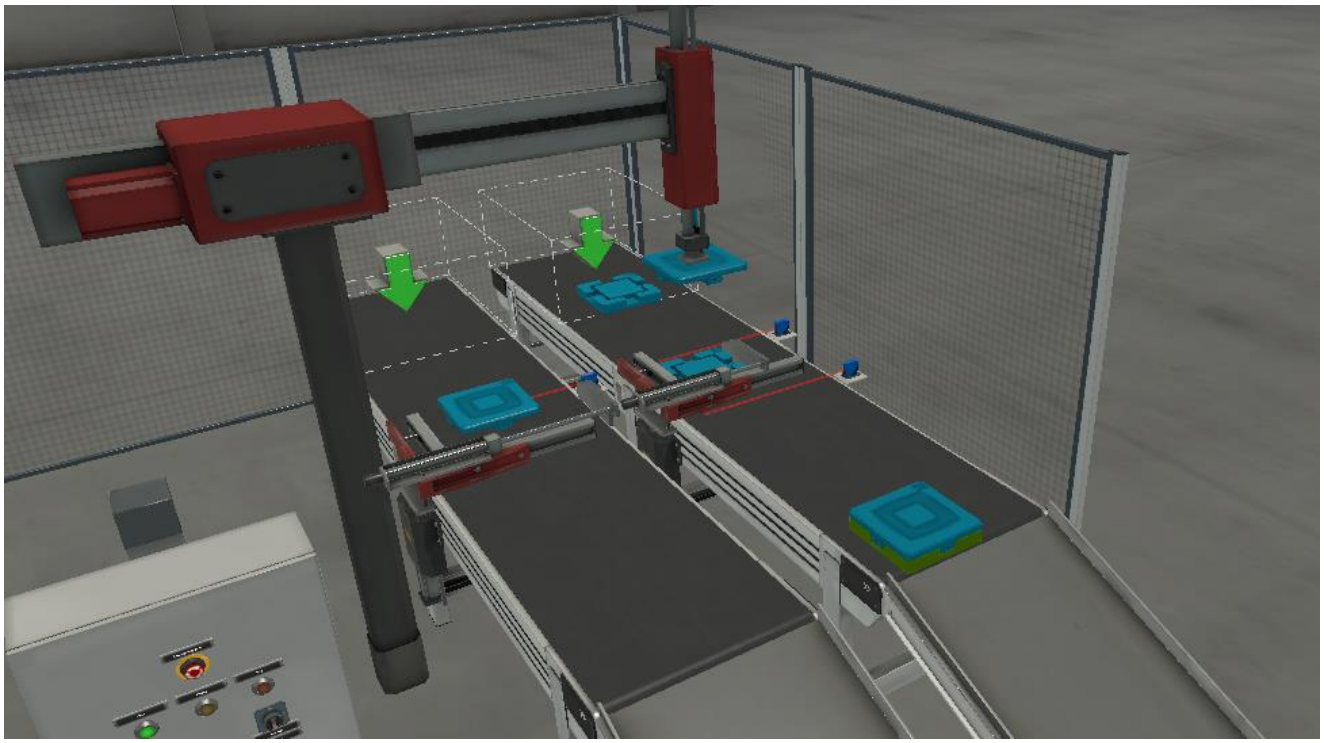
Control System Functional Specification For Assembler Scene

Selvija Armando

4/8/20



Introduction

A control program will be built for an assembler simulated in FactoryIO. The assembler uses two conveyors, a two-axis robotic arm and other sensors and actuators to couple lids and bases together. The project has the purposes of learning the best ways to design and write control software keeping in mind readability, reuse, safety, efficiency and ease of troubleshooting of the system. Therefore, the first action of the project is to write a Functional Specification Document. Here we will define all the hardware used: The plc and its modules, the software architecture and capabilities, system safety and modes of using the machine. The whole system will be described on detail for two main purposes: To create a clear image of the whole functioning of the system for the software developer and also to give insight on the innerworkings of it to outsiders who want to understand the project for various reasons.

At the Hardware requirements will be discussed the type of PLC used, sensors and input output modules.

Software Design Specification will be a detailed explanation of the way the program keeps track of actions, counts finished units, changes from one working mode to another, the patterns used and networking.

Functioning Modes part will explain how the machine will start and take the actions one after another in the automatic mode and how it will be controlled in the manual mode.

System Alarms will define faulty conditions and alarms which can happen during the use of the machine and how the control system will deal with them.

