Toni Berg

Automation Program Design of Device for Bending Coaxial Cables

Metropolia University of Applied Sciences

Bachelor of Engineering

Electrical and Automation Engineering

Bachelor's Thesis

26.5.2021

Abstract

Author:	Toni Berg
Title:	Automation Program Design of Device for Bending Coaxial Cables
Number of Pages:	35 pages
Date:	26 May 2021
Degree:	Bachelor of Engineering
Degree Programme:	Electrical and Automation Engineering
Professional Major:	Automation Engineering
Supervisors:	Mikko Peurala, Senior Technician Timo Tuominen, Senior Lecturer

---

In this thesis the working principle of a coaxial cable bending device created for Bluefors Oy and the process of designing and programming a control program for that device is described.
The purpose of this thesis work was to describe the designing and creation of the control program for the coaxial bending device while explaining the core functionality of the control program. The goal for this thesis work was to create a working control program for the device with which the process of cutting and bending of coaxial cable can be automated, thus making production more cost effective and less prone to human error.

The device is controlled through a programmable logic controller (PLC). Coaxial cable is moved through the device with two stepper motors attached to rollers. Another stepper motor with a bending attachment then bends the cable to the required dimensions. The coaxial cable is then cut to length using a cutter attached to a linear actuator.

The work was carried out by first designing and creating a sequential program based on the device design that should in theory accomplish the required functions of bending and cutting coaxial cables. When the prototype of the device was built the program was refined to be able to create bent cables according to requirements.

Testing was first done using a copper wire with similar dimensions to the actual coaxial cable used in production. When the bends were close to the required dimension the testing and adjusting of the program was finalized using the actual coaxial cable used in production. The goals given by Bluefors Oy and the mechanical constraints and requirements of the device were taken into consideration during programming.

The results of the thesis work are a working control program for the coaxial cable bending device that makes the automation of the bending process of coaxial cable possible.

Keywords: PLC, Coaxial Cable, design, device

Tiivistelmä

Tekijä:	Toni Berg
Otsikko:	Automation Program Design of Device for Bending Coaxial Cables
Sivumäärä:	35 sivua
Aika:	26.5.2021
Tutkinto:	Insinööri (AMK)
Tutkinto-ohjelma:	Sähkö- ja automaatiotekniikka
Ammatillinen pääaine:	Automaatiotekniikka
Ohjaajat:	Senior Technician Mikko Peurala Lehtori Timo Tuominen

Tässä opinnäytetyössä kuvataan Bluefors Oy:lle suunnitellun ja valmistetun koaksiaalikaapelin taivutukseen kehitetyn laitteen toimintaperiaate, automaatio-ohjelmiston suunnittelu ja ohjelmointi sekä käydään läpi ohjelmiston rakenne.

Opinnäytetyön tarkoitus oli kuvata koaksiaalikaapelia taivuttavan laitteen automaatio-ohjelmiston suunnittelu ja ohjelmointi sekä käydä läpi tuotetun ohjelman kannalta keskeiset funktiot. Opinnäytetyön tavoitteena oli luoda laitteelle ohjelmisto, jonka avulla koaksiaalikaapelin leikkaamis- ja taivutusprosessi pystyttiin automatisoimaan kaapeleiden tuotannon kustannustehokkuuden lisäämiseksi ja inhimillisten virheiden minimoimiseksi.

Laitetta ohjataan ohjelmoitavalla logiikalla, ja sen toimintaperiaate on seuraava: Kaksi askelmoottoria pyörittää rullia, jotka liikuttavat koaksiaalikaapelia laitteen läpi. Kolmas askelmoottori taivuttaa kaapelin haluttuun malliin erillisen taivutinlaitteen avulla. Lopuksi lineaariseen toimilaitteeseen kiinnitetty leikkuri leikkaa koaksiaalikaapelin oikeaan mittaan.

Työn alussa suunniteltiin ja luotiin alustava sekvenssiohjelma koneen suunnitelman mukaan. Kun laitteen prototyyppi saatiin rakennettua, ohjelmiston kehitystä jatkettiin niin, että se mahdollisti kaapeleiden taivutuksen vaatimuksien mukaisesti. Testaukset tehtiin aluksi kuparilangalla, jonka mitat olivat lähellä tuotannossa käytettävän koaksiaalikaapelin mittoja. Kun taitokset saatiin lähelle tavoitetta, viimeisteltiin ohjelma testaamalla laitteen toimintaa tuotannossa käytettävällä koaksiaalikaapelilla. Ohjelmiston toteutuksessa otettiin huomioon laitteen asettamat mekaaniset vaatimukset ja rajoitukset sekä Bluefors Oy:n laitteelle asettamat tavoitteet.

Opinnäytetyön tuloksena syntyi koaksiaalikaapelia taivuttavan laitteen ohjaukseen tarkoitettu ohjelmisto, joka mahdollisti taivutusprosessin automatisoinnin.

Avainsanat: PLC, koaksiaalikaapeli, suunnittelu, laite

Contents

List of Abbreviations

CPU	Central processing unit
GSD	General station description
HMI	Human machine interface
PLC	Programmable logic controller

1 Introduction

Bluefors Oy builds Cryogen-free Dilution Refrigerator Measurement Systems used in research. In the Dilution Refrigerator Measurement System there can be multiple coaxial cable lines that go through portions of the system. As of now all the coaxial cables are created by hand and the process of manufacturing includes cutting the cable stock to length, stripping the ends of the cable, soldering connectors to the ends of the cable, and finally bending the cable to a precise profile. Bending the cables by hand to this precise profile is slow and prone to human errors as the number of cables increase.

The purpose of this thesis work was, to describe the designing and creation of the control program for the coaxial bending device, while explaining the core functionality of the control program. The goal of this thesis work was, to create a control program for coaxial cable bending and cutting device, that will automate the bending and cutting of the coaxial cable increasing speed, minimizing loss of cable, and removing human error in the bending of the cable. This device was designed and created inhouse at Bluefors Oy according to the requirements from the company.

2 Coaxial Cable

Coaxial cable has an inner and outer conductor with a dielectric layer in between the two conductors. Because these two conductors are arranged this way, they follow the same geometric axis and are therefore in a coaxial arrangement. Because of the coaxial cables design, when the outer conductor is grounded and a voltage is applied to the inner conductor, the electromagnetic field is restricted to the dielectric layer, thus protecting the signal from outside interference. For this reason, coaxial cables are a great choice for transmitting signals that are susceptible to interference. There are multiple types of coaxial cables from which the semi rigid coaxial cable is important for the thesis work. [1.]

2.1 Semi-rigid Coaxial Cable

Semi rigid coaxial cable is made with a solid metal outer conductor that is essentially a pipe over the dielectric insulator between the conductors. This outer conductor is bendable but still allows for some flex in the cable. Because of the solid outer conductor, the semi-rigid coaxial cable provides excellent performance and can thoroughly block outside interference. [1.] The semi-rigid coaxial cable meant to be bent with the device is a cupronickel alloy cable. This cable is designed to be used in cryogenic conditions.

3 Device Function and Requirements

The device was meant to take straight coaxial cable and create bent coaxial cables of the correct length automatically. This was achieved by feeding the material through the device bending it and cutting it to length. Material feeding was done by using stepper motors attached to rollers that move the cable through the device. Bending was done by a stepper motor with a bending attachment that bends the cable with multiple small bends to achieve the wanted profile. Bending direction can be changed by pulling the pin down with a solenoid and moving the pin to the other side of the cable. Cutting was done by using linear actuators attached to cutters. Two cutters were used to avoid leaving excess material inside the device by having the rear cutter at the feeding end of the device and the front cutter close to the bending position. The layout of the device is presented in figure 1. The requirements for produced cables were for the bends to be uniform, and for the cable lengths to be within 1mm of the design length.

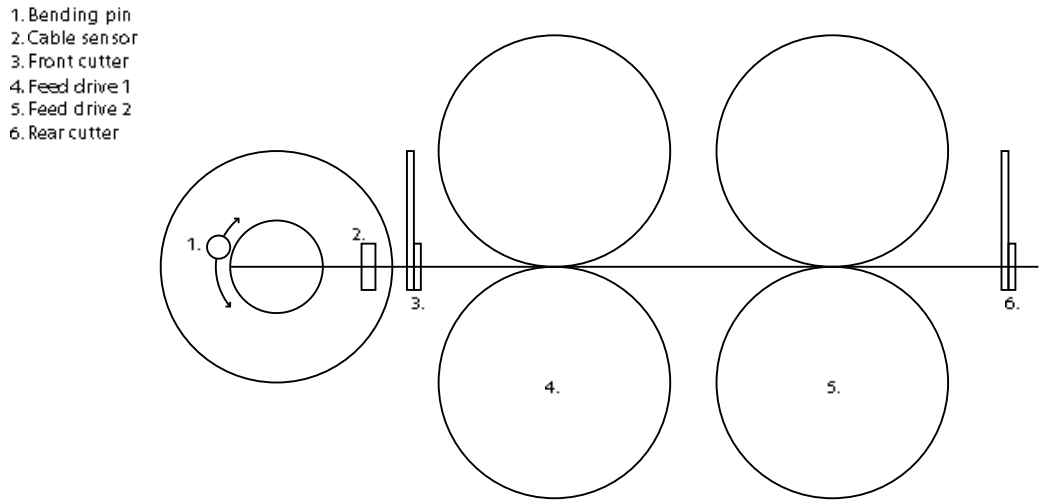

Figure 1. Layout of the device.

4 Ladder Programming Language

In ladder diagrams the power supply for the given circuit is always drawn as two vertical lines and the rest of the circuit in horizontal lines. When the diagram is drawn this way the ending result looks like a ladder. In a normal circuit diagram, the creator often tries to show the relative physical location of components and the wiring of it, but a ladder diagram only shows the electrical connections and how the control is exercised. In figure 2 the difference between a circuit diagram and a ladder diagram is shown. [2.]

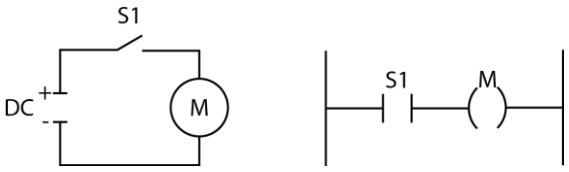

Figure 2. Difference between a ladder diagram and a circuit diagram.

Ladder programming is done in the form of ladder diagrams. As stated before, the vertical lines of the diagram are power lines, and the circuit is connected through

horizontal lines. The diagram is read from left to right and top to bottom with each horizontal line defining bottom and then start again from the top. Each horizontal line must start with an input and end in an output. Inputs are control signals to the programmable logic controller (PLC) for example a switch. Outputs are devices connected to the PLC for example a motor. All devices are shown in their normal conditions on the diagram. Normally open contacts will be open and normally closed contacts will be closed. Devices are labeled in the diagram and the same device can be used on multiple lines of the ladder diagram. All the I/O are addressed, and these addresses are saved in the memory of the PLC. [2.]

5 Hardware Choices

The hardware for the device was chosen during the mechanical design of the device and in this chapter, the reasons for these choices are clarified. The size of the material and the device were the biggest constraints in hardware choices.

5.1 PLC

Programmable logic controller is a device that can control actuators and devices in a predefined way. This predefined logic is programmed into the PLC and works with the information received from sensors, buttons, and human machine interface (HMI) to control the function of a machine.

There are multiple options for programmable logic controllers from different manufacturers like for example Beckhoff, Festo and Siemens. From these manufacturers Siemens was chosen as the company already had a device built with a Siemens S-1200 PLC being used in production. From Siemens's lineup of PLC, a plc with integrated safety function and Profinet compatibility was chosen for ease of expansion and for the control of chosen Stepper drivers.

