UNIVERSITÄT ZU LÜBECK
INSTITUT FÜR TECHNISCHE INFORMATIK

Exercise Sheet 1

Programmable logic controllers

Lecture *Real-Time Systems*, Summer semester 2021

Dr. Javad Ghofrani

A discussion forum for the exercise can be found at: `moodle.uni-luebeck.de`.

Exercise 1 Tutorial

Work through the TIA-Portal/SIMIT tutorial in the Moodle to familiarize yourself with the programming environment of our Siemens PLCs.

Exercise 2 PLC programming

A high bay warehouse is to be controlled by an et200sp PLC from Siemens. Such high bay warehouses are very common in modern warehouses. The warehouse consists of the rack with its bays and a robotic gripper. The robotic gripper can move horizontally, vertically and can move its cantilever forward and backwards to pick containers. Since the warehouse is part of a bigger factory it also has a conveyor belt to move parts to the next station.

The warehouse has four motors for its movement of which the motors of the horizontal- (Y-axis) and the vertical axis (Z-axis) have integrated encoders for position feedback. On the axis with encoders are reference switches installed which are used to find the home position and reference the encoders to it. The cantilever (X-axis) of the robotic gripper lacks an encoder it has two limit switches instead, each at one end of the axis. The last motor is used at the conveyor belt to move a container placed on it towards the next station or back from there to the robotic gripper. On the conveyor are two light barriers to detect if a container is at the start or end of the conveyor belt.

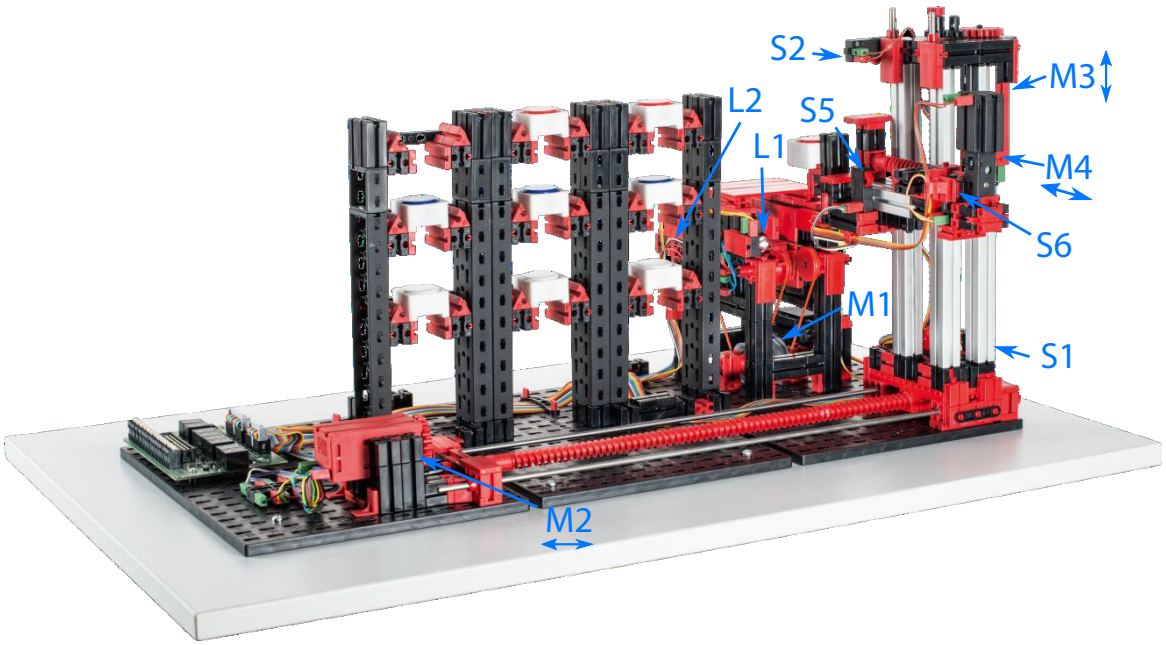

Figure 2.1: High bay warehouse - All variables have the prefix HBW_ and motors a suffix for their spinning direction.

Solve the following subtasks with the template provided in the Moodle.

(a) Homing - Ladder Diagram (LAD)

Whenever the PLC is turned on the reference for the encoders is lost and it is not known where the robotic gripper is.

Complete the function **Home** under "SIM-PLC/Program block/ HBW/Functions" so that the robotic gripper retracts its cantilever and then moves to the reference switches. The reference and limit switches (S1 to S6) return a logical 1 / true when they are activated. Use LAD as the programming language and make sure to set the motor voltage (VMOT) in the home function. When the homing is complete, a done flag has to be set.

(b) Conveyor belt - Function Block Diagram (FBD)

When a container is placed on the conveyor belt it has to be moved from one side to the other. To make sure that it has been transported completely to the opposite side, the motor M1 has to stay on for two more seconds after a container triggered a light barrier. The light barrier L2 is at the side of the next station and L1 is faced towards the robotic gripper. If a light barrier is interrupted it returns a logical 0 / False.

Create a function (not function block) with the name **MoveBelt** in the FBD programming language under "SIM-PLC/Program block/ HBW/Functions" that implements the behaviour. The function gets a light barrier as an input and writes to a motor on an output. The actual IO variables are connected when the function is called. You have to call your function in the function blocks **Put** and **Get** and assign the correct IO-variables.

(c) Control - GRAPH

The high bay warehouse can do two things. The first is picking a container from one of the bays, transporting it to the conveyor, move it to the end of the conveyor belt and wait there for the next station to pick a token that is in the container. The second thing is returning a container (full or empty) to one of the bays.

Complete the **HBW_Control** function block. Therefore you need to call the function blocks (Get) and **Put** depending on the variable **cmd**. If cmd = 0 the robotic griper has to get a container from the warehouse else the griper has to put a container back. The variable **start** tells you when to start the process of getting or returning a container. Whenever a task has been finished the sequence has to return to its ready state.

At this point, you should be able to compile your program without getting errors.

(d) Position control - Structured Control Language (SCL)

The last missing piece is a position controller. The task of the position controller is to read the destination coordinates, calculate the difference to the current position and control the motors to get to the destination. The controller has to decide which motor signals have to be set to move the robotic gripper to its destination.

Complete the **PositionControlerYZ** function block using SCL. Make sure that your controller reaches the target position and that the position error is not greater than **AllowedDeltaY/Z**.

Exercise 3 Simulation

In the initial state of the system, all bays of the warehouse have a container with a token of unknown color. The robotic gripper starts at a random position.

With the Inputs Y and Z, you can select the destination. The lower right bay is 0/0 and the upper left one 2/2. The start button executes the command specified with the Get/Put selector. After a token has been transported to the output to the next station you can take it with the **Take** button. A container on the conveyor is set to a value specified by the **Token** input when the start button is pressed.

Test your program in the simulation. You should be able to move containers and take their tokens.

The simulation implements for some cases a rudimentary collision detection to mimic the real high bay warehouse.