Screens will explain the HMIs and other displaying tools that will be used to output information, control in manual mode or troubleshoot the machine, which will be a great help for operators and technics.

Hardware Requirements

The whole system consists of sensors, actuators, buttons and the controller. Most inputs and outputs are digital. The assembler machine is simulated in FactoryIO and the simulation is controlled with the PLC simulator of Siemens. Here we'll get into the details of each part.

Inputs

Name	Data Type	Logical Address	Comment
Moving X	BOOL	%I0.0	Is true when the robotic arm is moving on the X axis
Moving Z	Bool	%I0.1	Is true when the robotic arm is moving on the Z axis
Item detected	BOOL	%I0.2	Is true when the grabber of the robotic arm detects an item at its end

Lid at place	Bool	%I0.3	Is the sensor before the clamper, it is true when the lid that is moving along the conveyor passes in front of the diffuse sensor
Lid Clamped	BOOL	%I0.4	It is true when then the clamper after activated reaches the end of its movement because it has clamped the lid
Pos. at limit(lids)	Bool	%I0.5	It is false when the clamper of the lids is moving vertically
Base at place	BOOL	%I0.6	It is false when the clamper of the bases is moving vertically
Base clamped	Bool	%I0.7	It is true when then the clamper after activated reaches the end of its movement because it has clamped the base
Pos. at limit(bases)	BOOL	%I1.0	It is false when the clamper of the bases is moving vertically
Part leaving	Bool	%I1.1	It is true when the completed part leaving the clamper passes in front of the diffuse sensor
Start	BOOL	%I1.2	It is true when the button is pressed, the physical button doesn't hold its state
Reset	Bool	%I1.3	It is true when the button is pressed, the physical button doesn't hold its state
Stop	BOOL	%I1.4	It is false when the button is pressed, the physical button doesn't hold its state
Emergency Stop	Bool	%I1.5	It is true when pressed and the physical button holds this state until pressed again
Auto	BOOL	%I1.6	It is true when the working mode switch is at the auto position
FactoryIO running	Bool	%I1.7	It is true as long as the simulation has started and it is running

