

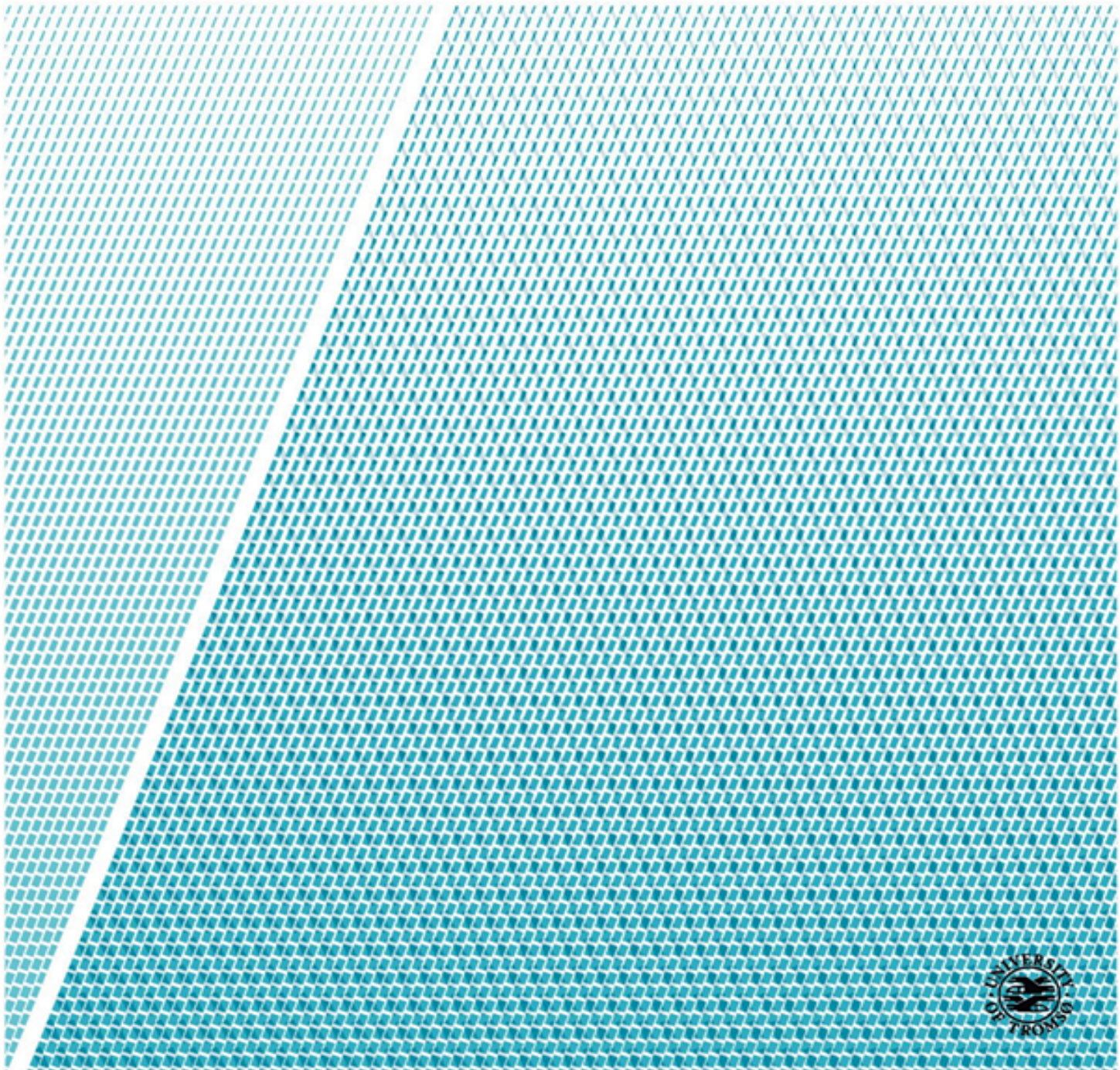


Faculty of Engineering Science and Technology
Department of Industrial Engineering

Design and Construction of a Flexible Manufacturing System Test Bench

Master thesis

Martin Røysland Andresen



Abstract

This thesis explores the history, concept and criteria of flexible manufacturing systems. A small-scale test bench is designed and constructed for educational and demonstrational purposes. The test bench is comprised of an industrial robot, a CNC machine, and a conveyor. These are all controlled and automated from a PLC with a connected touch panel for human machine interface.

It describes in detail the three main parts of building the test bench. First is the physical construction of the test bench and how the parts were machined using a metal band saw and a milling machine. Second is the interfacing between the components with electrical wiring, signal configuration and usage of OPC, PLC and PC to enable communication. The third and final step is the programming of the components where python, g-code, robot code and two different PLC codes all work together to create a seamless system with many customisation options.

Acknowledgements

Throughout the master thesis and master education as a whole I have had the pleasure of interacting with a vast amount of both fellow students and university staff. I have thoroughly enjoyed the long discussions, new experiences and of course the tremendous amount of new knowledge that has set me on a new course in life. I am very grateful to have had this opportunity and will carry it with me for the rest of my life.

There are some that have made a big impact on my master thesis and had a direct impact on the end result. First is head engineer Lazar Sibul and research assistant Dmitri Plotnikov at the university machine laboratory. Without their insight, guidance and help I would never have finished on time. Thank you very much for your patience and hard work.

Second is PhD candidate Beibei Shu who helped me at a critical time in my thesis. Your soldering skills enabled the robot communication and really saved my project.

Third is a fellow student and a dear friend, a former machine operator and an inspiration for us all, Peer Steinhauer. You have taught me so much about mechanical work, machining and life itself. Your willingness to help others never stops to astound me and it brings me great joy to call you my friend.

Fourth are my supervisors, the associate professors Jan-Arne Pettersen and Gabor Sziebig. You have both challenged me and my ways of thinking, and shared your wisdom to keep me on my course throughout the thesis. Your advises saved me from falling down the rabbit hole and helped me finish the project.

Finally, I would like to thank my dearest Maya, whom without I would never had started my masters, nor finished it. You have kept me sane, well fed and reasonably groomed for the past two years. All while taking your own masters degree. Your hard work and dedication is truly inspiring and I am in awe over how you still find time to love and care for me at the same time. This has been a fantastic journey and now we are ready to start the next together. I love you.

Table of Contents

1	Introduction	1
1.1	History	1
1.2	Flexible Manufacturing System	2
2	Existing solutions	5
3	Learning Goals	8
4	Available equipment	9
4.1	Nachi MZ07	9
4.2	Siemens 315 PLC	9
4.3	Siemens MP 370 touch panel	10
4.4	CNC milling machine	11
4.5	Trolley	11
5	Design	13
5.1	Virtual model	13
6	Construction	14
6.1	Robot	14
6.2	CNC machine	16
6.3	Clamps	17
6.4	Pneumatic air system	17
6.5	Conveyor	19
6.6	PLC	19
7	Interfacing	21
7.1	CNC	21
7.2	Conveyor	22
7.3	Robot	25
7.4	Touch panel	27
7.5	Pneumatic air system	27
8	Programming	28
8.1	PLC	28
8.2	CNC	29
8.3	Touch panel	31
8.4	Robot	32
9	Conclusion	34
9.1	Further improvements	34
Appendices		
A	User manual	i
B	Electrical drawings	xiv
C	Python code	xix
D	G-code	xxiv
E	Robot code	xxix
F	222 PLC code	xxxiii
G	315 PLC code	xliii

List of Figures

1	System24 - Illustration retouched [5]	1
2	Time line showing when FMS and related technologies were invented	2
3	Relationship between volume and variety in production systems	3
4	The five different types of FMS layouts - Illustrations recreated [14]	4
5	Example of a Festo Didactic FMS learning factory [17]	6
6	Communication lights by Werner Elektrik [21]	8
7	Nachi MZ07 robot	9
8	Siemens PLC with CPU 315-2 PN/DP, two DI modules and one DO module	10
9	Siemens MP 370 touch panel	10
10	3 axis CNC milling machine with JY5300 controller	11
11	Decommissioned demo factory - Used as base for the FMS test bench	12
12	Early design sketch of the layout	13
13	The design and strength analysis of the robot plate	14
14	The design and finished product of the robot pedestal	15
15	Cutting of the aluminium profiles with a band saw	16
16	Soldering the end stop wires on the CNC machine	17
17	Two pneumatic clamps with holders and two fixed sides with countersunk slots	18
18	Air system with intake on the right side	18
19	Conveyor with pneumatic extension - was not used due to damage	19
20	PLC cabinet with power supply	20
21	Bipolar stepper motor	22
22	Stepper motor driver	23
23	Positioning difference between full step and microstepping [36]	24
24	Structure	28
25	The completed flexible manufacturing system test bench	35

List of Tables

1	Signal combinations	22
2	Pulse train output based on input signals	24
3	PLC output to robot input	26
4	Robot output to PLC input	26
5	Connections between PLC and air solenoids	27
6	List of g-code programs and their functionality	30
7	Accuracy setting and corresponding allowed deviation from way point [42]	32

1 Introduction

At UiT The Arctic University of Norway, campus Narvik, there is a desire to have more practical exercises in the study plans for industrial engineering. This ensures that students participate in active learning, which gives a better understanding of concepts than traditional passive classroom learning [1].

This master thesis has been conducted in two parts. First a literature study was performed on the following subjects: Flexible Manufacturing Systems, their components and history. FMS and learning factories in education. Available equipment specifics. Learning factory didactic and maturity models. Research in the last topic did not yield much results as the articles were either too vague or too complicated.

The second, and main part of the thesis was to design and construct an FMS test bench that could act as a learning and experimentation tool for the students at the university.

1.1 History

One of the earliest forms of manufacturing automation is the gun copying lathe, invented by Thomas Blanchard in the 1820s [2]. It was seen as a great advancement in manufacturing and versions of it were later used to produce guns at all armouries in both the United States and England. His lathe subsequently evolved into the screw machine by hands of Christopher Miner Spencer in 1873 [3]. The screw machine was fully automatic and could produce multiple parts without human intervention. These machines used cams to automate the work, so once they were set up to produce a part they were not easily reprogrammable.

Shortly after the second world war, in the late 1940s - early 1950s, the Numerical Control (NC) machine was developed by John T. Parsons and Frank Stulen [4]. The machine could accurately produce complex parts based on coordinates from hole punch cards. It was originally conceived to produce helicopter propeller blades for Sikorsky Helicopters and was later popularised by the US Army for aircraft production. By the end of the decade the hole punch cards had been replaced with computers and thus the Computer Numerical Control (CNC) was created.

Around the same time, in 1954, the first industrial robot called Unimate was invented by George Devol [6]. He built the first prototype together with Joseph Engelberger in 1959 and sold it to General Motors. It was used to automate a hazardous job in a die casting factory which it did with great success and just a few years later there were about 450 Unimate robots in use.

In 1965, David Williamson was granted a patent on a concept he called *System24* [7]

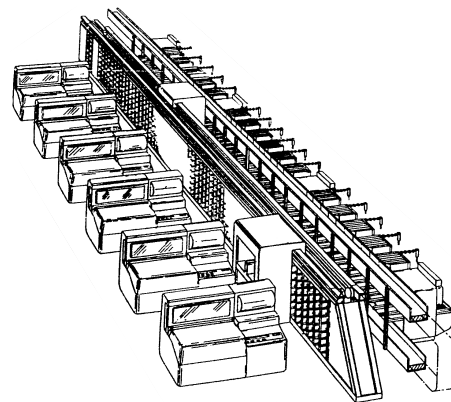


Figure 1: System24 - Illustration retouched [5]

as seen in figure 1. It was meant to operate 24 hours per day, whereby 16 of those would be unattended by humans. It had a computer controlling a NC machine and tool magazines, making it capable of producing a variety of parts. This is known as the first Flexible Manufacturing System (FMS).

Three years later, in 1968, the Programmable Logic Controller (PLC) was developed by Dick Morley and his team of engineers at Bedford and Associates. [8]. Again, the first customer was General Motors, who wanted to use them for controlling their production facilities. It was an instant success and the PLC industry grew rapidly. Today, there are dozens of PLC manufacturers worldwide. The most used PLCs are from Rockwell Automation, Schneider Electric, Mitsubishi, OMRON and Siemens [9].

Worldwide, there were 1.8 million industrial robots in use in 2016 and it is estimated to grow to over 3 million by 2020 [10]. The largest manufacturers of industrial robots, by robots in use, are Fanuc, Yaskawa, ABB, Kawasaki and Nachi [11]. A time line of the invention of FMS and the related technologies can be seen in figure 2.

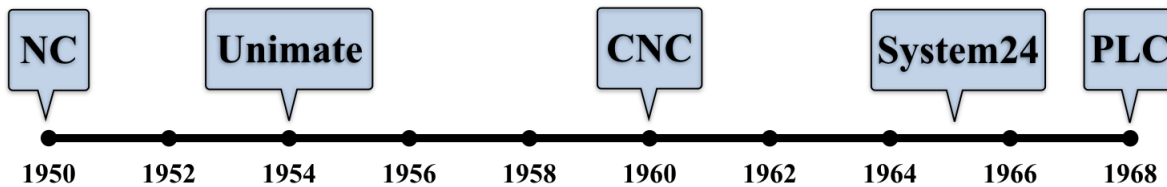


Figure 2: Time line showing when FMS and related technologies were invented

1.2 Flexible Manufacturing System

Shivanand et. al. has the following definition of FMS in their book on the subject:

“A flexible manufacturing system (FMS) is an arrangement of machines ... interconnected by a transport system. The transporter carries work to the machines on pallets or other interface units so that work-machine registration is accurate, rapid and automatic. A central computer controls both machines and transport system.” [12].

From this we can derive the three basic components of a flexible manufacturing system:

1. Workstations
2. Handling / transport / storage systems
3. Control system

Workstations are typically CNC machines, inspection stations or assembly stations. Handling systems include conveyors, industrial robots and storage racks. Control system can be a computer or Programmable Logic Controller (PLC).

According to CIRP Encyclopedia of Production Engineering there are seven different types of flexibility in an FMS [13]:

- | | |
|----------------------------------|--|
| 1. <i>Production flexibility</i> | How many different parts can be produced on the system |
| 2. <i>Expansion flexibility</i> | Can the system capacity or capability be easily expanded |
| 3. <i>Machine flexibility</i> | How many different operations can be done with one set-up |
| 4. <i>Product flexibility</i> | Can new products easily be introduced to the product mix |
| 5. <i>Routing flexibility</i> | How many alternate routes can a part take through the system |
| 6. <i>Volume flexibility</i> | Can the production volume be varied and still be profitable |
| 7. <i>Mix flexibility</i> | Can the system change between parts without affecting quantity |

FMS is best suited for mid-volume, mid-variety production. This means it strikes a middle ground between job shops and transfer lines as seen in figure 3.

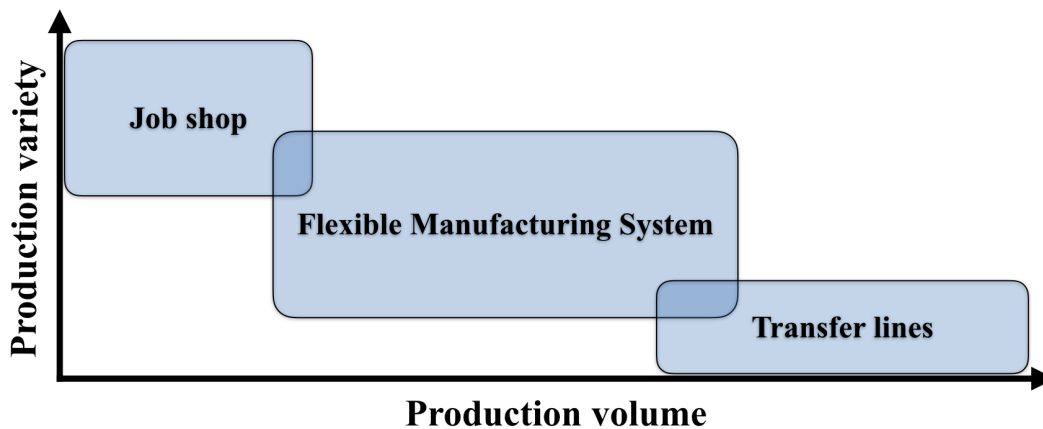


Figure 3: Relationship between volume and variety in production systems

There are five types of layouts for an FMS, as shown in figure 4 on the next page:

- | | |
|-------------------------|--|
| 1. <i>In-line</i> | Workstations are placed along a linear transfer system |
| 2. <i>Loop</i> | Workstations are placed outside a looping transport system |
| 3. <i>Ladder</i> | Workstations are placed inside a laddered transport system |
| 4. <i>Open field</i> | Workstations are placed inside several looping transport systems |
| 5. <i>Robot centred</i> | Workstations are placed around a revolving industrial robot |

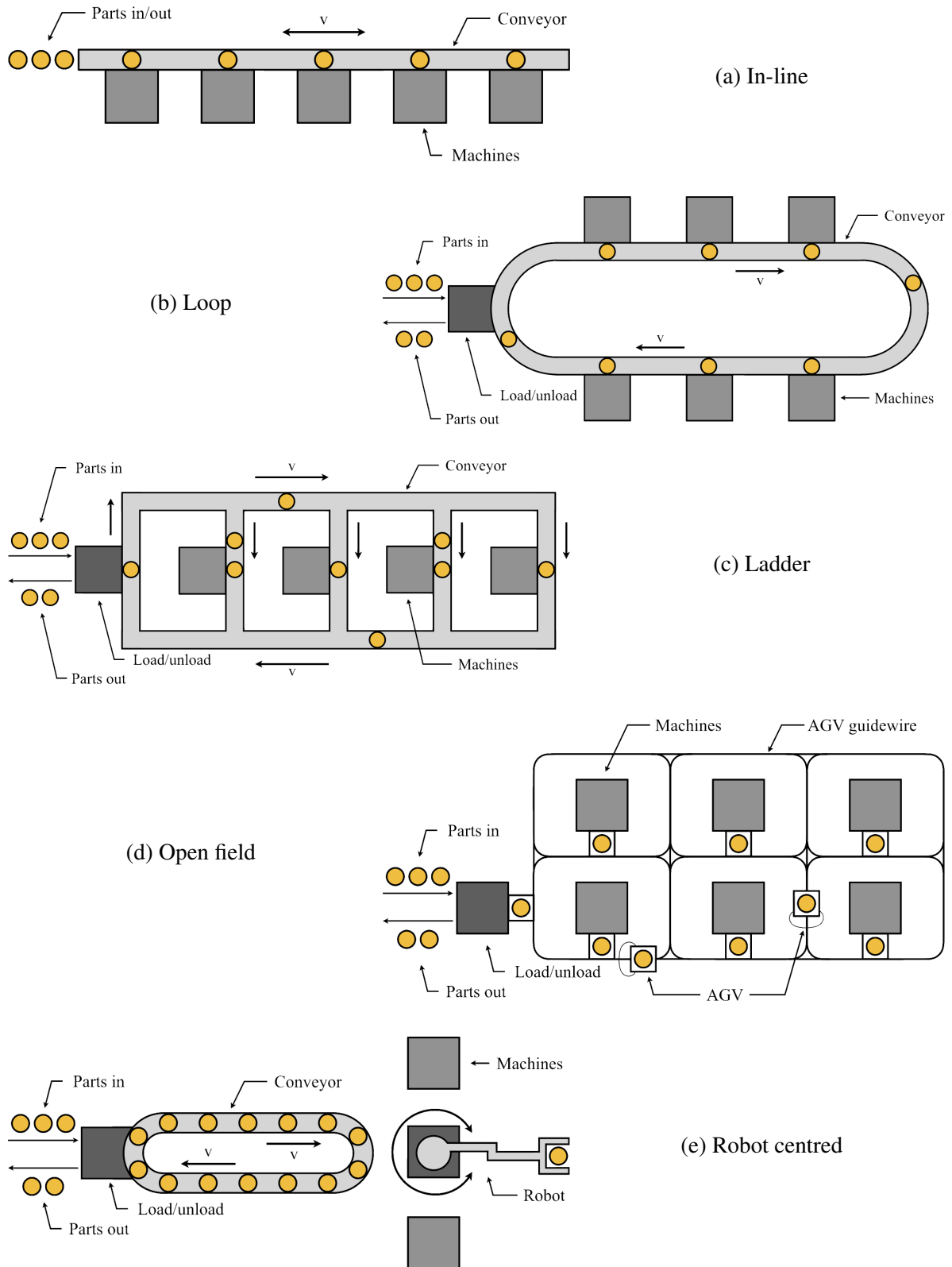


Figure 4: The five different types of FMS layouts - Illustrations recreated [14]

2 Existing solutions

There are commercially available learning factories that can be purchased and other universities has also developed mini factories for their own use in education. The size of the factories range from smaller than 1m² to larger than 40m².

Tampere university in Finland has an FMS training centre with a digital twin [15]. A digital twin is a virtual copy of the physical factory that is communicating with its real life counterpart. This combination allows the students to first experiment with the virtual model on the computer and simulate many different scenarios, and then test some of these on the physical FMS later.

Festo Didactic is a company that specialises in making training equipment and courses within factory and process automation [16]. They offer a plethora of learning factories, robots and courses. We will look into some of their FMS packages.

The MicroFMS comes with a robot and either a CNC mill, lathe, or both [17]. In an example work flow they produce chess pieces by combining both milling and lathing. The system has a buffer for both unprocessed and finished parts and the robot handles the work pieces in between the steps. It can be configured with a Supervisory Control And Data Acquisition (SCADA) system and Direct Numerical Control (DNC).

It is meant to teach CNC programming with g-codes, programming of industrial robots, signal processing and data management. Furthermore, the electrical and mechanical parts of the system can demonstrate pneumatics, milling and lathing processes, inter device communication and electrical drives.

The learning factories from Festo Didactic are modular and can be greatly expanded into complex systems by combining several packages. By combining their Computer Integrated Manufacturing package, iCIM, with the MicroFMS the system will gain Enterprise Resource Planning (ERP) and Computer-Aided Design and Manufacturing (CAD/CAM) capabilities. Other possible add-ons include a vision system with pallet recognition and sorting station.

Another combination for the MicroFMS is with the FMS 50 as seen in figure 5. This provides a quite extensive system with the following modules:

- Testing station
- Sorting station
- Handling station
- Conveyor system
- Distribution station
- 5 axis robot on slide
- CNC lathing machine
- CNC milling machine
- Supervisory SCADA system
- Automated storage and retrieval station

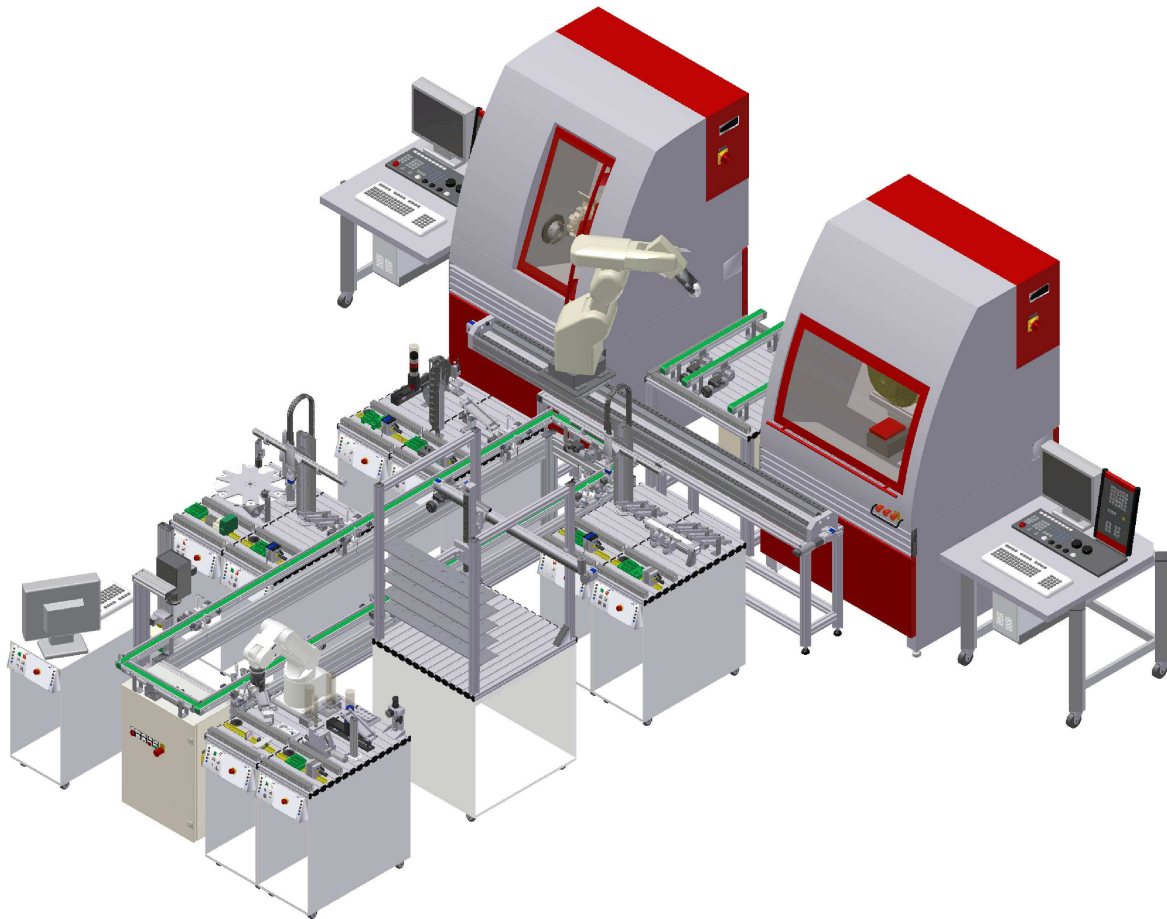


Figure 5: Example of a Festo Didactic FMS learning factory [17]

The Central Community College in Columbus, Nebraska has a Mechatronics Education Centre (MEC). Within the centre they have an FMS comprised of 10 modules [18]:

- Robot station
- Buffer station
- Testing station
- Sorting station
- Handling station
- Processing station
- Distribution station
- Pick and place station
- Fluid muscle press station
- Pallet transportation system

The modules can be used individually or in combination with the others.

In Barcelona, Spain, the Technical University of Catalonia has a flexible manufacturing cell with a SCADA system [19]. Their educational focus is on designing, configuring and using the SCADA system. It has the following components:

- Sorting station
- SCADA system
- Handling station
- Conveyor system
- Processing station (2x)

3 Learning Goals

In order to ensure that the FMS test bench would incorporate the desired learning goals to support the universities lectures, a joint meeting was held with the teachers in CAD/CAM, quality management, and robotics. In the meeting, several criteria and ideas were discussed.

Since the test bench would use the CNC machine it was important that there would still be a possibility to run custom g-code on it so students could test their own designs. The CNC machine would also need to have an automatic fixture mechanism to fully automate the process.

When it comes to quality management and lean six sigma there are several ways it could be used to demonstrate and teach the ideas within the subject. The students should be able to change input parameters and then be able to observe a change in the output. This could be related to performance and tolerance, for example that too high cutting speed results in inadequate accuracy.

Another element from lean six sigma are communication lights, also called stack lights or towers lights. These come in many variations, but the most common one is with a red, yellow and green light [20] as seen in the middle of figure 6. The lights will give information about the status of station. Green is ok, yellow is warning, and red is alarm. It can be used on buffers for example, with a green light if the buffer is healthy, yellow as it gets low, and red if it is depleted.



Figure 6: Communication lights by Werner Elektrik [21]

Poka-Yoke, meaning mistake-proofing [22], is a third element that can be implemented. The system should make sure that the right conditions are in place before executing the next step. If errors are detected they should be dealt with as soon as possible to prevent cascading errors.

4 Available equipment

The goal of the thesis is to design and build an FMS test bench for the UiT machine laboratory. The lab already has most of the equipment needed for this project and we will in this section be going through the available equipment that can be used for the learning factory.

4.1 Nachi MZ07

UiT - Narvik has several industrial robots in their lab. They range from large robots used for welding, to small SCARA robots used for assembly. The robot dedicated for this project is the Nachi MZ07. The definition of an industrial robot can be found in ISO 8373:2012:

“Automatically controlled, reprogrammable, multipurpose manipulator, programmable in three or more axes, which can be either fixed in place or mobile for use in industrial automation applications” [23].



Figure 7: Nachi MZ07 robot

The MZ07 seen in figure 7 is a fast and lightweight compact robot. It is an articulated robot with six joints and a reach of 723mm [24]. The robot mass is 30 kg and max payload is 7 kg, giving the robot a payload-mass ratio of 0.23. It is touted by Nachi as the worlds fastest robot in its class with a 300×25 mm movement cycle time of 0.31 seconds.

The robot comes with a control unit that has a built in software PLC [25]. Standard configuration includes Ethernet networking and either a compact or smart teach pendant. Through configuration it can also have profibus, profinet, devicenet and CC-link networking. Nachi's vision sensor, NV-Pro can also be added to the package.

4.2 Siemens 315 PLC

Programmable logic controllers (PLCs) is a type of computer that is built for industrial use and differs from a normal PC in several ways. A PLC is rugged and built to withstand harsh environments often found on industrial sites. It can have over thousand inputs and outputs, depending on the make and model, and is typically used for monitoring and controlling industrial systems [26].

For this project we will use the Siemens CPU 315-2 PN/DP as seen in figure 8. To get a better understanding of it we will start by deciphering the product name. CPU stands for Central Processing Unit, which means it is the unit executing the program. The 3 denotes the PLC family (300-series), followed by 15 which denotes the capabilities of the CPU. The last is an arbitrary number where higher is better. The 2 denotes the number of interfaces available and

PN/DP are the two types of interfaces. PN stands for ProfiNet, and DP stands for (Profibus) Decentralised Peripherals [27].

In addition to the CPU, a PLC can have several other modules. The most common ones are digital and analogue input and output. They are abbreviated as DI, DO, AI and AO. These are used to communicate with sensors, motors, buttons, actuators etc. Another common module is the DP/DP coupler which interconnects two profibus DP networks.

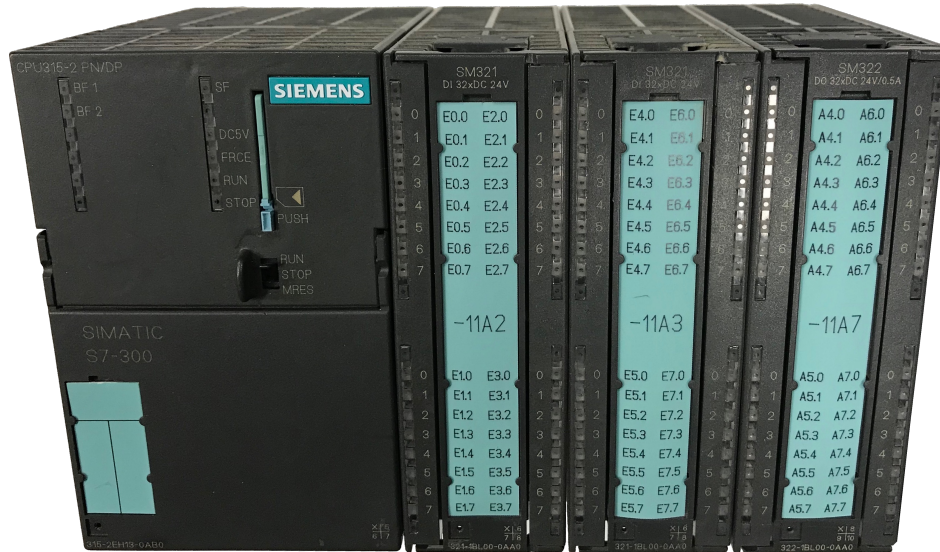


Figure 8: Siemens PLC with CPU 315-2 PN/DP, two DI modules and one DO module

4.3 Siemens MP 370 touch panel

The MP 370 touch panel from Siemens seen in figure 9 is an HMI panel that can be used to control and monitor various industrial systems. It is powered by Windows CE (Embedded Compact) with Siemens WinCC flexible [28]. The numerous connection ports it features are as follows:

- RS-442/485 for PLC connection
- RS-232/TTY for PLC connection
- RS-232 for configuration and printer
- PC-card slot
- CF-card slot
- DP/MPI/PPI
- USB
- Ethernet

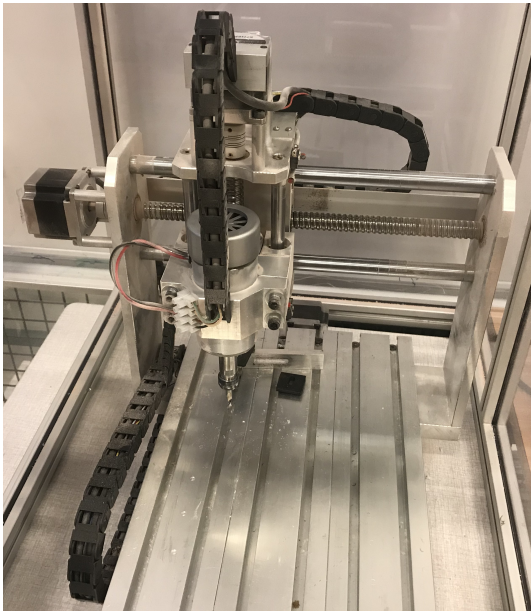


Figure 9: Siemens MP 370 touch panel

4.4 CNC milling machine

The UiT lab has a 3 axis CNC milling machine. It is controlled by a JY5300 CNC machine controller from MIB instruments. Both of these can be seen in figure 10 The JY5300 can get instructions from either a handweel pendant or a PC running a CNC software like Mach3.

The combination of JY5300 and Mach3 software provides the user with a lot of tools for CNC milling. There is a built in AutoCheck function that can check the Z height of the tool [29]. Precise control of stepper motors and spindle motor. Limit switches to prevent collision and water cooling system control.



(a) CNC machine



(b) JY5300

Figure 10: 3 axis CNC milling machine with JY5300 controller

4.5 Trolley

The university had a decommissioned portable demo factory from Festo Didactics that could be used as a base for the FMS test bench. It is a trolley that is 2 meters long and 70 centimetres wide and has cabinets inside as seen in figure 11. There are four cabinets; the largest one in the middle, a large side cabinet and two smaller side cabinets. I removed almost all of the equipment on it, but left some parts that could possibly be reused either in my project or a future expansion. The pieces that were kept was an elevator with tree shelves for storing the finished work pieces, and a conveyor right next to the elevator.

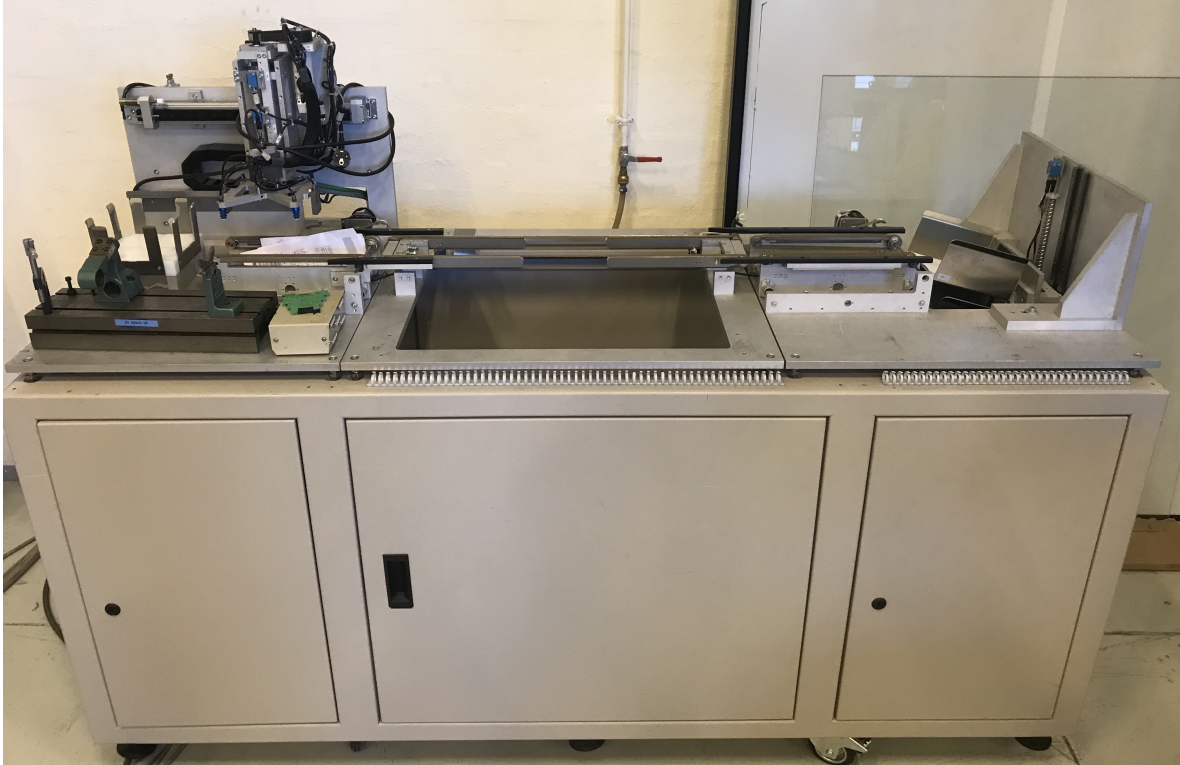


Figure 11: Decommissioned demo factory - Used as base for the FMS test bench

5 Design

When I set out to design the test bench I was quite constrained by the available equipment. Since I had a robot it was natural to go for a robot centred layout. The long layout of the trolley meant the most logical placement of the robot would be in the middle so it could have maximum reach. However, since the robot cannot reach very close to itself I would have to place it to the top or bottom. Since I decided to keep the existing conveyor and elevator there was only one side with enough room for the CNC machine. I made a quick sketch of how the layout would be as seen in figure 12.

There was a lot of uncertainties with how much time the different tasks of building the test bench would take. I therefore decided to start construction early to get a working version before adding a lot of extra functions. Given the flexibility of the robot it would be trivial to add functions at a later stage.

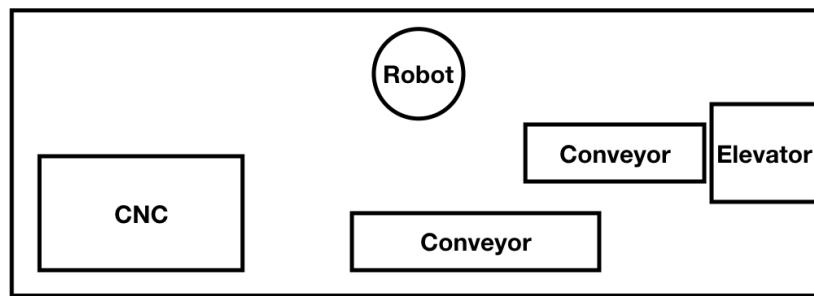


Figure 12: Early design sketch of the layout

5.1 Virtual model

To validate the design I decided to test it out in a virtual environment. Visual Components is a factory simulation software used by hundred of companies worldwide [30]. I used their latest iteration, version 4.1, to make a simple virtual model of the test bench. One of the strengths of Visual Components is their vast library of industrial robots. I found the Nachi MZ07 and simply dragged it into my model. I then arranged the rest of the components and checked if the robot was able to reach them. I found that the base of the robot was causing it to stand a lot taller than the rest, which meant it was hard to reach. I tried lowering the model down so the swivel would be flush with the top of the trolley and test reachability again. This worked much better so I needed to find a way to lower it when constructing the test bench. Even though it was a simple model, it gave me the confidence I needed to start construction.

6 Construction

This section describes the work performed with physically building the test bench and the decisions I made along the way.

6.1 Robot

The trolley has 3 top plates made of 1.5 cm thick aluminium. The outer ones are quite solid, but the middle one is mostly just a frame with a big hole in it. The robot would need to be placed in the middle of the FMS in order to ensure reach-ability to most areas. I therefore designed a new top plate for the middle part in SolidWorks that would be mostly solid and with a recessed area for the robot. I ran a strength analysis on the design in Autodesk Inventor based on max momentum numbers from the robot manufacturer. The stress and displacement of the plate was acceptable with approximate values of 100MPa and 0.2mm respectively. The design and the strength analysis can be seen in figure 13.

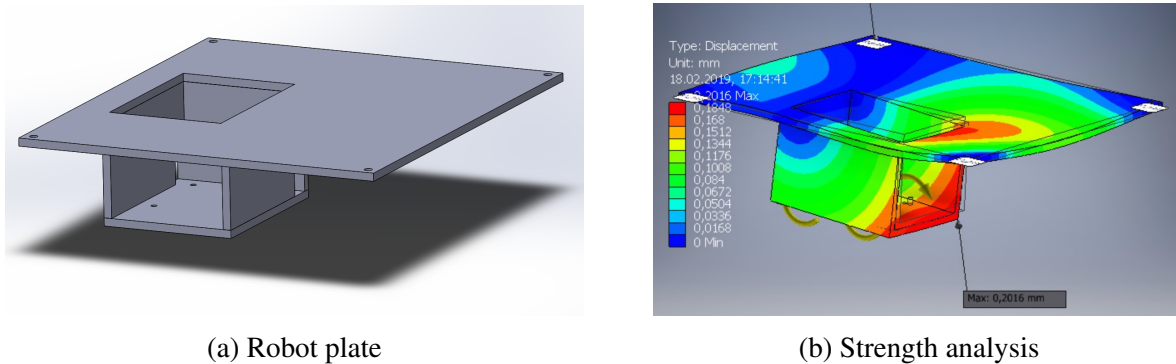
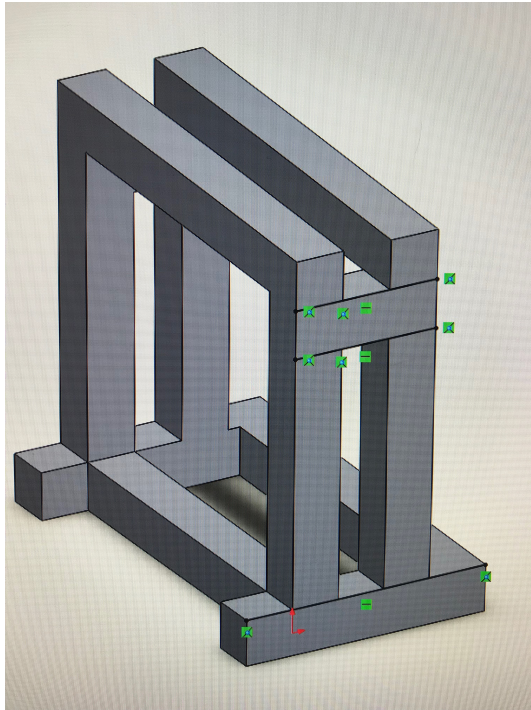


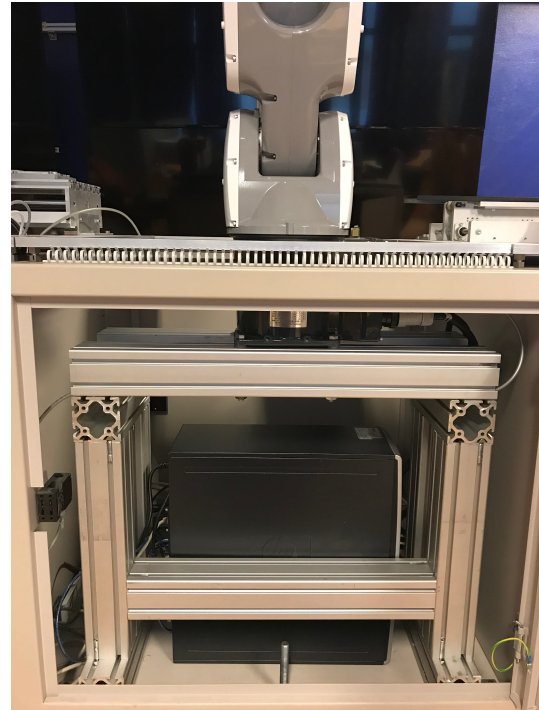
Figure 13: The design and strength analysis of the robot plate

When I started looking into actually making the plate however, I found out that the university did not have a plate at this size available and there could be long lead time on getting a new one. I therefore investigated other options and discovered that there was a lot of 80×80mm aluminium profiles available. The trolley has a thick solid bottom and I could build a pedestal for the robot with the profiles. I measured the length of each profile before working on my design. I had a lot of equipment I wanted to fit in the trolley and since the pedestal would take up a lot of valuable space in the middle cabinet I wanted to make it multipurpose. One of the largest items I needed in the trolley was the PC for the CNC machine, by making the robot pedestal wide I would not only be able to fit the PC inside, but also achieve higher stability. Again, I made the design in SolidWorks, but this time it was only to visualise the design before committing to it and start cutting it. I went through a few iterations before ending up with the final design seen in figure 14a.

Once I had the design and knew all the lengths I needed I selected the aluminium profiles that were the closest match for my needs in order to minimise waste. I also considered whether I could make multiple lengths out of one work piece and tried to save the longest pieces for the future. As seen in figure 15 I cut the profiles in a metal band saw to the correct length.



(a) Design



(b) Installed

Figure 14: The design and finished product of the robot pedestal

Afterwards I filed the sharp edges so there was no danger of cutting oneself on them. The profiles have two grooves on each side that are used when joining them together. There are only three different types of pieces needed in the joining process:

1. Regular machine bolts
2. Hollow screw to stop the bolt at the edge
3. Special nuts that can be inserted in the groove

The heads on the bolts I used were larger than the surface width of the groove, they were therefore inserted at the edge first. I then screwed in the hollow screw into the edge of the groove. The nut was inserted into the groove of the other work piece. Once these preparations were done I could hold the pieces together perpendicularly and screw the bolt into the nut. The profiles were then easily assembled, except for the two profiles on the top. My design had them laying on top two other profiles, which meant I had no perpendicular grooves meeting each other. I therefore used four 90° angles to attach them. However, before attaching them I drilled two 11mm holes with 160mm centre distance between them on each of the profiles. These were for fastening the robot on top of it. I also attached four 90° angles to the sides of the pedestal along the bottom so I could fasten it to trolley floor.

Once the pedestal was fully assembled I lifted it on through the middle cabinet door and made sure the bottom profiles were fully over to cabinet door side. Holes were drilled in the bottom of the trolley, bolts inserted and fastened with nuts. I then installed the robot on top of



Figure 15: Cutting of the aluminium profiles with a band saw

the pedestal and fastened it. To ensure maximum stability and working envelope of the robot I loosened the bolts between the bottom, horizontal and vertical profiles before sliding the vertical ones as far towards the cabinet door as possible. Once in place I fastened the bolts again. This resulted in the robot base now pressing against the aluminium plate on the top of the trolley. The installed pedestal can be seen in figure 14b.