Siemens offers a wide range of different PLCs that are split into two product lines SIMATIC S7-1200 and SIMATIC S7-1500.From these the S7-1200 series is aimed towards more compact solutions and the built in I/O can be extended with additional modules. From the S7-1200 series a fail-safe central processing unit (CPU) was

chosen, and it was a model CPU 1214 FC DC/DC/DC. This CPU supports up to 8 I/O modules for possible future expansion of the device.

The CPU also required a power supply unit, and a proper unit was chosen from Siemens which was power module PM1207. This power module works with 230VAC and supplies a stabilized 24VDC and 2.5A source for the PLC and its modules. This power supply was calculated to be enough to run all the sensors and actuators in use for the device. The stepper drives in use require more power and it was chosen to have a separate power supply for them.

For the safety functions of the device a safety input module was required, and the module chosen was SM 1226. This module is made to be used for safety related functions in a separate part of the program.

5.2 HMI

Human machine interface (HMI) is an interface that connects a person to a machine. Through this interface it is then possible to control or monitor the functionality of the machine.

For operator control of the device a HMI-device was necessary and such a device was chosen from Siemens lineup. KTP700 Basic panel was chosen with Profinet support for communication with the CPU.

5.3 Stepper Motor and Driver

A stepper motor unlike a normal DC motor is moved with electrical pulses to its multiple coils. When these coils are pulsed in the correct order the motor will move by steps for each pulse. The direction of the rotation can be controlled by the switching the polarity of these coils. Each step of the motor is a certain amount of rotation for example 1.8 degrees of rotation per step. These full steps of 1.8 degrees can further be broken down to smaller increments with the use of a stepper driver that supports micro stepping. Stepper driver is required to properly use a stepper motor and control the speed and position of the motor. Some stepper motors come with an encoder that keeps track of the position of the motor and communicates this to the stepper driver.

Using this encoder, a stepper motor can be used for precise movements of the motor. [3.]

The requirements for the stepper motor and driver were as follows: The stepper motor needs to be sized as NEMA 23 and have enough torque to be able to move and bend the coaxial cable. Driver and encoder should be integrated into the stepper motor to simplify wiring and for the driver to support Profinet communication. A stepper motor that met all the requirements was found from Schneider Electric from their Lexium MDrive product line. The Lexium MDrives had built in drivers and encoders with Profinet support, as an added benefit they came with a complete library to be used with the Siemens PLC.

5.4 Actuators

Actuator is a device that moves or moves parts of a device according to instructions from the PLC. Actuators usually require their own power supply to function. This power supply can come in a form of pressurized air, hydraulic pressure, or electrical power. The device has two cutters for cutting the coaxial cable. These cutters work by having linear actuators push and pull a lever mechanism that cuts the coaxial cable. The requirements for the linear actuators were as follows: The linear actuator needs to be powered by electricity and be small enough to fit the device.

5.5 Sensors

Sensor is a device that detects changes and transmits the information forward to the PLC. There are many kinds of sensors that detect different changes like pressure, light, temperature, and many others. Two different kinds of sensors were required for the project. These were an inductive proximity sensor and a thru-beam photoelectric sensor.

5.5.1 Inductive Proximity Sensor and Thru-beam Photoelectric Sensor

Inductive proximity sensors work with the principle of electromagnetic induction to detect metallic objects. When these objects pass through the magnetic field generated

by the sensor the change in the magnetic field is detected and the sensor sends the information to the CPU that it is attached to. [4.]

Thru-beam photoelectric sensors work by detecting a solid object blocking a beam of infrared light generated by a transmitter. In normal conditions a transmitter sends a beam of infrared light to a receiver in the device. When an object blocks this beam of infrared light the transmitter detects it, and the sensor activates sending the information to the CPU that it is attached to. [5.]

5.5.2 Chosen Sensors

For the device five proximity sensors were required. From these four sensors were used to detect the positions of linear actuators that will be used for cutting the cable. The last sensor was used to detect if the bending pin is extended or retracted. The sensors had to be small because of the space limitations in the device and a sensor from Panasonic was chosen. This inductive proximity sensor Panasonic GX-H12A can detect metal objects from 4mm distance from the sensor.

For the device two thru-beam photoelectric sensor were required to detect the stock coaxial cable in the device. The first thru-beam photoelectric sensor would detect the input of coaxial cable into the device and start the production sequence. The second light gate would detect when the cable has reached the cutter, and this would mark the zero position for production calculation. The only limiting factor for choosing the thru-beam photoelectric sensor was the diameter of the coaxial cable that it needed to sense. For this reason PM-L45 sensor from Panasonic was chosen. For this sensor, the detectable object had to be at least 0.80mm and fit our criteria.

6 Program Creation

The first goal was to create a first version of the programming logic that would run the device. This means that at least in theory it would be able to run the device and all its functions.

Practically the device should be able to sense the cable stock that is being fed in and run the rotation of production according to specifications received from the HMI and return to starting position.

The logic program was designed using ladder diagram (LAD) as a sequential program with three functions and three function blocks that run in the main program. These three functions handle the running mode of the device, outputs, and step execution. The three function blocks with their own memory handle HMI functionality, step definition and required math operations. Splitting the program into different functions made it more clear and easier to work with. All program steps were to be commented so that in future working and adding on to the code would be easier. According to this mentality all variables will were explained and named to be easy to work with.

The HMI was designed with function and ease of use in mind. The HMI was designed using WINCC a programming tool in TIA portal for HMI devices. The operator can choose different sets of cable from the HMI for the device to create. These cable sets are handled using a built-in recipe handling on the HMI. The state of production and the device should be apparent from the HMI to the operator. This was done with a simple illustration of the process and alarm view for checking problems.

6.1 Programming Software

In this chapter all the software used in programming the device are clarified. The software was chosen based on the used hardware as use of proprietary software was required to program the hardware.

6.1.1 Tia Portal

Control program for the coaxial bending device was created using Siemens TIA Portal V16. Within Siemens TIA Portal V16 there are multiple modules for programming different hardware for example STEP 7 for programming the PLC and WINCC for programming HMI displays. In this project both STEP 7 and WINCC were used for the creation of the program.

6.1.2 Lexium MDrive Software Suite

Lexium MDrive Software Suite is a tool to parametrize the stepper drivers. This software is used on all different models of drive control, for this project the ethernet interface tool was used and this can be seen in figure 3. All stepper drivers require their own IP-address to be set with this tool. The default IP-address set from factory was 192.168.33.1. To access this subnet the computer requires a static IP-address in the same subnet as the drive so the physical port on the working computer was set to 192.168.33.50.

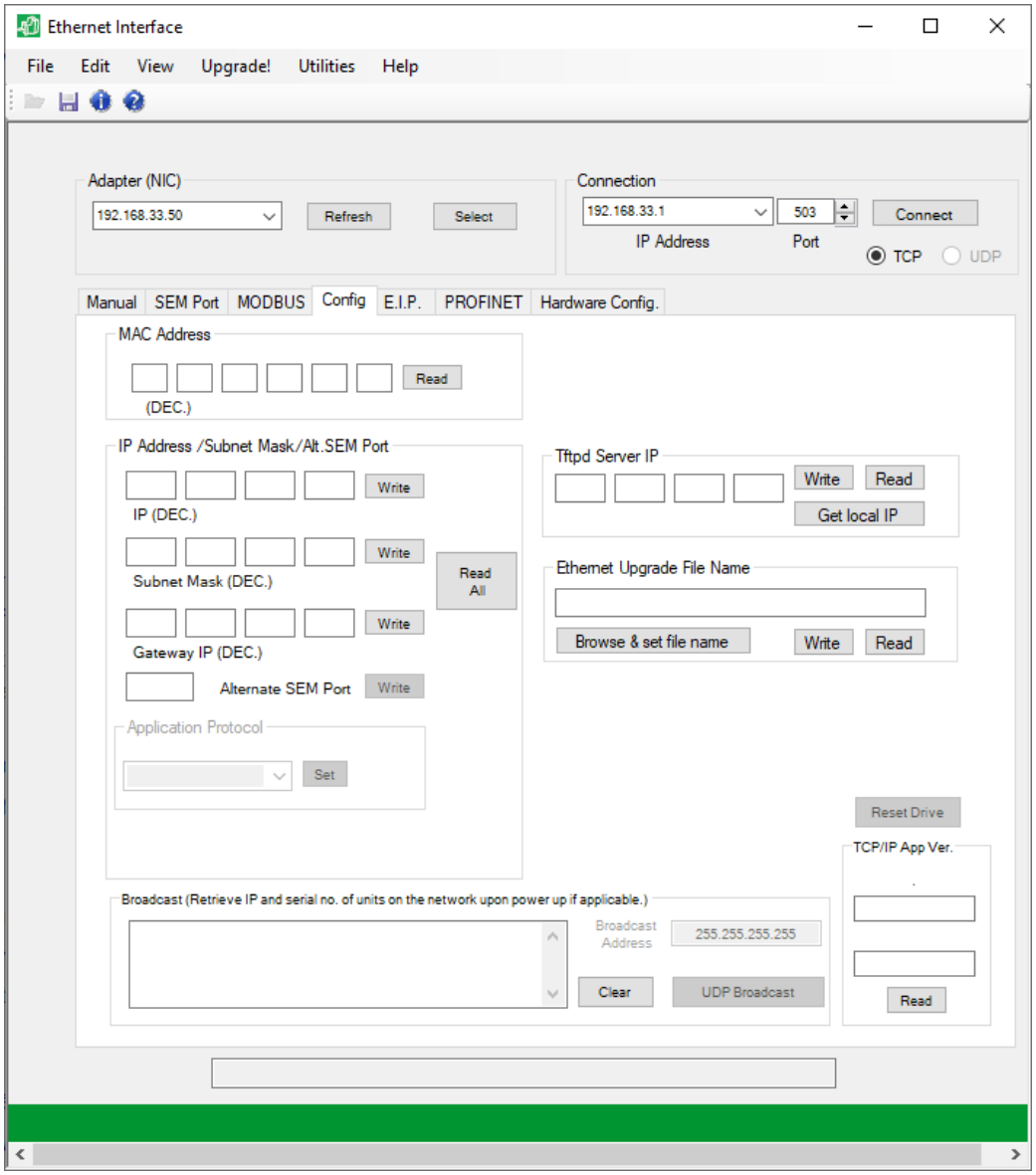

Figure 3. Lexium MDrive Ethernet Interface.

After connection to the drive was established, a password had to be set for the ethernet interface. The drive can use three different interfaces which are Profinet IO MODBUS/TCP and Ethernet/IP, as a PLC from Siemens was used to control the device Profinet IO was chosen as application protocol. Profinet IO devices require unique names and IP-addresses which were set with the tool. These IP-addresses and Profinet IO names can be seen in table 1.

Table 1. IP-addresses and profinet names.

IP-address	Profinet IO name
192.168.33.2	feed1
192.168.33.3	feed2
192.168.33.4	bend

For homing the stepper motors each drive requires a homing switch that is directly wired to the drives input channels. These inputs were set to be homing switches with the hardware config section of the tool. After all the parameters were set the drives were ready for use with the Siemens PLC.

6.2 TIA Portal Project Creation and Hardware Configuration

Programming started with the creation of a TIA portal project and then opening project view. The project view can be seen in figure 4. From the project view a device could be added through the “add new device” button. There can be multiple variations of the same PLC so the choosing the correct article number and firmware is paramount. For this project, the CPU 1214FC DC/DC/DC with article number 6ES7 214-1AF40-0XB0 and firmware number 4.3 was chosen.