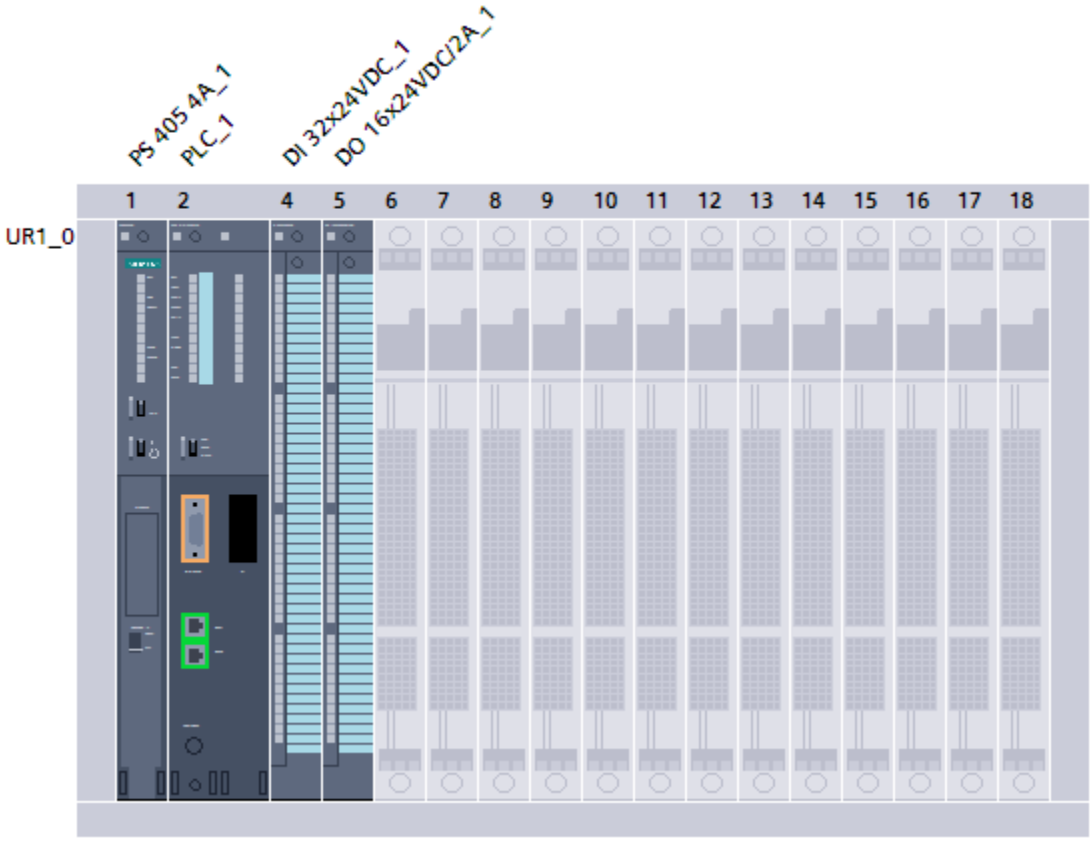
Outputs

Name	Data Type	Logical Address	Comment
Move X	Bool	%Q0.0	When true it moves the robotic arm on the X axis, when turned off it returns it to it origin.
Move Z	Bool	%Q0.1	When true it moves the robotic arm on the Z axis, when turned off it returns it to it origin.
Grab	Bool	%Q0.2	It activated the pneumatic grabber which is located at the end of the robotic arm. When is set to true it grabs the object otherwise it releases the object
Lids Conveyor	Bool	%Q0.3	When true the conveyor that transports the lids starts moving unidirectionally, when false it stops
Clamp lid	Bool	%Q0.4	When true it clamps the lid, when false it releases it
Pos. raise(lids)	Bool	%Q0.5	When true the clamper is moved horizontally to let the part pass in the conveyor, when false it doesn't move.
Bases conveyor	Bool	%Q0.6	When true the conveyor that transports the bases starts moving unidirectionally, when false it stops
Clamp base	Bool	%Q0.7	When true it clamps the base, when false it releases it
Pos. raise(bases)	Bool	%Q1.0	When true the clamper is moved horizontally to let the part pass in the conveyor, when false it doesn't move.

Start light	Bool	%Q1.1	When true it turns on the green light of the start button, when false it turns it off
Reset light	Bool	%Q1.2	When true it turns on the yellow light of the reset button, when false it turns it off
Stop light	Bool	%Q1.3	When true it turns on the red light of the stop button, when false it turns it off
Counter	Bool	%Q1.4	Outputs the number of completed parts

Controller and other modules

As a controller a Siemens Simatic S7 414-3 PN/DP with a PS 405 power supply of 24 VDC/ 4A and two digital input and output signal modules with respectively 32 and 16 connection pins. A cheaper and smaller controller can be chosen for the same task, but since the controller is being simulated and not bought there is no problem. The digital input module works with 24 VDC and the output module with 24VDC/ 2A signals. These modules are connected with the PLC simulator integrated on TIA Portal PLCSim V5.X which is connected to the plant simulating software.



Software Design Specification

Normally the software design part is the most complicated one, because the software needs to written or structured in a way that satisfies requirements such as: safety, quality, performance, readability, modularity, reusability and maintainability. By keeping these ideas in mind we will design the software that will control the assembler machine. A PLC software developer can write code that gets the job done much faster and easier if he doesn't constrain himself by the aforementioned criteria, but on the long run the time and effort saved by this

decision is much smaller than the time that will need to be spent later on troubleshooting problems and rewriting parts of the software. That kind of approach is not efficient and often dangerous. Therefore, let's get into the details of the software design. This part is written for the PLC software developer because this paragraph contains technical terms and practices that normally they are familiar with.

One of the most important things to keep in mind all the time is safety. The machine needs to work in a safe way almost all the time, except the cases when for troubleshooting reasons the safety system gets deactivated by the technician or engineer. Therefore, the software was built with faults and alarms variables within it which when the system detects a dangerous state for the operator or the machine itself during its work the machine stops all movement and other actions need to be taken to assure the safety of the machine by the operators or technicians. These unsafe ways of work are all that damage the machine or the operator. In our case the machine is caged so the interaction with humans is minimal, so we'll focus more on the safety of the machine. The technique that was used to prevent damage is the Five Rung Pattern[1]. This design pattern makes the program capable of making sure a motion was completed as it should before continuing to the next one. If there was a problem during a certain action the machine raises an alarm and stops its work. Therefore, this way the machine remembers which action was problematic and it doesn't damage itself.

To ensure quality of the program we just do a lot of research and brainstorming before writing any code, we fail a lot on our minds and on paper then when we're sure the best solution has been found we start implementing the code.