The CFD controller for the robot was placed in the large side cabinet underneath the elevator.

6.2 CNC machine

The CNC machine had previously been used by the university and it was therefore fully assembled and ready to use. During testing however I discovered that the end stops for the Y axis did not work. Upon further inspection I found that the drag chain fastening was not able to hold the drag chain in place as there was a lot of torque exerted on it. This had caused the drag chain to come loose and thus all the force had been on the two small wires, causing them to break. As seen in figure 16, I soldered the wires back on to the end stop sensors and attached the drag chain to the fastener with a cable tie. This ensured that some torsion of the drag chain was allowed while still keeping the force away from the wires.

The CNC was fastened on top of the trolley using three 90° angles. A wide one in the front and two smaller ones in the back as to not interfere with the Y axis motor. The wires go down a hole a centre hole in the plate and into one of the small side cabinets where the JY5300 CNC controller has been placed.

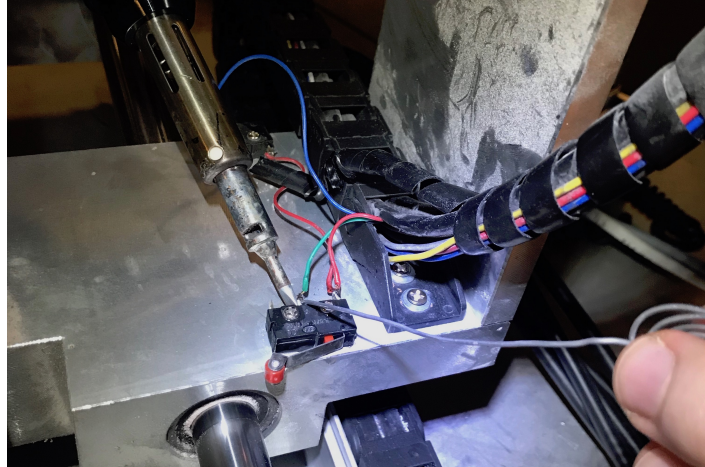


Figure 16: Soldering the end stop wires on the CNC machine

6.3 Clamps

To hold the cube in place I needed to create an automatic fastening system. I decided to make a fastener with two fixed sides and two sides with moving clamps. This allows for variety in cube size, but also high accuracy since the cube will always be pushed against the fixed corner. I found two pneumatic clamps with 20mm stroke that would work perfectly for my usage, but I needed to make some holders for them. I took a solid block of aluminium and precision machined the clamp holders and the fixed sides out of it with a milling machine.

There are grooves on the CNC base plate with 40mm centre distance between them. In order to allow maximum flexibility in the placement of the clamps I made long slots on the holders so they can slide before tightening the screws. They are fastened with the same system that is used for the robot pedestal profiles. For the side clamp holder I made two 40mm long slots so you will always be able to find a groove that fits and you can fasten it with two bolts no matter the placement. Likewise for the top clamp holder, this has one 80mm long slot so it too can be fastened with two bolts no matter the placement.

The fixed sides are equal to each other and both have one 80mm slot. The slots are counter-sunk so the bolt heads are flush with the highest point. This is important because the robot clamps come close to them and this allows a better grip on the cube. The clamps, their holders and the fixed sides can be seen in figure 17

6.4 Pneumatic air system

The air system for the clamps and the robot have been placed in the small side cabinet with the CNC controller. At the intake there is a filter to make sure the air is clean and no particles will destroy or jam the system. From the filter the air is split into two lines. On the first line it goes into a pressure regulator and then directly to the robot air intake. The other line also goes into a pressure regulator and then into a valve block. There are three valves in the block, two of them are used for closing the CNC fixture clamps while the third is for opening them. The air system can be seen in figure 18.



Figure 17: Two pneumatic clamps with holders and two fixed sides with countersunk slots

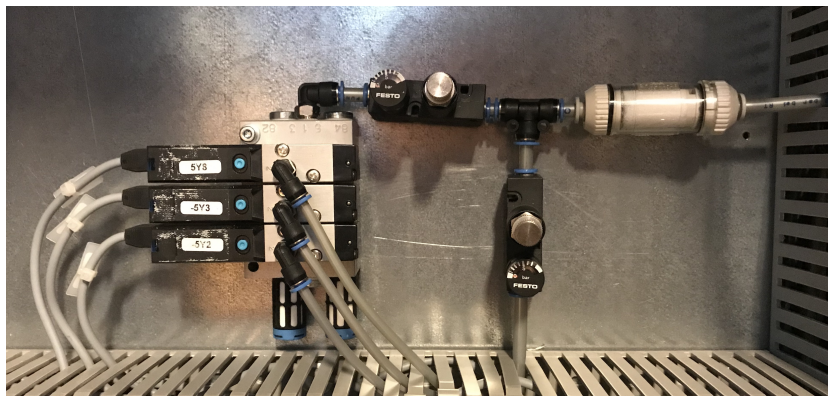


Figure 18: Air system with intake on the right side

6.5 Conveyor

From the design concept I had planned to use a conveyor to bring a solid cube to the robot for it to pick up. I found an used conveyor that was able to extend and retract using pneumatic cylinders. It was standing on a base, making it 30cm tall. It was perfect for my application.

It was originally equipped with a servo motor, but this needed a large and complicated driver to run it. I therefore decided to replace the motor with a stepper motor. The motors were almost identical in dimension, but i needed to expand the fastening holes on the conveyor to fit the stepper motor. I also needed to replace the motor cog as it did not match the conveyor belt. However, once I had the motor in place and started experimenting with the conveyor I discovered that one of the belts were damaged and it kept jumping off the track. Thus I could not use it and had to find another solution. The scrapped conveyor can be seen i figure 19.

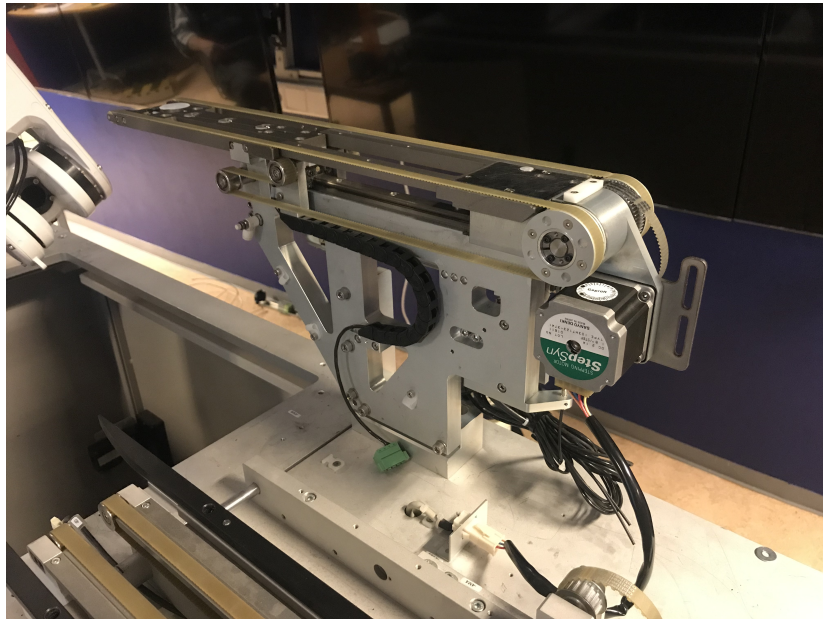


Figure 19: Conveyor with pneumatic extension - was not used due to damage

I decided to use the conveyor that I had saved from the original demo factory instead. The conveyor had two belts driven by an axle connected to the motor. The belts could be adjusted sideways perpendicular to the axle so I moved them as close together as possible in the middle. The distance between the belts was then 25mm and I could place a cube directly on them without needing a pallet.

6.6 PLC

The PLC have been placed in the other small side cabinet together with other control units and the 24 volts power supply. The equipment is securely mounted on rails and wires have mostly been laid in cable ducts, although I did run into problems with the 40 and 20 conductor cables for the robot. The PLC cabinet can be seen in figure 20.

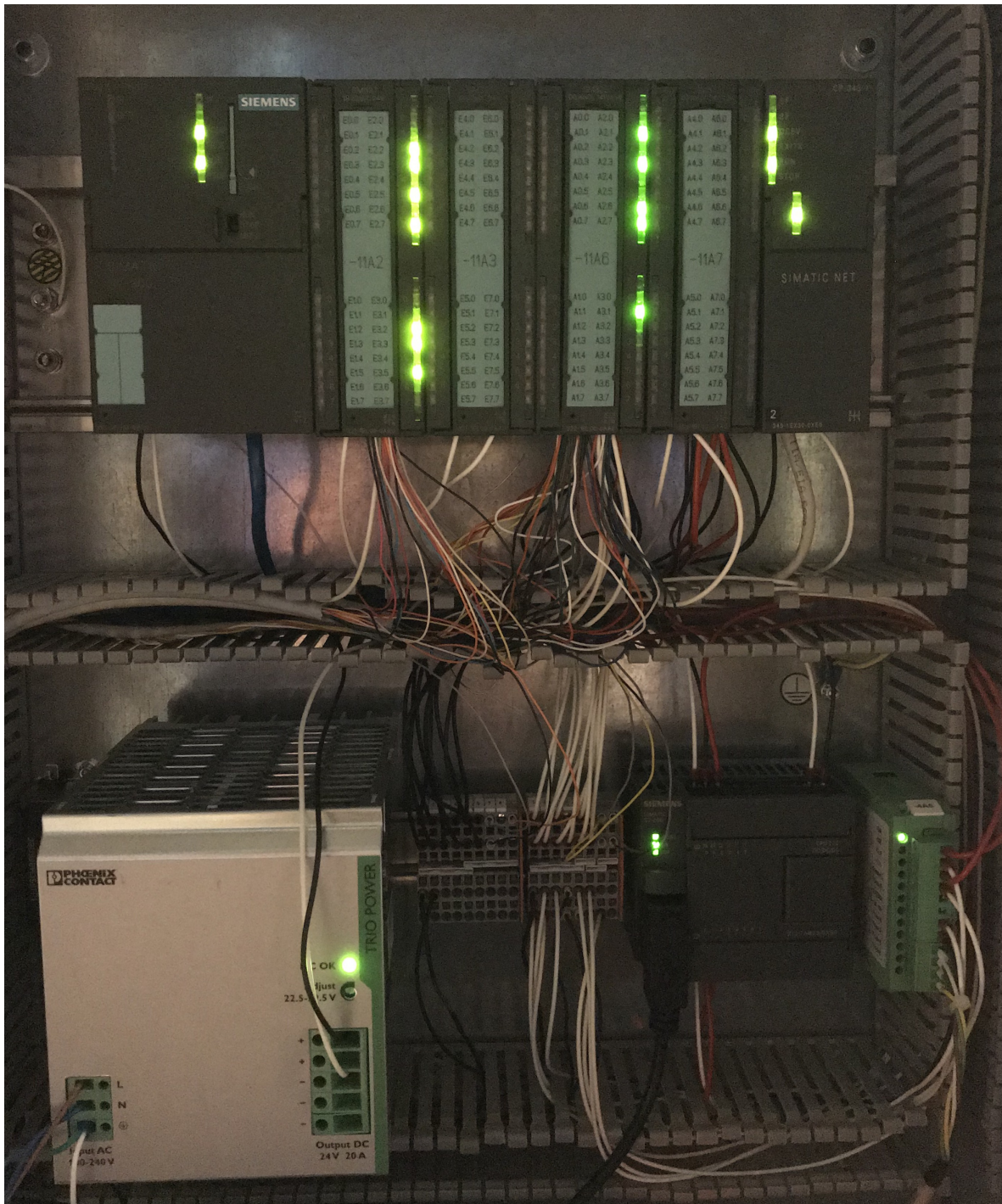


Figure 20: PLC cabinet with power supply

7 Interfacing

This section describes how all the components in the test bench are connected and how they communicate with each other. Electrical drawings can be found in appendix B.

7.1 CNC

The CNC is controlled from the Mach3 software on the PC. So I needed to solve two problems in to control the CNC from the PLC; first is communication between PLC and PC, and the second is a way to automatically control the Mach3 software. For the communication part I looked into OPC.

OPC is a standard for exchange of data between devices within industrial automation [31]. It was created in 1996 and was originally an acronym for OLE for Process Control where OLE again is an acronym for Object Linking and Embedding. As the standard has evolved, so has the meaning of the acronym, and today OPC stands for Open Platform Communications. With an OPC server running on a computer, it is possible to read and write values inside a connected PLC.

In WinCC Flexible it is possible to run an OPC server alongside the run time environment. This has been activated to enable communication and interaction between the PC and the PLC. The communication was tested with an OPC scout software and all signal communication has been verified. Once the OPC server was up and running and the communication verified, I needed to find a way to use and control the signals together with the Mach3 CNC software.

I investigated the Python programming language and its capabilities. A Python code does not need to be compiled before running, thus making it very fast to develop and troubleshoot [32]. It can also be extended with numerous extensions created by users all over the world. Since the PC is running Windows XP service pack 3 I was not able to use the latest version of Python, 3.7. I therefore had choice between version 2.7 and 3.4. I then started to look into which extensions I would need. In order to utilise the OPC communication I had established earlier I looked for a way to implement it in a Python code. I found an old extension called OpenOPC that works with Python 2.7. After installing it I wrote a quick test program and confirmed that I was able to both read and write values in the PLC through the OPC by using Python.

The next step was then to interact with the Mach3 CNC software. When searching for program automation I found the pywinauto extension. It can find windows and buttons in any software and give keyboard or mouse inputs. It was quite challenging to find all the buttons in Mach3 as some are not created as a standard button and therefore not recognised. I was able to get them all working by using a combination of button presses and keyboard shortcut inputs however. To get feedback from Mach3 I used a function to read the colour of the pixels on the screen. This is not an ideal solution as it is prone to errors if things are not in the correct place and some precautions had to be made. First, I made sure that the Mach3 software is always opened in full screen. Second, The mouse cursor is moved far away from

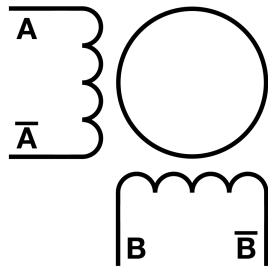
the measure area right before measuring. Third, the software samples the colour and compare against it later. This is important in case there is a slight change in colour. With this method I could get feedback on if a CNC code was running, or whether or not the reset alarm was active.

All of these functions have been joined together in one Python code that is running continuously and started automatically when the PC starts. Some of the extensions I used requires other extensions to also be installed. All in all, the following has been installed:

- Python 2.7.15
- OpenOPC 1.3.1
- pyro 3.16
- pywin32 224
- pywinauto 0.6.5
- comtypes 1.1.7
- six 1.12.0

7.2 Conveyor

The conveyor is driven by a bipolar stepper motor with 200 steps per revolution. A bipolar stepper motor is driven by two coils with two inputs each. A schematic of a bipolar stepper motor can be seen in figure 21. This type of motors are driven by giving a high signal on two inputs and a low on the other. A and \bar{A} must always be opposite of each other, and so must also B and \bar{B} . This leaves us with four different combinations as seen in table 1 below.



A	\bar{A}	B	\bar{B}
1	0	1	0
1	0	0	1
0	1	1	0
0	1	0	1

Figure 21: Bipolar stepper motor

Table 1: Signal combinations

The signals must be feed to the stepper motor in the correct sequence to turn the motor. The sequence also determines if the motor should turn clockwise or counterclockwise. It is normal to use a stepper motor driver to generate the signals for the motor In this project I have used a K480216A-24 from Middex Electronic [33]. This needs to receive a pulse signal for each time it should change the signal output and drive the motor one step further. It also needs a signal for which direction the motor should turn.

The +Dir and +Clk are the inputs for the direction and pulse signals accordingly. The -Dir and -Clk must be connected to ground or 0V so the driver can accurately interpret the difference in voltage when receiving signals. It also needs a 24v power supply in order to work. The outputs A , \bar{A} , B and \bar{B} are connected to the stepper motor.

Since the stepper motor driver need a pulse signal to drive I explored the options I had for

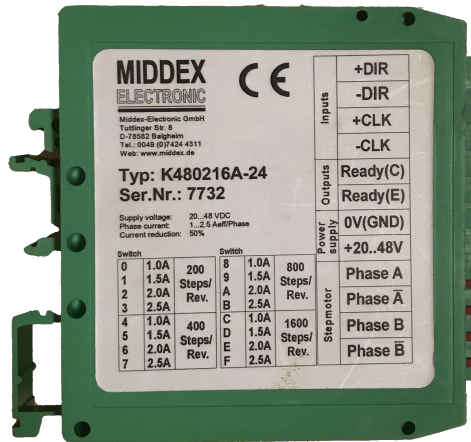


Figure 22: Stepper motor driver

making a pulse in the PLC. The 300 series PLCs have a built in function for generating a clock pulse. However, the fastest pulse from that function is 10Hz. At that frequency it takes 20 seconds to make one revolution of the motor. I estimated that I needed a pulse that would be 500 - 1000Hz for my conveyor.

I therefore tried to make a pulse using timers in the PLC. I set up two timers that would reset each other and alternate an output signal. The fastest timers I was allowed to make was 10ms, which meant the full pulse would have a cycle time of 20ms resulting in a 50Hz pulse. Still too slow. My third attempt was to use a set/reset flop flop switch in OB1 in the PLC. OB1 runs as fast as possible with a minimum execution time of 1ms depending on how much code the PLC need to run. I checked that my PLC was executing OB1 on 1 - 2ms. With this I got a 500Hz pulse, but it was not working reliably which resulted in jerky motion of the conveyor and therefore I could not use it.

I then started exploring options for using other hardware to generate the pulse train. For the 300 series PLCs, Siemens offer a separate module called FM 353 to drive stepper motors [34]. With this I would not need to use the stepper motor driver as this module would drive the motor directly. However, this would have to be purchased so I first checked what equipment was available at the university already. During my research on how to generate pulses I discovered that the 200 series PLCs from Siemens have a special function for outputting high frequency pulse trains. I found a Siemens 222 PLC in the university storage room and checked the specifications for it. It has two outputs that can make pulses up to 20kHz, more than enough for my usage [35].

To program the 200 series PLCs one must use Siemens Step 7 Micro/Win, which is the predecessor to Simatic Step 7. It is quite similar, albeit at bit more basic. Luckily it comes with a wizard for creating pulse train outputs with ramping and different frequencies so it was quite trivial to create code. I had to determine which frequency I should use for my motor and tried with many different speeds. I noticed that I was getting a lot of vibration on the motor which propagated to the conveyor belts and caused to blocks to slightly jump and therefore not follow the motion of the conveyor. This was especially noticeable with slow speed and I

attributed it to the low resolution of the stepper motor. With 200 steps per revolution, 1 step equals 1.8° . I therefore decided to look into microstepping.

Microstepping is a technique to artificially add more steps to a stepper motor by moving it part ways of the physical step at the time [36]. It is achieved by sending a pulse width modulated signal to the stepper motor with a 90° phase shift. This way you can get smoother motion between the steps which results in much less vibration. The difference can be seen figure 23.

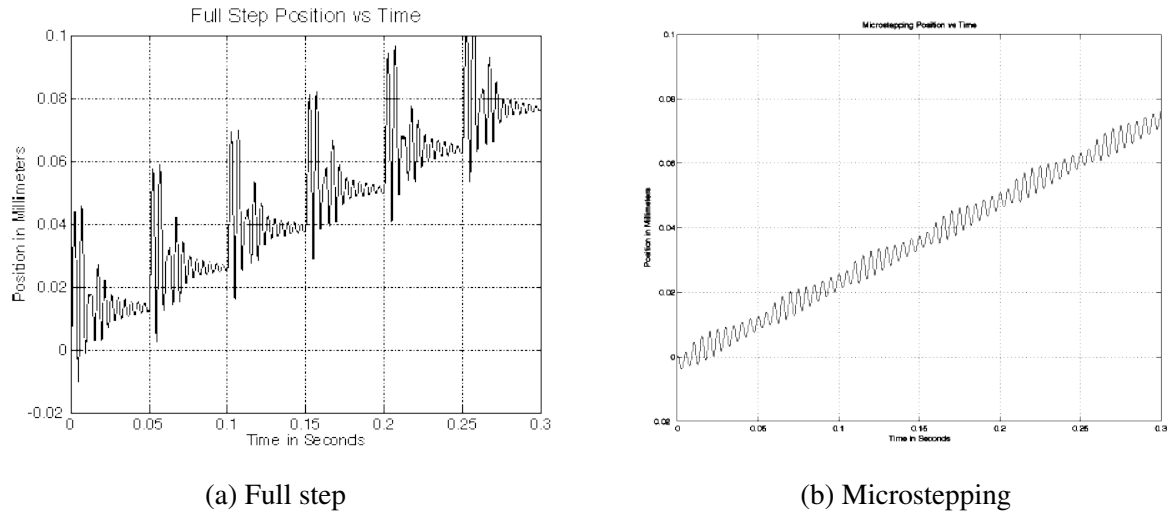


Figure 23: Positioning difference between full step and microstepping [36]

With the stepper motor driver from Middex Electronic it is very easy to achieve microstepping. It is simply a matter of turning a settings knob on the top of the unit. It can be set to either 200, 400, 800 or 1600 steps per revolution. Knowing that I could produce up to 20kHz pulses from the 222 PLC I decided to go for the 1600 steps per revolution, giving me 8 times higher resolution. I tested at various speeds and found that a 10kHz pulse resulted in a suitable fast speed. It was also very smooth with little vibration.

Since I wanted to have some choice in speed I made a simple program on the 222 PLC where I used the first two inputs (I0.0 and I0.1) to control the frequency of the pulse train on output Q0.0. The combination of inputs and resulting frequency can be seen in table 2.

I0.0	I0.1	Q0.0
0	0	0Hz
1	0	1000Hz
0	1	5000Hz
1	1	10000Hz

Table 2: Pulse train output based on input signals

As previously mentioned, the 222 PLC have two outputs that can make these high frequency pulse trains. In my construction I only use one stepper motor therefore only one output.

However, for future expansion opportunities I have made an identical logic for the other output in case it is needed later. The inputs I0.2 and I0.3 will then determine the frequency of output Q0.1. The full code can be seen in appendix F - 222 PLC code.

7.3 Robot

The CFD controller that comes with the MZ07 has an Ethernet port but this is only for programming and backup, not communication with external devices [25]. It has 3 card slots for optional expansions and in order to get the PLC interfaced with the robot controller I would need to acquire one. I checked which options were available and suitable for my project. I ended up with five options:

1. I/O Photo coupler 32 inputs / NPN Transistor 32 outputs
2. I/O Photo coupler 64 inputs / NPN Transistor 64 outputs
3. I/O Photo coupler 32 inputs / PNP Transistor 32 outputs
4. I/O Photo coupler 64 inputs / PNP Transistor 64 outputs
5. PROFINET board Slave 1CH

First I had to choose between I/O card or profinet card. A profinet would allow me to transfer vast amount of signals between the PLC and robot with 2048 inputs and 2048 outputs whereas an I/O card would have either 32 or 64 inputs and outputs [37]. I looked at how many signals I would need and found out it would be sufficient with 12 inputs and 8 outputs. An I/O card can be connected with any system that has digital inputs and outputs while the profinet card can only communicate with other profinet devices. Given that an I/O card with 32 inputs and outputs is more versatile, has more than enough signals and is also cheaper than the rest it was an obvious choice.

The only thing left was to decide between NPN and PNP. The acronyms stand for "Negative - Positive - Negative" and "Positive - Negative - Positive" respectively. The name refers to the transistor type used on the cards and how they work. An NPN card will give a high voltage signal on logic "0" and low voltage on logic "1" whereas a PNP card gives high voltage "1" and low voltage "0" [38]. In an industrial environment a PNP card is a safer choice since a power loss or failure on one of the outputs will only result in a "0" as opposed to an NPN card that will give a "1" in case of failure. I therefore ended up ordering option 3, I/O Photo coupler 32 inputs / PNP Transistor 32 outputs.

Once the card arrived I installed it in the CFD controller. However, it did not come with any cables for the inputs and outputs, only connectors where you could solder on your own cables. I got help from one a PhD candidate at the university, a former hardware engineer, to solder the wires. We used the cables that were available at hand and we found one 40 conductor cable and one 20 conductor cable. Since I needed more inputs than outputs we used the 40 conductor cable for the inputs. Both the inputs and outputs have a common signal per 8 signals. On the inputs you send a 24v signal on each input, so the common must be connected to 0v to complete the circuit. On the outputs you must feed the 24v signal they send out, thus the common must be connected to a 24v power supply.

The signal cables from the robot output were connected to an input module on the PLC, and inputs to the robot were connected from a PLC output module. The most important signals are pre-configured from the manufacturer and these have been kept even though I did not use all of them. I have also added a few signals. The signals and their names as they appear in the robot control system can be seen in table 3 and 4. A dash means that a wire is run, but no signal configured. All wiring may also be found in appendix B - Electrical drawings.

PLC	Robot	Name
Q8.0	I1	-
Q8.1	I2	-
Q8.2	I3	-
Q8.3	I4	-
Q8.4	I5	-
Q8.5	I6	-
Q8.6	I7	-
Q8.7	I8	-
Q9.0	I9	-
Q9.1	I10	-
Q9.2	I11	-
Q9.3	I12	-
Q9.4	I13	-
Q9.5	I14	-
Q9.6	I15	-
Q9.7	I16	-
Q10.0	I17	Program sel. bit 1
Q10.1	I18	Program sel. bit 2
Q10.2	I19	Program sel. bit 3
Q10.3	I20	Program sel. bit 4
Q10.4	I21	Program sel. bit 5
Q10.5	I22	Program sel. bit 6
Q10.6	I23	Program sel. bit 7
Q10.7	I24	Program sel. bit 8
Q11.0	I25	Program strobe
Q11.1	I26	Ext. unit play stop
Q11.2	I27	-
Q11.3	I28	-
Q11.4	I29	External reset
Q11.5	I30	Ext. play start
Q11.6	I31	MotorsON external
Q11.7	I32	MotorsOFF external

Table 3: PLC output to robot input

PLC	Robot	Name
I1.7	O16	-
I2.0	O17	-
I2.1	O18	-
I2.2	O19	Unit ready
I2.3	O20	Program ended
I2.4	O21	Error port
I2.5	O22	In playbk mode
I2.6	O23	Alarm port
I2.7	O24	Emergency stopped
I3.0	O25	In teach mode
I3.1	O26	Robot running
I3.2	O27	Ext. prg. sel. enabled
I3.3	O28	Ext. start enabled
I3.4	O29	Motors energized
I3.5	O30	State output
I3.6	O31	Home position
I3.7	O32	Information port

Table 4: Robot output to PLC input

7.4 Touch panel

The touch panel has three ways to connect to the PLC:

1. RS-232
2. RS-422/485
3. Ethernet

I have selected to use Ethernet to connect to PLC via the Simatic Net CP module. The CP module has been given IP address 192.168.0.3 while the touch panel has been given 192.168.0.4.

7.5 Pneumatic air system

Both the robot gripper and the clamps on CNC are driven by air pressure. In the robot there are 3 double solenoid valves that supply air output 1-6 [39]. These are controlled from the robot and commands to them must be given in the robot program.

For the CNC clamps there are 3 single solenoid valves that supply air. These are controlled from the PLC. Each solenoid valve has two wires. The solid black wire is connected to 0v while the black and red wire is connected to a PLC output. The connections can be seen in table 5. There are individual solenoids for closing the clamps so they can operate with a time difference. The clamps should always open simultaneously however, so this is controlled by a common solenoid for both clamps.

PLC	Solenoid
Q12.3	Clamp 1 close
Q12.4	Clamp 2 close
Q12.5	Clamps open

Table 5: Connections between PLC and air solenoids

8 Programming

8.1 PLC

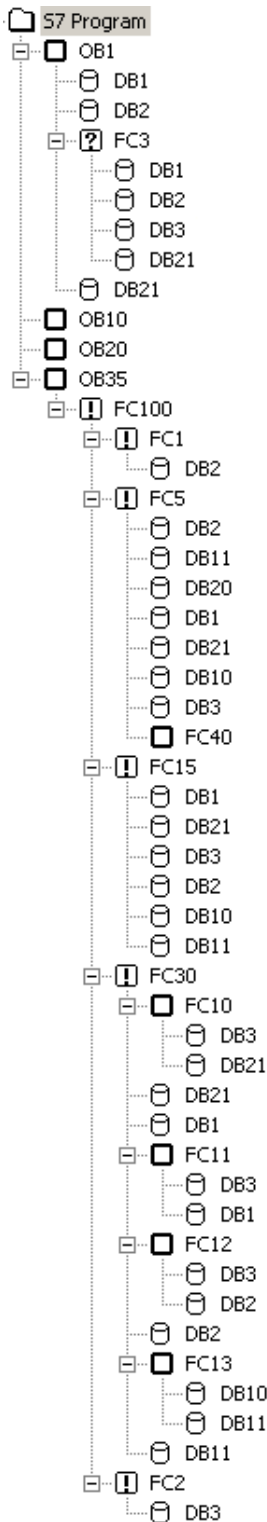


Figure 24: Structure

The PLC is the master of the entire FMS test bench. It communicates with all other units and gives them commands when necessary. A PLC program can be built by several different types of blocks. In my program I have used three different blocks:

1. OB - Organisation Block - Used to control execution of code
2. FC - Function - Used to program logic
3. DB - Data Block - Used to store variables

By standard, the only code that is executed on the PLC are the organisation blocks. In my program there are only two OBs in use, OB1 and OB35. As mentioned in section 7.2, OB1 executes its code as fast as possible with a minimum of 1ms. OB35 is a cyclic interrupt OB, meaning it will execute at predefined intervals. I have configured it to run every 10ms. All other code is called from these two OBs as seen in the program structure in figure 24.

In OB1 I first generate an "always true" and an "always false" signal that can be used to control other logic. It is imperative that these are correct, otherwise the program may malfunction. Next I call FC3 that will initialise variables either on PLC start up to make sure nothing unexpected happens, or when requested by the user via the touch panel. And that is it. Since OB1 executes as fast as possible there should not be too much code there, only the most critical.

From OB35 I call FC100 "ProgramCycle". And then from there, the main blocks in program are called. First out is FC1 "InputMapping" where all digital inputs are read from the modules and stored in DB2 "InputDB". Next is FC5 "SignalMapping" that handle internal signal routing between all connected units. It is followed by FC15 "Sequene" where the automatic control of the machines are coded. Next is FC30 "ManualControl" that allows the operator to test and functions individually. Finally FC2 "OutputMapping" is called that take all values from DB3 "OutputDB" and writes them to the output modules.

The reasoning behind this order is that first you must know which signals are inputted to the system. Then you see if the FMS should run in sequence as this has priority over manual control. In fact, if the FMS is in sequence, the manual control is disabled and we must know this before coming to the manual control in the program. At

the end, when the logic has changed whatever values it should change, we write it to the outputs.

The sequence code is the most complicated part of the PLC program and here is how it works:

1. If FMS is ready for sequence and Start button is pressed, the sequence starts
2. Open both clamps by setting open solenoid on and the two close solenoids off
3. Read move in program number from touch panel and send it to the robot.
4. Send reset program and start motors to the robot and wait two seconds.
5. Send start signal robot and wait additional two seconds.
6. Reset all signals sent to the robot and wait for robot program ended signal.
7. Close the clamps with the user defined delay from the touch panel.
8. Set CNC execute signal until CNC running is received. Wait until CNC is parked.
9. Open clamps again, same as in step 2.
10. Run robot program for moving out, same as step in step 3-6
11. Set conveyor speed output 1 and 2 based on user setting. Reset signals on timer.
12. End sequence and reset all signals.

The sequence can be run either continuously, or step by step. If continuous mode is selected the entire sequence will execute without any input from the user after sequence is started. Continuous mode can be turned on or off at any time during the sequence. If continuous mode is not selected, the sequence will halt in between the different operations and wait for the user to press "Next step" on the touch panel. The sequence will halt at these points:

- After clamps open, before robot move in starts
- After robot has put cube on CNC, before clamps close
- After clamps have closed, before CNC program starts
- After CNC program is finished, before clamps open
- After clamps open, before robot move out starts
- After robot has put cube on conveyor, before conveyor starts

The full code can be seen in appendix G - 315 PLC code.

8.2 CNC

A CNC machine gets its instructions in form of g-code [40]. In g-code you specify step by step where the machine should go and how it should go there. For my project I have made 10 different g-code files with different numbering. The code numbers and what they do can be seen in table 6. The full code can be seen in appendix D - G-code.

Code	Name	Functionality
0	Fail safe	Moves X axis 0.1mm back and forth
1	X axis check	Makes several cuts with various X positions
2	Y axis check	Makes several cuts with various Y positions
3	Mickey mouse	Cuts out a mickey mouse figure
4	Flyover	Moving CNC rapidly over cube without cutting
90	Parking	Parks the CNC machine away from the clamps
91	Z axis reset	Drives the Z axis to positive end stop
92	X axis reset	Drives the X axis to positive end stop
93	Y axis reset	Drives the Y axis to negative end stop
94	Zero position	Drives to edge of clamps for zeroing axes

Table 6: List of g-code programs and their functionality

As mentioned in section 7.1, I use python to control the CNC Mach3 software. Python is an object orientated programming language and most of the code is therefore used for defining objects which are later called. Following is a description of each object and what it does.

checkResetAlarm Starts by trying to reset the alarm by sending keystroke Ctrl+Alt+R. It then records 10 instances of a pixel on the alarm during 1 second with 0.1 second intervals. It then compares the 10 instances, if they are different from each other it means the alarm is active. It will then try to reset the alarm, make a new recording and compare again. If there is still a difference, the alarm could not be reset and it returns "True". If the comparison at any point shows that all 10 instances are equal to each other it means the alarm is not active and it returns "False".

checkAlarm Records 10 instances of a pixel on the alarm during 1 second with 0.1 second intervals. It then compares the 10 instances, if they are different from each other it means the alarm is active and it returns "True". If the instances are equal to each other it means the alarm is not active and it returns "False".

resetAlarm Trying to reset the alarm by sending keystroke Ctrl+Alt+R.

zeroAll Resets X, Y and Z axis measurement to zero by moving the mouse and clicking on the respective buttons.

loadCode This object must be called with an integer variable. It clicks on "Load G-Code" button and writes "C:\FMS\codeXX.txt" where the XX is the integer value. It then checks the on "Open as read-only" check mark and clicks open.

runCode First it records the pixel colour for for cycle not running. Then it clicks on "Rewind" button and enter keystroke Alt-R to execute G-code. It waits until the running pixel changes colour and enters a loop until it changes colour back again. While in the loop it checks several signals. If abort button is pressed it sends keystroke Alt-S and returns "False".

If the alarm is active it returns "False". There is also code for pausing the CNC, however, during testing it was discovered that Mach3 is not able to resume incremental steps when pausing. A pause functionality is therefore not implemented. If the loop is finished without any interruptions it returns "True".

startCode Clicks on "Rewind" button and enter keystroke Alt-R to execute G-code.

opcRead... There are several opcRead objects. They each read a specific signal in the OPC. If the quality is good it returns the value of the signal. Otherwise it will return the safest signal which is "False" except for abort where it is "True" and program number is "0".

opcWrite... These objects must be sent a "True" or "False" which is then written to respective OPC signal.

With all the objects defined the main program is quite short. First it tries to take control over the Mach3 software. If it is unable to do so, it is because it is not running. It will therefore start the software and wait. In both cases it will maximise the window as this imperative in order to read the correct pixels. It will check the alarm status with checkResetAlarm and then enter the main loop.

In the loop it checks for three different signals; Execute, park and initialize. For execute it loads the user defined program with loadCode, sets parked signal in OPC to "False" and runs the CNC program with runCode. If runCode returns "True" it will load the parking code (90) and if this as well returns "True" it will set the parked signal to "True"

If the park signal is received it will load the parking code (90) and run it. If runCode returns "True" the parked signal is set to "True"

Finally, if the initialise signal is received it will first set initialised and parked signal to "False". It will then load the Z axis reset code (91) and run it. It will reset the alarm and continue if the abort signal is "False". Then it does the same for X axis (92) and Y axis (93). Once all three axes are at the end positions it loads and runs the zero position code (94). After the code has executed successfully it will call the zeroAll object to reset all axes to zero and then set the initialised signal so "True". Finally it will load and run the parking code (90) and set parked signal to "True" if executed successfully.

8.3 Touch panel

The touch panel has been programmed in WinCC flexible 2008. I linked the project in WinCC to the PLC program in Step 7. This ensures that all changes done in Step 7 are synced over to WinCC. I created a two unit project, the first being the PC and second being the touch screen. When creating a project you may opt for various standard screens and functions. I opted for all of them and explored the functions. Most were not needed for my project so I manually removed them.

For the touch panel I created three screens with various buttons, inputs and status lights. A full description of the touch panel screens and their functions can be seen in appendix A - User manual. Each button had to be manually linked to a variable in a DB in the PLC. Most buttons are only active when you press them, but the continuous mode button acts as an on/off switch. I designed graphics for the CNC program selection buttons so the user can see what the end result will look like.

To get the sequence status lights I used an integer value that is set based on which step the sequence is on and if it is halted. This allowed me to create green lights as the step is executing and yellow blinking light when the step is halted. I have also used variables to alter the state and availability of the buttons. The start button for example is only available when all prerequisite conditions are met. To make it easier for the users to find out what is needed I have made status lights for all the starting conditions. This makes the system intuitive and easy to understand. I have also put in limitations on variable input fields so the users are not able to enter illegal values. Since there are 8 digital signals used to send the program number from the PLC to the robot it means the highest robot program number can be $2^8 - 1 = 255$.

8.4 Robot

The robot can be programmed either directly on the teach pendant or using an offline programming software like Nachi FD on Desk [41]. I chose the former option since I already had the teach pendant, but not the software. With the teach pendant you can manually drive the robot to the exact locations you want it to go and store them as way points. When storing a way point you must select how accurate the robot should be at that point on a scale from 1 to 8. The number defines how far from the stored way point the robot can create its path. For the Nachi MZ07, the settings number and deviation can be seen in table 7. A higher deviation means a smoother path can be created and less stress on the robot and its support structure.

Accuracy	Distance
1	0mm
2	5mm
3	10mm
4	25mm
5	50mm
6	100mm
7	200mm
8	500mm

Table 7: Accuracy setting and corresponding allowed deviation from way point [42]

You are also other variables that must be selected when storing a way like how the robot should move there, at what speed, and which tool it is using. Before I started programming the robot I found a suitable parked position that could act as starting and ending position for all operations. With all this in mind I created three different robot programs which can all be seen in appendix E.

The first is robot program number 200 which is for returning the robot to the parked position. It first opens the gripper and then drives slowly to parked position. The reason for driving slowly is because this program should be used after aborting a sequence or manually driving the robot. If the user detects that it is on collision course they will have more time to stop it may be less severe in low speed.

The parking program also acts as a base for the other two programs, so we always know the starting position. Program 201 is called "move in" and goes to the pick up position, closes the gripper and then moves to the CNC clamps via the parked position. It opens the gripper and returns to parked position.

The third and final robot program I made is 202 "move out". This again starts at the parked position, then moves to the CNC and closes the gripper. It will lift up and over to the conveyor via the parked position. At the conveyor it will open gripper and return to parked position.

9 Conclusion

I have ended up with a fully working test bench using the available equipment at the university as seen in figure 25. It has the three basic building blocks of an FMS; The CNC is the workstation, the robot and conveyor is the handling and transport system, and the 315 PLC is the control system. As regards to the seven types of flexibility it fulfils several of them:

1. *Production flexibility* The CNC can produce any pattern and the setup can handle different size of cubes.
2. *Expansion flexibility* There are available space within the reach of the robot and the 222 PLC is already programmed for adding another stepper motor.
3. *Machine flexibility* The CNC can perform milling operations, but not much else.
4. *Product flexibility* It is very easy to make new designs. Just add the g-code and select it from the touch panel.
5. *Routing flexibility* There is really only one route through the system, but the robot program can easily be changed and selected from the touch panel.
6. *Volume flexibility* Volume can be varied, however, the profitability is not applicable here.
7. *Mix flexibility* The system can produce any design the CNC can handle without affecting quality.

For a small scale test bench it is able to demonstrate many forms of flexibility. It incorporates many of the learning goals that I set at the beginning like being able to run custom g-code, use lean six sigma to measure tolerances, and poka yoke through interlocks.

9.1 Further improvements

With a working test bench there are many possibilities for improvements and add-ons. It could be possible to build a feeding system for the cubes so you can run several sequences in a row without human intervention. The elevator can be connected to the PLC so the finished cubes are delivered in different boxes. You could use sensors to stop the conveyor instead of a timer. There is already a sensor mounted between the belts that may be used. The free space next to the CNC could be used for either another cutting machine, or a buffer system. The flexibility of the robot really opens up a lot of possibilities for improvements.

Whether the test bench is expanded upon or used as it is, I hope it will inspire future students and spark curiosity around flexible automation.



Figure 25: The completed flexible manufacturing system test bench

References

- [1] Eberhard Abele et al. 'Learning factories for future oriented research and education in manufacturing'. In: *CIRP Annals* 66.2 (2017), pp. 803–826.
- [2] Springfield Republican. *Death of a Distinguished Inventor*. 1864. URL: <https://www.nytimes.com/1864/04/24/archives/death-of-a-distinguished-inventor.html> (visited on 26/10/2018).
- [3] Jessica Jenkins. *Christopher Miner Spencer, 19th-century Arms Manufacturer*. 2012. URL: <https://connecticuthistory.org/christopher-miner-spencer-successful-19th-c-arms-manufacturer/> (visited on 26/10/2018).
- [4] Russ Olexa. *The Father of the Second Industrial Revolution*. 2001. URL: <http://www.sme.org/Tertiary.aspx?id=36002&terms=father%20of%20the%20second%20industrial%20revolution> (visited on 27/10/2018).
- [5] *US patent 4621410 - Williamson System 24 CNC machine shop*. 2018. URL: https://commons.wikimedia.org/wiki/File:US_patent_4621410_-_Williamson_System_24_CNC_machine_shop.png (visited on 14/12/2018).
- [6] *UNIMATE - The First Industrial Robot*. 2018. URL: <https://www.robotics.org/joseph-engelberger/unimate.cfm> (visited on 07/11/2018).
- [7] Mikell P Groover. *Automation, production systems, and computer-integrated manufacturing*. 2nd ed. Upper Saddle River, N.J. : Prentice Hall ; London : Prentice-Hall International, 2001, p. 461. ISBN: 0130889784.
- [8] Alison Dunn. *The father of invention: Dick Morley looks back on the 40th anniversary of the PLC*. 2008. URL: <https://www.automationmag.com/features/the-father-of-invention-dick-morley-looks-back-on-the-40th-anniversary-of-the-plc.html> (visited on 31/10/2018).
- [9] Business Wire. *Technavio Announces Top Five Vendors in the Global Micro PLC Market from 2016 to 2020*. 2016. URL: <https://www.businesswire.com/news/home/20160502005493/en/Technavio-Announces-Top-Vendors-Global-Micro-PLC> (visited on 31/10/2018).
- [10] International Federation of Robotics. *Robots double worldwide by 2020*. 2018. URL: <https://ifr.org/ifr-press-releases/news/robots-double-worldwide-by-2020> (visited on 07/11/2018).
- [11] Abdul Montaqim. *Top 14 industrial robot companies and how many robots they have around the world*. 2015. URL: <http://roboticsandautomationnews.com/2015/07/21/top-8-industrial-robot-companies-and-how-many-robots-they-have-around-the-world/812/> (visited on 07/11/2018).
- [12] H.K. Shivanand, M.M. Benal and V. Koti. *Flexible Manufacturing System*. New Age International, 2006. ISBN: 9788122425598.
- [13] 'Flexible Manufacturing Systems'. In: *CIRP Encyclopedia of Production Engineering*. Ed. by Luc Laperrière and Gunther Reinhart. Berlin, Heidelberg: Springer Berlin Heidelberg, 2014, pp. 524–530. ISBN: 978-3-642-20617-7.

- [14] Mikell P Groover. *Fundamentals of modern manufacturing : materials, processes, and systems*. 5th ed. Englewood Cliffs, NJ: Prentice Hall, 2012, p. 1018. ISBN: 978-1-118-231463.
- [15] Ville Toivonen et al. ‘The FMS Training Center - a versatile learning environment for engineering education’. In: *Procedia Manufacturing* 23 (2018). “Advanced Engineering Education & Training for Manufacturing Innovation” 8th CIRP Sponsored Conference on Learning Factories (CLF 2018), pp. 135–140.
- [16] *Festo Didactic – Technical Education Solutions*. 2018. URL: <https://www.festo-didactic.com/int-en/company/?fbid=aW50LmVuLjU1Ny4xNy4xMC4zNDQ0LjQxNDE> (visited on 11/12/2018).
- [17] *Festo Didactic – MicroFMS*. 2018. URL: <https://www.festo-didactic.com/ov3/media/customers/1100/00987874001075223761.pdf> (visited on 11/12/2018).
- [18] *MEC: Lab Equipment*. URL: <http://www.mechatronics-mec.org/igsbase/igstemplate.cfm?SRC=DB&SRCN=&GnavID=7&SnavID=5> (visited on 17/12/2018).
- [19] Sarah Reynard et al. ‘Flexible manufacturing cell SCADA system for educational purposes’. In: *Computer Applications in Engineering Education* 16.1 (2008), pp. 21–30.
- [20] Chris A. Ortiz and Murry R. Park. *Visual Controls: Applying Visual Management to the Factory*. Productivity Press, 2010. ISBN: 9781439820902.
- [21] *Signal towers*. 2018. URL: <http://www.wernerelektrik.com/signal-tower-signal-light/> (visited on 03/12/2018).
- [22] *What is the Poka Yoke Technique?* 2018. URL: <https://kanbanize.com/lean-management/improvement/what-is-poka-yoke/> (visited on 03/12/2018).
- [23] ISO 8373:2012(E). *Robots and robotic devices — Vocabulary*. Standard. Geneva, CH: International Organization for Standardization, 2012.
- [24] *Nachi MZ07 - Specifications*. 2016. URL: <http://www.nachi-fujikoshi.co.jp/eng/rob/hand/mz07c.htm> (visited on 08/11/2018).
- [25] *Nachi MZ07 - CFD Controller*. 2016. URL: <http://www.nachi-fujikoshi.co.jp/eng/rob/contr/cfd.a.htm> (visited on 13/11/2018).
- [26] Tony Atkins and Marcel Escudier. ‘Programmable Logic Controller’. In: *A Dictionary of Mechanical Engineering*. 1st ed. Oxford University Press, 2013. ISBN: 9780199587438.
- [27] *S7-300 CPU 31xC and CPU 31x: Technical specifications*. Siemens. Mar. 2011. 420 pp. URL: https://cache.industry.siemens.com/dl/files/906/12996906/att_70325/v1/s7300_cpu_31xc_and_cpu_31x_manual_en-US.en-US.pdf (visited on 01/11/2018).
- [28] *SIMATIC HMI MP 370*. Siemens. Mar. 2004. 254 pp. URL: https://cache.industry.siemens.com/dl/files/667/19106667/att_75795/v1/BA_Bediengeraet_MP_370.e.pdf (visited on 19/12/2018).
- [29] *JY5300 CNC Controller & MHC2 handwheel user guide v3.0*. MIB Instruments Ltd. July 2013. 41 pp. URL: https://www.casogo.com/YSP/JY5300_V3_User_guide.pdf (visited on 26/12/2018).