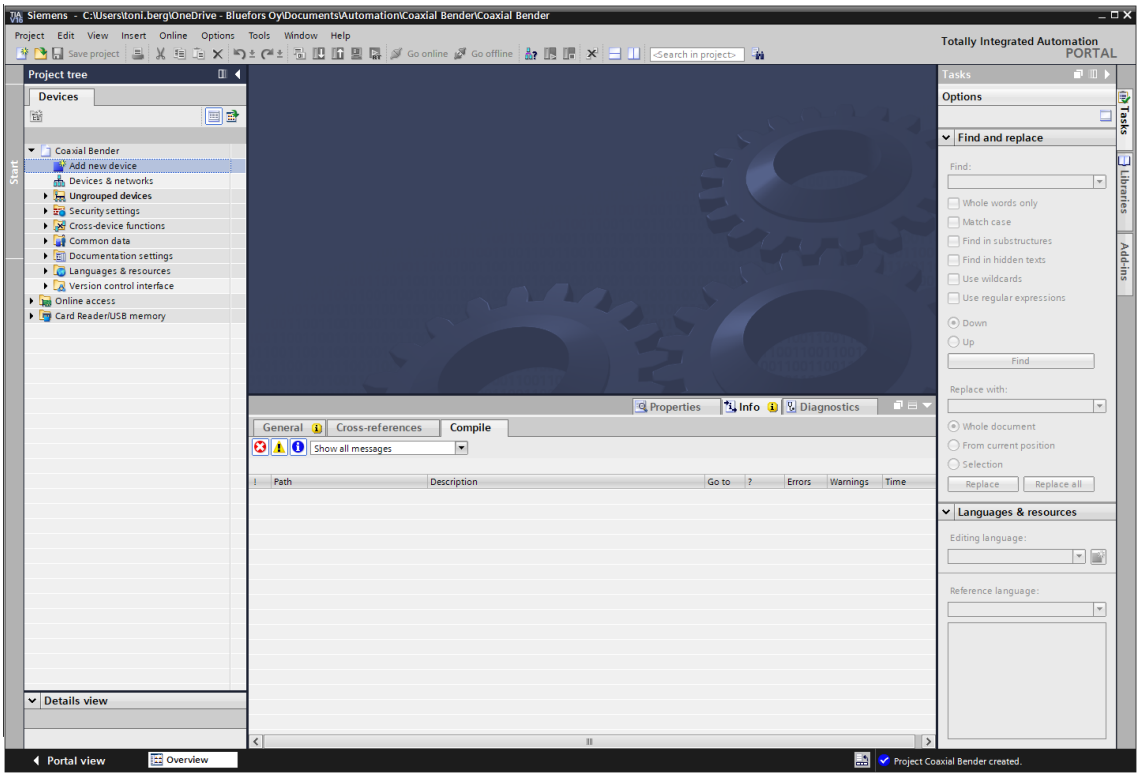

Figure 4. TIA portal project view.

After the PLC was chosen the PLC rack could be configured. The failsafe digital input card F-DI 8/16x24VDC with article number 6ES7 226-6BA32-0XB0 was dragged and dropped from the catalog to the rack. No other devices were directly connected to the PLC, so the rack configuration was complete. The PLC required a Profinet name and IP-address to work properly with other devices in the network. These were set as PLC_1 and 192.168.33.1 with a subnet mask 255.255.255.0. The system and clock memory bits were also activated from the plc settings.

The HMI panel was then added through the add new device button and the correct SIMATIC Basic Panel was chosen from the catalog. The panel was KTP700 Basic with article number 6AV2 123-2GB03-0AX0. The HMI required a Profinet name and IP-address to work so these were set as HMI_1 and 192.168.33.5 with the same subnet 255.255.255.0.

Adding the stepper drivers to the hardware config was done according to instructions from Schneider Electric using their GSDML files that can be downloaded from their

website. These files were placed in the TIA portal installation folder and installed from the project view through options menu and manage general station files (GSD) option.

After installing the GSD files the libraries were added through options menu and global libraries option. From this menu the archived library for the drivers was installed. After the libraries were installed the data types were added to the project and the drives were added to the hardware config as Profinet IO devices. All the drives and the HMI were then connected to the same network in the topology view. The physical connections were done using a Profinet switch.

Each drive required I/O addresses to be assigned to the modules and these were done by adding input and output modules in the device view of the drives. Adding these modules assigned input and output addresses to the drives that were used in PLC tags.

For data exchange between the devices PLC tags were defined for input and output sides. These were done in the PLC tags menu. Each drive requires 128 bytes of input and output memory and the tags should be named so you can easily differentiate between the two. The data types of these tags were “TLexium_MDrv_Drv2Plc” for the input side and “TLexium_MDrv_Plc_2_Drv” for the output side. The addresses for the tags were taken from the drive device view and the data exchange was then complete.

Finally, the actual function block for controlling the drives was added to the project by dragging it from the library. Each drive required its own function block and database to function properly.

6.3 Main Program

The main program OB1 is the heart of the code that is always run and every function and function block in it will be run in cycles starting from network 1 and continuing for as many networks as is used. For this project, the three functions and three function blocks were called in the main program as follows. First network operating modes, second network HMI, third network step definition, fourth network step execution, fifth network output assignments and sixth network Math as seen in figure 5.

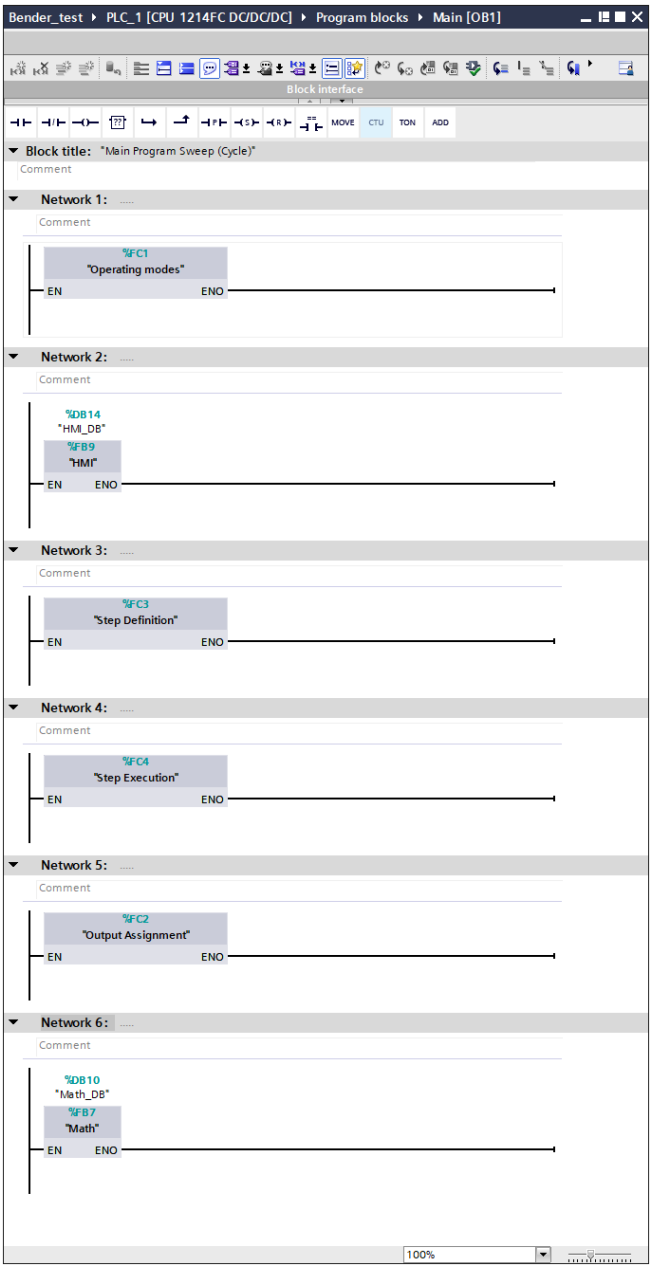

Figure 5. Main program and its networks.

6.4 Function Operating Modes

There are four operating modes for the device error mode, pre-production mode, manual mode, and automatic mode. Error mode is active if any of the steppers are in error state or emergency stop is triggered. Pre-production mode is only active when the device is started and none of the other modes are active. In manual mode the operator can manually move the motors and actuators of the device through the HMI screen.

Accessing this mode will be password protected so normal operators will not be able to access it as with manual operations serious harm can be caused to the device or the operator if used incorrectly. Automatic mode is activated when a recipe is loaded from the HMI.

6.5 Function Block HMI

The HMI function block has eight networks, and the first network moves recipe data from the HMI to production control. This is done by moving an array of 11 DINT variables from “Cable_Length” array to the “Step Data” database. In the “Step Data” database there is an array of 11 DINT variables called “ProductionControl.CableLenghts[0]” that stores the values for use in production. This can be seen in figure 6.

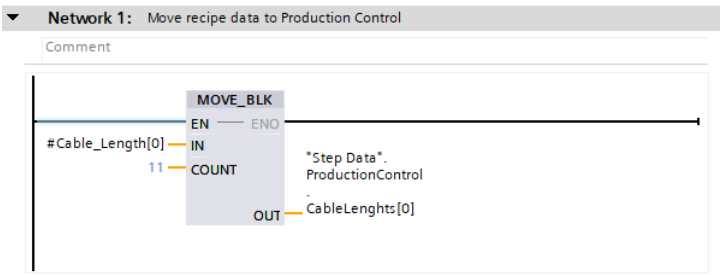

Figure 6. Network 1 of HMI Function Block.

The second network collects and sets the error bits for HMI alarms. During the writing of the program and testing there was only a single emergency stop button that was in use and the error bit for this button is set in this network when the emergency stop is activated. This can be seen in figure 7.

Figure 7. Network 2 of HMI function block.

The networks three, four and five handle manual operation of the bending pin movement. The device must be in manual mode and the stepper motor must be enabled for any of the manual control to work. If one direction of movement is active the other direction cannot be activated at the same time. Network three moves the bending pin a set amount to the left, this amount is set on the HMI to the variable “BendAmount” which is moved to stepper driver through the variable "Step Data.BendStepperBits.BendPOS”. After moving the bending amount a memory bit “BendLM” is set to start movement of the motor. Network four does the same as network three but to the other direction. The amount of movement comes from the same variable “BendAmount” and is multiplied by -1 to achieve movement in the other direction. This negative value is saved in a variable called “BendAmountRight” and sent to the driver through variable "Step Data.BendStepperBits.BendPOS”. After moving the bending amount a memory bit “BendRM” is set to start movement of the motor. The movement is controlled through the HMI with buttons to left and right. Movement of the bending pin is restricted to be less then 20000 steps and when 20000 steps is reached no more movement is allowed. These limits ensure that manual control cannot cause a collision of the bending pin. Network five reads the memory bits “BendLM” and “BendRM” and starts motion of the motor by setting the variable "Step Data.BendStepperBits.StartBendM”. These can be seen in figure 8.

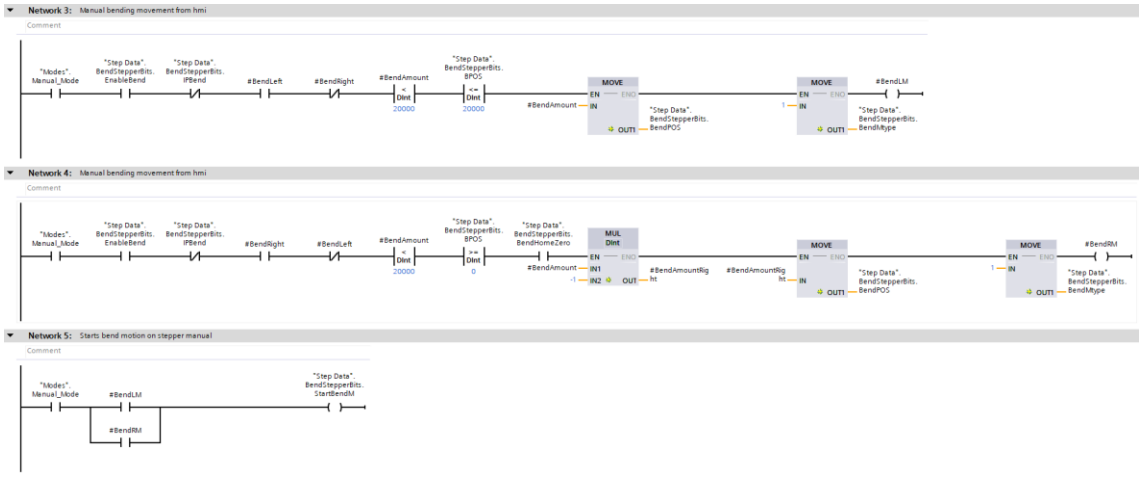

Figure 8. Networks three to five of HMI function block.

