TTK4175 - Instrumentation Systems
AIM

Håkon Espeland – 772703
Ali Al-Jumaili – 772170

Group 05

07. February 2017

NTNU – Trondheim
Norwegian University of
Science and Technology

Contents

1 Part 1: Hardware Test

a) When the motor relay[1] is forced by pressing the blue button, the second relay, relay[2] is also triggered. This indicates an interlock between the two relays.

b) Overload on the motor, triggers the motor protection, triggering the third relay, relay[3] and is also forcing off the second relay. In a) it was asked what the interlock indicated; It indicates that it has something to do with the "safety functions", if it do not trigger along with relay[1], the motor will not start.

2 Part 2: Configuration of Flow sheet and Trend

In this part we got to know the basic functions and menus. We also added the motor control module, which at first was tested in internal mode by hooking the "Start" output to the "kontaktor", and sending a running signal back. To make the motor start, we needed to connect the Motor/Start in RMP422.

To implement the logic control, we read a lot on "additional information" on the motor control module, and decided to use the following terminals:

NotOk1 = 1, Motor stopped if it is running and restart is blocked.

NotOk1 = 0, Start possible if external mode is activated and Off is connected.

InEx = 1, External mode.

InEx = 0, Internal mode.

Off = 0, OnOff enabled.

Off = 1, OnOff disabled.

OnOff = 0, initial value.

OnOff = 1, starts the motor if following criterias is forfilled:

InEx = 1, Off is connected, NotOk1 = 0.

We then implemented two "plclg" modules, where the first one triggered the NotOk1 terminal by having: Auto.invMeas **AND** 0or1.invMeas. This logic implies that the motor will always (hence logic) go to "safe-state" when switching to **mode 0**, stopping the motor if it is running and block restart.

The second "plglg" were used to implement the logic for external start. To do this, we used the RS block (Set-Reset), so that we were able to set the OnOff terminal by switching Start at the switch, and also reset the RS block when losing 0or1 high. This creates some form of redundancy in the system, having two logic functions stopping the motor if the switch turns to position 0.

We did not use the terminal NoILock1, which is a terminal that overrides the NotOk1 terminal, to improve the safety complexity. The **InEx** was connected to Auto.invMeas, making internal start impossible when the switch is not in Auto.