- [30] *Visual Components - A world leader in 3D simulation*. 2019. URL: <https://www.visualcomponents.com/about-us/> (visited on 28/05/2019).
- [31] *What is OPC?* URL: <https://opcfoundation.org/about/what-is-opc/> (visited on 02/04/2019).
- [32] *What is Python? Executive Summary*. URL: <https://www.python.org/doc/essays/blurb/> (visited on 03/04/2019).
- [33] *Drivers for 2-phase stepper motors*. URL: http://middex.de/en/motor_schrittmotoren.php (visited on 14/05/2019).
- [34] *Positioning Module FM 353*. URL: <https://w3.siemens.com/mcms/programmable-logic-controller/en/advanced-controller/s7-300/function-modules/fm353/pages/default.aspx> (visited on 15/05/2019).
- [35] Siemens. *S7-200 Programmable Controller System Manual*. Aug. 2005, p. 17. 534 pp. URL: http://www1.siemens.cz/ad/current/content/data_files/automatizacni_systemy/mikrosystemy/simatic_s7200/manual_s7_200_2005_en.pdf (visited on 15/05/2019).
- [36] Danielle Collins. *What is microstepping?* 2017. URL: <https://www.linearmotiontips.com/microstepping-basics/> (visited on 16/05/2019).
- [37] Nachi. *CFD controller technical document 2*. Nov. 2013. 178 pp.
- [38] *What is the difference between PNP and NPN when describing 3 wire connection of a sensor?* 2019. URL: <https://www.schneider-electric.co.uk/en/faqs/FA142566/> (visited on 17/05/2019).
- [39] Nachi. *CFD controller technical document 1*. June 2015. 180 pp.
- [40] *Getting Started with G-Code*. 2019. URL: <https://www.autodesk.com/industry/manufacturing/resources/manufacturing-engineer/g-code> (visited on 23/05/2019).
- [41] *FD On Desk Simulation Software*. 2019. URL: <http://www.nachirobotics.com/fd-controller/fd-on-desk/> (visited on 30/05/2019).
- [42] Nachi. *CFD controller instruction manual basic operations manual*. Aug. 2015, p. 78. 310 pp.

A User manual



Flexible Manufacturing System Test Bench

User manual

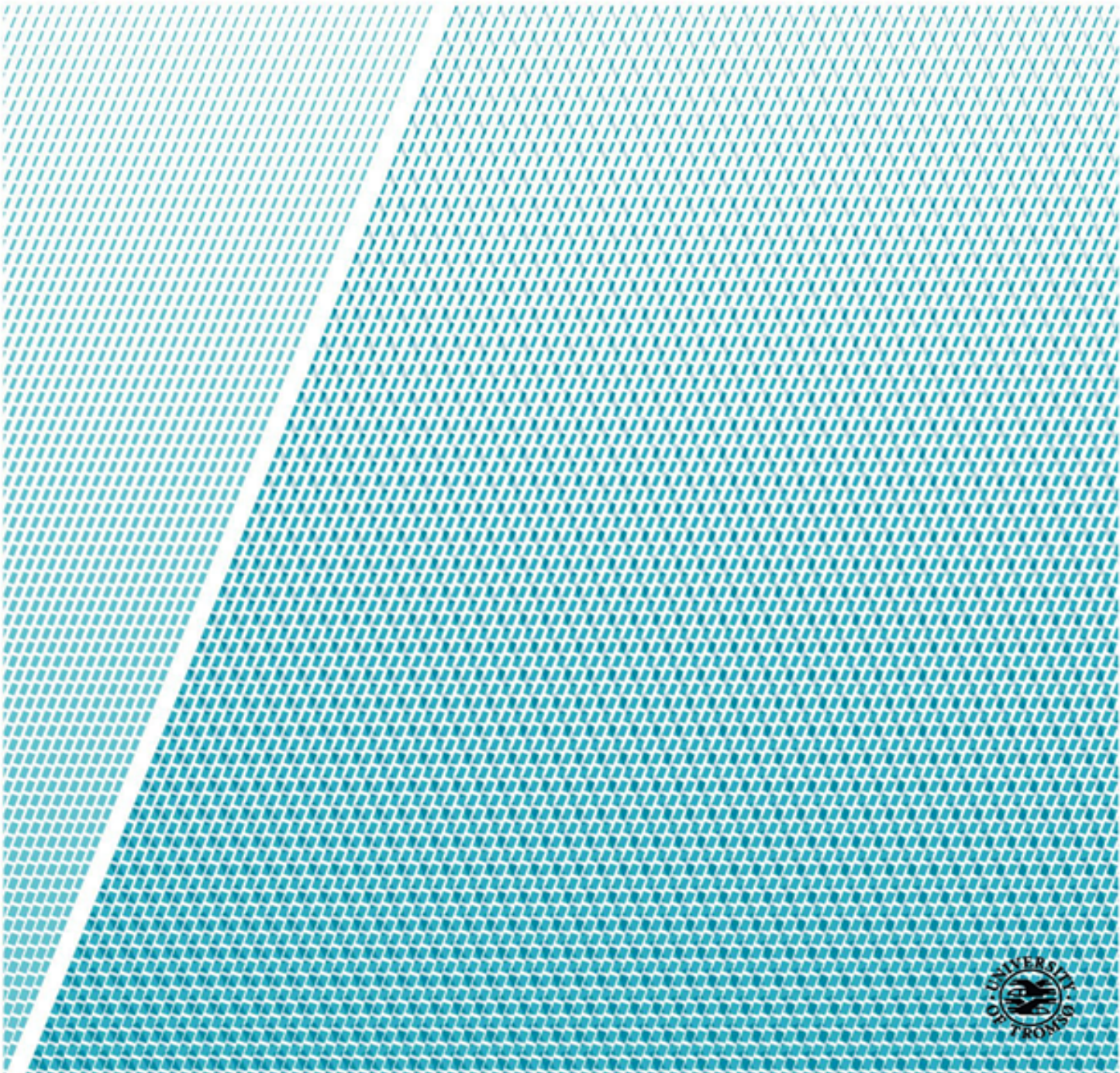


Table of Contents

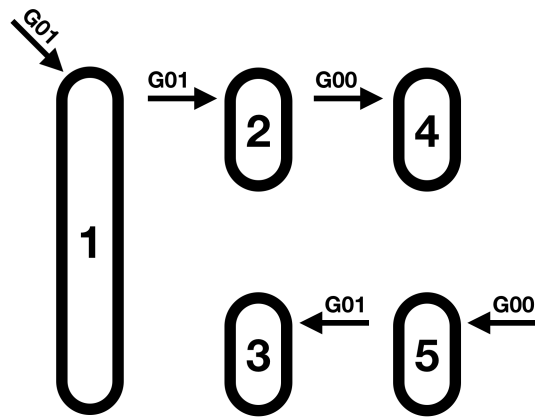
- 1 Purpose 1
- 2 Start up 2
- 3 Touch screen 3
 - 3.1 Start Screen 3
 - 3.2 Program Screen 4
 - 3.3 Settings Screen 5
 - 3.4 Service Screen 6
- 4 Run the test bench 7
 - 4.1 Start sequence 7
 - 4.2 Test functions 7
 - 4.3 Finish up 7
- 5 Customisation options 8
 - 5.1 Custom CNC program 8
 - 5.2 Custom robot program 8
 - 5.3 Custom clamp closing time 8
 - 5.4 Custom cube size 9
 - 5.5 Custom conveyor speed 9
- 6 Troubleshooting 10

1 Purpose

The flexible manufacturing system (FMS) test bench is made for demonstrational and educational purposes. The robot will take a wooden cube from the start position and place it in the CNC machine. The CNC will mill a design of your choosing before the robot will take the cube out and place it on the conveyor where it is transported out.

The system is set up to handle wooden cubes in sizes between $42 \times 42 \text{mm}$ and $58 \times 58 \text{mm}$. However, it is recommended to use a cube that is approximately $50 \times 50 \text{mm}$. The auto initialisation sequence will set zero Z at 50mm above the CNC base. It is therefore important that either the cube height is 50mm, or a manual adjustment of the Z height is done in Mach3. This must then be done after initialisation sequence is complete.

The FMS test bench comes with 3 pre-programmed CNC designs. The first two are called X-axis and Y-axis. These are for measuring inaccuracies in the CNC machine. It makes 5 cuts as shown in the figure to the right. The approach to the cut and g-code is also shown. The first cut is the baseline to measure the 4 other cuts against. Cut 2 and 3 are approached slowly from each side while cut 4 and 5 are approached fast from each side. With this setup you can measure the difference in accuracy when using G00 and G01 as well as the difference in accuracy based on the approach side.



The Y-axis program does the exact same thing, except it is turned 90° to the side. There is also two deeper cuts at the edges of the first cut on the Y-axis program so you can tell the finished blocks apart later.

The final program cuts out a mickey mouse figure, this is only for demonstrational purpose. The users can experiment with their own CNC and robot programs as well as adjusting conveyor speed and clamp delays. It is also possible to rearrange the clamps on the CNC and customise it as you want.

2 Start up

The FMS test bench has 4 different cabinets:



The test bench must be started in the following order (cabinet number in parenthesis):

1. Plug in power strip (4) - PLC and touch panel will power up
2. Power on robot controller (3)
3. Power on CNC controller (1)
4. Power on PC (2) - Wait until Mach3 is launched and alarm cleared automatically
5. Connect compressed air
6. Set robot controller in playback mode (3)
7. Set robot teach pendant in playback mode
8. Enable external program and start on robot teach pendant
9. Open Settings screen on touch panel - Press "Initialize CNC" - Wait for it to finish

For step 6, 7 and 8 it should look like this:

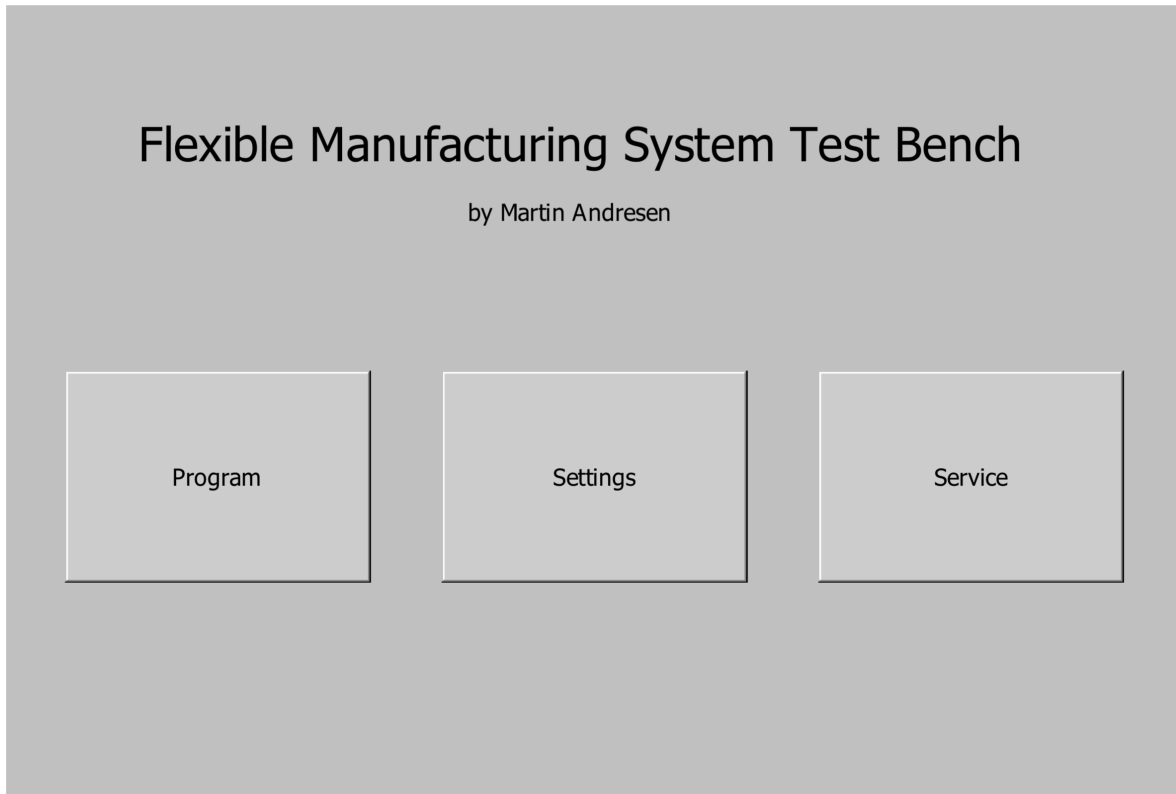


Warning!

Make sure there is no wooden cube on the CNC when initialising it as it may collide.

3 Touch screen

3.1 Start Screen

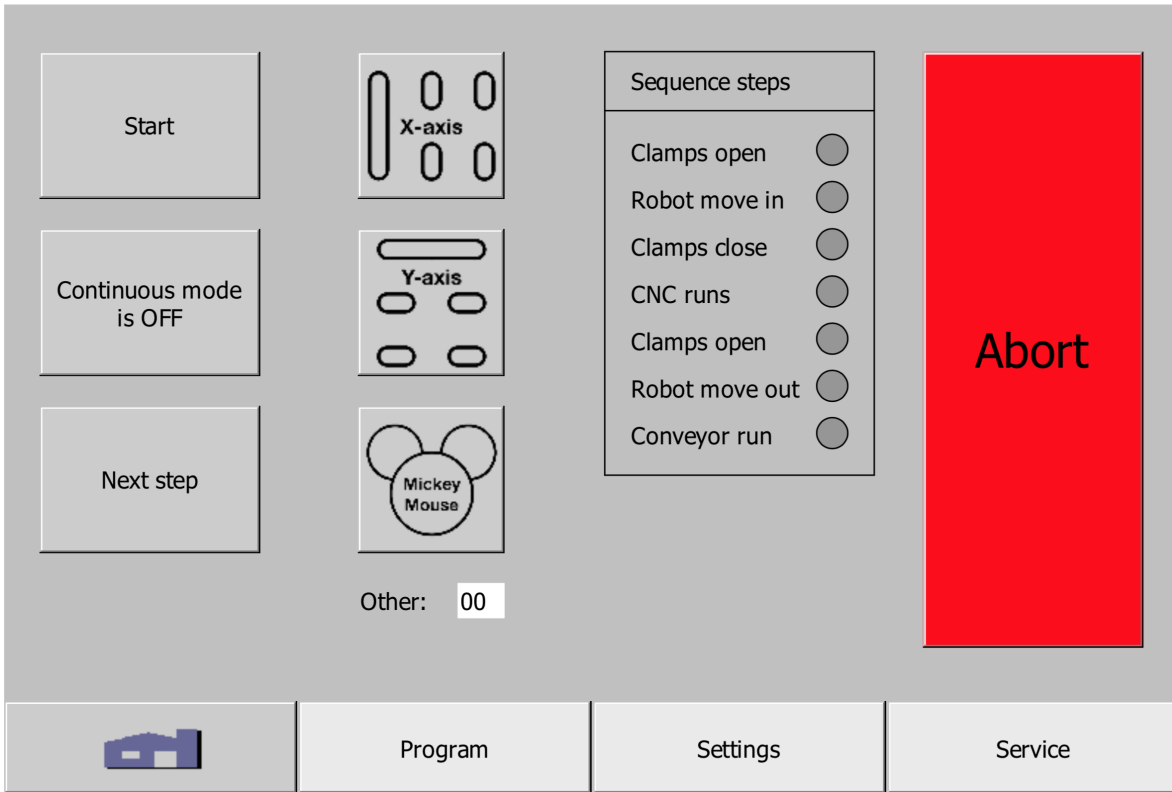


This screen is only for selecting other screens

Program	Go to program screen
Settings	Go to settings screen
Service	Go to service screen

These buttons can be found on the bottom of all other screen along with a home button that will lead back to this screen

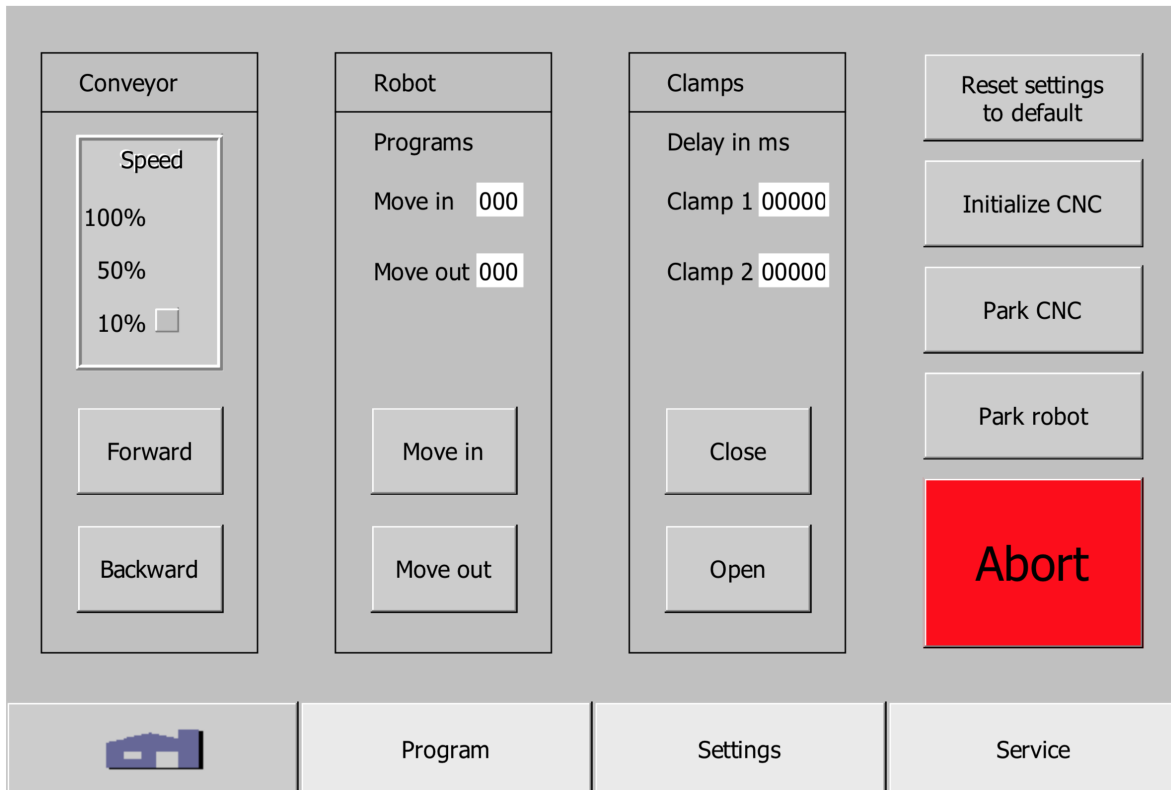
3.2 Program Screen



This screen is for selecting CNC program and running FMS in sequence

Start	Start sequence with selected program
Continuous	Only available if all lights in Service screen are green Select this to make the sequence execute non-stop Can be selected and deselected at any time Will become green when it is active
Next step	Press to proceed to next step when sequence is halted Will become yellow sequence is halted and button is available
X-axis	Select X-axis check CNC program
Y-axis	Select Y-axis check CNC program
Mickey Mouse	Select Mickey Mouse CNC program
Other	Select a custom CNC program [1-89]
Sequence steps	Indicates which sequence step is active Will blink yellow if sequence is halted
Abort	Aborts the sequence and stops all machines

3.3 Settings Screen

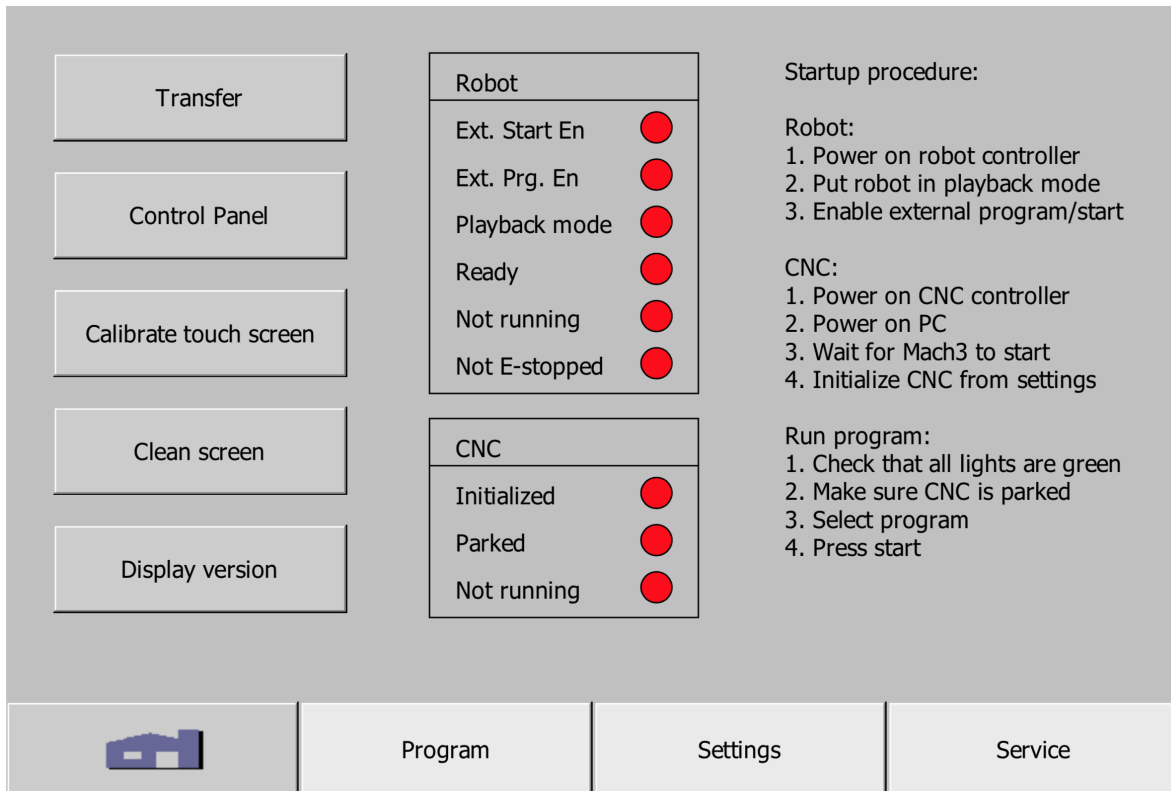


This screen is for selecting settings and testing single functions

Speed	Select conveyor speed
Forward	Run conveyor forward
Backward	Run conveyor backward
Move in input	Select robot program for move in [1-255]
Move out input	Select robot program for move out [1-255]
Move in button	Test run move in robot program
Move out button	Test run move out robot program
Clamp 1	Time delay for clamp 1 to close in milliseconds [0-10000]
Clamp 2	Time delay for clamp 2 to close in milliseconds [0-10000]
Close	Close clamps
Open	Open clamps
Reset	Reset all settings to default value
Initialize CNC	Drive CNC to all end stops and zero axes at clamps
Park CNC	Drive CNC to parked position
Park robot	Drive robot to parked position
Abort	Aborts the sequence and stops all machines

All functions except Abort are disabled when FMS is in sequence

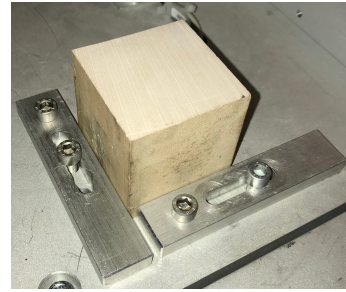
3.4 Service Screen



This screen is for service functions and start condition checks

Transfer	Transfer new software to the touch panel
Control panel	Open the Windows control panel
Calibrate	Calibrate the touch screen
Clean screen	Disables the touch screen for 15 seconds so you can clean it
Display version	Displays the installed software and their versions
Robot	Displays the sequence start conditions for the robot
CNC	Displays the sequence start conditions for the CNC

4 Run the test bench



4.1 Start sequence

1. Place a wooden cube in the starting position →
2. Open Service screen and check that all lights are green
3. Open Settings screen and set your preferred settings
4. Open Program screen and select the CNC program you would like to run
5. Select Continuous mode if you would like the sequence to execute non-stop
6. Press start button to start sequence
7. If not in continuous mode, press Next step when it is yellow to proceed



Warning!

If you have manually run the CNC with the hand wheel pendant you must initialise the CNC by pressing the "Initialize CNC" button in Settings screen.



Warning!

If you have manually run the CNC with Mach3 you must park the CNC by pressing the "Park CNC" in Settings screen.

4.2 Test functions

To test the various functions go to the Settings screen. There are buttons for testing conveyor, robot and clamps. For safety reasons, testing of a CNC program must be done from Mach3 on the PC.



Warning!

Only test one function at the time. The CNC and robot may collide.

4.3 Finish up

When leaving the FMS test bench it is recommended to run all machines to parked position. This is done automatically in the sequence. If manual functions have been used, or sequence aborted, you may use the Park CNC and Park robot buttons in Settings screen. The clamps should be left in open position.

5 Customisation options

5.1 Custom CNC program

In order to create and run a custom CNC program you must make a g-code file and save it as "codeXX.txt" in the C:\FMS folder on the computer. The XX must be a number between 5 and 89. Do not use trailing zero before single digit numbers. Once the file is saved you can run the program by selecting the same number in the Program screen before starting the sequence.

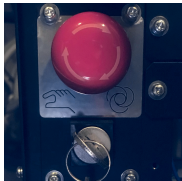


Warning!

Do not delete code 0-4 or code 90-94 from the FMS folder.

5.2 Custom robot program

In order to create and run a custom robot program you must first switch the robot controller and teach pendant to teach mode. Then you must set the program and start to internal on the teach pendant. It should look like this:



You must then find an empty program number between 1 and 255 and make your program. Once the robot program has been made and saved it can be selected in the Settings screen. You must remember to put the robot back in playback mode and enable external program and start again.



Warning!

Do not delete robot code 200-202.

5.3 Custom clamp closing time

The clamps will close with the delay set in the Settings screen. The time is given in milliseconds. 1000 milliseconds = 1 second. The clamps can be given a delay from 0 to 10000ms. Default delay is 0ms for clamp 1 and 1000ms for clamp 2. The clamp on order can be changed by having a longer delay for clamp 1 than clamp 2.

5.4 Custom cube size

By default, the robot gripper and the CNC clamps are set up for cube sizes between 42×42 mm and 58×58 mm. Both the gripper and clamps can be easily adjusted with a hex key to accommodate different sizes. It is not recommended to put the clamps closer together than standard as they may then hit each other.



Warning!

If the clamps position are changed you must update robot program 201 and 202, and CNC program 94 with the new position.

5.5 Custom conveyor speed

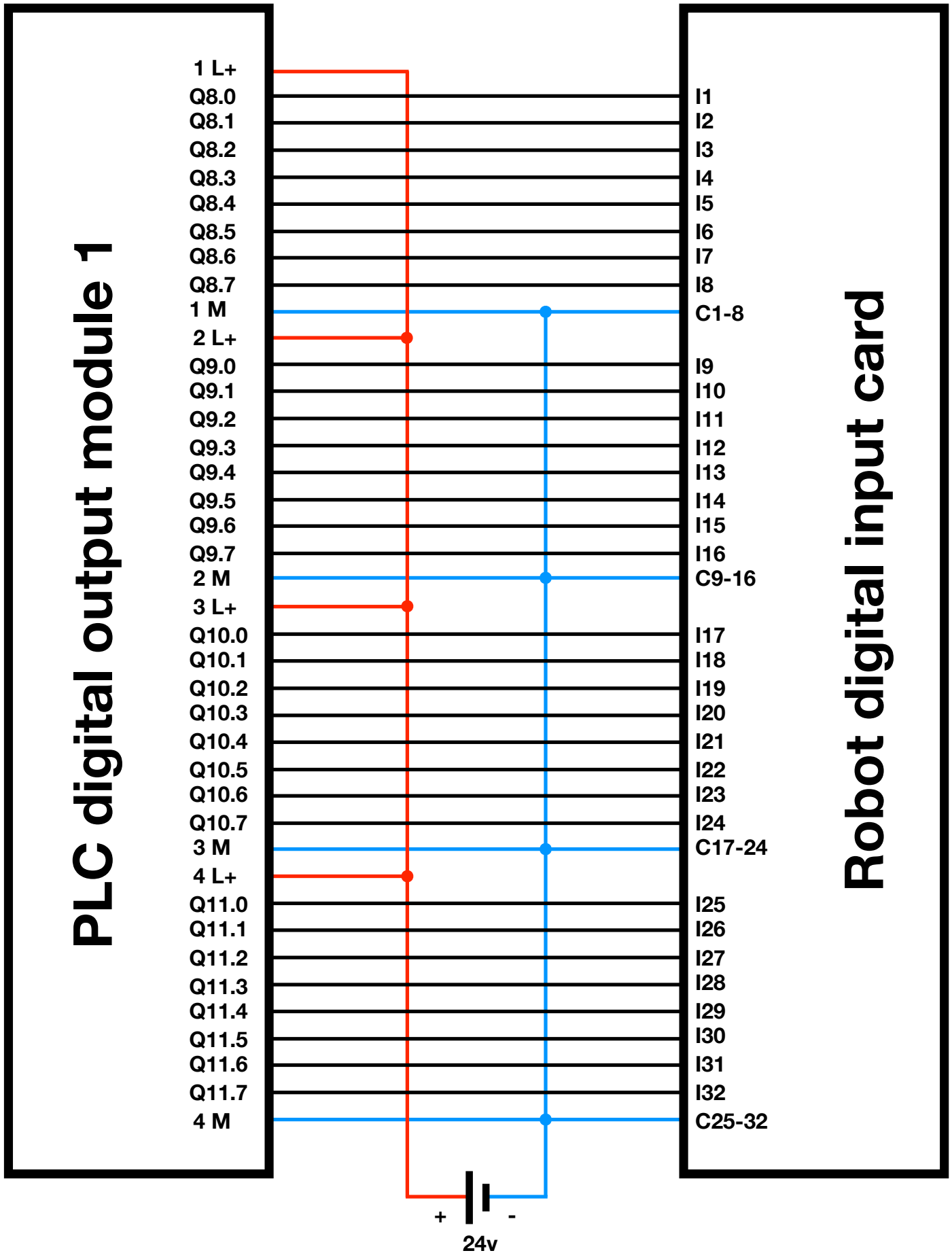
The conveyor has three different speed settings that will vary the frequency output to the stepper motor driver. A timer will stop the conveyor when the cube reaches the pick up point when running in sequence mode.

6 Troubleshooting

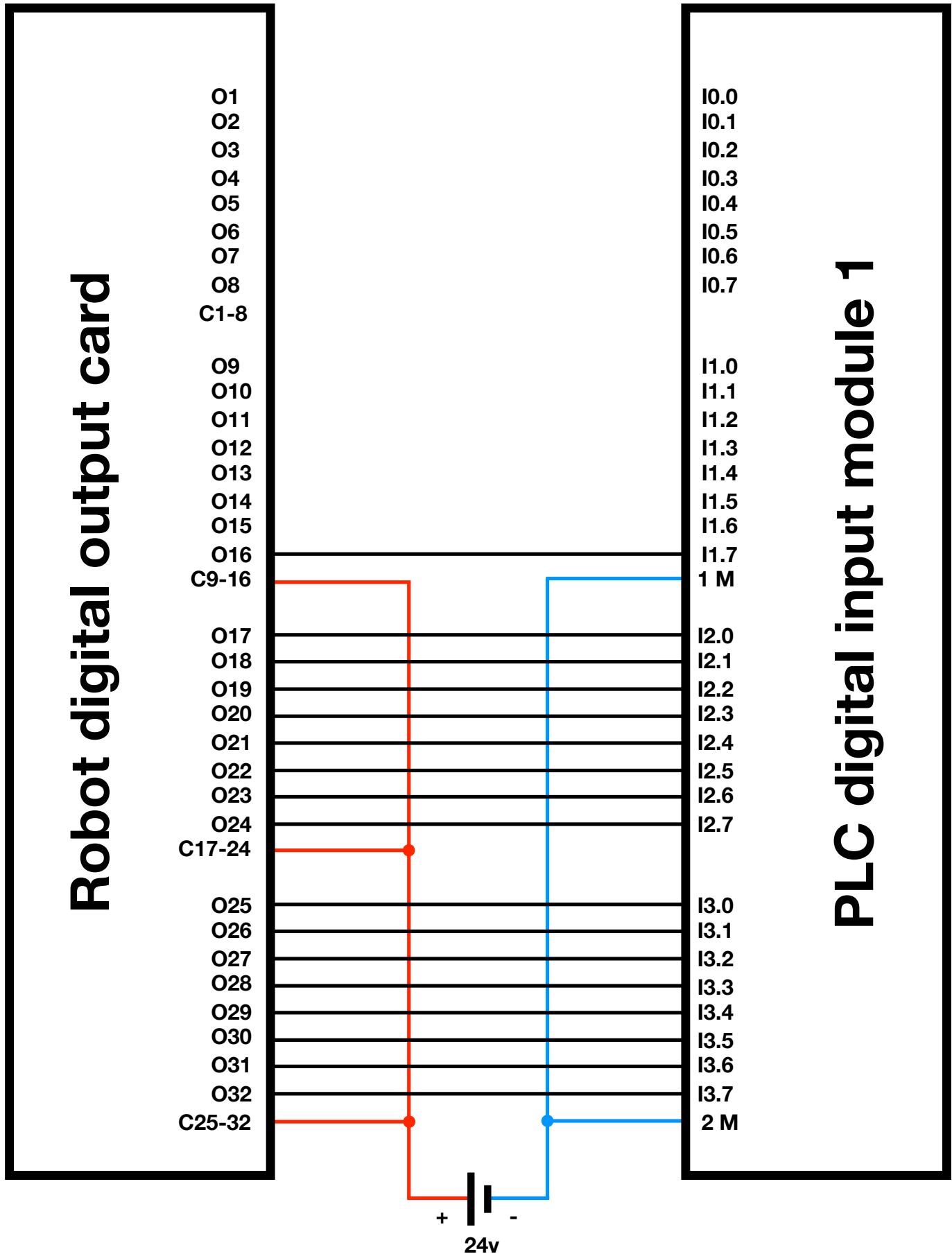
Fault	Remedy
CNC is not moving	<ul style="list-style-type: none">- Check start conditions on Service screen- Power on CNC controller- Power on PC- Release CNC controller e-stop- Release robot e-stop- Does the selected CNC program exist?- Check Ethernet cable- Is WinCC runtime running? - No: Click on "Start FMS"- Is Python script running? - No: Click on "Start Python"
Robot is not moving	<ul style="list-style-type: none">- Check start conditions on Service screen- Power on robot controller- Release robot e-stop- Set robot in playback mode- Enable external program and start- Does the selected robot program exist?- Check wiring
Clamps are not moving	<ul style="list-style-type: none">- Connect compressed air- Check clamp timer in Settings screen- Check wiring
Conveyor is not moving	<ul style="list-style-type: none">- Is 222 PLC in run mode?- Check wiring
Files are missing	All files are backed up in C:\FMS\Backup

B Electrical drawings

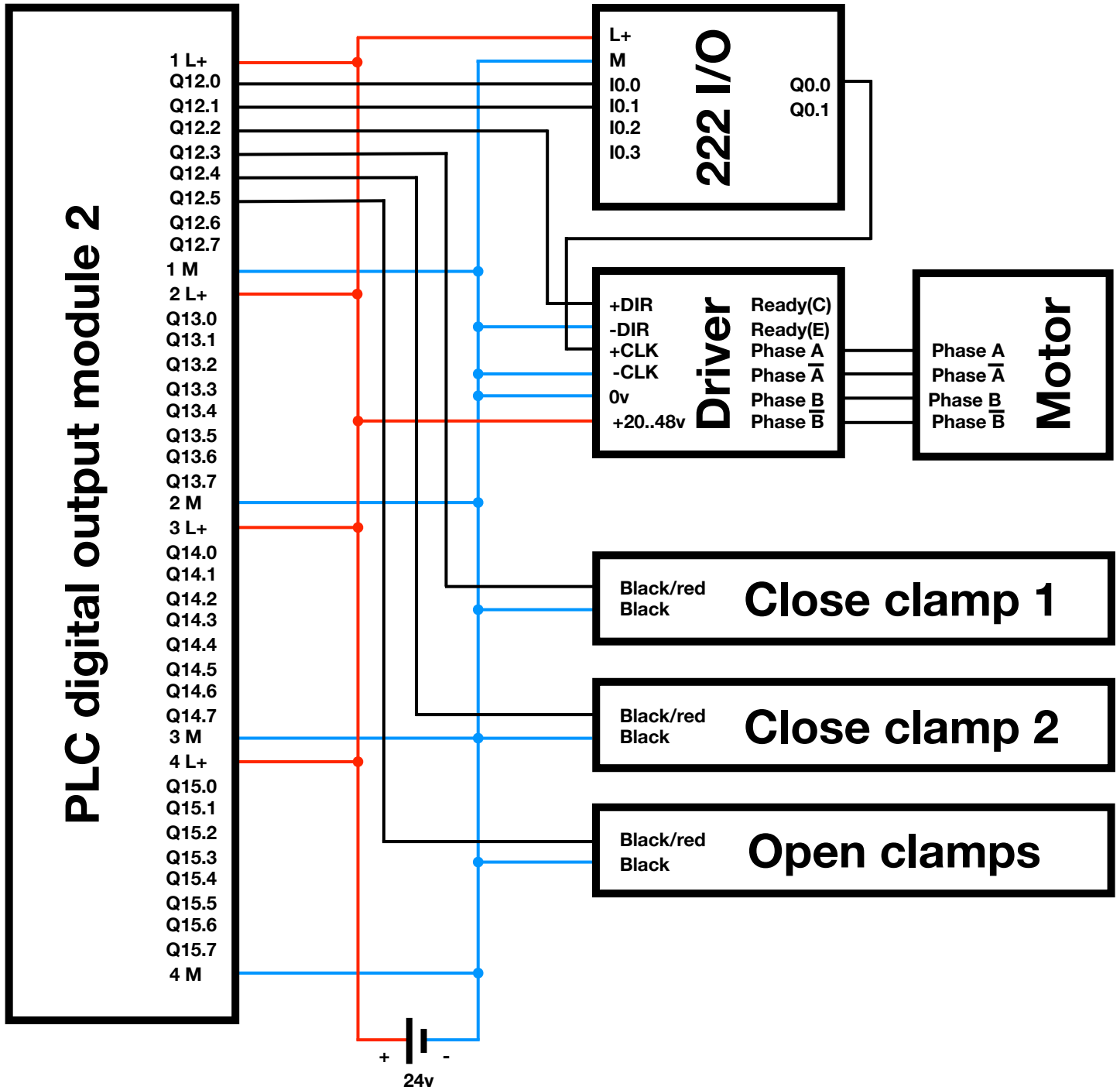
PLC output to robot input



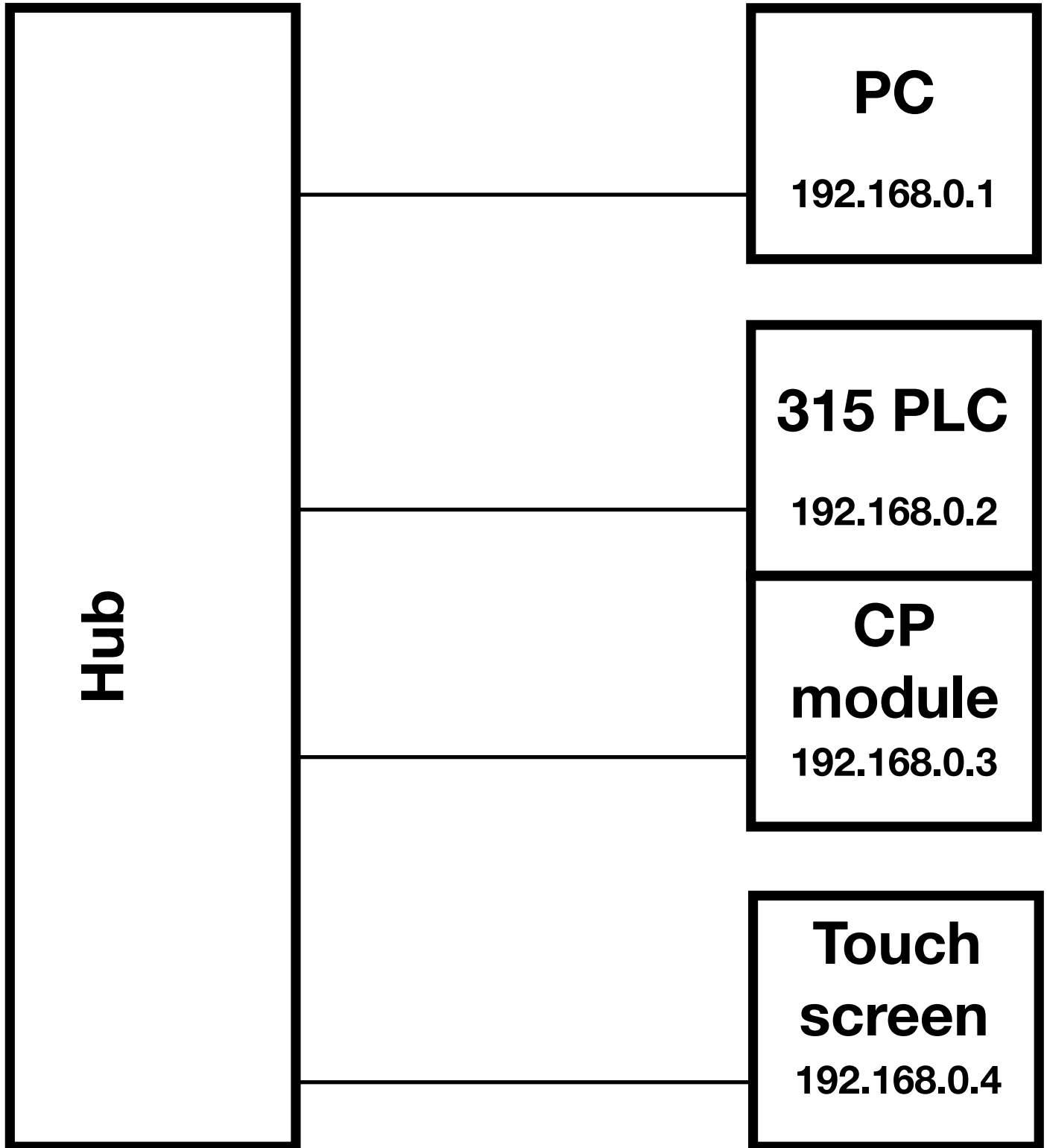
Robot output to PLC input



PLC output to stepper motor and air solenoids



Ethernet network topology



C Python code

```

import time
import OpenOPC
import win32gui
import pywinauto

Running = False
opc = OpenOPC.client()
opc.connect('OPC.SimaticHMI.HmiRTm.1')

def checkResetAlarm():
    app.Mach3CNCLicencedTo.type_keys('^%r')
    alarmList = []
    pywinauto.mouse.move(coords=(850,530))
    for x in range(10):
        alarmList.append(win32gui.GetPixel(win32gui.GetDC(win32gui.GetActivewindow()),100,868))
        time.sleep(0.1)
    if alarmList[1:] != alarmList[:-1]:
        app.Mach3CNCLicencedTo.type_keys('^%r')
        alarmList[:] = []
        pywinauto.mouse.move(coords=(850,530))
        for x in range(10):
            alarmList.append(win32gui.GetPixel(win32gui.GetDC(win32gui.GetActivewindow()),100,868))
            time.sleep(0.1)
            if alarmList[1:] != alarmList[:-1]:
                return True
        alarmList[:] = []
    return False

def checkAlarm():
    alarmList = []
    pywinauto.mouse.move(coords=(850,530))
    for x in range(10):
        alarmList.append(win32gui.GetPixel(win32gui.GetDC(win32gui.GetActivewindow()),100,868))
        time.sleep(0.1)
    if alarmList[1:] != alarmList[:-1]:
        return True
    else:
        return False

def resetAlarm():
    app.Mach3CNCLicencedTo.type_keys('^%r')

def zeroAll():
    pywinauto.mouse.click(button='left', coords=(735,140))
    pywinauto.mouse.click(button='left', coords=(735,220))
    pywinauto.mouse.click(button='left', coords=(735,290))
    pywinauto.mouse.move(coords=(850,530))

def loadCode(num):
    if opcReadPause() == False and opcReadAbort() == False:
        app.Mach3CNCLicencedTo.LoadGCode.click()
        app.Open.type_keys("C:\FMS\code")
        app.Open.type_keys(num)
        app.Open.type_keys(".txt")
        app.Open.OpenAsReadOnly.check()
        app.Open.Open.click()
        time.sleep(2)

def runCode():
    if opcReadPause() == False and opcReadAbort() == False:
        pywinauto.mouse.move(coords=(850,530))
        notRunning = win32gui.GetPixel(win32gui.GetDC(win32gui.GetActivewindow()),100,594)
        app.Mach3CNCLicencedTo.RewindCtrlW.click()
        app.Mach3CNCLicencedTo.type_keys('%r')
        Running = True
        opcWriteRunning(True)
        while (win32gui.GetPixel(win32gui.GetDC(win32gui.GetActivewindow()),100,594) ==
notRunning):