The networks six, seven and eight control manual movement of feeding stepper motors by one-millimeter increments. The device must be in manual mode and the stepper drivers need to be enabled for manual control to work. If one direction of movement is

activated the other direction of movement cannot be activated at the same time. Network six controls movement of the feeding stepper motors pulling cable into the device. One-millimeter movements inwards are achieved by moving “-106” in to the variable "Step Data.FeedStepperBits.FeedPOS” and setting a memory bit “FeedINM”. Network seven controls movement of the feeding stepper motors to the other direction pushing the cable back out of the device. One-millimeter movements outwards are achieved by moving “106” in to variable "Step Data.FeedStepperBits.FeedPOS” and setting a memory bit “FeedOUTM”. Network eight reads the memory bits “FeedNM” and “FeedOUTM” and starts motion of the feed motors by setting the variable "Step Data.FeedStepperBits.StartFeedersM”. These can be seen in figure 9.

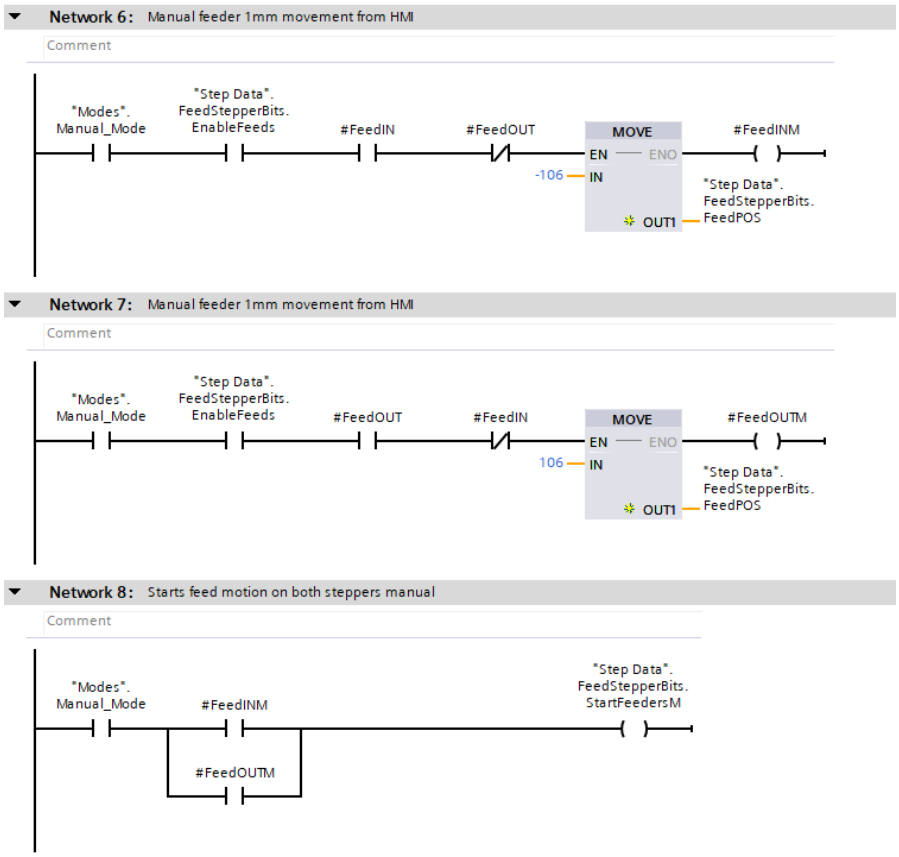

Figure 9. Network six to eight of HMI function block.

During testing it was found out that when starting the device and trying to start production for the first time all the drives go into the same error state when homing. This error has the code 85 and is caused by trying to home while moving or trying to move while homing. Network nine is to handle this startup error by reading the error

codes from all stepper drivers and if all of these are with the code 85 a memory bit “#StartupError” is set and a button to clear the error is visible on the HMI. No actual reason for this error has been found and it only occurs when the device is restarted so a button to easily clear it was created. This can be seen in figure 10.

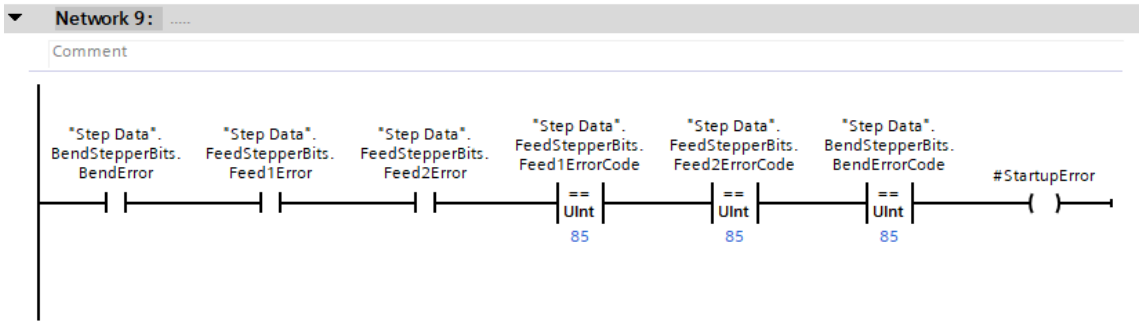

Figure 10. Network nine of HMI function block.

6.6 Function Block Step Definition

Step definition works together with step execution to define and execute all steps required to run the sequence of production. Step definition defines the requirements for each step in the sequence and moves the sequence forward if these requirements are achieved. The sequence can be broken down to four main parts which are material feeding steps, material bending steps, bending direction changes and material cutting.

6.6.1 Material Feeding in Step Definition

Material feeding steps feed the material the required amount and can only move to the next step when the stepper drivers are not in a state of “in progress” and are in the state of “done”. These states are read from the stepper drivers and saved in variables "Step Data.FeedStepperBits.IPF1”, "Step Data.FeedStepperBits.IPF2”, "Step Data.FeedStepperBits.DF1” and "Step Data.FeedStepperBits.DF2”. When these conditions for the movement are met the position of the drives are checked against a calculated limit that is stored in variables "Step Data.FeedStepperBits.F1Limit” and "Step Data.FeedStepperBits.F2Limit”. If these values match the next step can be activated after a ON-delay timer runs out. This step timer is for communication delays

with the stepper drivers and is 25ms on each step. Figure 11 shows an example of a feeding step.

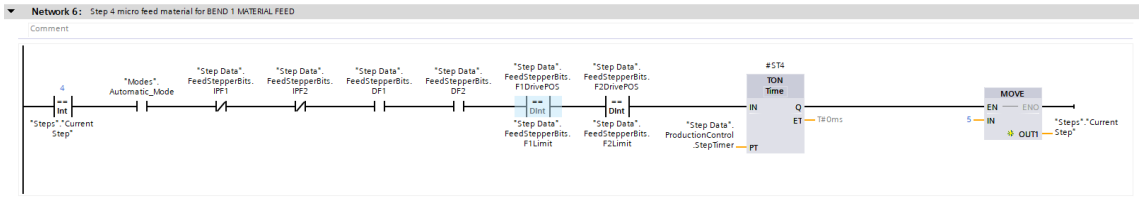

Figure 11. Material feeding step in step definition.

Steps from the step zero to step two are feeding steps with different requirements. These steps are called at the start of production to zero bending pin location for the bending stepper driver and to zero the material location and move it to cutter position.

Step three is the first step of actual production and at the end of step three a CTU counter block is used to count what cable of the recipe is being produced.

6.6.2 Material Bending in Step Definition

Material bending is achieved by multiple small bends and feeds to achieve the required curvature. This is achieved by repeating three steps in the sequence of feeding material for one millimeter, bending the material and returning the bending pin to the original location. The requirements for one-millimeter feeds are the same as stated before for all other material feeding steps. The requirements for first bending movement are for the stepper driver to not be in the state of “in progress” and should be in the state of “done”. These states are read from the stepper drivers and saved in variables "Step Data.BendStepperBits.IPBend” and "Step Data.BendStepperBits.DBend” When these conditions are met the position of the stepper driver is checked against the amount of movement that was sent to the driver. If these values match the next step can be activated after a ON-delay timer runs out, this step timer is for communication delays with the stepper drivers and is 25ms on each step. The requirements for the second bending movement are the same as stated before until the movement to the next step. The next step is determined by a CTU counter block that counts each time the sequence moves back to the first bending step. Each bend has its own limit variable "Step Data.ProductionControl.DEG45Limit1” to

determine the degree of bending. If the counter is below this threshold the sequence will move back to the first bending step and when the limit is reached it will move on to the next step in the sequence. The next step in the sequence will then reset the CTU counter block through the variable "Step Data.ProductionControl.RsetDEG45_1”.

Due to the position of the cutter in the device the shortest cable that we produce needs to be cut in the middle of the bend. This is done on the fourth bend after completing eight bending movements. After cutting the rest of the bending movements are done normally.

The second bending movement has an added condition at the end when doing the last cable that is also the longest cable it will jump to a special feeding step that feeds a portion of the normal feed, cuts the cable with the rear cutter, and feeds the rest of the normal feed. This was done to cut the excess material outside the device for easy removal.

6.6.3 Bending Direction Change in Step Definition

Bending direction change is achieved by three steps that retract the bending pin move the bending pin to the other side of the material and extend the bending pin. The first step requirements are for an inductive sensor to detect the retracted bending pin and then move to the next step. Second step requirements are for the bending pin to have moved to the next position and its state is not “in progress”, is “done” and the location is checked against the amount of movement sent to the stepper driver. If these conditions are met the sequence will move to the next step. Third step checks that the bending pin releases and is not detected by the inductive sensor and can then move on with the sequence. These steps are shown in figure 12.

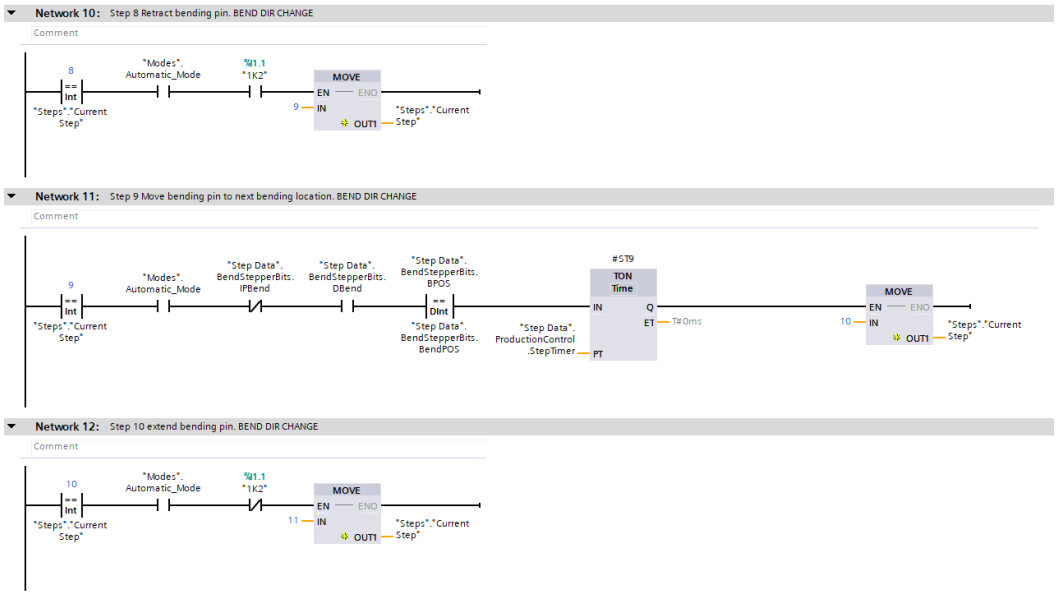

Figure 12. Bending direction change steps in step definition.