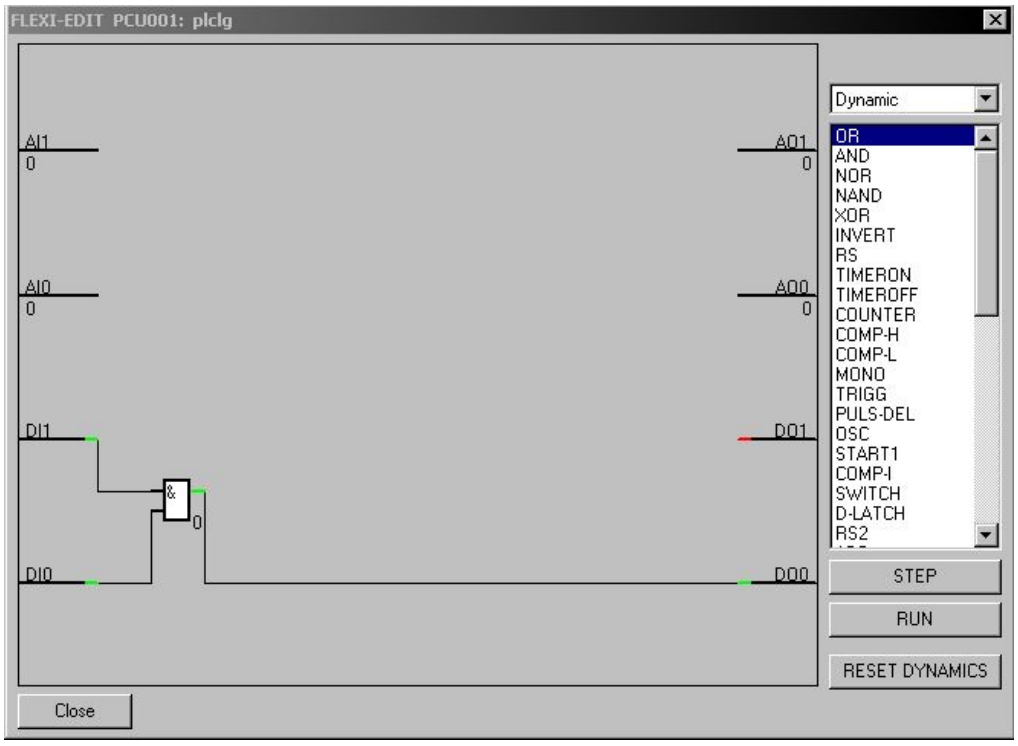

Figure 1: plclg 1

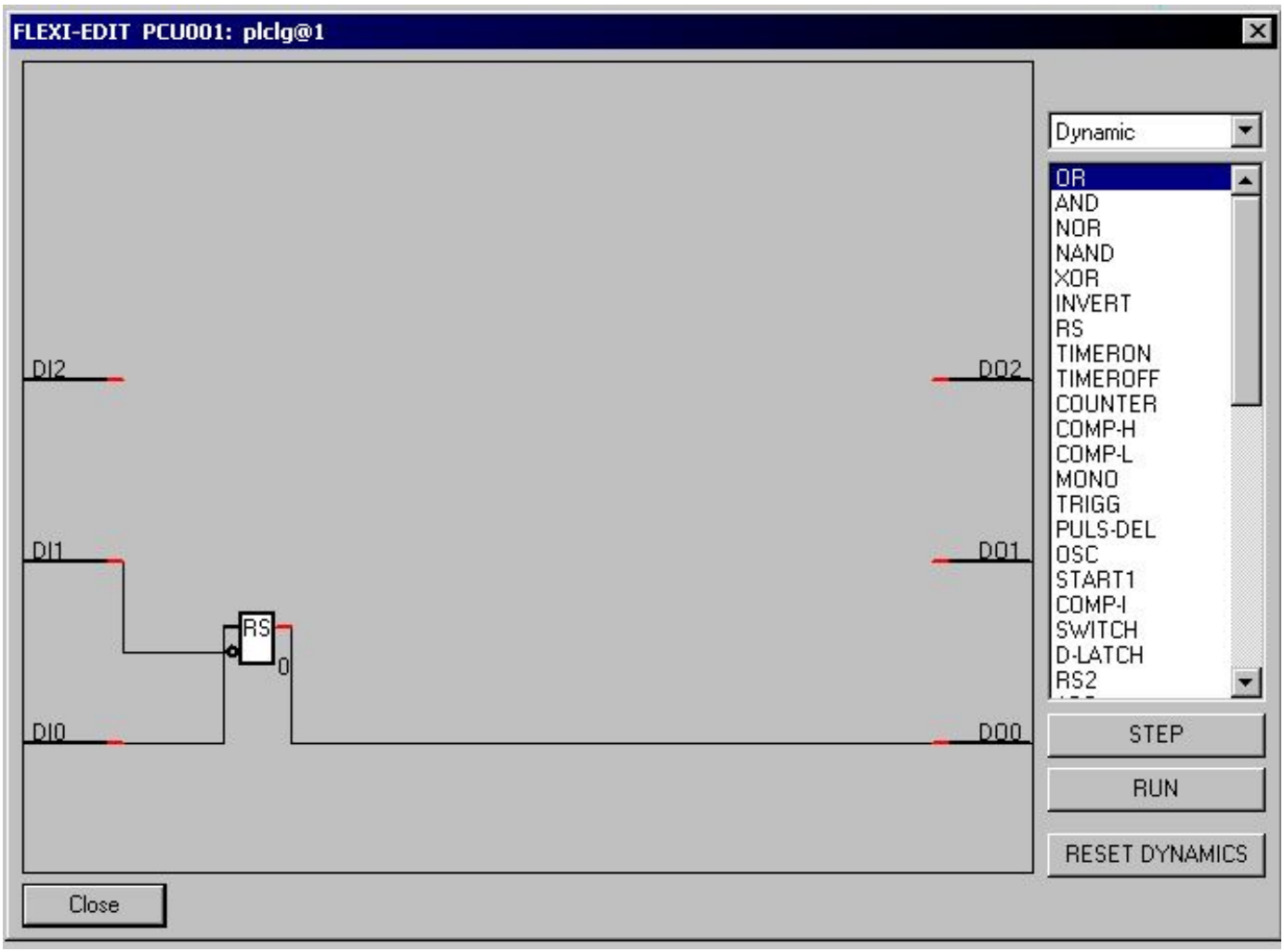

Figure 2: plclg 2

Testing our logic, it turned out very well, functioning as described in the assignment text.

Improving our logic, we could have made everything in one plclg module, with **4 DI** terminals and **2 DO** terminals.

