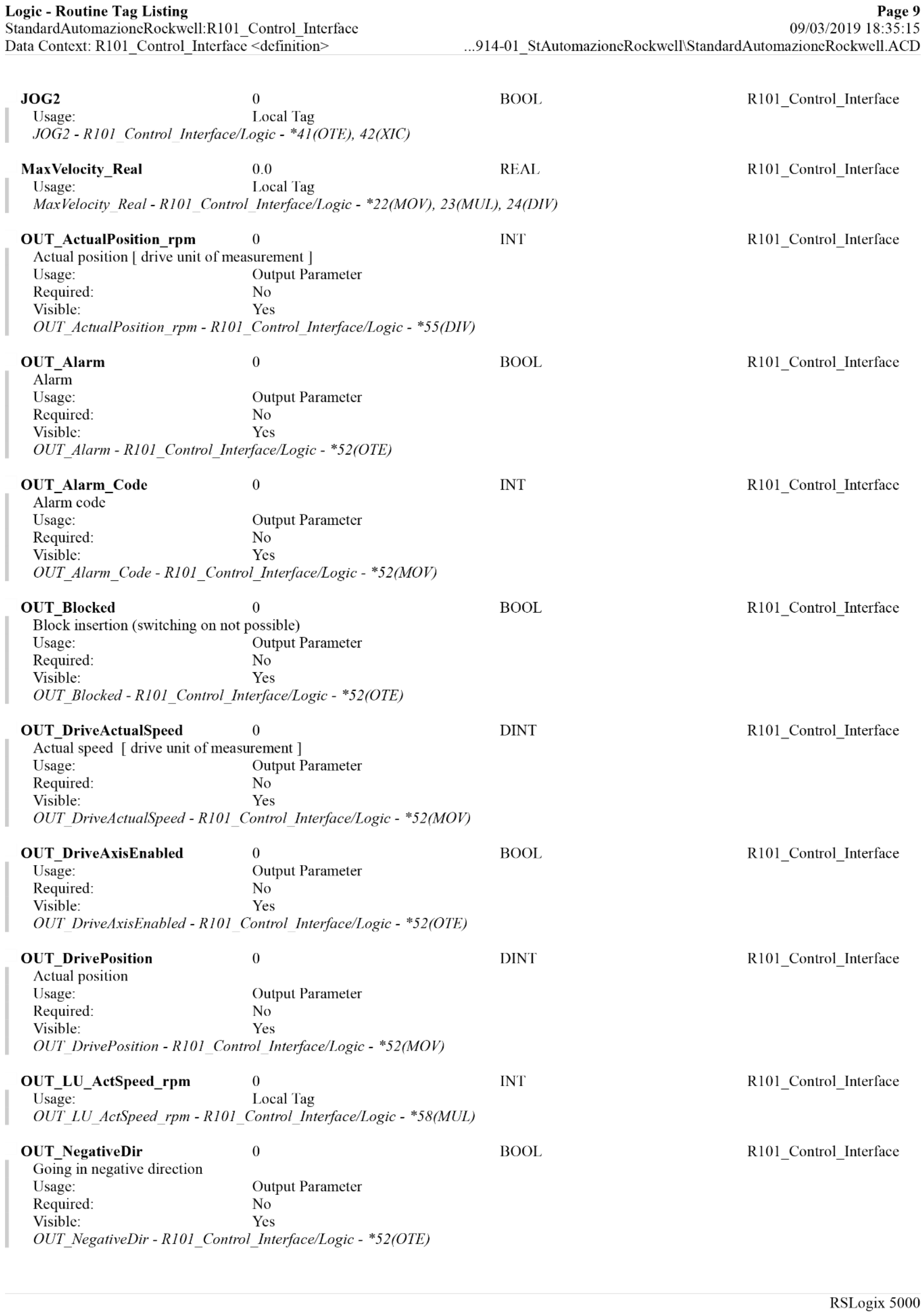
```
Logic - Routine Tag Listing                                                                       Page 6
StandardAutomazioneRockwell:R101_Control_Interface                                     09/03/2019 18:35:14
Data Context: R101_Control_Interface <definition>    ...914-01_StAutomazioneRockwell\StandardAutomazioneRockwell.ACD

Control_RockJOG (Continued)
  Usage:                      Output Parameter
  Required:                   Yes
  Visible:                    Yes
  Control_RockJOG - R101_Control_Interface/Logic - *43(OTE)

Control_RockMOPDec            0                        BOOL                     R101_Control_Interface
  Usage:                      Output Parameter
  Required:                   Yes
  Visible:                    Yes
  Control_RockMOPDec - R101_Control_Interface/Logic - *11(OTE)

Control_RockMOPIn             0                        BOOL                     R101_Control_Interface
  Usage:                      Output Parameter
  Required:                   Yes
  Visible:                    Yes
  Control_RockMOPIn - R101_Control_Interface/Logic - *11(OTE)

Control_RockNegative          0                        BOOL                     R101_Control_Interface
  Usage:                      Output Parameter
  Required:                   Yes
  Visible:                    Yes
  Control_RockNegative - R101_Control_Interface/Logic - *33(OTL), *34(OTU), *44(OTL), *45(OTU)

Control_RockPositive          0                        BOOL                     R101_Control_Interface
  Usage:                      Output Parameter
  Required:                   Yes
  Visible:                    Yes
  Control_RockPositive - R101_Control_Interface/Logic - *33(OTU), *34(OTL), *44(OTU), *45(OTL)

Control_RUN                   0                        BOOL                     R101_Control_Interface
  Usage:                      Output Parameter
  Required:                   No
  Visible:                    Yes
  Control_RUN - R101_Control_Interface/Logic - *50(OTE)

Control_SetReferencePoint     0                        BOOL                     R101_Control_Interface
  Set Zero position
  Usage:                      Output Parameter
  Required:                   No
  Visible:                    Yes
  Control_SetReferencePoint - R101_Control_Interface/Logic - *31(OTU), *35(OTU), *36(OTU), *40(OTU), *42(OTU), *49(OTL)

Control_SpeedDirection        0                        BOOL                     R101_Control_Interface
  Usage:                      Output Parameter
  Required:                   No
  Visible:                    Yes
  Control_SpeedDirection - R101_Control_Interface/Logic - *33(OTL), *34(OTU), *44(OTL), *45(OTU)

Control_SpeedPosition         0                        DINT                     R101_Control_Interface
  Usage:                      Output Parameter
  Required:                   No
  Visible:                    No
  Control_SpeedPosition - R101_Control_Interface/Logic - *18(MOV)

Control_Start                 0                        BOOL                     R101_Control_Interface
  Usage:                      Output Parameter
  Required:                   Yes
  Visible:                    Yes
  Control_Start - R101_Control_Interface/Logic - *5(OTE)

Control_Stop                  0                        BOOL                     R101_Control_Interface
  Usage:                      Output Parameter
  Required:                   Yes
  Visible:                    Yes
  Control_Stop - R101_Control_Interface/Logic - *7(OTE)

                                                                                          RSLogix 5000
```



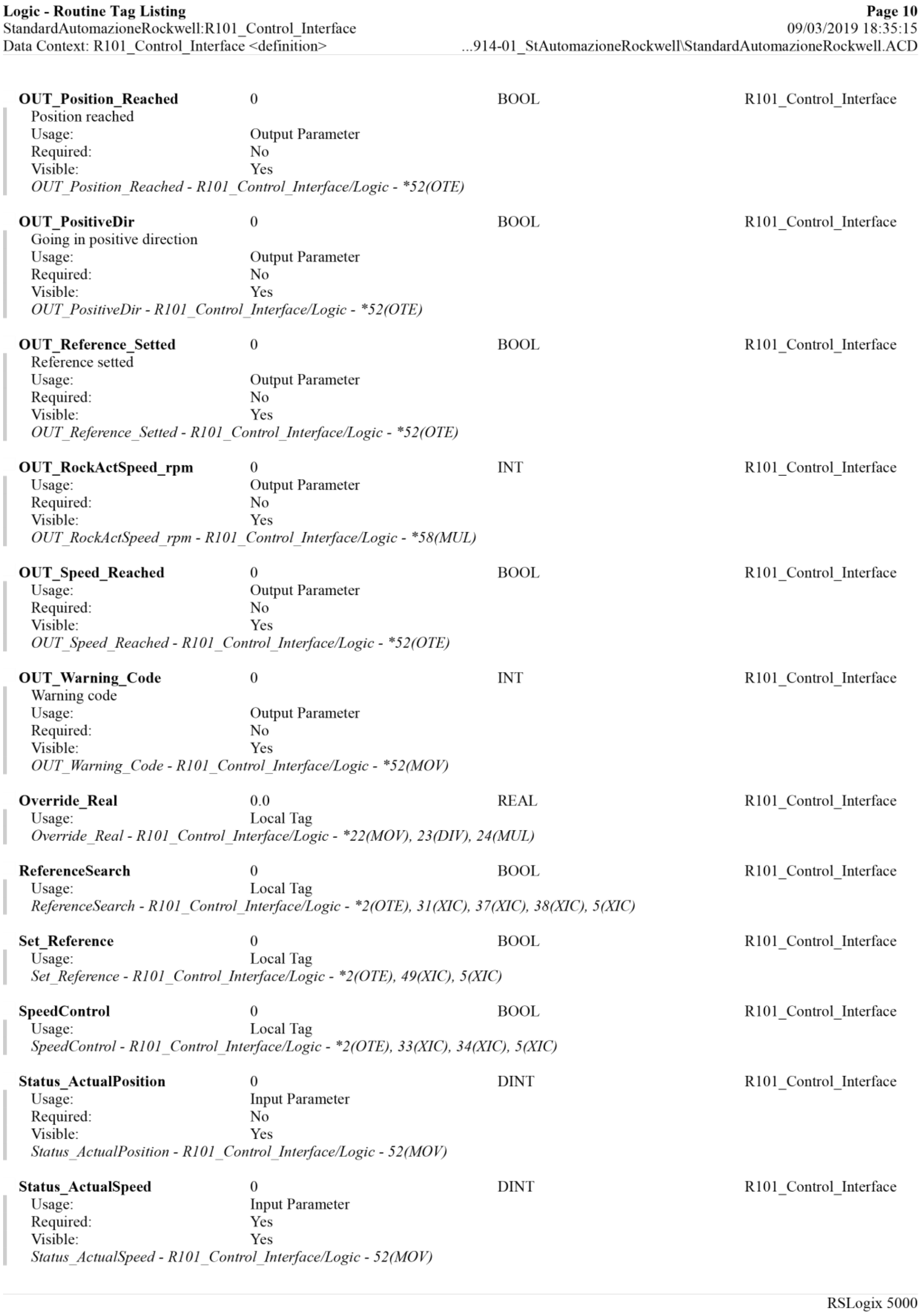
```
Logic - Routine Tag Listing                                                                     Page 7
StandardAutomazioneRockwell:R101_Control_Interface                                 09/03/2019 18:35:14
Data Context: R101_Control_Interface <definition>      ...914-01_StAutomazioneRockwell\StandardAutomazioneRockwell.ACD

IN_Acc_Override                 0                        INT                     R101_Control_Interface
  AUTO Block 6 acceleration/ deceleration override
  Usage:                        Input Parameter
  Required:                     No
  Visible:                      Yes
  IN_Acc_Override - R101_Control_Interface/Logic - 15(MOV), 19(GRT), 19(LES)

IN_CancelTraversing             0                        BOOL                    R101_Control_Interface
  Usage:                        Input Parameter
  Required:                     No
  Visible:                      Yes
  IN_CancelTraversing - R101_Control_Interface/Logic - 6(XIC)

IN_CmdAbsolutePositioning       0                        BOOL                    R101_Control_Interface
  Usage:                        Input Parameter
  Required:                     No
  Visible:                      Yes
  IN_CmdAbsolutePositioning - R101_Control_Interface/Logic - 2(XIC), 2(XIO)

IN_CmdHomeRes                   0                        BOOL                    R101_Control_Interface
  Usage:                        Input Parameter
  Required:                     No
  Visible:                      Yes
  IN_CmdHomeRes - R101_Control_Interface/Logic - 2(XIC), 2(XIO)

IN_CmdHomeSet                   0                        BOOL                    R101_Control_Interface
  Usage:                        Input Parameter
  Required:                     No
  Visible:                      Yes
  IN_CmdHomeSet - R101_Control_Interface/Logic - 2(XIC), 2(XIO)

IN_CmdJogNeg                    0                        BOOL                    R101_Control_Interface
  Usage:                        Input Parameter
  Required:                     No
  Visible:                      Yes
  IN_CmdJogNeg - R101_Control_Interface/Logic - 2(XIC), 2(XIO)

IN_CmdJogNeg_Speed              0                        BOOL                    R101_Control_Interface
  Usage:                        Input Parameter
  Required:                     No
  Visible:                      Yes
  IN_CmdJogNeg_Speed - R101_Control_Interface/Logic - 2(XIC), 2(XIO)

IN_CmdJogPos                    0                        BOOL                    R101_Control_Interface
  Usage:                        Input Parameter
  Required:                     No
  Visible:                      Yes
  IN_CmdJogPos - R101_Control_Interface/Logic - 2(XIC), 2(XIO)

IN_CmdJogPos_Speed              0                        BOOL                    R101_Control_Interface
  Usage:                        Input Parameter
  Required:                     No
  Visible:                      Yes
  IN_CmdJogPos_Speed - R101_Control_Interface/Logic - 2(XIC), 2(XIO)

IN_CmdSpeed                     0                        BOOL                    R101_Control_Interface
  Usage:                        Input Parameter
  Required:                     Yes
  Visible:                      Yes
  IN_CmdSpeed - R101_Control_Interface/Logic - 2(XIC), 2(XIO)

IN_Dec_Override                 0                        INT                     R101_Control_Interface
  AUTO Block 6 acceleration/ deceleration override ...100%]
  Usage:                        Input Parameter
  Required:                     No
  Visible:                      Yes
  IN_Dec_Override - R101_Control_Interface/Logic - 15(MOV), 19(GRT), 19(LES)

                                                                                          RSLogix 5000
```