6.6.4 Material Cutting in Step Definition

Material cutting was achieved by extending and retracting a linear actuator attached to blades in two steps. First step checks that the linear actuator has extended all the way to an inductive sensor and then moves to the next step. Second step checks that the linear actuator has retracted to another inductive sensor and can then move on in the sequence. These steps are shown in figure 13.

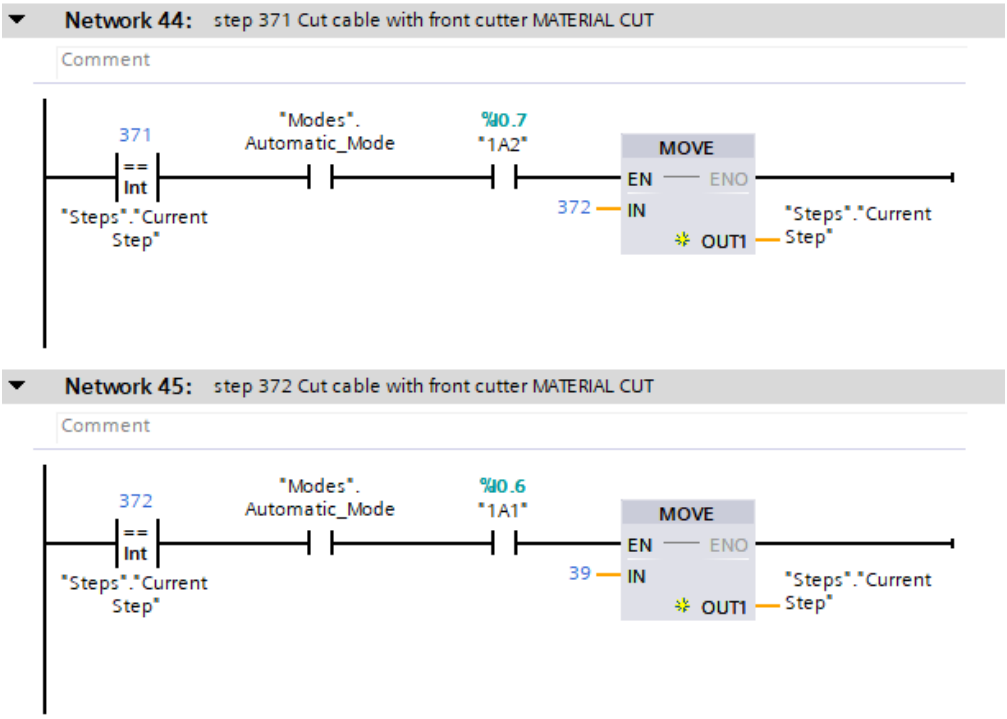

Figure 13. Material cutting steps in step definition.

6.7 Function Step Execution

Step execution works together with step definition to define and execute all steps required to run the sequence of production. Step execution executes the movements and sets outputs as defined in the sequence. Just like the step definition the sequence can be broken down to four main parts that are material feeding steps, material bending steps, bending direction changes and material cutting.

6.7.1 Material Deeding in Step Execution

Material feeding is achieved by moving the feed amount to the variable "Step Data.FeedStepperBits.FeedPOS” and setting a temporary memory bit “SF5”. A temporary memory bit is used to avoid setting the same bit in multiple networks to avoid incorrect states of the output. Each feeding movement has its own temporary memory bit that sets the memory bit "Step Data.FeedStepperBits.StartFeeders” on the positive edge. Some other actions can be executed at the same time as seen in figure 14 where the counter for the first bend is reset by setting the variable "Step Data.ProductionControl.RsetDEG45_1”.

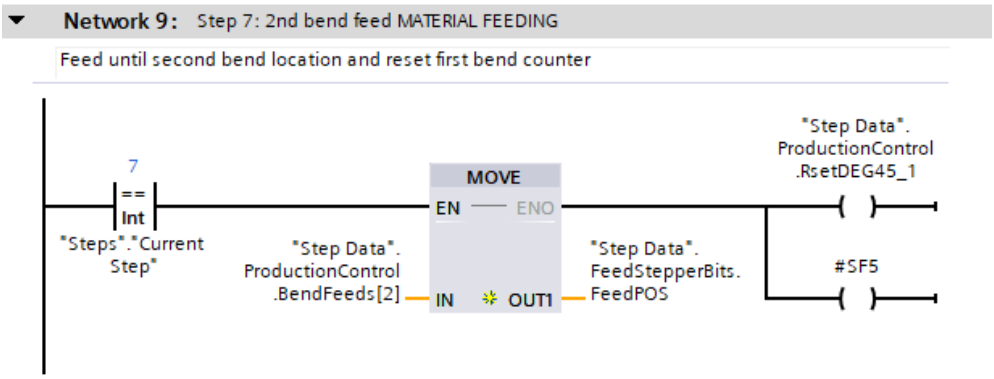

Figure 14. Material feeding steps in step execution.

6.7.2 Material Bending in Step Execution

Material bending is executed in three steps. The first step feeds the material one-millimeter by moving -106 to the feeding stepper driver through the variable "Step Data.ProductionControl.microstep" in to "Step Data.FeedStepperBits.FeedPOS" and setting the temporary memory bit "SF3". As stated before, each feed has its own temporary memory bit to avoid incorrect state of the output. At the same time the value for first bending pin location is moved from the variable "Step Data.ProductionControl.BendPinLocations.BM2" to "Step Data.ProductionControl.BendPinLocations.BMCarrier" on the first cycle. To avoid any kinks in the material during bending the amount of bending is incremental and the "BMCarrier" variable is used to carry the bending pin location through the repeated movements. The next two rungs state the amount of increment on each cycle of the bending movement and is based on value of the limit counter in variable "DRG45Counter1.CV". Each counter has its own variable to store the current count. As seen in figure 15 in the first 6 cycles of the bending movements the increment is 155 steps and for the rest the increment is 0.

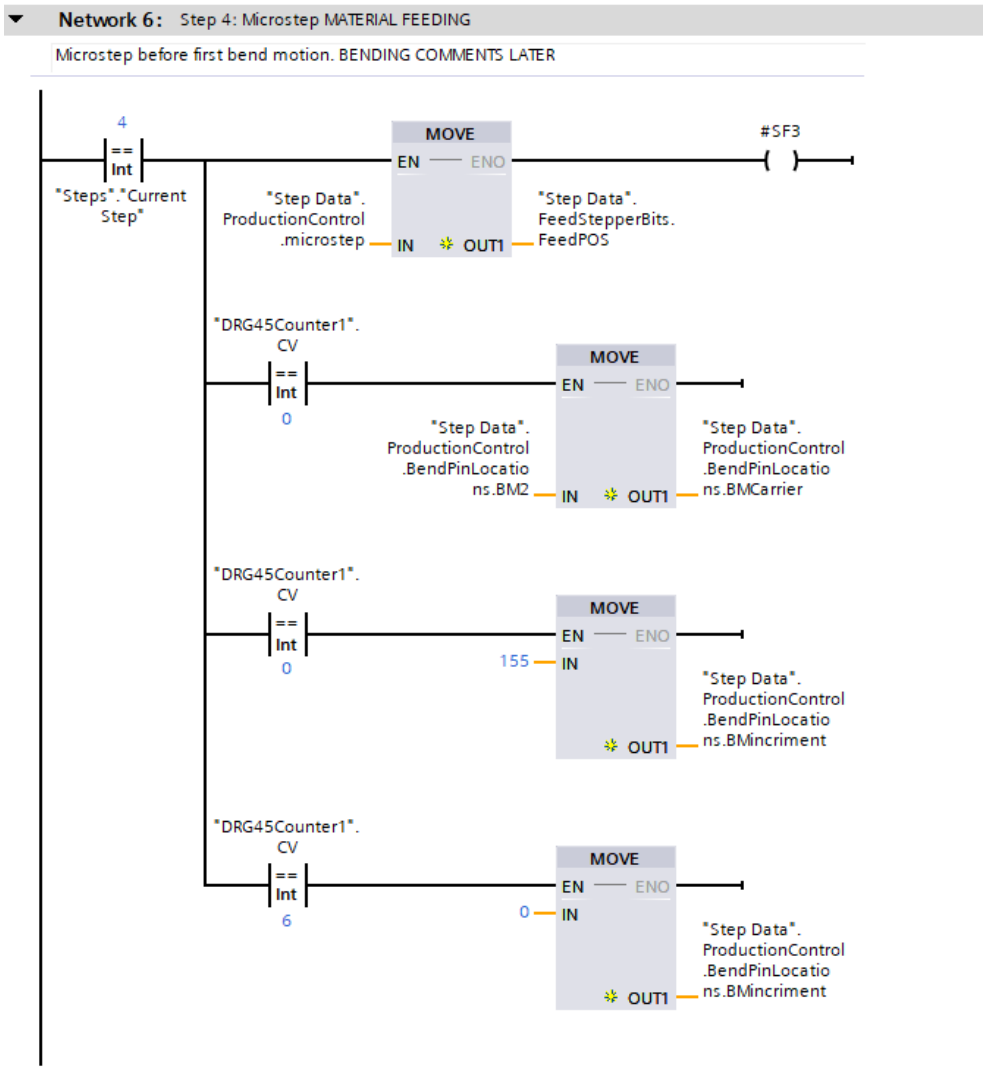

Figure 15. Material bending step one in step execution.

The first bend movement is done by moving the value of variable "Step Data.ProductionControl.BendPinLocations.BMCarrier" to variable "Step Data.BendStepperBits.BendPOS" and setting the temporary memory bit "SB1". Just like with feeding movements each bending movement has its own temporary memory bit to avoid incorrect state of the output. This is shown in figure 16.

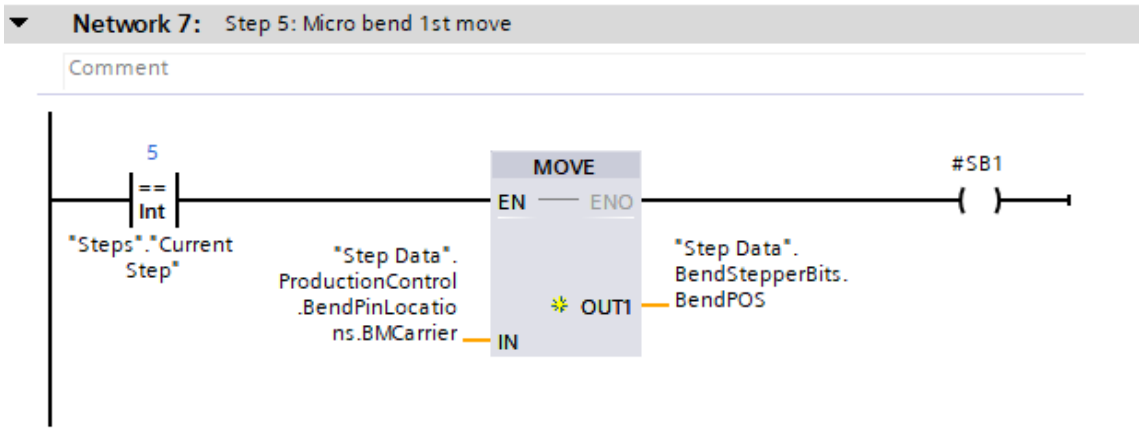

Figure 16. Material bending step two in step execution.