```

```

        time.sleep(0.1)
    while (win32gui.GetPixel(win32gui.GetDC(win32gui.GetActiveWindow()),100,594) !=
notRunning):
        pywinauto.mouse.move(coords=(850,530))
        if opcReadPause() == True:
            app.Mach3CNCLicensedTo.type_keys("{SPACE}")
            if opcReadPaused() == False:
                opcWritePaused(True)
                Running = False
                opcWriteRunning(False)
            elif opcReadPaused() == True:
                opcWritePaused(False)
                Running = True
                opcWriteRunning(True)
            while opcReadPause() == True:
                time.sleep(0.1)
        if opcReadAbort() == True:
            app.Mach3CNCLicensedTo.type_keys('%s')
            running = False
            return False
        if checkAlarm() == True:
            running = False
            return False
    Running = False
    opcWriteRunning(False)
    return True

def startCode():
    app.Mach3CNCLicensedTo.RewindCtrlw.click()
    app.Mach3CNCLicensedTo.type_keys('%r')

def opcReadProgram():
    if opc.read('PLtoCNC.Program')[1] == 'Good':
        return opc.read('PLtoCNC.Program')[0]
    else:
        return 0

def opcReadExecute():
    if opc.read('PLtoCNC.Execute')[1] == 'Good':
        return opc.read('PLtoCNC.Execute')[0]
    else:
        return False

def opcReadPause():
    if opc.read('PLtoCNC.Pause')[1] == 'Good':
        return opc.read('PLtoCNC.Pause')[0]
    else:
        return False

def opcReadAbort():
    if opc.read('PLtoCNC.Abort')[1] == 'Good':
        return opc.read('PLtoCNC.Abort')[0]
    else:
        return False

def opcReadInitialize():
    if opc.read('PLtoCNC.Initialize')[1] == 'Good':
        return opc.read('PLtoCNC.Initialize')[0]
    else:
        return False

def opcReadPark():
    if opc.read('PLtoCNC.Park')[1] == 'Good':
        return opc.read('PLtoCNC.Park')[0]
    else:
        return False

def opcReadPaused():
    if opc.read('CNtoPLC.Paused')[1] == 'Good':
        return opc.read('CNtoPLC.Paused')[0]
    else:

```



```

    return False

def opcWriteResetAlarm(Alar):
    opc.write(('CNCToPLC.ResetAlarm', Alar))

def opcWriteRunning(Run):
    opc.write(('CNCToPLC.Running', Run))

def opcWritePaused(Paus):
    opc.write(('CNCToPLC.Paused', Paus))

def opcWriteInitialized(Init):
    opc.write(('CNCToPLC.Initialized', Init))

def opcWriteParked(Par):
    opc.write(('CNCToPLC.Parked', Par))

#-----
# Main program start
#-----

try:
    app = pywinauto.Application(backend='win32').connect(path = r'C:/Mach3/Mach3.exe')
except:
    app = pywinauto.Application(backend='win32').start('C:/Mach3/Mach3.exe /p Mach3Mill')
    app.Mach3CNCLicensedTo.maximize()
    time.sleep(5)

dlg_spec = app.Mach3CNCLicensedTo
actionable_dlg = dlg_spec.wait('visible')

if checkResetAlarm() == True:
    opcWriteResetAlarm(True)

while True:

    if opcReadExecute() == True and Running == False and opcReadAbort() == False:
        loadCode(opcReadProgram())
        opcWriteParked(False)
        if runCode() == True:
            loadCode(90)
            if runCode() == True:
                opcWriteParked(True)

    if opcReadPark() == True and Running == False and opcReadAbort() == False:
        loadCode(90)
        if runCode() == True:
            opcWriteParked(True)

    if opcReadInitialize() == True and Running == False and opcReadAbort() == False:
        opcWriteInitialized(False)
        opcWriteParked(False)
        loadCode(91)
        runCode()
        if opcReadAbort() == False:
            resetAlarm()
            loadCode(92)
            runCode()
            if opcReadAbort() == False:
                resetAlarm()
                loadCode(93)
                runCode()
                if opcReadAbort() == False:
                    resetAlarm()
                    loadCode(94)
                    if runCode() == True:
                        zeroAll()
                        opcWriteInitialized(True)
                        loadCode(90)
                        if runCode() == True:

```

```
FMS.py  
opcWriteParked(True)
```

```
time.sleep(0.2)
```

D G-code

Code0.txt - Failsafe

```
N10 G17 G40 G91 G71 G96 G94
N20 T1 D1H1 S1000 M3
N30 G00 X0.1
N40 G00 X-0.1
N50 M30
```

Code1.txt - X axis check

```
N10 G17 G40 G90 G71 G96 G94
N20 T1 D1H1 S1000 M5
N30 G00 Z5
N40 G00 X8 Y42
N50 G01 X10 Y40 F50 M3
N60 G01 Z-3
N70 G01 Y10
N80 G01 Z3
N90 G00 X23 Y40
N100 G01 X25
N110 G01 Z-3
N120 G01 Y30
N130 G01 Z3
N140 G00 Y20 X27
N150 G01 x25
N160 G01 Z-3
N170 G01 Y10
N180 G01 Z3
N190 G00 Y40
N200 G00 X40
N210 G01 Z-3
N220 G01 Y30
N230 G01 Z3
N240 G00 X55 Y20
N250 G00 X40
N260 G01 Z-3
N270 G01 Y10
N280 G01 Z5
N290 M30
```

Code2.txt - Y axis check

```
N10 G17 G40 G90 G71 G96 G94
N20 T1 D1H1 S1000 M5
N30 G00 Z5
N40 G00 X42 Y42
N50 G01 X40 Y40 F50 M3
N60 G01 Z-5
N70 G01 Z-3
N80 G01 X10
N90 G01 Z-5
N100 G01 Z3
N110 G00 X40 Y27
N120 G01 Y25
N130 G01 Z-3
N140 G01 X30
N150 G01 Z3
N160 G00 X20 Y23
N170 G01 Y25
N180 G01 Z-3
N190 G01 X10
N200 G01 Z3
N210 G00 X40
N220 G00 Y10
N230 G01 Z-3
N240 G01 X30
N250 G01 Z3
N260 G00 X20 Y-5
N270 G00 Y10
N280 G01 Z-3
N290 G01 X10
N300 G01 Z5
N310 M30
```

Code3.txt - Mickey Mouse

```
N10 G90 G54 G71 G40 G17 G94
N20 G97 T1 S1000 M3
N30 G00 Z5
N40 G00 X23 Y17
N45 G01 X25 Y19 F50
N50 G01 Z-3
N60 G01 X29
N70 G02 I-4 J0
N80 G01 X33
N90 G02 I-8 J0
N100 G01 X36.5
N110 G02 I-11.5 J0
N120 G01 Z5
N130 G00 X14 Y32
N135 G01 X12 Y34
N140 G01 Z-3
N150 G01 X16
N160 G02 I-4 J0
N170 G01 X18.5
N180 G02 I-6.5 J0
N190 G01 Z5
N200 G00 X36
N205 G01 X38 Y34
N210 G01 Z-3
N220 G01 X42
N230 G02 I-4 J0
N240 G01 X44.5
N250 G02 I-6.5 J0
N260 G01 Z5
N270 G00 X24 Y28.5
N275 G01 X22 Y26.5
N280 G01 Z-5
N290 G03 X21.5 Y30 R4
N300 G03 X19.5 Y19 R9
N310 G01 Y16
N320 G01 X16.5
N330 G03 X14.3 Y15 R4
N340 G03 X35.7 I10.7 J4
N350 G03 X33.5 Y16 R4
N360 G01 X30.5
N370 G01 Y19
N380 G03 X28.5 Y30 R9
N390 G03 X28 Y26.5 R4
N400 G01 X20
N410 G01 Y21.5
N420 G01 X30
N430 G01 Y16
N440 G01 X20
N450 G01 Y10.5
N460 G01 X30
N470 G01 Z5
N480 M30
```

Code4.txt - Flyover

```
N10 G90 G54 G71 G40 G17 G94
N20 G97 T1 S1000 M3
N30 G00 X10 Y10 Z10
N40 G00 X40 Y40
N50 G00 X10
N60 G00 X40 Y10
N70 M30
```

Code90.txt - Parking

```
N10 G17 G40 G71 G90 G94
N20 G97 T1 D1 H1 S0 F150 M5
N30 G00 Z20
N40 G00 X0 Y200
N300 M30
```

Code91.txt - Z axis reset

```
N10 G17 G40 G71 G91 G94
N20 G97 T1 D1 H1 S0 F150 M5
N30 G00 Z100
N300 M30
```

Code92.txt - X axis reset

```
N10 G17 G40 G71 G91 G94
N20 G97 T1 D1 H1 S0 F150 M5
N30 G00 x250
N300 M30
```

Code93.txt - Y axis reset

```
N10 G17 G40 G71 G91 G94
N20 G97 T1 D1 H1 S0 F150 M5
N30 G00 Y-350
N300 M30
```

Code94.txt - Zero position

```
N10 G17 G21 G40 G71 G91 G94
N20 G97 T1 D1 H1 S0 F150 M5
N30 G00 X-146 Y104,85 Z-24,75
N300 M30
```


E Robot code

Robot program 200 - Parking

The image shows a robot control interface with the following elements:

- Teach** button with a hand icon.
- E.STOP** button in a red box.
- Program** field: 200 [EX]
- Step** field: 5 STEPS, 0
- Date/Time** field: 6/5/2019 17:15
- Unit** field: M1: MZ07-01
- Manual Speed** field: 2
- Robot** icon.
- Robot Program** header with **UNIT1** on the right.
- 10.0 % JOINT A5 T2 S3** line.
- 0 [START]** line.
- 1 SETM[097,1]** with **FN105;Output signal**.
- 2 SETM[099,0]** with **FN105;Output signal**.
- 3 DELAY[2]** with **FN50;Timer delay**.
- 4 10.0 % JOINT A1 T2 S3** line.
- 5 END** with **FN92;End**.
- [EOF]** at the bottom left.

Robot program 201 - Move in

Teach 	Program	Step	6/5/2019 17:14	
	201 [EX]	19 STEPS 19		

M1:
MZ07-01



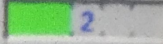
Manual Speed
2

[1] Robot Program UNIT1

	10.0 %	JOINT	A5	T2	S3	
0	[START]					
1	SETM[097,1]					FN105;Output signal
2	SETM[099,0]					FN105;Output signal
3	DELAY[2]					FN50;Timer delay
4	10.0 %	JOINT	A8	T2		
5	40.0 %	JOINT	A1	T2	S3	
6	5.0 %	LIN	A1	T2	S3	
7	SETM[097,0]					FN105;Output signal
8	SETM[099,1]					FN105;Output signal
9	DELAY[2]					FN50;Timer delay
10	5.0 %	LIN	A1	T2	S3	
11	40.0 %	JOINT	A8	T2		
12	40.0 %	JOINT	A1	T2	S3	
13	5.0 %	LIN	A1	T2	S3	
14	SETM[097,1]					FN105;Output signal
15	SETM[099,0]					FN105;Output signal
16	DELAY[2]					FN50;Timer delay
17	5.0 %	LIN	A1	T2	S3	
18	40.0 %	JOINT	A1	T2	S3	
19	END					FN92;End

[EOF]

Robot program 202 - Move out

Teach  E.STOP	Program	Step	6/5/2019 17:15		
	202 [EX]	19 STEPS 0			
					M1: MZ07-01 Manual Speed  2
[1] Robot Program					UNIT1
10.0 % JOINT A5 T2 S3					
0 [START]					
1	SETM[097,1]				FN105;Output signal
2	SETM[099,0]				FN105;Output signal
3	DELAY[2]				FN50;Timer delay
4	10.0 %	JOINT	A8	T2	S3
5	40.0 %	JOINT	A1	T2	S3
6	5.0 %	LIN	A1	T2	S3
7	SETM[097,0]				FN105;Output signal
8	SETM[099,1]				FN105;Output signal
9	DELAY[2]				FN50;Timer delay
10	5.0 %	LIN	A1	T2	S3
11	40.0 %	JOINT	A8	T2	S3
12	40.0 %	JOINT	A1	T2	S3
13	5.0 %	LIN	A1	T2	S3
14	SETM[097,1]				FN105;Output signal
15	SETM[099,0]				FN105;Output signal
16	DELAY[2]				FN50;Timer delay
17	5.0 %	LIN	A1	T2	S3
18	40.0 %	JOINT	A1	T2	S3
19	END				FN92;End
[EOF]					

F 222 PLC code

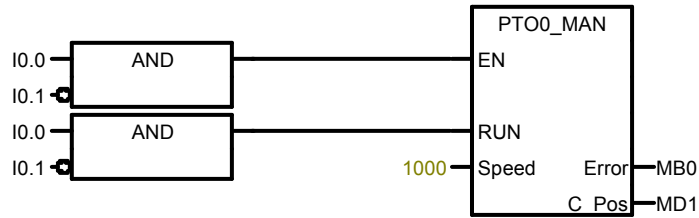
Block: MAIN
 Author:
 Created: 04/24/2019 06:11:16 pm
 Last Modified: 04/25/2019 12:33:32 pm

Symbol	Var Type	Data Type	Comment
	TEMP		
	TEMP		
	TEMP		
	TEMP		

PROGRAM COMMENTS

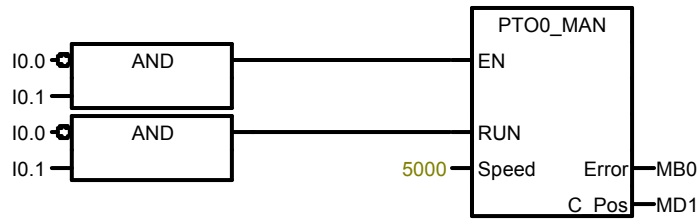
Network 1 Stepper 1 Slow

Sends a 1000 Hz pulse on output 0.0



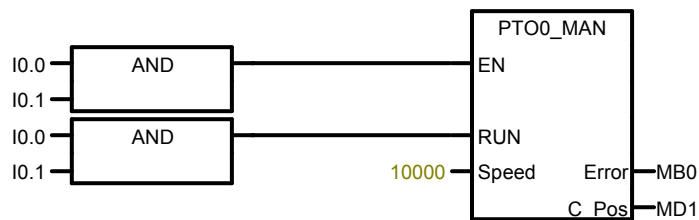
Network 2 Stepper 1 Medium

Sends a 5000 Hz pulse on output 0.0

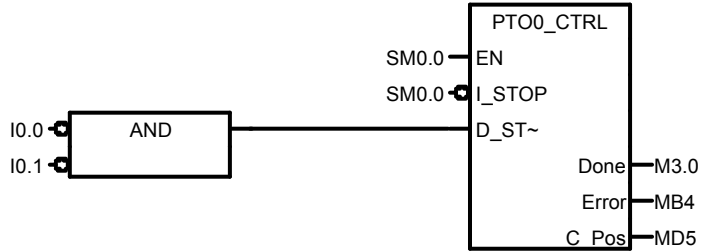


Network 3 Stepper 1 Fast

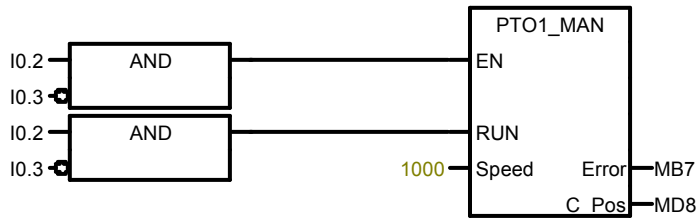
Sends a 10000 Hz pulse on output 0.0



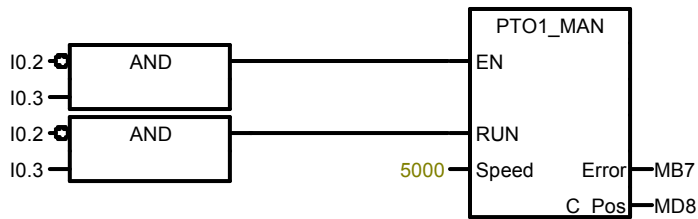
Network 4 Stepper 1 Stop
Ramps down pulse train to zero



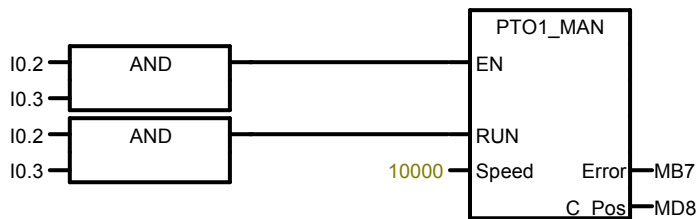
Network 5 Stepper 2 Slow
Sends a 1000 Hz pulse on output 0.1



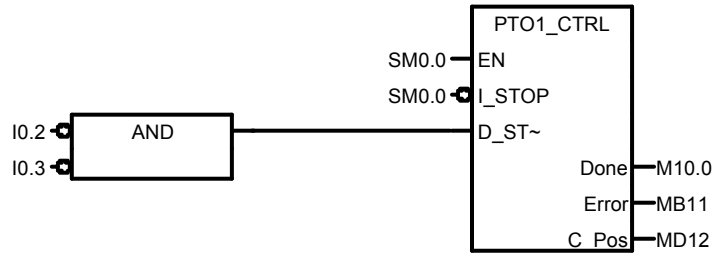
Network 6 Stepper 2 Medium
Sends a 5000 Hz pulse on output 0.1



Network 7 Stepper 2 Fast
Sends a 10000 Hz pulse on output 0.1



Network 8 Stepper 2 Stop
Ramps down pulse train to zero



Block: PTO0_CTRL
Author: Position Control Wizard
Created: 04/25/2019 12:15:48 pm
Last Modified: 04/25/2019 12:15:48 pm

	Symbol	Var Type	Data Type	Comment
	EN	IN	BOOL	
L0.0	I_STOP	IN	BOOL	Stop current movement immediately
L0.1	D_STOP	IN	BOOL	Stop current movement by decelerating.
		IN		
		IN_OUT		
L0.2	Done	OUT	BOOL	On when the PTO operation completes the command
LB1	Error	OUT	BYTE	0=No error. See POU comment for additional errors
LD2	C_Pos	OUT	DINT	Number of segments in progress
		OUT		
		TEMP		



This instruction was generated by the PTO/PWM wizard for use with output Q0.0.

Block: PTO0_RUN
 Author: Position Control Wizard
 Created: 04/25/2019 12:15:48 pm
 Last Modified: 04/25/2019 12:15:48 pm

	Symbol	Var Type	Data Type	Comment
	EN	IN	BOOL	
L0.0	START	IN	BOOL	Send command to PTO if not active
LB1	Profile	IN	BYTE	Number of motion profile to run
L2.0	Abort	IN	BOOL	Abort the Run command
		IN		
		IN_OUT		
L2.1	Done	OUT	BOOL	On when the PTO operation completes the command
LB3	Error	OUT	BYTE	0=No error. See POU comment for additional errors
LB4	C_Profile	OUT	BYTE	Current profile
LB5	C_Step	OUT	BYTE	Current step in the profile
LD6	C_Pos	OUT	DINT	Current position
		OUT		
		TEMP		



This instruction was generated by the PTO/PWM wizard for use with output Q0.0. The PTOx_RUN (Run Motion Profile) instruction is used to command the linear PTO operation to execute a motion profile specified in the wizard configuration. The following motion profiles are defined for this operation:

 Profile 'HighSpeed' defines a single speed move at 1000 pulses/sec

Block: PTO0_MAN
 Author: Position Control Wizard
 Created: 04/25/2019 12:15:48 pm
 Last Modified: 04/25/2019 12:15:48 pm

	Symbol	Var Type	Data Type	Comment
	EN	IN	BOOL	
L0.0	RUN	IN	BOOL	RUN/STOP accelerate to target speed and target position
LD1	Speed	IN	DINT	Target speed
		IN		
		IN_OUT		
LB5	Error	OUT	BYTE	0=No error. See POU comment for additional errors
LD6	C_Pos	OUT	DINT	Current position
		OUT		
		TEMP		



This instruction was generated by the PTO/PWM wizard for use with output Q0.0. The PTOx_MAN (Manual Mode) instruction is used to control the linear PTO in manual mode. In manual mode, PTO can be commanded at different speeds. While the PTOx_MAN instruction is enabled, only the PTOx_CTRL instructions are allowed.

Block: PTO1_CTRL
Author: Position Control Wizard
Created: 04/25/2019 12:29:38 pm
Last Modified: 04/25/2019 12:29:38 pm

	Symbol	Var Type	Data Type	Comment
	EN	IN	BOOL	
L0.0	I_STOP	IN	BOOL	Stop current movement immediately
L0.1	D_STOP	IN	BOOL	Stop current movement by decelerating.
		IN		
		IN_OUT		
L0.2	Done	OUT	BOOL	On when the PTO operation completes the command
LB1	Error	OUT	BYTE	0=No error. See POU comment for additional errors
LD2	C_Pos	OUT	DINT	Number of segments in progress
		OUT		
		TEMP		



This instruction was generated by the PTO/PWM wizard for use with output Q0.1.

Block: PTO1_RUN
 Author: Position Control Wizard
 Created: 04/25/2019 12:29:38 pm
 Last Modified: 04/25/2019 12:29:38 pm

	Symbol	Var Type	Data Type	Comment
	EN	IN	BOOL	
L0.0	START	IN	BOOL	Send command to PTO if not active
LB1	Profile	IN	BYTE	Number of motion profile to run
L2.0	Abort	IN	BOOL	Abort the Run command
		IN		
		IN_OUT		
L2.1	Done	OUT	BOOL	On when the PTO operation completes the command
LB3	Error	OUT	BYTE	0=No error. See POU comment for additional errors
LB4	C_Profile	OUT	BYTE	Current profile
LB5	C_Step	OUT	BYTE	Current step in the profile
LD6	C_Pos	OUT	DINT	Current position
		OUT		
		TEMP		



This instruction was generated by the PTO/PWM wizard for use with output Q0.1. The PTOx_RUN (Run Motion Profile) instruction is used to command the linear PTO operation to execute a motion profile specified in the wizard configuration. The following motion profiles are defined for this operation:

 Profile 'Profile0_0' defines a single speed move at 1000 pulses/sec

Block: PTO1_MAN
 Author: Position Control Wizard
 Created: 04/25/2019 12:29:38 pm
 Last Modified: 04/25/2019 12:29:38 pm

	Symbol	Var Type	Data Type	Comment
	EN	IN	BOOL	
L0.0	RUN	IN	BOOL	RUN/STOP accelerate to target speed and target position
LD1	Speed	IN	DINT	Target speed
		IN		
		IN_OUT		
LB5	Error	OUT	BYTE	0=No error. See POU comment for additional errors
LD6	C_Pos	OUT	DINT	Current position
		OUT		
		TEMP		



This instruction was generated by the PTO/PWM wizard for use with output Q0.1. The PTOx_MAN (Manual Mode) instruction is used to control the linear PTO in manual mode. In manual mode, PTO can be commanded at different speeds. While the PTOx_MAN instruction is enabled, only the PTOx_CTRL instructions are allowed.

G 315 PLC code

OB1 - <offline>

```

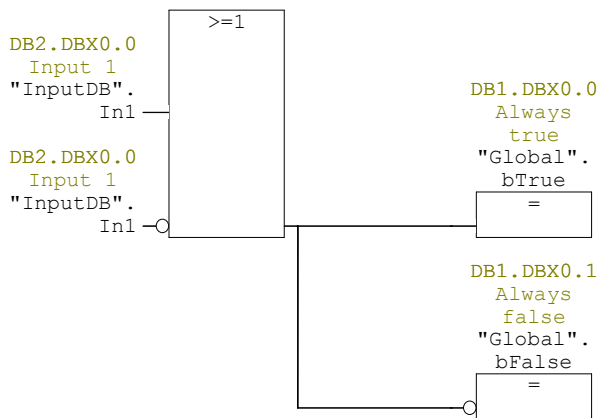
Name:
Author:
Family:
Version: 0.1
Block version: 2
Time stamp Code: 05/03/2019 11:11:32 AM
Interface: 02/15/1996 04:51:12 PM
Lengths (block/logic/data): 00178 00058 00022
    
```

Name	Data Type	Address	Comment
TEMP		0.0	
OB1_EV_CLASS	Byte	0.0	Bits 0-3 = 1 (Coming event), Bits 4-7 = 1 (Event class 1)
OB1_SCAN_1	Byte	1.0	1 (Cold restart scan 1 of OB 1), 3 (Scan 2-n of OB 1)
OB1_PRIORITY	Byte	2.0	Priority of OB Execution
OB1_OB_NUMBR	Byte	3.0	1 (Organization block 1, OB1)
OB1_RESERVED_1	Byte	4.0	Reserved for system
OB1_RESERVED_2	Byte	5.0	Reserved for system
OB1_PREV_CYCLE	Int	6.0	Cycle time of previous OB1 scan (milliseconds)
OB1_MIN_CYCLE	Int	8.0	Minimum cycle time of OB1 (milliseconds)
OB1_MAX_CYCLE	Int	10.0	Maximum cycle time of OB1 (milliseconds)
OB1_DATE_TIME	Date_And_Time	12.0	Date and time OB1 started

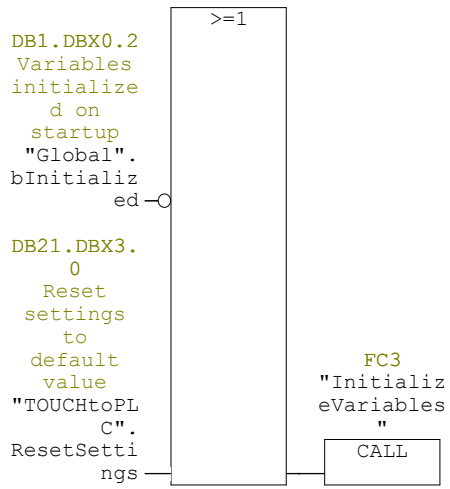
Block: OB1 "Main Program Sweep (Cycle)"

Network: 1 Always true

Generate bTrue and bFalse



Network: 2 Initialize variables on startup



OB35 - <offline>

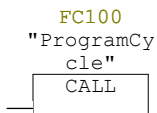
"CYC_INT5" Cyclic Interrupt 5
Name: **Family:**
Author: **Version:** 0.1
Block version: 2
Time stamp Code: 04/11/2019 07:58:21 PM
Interface: 02/15/1996 04:51:11 PM
Lengths (block/logic/data): 00120 00008 00020

Name	Data Type	Address	Comment
TEMP		0.0	
OB35_EV_CLASS	Byte	0.0	Bits 0-3 = 1 (Coming event), Bits 4-7 = 1 (Event class 1)
OB35_STRT_INF	Byte	1.0	16#36 (OB 35 has started)
OB35_PRIORITY	Byte	2.0	Priority of OB Execution
OB35_OB_NUMBR	Byte	3.0	35 (Organization block 35, OB35)
OB35_RESERVED_1	Byte	4.0	Reserved for system
OB35_RESERVED_2	Byte	5.0	Reserved for system
OB35_PHASE_OFFSET	Word	6.0	Phase offset (msec)
OB35_RESERVED_3	Int	8.0	Reserved for system
OB35_EXC_FREQ	Int	10.0	Frequency of execution (msec)
OB35_DATE_TIME	Date_And_Time	12.0	Date and time OB35 started

Block: OB35 "Cyclic Interrupt"

Network: 1

Call program cycle



FC1 - <offline>

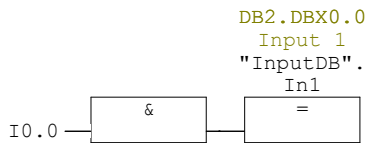
"InputMapping"

Name: Family:
Author: Version: 0.1
Block version: 2
Time stamp Code: 04/11/2019 07:34:23 PM
Interface: 04/11/2019 03:48:16 PM
Lengths (block/logic/data): 00732 00514 00000

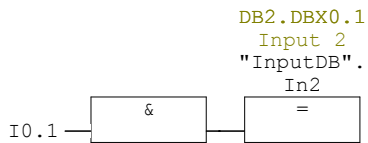
Name	Data Type	Address	Comment
IN		0.0	
OUT		0.0	
IN_OUT		0.0	
TEMP		0.0	
RETURN		0.0	
RET_VAL		0.0	

Block: FC1

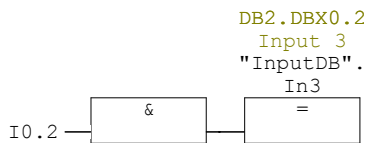
Network: 1 Input 1



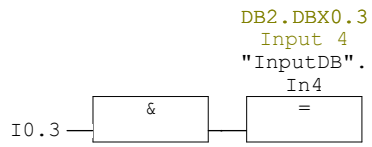
Network: 2 Input 2



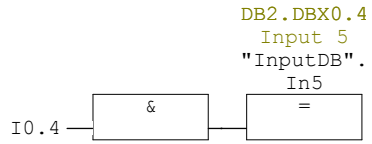
Network: 3 Input 3



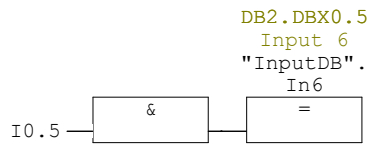
Network: 4 Input 4



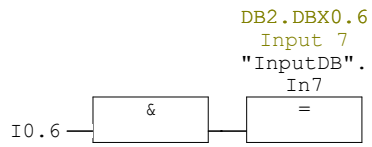
Network: 5 Input 5



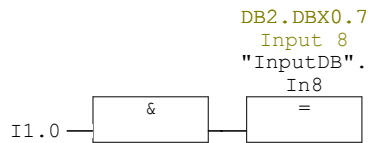
Network: 6 Input 6



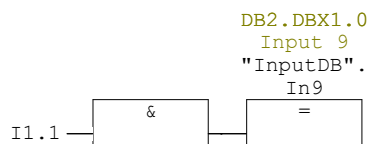
Network: 7 Input 7



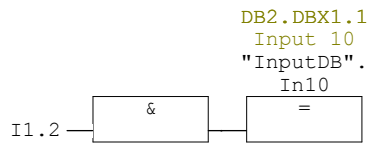
Network: 8 Input 8



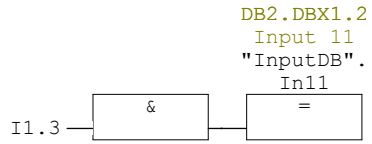
Network: 9 Input 9



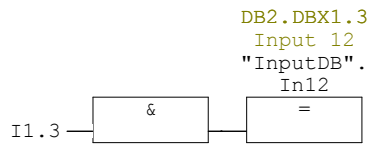
Network: 10 Input 10



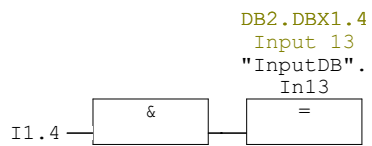
Network: 11 Input 11



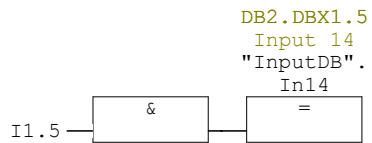
Network: 12 Input 12



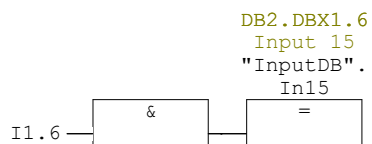
Network: 13 Input 13



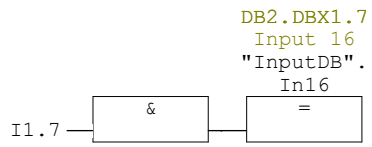
Network: 14 Input 14



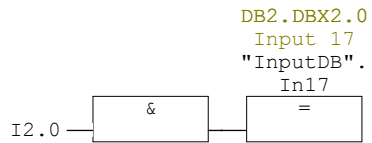
Network: 15 Input 15



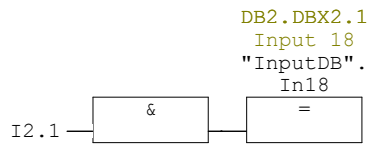
Network: 16 Input 16



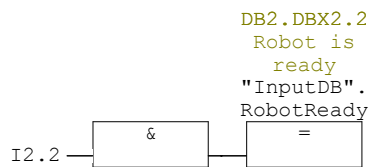
Network: 17 Input 17



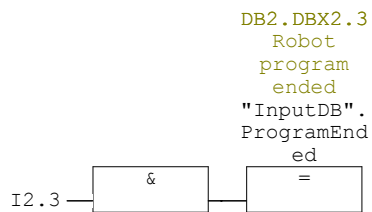
Network: 18 Input 18



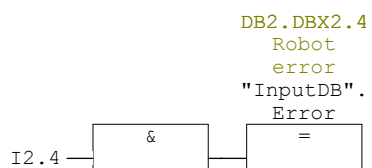
Network: 19 Input 19



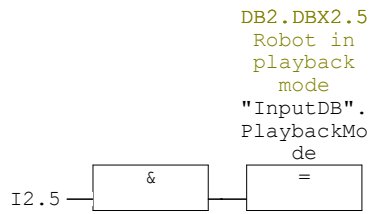
Network: 20 Input 20



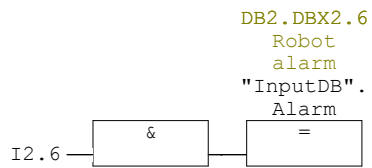
Network: 21 Input 21



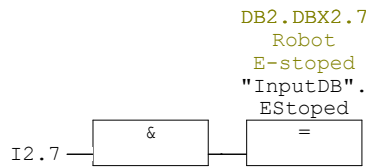
Network: 22 Input 22



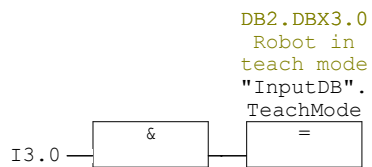
Network: 23 Input 23



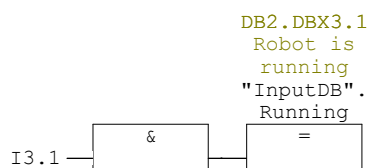
Network: 24 Input 24



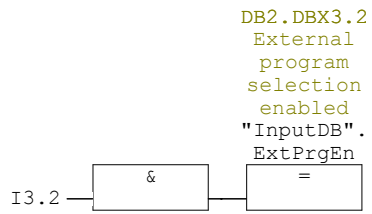
Network: 25 Input 25



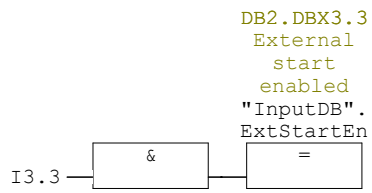
Network: 26 Input 26



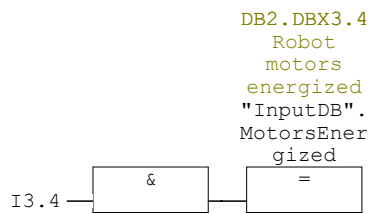
Network: 27 Input 27



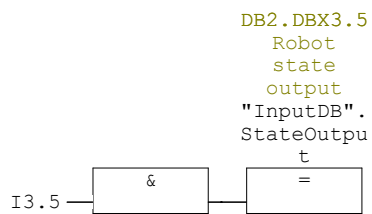
Network: 28 Input 28



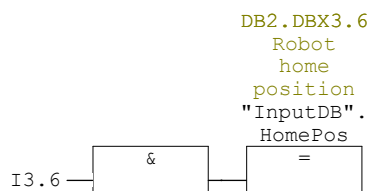
Network: 29 Input 29



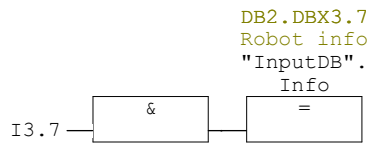
Network: 30 Input 30



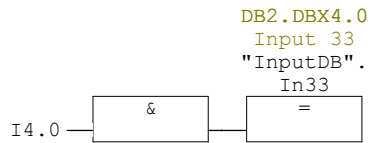
Network: 31 Input 31



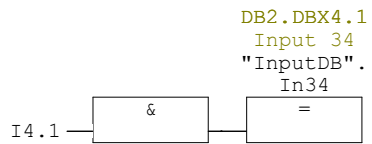
Network: 32 Input 32



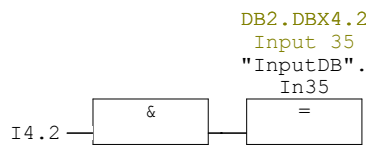
Network: 33 Input 33



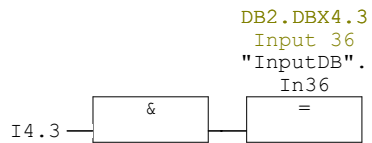
Network: 34 Input 34



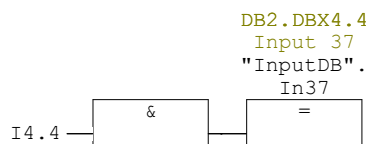
Network: 35 Input 35



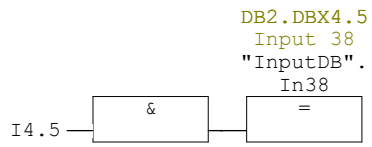
Network: 36 Input 36



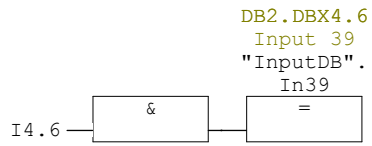
Network: 37 Input 37



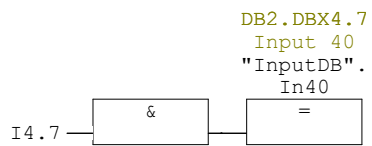
Network: 38 Input 38



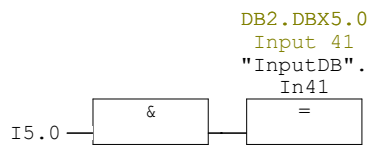
Network: 39 Input 39



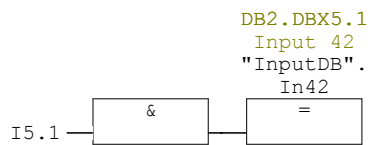
Network: 40 Input 40



Network: 41 Input 41



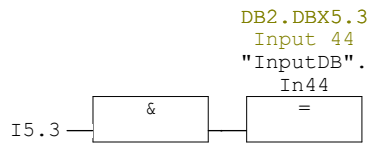
Network: 42 Input 42



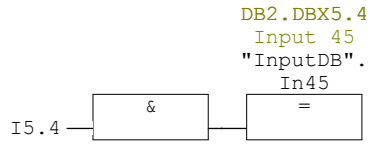
Network: 43 Input 43



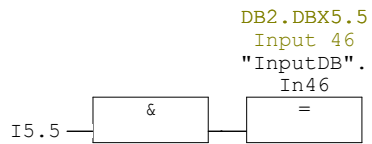
Network: 44 Input 44



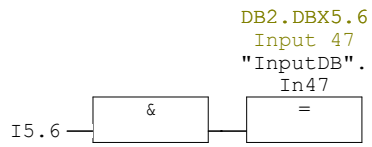
Network: 45 Input 45



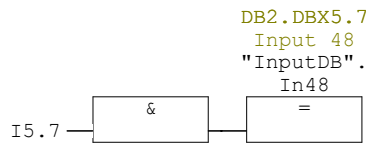
Network: 46 Input 46



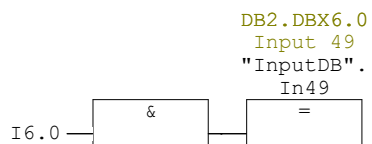
Network: 47 Input 47



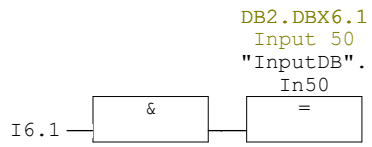
Network: 48 Input 48



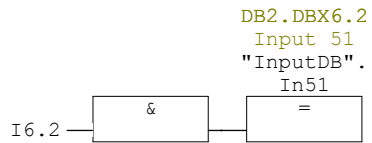
Network: 49 Input 49



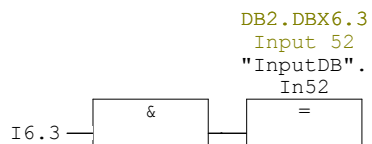
Network: 50 Input 50



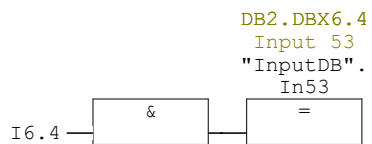
Network: 51 Input 51



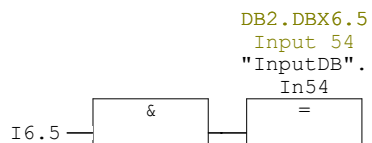
Network: 52 Input 52



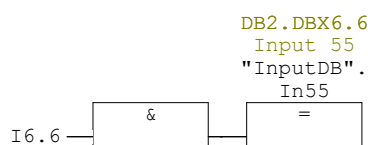
Network: 53 Input 53



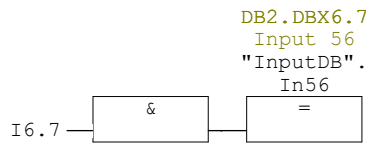
Network: 54 Input 54



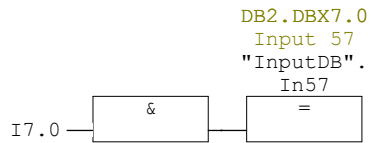
Network: 55 Input 55



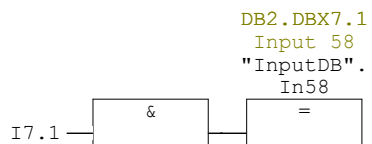
Network: 56 Input 56



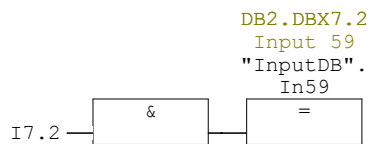
Network: 57 Input 57



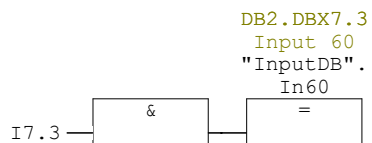
Network: 58 Input 58



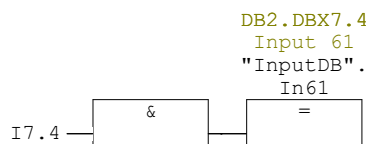
Network: 59 Input 59



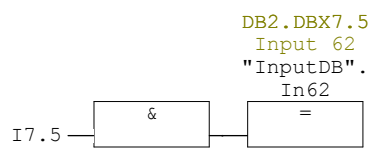
Network: 60 Input 60



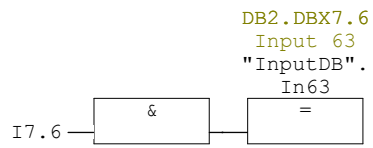
Network: 61 Input 61



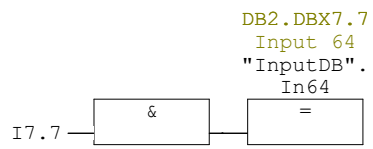
Network: 62 Input 62



Network: 63 Input 63



Network: 64 Input 64



FC2 - <offline>

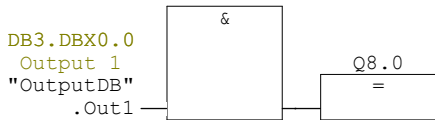
"OutputMapping"

Name: Family:
Author: Version: 0.1
Block version: 2
Time stamp Code: 05/06/2019 02:29:32 PM
Interface: 04/11/2019 02:40:41 PM
Lengths (block/logic/data): 00678 00460 00000

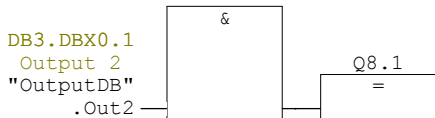
Name	Data Type	Address	Comment
IN		0.0	
OUT		0.0	
IN_OUT		0.0	
TEMP		0.0	
RETURN		0.0	
RET_VAL		0.0	

