Assignments for PLC classes

Lecture 05 (Homework for Lecture 06)

Assignment 1 – Alternating light

Create a new task.

Declare a USINT variable in the local scope.

Increment the variable with 1 every cycle in the cyclic part of the program.

Observe how the variable is incrementing in the Watch.

Try this in different task classes and observe the difference.

Assignment 2 – Bit shifting

Create a small program, where you declare:

- One local variable of type BOOL called "shift".
- One local array of BOOLs with 16 elements (type is BOOL[0..15]).
- One local variable of type USINT called "i".

Set index 15 of the bool array to "TRUE" in the init-part of the program.

Write assignments, such that every bit is shifted to the left. The bits should be rotated, such that the MSB should become the LSB. For each cycle of the program, the shifting should be applied.

Adjust the cycle time to 500ms.

The variables “i” and “shift” is not needed for this exercise.

Lecture 06 (Homework for Lecture 07)

Assignment 1

Enhance Lecture 05, Assignment 2 to use a FOR loop instead of 15 single statements.

Allow the “user” (through the monitor) to enable/disable shifting using the boolean variable “shift”.

Assignment 2

Make a new program, that performs the exact same as Assignment 1. However, the program should use a timer to adjust the time between each rotation of bits. Your program should probably be in a cyclic with a cycle time less than 100ms.

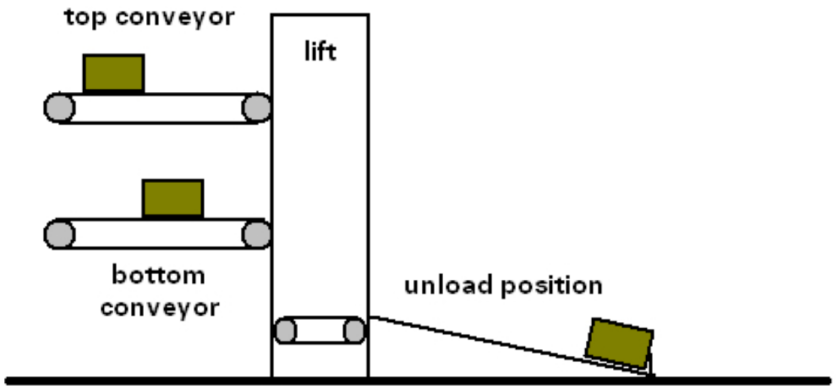
Lecture 07 (Homework for Lecture 08)

Assignment 1

Create a program that detects a falling edge (you are not allowed to use F_TRIG) and sets an output variable to TRUE when the falling edge is detected.

Assignment 2

Create boolean expressions to control doConvTop, doConvBottom, doLiftTop, doLiftBottom and doLiftUnLoad (NOT using IF-statements, only boolean expressions)



Two conveyor belts (doConvTop, doConvBottom) are being used to transport boxes to a lift. If the photocell (diConvTop or diConvBottom) is activated, the corresponding conveyor belt is stopped and the lift is requested. If the lift is not being requested, it returns to the appropriate position (doLiftTop, doLiftBottom).

When the lift is in the requested position (diLiftTop, diLiftBottom), the lift conveyor belt (doConvLift) is turned on until the box is completely on the lift (diBoxLift).

The lift then moves to the unloading position (doLiftUnload). When it reaches the position (diLiftUnload), the box is moved onto the unloading belt.

As soon as the box has left the lift, the lift is free for the next request.

Lecture 08 (Homework for Lecture 09)

Assignment 1

Implement the CarAlarm with “flag” (see example in notes for lecture 08). Use simulation to verify a working car alarm.

Assignment 2
Construct a state diagram for a single traffic light. Implement the state diagram using structured text. There should be one "input" to the traffic light that causes the light to go from green to red and from red to green (including the yellow light). Use simulation to verify that the traffic light can change.

Assignment 3
A greenhouse is equipped with electric blinds that needs to be controlled automatically. There is no position encoder on the blinds, and it is only possible to detect whether the curtain is opening or closing.

There are physical end stops that will stop the curtain. However, the motors do not automatically turn of when the curtain hits the end stop and continues to pull the curtain causing a malfunction if not stopped by the system. When a curtain reaches the end, the program must stop the motor. When the curtain stops moving the signal that detects whether the curtain is closing or opening will become false.