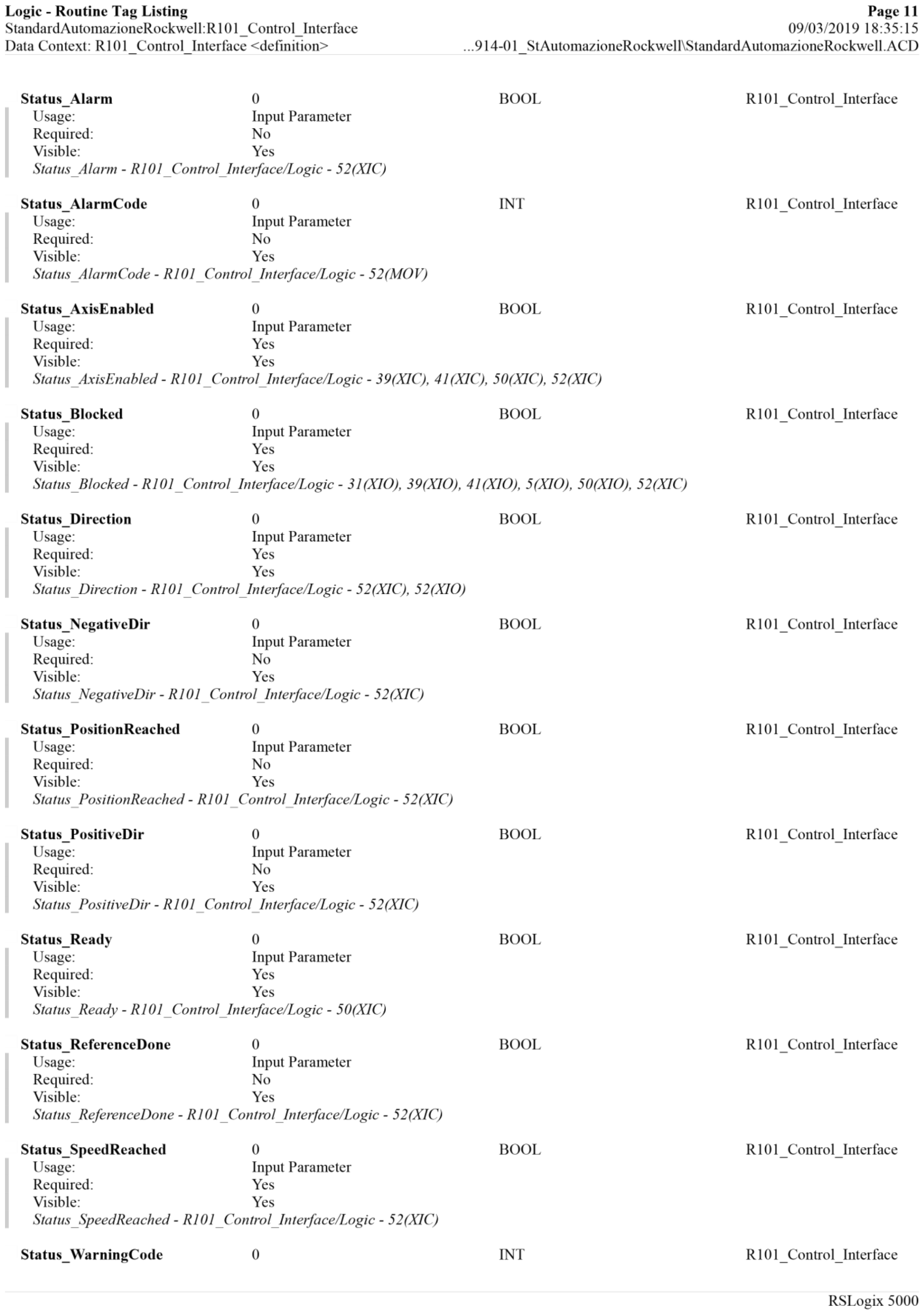
```
Logic - Routine Tag Listing                                                                       Page 8
StandardAutomazioneRockwell:R101_Control_Interface                                    09/03/2019 18:35:15
Data Context: R101_Control_Interface <definition>      ...914-01_StAutomazioneRockwell\StandardAutomazioneRockwell.ACD

  IN_Direction              0                        BOOL                      R101_Control_Interface
    Usage:                  Input Parameter
    Required:               Yes
    Visible:                Yes
    IN_Direction - R101_Control_Interface/Logic - 33(XIC), 34(XIO), 35(XIO), 36(XIC)

  IN_Enable_Axis            0                        BOOL                      R101_Control_Interface
    Enable Axis
    Usage:                  Input Parameter
    Required:               Yes
    Visible:                Yes
    IN_Enable_Axis - R101_Control_Interface/Logic - 5(XIC)

  IN_FlyRef                 0                        BOOL                      R101_Control_Interface
    Usage:                  Input Parameter
    Required:               No
    Visible:                Yes
    IN_FlyRef - R101_Control_Interface/Logic - 4(XIC)

  IN_Home_Direction         0                        BOOL                      R101_Control_Interface
    Usage:                  Input Parameter
    Required:               No
    Visible:                Yes
    IN_Home_Direction - R101_Control_Interface/Logic - 37(XIO), 38(XIC)

  IN_MDI_Mode               0                        BOOL                      R101_Control_Interface
    Enable MDI mode
    Usage:                  Input Parameter
    Required:               No
    Visible:                Yes
    IN_MDI_Mode - R101_Control_Interface/Logic - 50(XIC)

  IN_Override               0                        INT                       R101_Control_Interface
    Speed override [0...100%]
    Usage:                  Input Parameter
    Required:               Yes
    Visible:                Yes
    IN_Override - R101_Control_Interface/Logic - 17(MOV)

  IN_PosTargetDest          0                        INT                       R101_Control_Interface
    Usage:                  Input Parameter
    Required:               No
    Visible:                Yes
    IN_PosTargetDest - R101_Control_Interface/Logic - 13(MOV)

  IN_ResetAlarm             0                        BOOL                      R101_Control_Interface
    Alarm reset
    Usage:                  Input Parameter
    Required:               Yes
    Visible:                Yes
    IN_ResetAlarm - R101_Control_Interface/Logic - 48(XIC)

  IN_SpeedMax               0                        DINT                      R101_Control_Interface
    Usage:                  Input Parameter
    Required:               Yes
    Visible:                Yes
    IN_SpeedMax - R101_Control_Interface/Logic - 18(MOV), 44(MUL), 45(MUL)

  IN_StopCycle              0                        BOOL                      R101_Control_Interface
    Usage:                  Input Parameter
    Required:               Yes
    Visible:                Yes
    IN_StopCycle - R101_Control_Interface/Logic - 5(XIO), 7(XIC), 7(XIO)

  JOG1                      0                        BOOL                      R101_Control_Interface
    Usage:                  Local Tag
    JOG1 - R101_Control_Interface/Logic - *39(OTE), 40(XIC)

                                                                                          RSLogix 5000
```



```
Logic - Routine Tag Listing                                                              Page 9
StandardAutomazioneRockwell:R101_Control_Interface                          09/03/2019 18:35:15
Data Context: R101_Control_Interface <definition>   ...914-01_StAutomazioneRockwell\StandardAutomazioneRockwell.ACD

JOG2                        0                     BOOL                    R101_Control_Interface
  Usage:                    Local Tag
  JOG2 - R101_Control_Interface/Logic - *41(OTE), 42(XIC)

MaxVelocity_Real            0.0                   REAL                    R101_Control_Interface
  Usage:                    Local Tag
  MaxVelocity_Real - R101_Control_Interface/Logic - *22(MOV), 23(MUL), 24(DIV)

OUT_ActualPosition_rpm      0                     INT                     R101_Control_Interface
  Actual position [ drive unit of measurement ]
  Usage:                    Output Parameter
  Required:                 No
  Visible:                  Yes
  OUT_ActualPosition_rpm - R101_Control_Interface/Logic - *55(DIV)

OUT_Alarm                   0                     BOOL                    R101_Control_Interface
  Alarm
  Usage:                    Output Parameter
  Required:                 No
  Visible:                  Yes
  OUT_Alarm - R101_Control_Interface/Logic - *52(OTE)

OUT_Alarm_Code              0                     INT                     R101_Control_Interface
  Alarm code
  Usage:                    Output Parameter
  Required:                 No
  Visible:                  Yes
  OUT_Alarm_Code - R101_Control_Interface/Logic - *52(MOV)

OUT_Blocked                 0                     BOOL                    R101_Control_Interface
  Block insertion (switching on not possible)
  Usage:                    Output Parameter
  Required:                 No
  Visible:                  Yes
  OUT_Blocked - R101_Control_Interface/Logic - *52(OTE)

OUT_DriveActualSpeed        0                     DINT                    R101_Control_Interface
  Actual speed  [ drive unit of measurement ]
  Usage:                    Output Parameter
  Required:                 No
  Visible:                  Yes
  OUT_DriveActualSpeed - R101_Control_Interface/Logic - *52(MOV)

OUT_DriveAxisEnabled        0                     BOOL                    R101_Control_Interface
  Usage:                    Output Parameter
  Required:                 No
  Visible:                  Yes
  OUT_DriveAxisEnabled - R101_Control_Interface/Logic - *52(OTE)

OUT_DrivePosition           0                     DINT                    R101_Control_Interface
  Actual position
  Usage:                    Output Parameter
  Required:                 No
  Visible:                  Yes
  OUT_DrivePosition - R101_Control_Interface/Logic - *52(MOV)

OUT_LU_ActSpeed_rpm         0                     INT                     R101_Control_Interface
  Usage:                    Local Tag
  OUT_LU_ActSpeed_rpm - R101_Control_Interface/Logic - *58(MUL)

OUT_NegativeDir             0                     BOOL                    R101_Control_Interface
  Going in negative direction
  Usage:                    Output Parameter
  Required:                 No
  Visible:                  Yes
  OUT_NegativeDir - R101_Control_Interface/Logic - *52(OTE)

                                                                                   RSLogix 5000
```



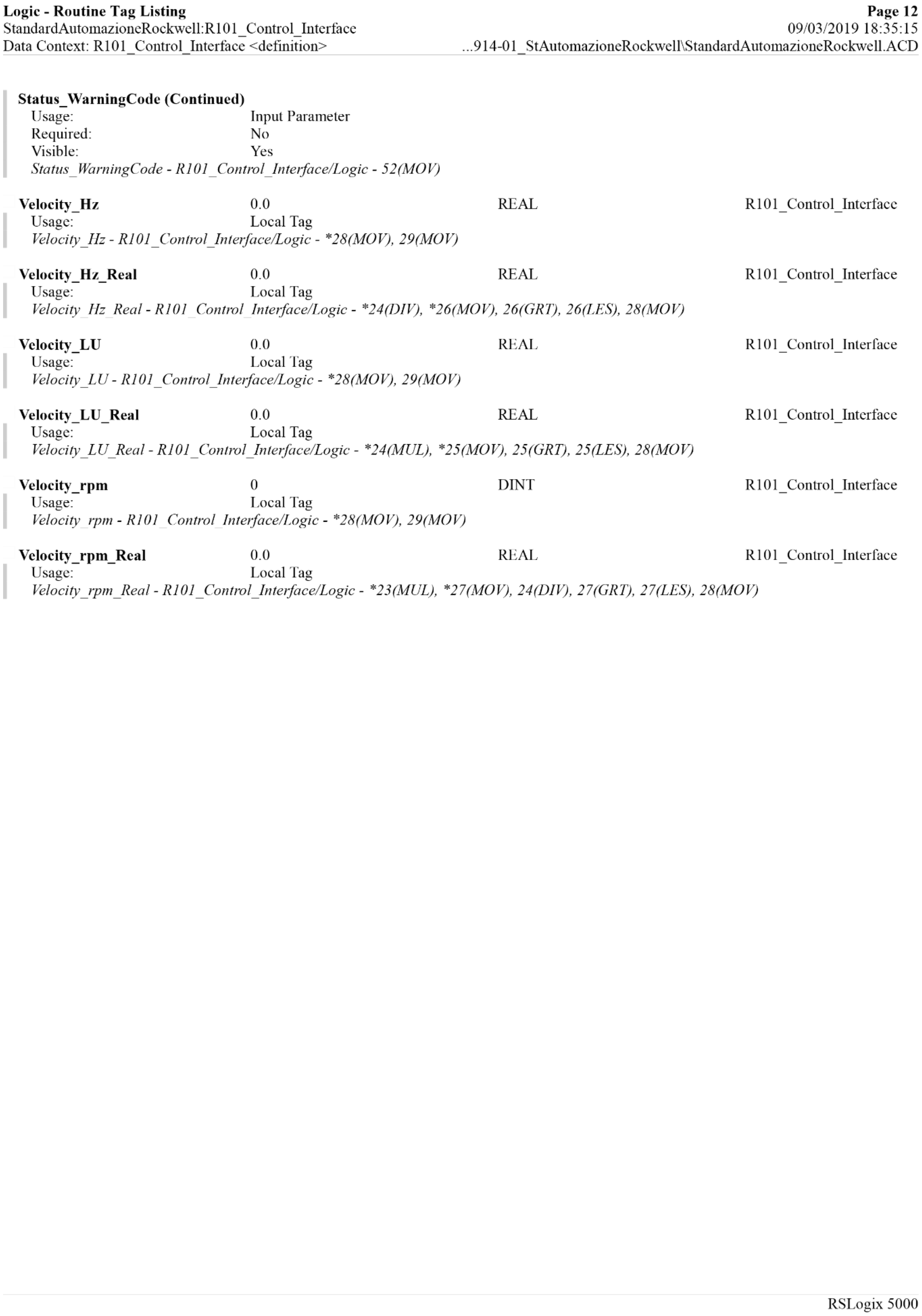
```
Logic - Routine Tag Listing                                                              Page 10
StandardAutomazioneRockwell:R101_Control_Interface                                 09/03/2019 18:35:15
Data Context: R101_Control_Interface <definition>      ...914-01_StAutomazioneRockwell\StandardAutomazioneRockwell.ACD

OUT_Position_Reached      0                       BOOL                    R101_Control_Interface
  Position reached
  Usage:                  Output Parameter
  Required:               No
  Visible:                Yes
  OUT_Position_Reached - R101_Control_Interface/Logic - *52(OTE)

OUT_PositiveDir           0                       BOOL                    R101_Control_Interface
  Going in positive direction
  Usage:                  Output Parameter
  Required:               No
  Visible:                Yes
  OUT_PositiveDir - R101_Control_Interface/Logic - *52(OTE)

OUT_Reference_Setted      0                       BOOL                    R101_Control_Interface
  Reference setted
  Usage:                  Output Parameter
  Required:               No
  Visible:                Yes
  OUT_Reference_Setted - R101_Control_Interface/Logic - *52(OTE)

OUT_RockActSpeed_rpm      0                       INT                     R101_Control_Interface
  Usage:                  Output Parameter
  Required:               No
  Visible:                Yes
  OUT_RockActSpeed_rpm - R101_Control_Interface/Logic - *58(MUL)

OUT_Speed_Reached         0                       BOOL                    R101_Control_Interface
  Usage:                  Output Parameter
  Required:               No
  Visible:                Yes
  OUT_Speed_Reached - R101_Control_Interface/Logic - *52(OTE)

OUT_Warning_Code          0                       INT                     R101_Control_Interface
  Warning code
  Usage:                  Output Parameter
  Required:               No
  Visible:                Yes
  OUT_Warning_Code - R101_Control_Interface/Logic - *52(MOV)

Override_Real             0.0                     REAL                    R101_Control_Interface
  Usage:                  Local Tag
  Override_Real - R101_Control_Interface/Logic - *22(MOV), 23(DIV), 24(MUL)

ReferenceSearch           0                       BOOL                    R101_Control_Interface
  Usage:                  Local Tag
  ReferenceSearch - R101_Control_Interface/Logic - *2(OTE), 31(XIC), 37(XIC), 38(XIC), 5(XIC)

Set_Reference             0                       BOOL                    R101_Control_Interface
  Usage:                  Local Tag
  Set_Reference - R101_Control_Interface/Logic - *2(OTE), 49(XIC), 5(XIC)

SpeedControl              0                       BOOL                    R101_Control_Interface
  Usage:                  Local Tag
  SpeedControl - R101_Control_Interface/Logic - *2(OTE), 33(XIC), 34(XIC), 5(XIC)

Status_ActualPosition     0                       DINT                    R101_Control_Interface
  Usage:                  Input Parameter
  Required:               No
  Visible:                Yes
  Status_ActualPosition - R101_Control_Interface/Logic - 52(MOV)

Status_ActualSpeed        0                       DINT                    R101_Control_Interface
  Usage:                  Input Parameter
  Required:               Yes
  Visible:                Yes
  Status_ActualSpeed - R101_Control_Interface/Logic - 52(MOV)

                                                                                   RSLogix 5000
```