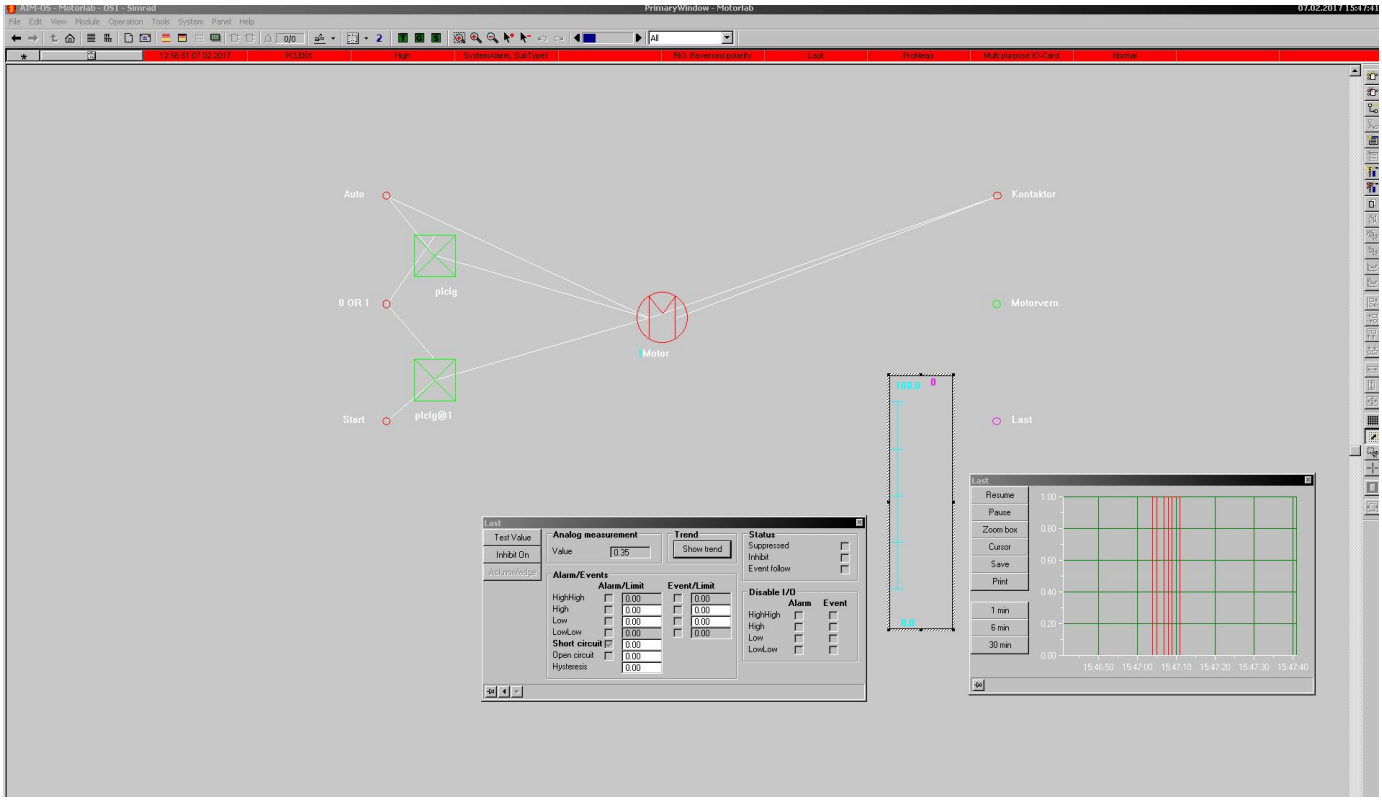

Figure 3: System Overview

When the motor logic functioned as desired, it was time to implement a measurement module for the load. This was done using a bar graph, ranging from 0 to 100. To do this we first created an Alarm.

Scaling the measurement signal in the Parameter View window, was harder than we thought. It seemed like the bias functioned as some kind of off-set, and the gain as some king of multiplier for the signal. Setting these parameters to gain $= 5$ and bias $= 1.3$ made the bar graph represent the load ranged from 0-400 as $0 - 100\%$, triggering the motor protecting when the load closed up on 100%.

We also created a time-series for the load, with historical storage ENABLED.

Creating Trend:

New image created, where a new curve was defined under the name Last/Meas1/PRIM.

Due to the timeseries, we could move in/out of the trend-image. If this was deactivated, the trend would always start from the beginning.