Block: FC2

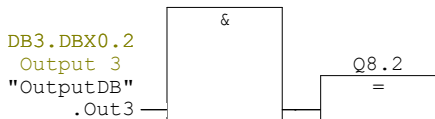
Network: 1 Output 1



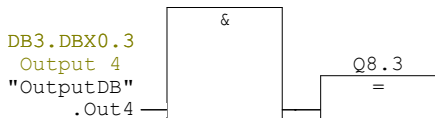
Network: 2 Output 2



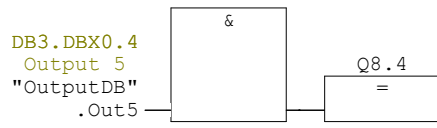
Network: 3



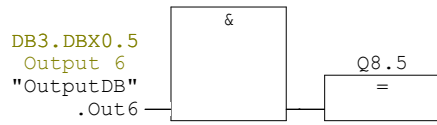
Network: 4



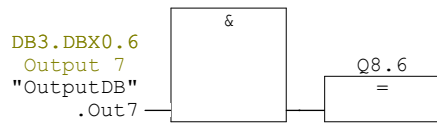
Network: 5



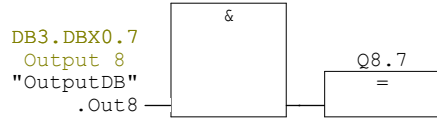
Network: 6



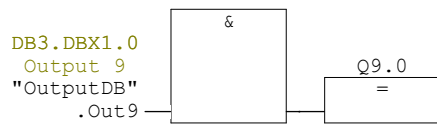
Network: 7



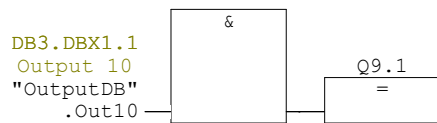
Network: 8



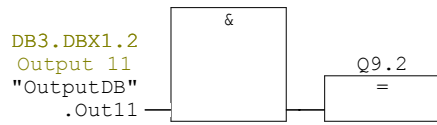
Network: 9



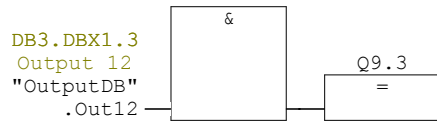
Network: 10



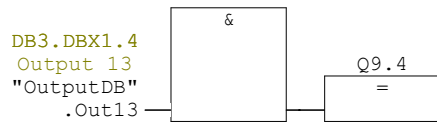
Network: 11



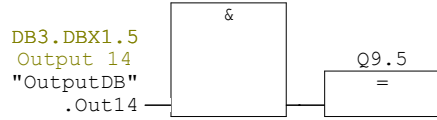
Network: 12



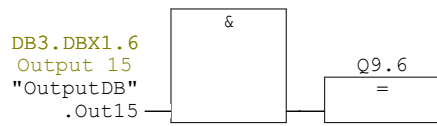
Network: 13



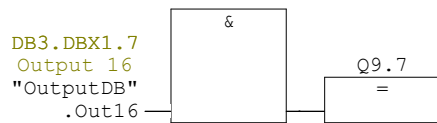
Network: 14



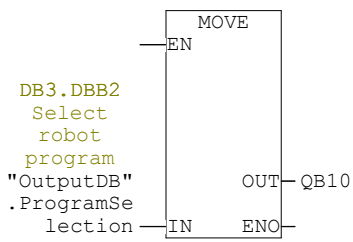
Network: 15



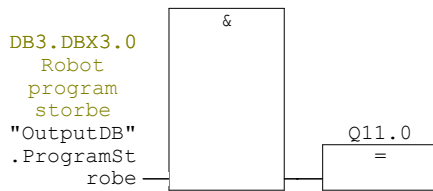
Network: 16



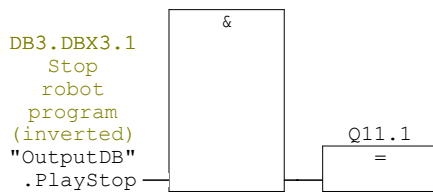
Network: 17



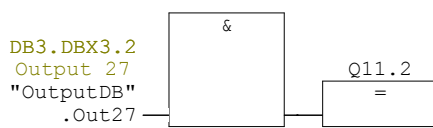
Network: 18



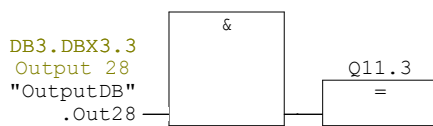
Network: 19



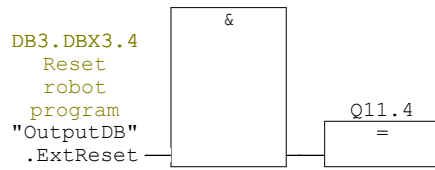
Network: 20



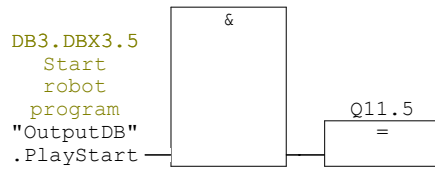
Network: 21



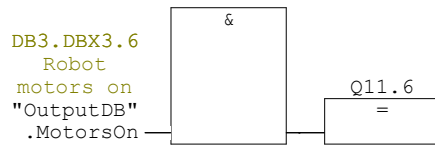
Network: 22



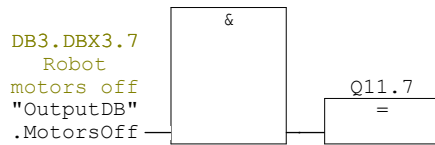
Network: 23



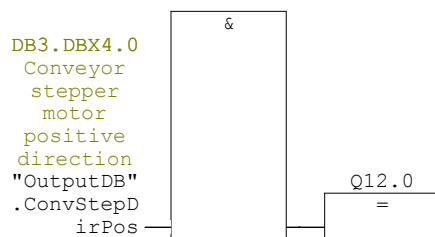
Network: 24



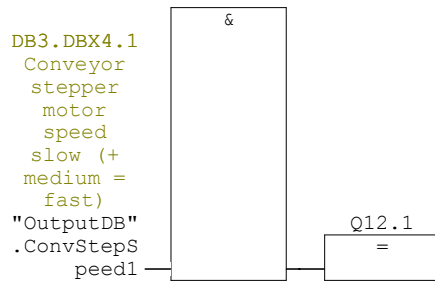
Network: 25



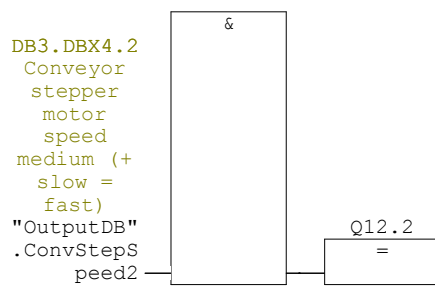
Network: 26



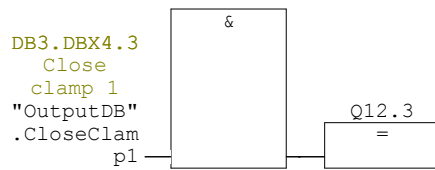
Network: 27



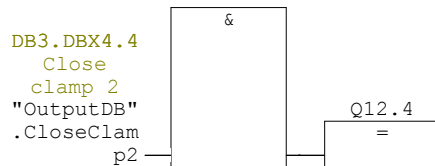
Network: 28



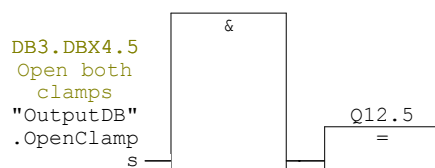
Network: 29



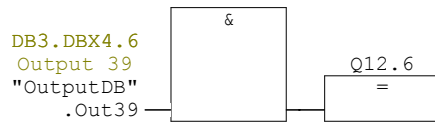
Network: 30



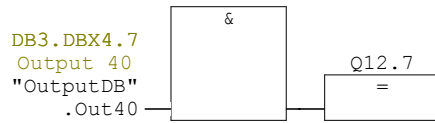
Network: 31



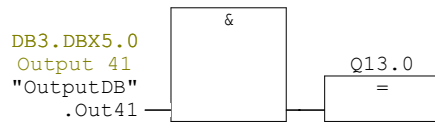
Network: 32



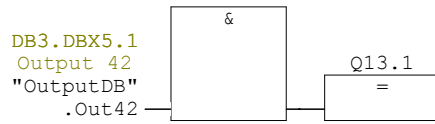
Network: 33



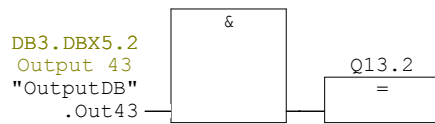
Network: 34



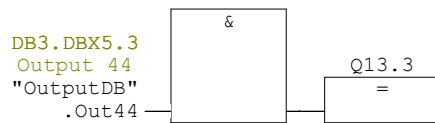
Network: 35



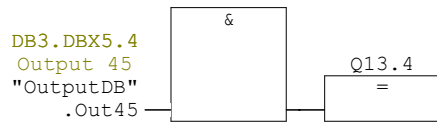
Network: 36



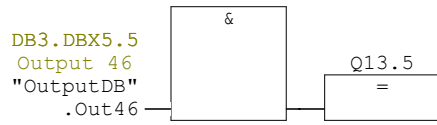
Network: 37



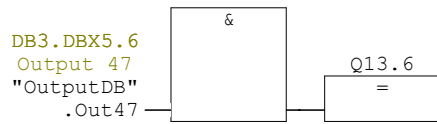
Network: 38



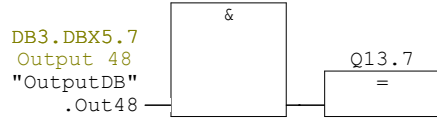
Network: 39



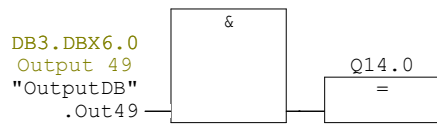
Network: 40



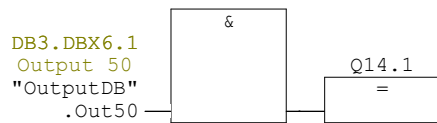
Network: 41



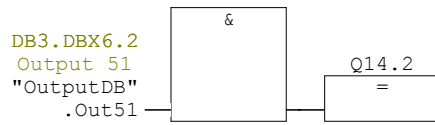
Network: 42



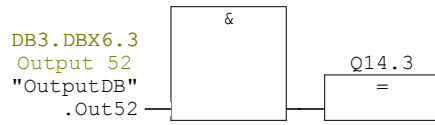
Network: 43



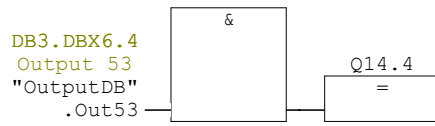
Network: 44



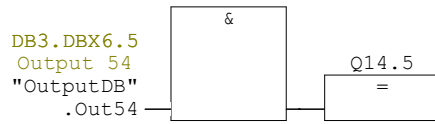
Network: 45



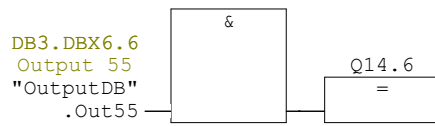
Network: 46



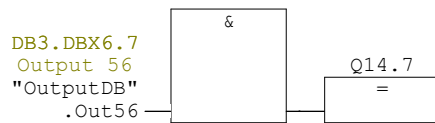
Network: 47



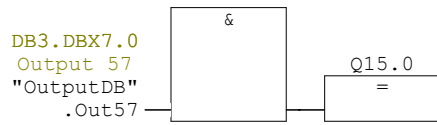
Network: 48



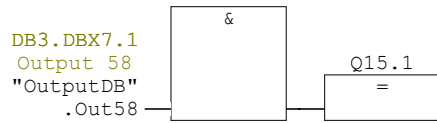
Network: 49



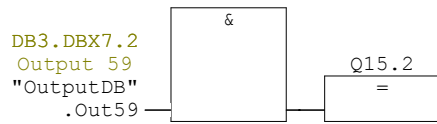
Network: 50



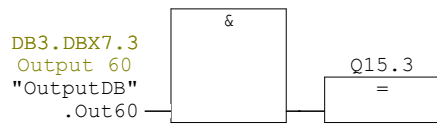
Network: 51



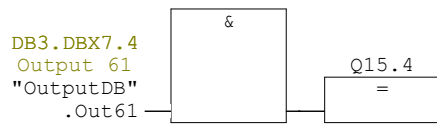
Network: 52



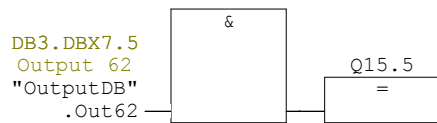
Network: 53



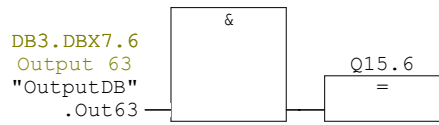
Network: 54



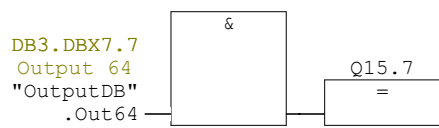
Network: 55



Network: 56



Network: 57



FC3 - <offline>

"InitializeVariables"

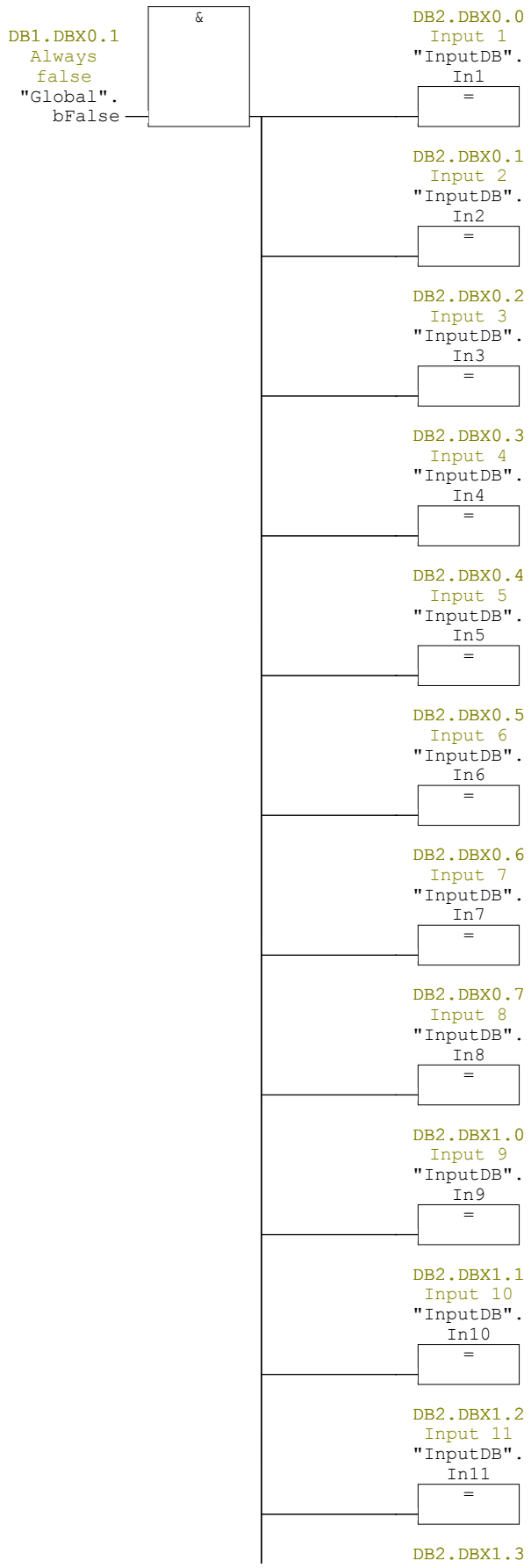
Name:
Author:
Time stamp Code:
Lengths (block/logic/data):

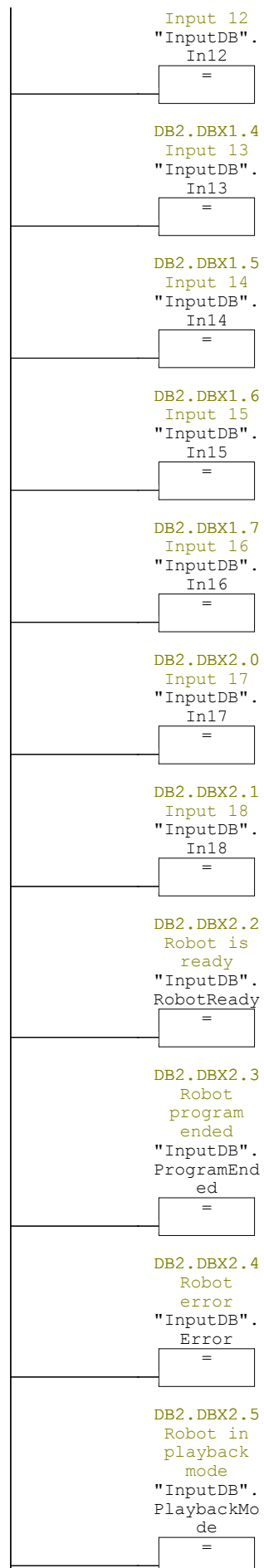
Family:
Version: 0.1
Block version: 2
05/09/2019 03:50:40 PM
04/27/2019 04:16:49 PM
01272 01156 00002

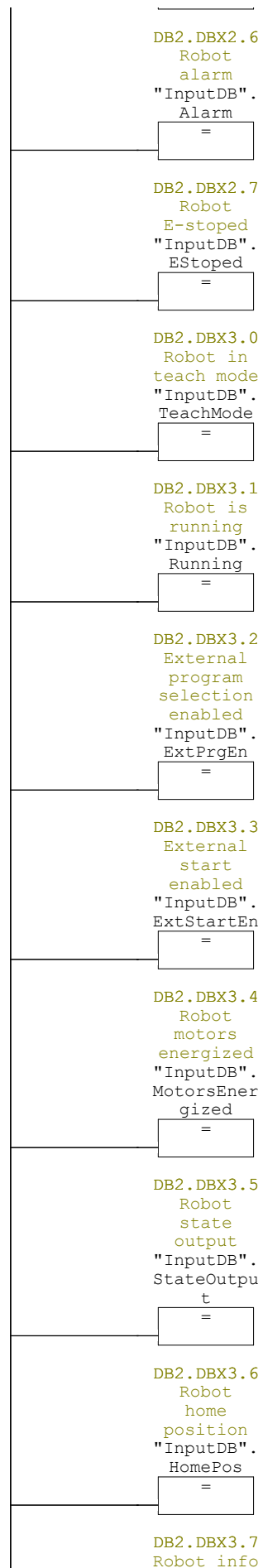
Name	Data Type	Address	Comment
IN		0.0	
OUT		0.0	
IN_OUT		0.0	
TEMP		0.0	
RETURN		0.0	
RET_VAL		0.0	

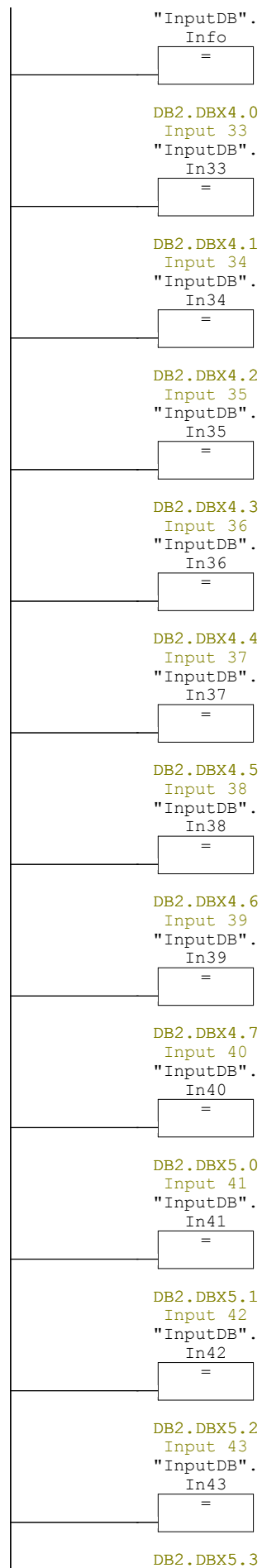
Block: FC3

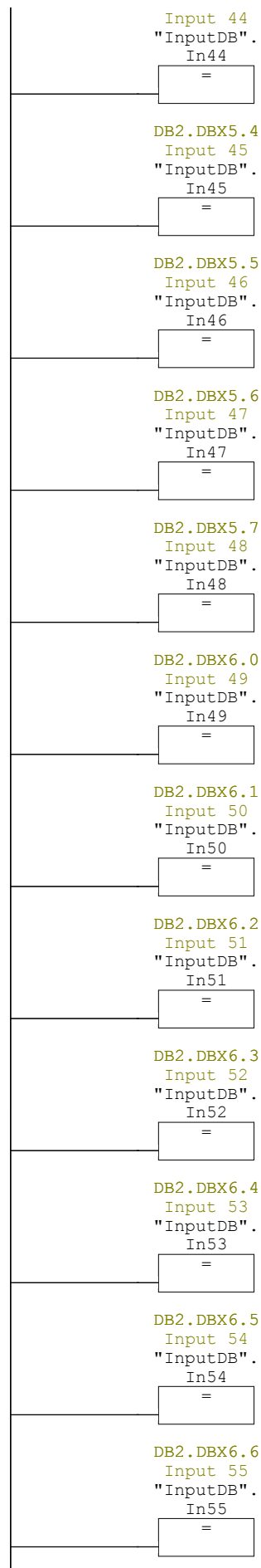
Network: 1 Reset input DB

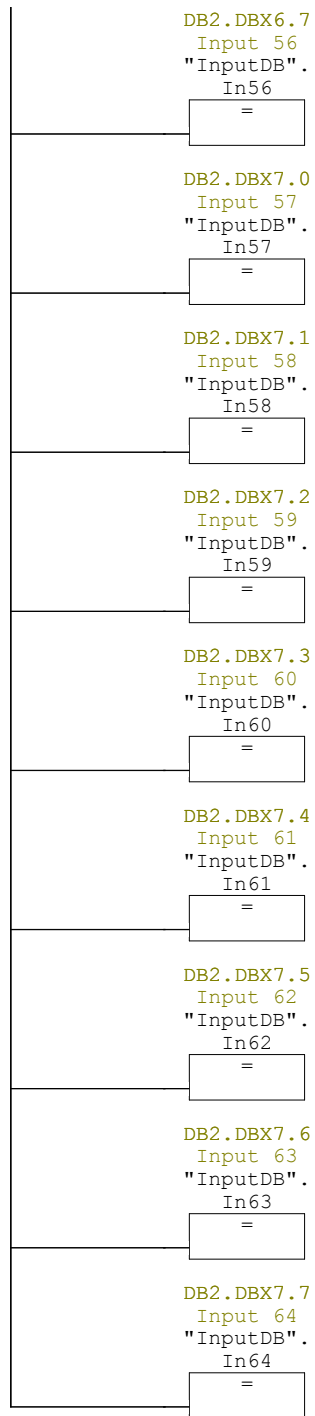




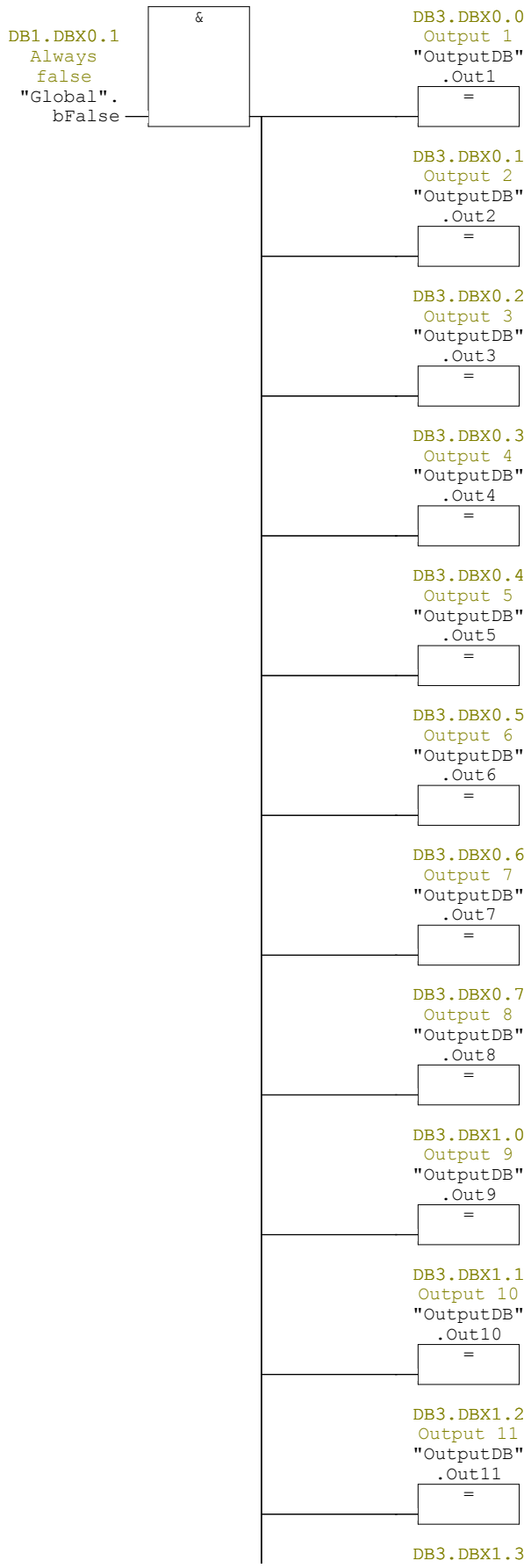


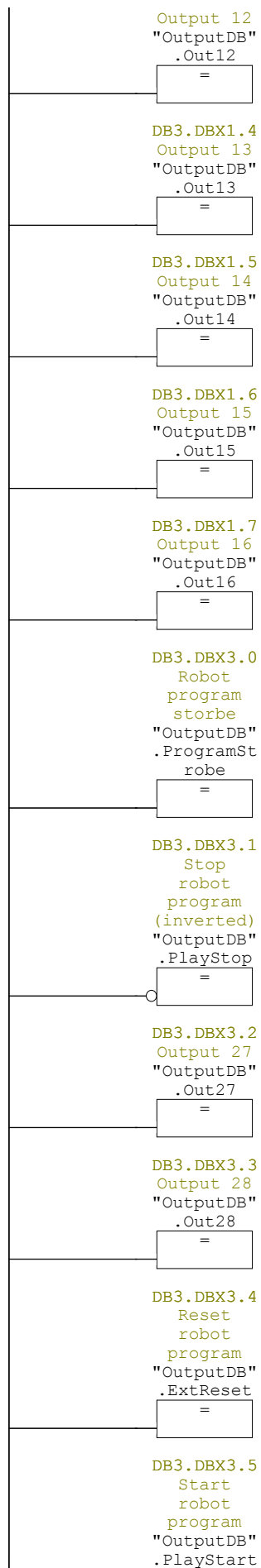


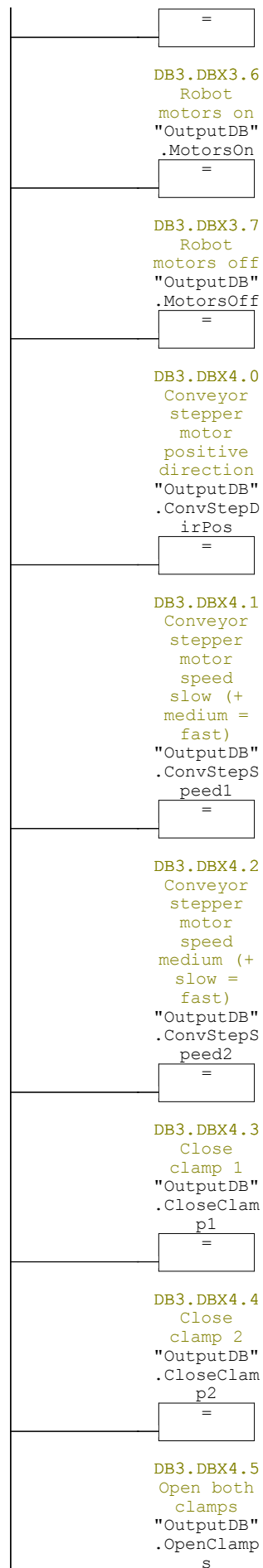


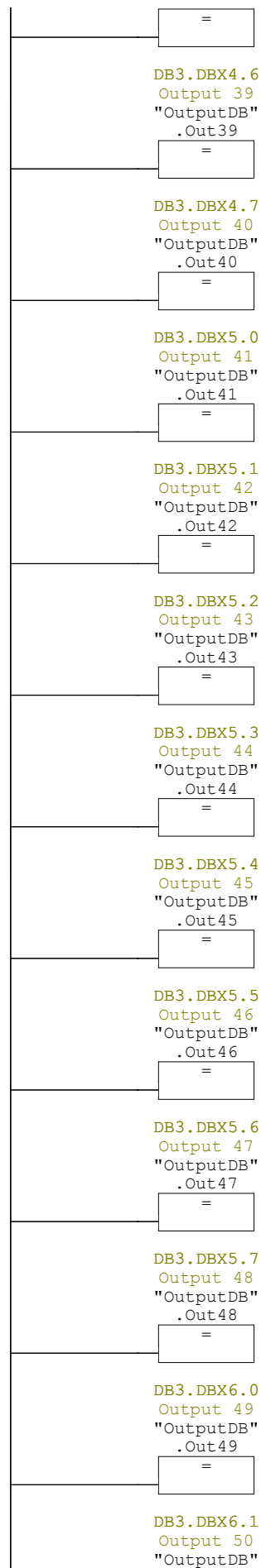


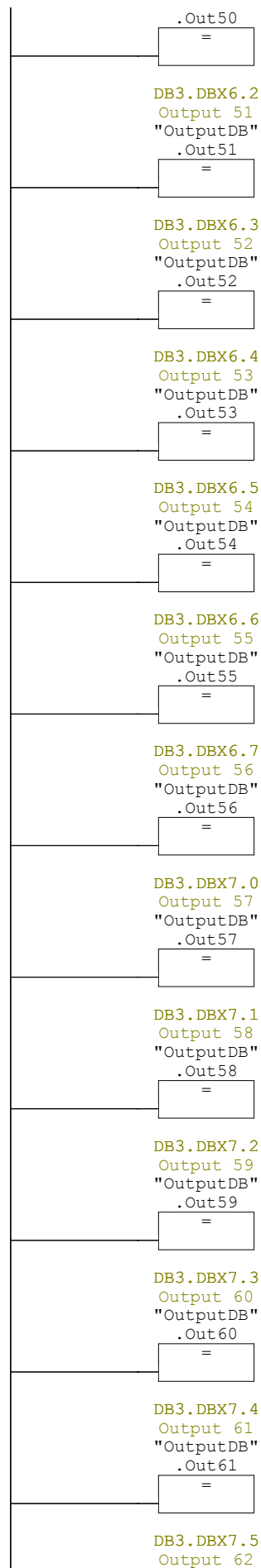
Network: 2 Reset output DB

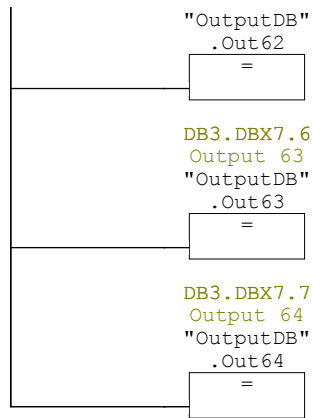






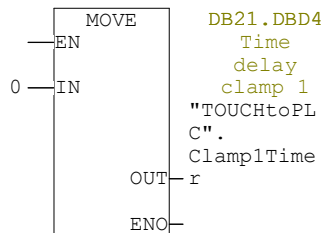






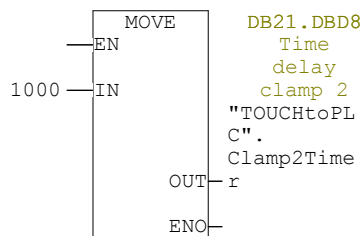
Network: 3

Set default delay for clamp 1 close



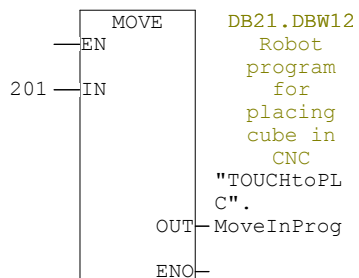
Network: 4

Set default delay for clamp 2 close



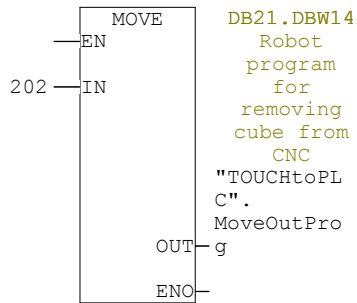
Network: 5

Set default move in program



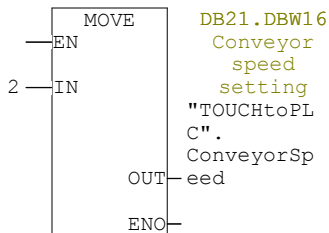
Network: 6

Set default move out program



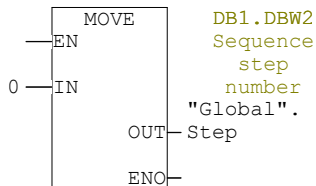
Network: 7

Set default conveyor speed



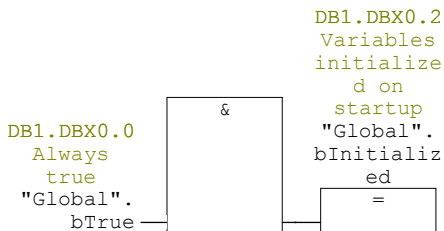
Network: 8

Set step number to 0



Network: 9 Set initialized bit

Must be last network!



FC5 - <offline>

"SignalMapping"

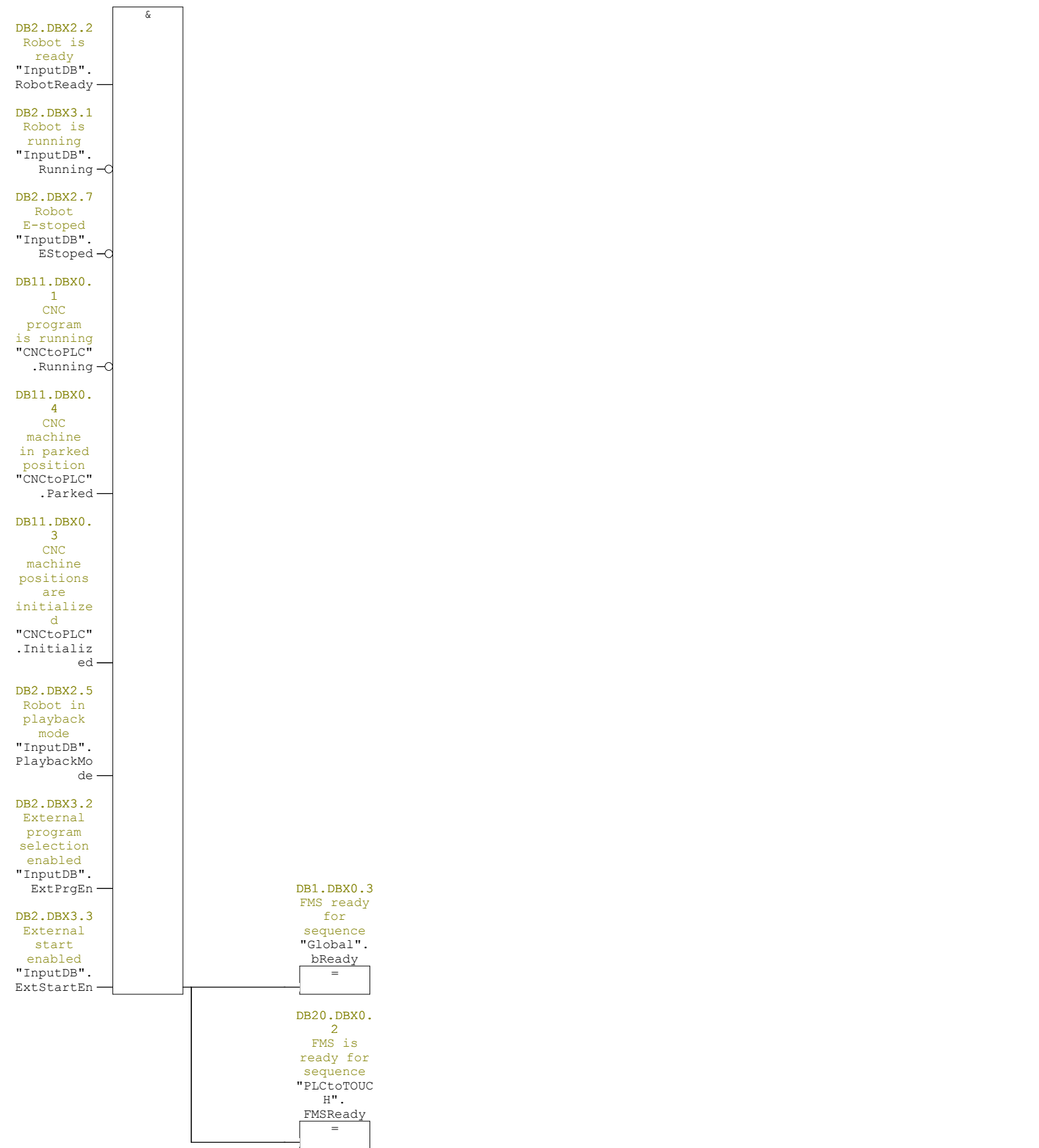
Name:
Author:
Time stamp Code:
Lengths (block/logic/data):

Family:
Version: 0.1
Block version: 2
05/13/2019 06:33:52 PM
05/06/2019 06:37:23 PM
01164 00978 00008

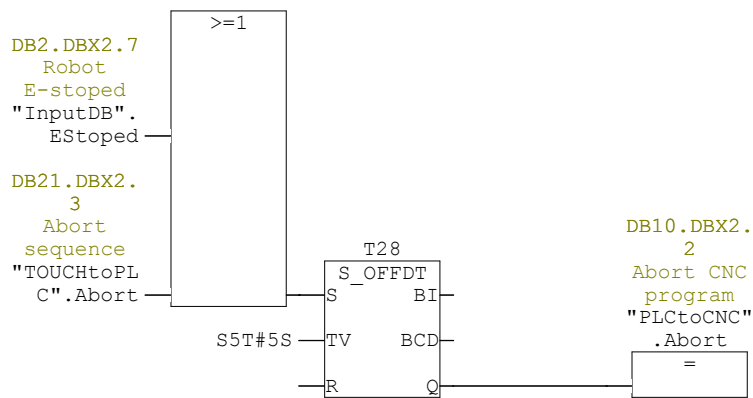
Name	Data Type	Address	Comment
IN		0.0	
OUT		0.0	
IN_OUT		0.0	
TEMP		0.0	
RETURN		0.0	
RET_VAL		0.0	

Block: FC5

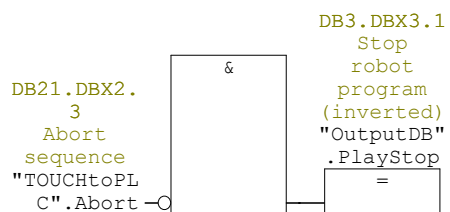
Network: 1 FMS ready



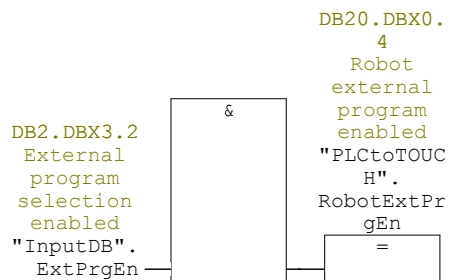
Network: 2 Send robot e-stop and touch abort to CNC



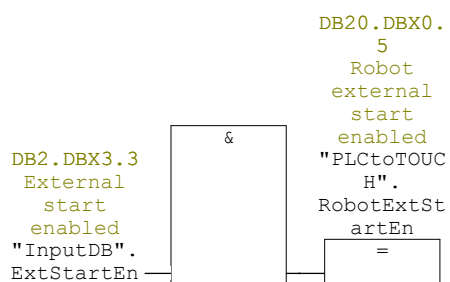
Network: 3 Stop robot program



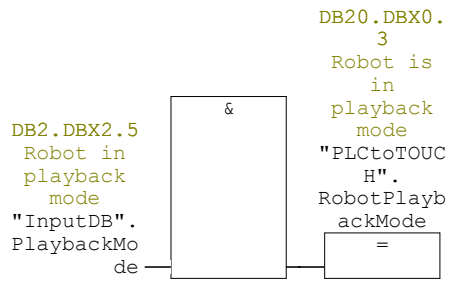
Network: 4 Robot external program enabled