The second bend movement and the last step in bending cycle is activate once with a positive edge trigger. The positive edge trigger ensures that when adding the increment value from variable "Step Data.ProductionControl.BendPinLocations.BMincriment" to the variable "Step Data.ProductionControl.BendPinLocations.BMCarrier" is done only once and saved into the same carrier variable. The bending pin is then returned to the starting position by moving the value from variable "Step Data.ProductionControl.BendPinLocations.BM1" to "Step Data.BendStepperBits.BendPOS" and setting a temporary memory bit "SB2". This is shown in figure 17.

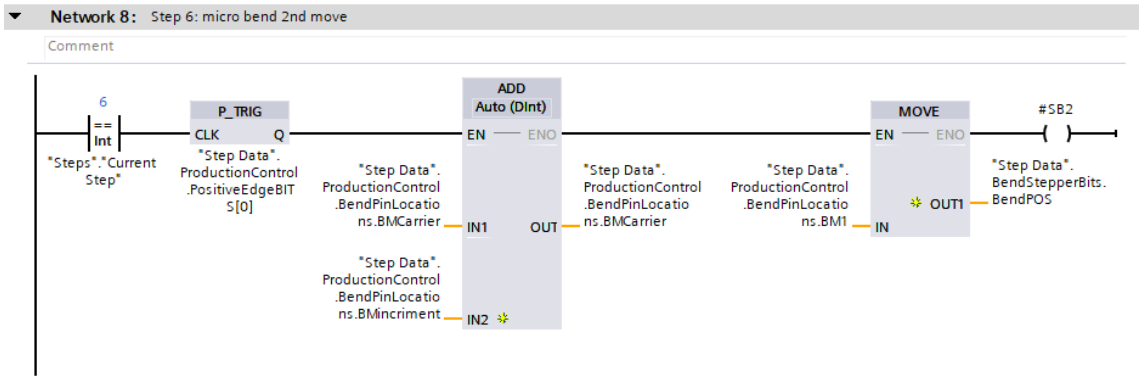

Figure 17. Material bending step three in step execution.

6.7.3 Bending Direction Change in Step Execution

Bending direction change is achieved in three steps. The first step sets an output through the variable "Step Data.Outputs.BendPin" that retracts the pin by activating a linear solenoid where the bending pin is attached. The second step moves the bending pin to the other side of the cable by moving the value of variable "Step Data.ProductionControl.BendPinLocations.BM4" to "Step Data.BendStepperBits.BendPOS" and setting a temporary memory bit "SB3". The third step resets the variable "Step Data.Outputs.BendPin" that extends the bending pin by deactivating the linear solenoid. These networks are shown in figure 18.

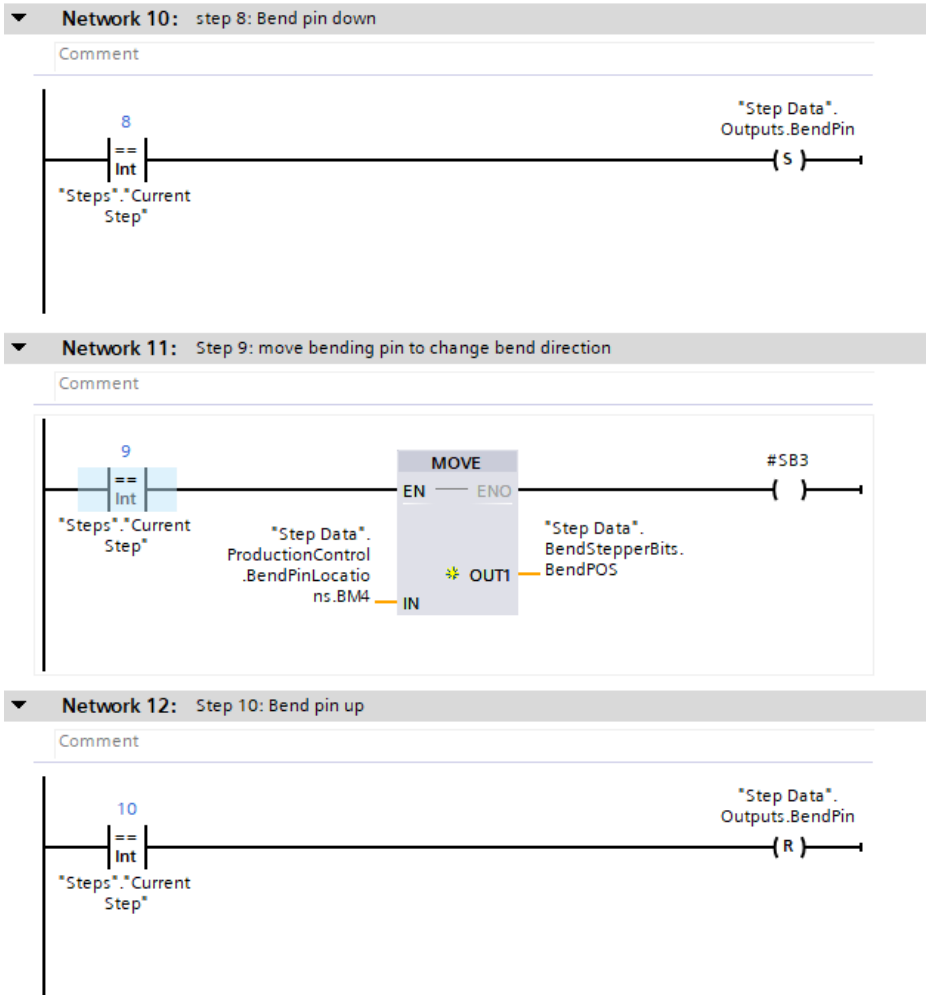

Figure 18. Bending direction change steps in step execution.

6.7.4 Material Cutting in Step Execution

Cutting the material is achieved by setting the variable "Step Data.Outputs.Cut2” that extends the linear actuator attached to blades. This linear actuator retracts automatically when the output is released. The cutter is used in two steps and both steps activate the same output on the same network as seen in figure 19.

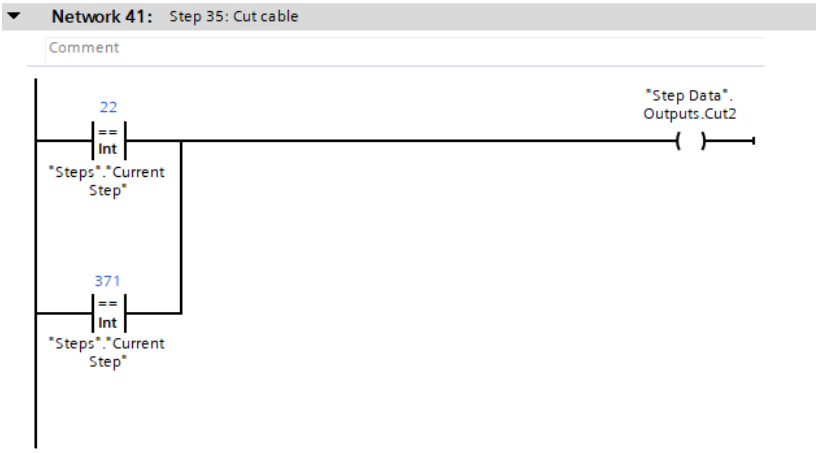

Figure 19. Material cutting step in step execution.

6.8 Function Output Assignment

Output assignment is used to set the actual digital outputs in a centralized manner to avoid causing incorrect states. It also handles communication with the stepper drivers through function blocks from the manufacturer. The first three networks are used for communication with each stepper driver starting with the stepper driver for the bender, feeding driver one and feeding driver 2.

Networks four to nine handle setting of the digital outputs for actuators and lights on the device. There are two linear actuators “1A” and ”2A” that are set in the networks four and five through the variables "Step Data.Outputs.Cut1” and "Step Data.Outputs.Cut2”. These networks are shown in figure 20.

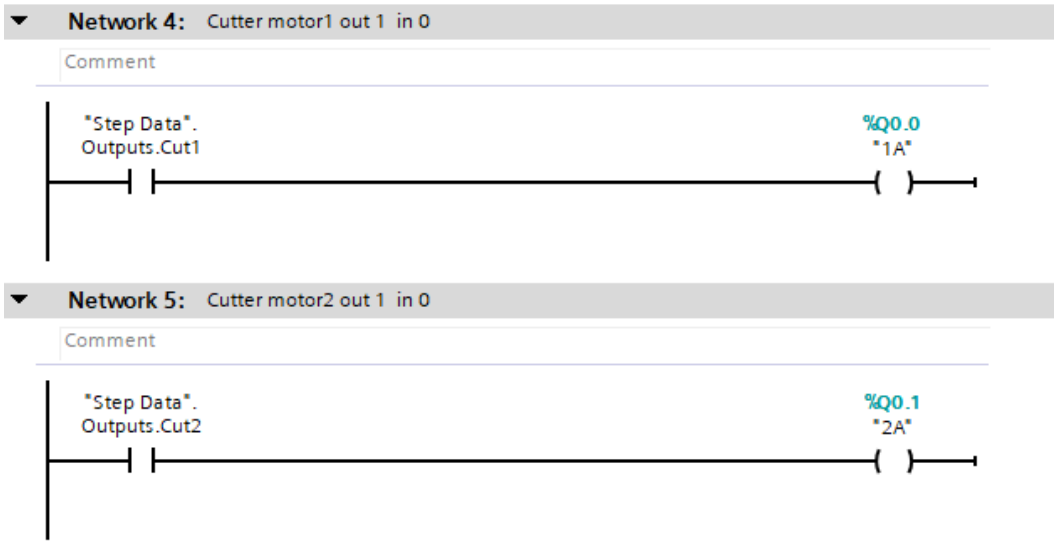

Figure 20. Linear actuator control networks.

Network six is used to activate the linear solenoid “1K” through the variable "Step Data.Outputs.BendPin” as shown in figure 21.

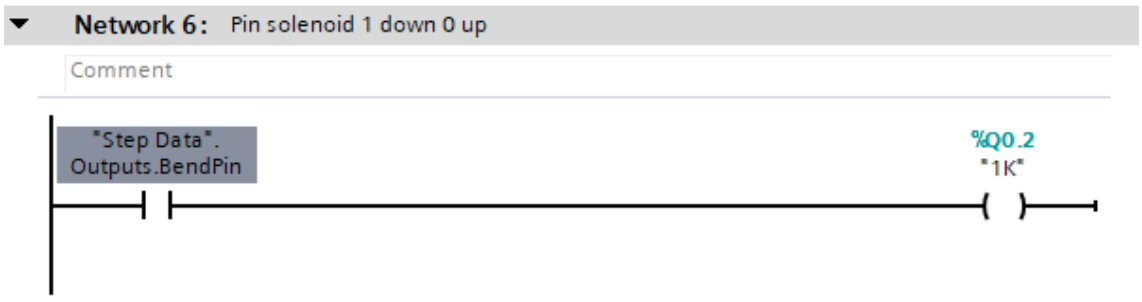

Figure 21. Linear solenoid control network.

Networks seven to nine are used to inform the user of the device state through three different color lights. Different operating modes have different colors. Network seven is used to set the green light “H1” through variables "Modes.Automatic_Mode” and "Modes.PreProd_Mode” so when the device is in either of these states the green light is on.Network eight is used to set the yellow light “H2” through the variable "Modes.Manual_Mode” so when the device is in manual mode the yellow light is on. Network nine is used to set the red light “H3” through the variable "Modes.Error_Mode” so when the device is in error mode the red light is on. These networks are shown in figure 22.