**Logic - Routine Tag Listing** **Page 11**
StandardAutomazioneRockwell:R101_Control_Interface 09/03/2019 18:35:15
Data Context: R101_Control_Interface <definition> ...914-01_StAutomazioneRockwell\StandardAutomazioneRockwell.ACD

```
Status_Alarm                 0                  BOOL            R101_Control_Interface
  Usage:                     Input Parameter
  Required:                  No
  Visible:                   Yes
  Status_Alarm - R101_Control_Interface/Logic - 52(XIC)

Status_AlarmCode             0                  INT             R101_Control_Interface
  Usage:                     Input Parameter
  Required:                  No
  Visible:                   Yes
  Status_AlarmCode - R101_Control_Interface/Logic - 52(MOV)

Status_AxisEnabled           0                  BOOL            R101_Control_Interface
  Usage:                     Input Parameter
  Required:                  Yes
  Visible:                   Yes
  Status_AxisEnabled - R101_Control_Interface/Logic - 39(XIC), 41(XIC), 50(XIC), 52(XIC)

Status_Blocked               0                  BOOL            R101_Control_Interface
  Usage:                     Input Parameter
  Required:                  Yes
  Visible:                   Yes
  Status_Blocked - R101_Control_Interface/Logic - 31(XIO), 39(XIO), 41(XIO), 5(XIO), 50(XIO), 52(XIC)

Status_Direction             0                  BOOL            R101_Control_Interface
  Usage:                     Input Parameter
  Required:                  Yes
  Visible:                   Yes
  Status_Direction - R101_Control_Interface/Logic - 52(XIC), 52(XIO)

Status_NegativeDir           0                  BOOL            R101_Control_Interface
  Usage:                     Input Parameter
  Required:                  No
  Visible:                   Yes
  Status_NegativeDir - R101_Control_Interface/Logic - 52(XIC)

Status_PositionReached       0                  BOOL            R101_Control_Interface
  Usage:                     Input Parameter
  Required:                  No
  Visible:                   Yes
  Status_PositionReached - R101_Control_Interface/Logic - 52(XIC)

Status_PositiveDir           0                  BOOL            R101_Control_Interface
  Usage:                     Input Parameter
  Required:                  No
  Visible:                   Yes
  Status_PositiveDir - R101_Control_Interface/Logic - 52(XIC)

Status_Ready                 0                  BOOL            R101_Control_Interface
  Usage:                     Input Parameter
  Required:                  Yes
  Visible:                   Yes
  Status_Ready - R101_Control_Interface/Logic - 50(XIC)

Status_ReferenceDone         0                  BOOL            R101_Control_Interface
  Usage:                     Input Parameter
  Required:                  No
  Visible:                   Yes
  Status_ReferenceDone - R101_Control_Interface/Logic - 52(XIC)

Status_SpeedReached          0                  BOOL            R101_Control_Interface
  Usage:                     Input Parameter
  Required:                  Yes
  Visible:                   Yes
  Status_SpeedReached - R101_Control_Interface/Logic - 52(XIC)

Status_WarningCode           0                  INT             R101_Control_Interface
```

RSLogix 5000



**Logic - Routine Tag Listing** **Page 12**
StandardAutomazioneRockwell:R101_Control_Interface 09/03/2019 18:35:15
Data Context: R101_Control_Interface <definition> ...914-01_StAutomazioneRockwell\StandardAutomazioneRockwell.ACD

**Status_WarningCode (Continued)**
Usage: Input Parameter
Required: No
Visible: Yes
*Status_WarningCode - R101_Control_Interface/Logic - 52(MOV)*

**Velocity_Hz** 0.0 REAL R101_Control_Interface
Usage: Local Tag
*Velocity_Hz - R101_Control_Interface/Logic - *28(MOV), 29(MOV)*

**Velocity_Hz_Real** 0.0 REAL R101_Control_Interface
Usage: Local Tag
*Velocity_Hz_Real - R101_Control_Interface/Logic - *24(DIV), *26(MOV), 26(GRT), 26(LES), 28(MOV)*

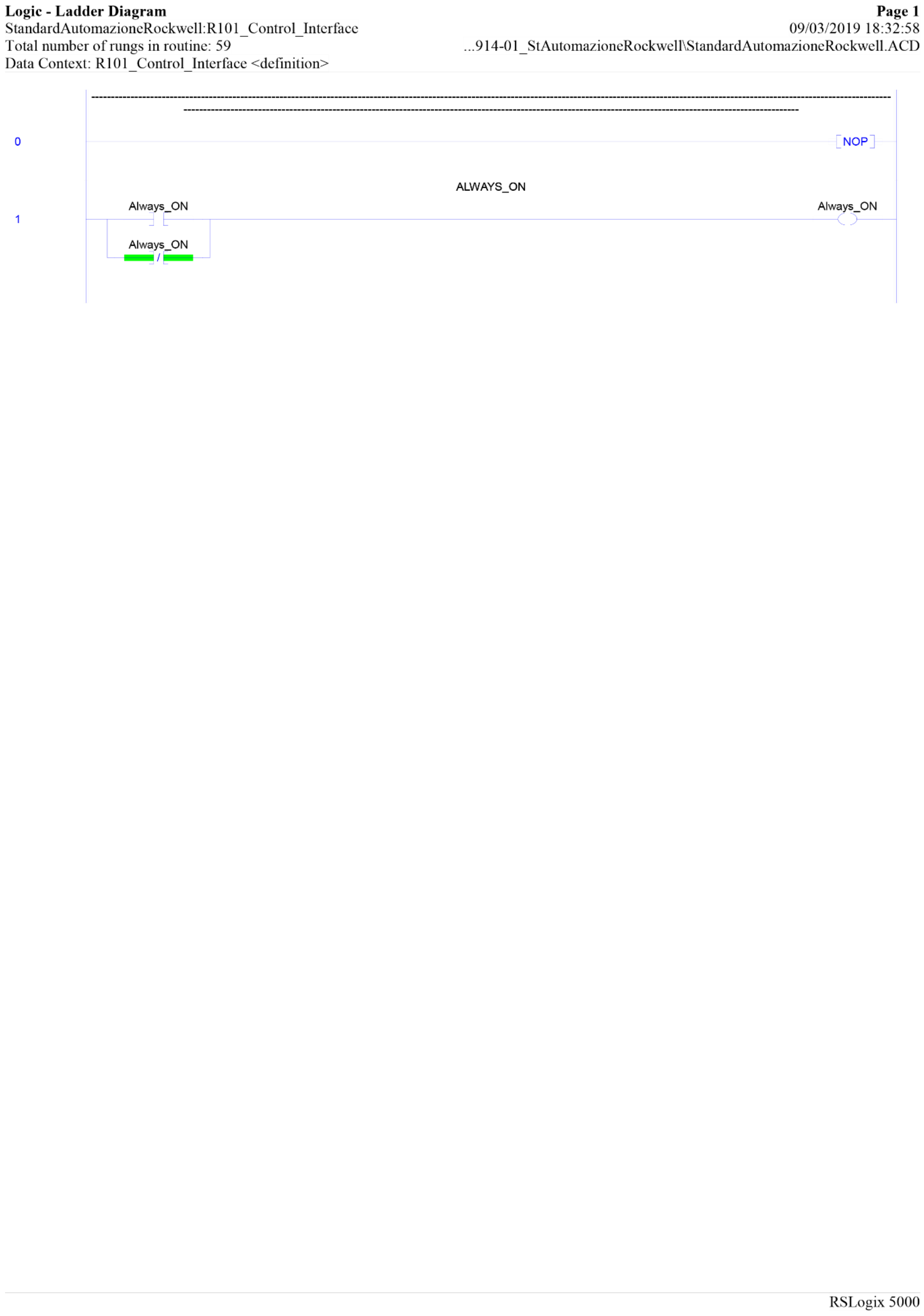
**Velocity_LU** 0.0 REAL R101_Control_Interface
Usage: Local Tag
*Velocity_LU - R101_Control_Interface/Logic - *28(MOV), 29(MOV)*

**Velocity_LU_Real** 0.0 REAL R101_Control_Interface
Usage: Local Tag
*Velocity_LU_Real - R101_Control_Interface/Logic - *24(MUL), *25(MOV), 25(GRT), 25(LES), 28(MOV)*

**Velocity_rpm** 0 DINT R101_Control_Interface
Usage: Local Tag
*Velocity_rpm - R101_Control_Interface/Logic - *28(MOV), 29(MOV)*

**Velocity_rpm_Real** 0.0 REAL R101_Control_Interface
Usage: Local Tag
*Velocity_rpm_Real - R101_Control_Interface/Logic - *23(MUL), *27(MOV), 24(DIV), 27(GRT), 27(LES), 28(MOV)*

RSLogix 5000



**Logic - Ladder Diagram** | **Page 1**

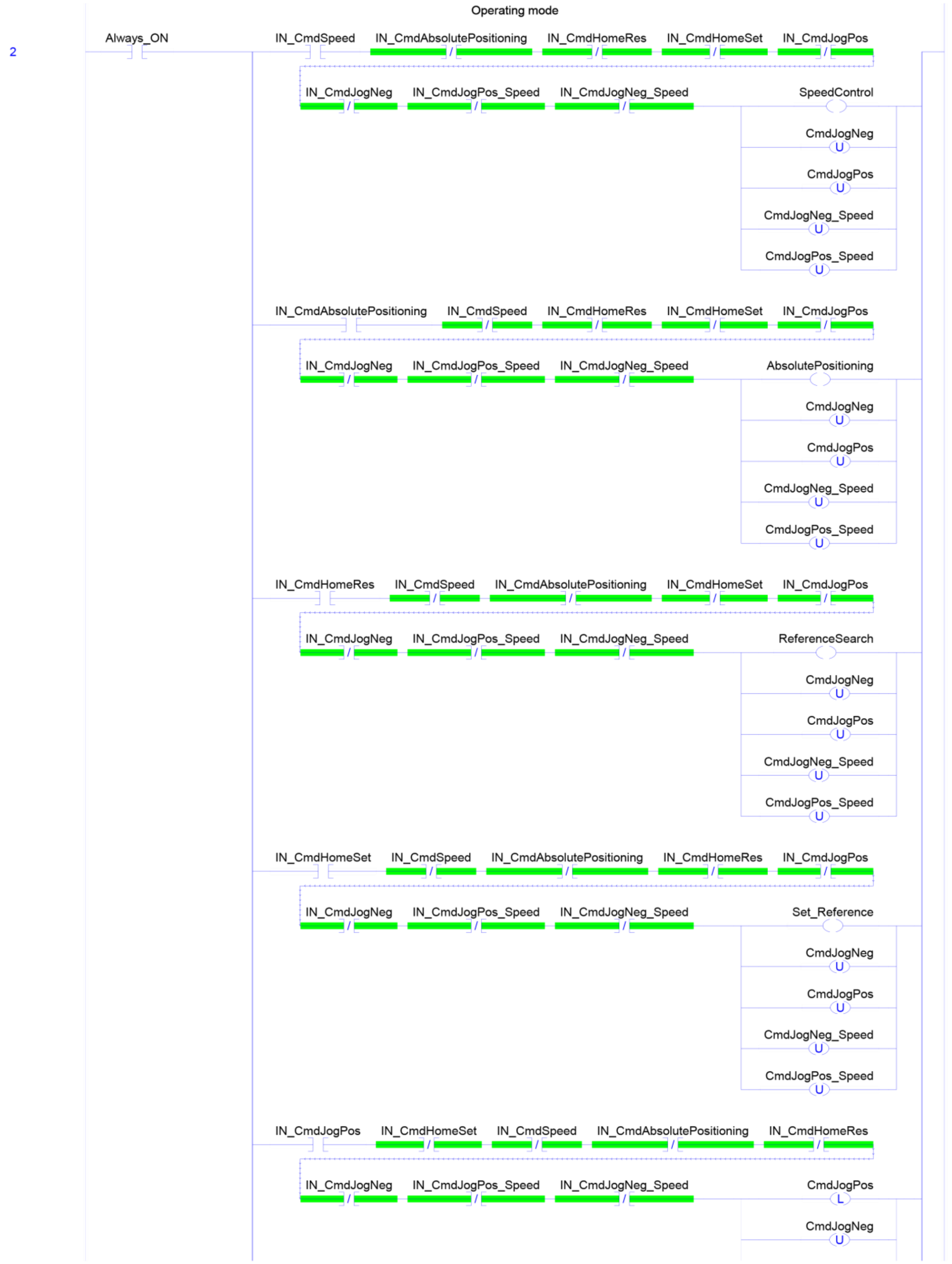
StandardAutomazioneRockwell:R101_Control_Interface | 09/03/2019 18:32:58

Total number of rungs in routine: 59 | ...914-01_StAutomazioneRockwell\StandardAutomazioneRockwell.ACD