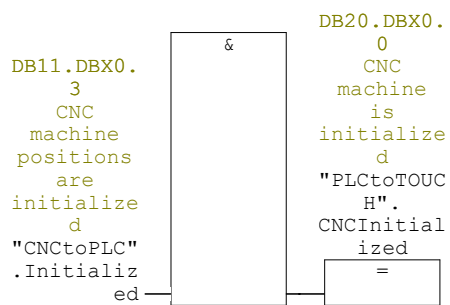
Network: 5 Robot external start enabled



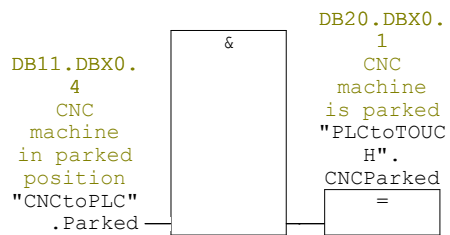
Network: 6 Robot is in playback mode



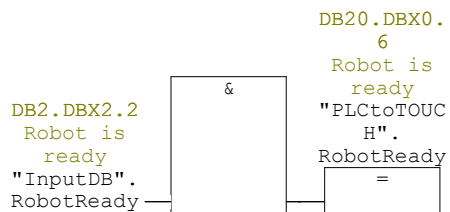
Network: 7 CNC machine is initialized



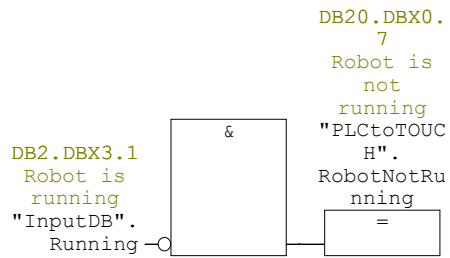
Network: 8 CNC machine is parked



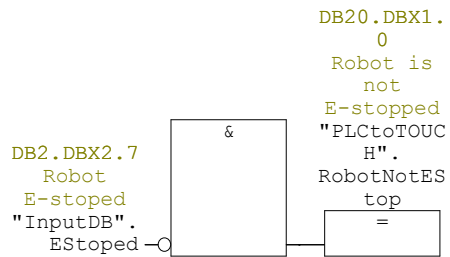
Network: 9 Robot is ready



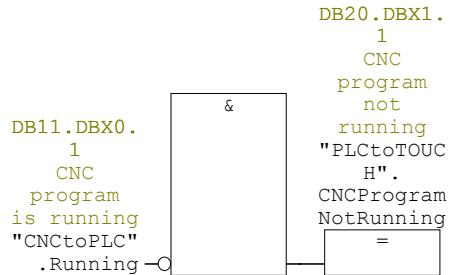
Network: 10 Robot is not running



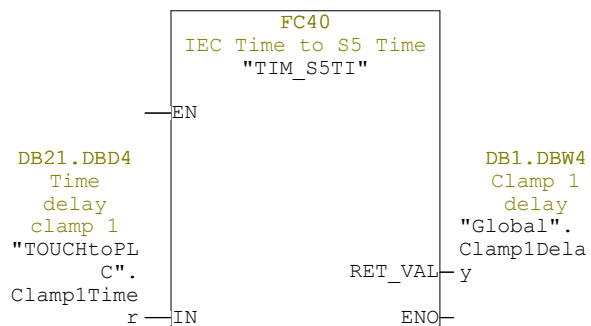
Network: 11 Robot is not E-stopped



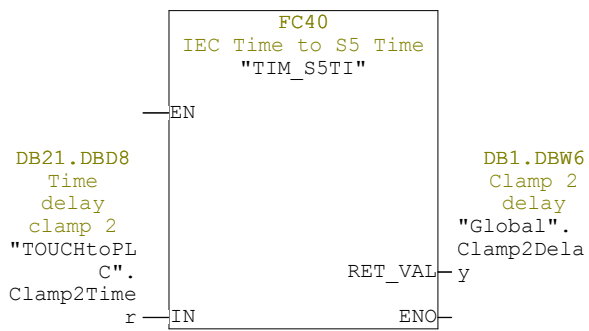
Network: 12 CNC program not running



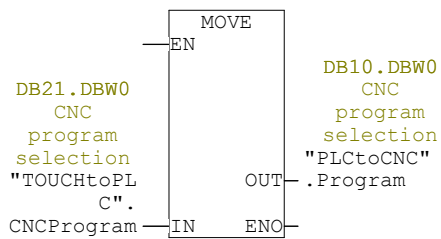
Network: 13



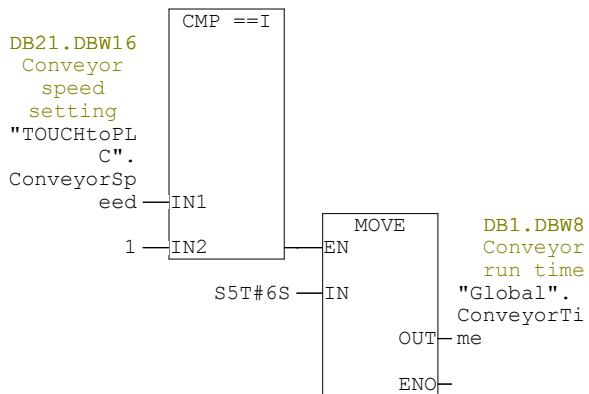
Network: 14



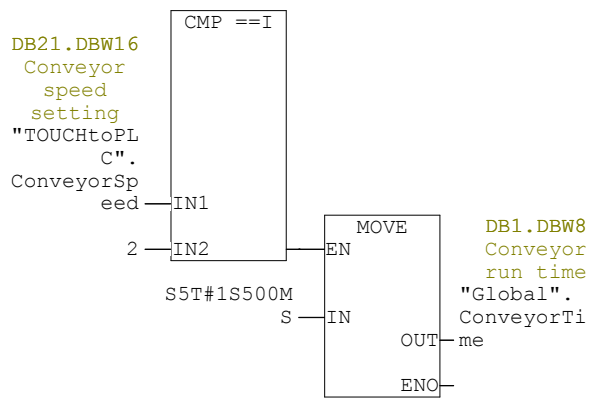
Network: 15



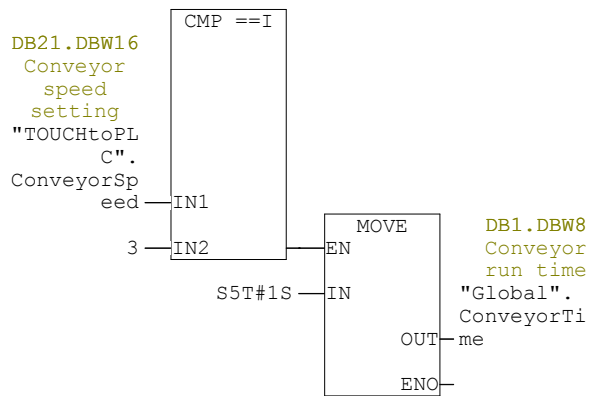
Network: 16



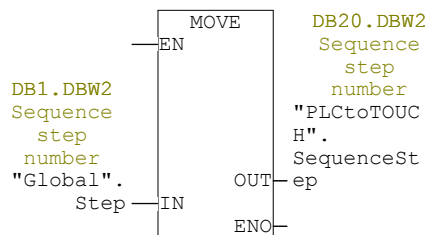
Network: 17



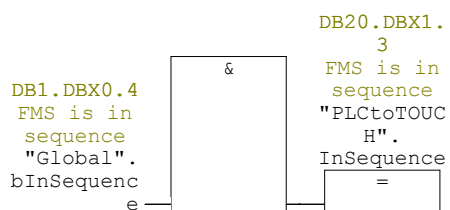
Network: 18



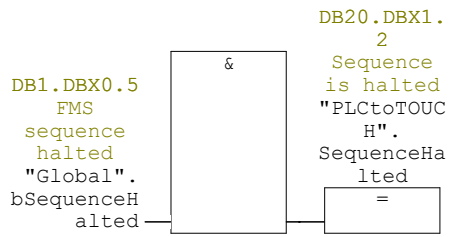
Network: 19



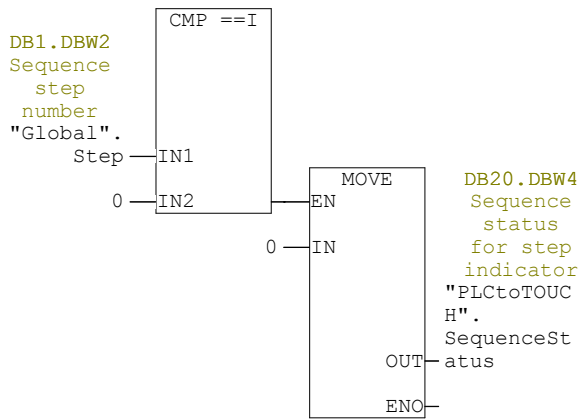
Network: 20 FMS is in sequence



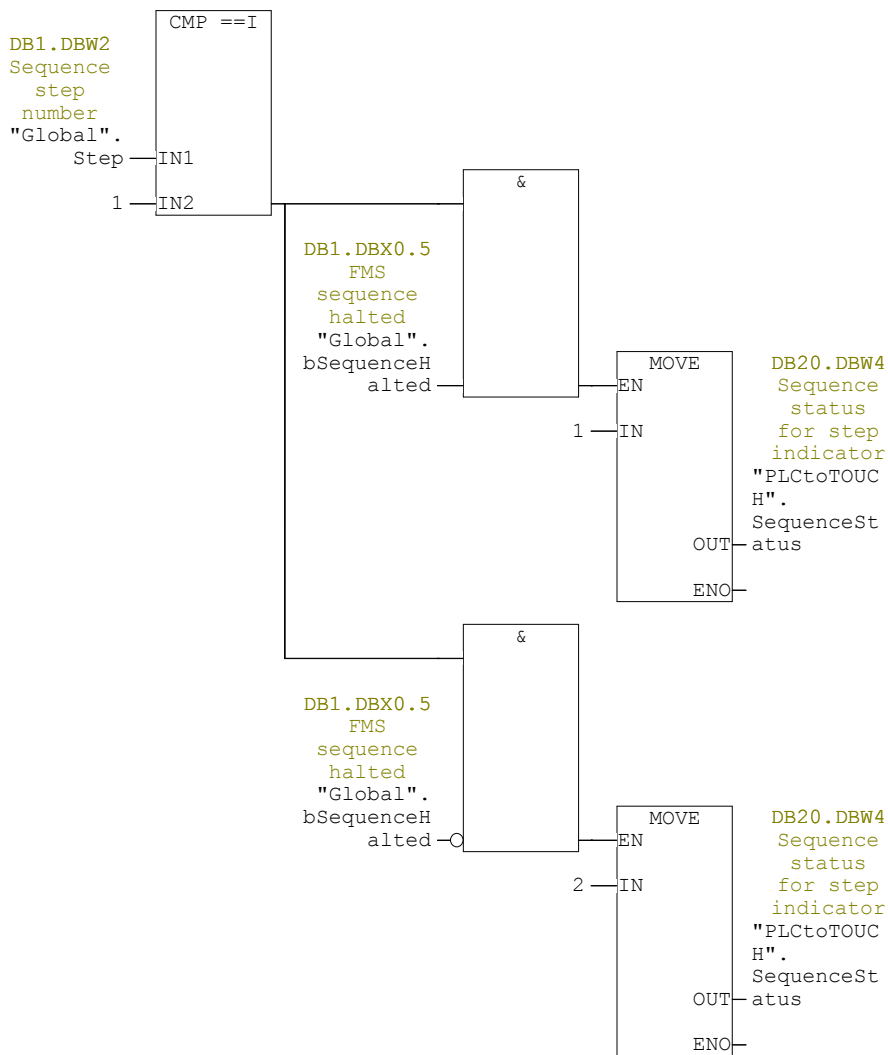
Network: 21 Sequence is halted



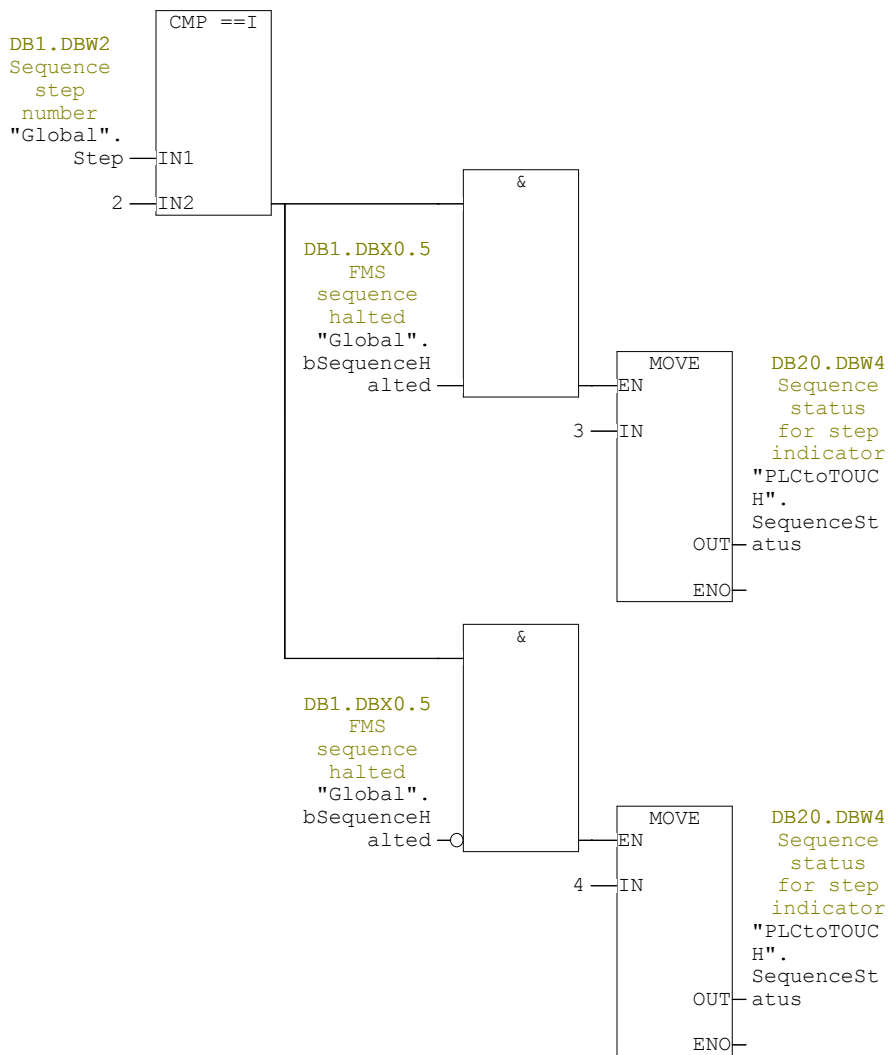
Network: 22



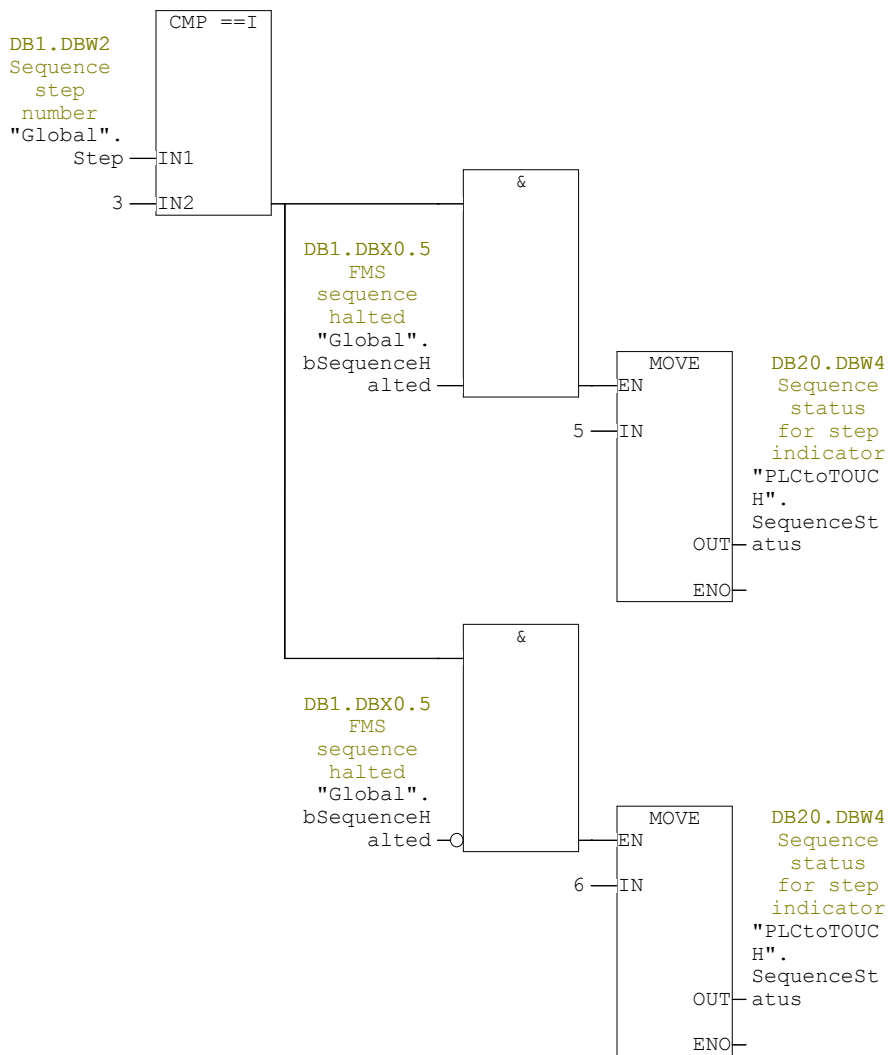
Network: 23 Sequence status step 1 clamps open



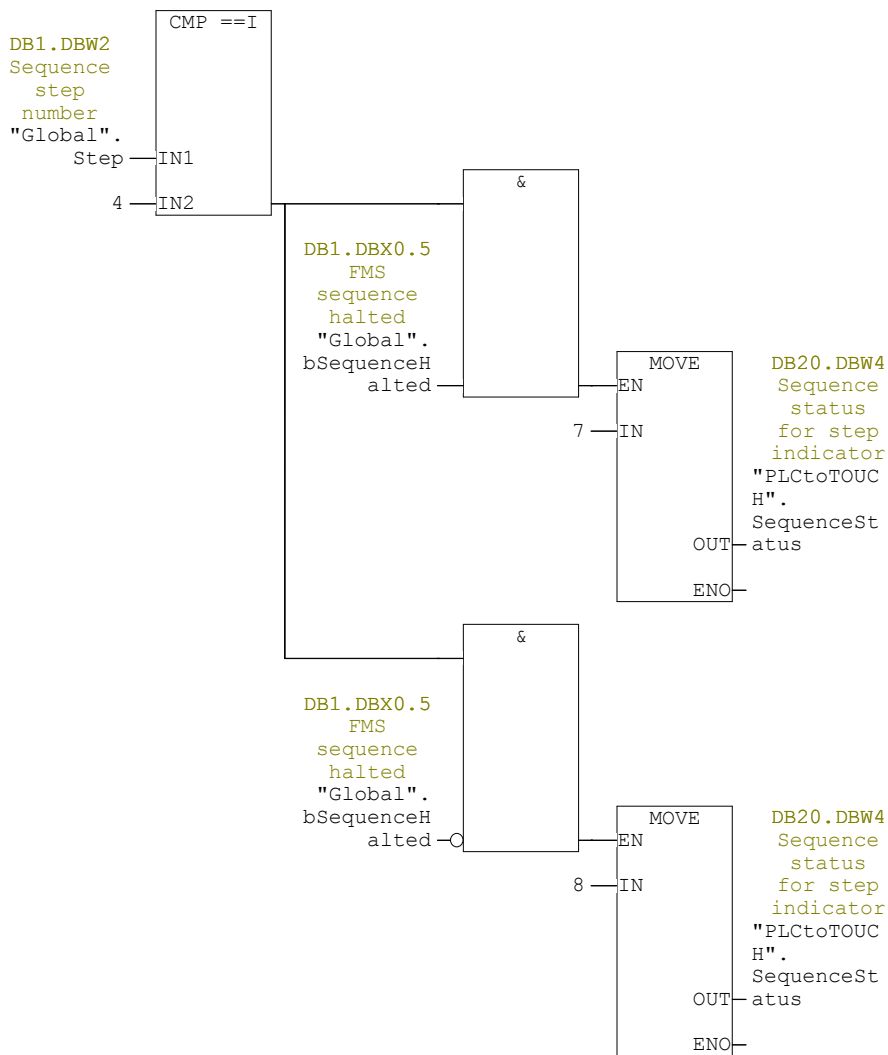
Network: 24 Sequence status step 2 robot move in



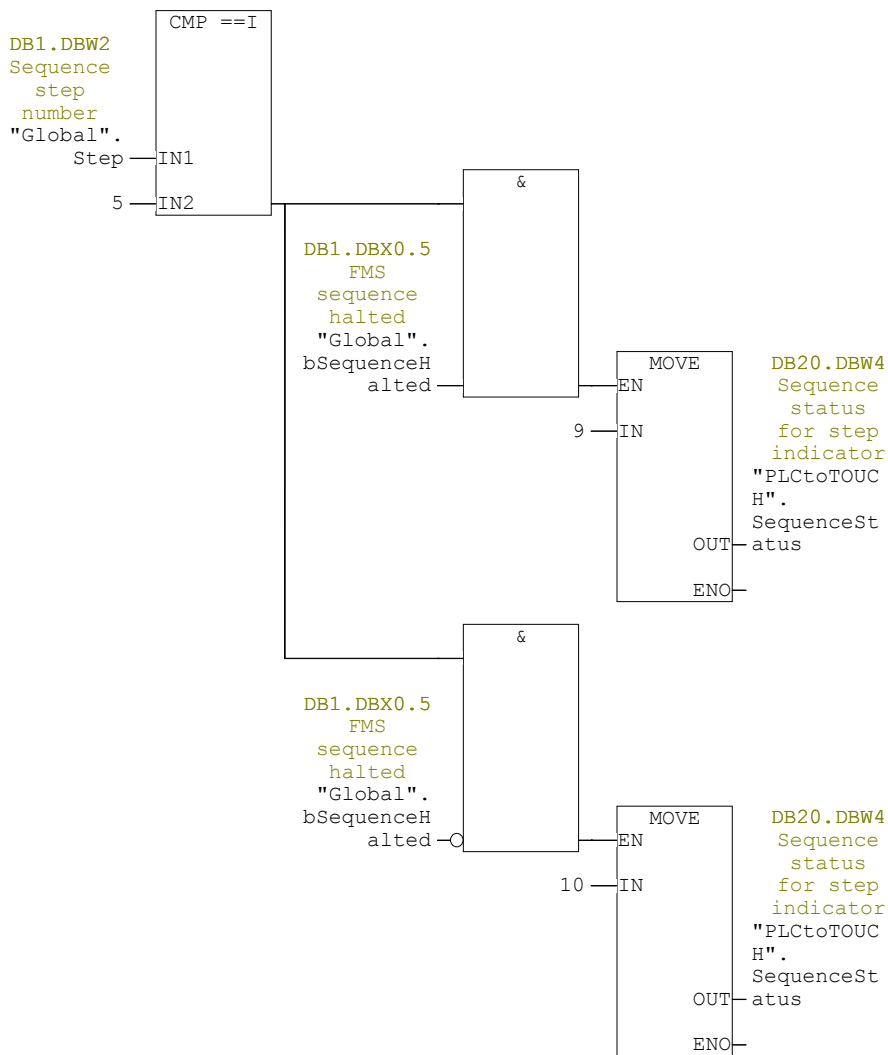
Network: 25 Sequence status step 3 clamps close



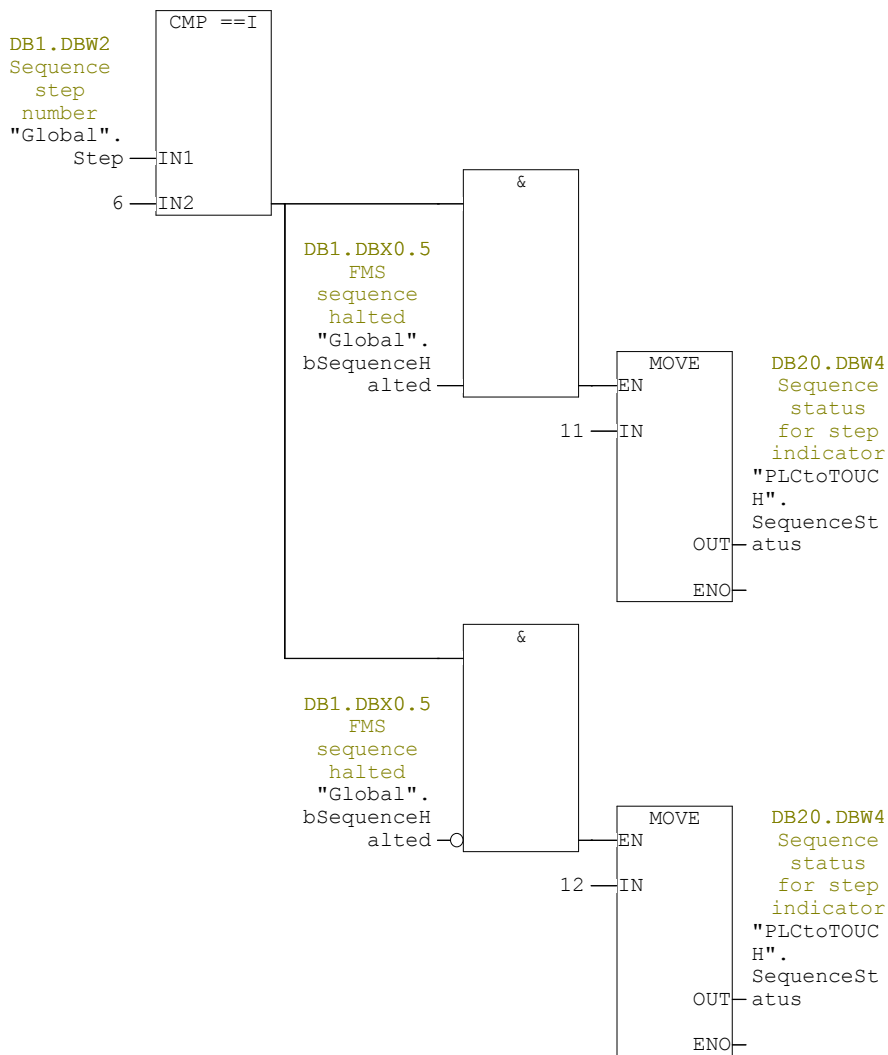
Network: 26 Sequence status step 4 CNC run



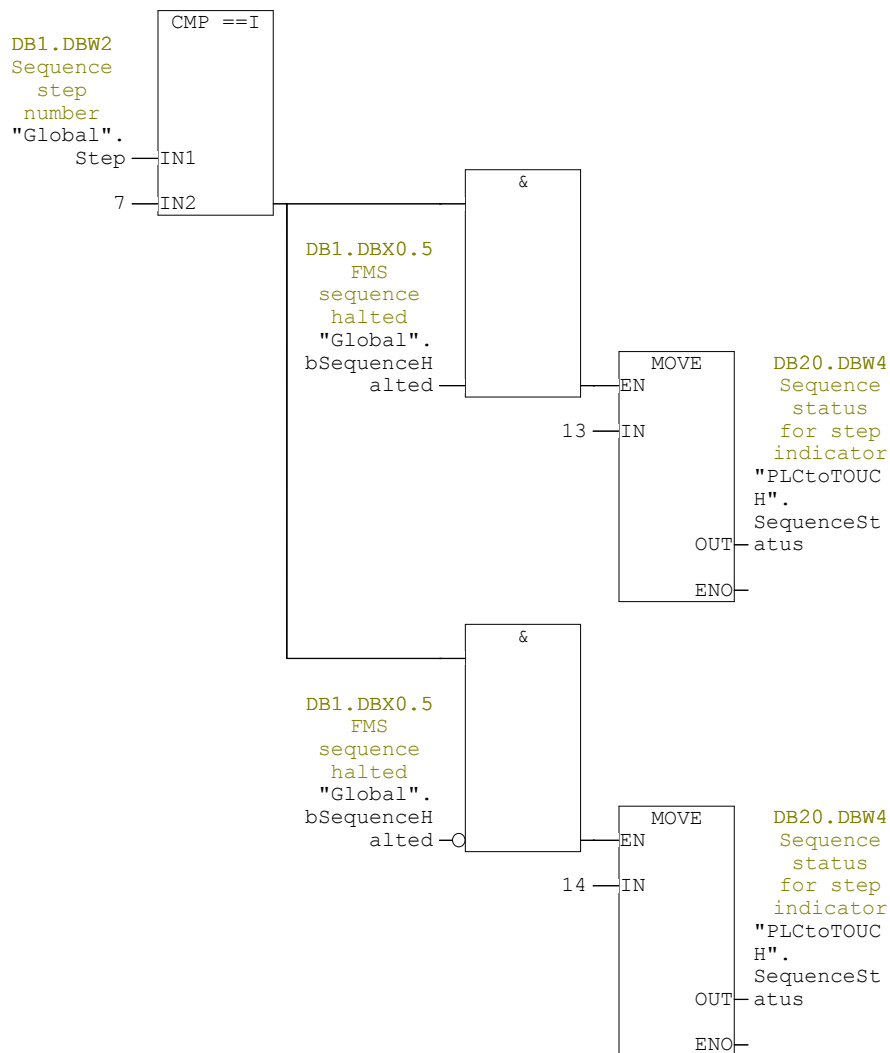
Network: 27 Sequence status step 5 clamps open



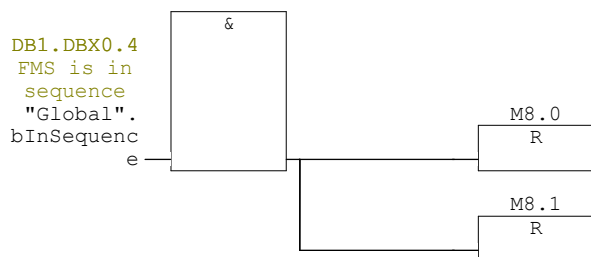
Network: 28 Sequence status step 6 robot move out



Network: 29 Sequence status step 7 conveyor run



Network: 30 Reset manual clamps bits when running sequence



FC10 - <offline>

"StepperMotor"

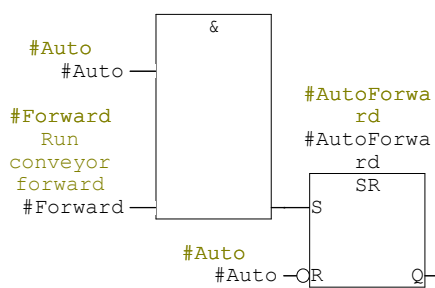
Name:
Author:
Time stamp Code:
Interface:
Lengths (block/logic/data):

Family:
Version: 0.1
Block version: 2
 05/08/2019 02:05:29 PM
 05/08/2019 02:05:29 PM
 00452 00322 00004

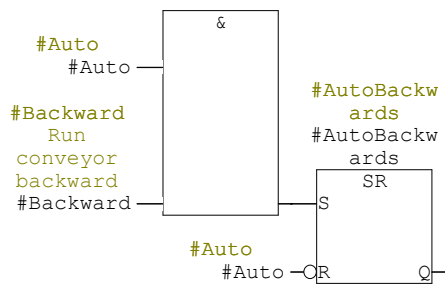
Name	Data Type	Address	Comment
IN		0.0	
Forward	Bool	0.0	Run conveyor forward
Backward	Bool	0.1	Run conveyor backward
Auto	Bool	0.2	
Speed	Int	2.0	Speed setting 1-3
OUT		0.0	
Finished	Bool	4.0	
IN_OUT		0.0	
TEMP		0.0	
Elapsed	S5Time	0.0	
AutoForward	Bool	2.0	
AutoBackwards	Bool	2.1	
RETURN		0.0	
RET_VAL		0.0	

Block: FC10

Network: 1

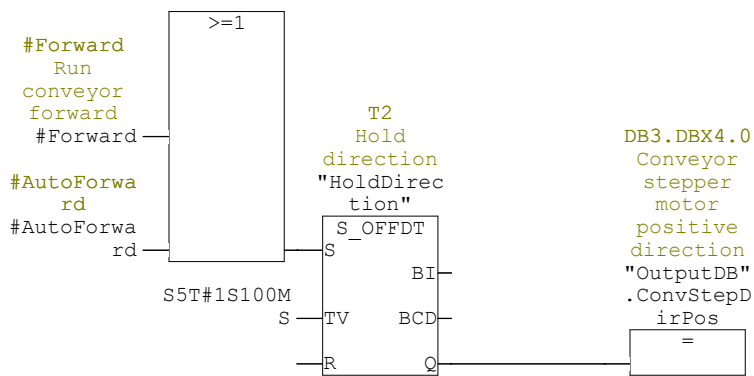


Network: 2 Conveyor stepper motor speed medium (+ slow = fast)

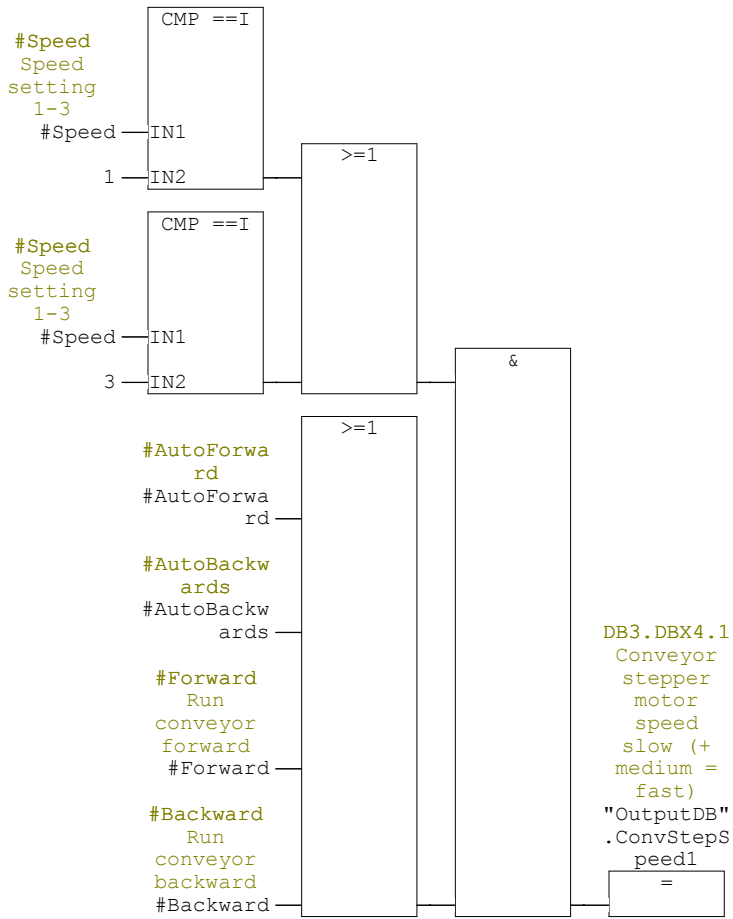


Network: 3 Conveyor stepper motor positive direction

Direction control with off delay timer to keep direction high during ramp down

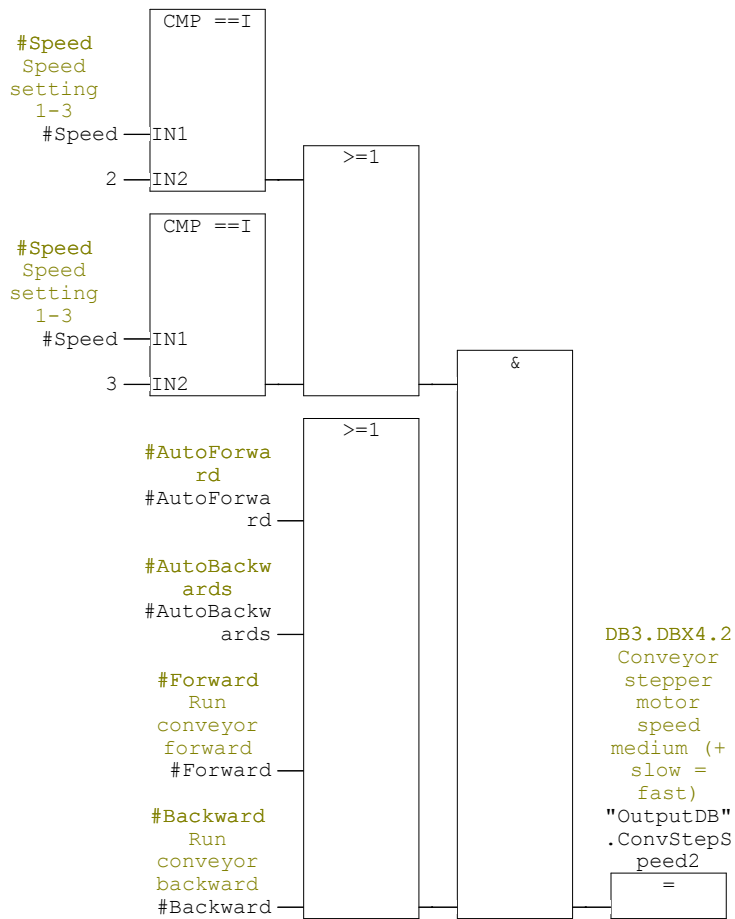


Network: 4 Conveyor stepper motor speed control 1
Speed control 1

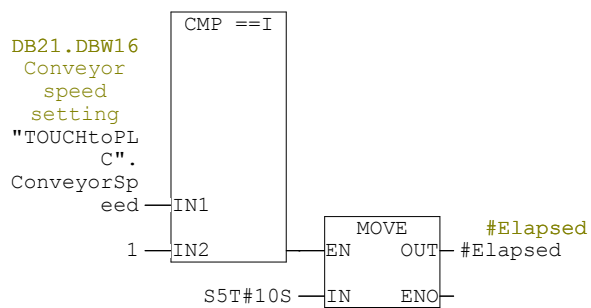


Network: 5 Conveyor stepper motor speed control 2

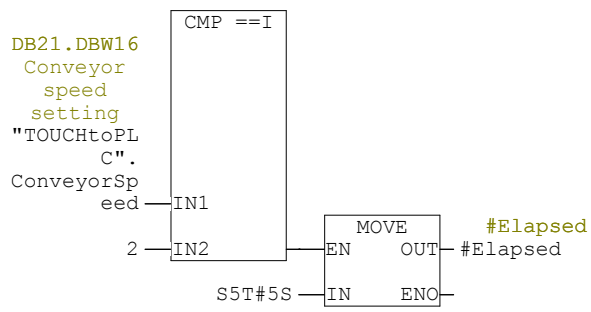
Speed control 2



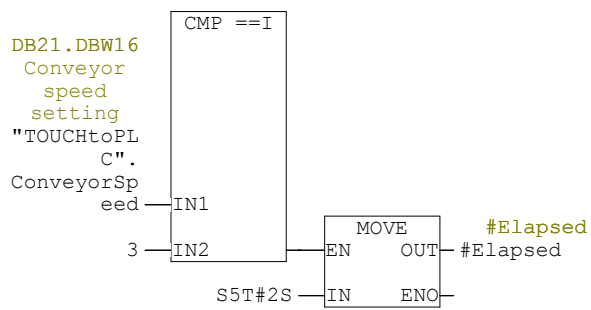
Network: 6



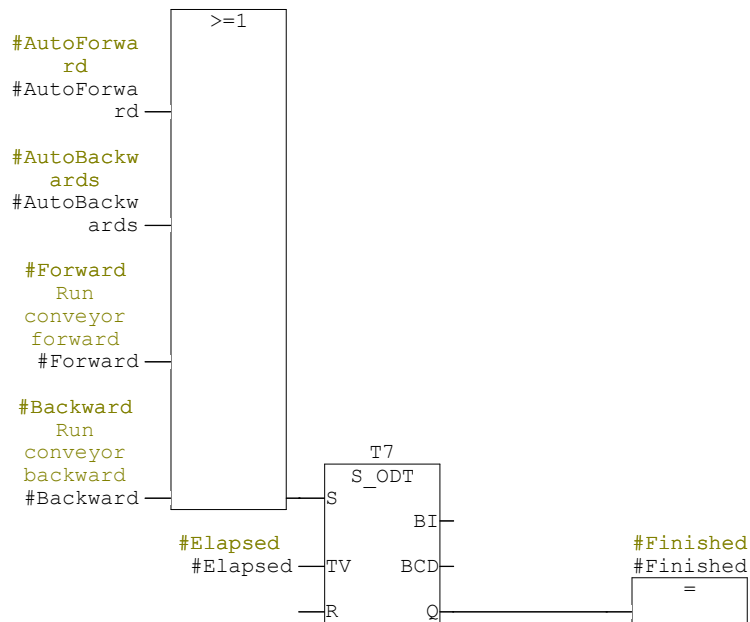
Network: 7



Network: 8



Network: 9 Conveyor stepper motor speed slow (+ medium = fast)



FC11 - <offline>

"Clamps"

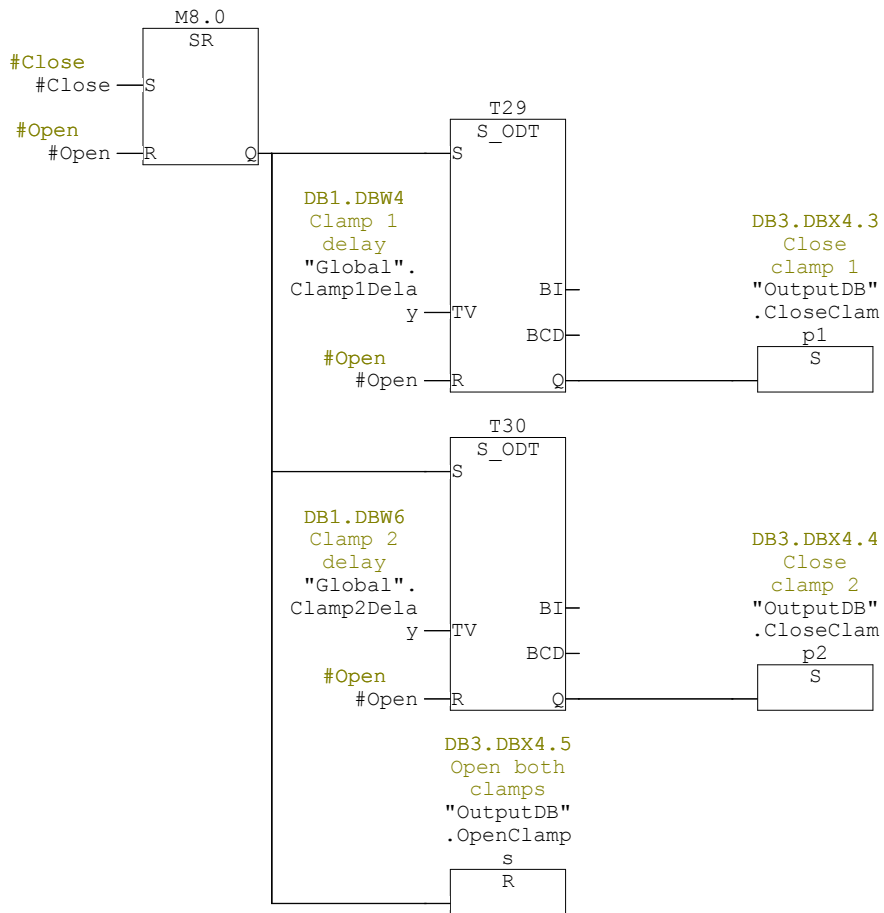
Name:
Author:
Time stamp Code:
Interface:
Lengths (block/logic/data):

Family:
Version: 0.1
Block version: 2
05/13/2019 06:34:13 PM
05/13/2019 06:21:39 PM
00232 00124 00006

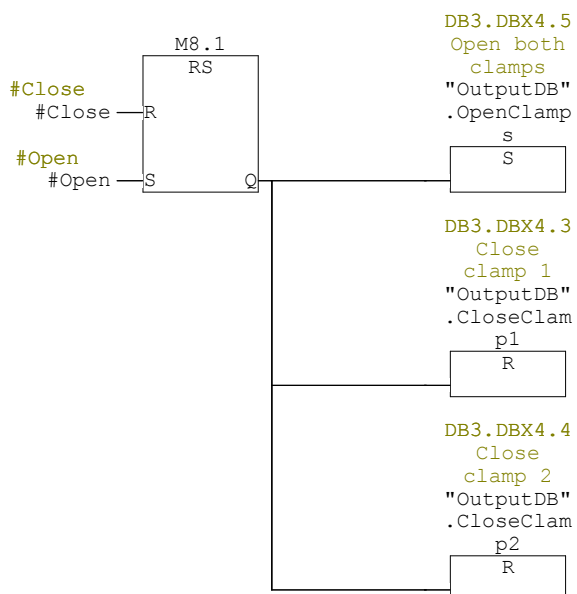
Name	Data Type	Address	Comment
IN		0.0	
Close	Bool	0.0	
Open	Bool	0.1	
OUT		0.0	
IN_OUT		0.0	
TEMP		0.0	
Clamp1Time	S5Time	0.0	
Clamp2Time	S5Time	2.0	
RETURN		0.0	
RET_VAL		0.0	

Block: FC11

Network: 1 Close clamps



Network: 2 Open both clamps



FC12 - <offline>

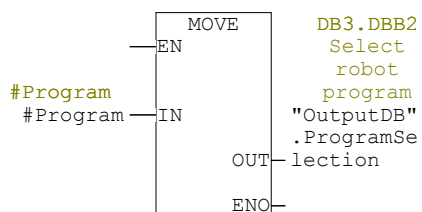
"Robot"

Name:
Author:
Family:
Version: 0.1
Block version: 2
Time stamp Code: 05/09/2019 07:18:01 PM
Interface: 05/06/2019 03:01:59 PM
Lengths (block/logic/data): 00274 00156 00002

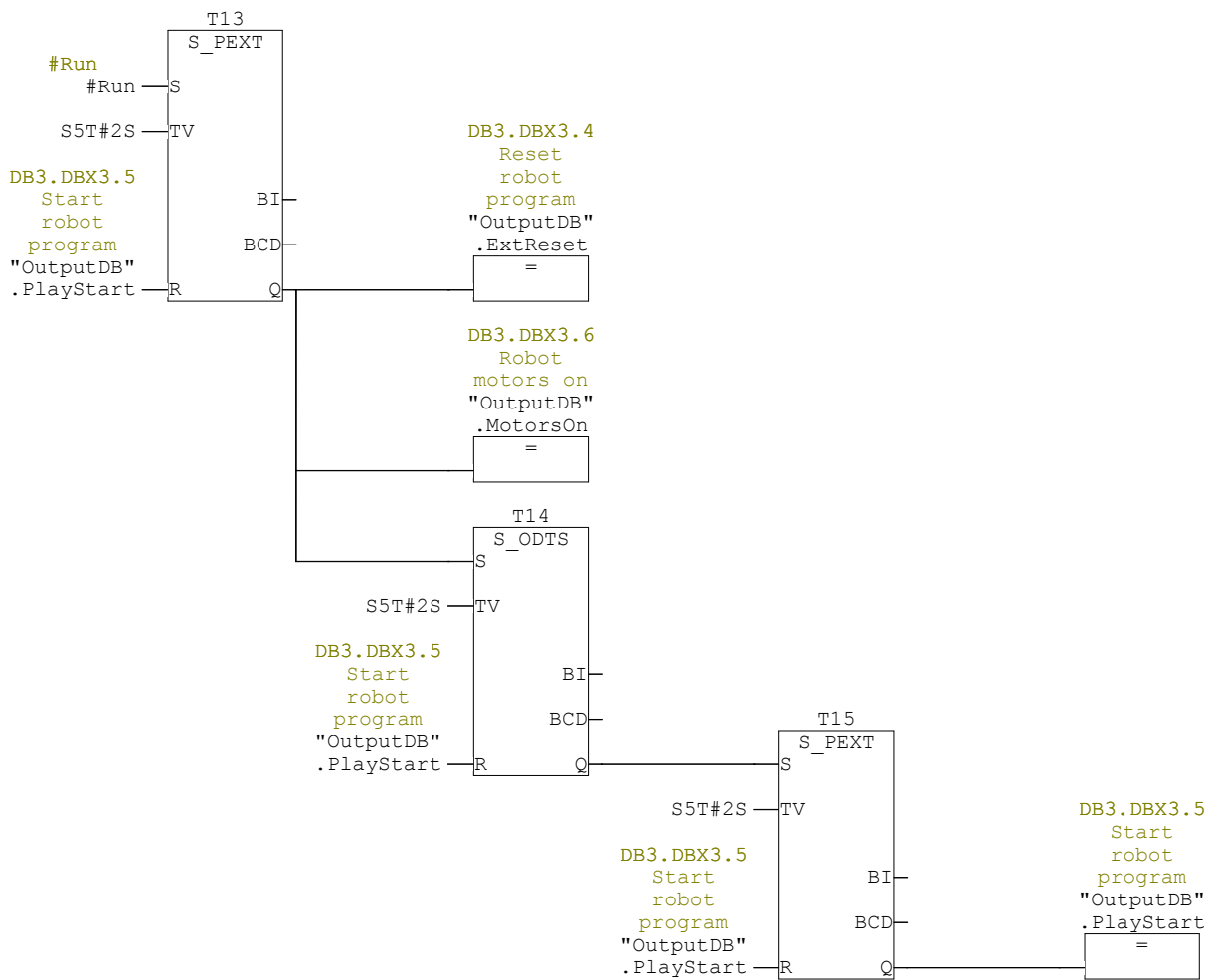
Name	Data Type	Address	Comment
IN		0.0	
Program	Int	0.0	
Run	Bool	2.0	
OUT		0.0	
Finished	Bool	4.0	
IN_OUT		0.0	
TEMP		0.0	
Reset	Bool	0.0	
ProgEnd	Bool	0.1	
ProgStatus	Bool	0.2	
Dummy	Bool	0.3	
RETURN		0.0	
RET_VAL		0.0	

Block: FC12

Network: 1 Select program



Network: 2

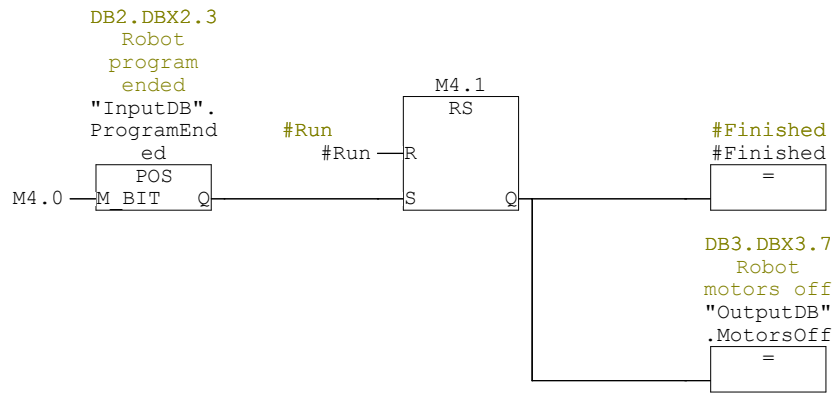


Network: 3 Reset robot program and start

Network 2 and 3 timing: Send reset and motors on signal for 2 sec. Then send program start.