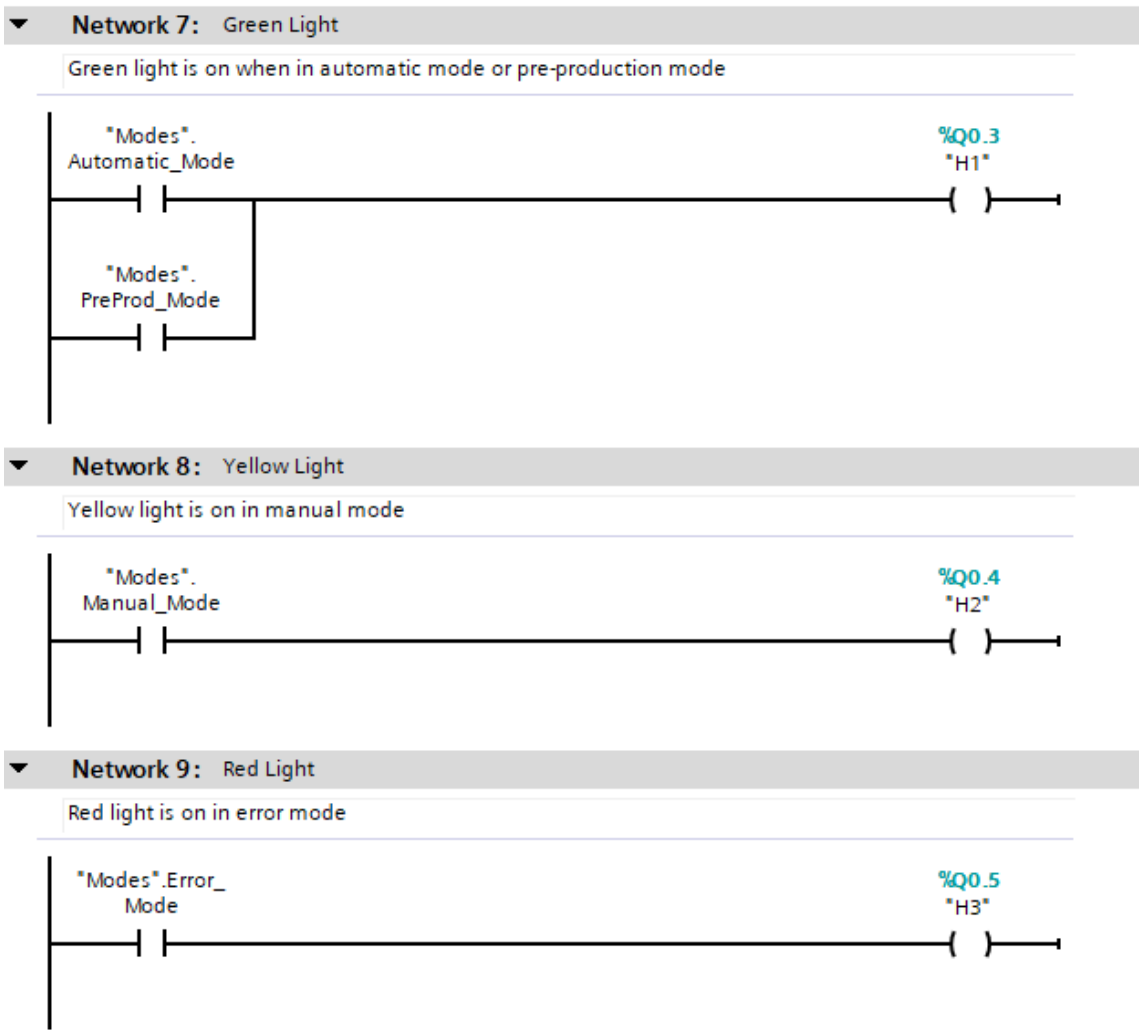

Figure 22. Light control networks.

6.9 Function Block Math

Math function is used to convert steps to millimeters and calculate the starting position of the cable. Due to the design of the device the homing sensor that senses the cable is after the cutter. Because of this the first feeding move of the sequence is to pull the cable back to the cutter and this value is stored in the variable "Step Data.ProductionControl.BendFeeds[0]". To show the user the correct position of the cable in regards to the cutter, the first feed must be subtracted from the actual position from the variable "Feed1_DB.Pos_Fdbk" that comes from the feeding stepper driver and saved in another variable "#Feed1ZeroedPOS". This is repeated to the second feeding driver variables "Feed2_DB.Pos_Fdbk" and "#Feed2ZeroedPOS". These networks are shown in figure 23.

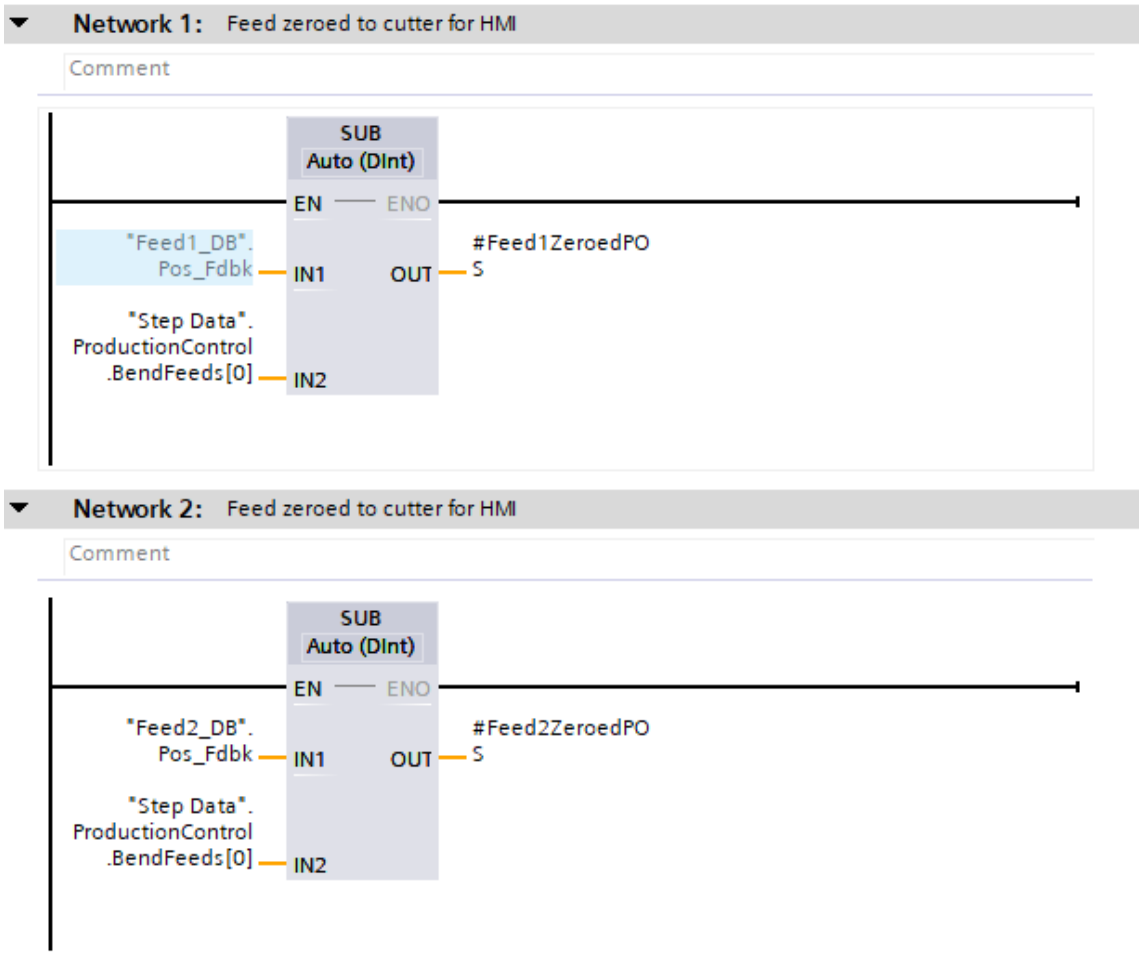

Figure 23. Feed zeroing networks.

These variables that are now zeroed to the cutter position are then multiplied by the calculated one step in millimeters that is stored in variable “#1Step” and saved in variables “#Pos_mm_F1” and “#Pos_mm_F2” as shown in figure 24. These variables can then be used in the HMI to indicate the length of cable that has passed through the cutter.

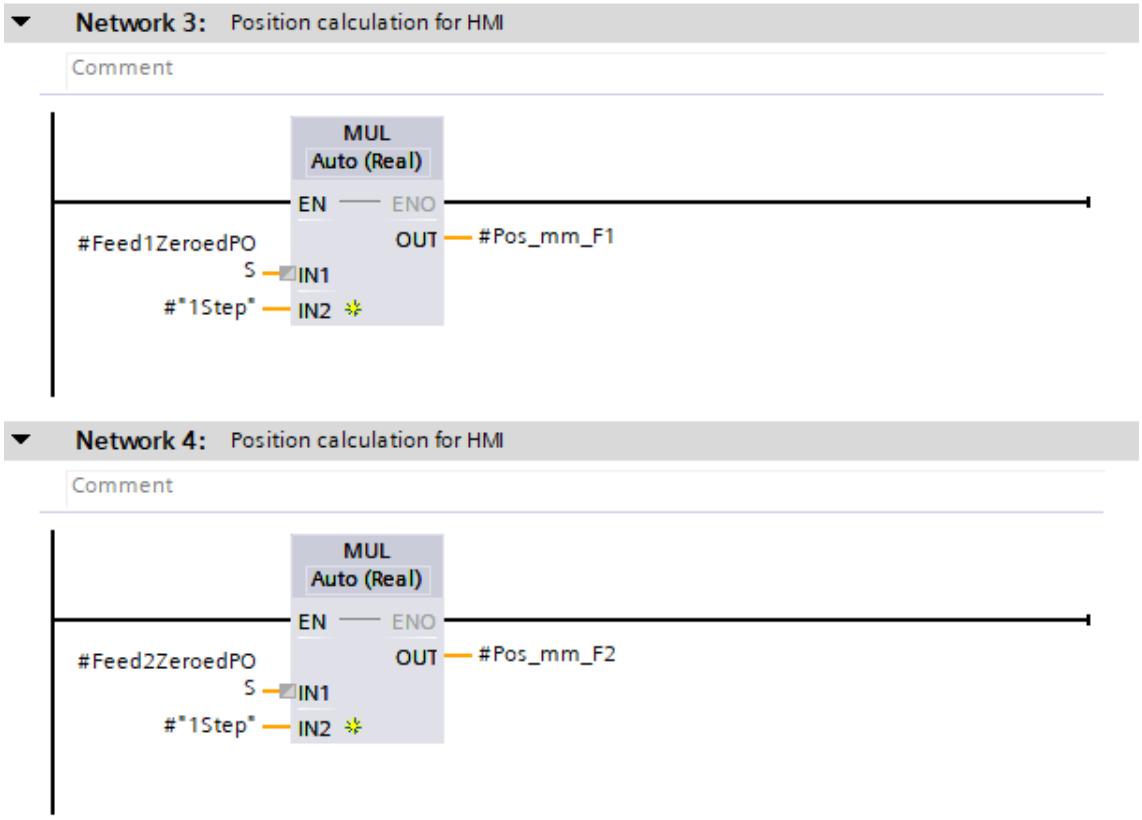

Figure 24. Position calculation from steps to millimetres.

6.10 HMI Screens

The HMI screens were designed with ease of use in mind with enough information for proper use of the device. This was achieved by having a production screen, recipe loading screen, a manual control screen and a menu screen for navigation.