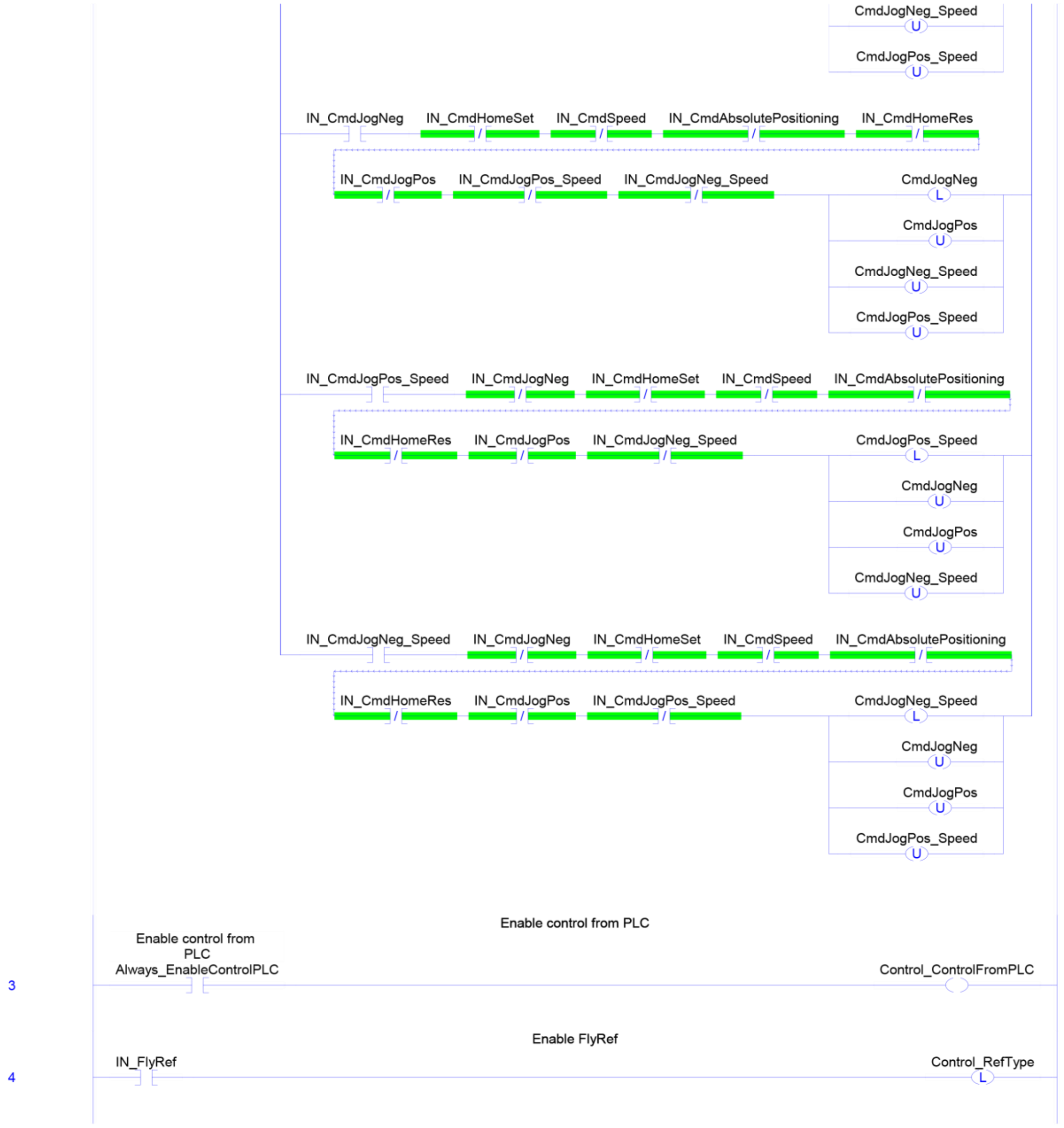
Data Context: R101_Control_Interface <definition>

RSLogix 5000

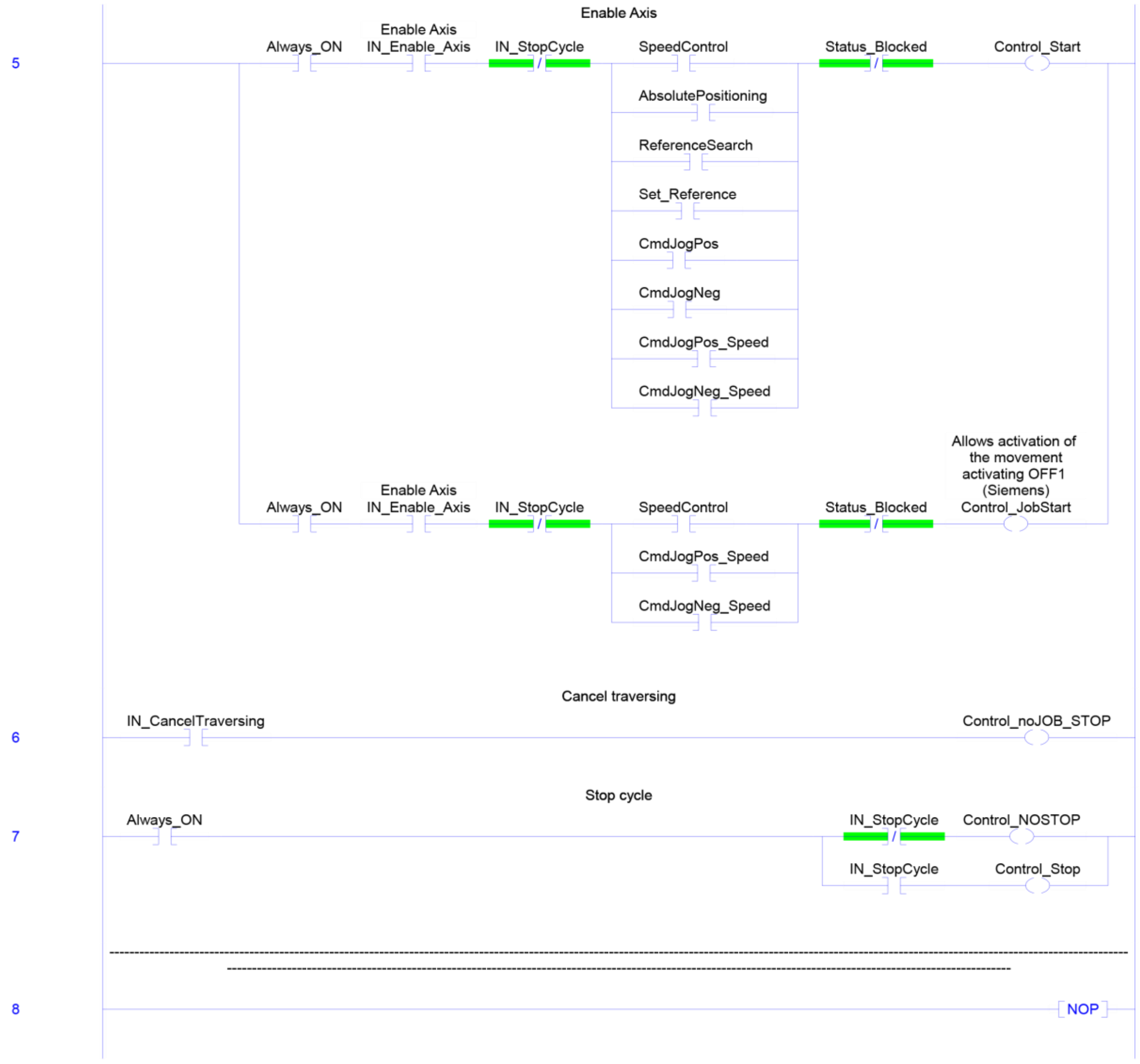
**Logic - Ladder Diagram** **Page 2**
StandardAutomazioneRockwell:R101_Control_Interface 09/03/2019 18:32:59
Total number of rungs in routine: 59 ...914-01_StAutomazioneRockwell\StandardAutomazioneRockwell.ACD
Data Context: R101_Control_Interface <definition>



**Logic - Ladder Diagram** **Page 3**
StandardAutomazioneRockwell:R101_Control_Interface 09/03/2019 18:32:59
Total number of rungs in routine: 59 ...914-01_StAutomazioneRockwell\StandardAutomazioneRockwell.ACD
Data Context: R101_Control_Interface <definition>



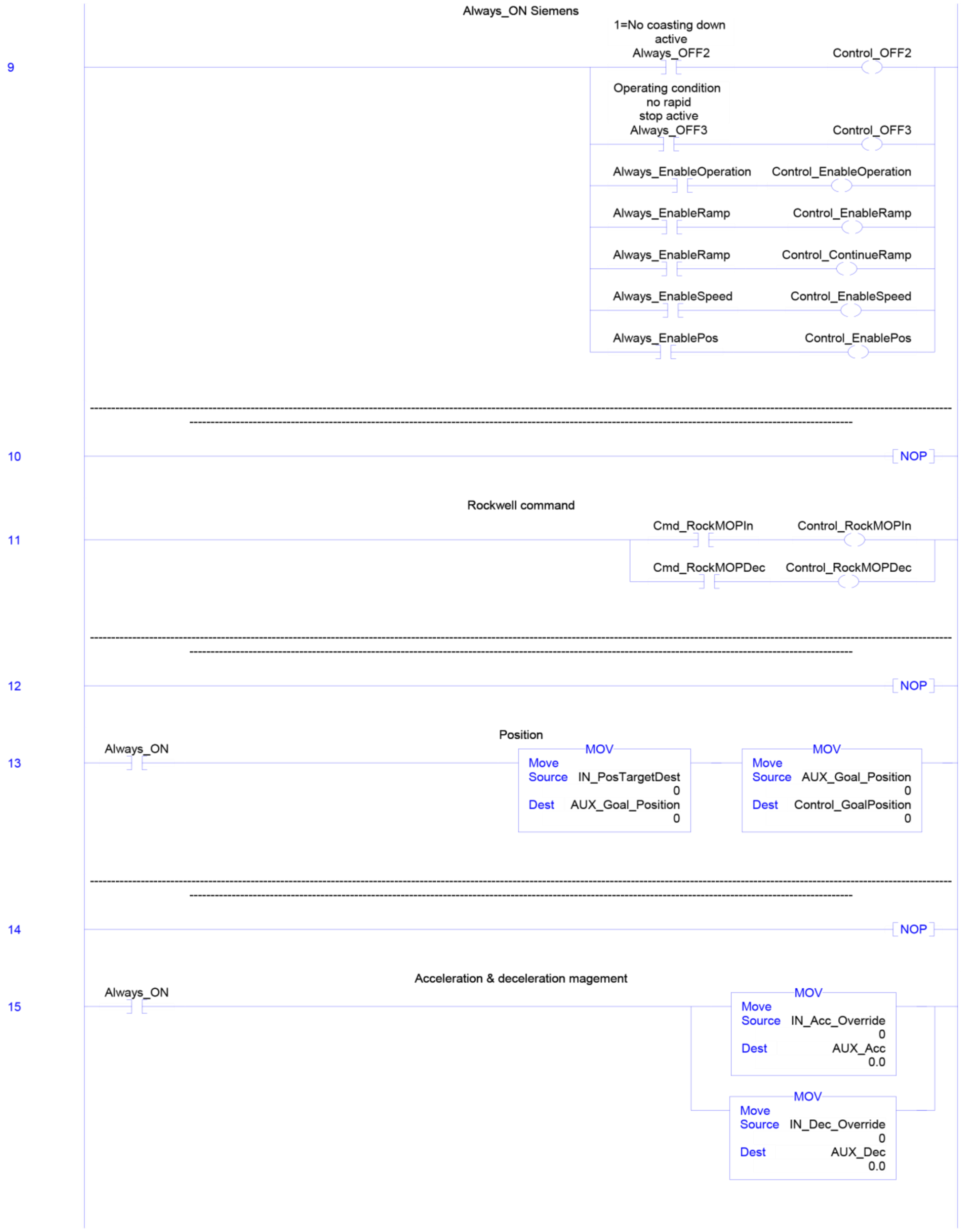
**Logic - Ladder Diagram** **Page 4**
StandardAutomazioneRockwell:R101_Control_Interface 09/03/2019 18:32:59
Total number of rungs in routine: 59 ...914-01_StAutomazioneRockwell\StandardAutomazioneRockwell.ACD
Data Context: R101_Control_Interface <definition>



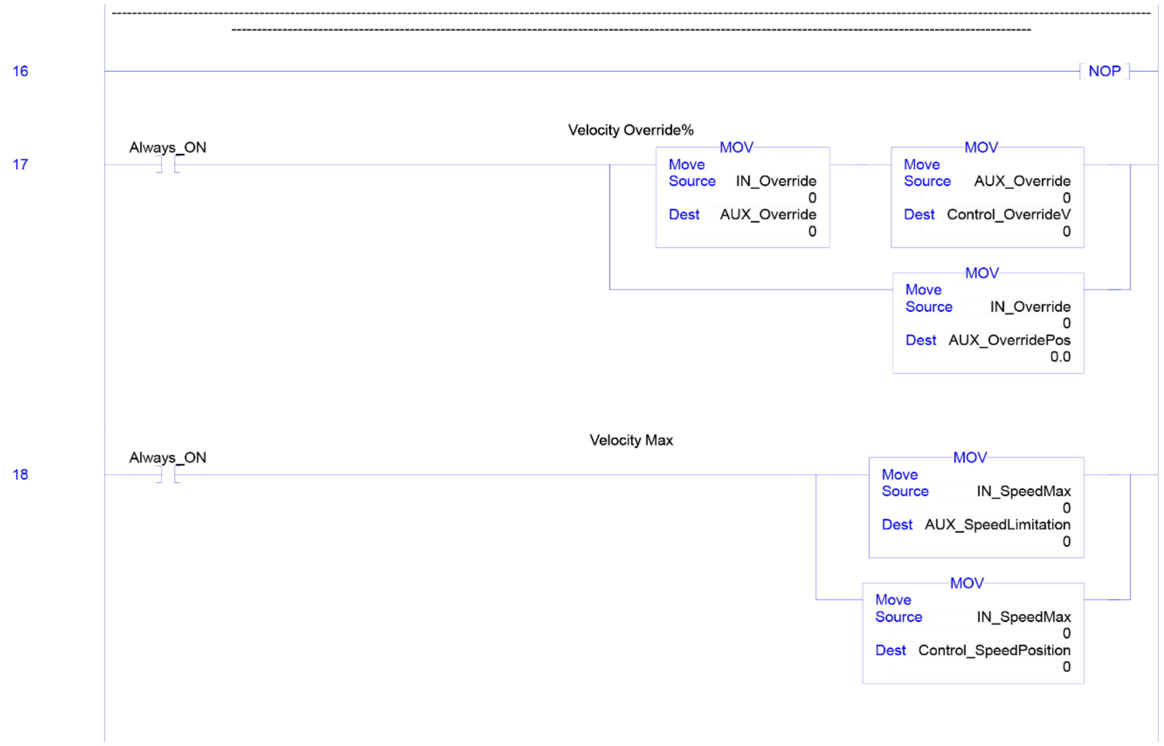
RSLogix 5000

**Logic - Ladder Diagram**
StandardAutomazioneRockwell:R101_Control_Interface
Total number of rungs in routine: 59
Data Context: R101_Control_Interface <definition>