Performance and efficiency are really important for the company that need the solution integrated in a plant. To test for efficiency and performance in this project we just make sure that we can't find a better way of organizing the work of each part than the current one we have, if we do we implement that one. This iterative way of thinking gets us to the best solution possible.

The code needs to be readable for someone who reads the code for the first time and for those who are troubleshooting it. For this project Ladder Logic was used so to make it more readable we used as little number of variables as possible on each rung, and when there were needed many variables valueholder variables(internal relays) were used. Also every piece of code is placed in a different rung and comments are used extensively throughout the program.

To ensure that the program is modular and its parts are reusable every bit of functionality was written in a different code block so this way that functionality can be used many times.

The maintainability of the machine is ensured by the fact that its program it's easily troubleshooted and modified.

All these qualities of the system make the machine a good investment for the company that would need such a machine, because it can live long, can be easily integrated, modified, troubleshooted and during all its life is efficient.

The machine should take the actions in a chronological order only it also needs to do two movements on the same time because of the efficiency requirement mentioned before so some actions of its work are taken synchronously some asynchronously. The conveyors need to move even when the robotic arm is moving, so this way the arm finds the lid ready when it arrives to the part, this increases a lot efficiency and lowers the cycle time of the machine's work. A pattern that satisfies this is the Step Pattern[2].

[1] This Design Pattern is explained in the Contact and Coil blog.
[2] This Design Pattern is explained in the Contact and Coil blog

Functioning modes

The system in normal functioning it has a simple job: to take lids and bases at the input conveyor, couple them and them to output and count them. This is the automatic mode where everything gets done in the fastest and most efficient way possible. But in cases when problems arise with the control system and those problems need troubleshooting the system needs to work in a way which allows the technician to control its every action, this in turn aids a lot the troubleshooting of the problems. This is the manual mode of functioning.

Automatic Mode

In this mode the machine is completely controlled by the plc program which in turn makes decisions based on the state of inputs, outputs and internal variables. In this mode nothing needs to be done by the operators except starting and stopping the machine.

Manual Mode

In the manual mode the machine is controlled using HMI devices. This mode is created for troubleshooting purposes. Therefore, to aid the technician in finding problems we divide the whole process on actions and then make it possible to activate each action in isolation which in turn isolates the problem. The division of the process into actions will be done not only in chronological order but also based on the machine functionality. Therefore, the actions are divided with the purpose of aiding troubleshooting. In the panel near to the machine there is a rotary switch which controls the functioning of the machine. When set to Manual the machine passes into the manual mode and then can be controlled by the HMI simulated with TIA Portal which will be defined in Screens. actions are:

1. Activate lid conveyor
2. Activate bases conveyor
3. Clamp lid
4. Unclamp lid
5. Clamp base
6. Unclamp base
7. Pick up lid
8. Move horizontally robotic arm
9. Place lid on base
10. Release lid
11. Move robotic arm to origin
12. Move up lids clamper
13. Move down lids clamper
14. Move up bases clamper
15. Move down bases clamper

Faults and alarms will be defined which will indicate what has gone wrong when the machine fails, those will be defined in System Faults and Alarms.

System Faults and Alarms

Faults and Alarms are detected conditions and states of the machine that is either dangerous, erroneous or self-damaging. Therefore, to detect these states we use the PLC program and the information we get from the sensors. The machine is caged so there aren't any sensors or programs to protect operators because they won't be around the machine. Faults are erroneous states which are dangerous for the machine so when they are detected the machine is shut down. Alarms are similar to faults but mostly they don't stop the machine because they aren't as

