Programming Report for Stations 1 and 2

Christoph Echtinger-Sieghart **(00304130)**
Thomas Hartberger **(01327876)**

May 4, 2018

Abstract

After a brief overview of the task at hand, we will discuss the methodology used for implementing the required functionality. First we discuss our high-level approach and then provide some implementation details. We finish this report with a quick overview of the Human-Machine-Interface.

1 Overview

We were tasked to program the Stations 1 ("Vereinzeln") and 2 ("Prüfen") according to the specification derived during previous modelling using SysML. The programming was to be done using the Siemens TIA Portal application. Since port assignments of the various sensors and actuators were not know beforehand, the first task was to derive these assignments and to check if all the sensors and actuators were working as specified. During this process we discovered that the sensor for detecting the end position of the magazine's slider was non-operational. Since this sensor's data is not critical, we decided to go ahead and create a virtual sensor using a timer based solution.

2 Implementation

2.1 High-level Implementation

Stations 1 and 2 naturally decompose into three finite-state machines. The first state machine being Station 1, the second state machine being Station

2 without the ramp and the third state machine represents the ramp. These three state machines run in parallel, but some state transitions are dependent on another state machine being in a specific state. These dependencies synchronize the state machines at critical points of the operation. A special state machine called `Initialization` is responsible for getting the system into a well-defined state. The `Initialization` state machine is always executed before the system starts normal operation.

A detailed specification of the state machines and their interaction can be found in the combined SysML model created for Stations 1 and 2.

2.2 Detailed Implementation

The major part of the implementation is done using **FBD** - for the small remainder we used **SCL**. We used a structured approach during implementation. First we encapsulated all of the actuators into functional blocks that provide a uniform interface (see Table 1).

Inputs		Outputs
`enable`		`position_reached`
`move_*`		

Table 1: Interface for actuator blocks

The general idea of the interface is as follows: Whenever an actuator is allowed to move it's `enable` input is high. Since all our actuators know only of two directions in which to move, the `move_*` input is enough to specify the desired direction of movement. Whenever an actuator has finished moving in the desired direction the output `position_reached` goes high.

This interface for the actuators was then combined with the state machine approach. The `enable` inputs of the actuators function as a kind of sensitivity list with regard to the states where they should be active. The state transitions are then often dependent on the output `position_reached` going high and some additional constraints. In the case of an emergency shutdown a special state that does not enable any actuators is entered.

3 Human-Machine-Interface

To start normal operation the operator must press the green button. A controlled shutdown can be initiated by pressing the yellow button. Passing a material unit on to the next station is done by pressing the white button. Pressing the red button initiates an emergency shutdown and halts operation immediately.

A detailed description of the Human-Machine-Interface can be found in the use-case diagram in the combined SysML model created for Stations 1 and 2.