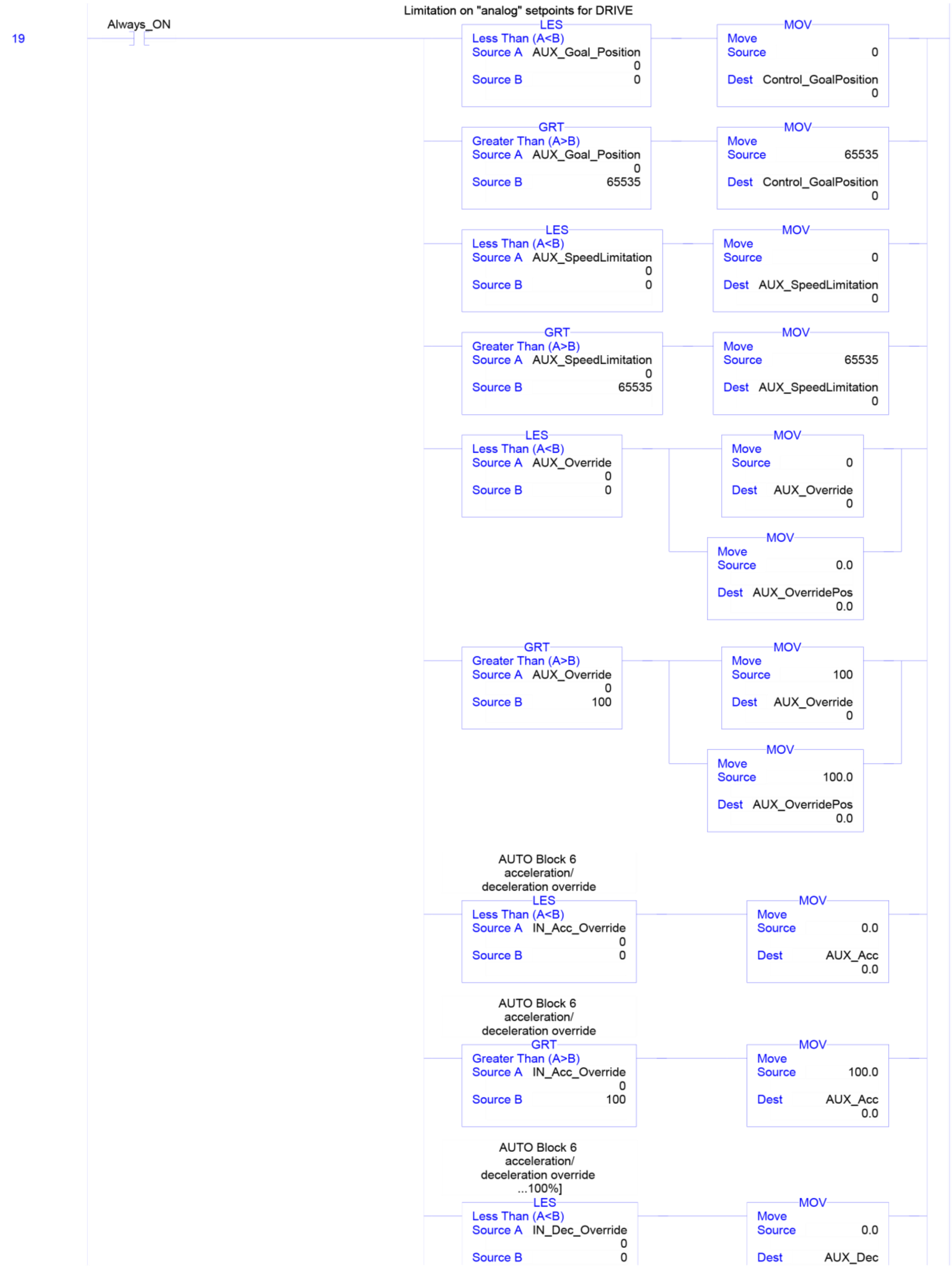
**Page 5**
09/03/2019 18:33:00
...914-01_StAutomazioneRockwell\StandardAutomazioneRockwell.ACD

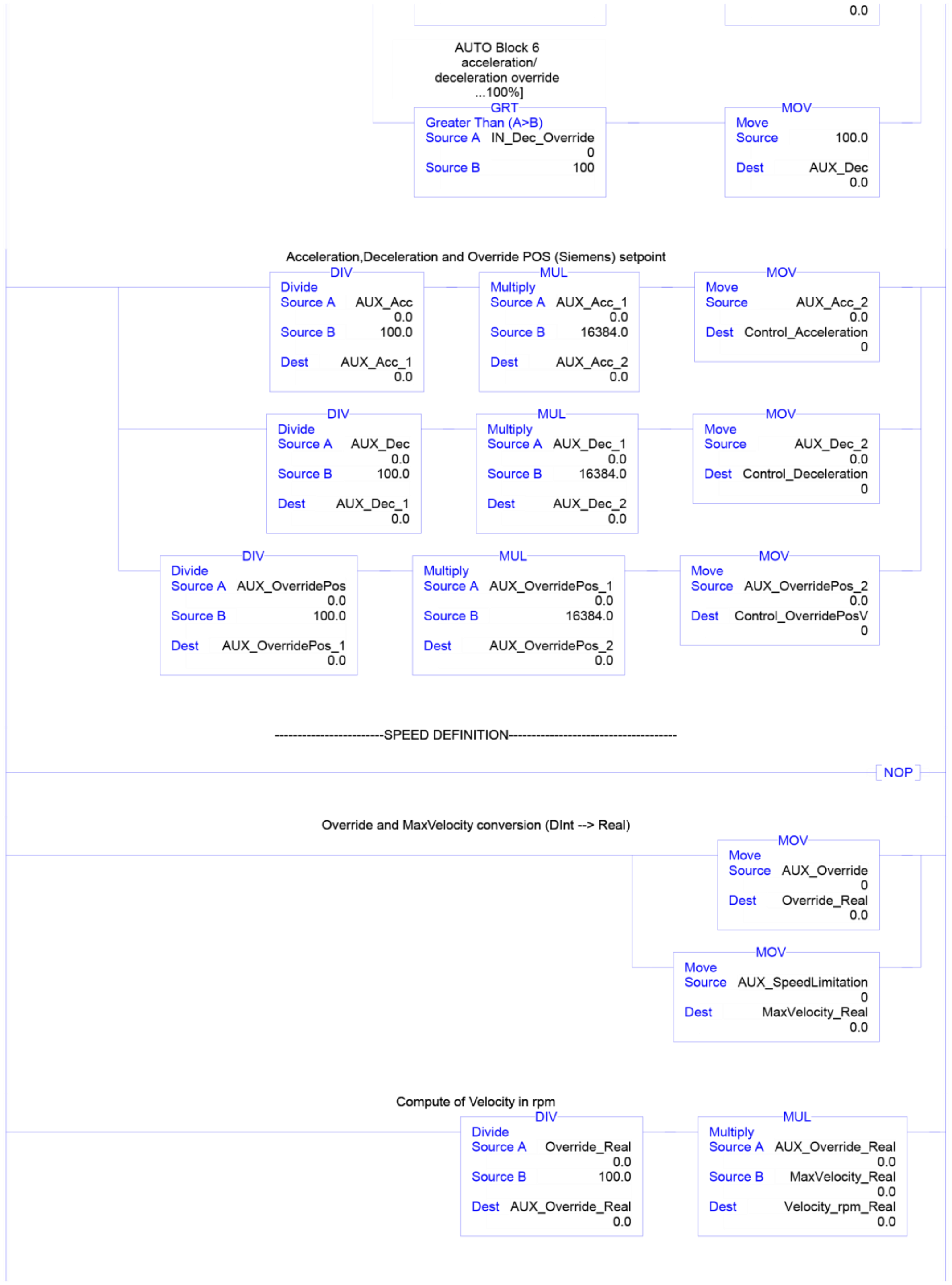


**Logic - Ladder Diagram** **Page 6**
StandardAutomazioneRockwell:R101_Control_Interface 09/03/2019 18:33:04
Total number of rungs in routine: 59 ...914-01_StAutomazioneRockwell\StandardAutomazioneRockwell.ACD
Data Context: R101_Control_Interface <definition>



**Logic - Ladder Diagram** **Page 7**
StandardAutomazioneRockwell:R101_Control_Interface 09/03/2019 18:33:05
Total number of rungs in routine: 59 ...914-01_StAutomazioneRockwell\StandardAutomazioneRockwell.ACD
Data Context: R101_Control_Interface <definition>





0.0

AUTO Block 6 acceleration/ deceleration override ...100%]

GRT
Greater Than (A>B)
Source A IN_Dec_Override
0
Source B 100

MOV
Move
Source 100.0
Dest AUX_Dec
0.0

Acceleration,Deceleration and Override POS (Siemens) setpoint

20

DIV
Divide
Source A AUX_Acc
0.0
Source B 100.0
Dest AUX_Acc_1
0.0

MUL
Multiply
Source A AUX_Acc_1
0.0
Source B 16384.0
Dest AUX_Acc_2
0.0

MOV
Move
Source AUX_Acc_2
0.0
Dest Control_Acceleration
0

DIV
Divide
Source A AUX_Dec
0.0
Source B 100.0
Dest AUX_Dec_1
0.0

MUL
Multiply
Source A AUX_Dec_1
0.0
Source B 16384.0
Dest AUX_Dec_2
0.0

MOV
Move
Source AUX_Dec_2
0.0
Dest Control_Deceleration
0

DIV
Divide
Source A AUX_OverridePos
0.0
Source B 100.0
Dest AUX_OverridePos_1
0.0

MUL
Multiply
Source A AUX_OverridePos_1
0.0
Source B 16384.0
Dest AUX_OverridePos_2
0.0

MOV
Move
Source AUX_OverridePos_2
0.0
Dest Control_OverridePosV
0

------------------------SPEED DEFINITION-------------------------------------

21

NOP

Override and MaxVelocity conversion (DInt --> Real)

22

MOV
Move
Source AUX_Override
0
Dest Override_Real
0.0

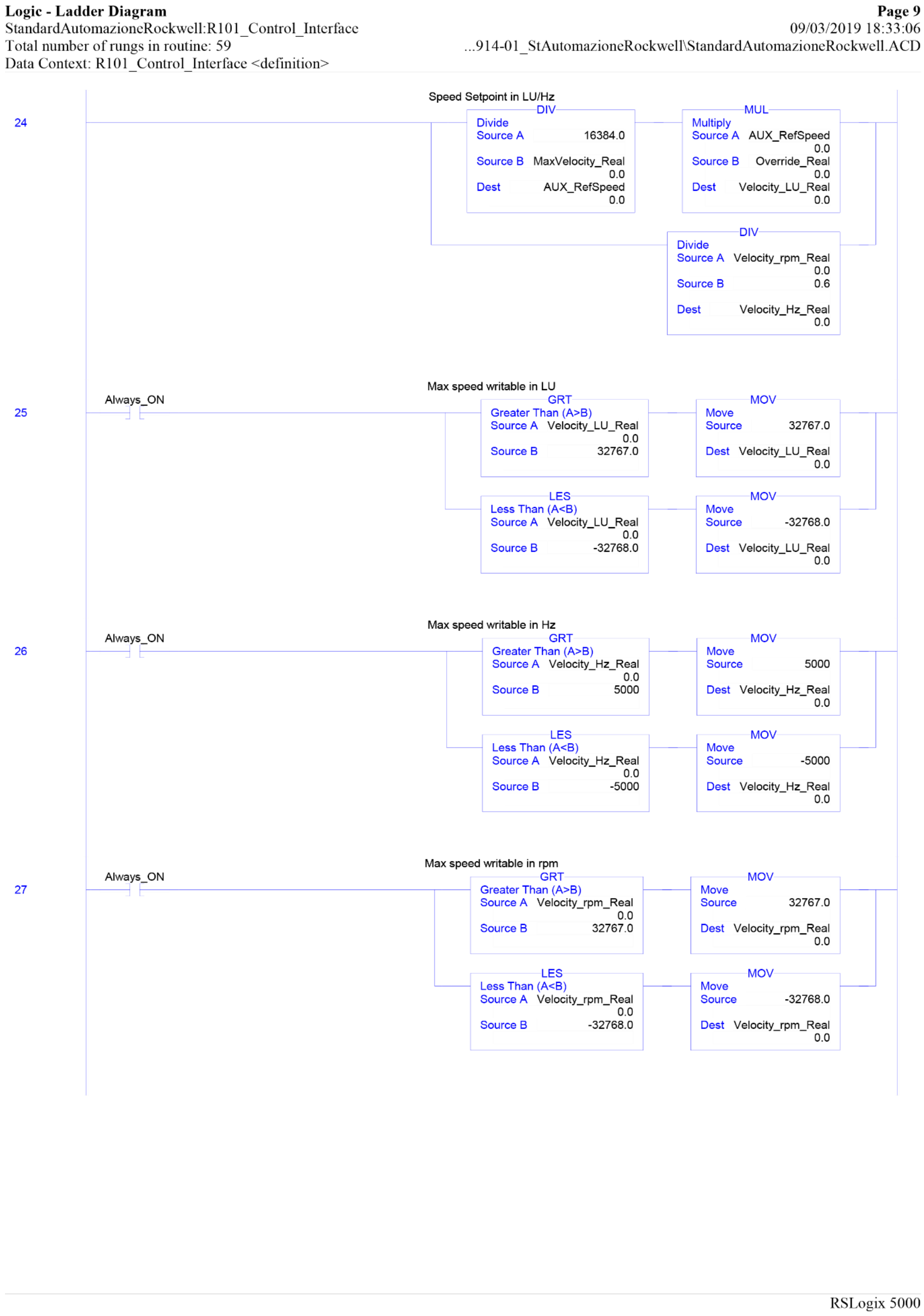
MOV
Move
Source AUX_SpeedLimitation
0
Dest MaxVelocity_Real
0.0