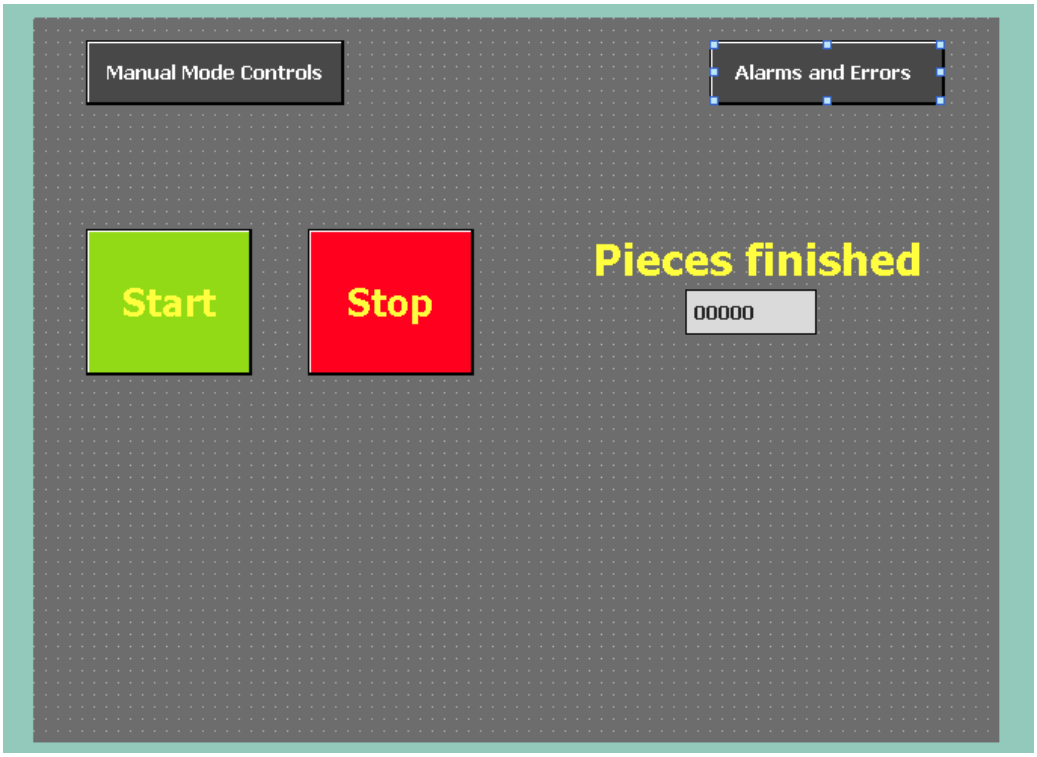
dangerous. Alarms remain latched or active until the fault is addressed manually by a human and then the faults can be cleared and every fault has a unique ID which aids the troubleshooting process.

There are some types of erroneous states that need to be addressed with this machine:

- Fault 001 :A certain movement of some actuator gets blocked, we detect this by using a timer and see if the specified action or movement takes too long to complete.
- Fault 002 :A part which is being transported takes too long to arrive to its destination.
- Fault 003 :The machine doesn't get a new part for a long time.
- Fault 004 :If one of the sequential actions executes with no problems and then the next one in line doesn't there must be a problem such as: a part falls to the ground or gets blocked somewhere. This state causes a fault. **Therefore, if a part needs to be picked up for inspection it can't be retrieved anywhere else but when it is on the entry or exit conveyors, except when it is absolutely necessary the part can be picked and then the alarms should be reset .**
- Fault 005 : If a sensor gets blocked in a certain state which shouldn't have that state for a long time then there must be a problem with the sensor or the working of the system, for example the diffuse sensor normally have a signal value True or High except when something passes in front of them, but if the sensor has the value False or Low for a unusual amount of time for an object to pass in front of it then there must be a problem with the sensor, its connections or the object got blocked.

Screens

There are used 3 HMI screens for this machine, the first one is the root screen where the machine functions in automatic mode and the machine can only be started or stopped. To change the mode of functioning to manual the switch on the panel of the machine needs to be switched to manual and then in the HMI appears a button that takes the operator to the manual controls screen. In this screen all the actions beforementioned can be controlled manually. And then the final screen is the screen that gives information about the alarms and warning of the system.




Home
Hold buttons pressed up to 1 sec
Lid Conveyor OFF
Base Conveyor OFF
Pick up lid
Clamp lid
Clamp base
Move arm horizontally
Unclamp lid
Unclamp base
Place lid on base
Move up lid clamper
Move up base clamper
Release lid
Move down lid clamper
Move down base clamper
Return arm to origin
Home
1/1/1999 12:00:00 PM 4711 Message Text .... Message Text .... Message Text .... Messag...
1/1/1999 12:00:00 PM 4711 Message Text .... Message Text .... Message Text .... Messag...
1/1/1999 12:00:00 PM 4711 Message Text .... Message Text .... Message Text .... Messag...
1/1/1999 12:00:00 PM 4711 Message Text .... Message Text .... Message Text .... Messag...
1/1/1999 12:00:00 PM 4711 Message Text .... Message Text .... Message Text .... Messag...
?
!