Network: 4 Reset robot program and start

Network: 5 Robot motors on



FC13 - <offline>

"CNC"

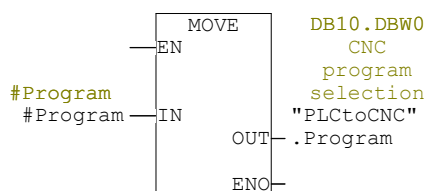
Name:
Author:
Time stamp Code:
Lengths (block/logic/data):

Family:
Version: 0.1
Block version: 2
 05/09/2019 07:08:45 PM
 05/09/2019 01:18:55 PM
 00228 00116 00002

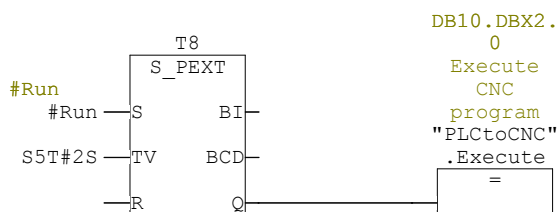
Name	Data Type	Address	Comment
IN		0.0	
Program	Int	0.0	
Run	Bool	2.0	
Initialize	Bool	2.1	
Park	Bool	2.2	
OUT		0.0	
Finished	Bool	4.0	
IN_OUT		0.0	
TEMP		0.0	
Dummy	Bool	0.0	
RETURN		0.0	
RET_VAL		0.0	

Block: FC13

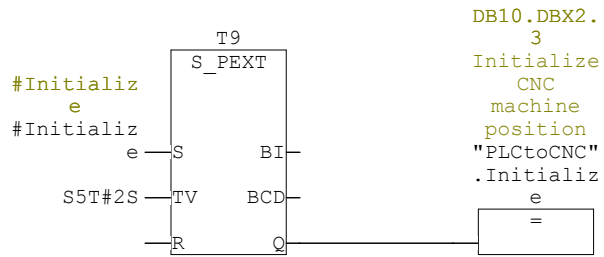
Network: 1 CNC program selection



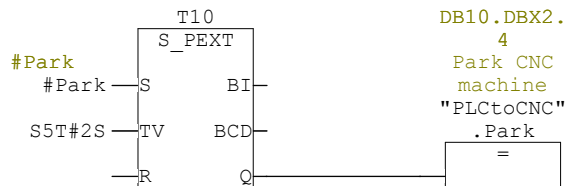
Network: 2 Execute CNC program



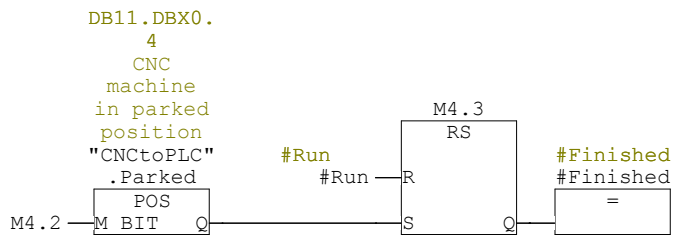
Network: 3 Execute CNC program



Network: 4 Initialize CNC machine position



Network: 5



FC15 - <offline>

"Sequence"

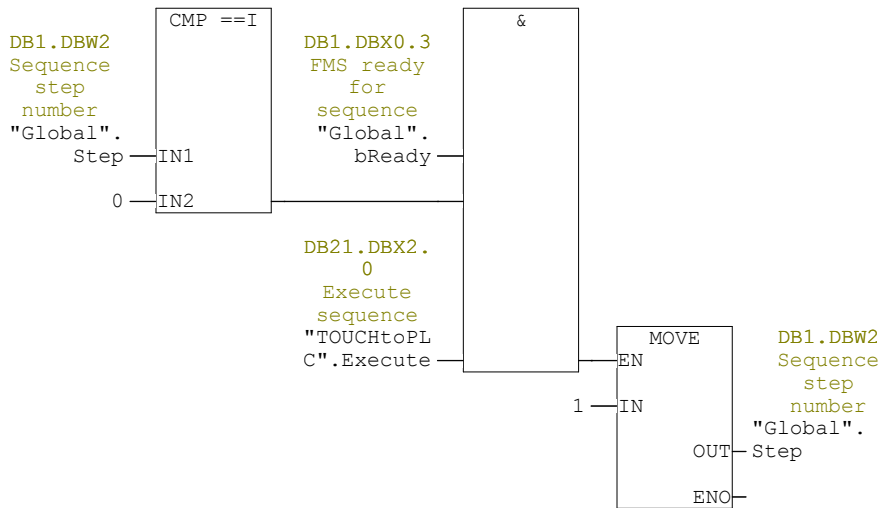
Name:
Author:
Time stamp Code:
Lengths (block/logic/data):

Family:
Version: 0.1
Block version: 2
 05/13/2019 02:34:25 PM
 05/10/2019 11:38:32 AM
 01874 01740 00002

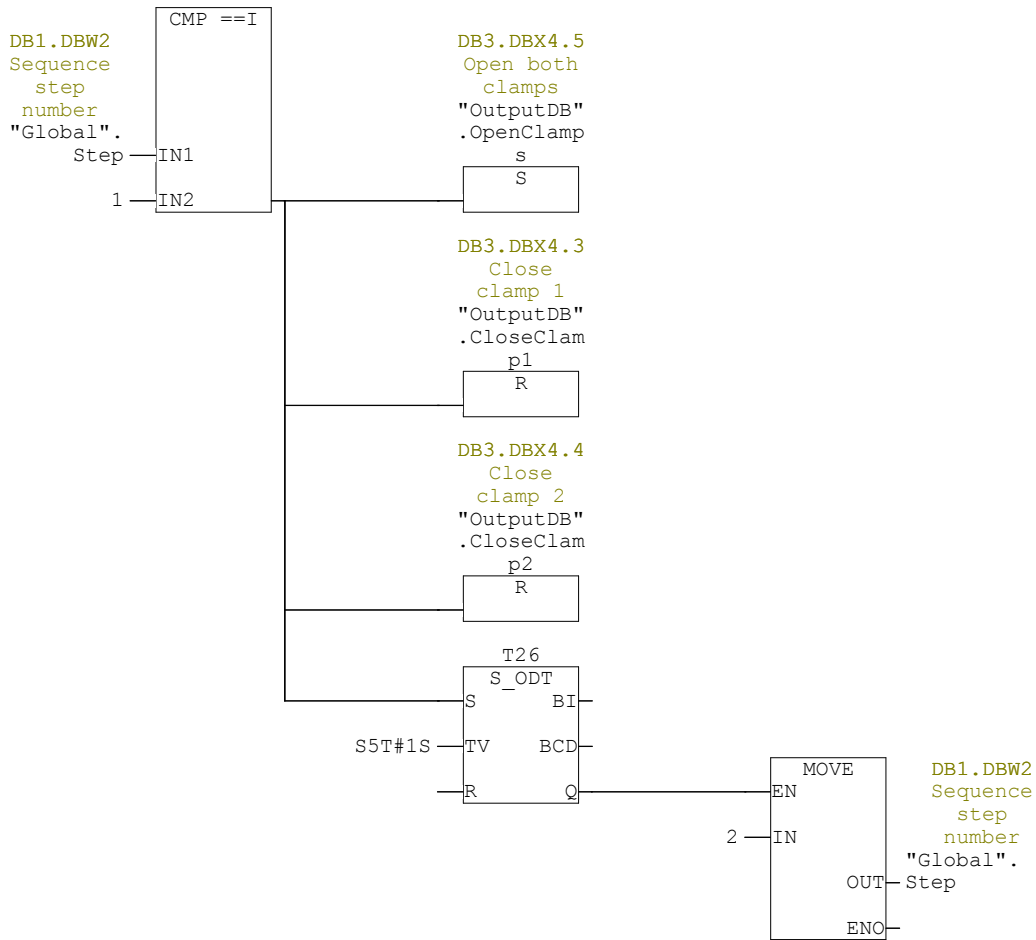
Name	Data Type	Address	Comment
IN		0.0	
OUT		0.0	
IN_OUT		0.0	
TEMP		0.0	
RETURN		0.0	
RET_VAL		0.0	

Block: FC15

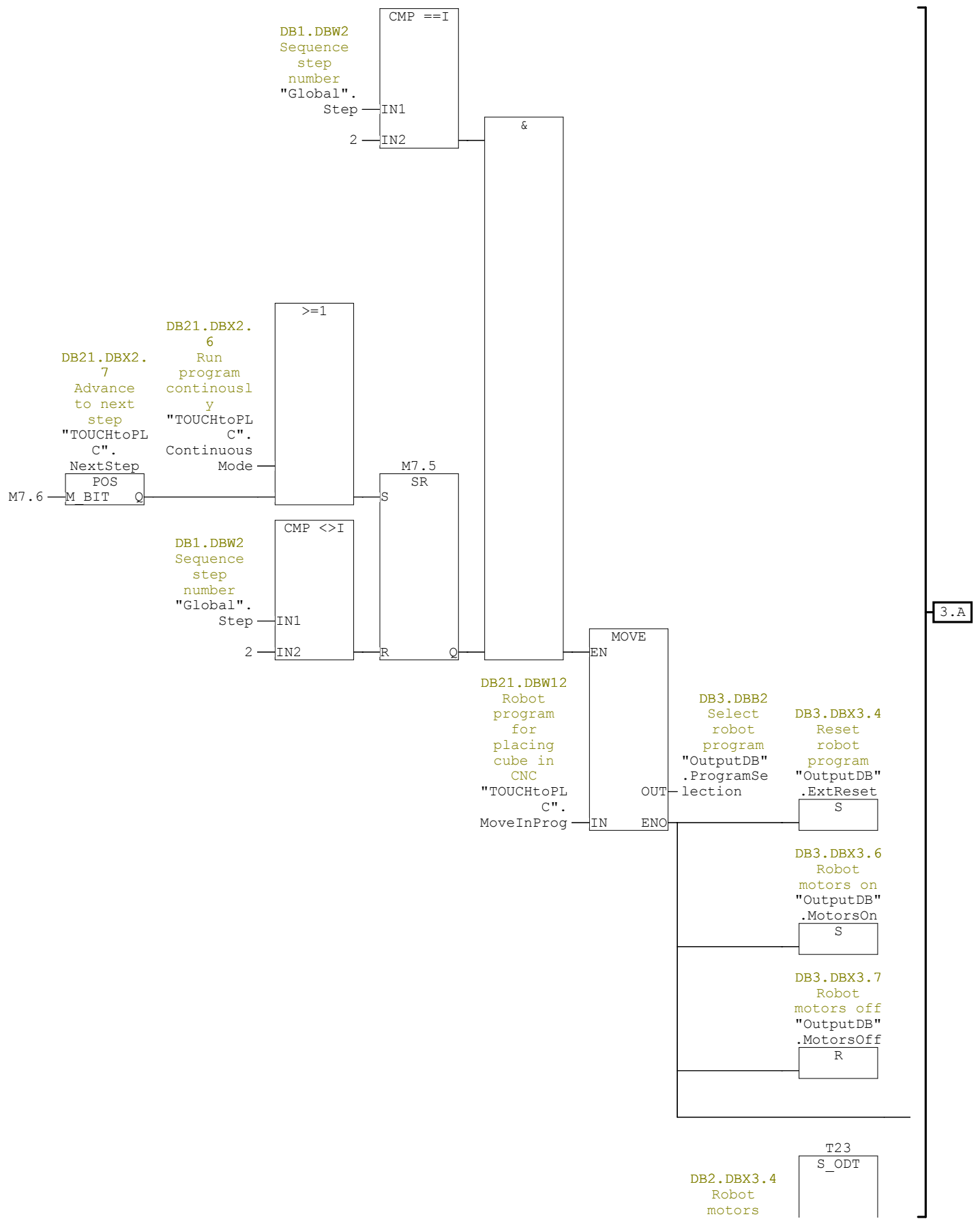
Network: 1



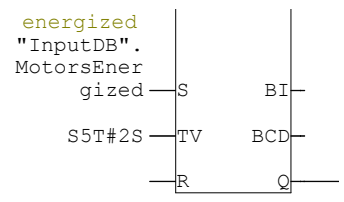
Network: 2 Open both clamps



Network: 3 Run robot program



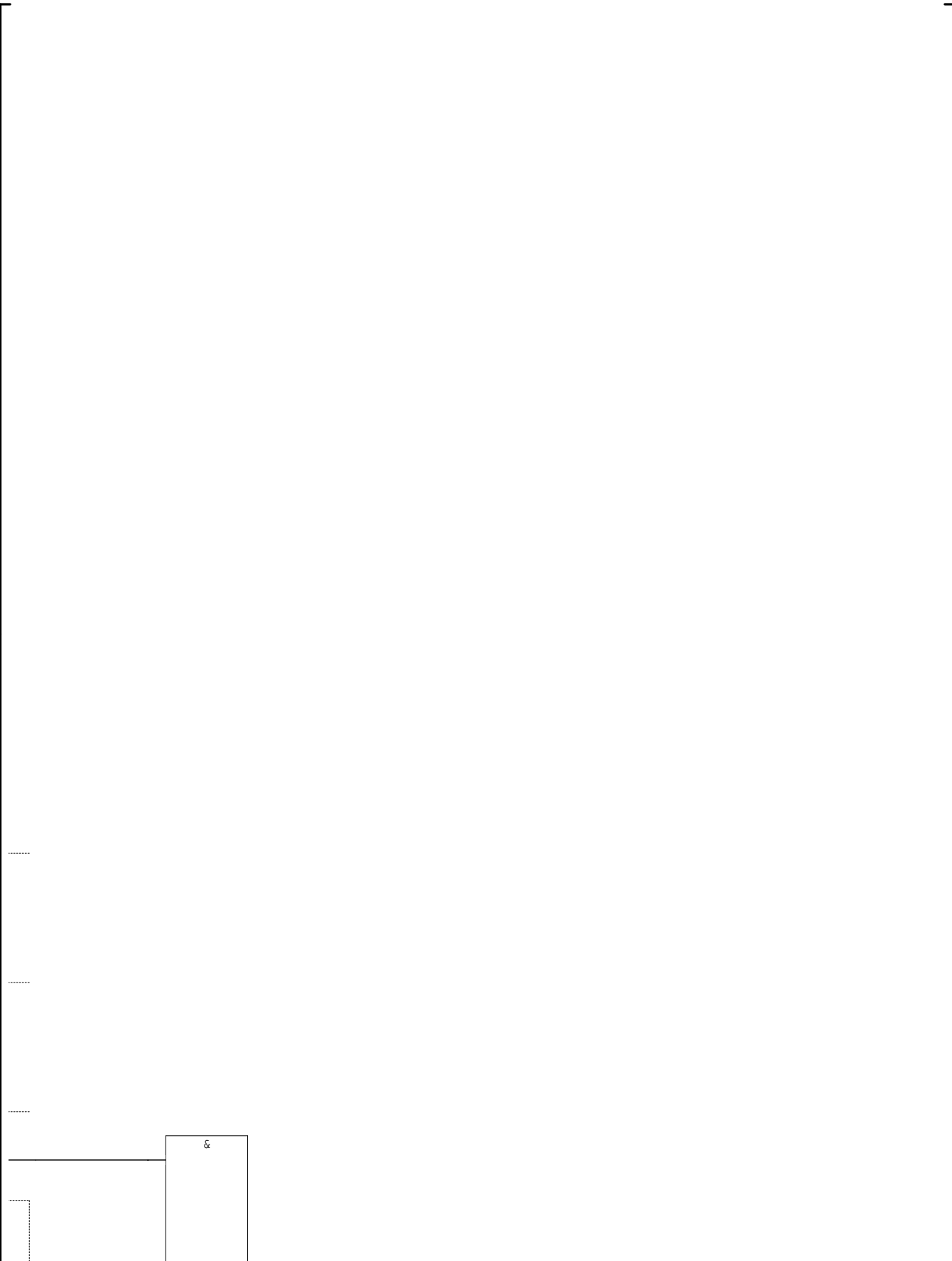
3.A

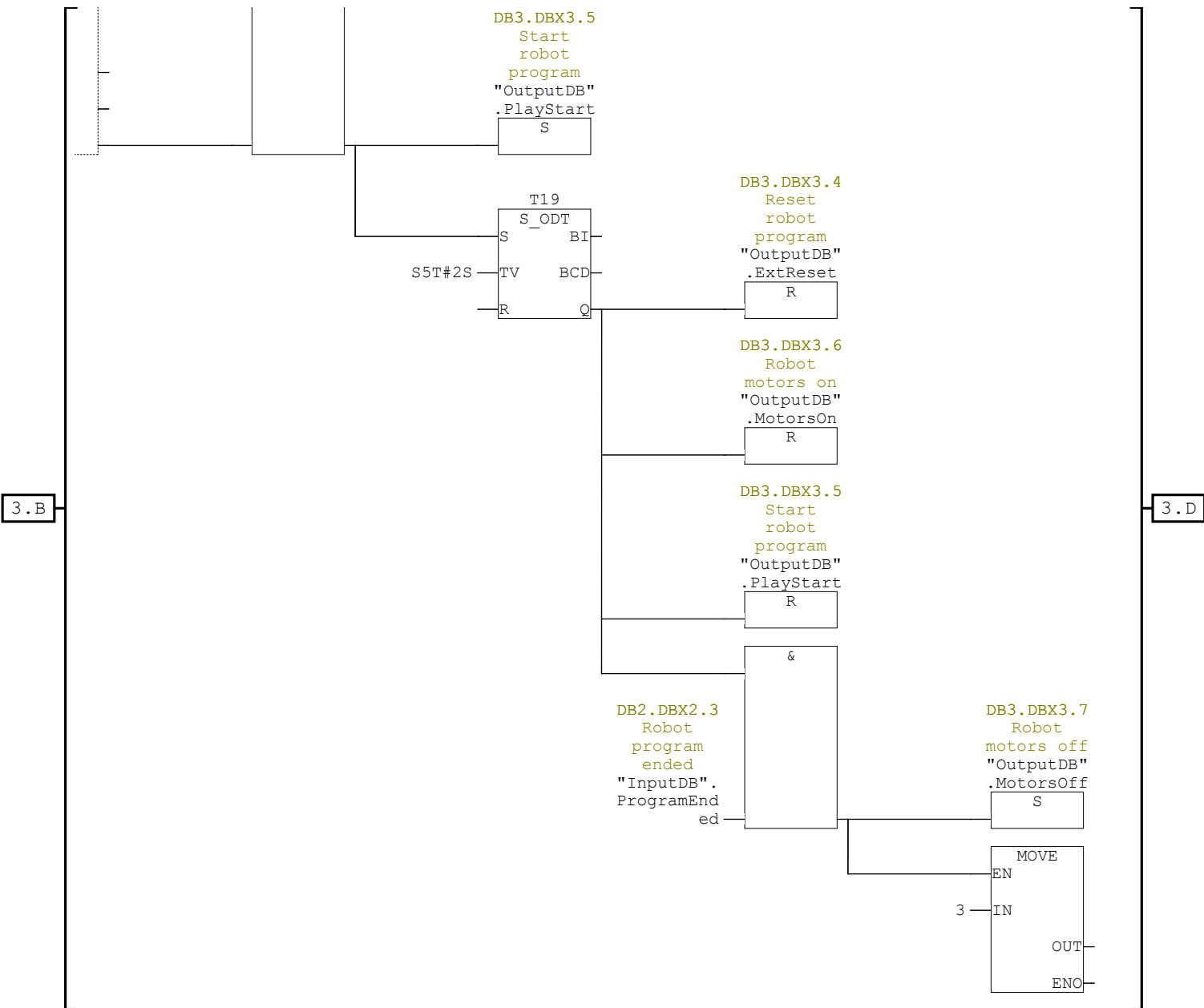


3.B

3.A

3.C



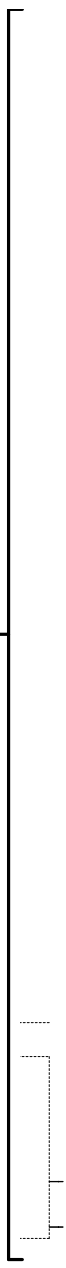


3.B

3.D

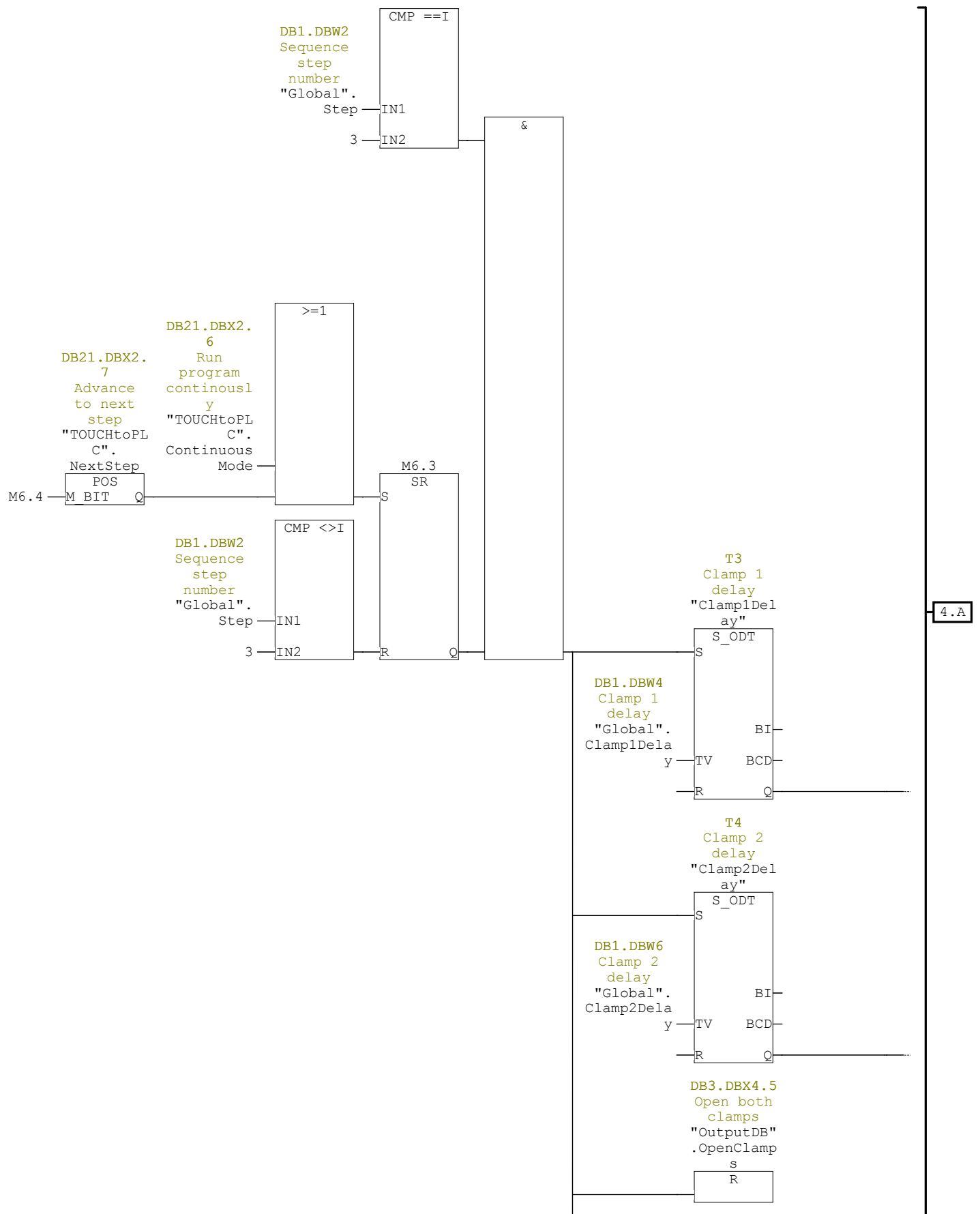
3.C

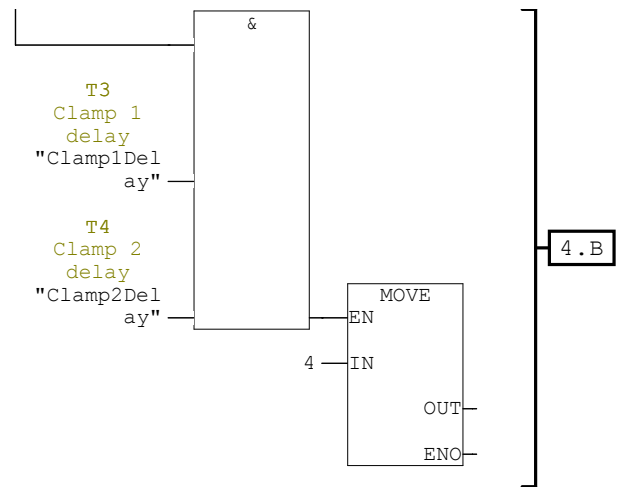
3.D



DB1.DBW2
Sequence
step
number
"Global".
Step

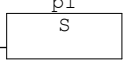
Network: 4 Close clamps



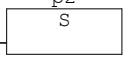


4.A

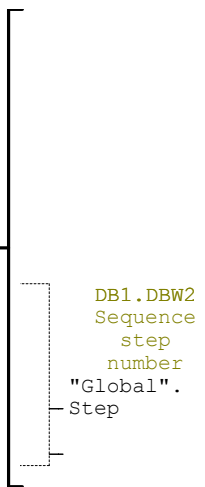
DB3.DBX4.3
Close
clamp 1
"OutputDB"
.CloseClam
p1



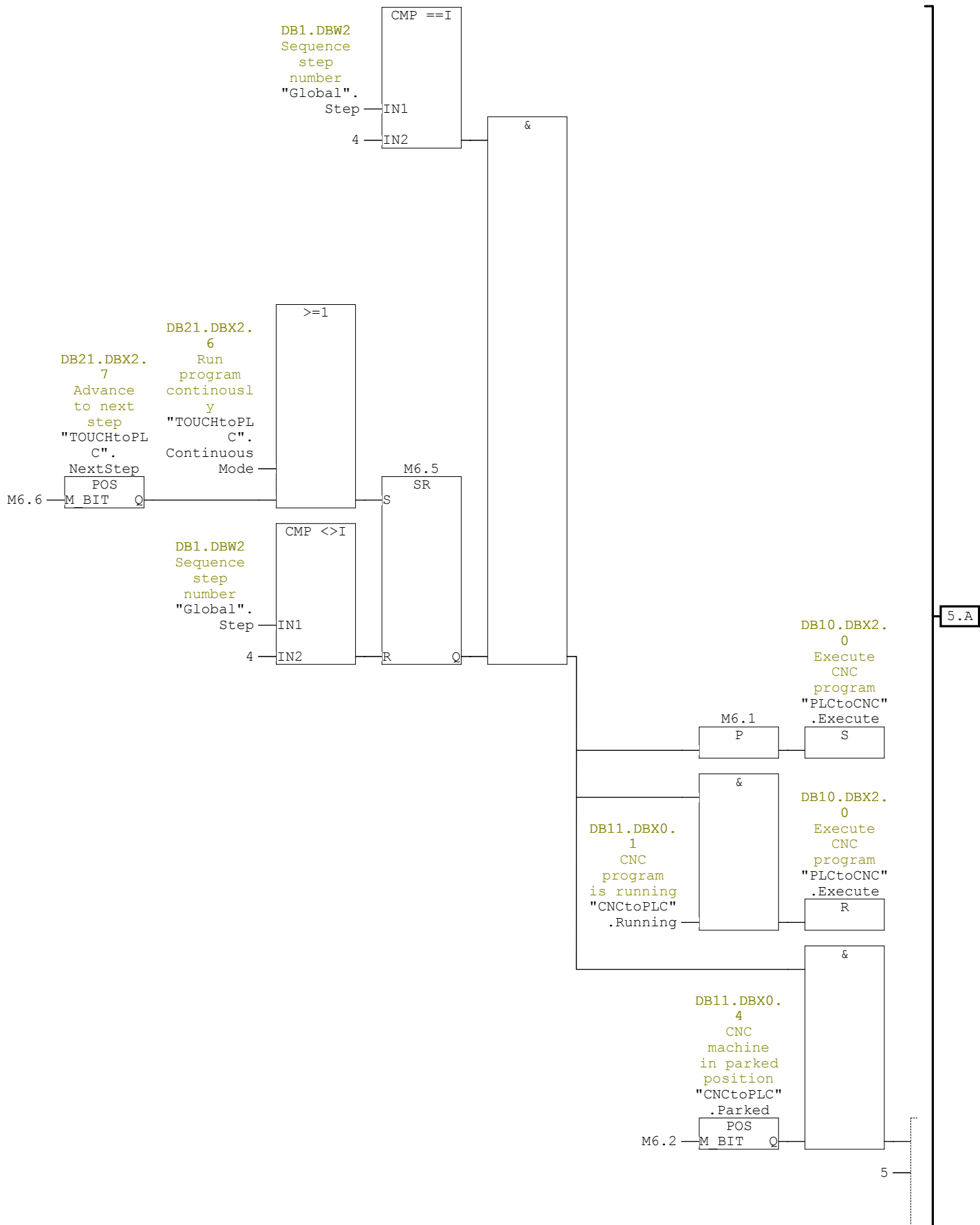
DB3.DBX4.4
Close
clamp 2
"OutputDB"
.CloseClam
p2



4.B

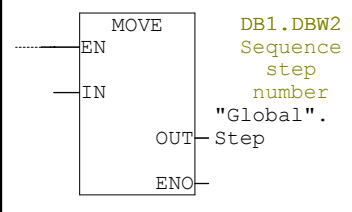


Network: 5 Run CNC program

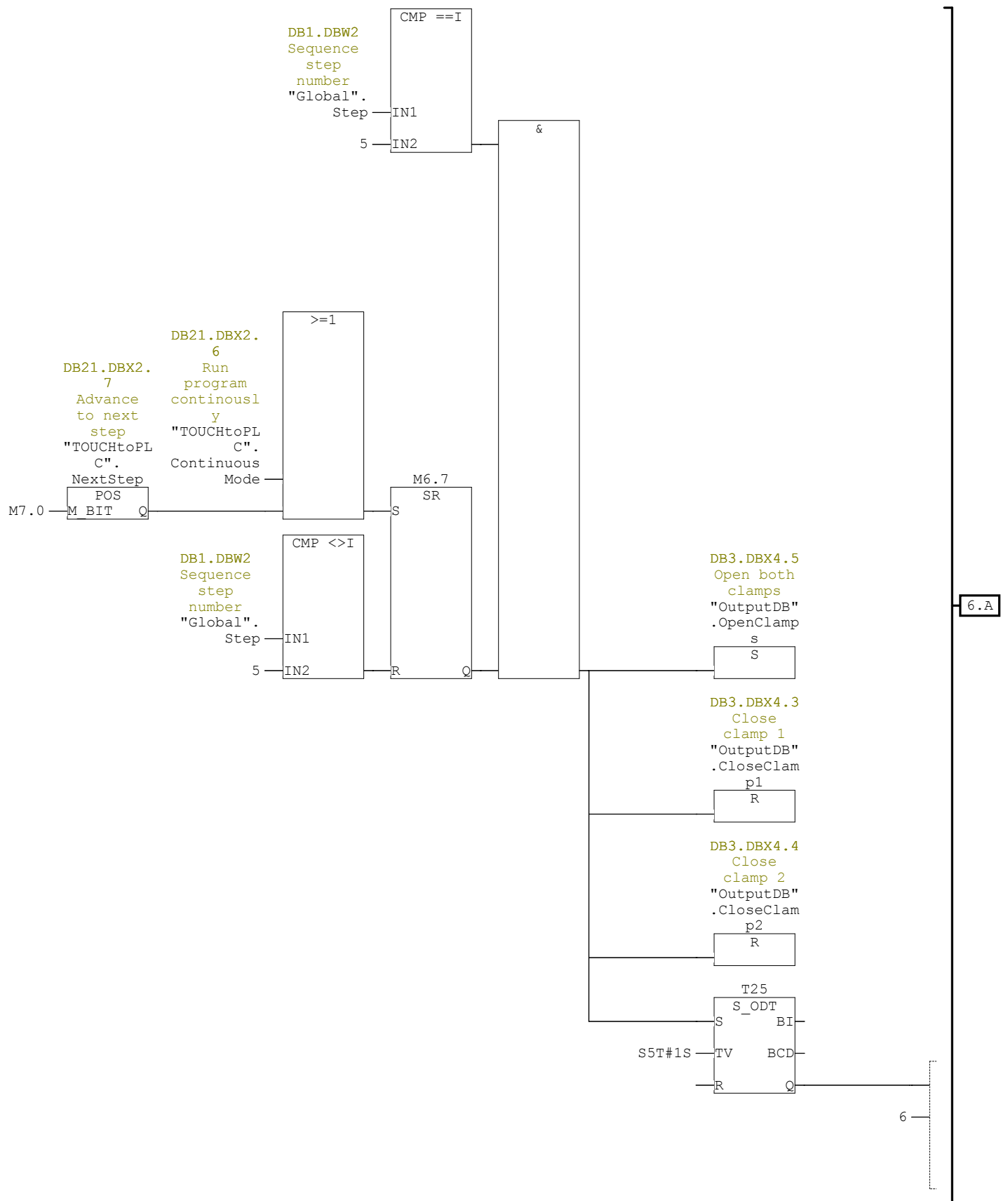


5.B

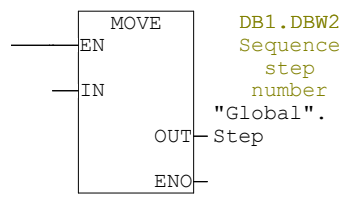
5.A



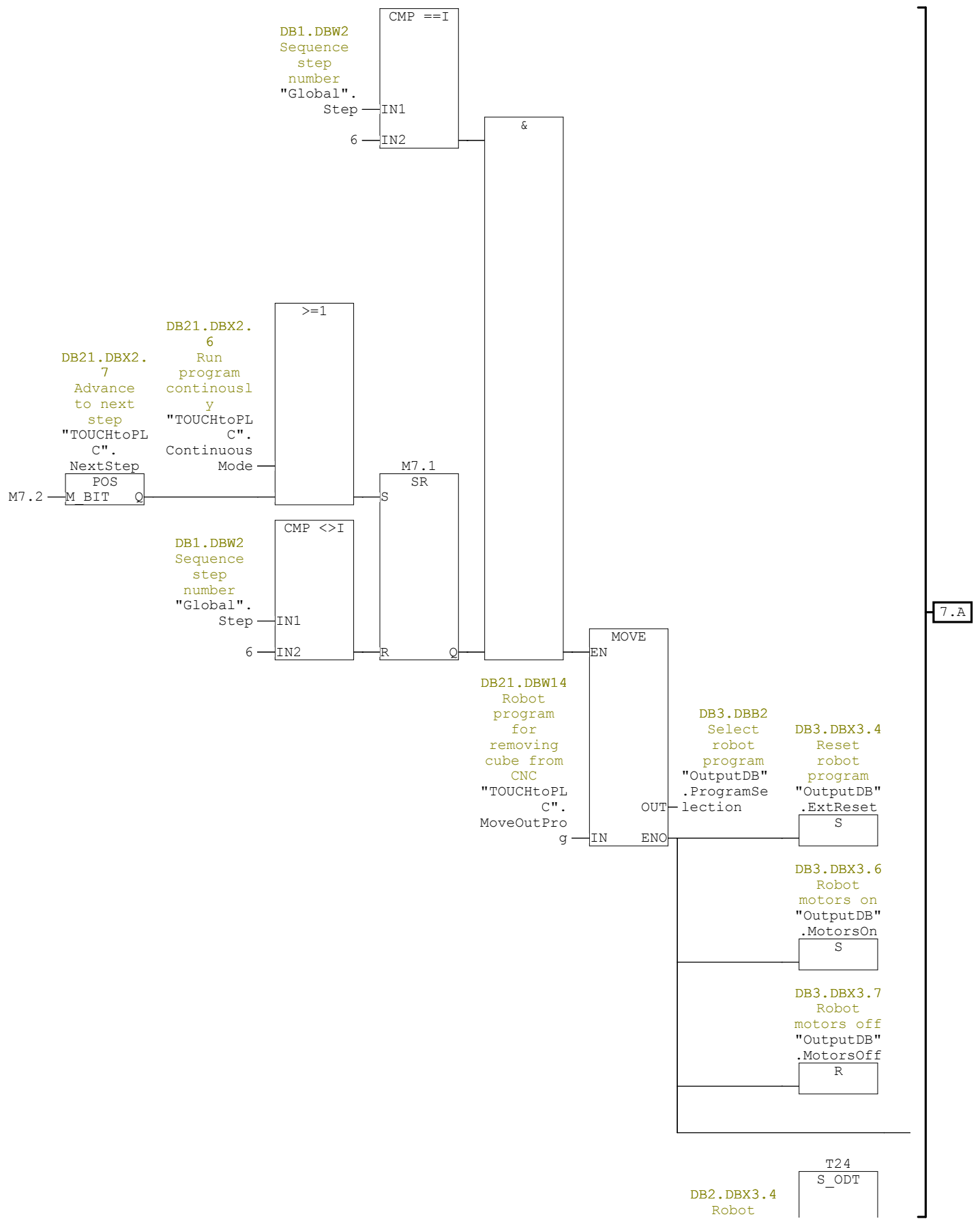
Network: 6 Open both clamps



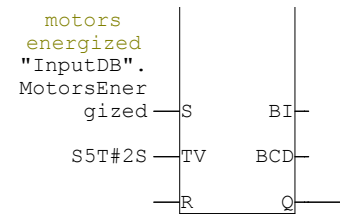
6.A



Network: 7 Run robot program



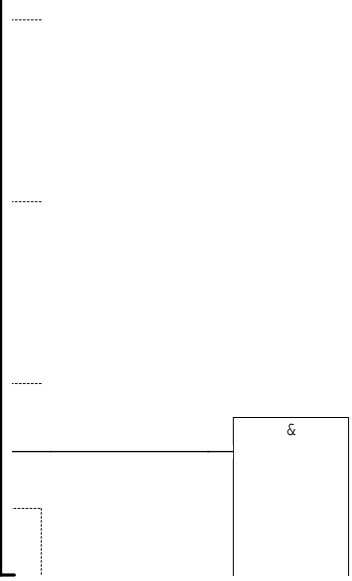
7.A

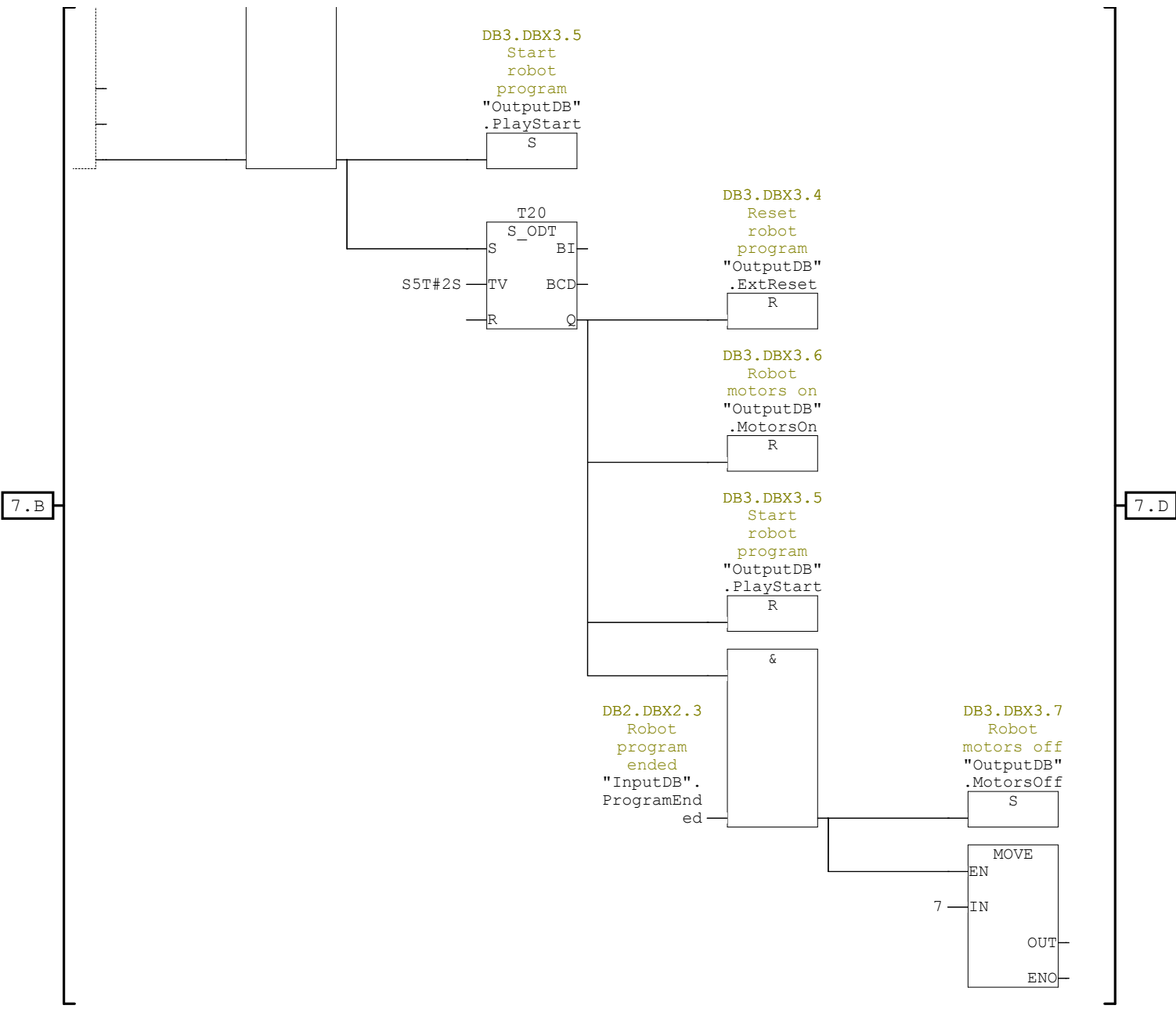


7.B

7.A

7.C



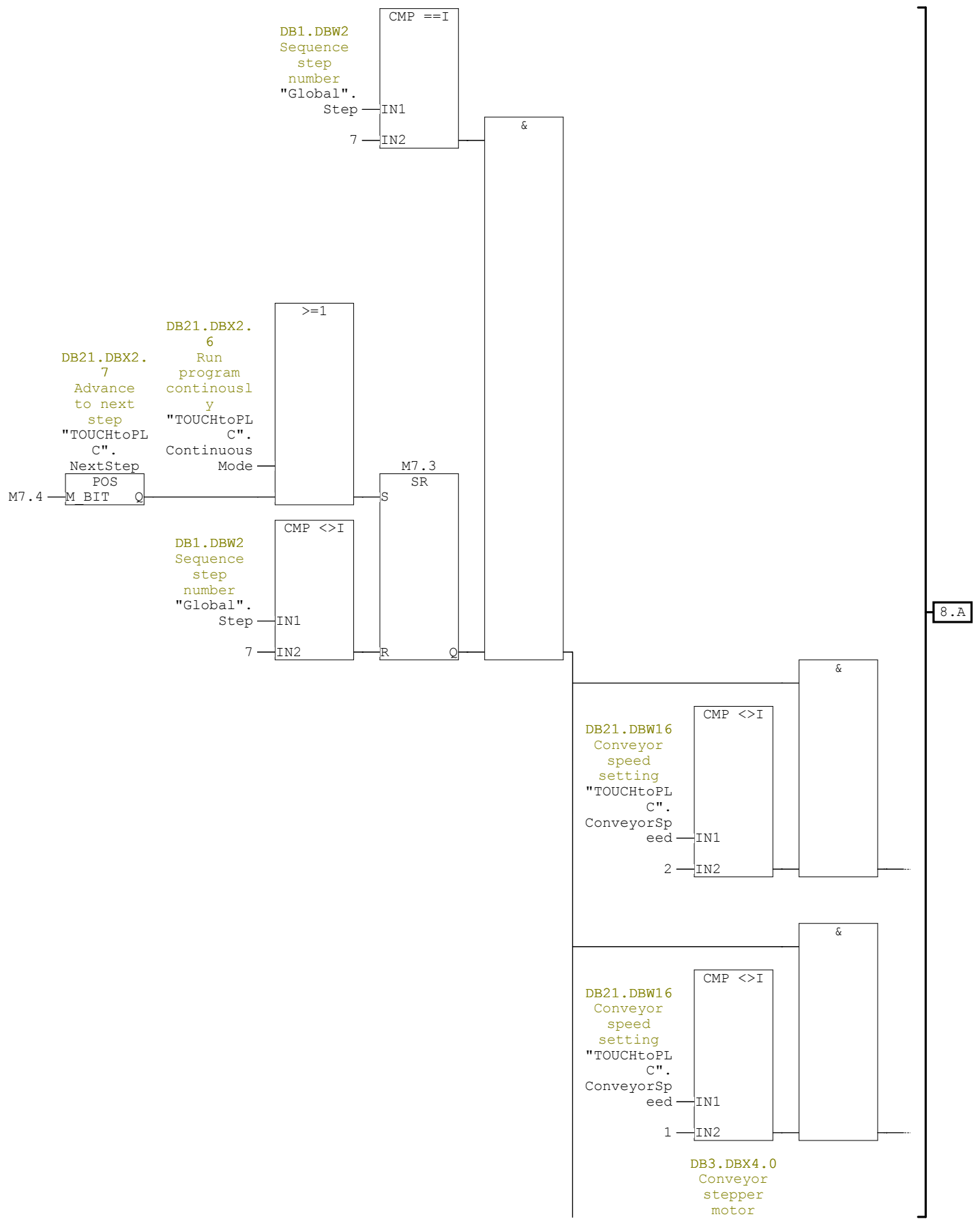


7.C

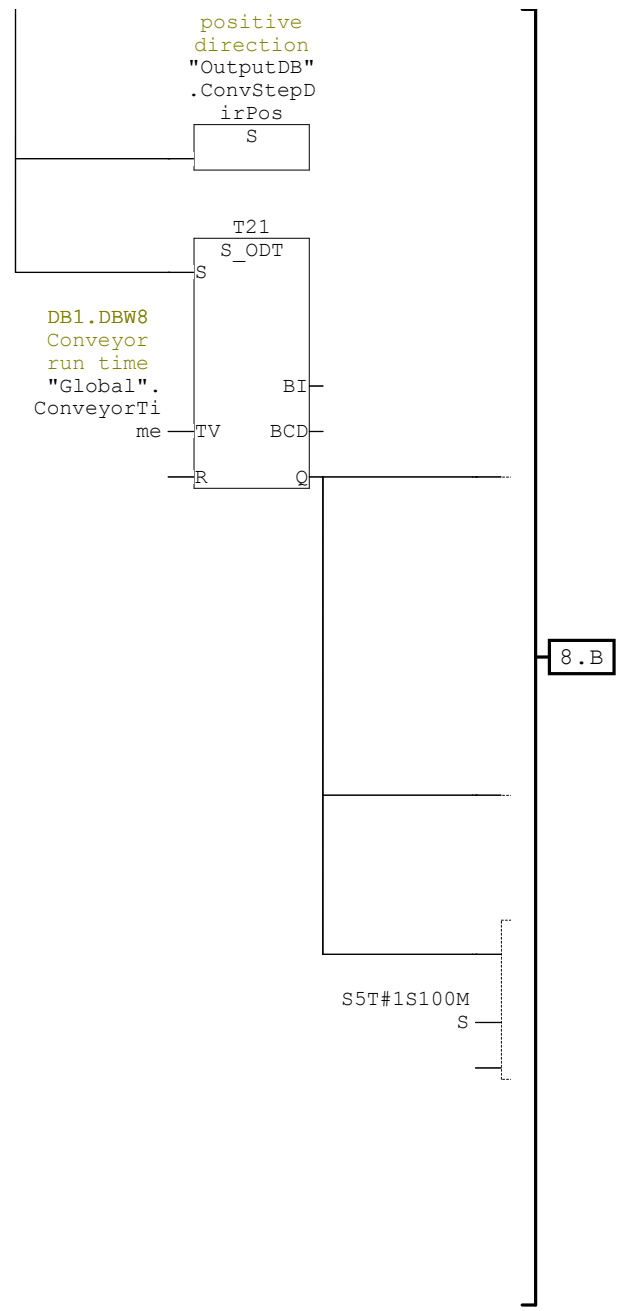
7.D

DB1.DBW2
Sequence
step
number
"Global".
Step

Network: 8 Run Conveyor



8.A



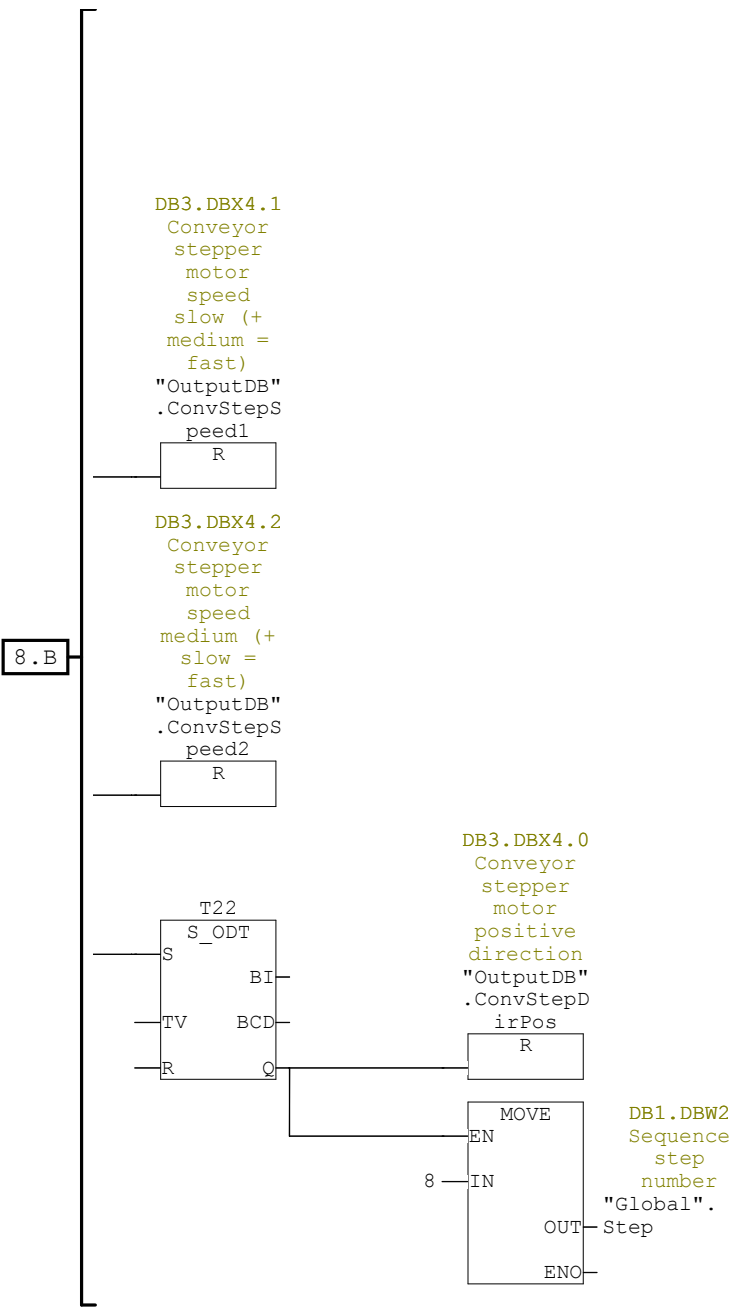
8.A

DB3.DBX4.1
Conveyor
stepper
motor
speed
slow (+
medium =
fast)
"OutputDB"
.ConvStepS
peed1