Compute of Velocity in rpm

23

DIV
Divide
Source A Override_Real
0.0
Source B 100.0
Dest AUX_Override_Real
0.0

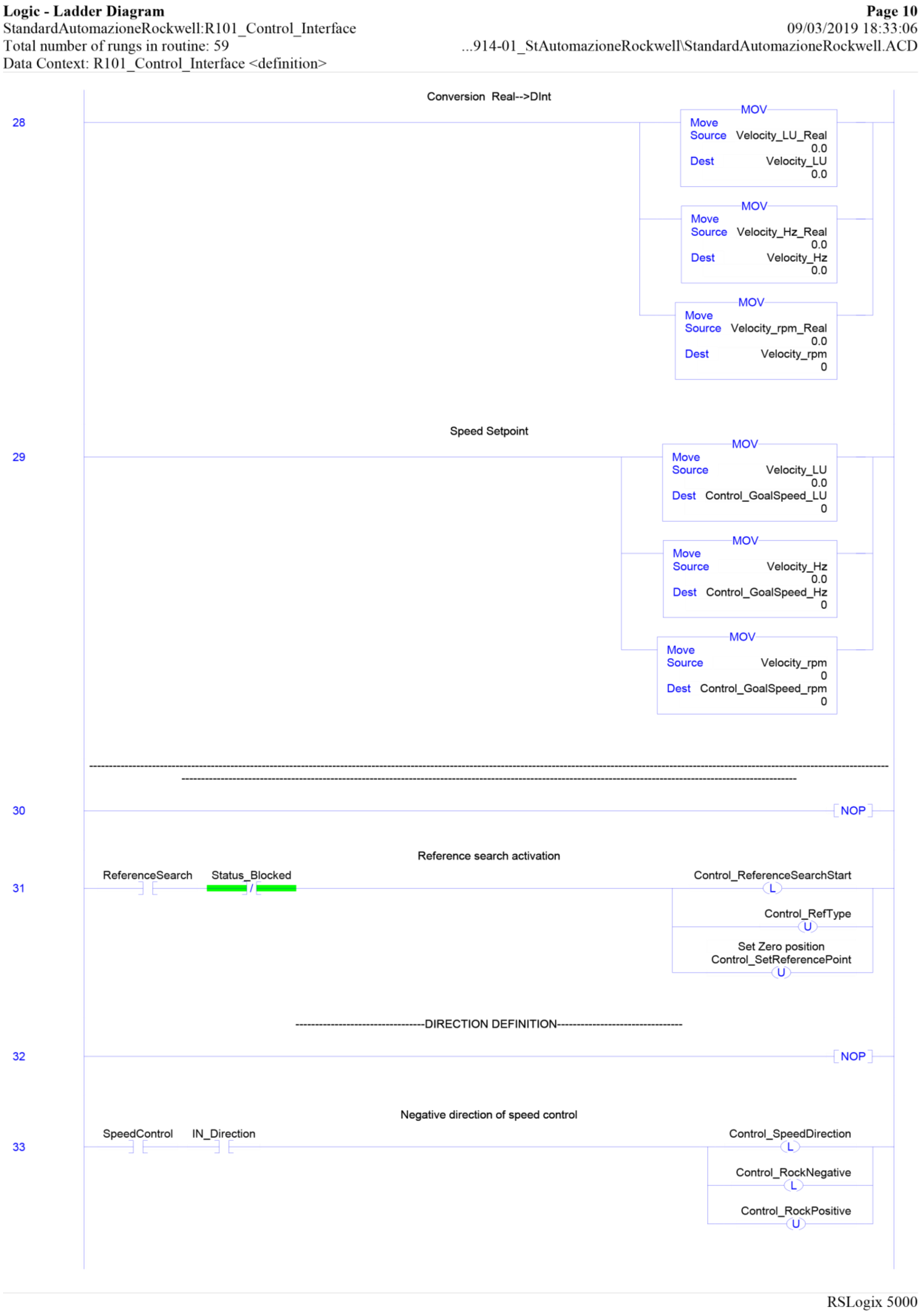
MUL
Multiply
Source A AUX_Override_Real
0.0
Source B MaxVelocity_Real
0.0
Dest Velocity_rpm_Real
0.0



**Logic - Ladder Diagram** **Page 9**
StandardAutomazioneRockwell:R101_Control_Interface 09/03/2019 18:33:06
Total number of rungs in routine: 59 ...914-01_StAutomazioneRockwell\StandardAutomazioneRockwell.ACD
Data Context: R101_Control_Interface <definition>

```
24    Speed Setpoint in LU/Hz
      DIV                          MUL
      Divide                       Multiply
      Source A        16384.0      Source A   AUX_RefSpeed
                                                       0.0
      Source B  MaxVelocity_Real   Source B    Override_Real
                          0.0                          0.0
      Dest          AUX_RefSpeed   Dest     Velocity_LU_Real
                          0.0                          0.0
                                   DIV
                                   Divide
                                   Source A  Velocity_rpm_Real
                                                       0.0
                                   Source B            0.6
                                   Dest      Velocity_Hz_Real
                                                       0.0

25    Always_ON
      Max speed writable in LU
      GRT                            MOV
      Greater Than (A>B)             Move
      Source A  Velocity_LU_Real     Source        32767.0
                           0.0
      Source B         32767.0       Dest  Velocity_LU_Real
                                                       0.0
      LES                            MOV
      Less Than (A<B)                Move
      Source A  Velocity_LU_Real     Source       -32768.0
                           0.0
      Source B        -32768.0       Dest  Velocity_LU_Real
                                                       0.0

26    Always_ON
      Max speed writable in Hz
      GRT                            MOV
      Greater Than (A>B)             Move
      Source A  Velocity_Hz_Real     Source           5000
                           0.0
      Source B            5000       Dest  Velocity_Hz_Real
                                                       0.0
      LES                            MOV
      Less Than (A<B)                Move
      Source A  Velocity_Hz_Real     Source          -5000
                           0.0
      Source B           -5000       Dest  Velocity_Hz_Real
                                                       0.0

27    Always_ON
      Max speed writable in rpm
      GRT                            MOV
      Greater Than (A>B)             Move
      Source A  Velocity_rpm_Real    Source        32767.0
                           0.0
      Source B         32767.0       Dest  Velocity_rpm_Real
                                                       0.0
      LES                            MOV
      Less Than (A<B)                Move
      Source A  Velocity_rpm_Real    Source       -32768.0
                           0.0
      Source B        -32768.0       Dest  Velocity_rpm_Real
                                                       0.0
```

RSLogix 5000



**Logic - Ladder Diagram** **Page 10**
StandardAutomazioneRockwell:R101_Control_Interface 09/03/2019 18:33:06
Total number of rungs in routine: 59 ...914-01_StAutomazioneRockwell\StandardAutomazioneRockwell.ACD
Data Context: R101_Control_Interface <definition>

Conversion Real-->DInt

28
MOV Move Source Velocity_LU_Real 0.0 Dest Velocity_LU 0.0
MOV Move Source Velocity_Hz_Real 0.0 Dest Velocity_Hz 0.0
MOV Move Source Velocity_rpm_Real 0.0 Dest Velocity_rpm 0

Speed Setpoint

29
MOV Move Source Velocity_LU 0.0 Dest Control_GoalSpeed_LU 0
MOV Move Source Velocity_Hz 0.0 Dest Control_GoalSpeed_Hz 0
MOV Move Source Velocity_rpm 0 Dest Control_GoalSpeed_rpm 0

--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------------------------------------------

30 NOP

Reference search activation

31 ReferenceSearch Status_Blocked
Control_ReferenceSearchStart L
Control_RefType U
Set Zero position
Control_SetReferencePoint U

--------------------------------DIRECTION DEFINITION--------------------------------

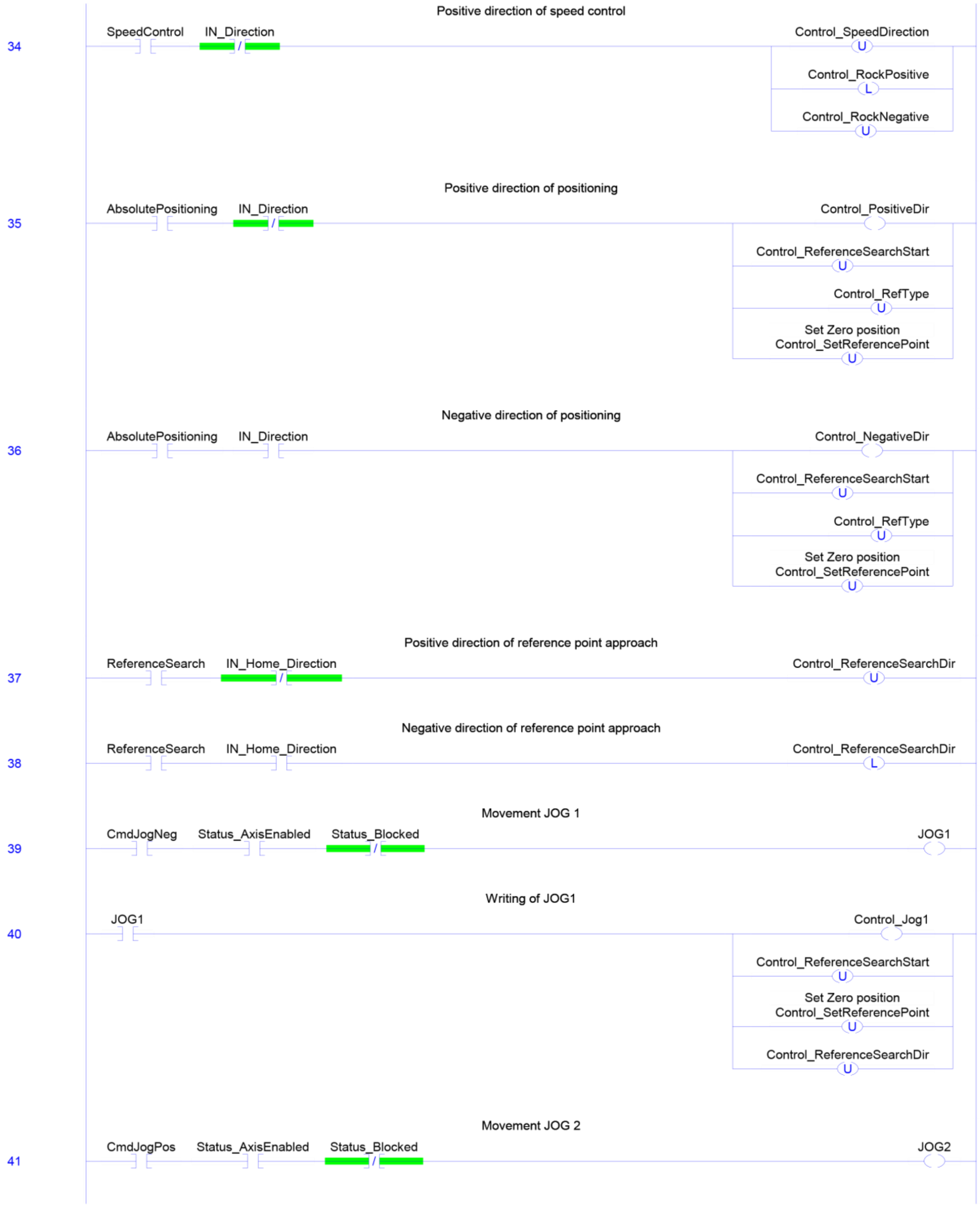
32 NOP

Negative direction of speed control

33 SpeedControl IN_Direction
Control_SpeedDirection L
Control_RockNegative L
Control_RockPositive U

RSLogix 5000

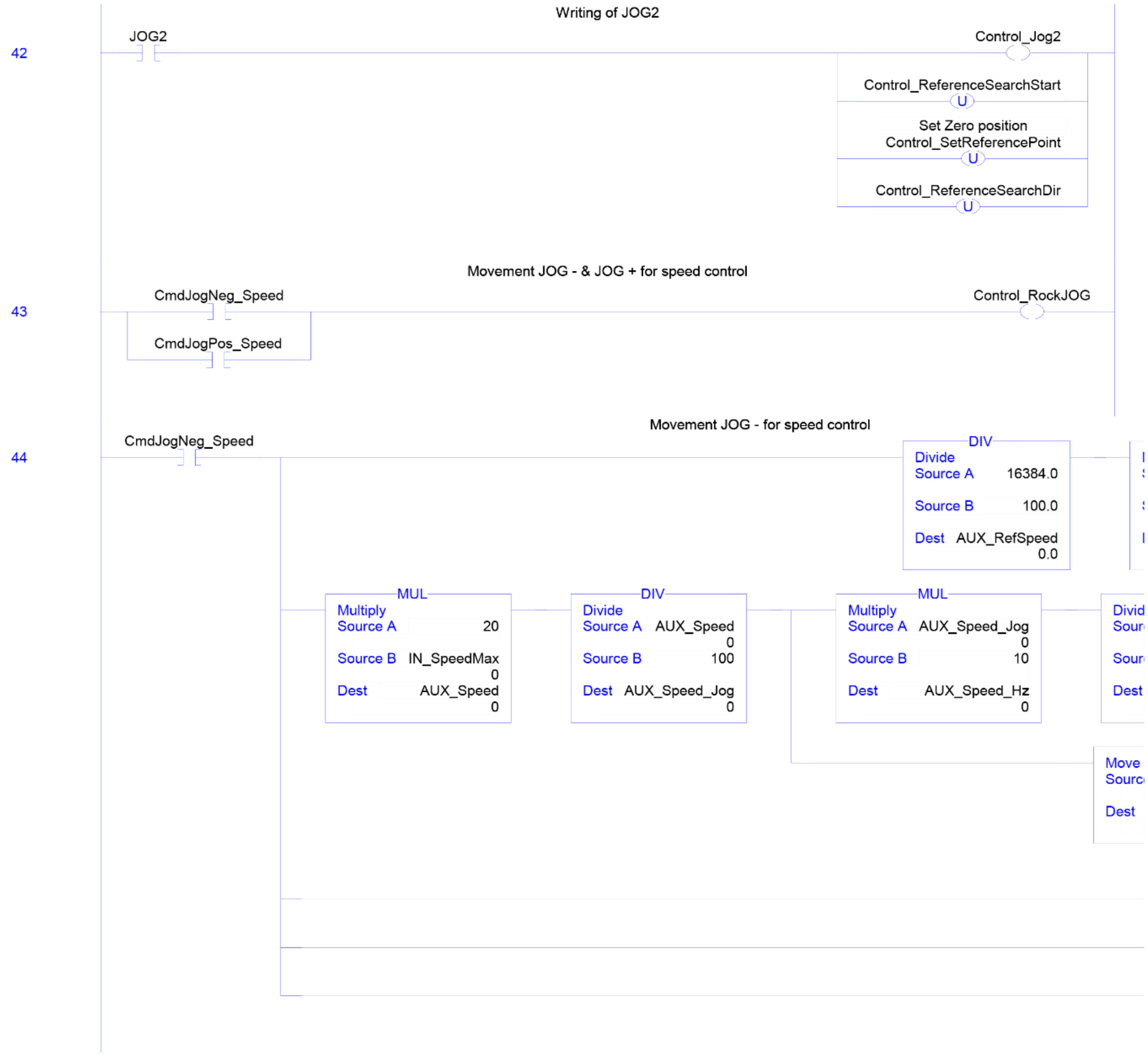
**Logic - Ladder Diagram** **Page 11**
StandardAutomazioneRockwell:R101_Control_Interface 09/03/2019 18:33:09
Total number of rungs in routine: 59 ...914-01_StAutomazioneRockwell\StandardAutomazioneRockwell.ACD
Data Context: R101_Control_Interface <definition>