Two inputs are used to signal start opening or start closing.

Design a state machine for a single curtain and implement this in structured text.

Lecture 09 (Homework for Lecture 10)

Assignment 1
Use the TCP server from the example

- Send a long string
- Send a short string
- Observe what happens

Reimplement the TCP server from the example, to improve the observed error.

YAT (Yet Another Terminal, https://sourceforge.net/projects/y-a-terminal/)

Assignment 2
Extend the TCP server from exercise 1

- Create two command arrays data1 and data2 with length equal to the receiveData-array length
- After transmission copy the received data to the command array that is not in use
    - However, do only copy when process is ready.
    - Use bool variables to identify whether process is ready to receive new data
    - Use a bool or int to identify which buffer is ready to receive data

Lecture 10 (Homework for Lecture 11)

Assignment 1

Extend the OPC-UA program with an extra OPC-UA method called ”Add”. Use two input variables and one output variable.
UaExpert (https://www.unified-automation.com/downloads/opc-ua-clients.html)

Lecture 11 (Homework for Lecture 12)

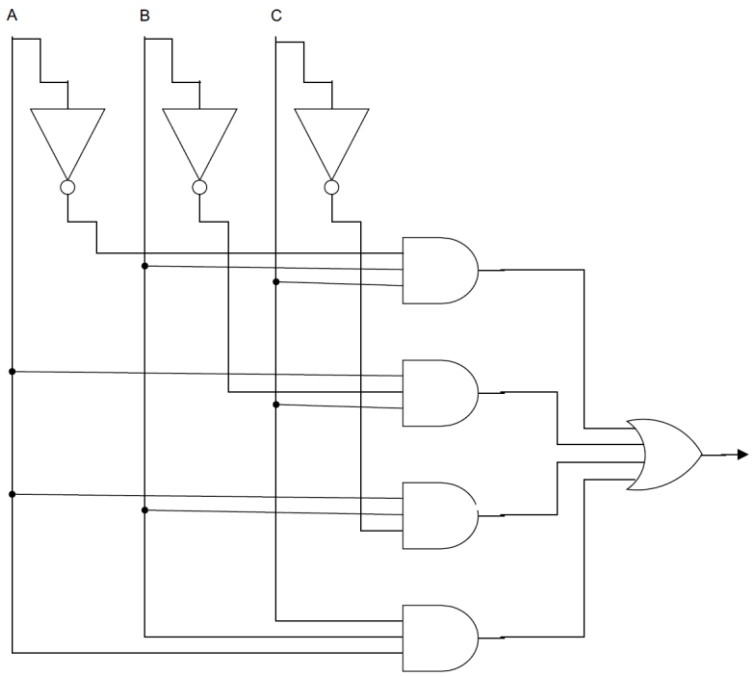
Assignment 1

Implement a function in an Automation Studio project using structured text. The program should use the Pythagorean theorem to calculate the length of the hypothenuse in a right-angled triangle.

Use an appropriate number of inputs and argue that you have used the correct variable types for input and output.

Assignment 2

Write a function that calculates the output described in the following diagram:



Assignment 3

Write a function block R_EDGETRIG that registers a rising edge on an input-signal. Name the output Q. You are not allowed to use the function block R_TRIG.

Assignment 4

Write a function block RF_EDGETRIG that registers a rising or falling edge on an input-signal. Name the output Q. You are not allowed to use the function blocks R_TRIG, F_TRIG or RF_TRIG.

Assignment 5

Create a function block that can handle the five states: STATE_STARTING, STATE_RESETTING, STATE_EXECUTE, STATE_COMPLETE, STATE_STOP.

Use an enumeration to implement the state variable.

The function block should have the following inputs: EN (enable) and next state (NS).

The function block should have the following outputs: Q (current state) and ERR (error output).

The following state changes are allowed:

- From all states to STATE_STOP
- From STATE_STARTING to STATE_EXECUTE
- From STATE_EXECUTE to STATE_COMPLETE
- From STATE_COMPLETE to STATE_RESETTING
- From STATE_RESETTING to STATE_STARTING
- From STATE_STOP to STATE_STARTING

All other state changes are not allowed. If other state changes are tried executed an error output must be given.

Assignment 6

Implement a function generator that can generate a sine wave, a triangle and a sawtooth signal.