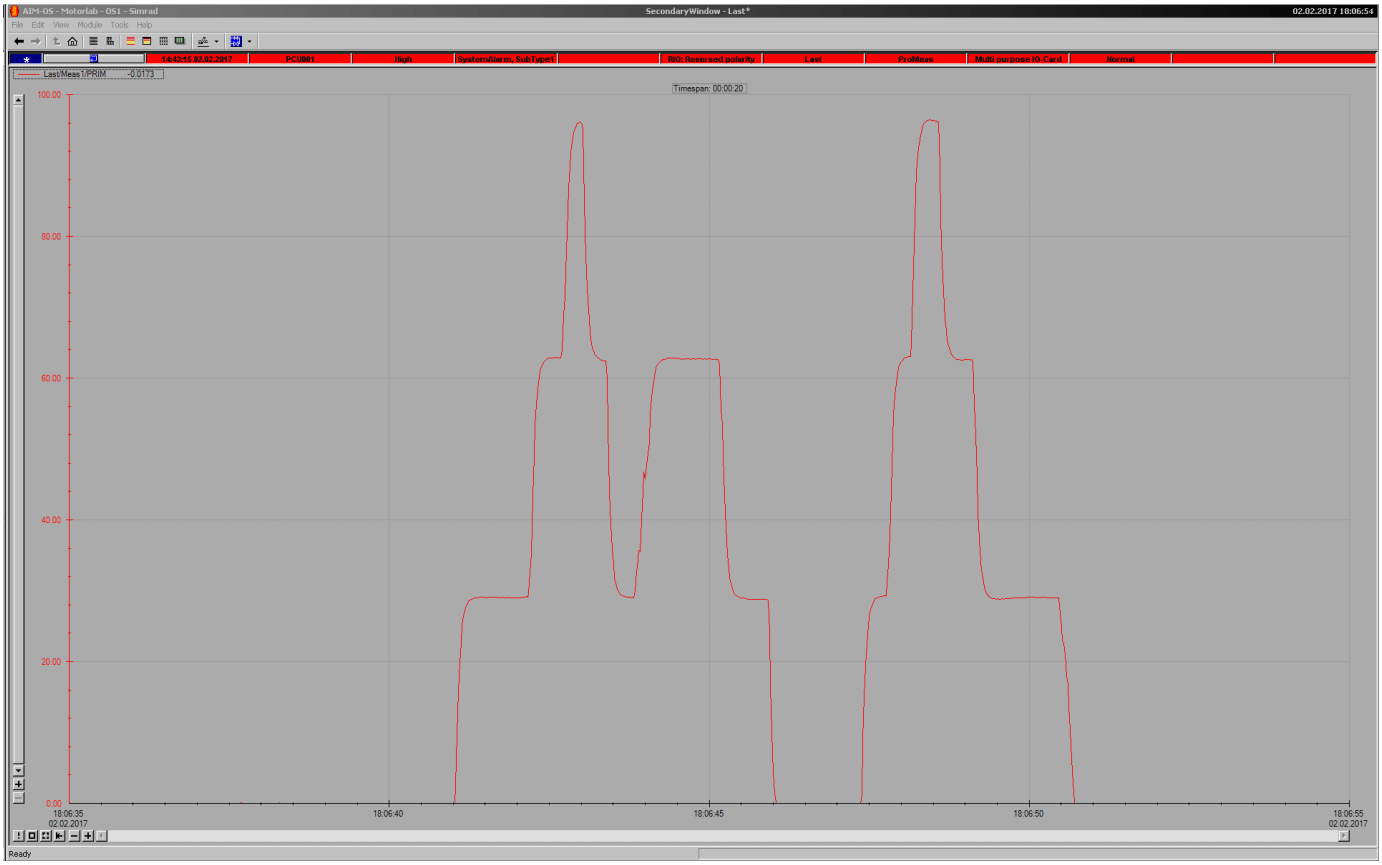

Figure 4: Trend Image

Alarms:

We created the alarms shown below. One for the load exceeding 100 (low priority, yellow), one for the switch not being in Auto (low priority, yellow) and one for the motor protection (high priority, red).

Figure 5: Load Alarm

Figure 6: Not in Auto Alarm

Figure 7: Alarm for Motor Protection

3 Part 3: Evaluation

a) Evaluation of the assignment text:

The theory in the assignment text was good and precise, but some of the references to the documentation found on the lab was poor due to missing pages, messy documentation, missing page numbering etc.

We found it much more useful to actively use the "Additional Information" inside the AIM system.

The assignment text could be more "modularized" with more clearity in part-tasks.

b) Evalution of the assignment:

We have learnt the AIM basics, and how terminals has to be connected in order to make a module function in a desirable way.

This lab could have included more information and tasks regarding the logical functions in AIM. Is it possible to perphaps do some of the tasks in Structured Text (ST), if there is any way of cleaning up/hiding the lines in the overview etc.

We used approximately 14 hours, due to the first 8 hours we did not accomplish the external functions. We then decided to start all over again the week after, and then finished the lab in 6 hours.