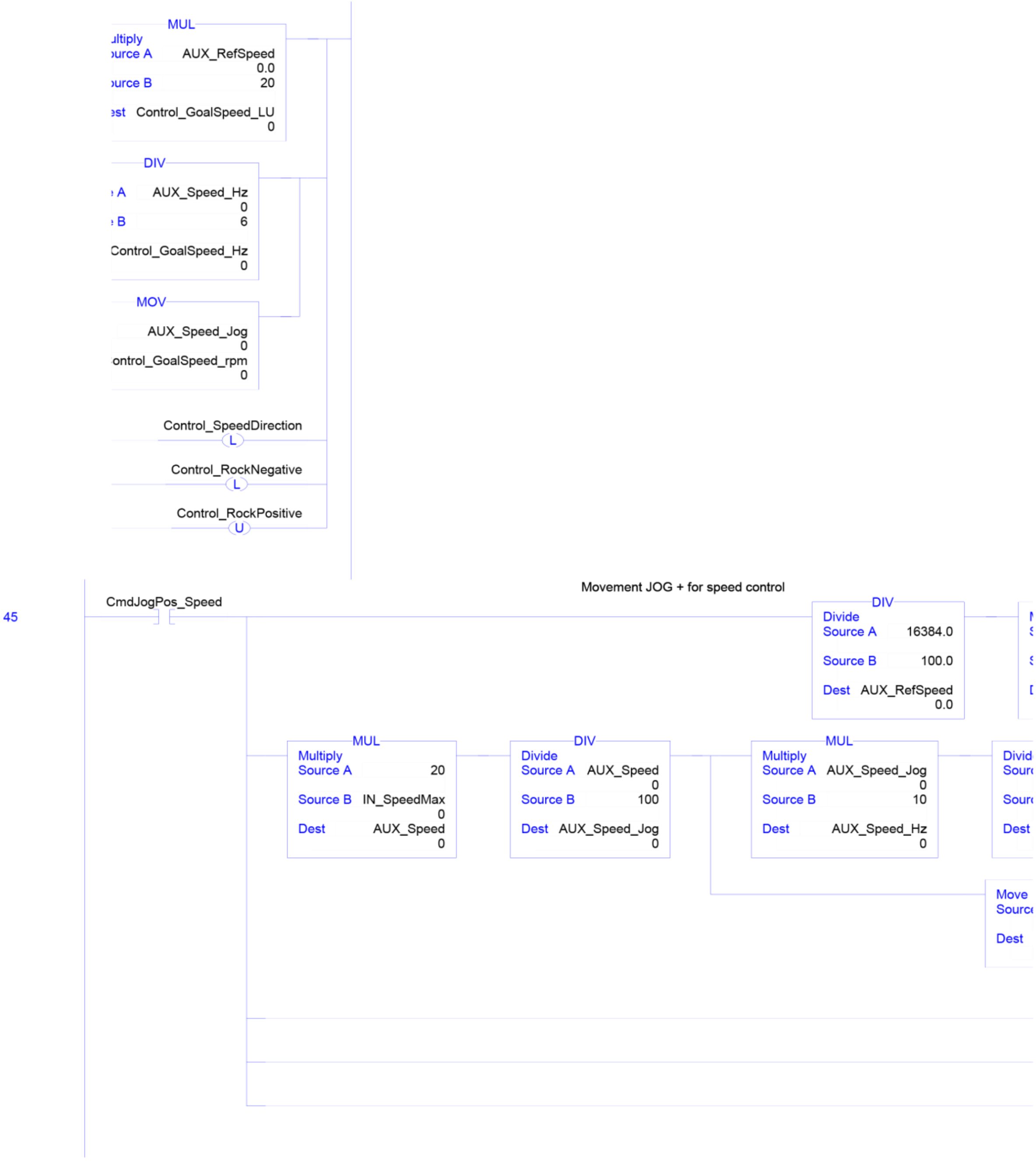


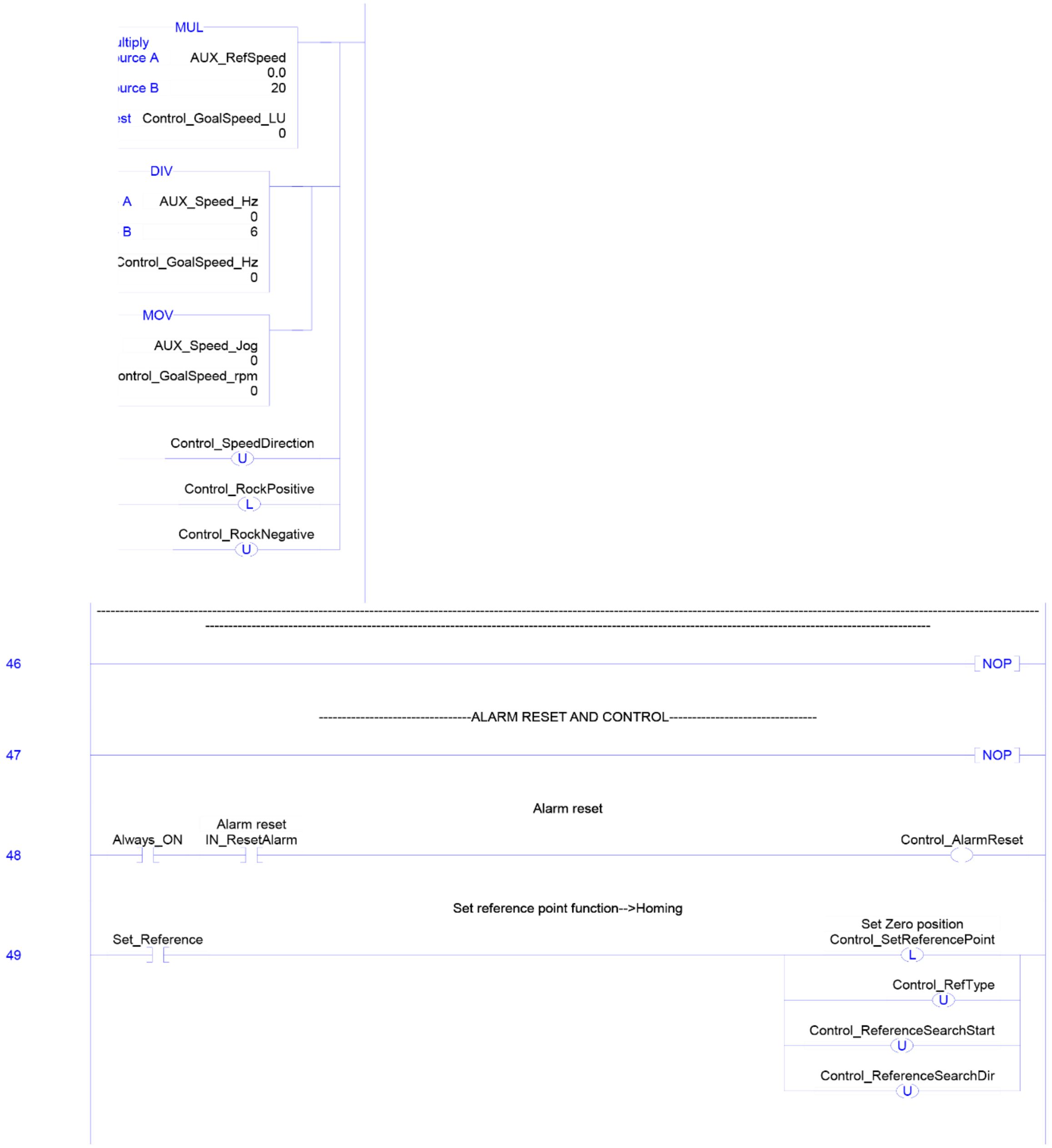
RSLogix 5000

**Logic - Ladder Diagram**
StandardAutomazioneRockwell:R101_Control_Interface
Total number of rungs in routine: 59
Data Context: R101_Control_Interface <definition>

**Page 12**
09/03/2019 18:33:09
...914-01_StAutomazioneRockwell\StandardAutomazioneRockwell.ACD




MUL
ıltiply
ɔurce A AUX_RefSpeed
0.0
ɔurce B 20
əst Control_GoalSpeed_LU
0
DIV
: A AUX_Speed_Hz
0
: B 6
Control_GoalSpeed_Hz
0
MOV
AUX_Speed_Jog
0
ontrol_GoalSpeed_rpm
0
Control_SpeedDirection
L
Control_RockNegative
L
Control_RockPositive
U
45
Movement JOG + for speed control
CmdJogPos_Speed
DIV
Divide
Source A 16384.0
Source B 100.0
Dest AUX_RefSpeed
0.0
MUL
Multiply
Source A 20
Source B IN_SpeedMax
0
Dest AUX_Speed
0
DIV
Divide
Source A AUX_Speed
0
Source B 100
Dest AUX_Speed_Jog
0
MUL
Multiply
Source A AUX_Speed_Jog
0
Source B 10
Dest AUX_Speed_Hz
0
Divid
Sourc
Sourc
Dest
Move
Sourc
Dest



------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
-------------------------------------------------------------------------------------------------------------------------------------------------------

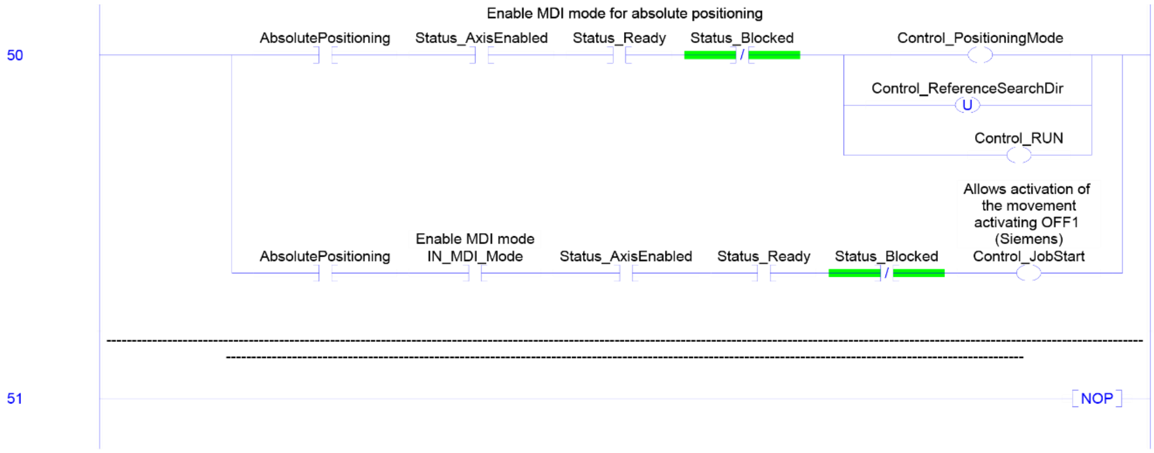
---------------------------------ALARM RESET AND CONTROL--------------------------------

Alarm reset

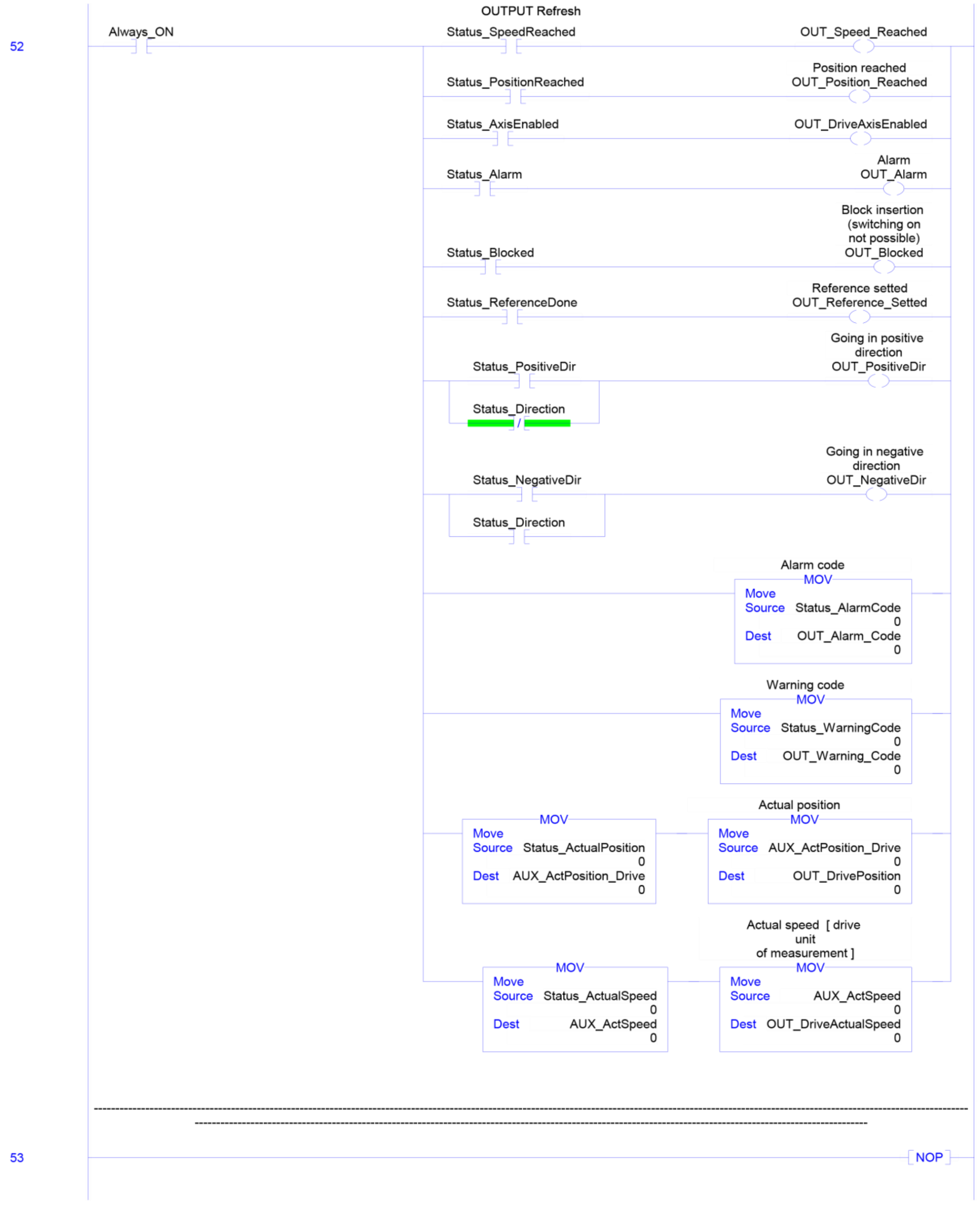
Set reference point function-->Homing

**Logic - Ladder Diagram**
StandardAutomazioneRockwell:R101_Control_Interface
Total number of rungs in routine: 59
Data Context: R101_Control_Interface <definition>