The production screen is what the operator sees when using the device. It has an illustration of the production process indicating at what point the production is at. This is done by having a drawing of the bending profile with indicators that turn green when a portion is done. There are also indicators for the length of cable fed through the device during production run and a list of cables to be produced. This production screen is shown in figure 25. The alarms are shown on the production screen.

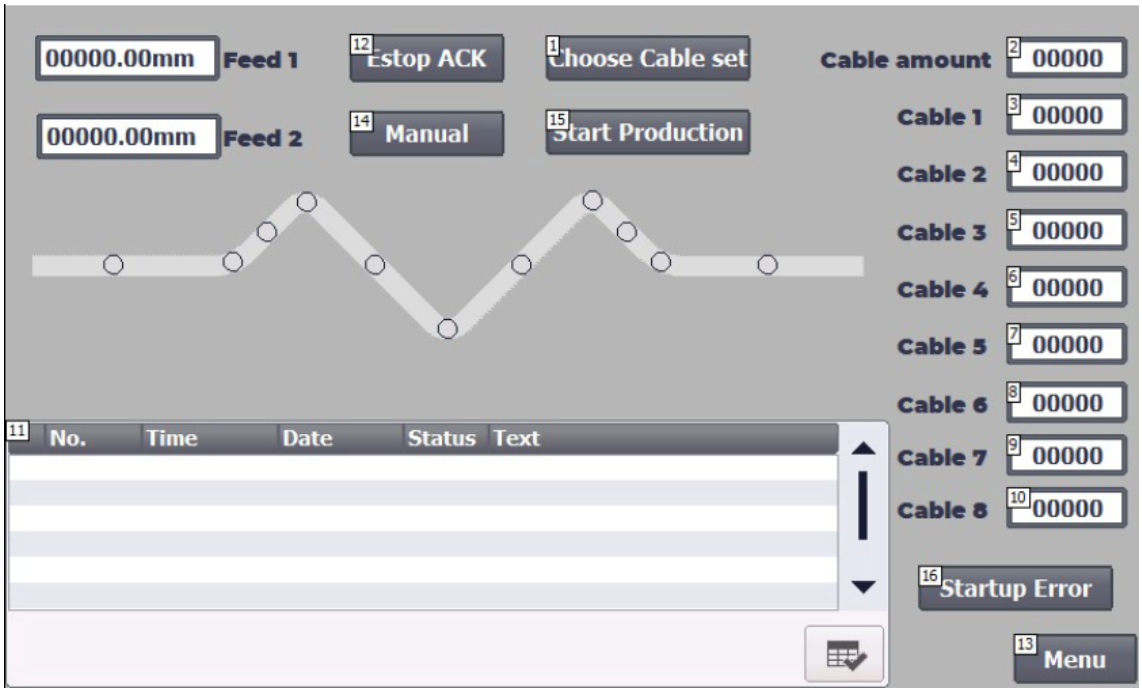

Figure 25. Production screen for the operator.

There are a total of six buttons on the production screen and their functions are as follows: “Estop ACK” Acknowledges emergency stop alarm and returns device to normal condition. “Choose Cable set” Opens the recipe view for so the operator can choose the cables to be produced. “Manual” Open manual control screen. “Start Production” starts production run. “Startup Error” Only shown when the device is powered on for the first time and the drives go into an error state. Pressing will acknowledge error and return the device to normal condition. “Menu” Opens menu screen.

Recipe loading screen is used to choose sets of cable for production and to download it to the PLC. This screen can be accessed through the menu of straight from the production screen. The recipe loading screen can be seen in figure 26.

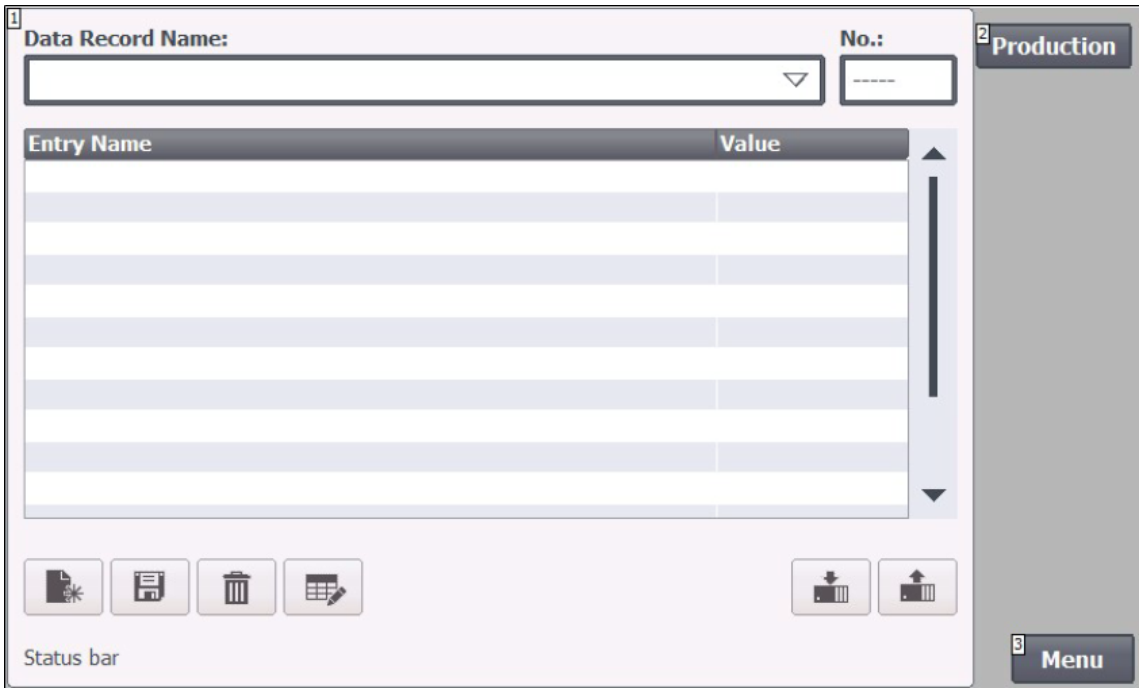

Figure 26. Recipe loading screen.

Manual control screen is only accessible by administrative users as manual control of the device can cause damage to the device or the operator. This screen can be accessed through the menu or from the production screen. In the manual screen there are buttons for manual control of actuators and motors of the device. This screen is shown in figure 27.

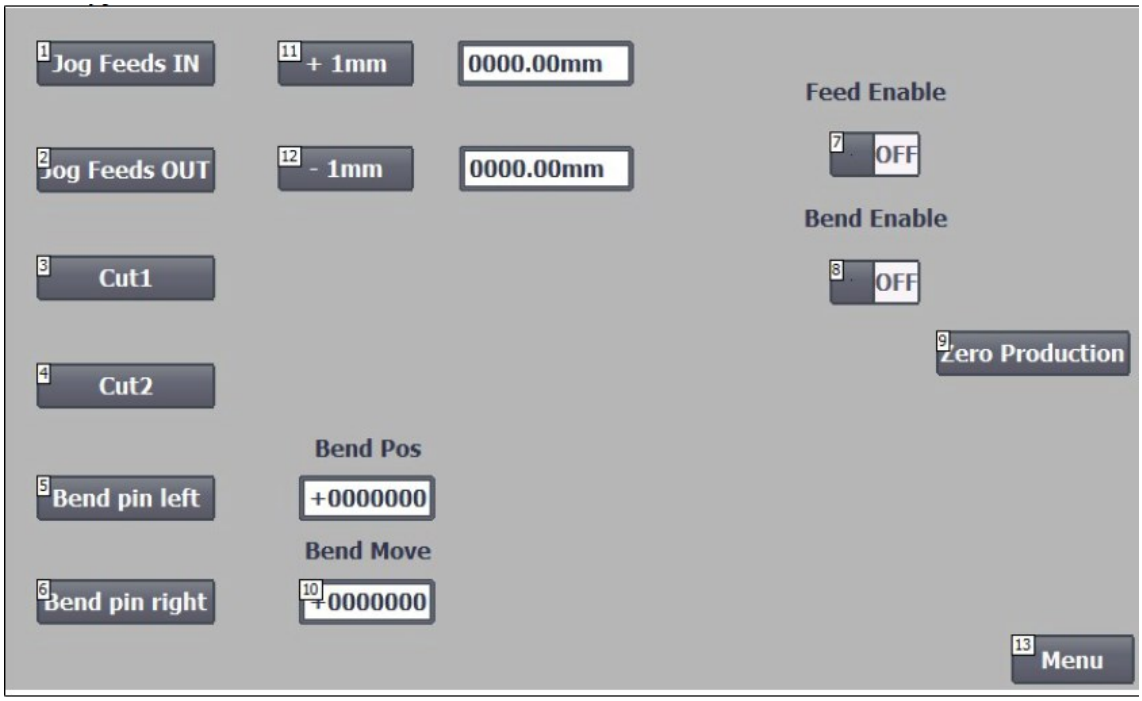

Figure 27. Manual control screen.

There are a total of ten buttons on the manual control screen and their functions are as follows: “Jog Feeds IN” jogs the feed motors inwards when the button is pressed. “Jog Feeds OUT” jogs the feed motors outwards when the button is pressed. “Cut1” preforms cutting action with cutter 1. “Cut2” preforms cutting action with cutter 2. “Bend pin left” moves the bending pin to the left for the amount set in “Bend Move”. “Bends pin right” moves the bending pin to the right for the amount set in “Bend Move”. “+1mm” moves the feeding motors 1mm inwards. “-1mm” moves the feeding motors 1mm outwards. “Zero Production” returns production to step 0 and moves bending pin to starting location. “Menu” opens menu screen for navigation. “Feed Enable” and “Bend Enable” are switches to enable or disable the bending motors.

7 Results

The first version of the control program was created before the device prototype was built and was used as a starting point for the program. For the first version the focus was on creating a sequential base for the program that can create different length cables with the same bending profile. When the prototype was finished the programming continued with the device while testing functionality with a copper wire

with similar physical size to the coaxial cable. Due to the device design the original sequence that was created did not work and the sequence had to be modified. In the original sequence cables were cut after bending and the final cable was cut to length before bending. In the end the working sequence was to have the cables produced in a certain order starting with the shortest cables that had to be cut to length during the bending. Middle length cables were then produced by bending the cable and then cutting it to length after bending was done. The last cable produced by the device had to also be the longest one in production and was cut to length by the rear cutter in between bends to achieve the correct length and leave the excess cable for easy removal.

As the device was still a prototype during the programming period there were mechanical problems that affected the programming of the device. The problems found during testing were fixed by the designer of the device and it was possible to get the program to function correctly. The requirements of having identical bends and correct cable lengths were achieved and the device was put into production.

References

1 Flexible vs. Semi Rigid vs. Rigid RF (Coax) Cable Assemblies. [online]. 2012. Customcable.ca https://customcable.ca/flexible-semi-rigid-rf-coax-cable-assemblies/ 13.01.2012. Read 28.1.2021

2 Bolton, William. Programmable Logic Controllers. [online]. 2006. Pogoni.http://pogoni.etf.rs/VezbeRP/W.Boltorn%20Ladder%20and%20FBD%2001-Ch11.pdf Read 28.1.2021

3 What is a Stepper Motor: types & Its Working. [online]. Elprocus. https://www.elprocus.com/stepper-motor-types-advantages-applications/ Read 30.1.2021

4 Budimir, Miles. What are inductive proximity sensors. [online]. 2019. Motioncontroltips. https://www.motioncontroltips.com/what-are-inductive-proximity-sensors/ 05.04.2019. Read 29.1.2021

5 Allison, Jeff, Frigyes, Gary and Myers, Ed. Fundamentals of Photoelectric Sensors. [online]. 2010. Automation.com. https://www.automation.com/en-us/articles/2014-1/fundamentals-of-photoelectric-sensors 13.06.2010. Read 29.1.2021