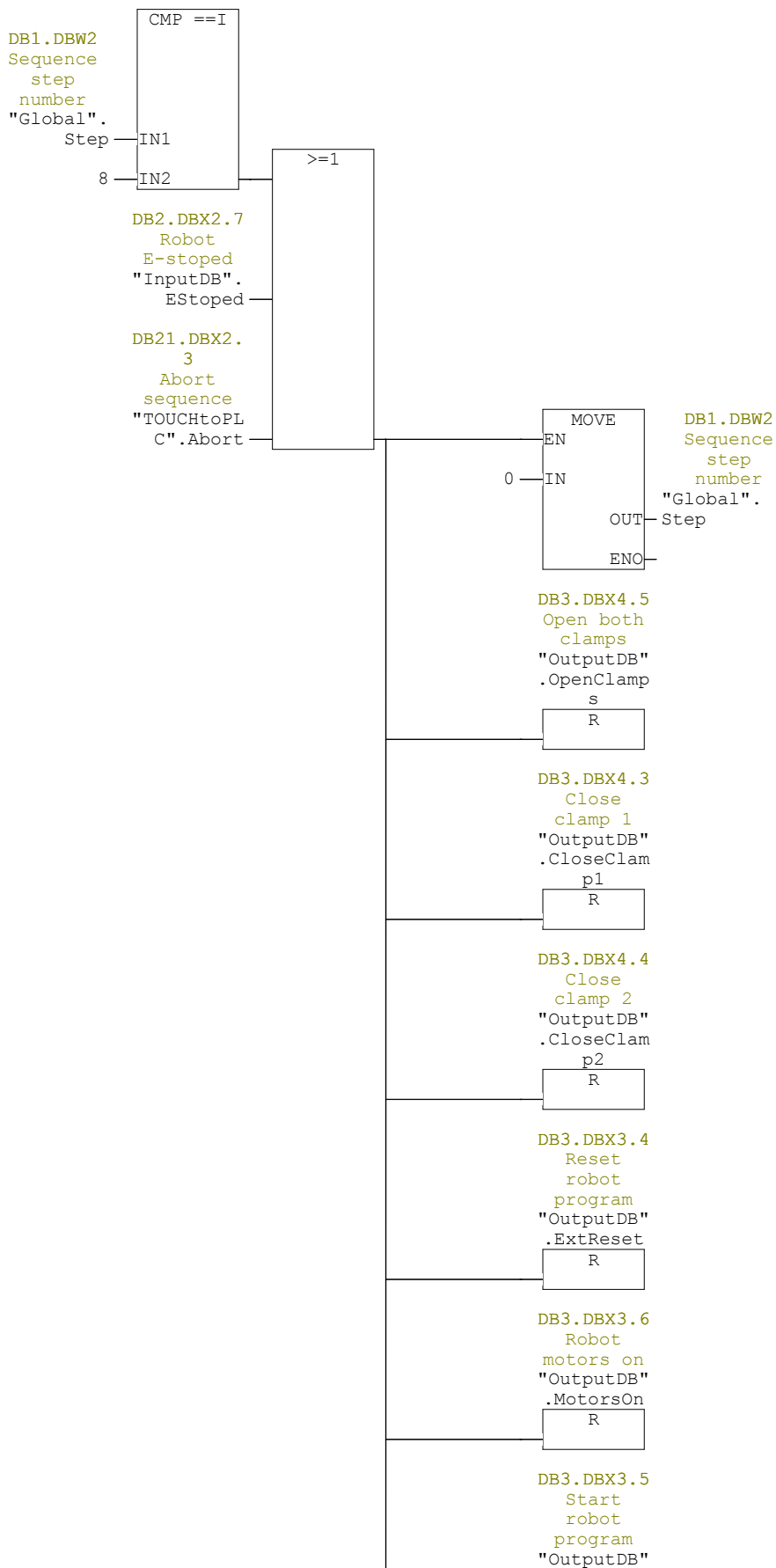
S

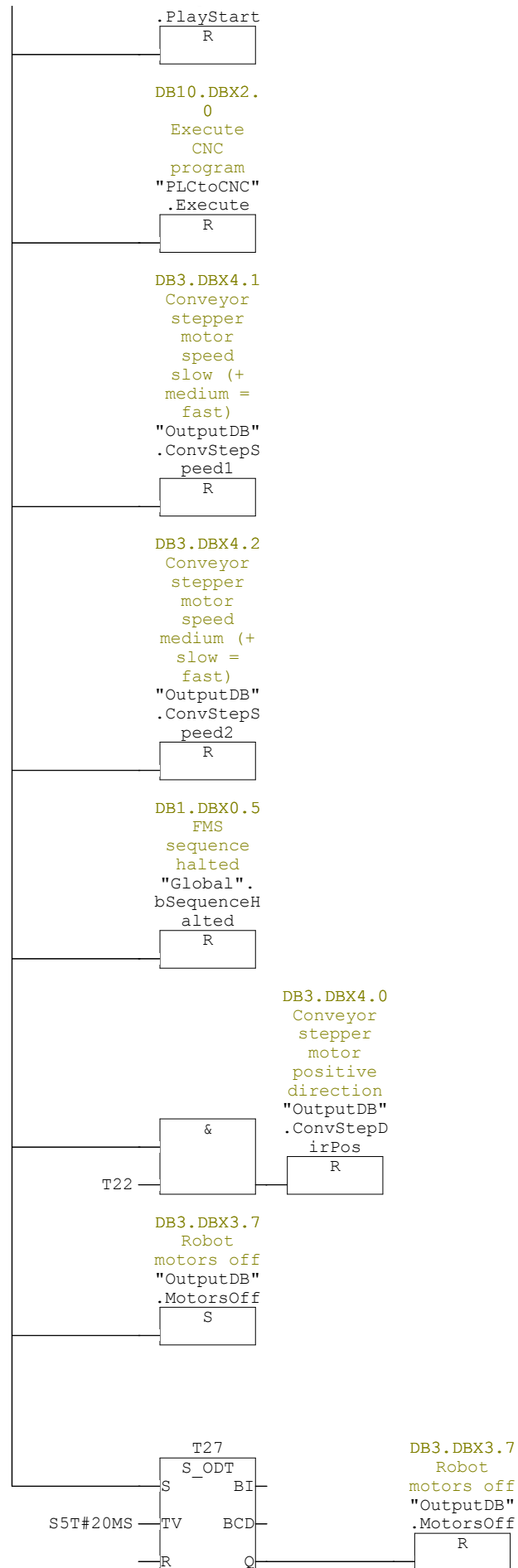
DB3.DBX4.2
Conveyor
stepper
motor
speed
medium (+
slow =
fast)
"OutputDB"
.ConvStepS
peed2

S

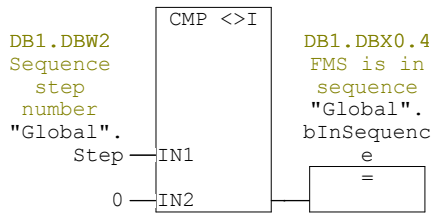


Network: 9 Reset sequence

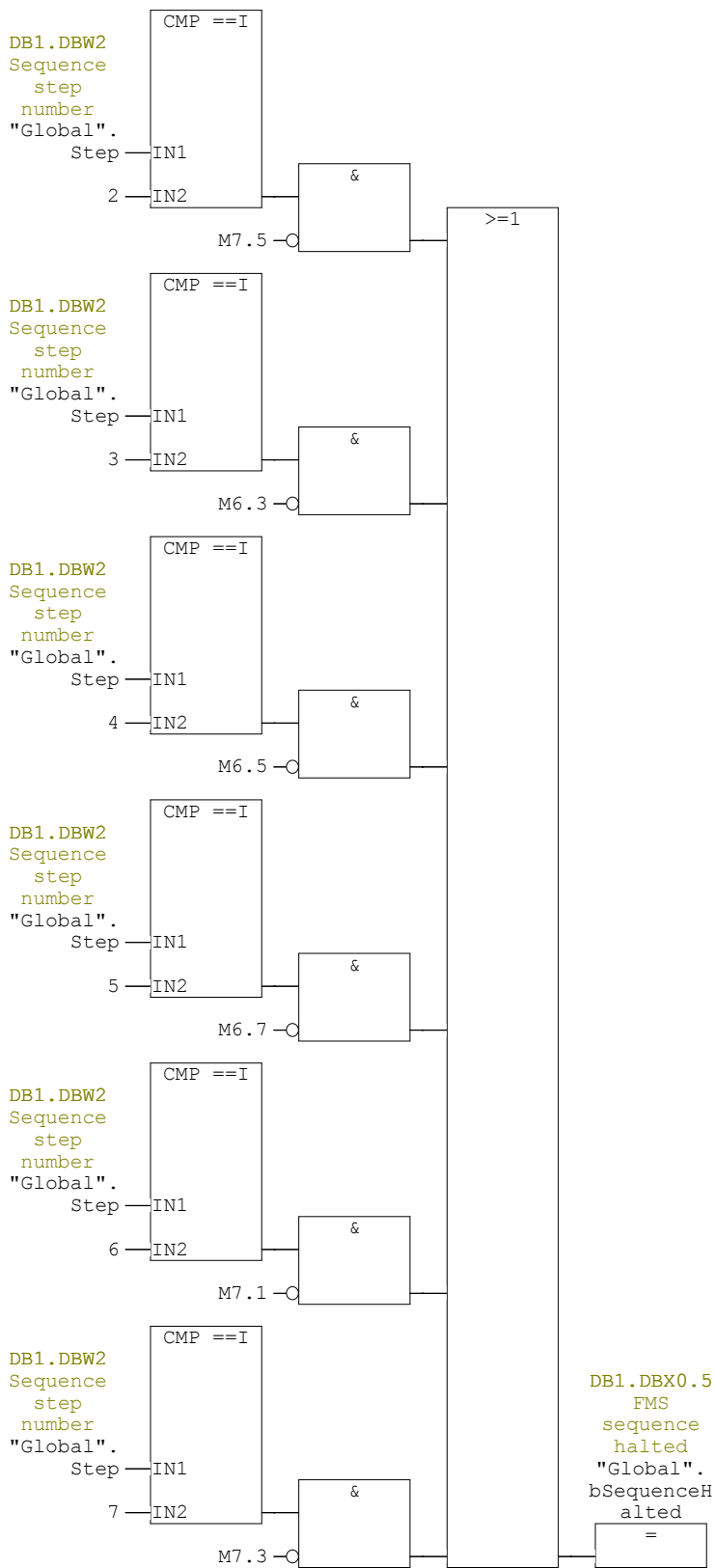




Network: 10 FMS is in sequence



Network: 11 FMS sequence halted



FC30 - <offline>

"ManualControl"

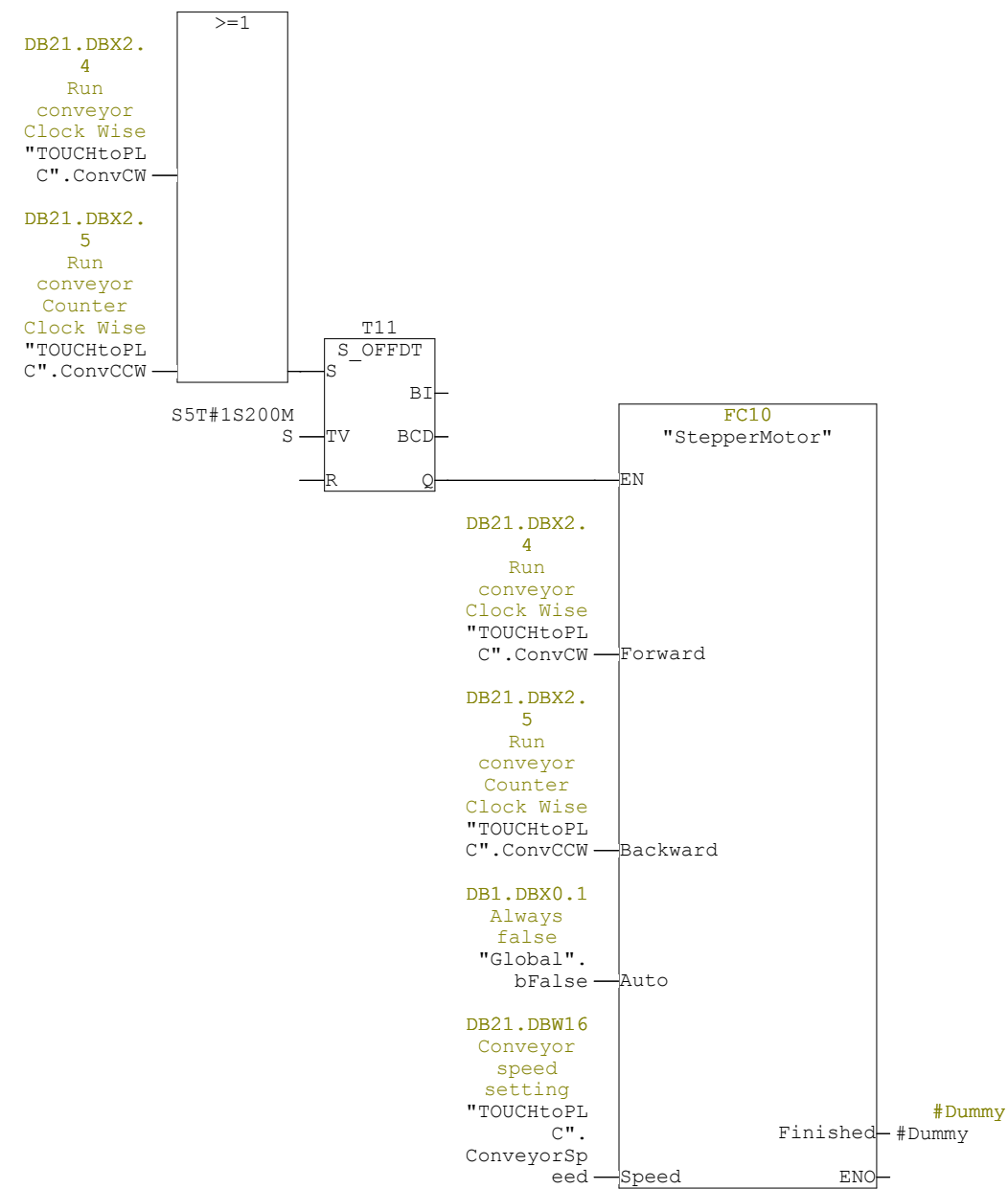
Name:
Author:
Time stamp Code:
Lengths (block/logic/data):

Family:
Version: 0.1
Block version: 2
05/13/2019 06:37:24 PM
04/11/2019 08:08:20 PM
00744 00618 00006

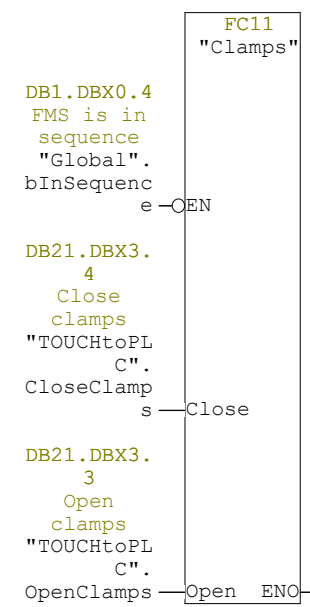
Name	Data Type	Address	Comment
IN		0.0	
OUT		0.0	
IN_OUT		0.0	
TEMP		0.0	
Dummy	Bool	0.0	
MoveInDone	Bool	0.1	
ClampsDone	Bool	0.2	
MoveOutDone	Bool	0.3	
RobotParked	Bool	0.4	
CNCDone	Bool	0.5	
RETURN		0.0	
RET_VAL		0.0	

Block: FC30

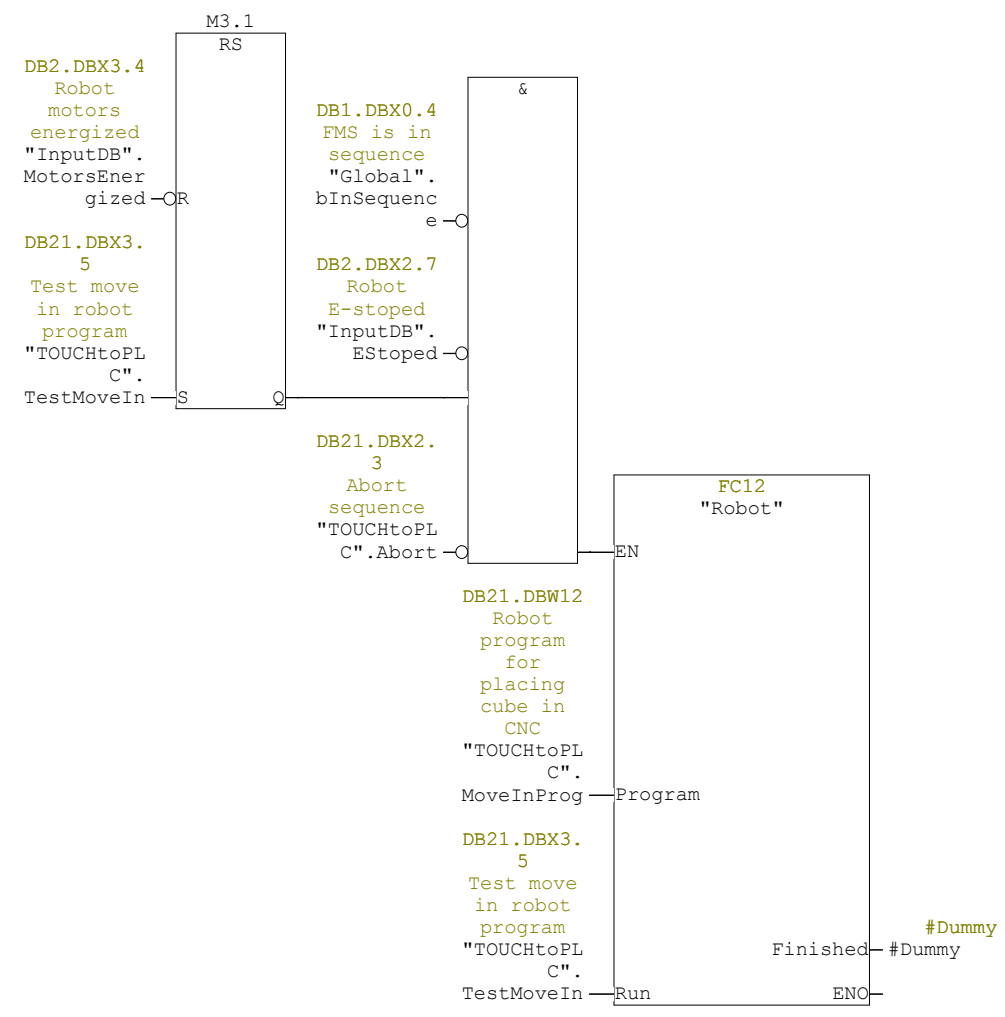
Network: 1 Conveyor stepper motor manual control



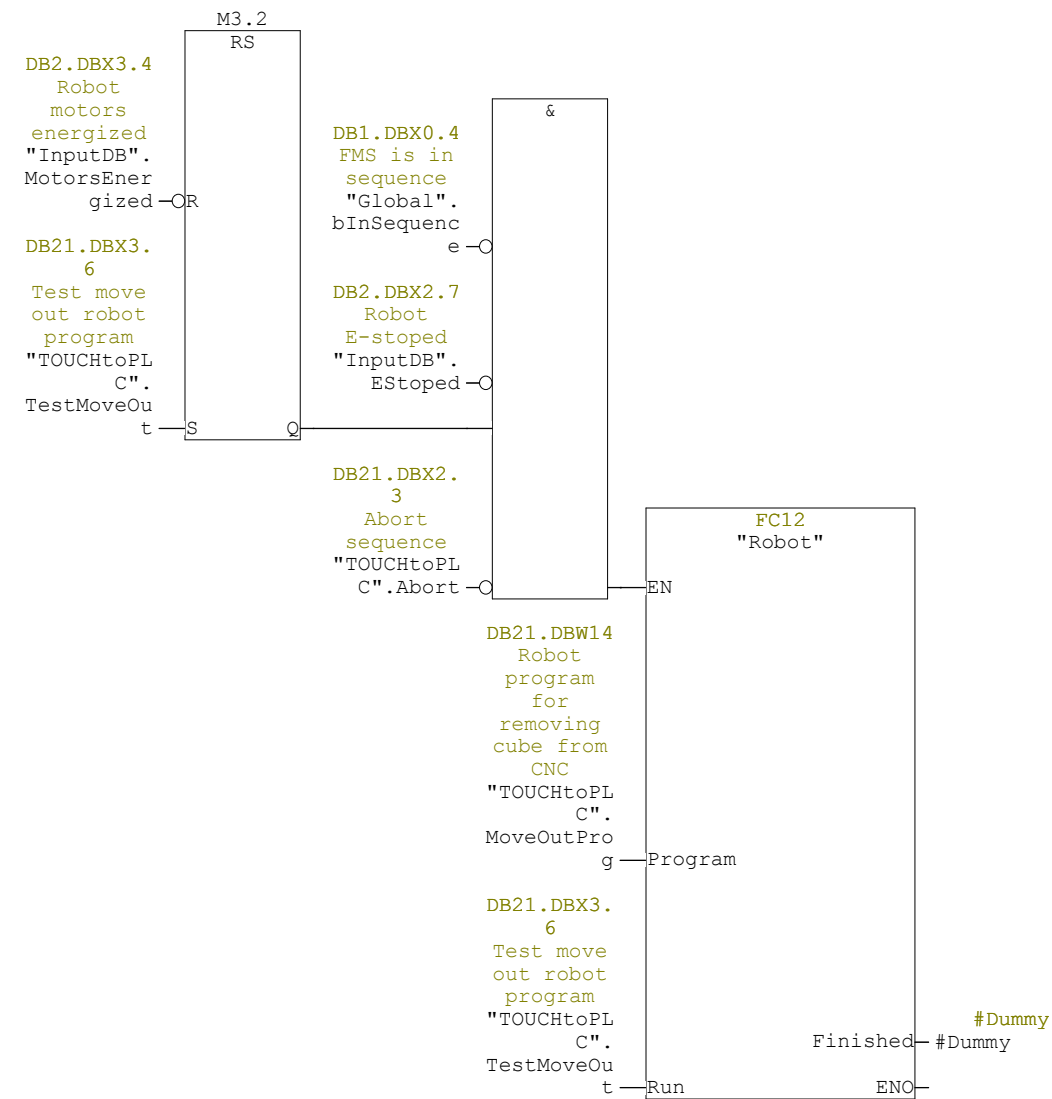
Network: 2 Manual clamps control



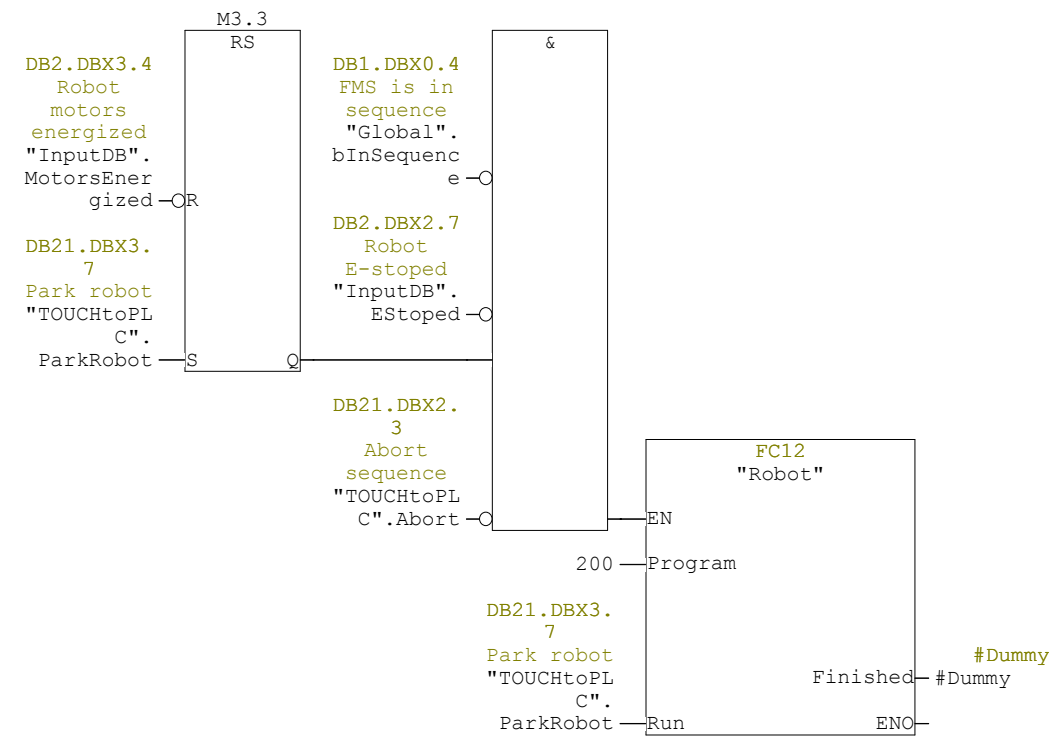
Network: 3 Test robot move in program



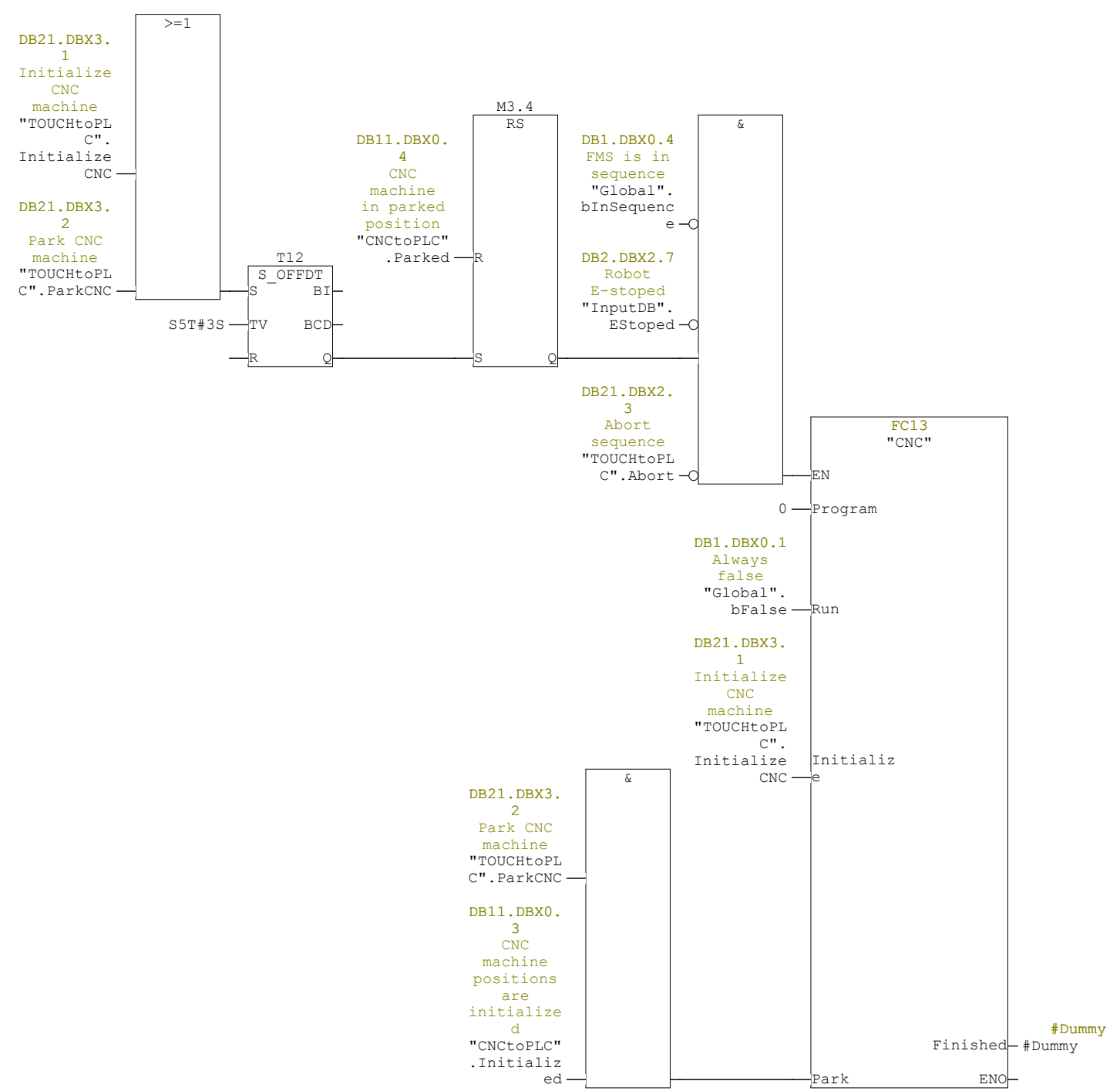
Network: 4 Test robot move out program



Network: 5 Park robot



Network: 6 Initialize or park CNC



FC100 - <offline>

"ProgramCycle"

Name:
Author:
Time stamp Code:
Interface:
Lengths (block/logic/data):

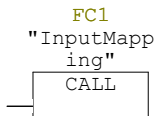
Family:
Version: 0.1
Block version: 2
 05/10/2019 03:04:20 PM
 04/11/2019 07:57:22 PM
 00132 00032 00000

Name	Data Type	Address	Comment
IN		0.0	
OUT		0.0	
IN_OUT		0.0	
TEMP		0.0	
RETURN		0.0	
RET_VAL		0.0	

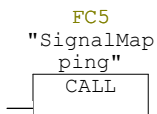
Block: FC100

Network: 1

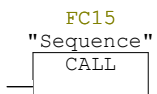
Read inputs - Always first network



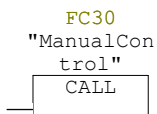
Network: 2



Network: 3



Network: 4



Network: 5

Write outputs - Always last network

FC2

"OutputMap
ping"

CALL

DB1 - <offline> - Declaration view

"Global"

Global data block DB 1

Name: **Family:**
Author: **Version:** 0.1
 Block version: 2
Time stamp Code: 05/10/2019 03:03:57 PM
 Interface: 05/10/2019 03:03:57 PM
Lengths (block/logic/data): 00120 00010 00000

Block: DB1

Address	Name	Type	Initial value	Comment
0.0		STRUCT		
+0.0	bTrue	BOOL	TRUE	Always true
+0.1	bFalse	BOOL	FALSE	Always false
+0.2	bInitialized	BOOL	FALSE	Variables initialized on startup
+0.3	bReady	BOOL	FALSE	FMS ready for sequence
+0.4	bInSequence	BOOL	FALSE	FMS is in sequence
+0.5	bSequenceHalted	BOOL	FALSE	FMS sequence halted
+2.0	Step	INT	0	Sequence step number
+4.0	Clamp1Delay	S5TIME	S5T#0MS	Clamp 1 delay
+6.0	Clamp2Delay	S5TIME	S5T#0MS	Clamp 2 delay
+8.0	ConveyorTime	S5TIME	S5T#0MS	Conveyor run time
=10.0		END_STRUCT		

DB2 - <offline> - Declaration view

"InputDB"

Global data block DB 2

Name:
Author:
Time stamp Code:
Lengths (block/logic/data):

Family:
Version: 0.1
Block version: 2
 05/06/2019 06:36:52 PM
Interface: 04/11/2019 07:15:13 PM
 00224 00008 00000

Block: DB2

Address	Name	Type	Initial value	Comment
0.0		STRUCT		
+0.0	In1	BOOL	FALSE	Input 1
+0.1	In2	BOOL	FALSE	Input 2
+0.2	In3	BOOL	FALSE	Input 3
+0.3	In4	BOOL	FALSE	Input 4
+0.4	In5	BOOL	FALSE	Input 5
+0.5	In6	BOOL	FALSE	Input 6
+0.6	In7	BOOL	FALSE	Input 7
+0.7	In8	BOOL	FALSE	Input 8
+1.0	In9	BOOL	FALSE	Input 9
+1.1	In10	BOOL	FALSE	Input 10
+1.2	In11	BOOL	FALSE	Input 11
+1.3	In12	BOOL	FALSE	Input 12
+1.4	In13	BOOL	FALSE	Input 13
+1.5	In14	BOOL	FALSE	Input 14
+1.6	In15	BOOL	FALSE	Input 15
+1.7	In16	BOOL	FALSE	Input 16
+2.0	In17	BOOL	FALSE	Input 17
+2.1	In18	BOOL	FALSE	Input 18
+2.2	RobotReady	BOOL	FALSE	Robot is ready
+2.3	ProgramEnded	BOOL	FALSE	Robot program ended
+2.4	Error	BOOL	FALSE	Robot error
+2.5	PlaybackMode	BOOL	FALSE	Robot in playback mode
+2.6	Alarm	BOOL	FALSE	Robot alarm
+2.7	EStoped	BOOL	FALSE	Robot E-stoped
+3.0	TeachMode	BOOL	FALSE	Robot in teach mode
+3.1	Running	BOOL	FALSE	Robot is running
+3.2	ExtPrgEn	BOOL	FALSE	External program selection enabled
+3.3	ExtStartEn	BOOL	FALSE	External start enabled
+3.4	MotorsEnergized	BOOL	FALSE	Robot motors energized
+3.5	StateOutput	BOOL	FALSE	Robot state output
+3.6	HomePos	BOOL	FALSE	Robot home position
+3.7	Info	BOOL	FALSE	Robot info
+4.0	In33	BOOL	FALSE	Input 33
+4.1	In34	BOOL	FALSE	Input 34
+4.2	In35	BOOL	FALSE	Input 35
+4.3	In36	BOOL	FALSE	Input 36
+4.4	In37	BOOL	FALSE	Input 37
+4.5	In38	BOOL	FALSE	Input 38
+4.6	In39	BOOL	FALSE	Input 39
+4.7	In40	BOOL	FALSE	Input 40
+5.0	In41	BOOL	FALSE	Input 41
+5.1	In42	BOOL	FALSE	Input 42
+5.2	In43	BOOL	FALSE	Input 43
+5.3	In44	BOOL	FALSE	Input 44
+5.4	In45	BOOL	FALSE	Input 45
+5.5	In46	BOOL	FALSE	Input 46
+5.6	In47	BOOL	FALSE	Input 47
+5.7	In48	BOOL	FALSE	Input 48
+6.0	In49	BOOL	FALSE	Input 49
+6.1	In50	BOOL	FALSE	Input 50
+6.2	In51	BOOL	FALSE	Input 51
+6.3	In52	BOOL	FALSE	Input 52
+6.4	In53	BOOL	FALSE	Input 53
+6.5	In54	BOOL	FALSE	Input 54
+6.6	In55	BOOL	FALSE	Input 55

Address	Name	Type	Initial value	Comment
+6.7	In56	BOOL	FALSE	Input 56
+7.0	In57	BOOL	FALSE	Input 57
+7.1	In58	BOOL	FALSE	Input 58
+7.2	In59	BOOL	FALSE	Input 59
+7.3	In60	BOOL	FALSE	Input 60
+7.4	In61	BOOL	FALSE	Input 61
+7.5	In62	BOOL	FALSE	Input 62
+7.6	In63	BOOL	FALSE	Input 63
+7.7	In64	BOOL	FALSE	Input 64
=8.0		END_STRUCT		

DB3 - <offline> - Declaration view

"OutputDB"

Global data block DB 3

Name: **Family:**
Author: **Version:** 0.1
 Block version: 2
Time stamp Code: 05/08/2019 08:05:53 PM
 Interface: 05/06/2019 02:23:54 PM
Lengths (block/logic/data): 00210 00008 00000

Block: DB3

Address	Name	Type	Initial value	Comment
0.0		STRUCT		
+0.0	Out1	BOOL	FALSE	Output 1
+0.1	Out2	BOOL	FALSE	Output 2
+0.2	Out3	BOOL	FALSE	Output 3
+0.3	Out4	BOOL	FALSE	Output 4
+0.4	Out5	BOOL	FALSE	Output 5
+0.5	Out6	BOOL	FALSE	Output 6
+0.6	Out7	BOOL	FALSE	Output 7
+0.7	Out8	BOOL	FALSE	Output 8
+1.0	Out9	BOOL	FALSE	Output 9
+1.1	Out10	BOOL	FALSE	Output 10
+1.2	Out11	BOOL	FALSE	Output 11
+1.3	Out12	BOOL	FALSE	Output 12
+1.4	Out13	BOOL	FALSE	Output 13
+1.5	Out14	BOOL	FALSE	Output 14
+1.6	Out15	BOOL	FALSE	Output 15
+1.7	Out16	BOOL	FALSE	Output 16
+2.0	ProgramSelection	BYTE	B#16#0	Select robot program
+3.0	ProgramStrobe	BOOL	FALSE	Robot program storbe
+3.1	PlayStop	BOOL	FALSE	Stop robot program (inverted)
+3.2	Out27	BOOL	FALSE	Output 27
+3.3	Out28	BOOL	FALSE	Output 28
+3.4	ExtReset	BOOL	FALSE	Reset robot program
+3.5	PlayStart	BOOL	FALSE	Start robot program
+3.6	MotorsOn	BOOL	FALSE	Robot motors on
+3.7	MotorsOff	BOOL	FALSE	Robot motors off
+4.0	ConvStepDirPos	BOOL	FALSE	Conveyor stepper motor positive direction
+4.1	ConvStepSpeed1	BOOL	FALSE	Conveyor stepper motor speed slow (+ medium = fast)
+4.2	ConvStepSpeed2	BOOL	FALSE	Conveyor stepper motor speed medium (+ slow = fast)
+4.3	CloseClamp1	BOOL	FALSE	Close clamp 1
+4.4	CloseClamp2	BOOL	FALSE	Close clamp 2
+4.5	OpenClamps	BOOL	FALSE	Open both clamps
+4.6	Out39	BOOL	FALSE	Output 39
+4.7	Out40	BOOL	FALSE	Output 40
+5.0	Out41	BOOL	FALSE	Output 41
+5.1	Out42	BOOL	FALSE	Output 42
+5.2	Out43	BOOL	FALSE	Output 43
+5.3	Out44	BOOL	FALSE	Output 44
+5.4	Out45	BOOL	FALSE	Output 45
+5.5	Out46	BOOL	FALSE	Output 46
+5.6	Out47	BOOL	FALSE	Output 47
+5.7	Out48	BOOL	FALSE	Output 48
+6.0	Out49	BOOL	FALSE	Output 49
+6.1	Out50	BOOL	FALSE	Output 50
+6.2	Out51	BOOL	FALSE	Output 51
+6.3	Out52	BOOL	FALSE	Output 52
+6.4	Out53	BOOL	FALSE	Output 53
+6.5	Out54	BOOL	FALSE	Output 54
+6.6	Out55	BOOL	FALSE	Output 55
+6.7	Out56	BOOL	FALSE	Output 56
+7.0	Out57	BOOL	FALSE	Output 57
+7.1	Out58	BOOL	FALSE	Output 58
+7.2	Out59	BOOL	FALSE	Output 59
+7.3	Out60	BOOL	FALSE	Output 60
+7.4	Out61	BOOL	FALSE	Output 61
+7.5	Out62	BOOL	FALSE	Output 62
+7.6	Out63	BOOL	FALSE	Output 63
+7.7	Out64	BOOL	FALSE	Output 64
=8.0		END_STRUCT		

DB10 - <offline> - Declaration view

"PLCtoCNC"

Global data block DB 10

Name:
Author:
Time stamp Code:
Lengths (block/logic/data):

Family:
Version: 0.1
Block version: 2
 05/08/2019 06:57:29 PM
 Interface: 05/08/2019 06:57:29 PM
 00106 00004 00000

Block: DB10

Address	Name	Type	Initial value	Comment
0.0		STRUCT		
+0.0	Program	INT	1	CNC program selection
+2.0	Execute	BOOL	FALSE	Execute CNC program
+2.1	Pause	BOOL	FALSE	Pause CNC program
+2.2	Abort	BOOL	FALSE	Abort CNC program
+2.3	Initialize	BOOL	FALSE	Initialize CNC machine position
+2.4	Park	BOOL	FALSE	Park CNC machine
=4.0		END_STRUCT		

DB11 - <offline> - Declaration view

"CNCToPLC"

Global data block DB 11

Name:
Author:
Time stamp Code:
Lengths (block/logic/data):

Family:
Version: 0.1
Block version: 2
 04/27/2019 03:58:49 PM
Interface: 04/27/2019 03:58:49 PM
 00100 00002 00000

Block: DB11

Address	Name	Type	Initial value	Comment
0.0		STRUCT		
+0.0	ResetAlarm	BOOL	FALSE	Reset alarm active on CNC
+0.1	Running	BOOL	FALSE	CNC program is running
+0.2	Paused	BOOL	FALSE	CNC program is paused
+0.3	Initialized	BOOL	FALSE	CNC machine positions are initialized
+0.4	Parked	BOOL	FALSE	CNC machine in parked position
=2.0		END_STRUCT		

DB20 - <offline> - Declaration view

"PLCtoTOUCH"

Global data block DB 20

Name:
Author:
Time stamp Code:
Lengths (block/logic/data):

Family:
Version: 0.1
Block version: 2
 05/13/2019 12:59:10 PM
 Interface: 05/13/2019 12:59:10 PM
 00122 00006 00000

Block: DB20

Address	Name	Type	Initial value	Comment
0.0		STRUCT		
+0.0	CNCInitialized	BOOL	FALSE	CNC machine is initialized
+0.1	CNCParked	BOOL	FALSE	CNC machine is parked
+0.2	FMSReady	BOOL	FALSE	FMS is ready for sequence
+0.3	RobotPlaybackMode	BOOL	FALSE	Robot is in playback mode
+0.4	RobotExtPrgEn	BOOL	FALSE	Robot external program enabled
+0.5	RobotExtStartEn	BOOL	FALSE	Robot external start enabled
+0.6	RobotReady	BOOL	FALSE	Robot is ready
+0.7	RobotNotRunning	BOOL	FALSE	Robot is not running
+1.0	RobotNotEStop	BOOL	FALSE	Robot is not E-stopped
+1.1	CNCProgramNotRunning	BOOL	FALSE	CNC program not running
+1.2	SequenceHalted	BOOL	FALSE	Sequence is halted
+1.3	InSequence	BOOL	FALSE	FMS is in sequence
+2.0	SequenceStep	INT	0	Sequence step number
+4.0	SequenceStatus	INT	0	Sequence status for step indicator
=6.0		END_STRUCT		

DB21 - <offline> - Declaration view

"TOUCHtoPLC"

Global data block DB 21

Name: **Family:**
Author: **Version:** 0.1
 Block version: 2
Time stamp Code: 05/08/2019 06:30:57 PM
 Interface: 05/08/2019 06:30:57 PM
Lengths (block/logic/data): 00154 00018 00000

Block: DB21

Address	Name	Type	Initial value	Comment
0.0		STRUCT		
+0.0	CNCProgram	INT	0	CNC program selection
+2.0	Execute	BOOL	FALSE	Execute sequence
+2.1	Pause	BOOL	FALSE	Pause sequence
+2.2	Resume	BOOL	FALSE	Resume sequence
+2.3	Abort	BOOL	FALSE	Abort sequence
+2.4	ConvCW	BOOL	FALSE	Run conveyor Clock Wise
+2.5	ConvCCW	BOOL	FALSE	Run conveyor Counter Clock Wise
+2.6	ContinuousMode	BOOL	FALSE	Run program continously
+2.7	NextStep	BOOL	FALSE	Advance to next step
+3.0	ResetSettings	BOOL	FALSE	Reset settings to default value
+3.1	InitializeCNC	BOOL	FALSE	Initialize CNC machine
+3.2	ParkCNC	BOOL	FALSE	Park CNC machine
+3.3	OpenClamps	BOOL	FALSE	Open clamps
+3.4	CloseClamps	BOOL	FALSE	Close clamps
+3.5	TestMoveIn	BOOL	FALSE	Test move in robot program
+3.6	TestMoveOut	BOOL	FALSE	Test move out robot program
+3.7	ParkRobot	BOOL	FALSE	Park robot
+4.0	Clamp1Timer	TIME	T#0MS	Time delay clamp 1
+8.0	Clamp2Timer	TIME	T#1S	Time delay clamp 2
+12.0	MoveInProg	INT	0	Robot program for placing cube in CNC
+14.0	MoveOutProg	INT	0	Robot program for removing cube from CNC
+16.0	ConveyorSpeed	INT	0	Conveyor speed setting
=18.0		END_STRUCT		