**Page 15**
09/03/2019 18:33:12
...914-01_StAutomazioneRockwell\StandardAutomazioneRockwell.ACD



**Logic - Ladder Diagram** **Page 16**
StandardAutomazioneRockwell:R101_Control_Interface 09/03/2019 18:33:13
Total number of rungs in routine: 59 ...914-01_StAutomazioneRockwell\StandardAutomazioneRockwell.ACD
Data Context: R101_Control_Interface <definition>

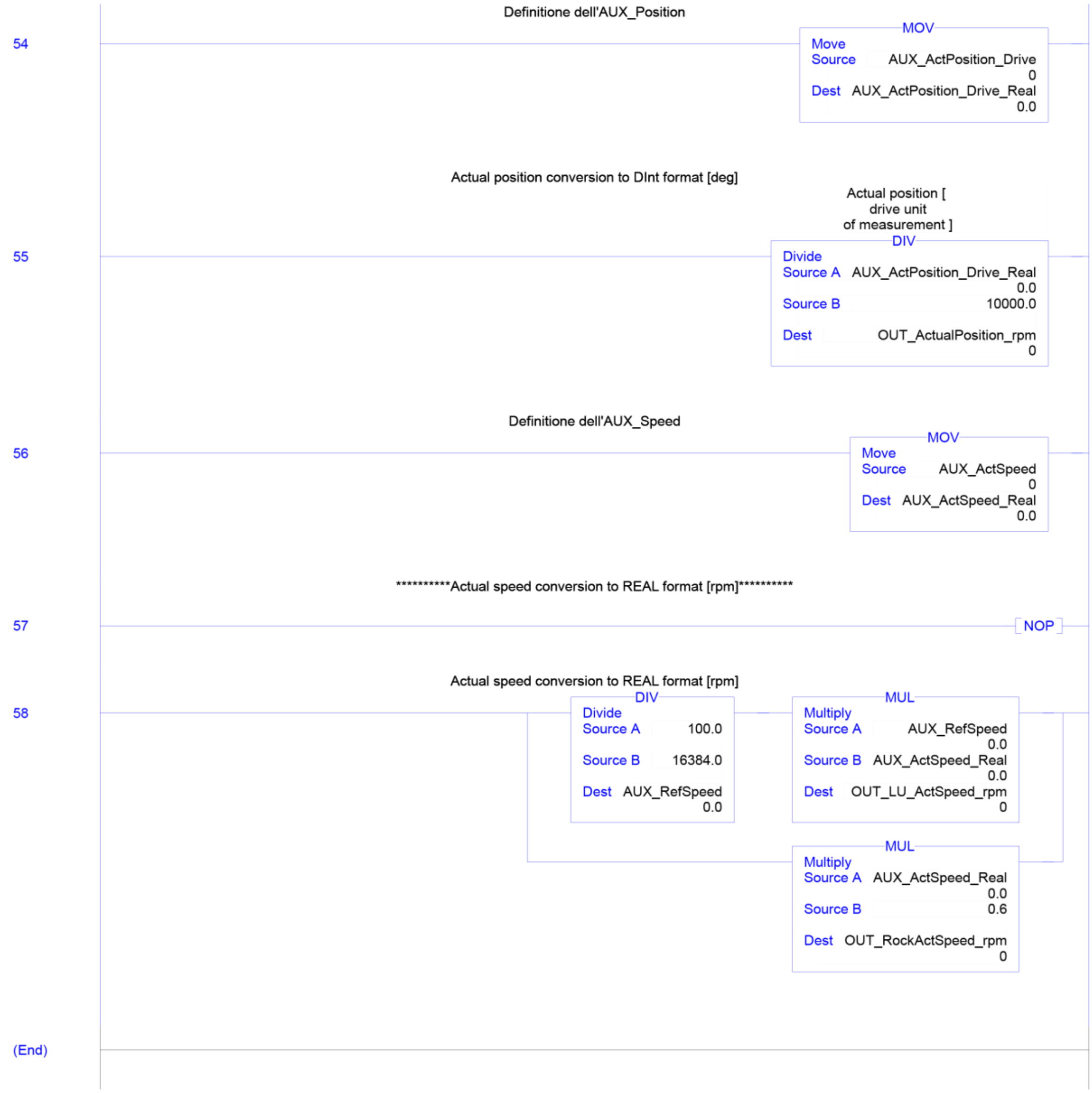


**Logic - Ladder Diagram** **Page 17**

StandardAutomazioneRockwell:R101_Control_Interface 09/03/2019 18:33:14

Total number of rungs in routine: 59 ...914-01_StAutomazioneRockwell\StandardAutomazioneRockwell.ACD

Data Context: R101_Control_Interface <definition>



54

Definitione dell'AUX_Position

MOV
Move
Source AUX_ActPosition_Drive
0
Dest AUX_ActPosition_Drive_Real
0.0

55

Actual position conversion to DInt format [deg]

Actual position [
drive unit
of measurement ]

DIV
Divide
Source A AUX_ActPosition_Drive_Real
0.0
Source B 10000.0
Dest OUT_ActualPosition_rpm
0

56

Definitione dell'AUX_Speed

MOV
Move
Source AUX_ActSpeed
0
Dest AUX_ActSpeed_Real
0.0

57

**********Actual speed conversion to REAL format [rpm]**********

NOP

58

Actual speed conversion to REAL format [rpm]

DIV
Divide
Source A 100.0
Source B 16384.0
Dest AUX_RefSpeed
0.0

MUL
Multiply
Source A AUX_RefSpeed
0.0
Source B AUX_ActSpeed_Real
0.0
Dest OUT_LU_ActSpeed_rpm
0

MUL
Multiply
Source A AUX_ActSpeed_Real
0.0
Source B 0.6
Dest OUT_RockActSpeed_rpm
0

(End)

RSLogix 5000

## Bibliography.

1. Tudisco, Giuseppe. “Generalità sui PLC”. [PDF file]. http://www.iisgalileiartiglio.gov.it/appunti-lezioni/elettronica/Dispensa_PLC.pdf
2. International Electrotechnical Commission (2003). “International Standard”. [PDF file]. https://webstore.iec.ch/preview/info_iec61131-1%7Bed2.0%7Den.pdf
3. Mauri, Marco (2001). “Introduzione alla norma IEC 61131-3”. [PDF file]. http://docenti.etec.polimi.it/IND32/Didattica/AzionamentixAutomazione/files/Introduzione%20alla%20norma%20IEC%201131-3.pdf
4. Holtkamp, Bernhard and Iyer, Anandi (2017). “Industry 4.0. The Future of Indo-German Industrial Collaboration”. [PDF file]. https://www.bertelsmann-stiftung.de/fileadmin/files/user_upload/BSt_Industrie4.0_STUDIE_web.pdf
5. Dal Prà, Marco (2005). “Manuale di programmazione dei PLC”. [PDF file]. http://www.plcforum.info/didattica/dalpra/Manuale_PLC_Software.pdf
6. Dr. D. J. Jackson. “Programmable Logic Controllers”. [PDF file]. http://jjackson.eng.ua.edu/courses/ece485/lectures/LECT03.pdf
7. Weerts, Mariëlle. “Function Block Diagrams for Programmable Logic Controllers”. [PDF file]. https://pdfs.semanticscholar.org/8f32/0bb0f7a902ca2a5a552e0562f4c1fe340b1e.pdf
8. Siemens (2012). “TIA Portal V12”. [PDF file]. https://w5.siemens.com/italy/web/AD/ProdottieSoluzioni/Sistemiautomazionenew/Eventi/Documents/presentazioni%20simatic%20live/1500_Programmazione.pdf?istablet=true,_blank
9. Siemens (2008). “High Level Language Programming with S7-SCL”. [PDF file]. https://w3.siemens.com/mcms/sce/en/advanced_training/training_material/classic-modules/tabcardpages/Documents/programming-languages/c02_s7scl_en.pdf

10. Capuzzimati, Mario. "Operazioni aritmatiche". [PDF file]. http://www.cumacini.altervista.org/Sistemi/PLC_Operazioni_aritmetiche.pdf
11. Siemens (2010). "Automatizzare e trarre immediato profitto con lo standard leader Industrial Ethernet.". [PDF file]. https://w5.siemens.com/italy/web/AD/ProdottieSoluzioni/Sistemiautomazionenew/homepageProfinet/Documents/Brochure_PROFINET.pdf
12. Siemens (2011). "SINAMICS G120. Frequency inverters with Control Units CU230P-2 HVAC, CU230P-2 DP, CU230P-2 CAN". [PDF file]. https://w5.siemens.com/web/cz/cz/corporate/portal/home/produkty_a_sluzby/IBT/mereni_a_regulace/frekvencni_menice/Documents/G120P-0.37_35A_Operating_Instructions_en.pdf
13. Siemens (2011). "Structured Control Language per Step7 V11". [PDF file]. https://w5.siemens.com/italy/web/AD/ProdottieSoluzioni/Sistemiautomazionenew/Eventi/Documents/presentazioni%20simatic%20live/TIA_PORTAL-SCL.pdf?ismobile=true
14. Siemens (2016). "Function blocks to control the SINAMICS with SIMATIC S7 in TIA-Portal". [PDF file]. https://support.industry.siemens.com/cs/document/109475044/sinamics-blocks-drivelib-for-the-control-in-the-tia-portal?dti=0&lc=en-WW
15. Siemens (2017). "Configurazione dell'hardware e progettazione di collegamenti STEP 7". [PDF file]. https://support.industry.siemens.com/cs/document/109751824/simatic-configurazione-dell'hardware-e-progettazione-di-collegamenti-step-7?dti=0&lc=it-IT
16. Siemens (2010). "Ladder Logic (LAD) for S7-300 and S7-400 Programming". [PDF file]. https://cache.industry.siemens.com/dl/files/822/45523822/att_82001/v1/s7kop__b.pdf
17. Rockwell Automation (2017). "Convertitore di frequenza PowerFlex serie 520. Guida di messa in funzione rapida". [PDF file]. https://literature.rockwellautomation.com/idc/groups/literature/documents/qs/520-qs001_-it-e.pdf

18. Porcaro, Giuseppe e Tattoli, Maria. “I controllori a logica programmabile”. [PDF file].

## Sitography.

1. https://ec.europa.eu/eip/ageing/standards/ict-and-communication/user-interface/iec-61131_en
2. http://www.learnabout-electronics.org/Digital/dig16.php
3. http://www.plccenter.cn/Siemens_Step7/bas00086.htm
4. https://www.plcacademy.com/ladder-logic-tutorial/
5. https://www.quora.com/What-is-the-most-obsolete-high-level-programming-language
6. https://www.electroyou.it/gilberto/wiki/il-linguaggio-scl-nella-programmazione-dei-plc
7. https://automationforum.in/t/counters-in-plc-ladder-diaghram/2745
8. https://www.courses.psu.edu/e_met/e_met430_jar14/prgflo/jmp.html
9. http://automation-bd.blogspot.com/2012/07/conditional-jump.html
10. https://w5.siemens.com/italy/web/AD/ProdottieSoluzioni/Sistemiautomazionenew/homepageProfinet/X/Pages/ProfinetIORealTime.aspx
11. https://w5.siemens.com/italy/web/AD/ProdottieSoluzioni/Sistemiautomazionenew/homepageProfinet/X/Pages/ProfinetIRT.aspx
12. https://w5.siemens.com/italy/web/AD/ProdottieSoluzioni/Sistemiautomazionenew/homepageProfinet/X/Pages/ProfinetCBA.aspx
13. https://www.google.it/search?q=funzionamento+base+dei+plc&source=lnms&tbm=isch&sa=X&ved=0ahUKEwj3x_Gite_eAhWisKQKHb6MC7UQ_AUIDigB&biw=1536&bih=734#imgrc=UfeAw0Zt851lxM:
14. https://www.google.com/search?q=bus+system&source=lnms&tbm=isch&sa=X&ved=0ahUKEwjuvuzgh5bfAhXBp4sKHav5AnQQ_AUIDigB&biw=1536&bih=734#imgrc=o9Bmp3KwQNf2EM:
15. https://www.google.com/search?biw=1536&bih=734&tbm=isch&sa=1&ei=CskOXOvZJMWckwX_sIXoCw&q=pannello+operatore+per+plc&oq=pannello+operatore+per+plc&gs_l=img.3...28129.32697..32944...0.0..1.199.2342.22j4......1....1..gws-wiz-img.......0j0i67j0i8i30j0i24j0i30.FYZ2iJbIEfk#imgrc=6OxoVzQOKQicvM:
16. https://www.ecosia.org/images?q=onda+analogica#id=85E341536A74286B731277203F78A6EBC86FEE54
17. http://laboratorioscolastico.altervista.org/it_IT/il-releelettromagnetico-principio-di-funzionamento/
18. http://www.treccani.it/enciclopedia/transistor/

19. https://www.google.com/search?q=transistor&source=lnms&tbm=isch&sa=X&ved=0ahUKEwiajqbSr6XfAhWQjqQKHd8NApcQ_AUIDigB&biw=1536&bih=734#imgrc=gaLwKmYnRGdSyM:
20. https://www.google.com/search?biw=1536&bih=734&tbm=isch&sa=1&ei=k8YYXOuFD8m2kwXAq4igCw&q=industry+4.0+structure&oq=industry+4.0+structure&gs_l=img.3...35106.35440..35711...0.0..0.160.307.0j2......1....1..gws-wiz-img.dvLuk1kNheI#imgrc=heENoVA6cW-oFM:
21. https://www.ecosia.org/images?q=industry+3.0+#id=49F249191E9104589C705583B86A422EE91CF299
22. https://instrumentsignpost.files.wordpress.com/2013/07/3rd_cit_year_student_project_energy_generation_and_monitoring.jpg
23. https://www.processindustryinformer.it/prodotto-update/espanso-powerflex-520-serie-di-compact-ac-drive-Debutti-powerflex-523-ac-drive-to-meet-macchine-builder-needs-per-appena-abbastanza-control/
24. https://www.youmath.it/lezioni/fisica/unita-di-misura/misure-di-frequenza/3101-rpm-giri-al-minuto.html
25. https://www.revereelectric.com/rockwellautomationhighlights
26. https://www.controleng.com/articles/ec-allen-bradley-kinetix-5500-servo-drive-with-integrated-safety/
27. https://ab.rockwellautomation.com/it/Motion-Control/Servo-Drives/Kinetix-Indexing-and-Component/Kinetix-300-Servo-Drive
28. https://www.plc-city.com/shop/9419-large_default/6sl3200-3ax00-0ul1-nfs.jpg
29. https://w3.siemens.com/mcms/topics/en/sidemo/systeme/pages/default.aspx
30. https://www.plc-city.com/shop/en/siemens-simatic-s7-1500-cpu/6es7511-1fk01-0ab0.html
31. https://it.emcelettronica.com/comunicazione-dati-in-tcpip