Exercise 4

Contents

Introduction

In this exercise you will program an electrical car which is driving 10 km to a destination but is slowed down by stoplights. To achieve this behaviour a state machine is used.

Exercises

1. Connect PC and PLC

To establish a connection between your PC and the PLC work through the following topics of the "TwinCAT Guide":

1. Power the PLC
2. Establish connection
3. Open System Manager
4. Select Target System

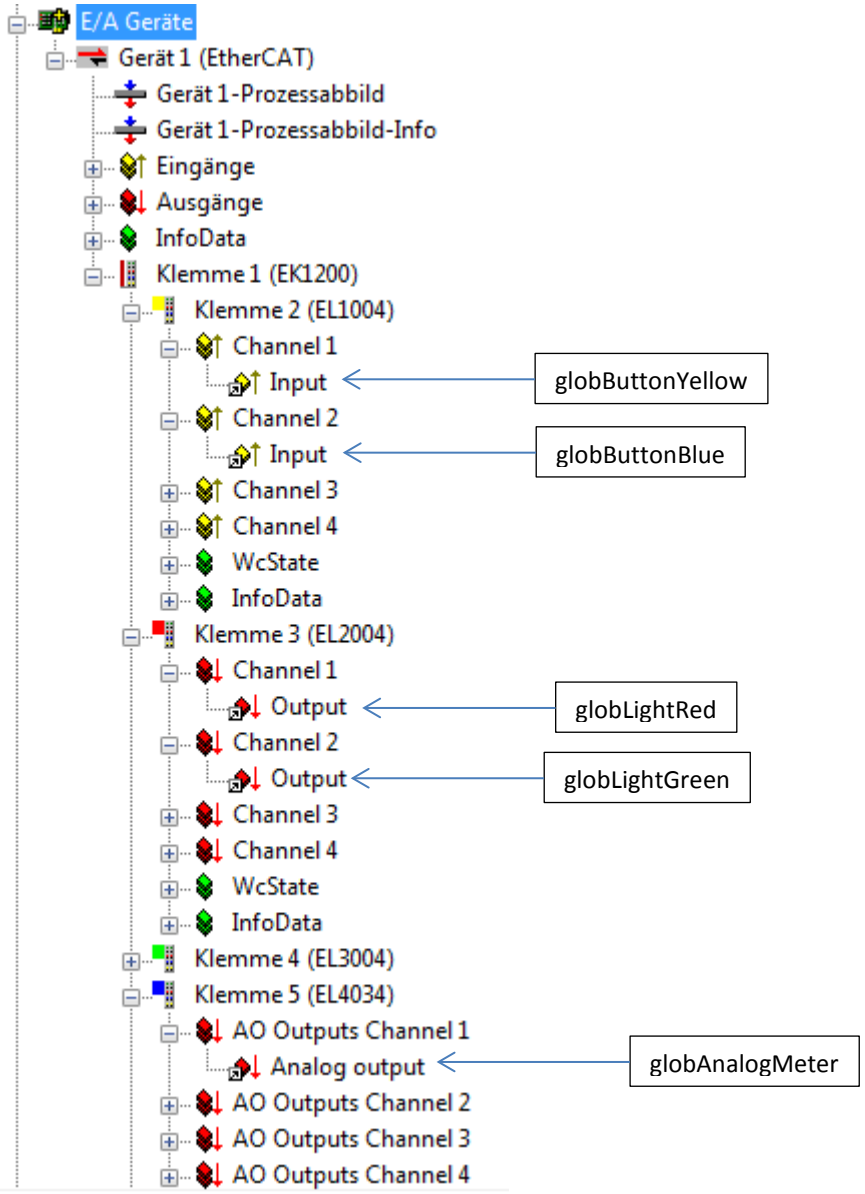
2. Download and build project

1. Download and save the exercise template project called Ex4_template.pro
2. Open the template project in "PLC Control".
3. Have a look at the global variables in the ressources tab of the project. These variables will be connected to the hardware.
4. Build the project.

3. Configure hardware

1. In "System Manager" change to "Config Mode"
2. Work through chapter "Automatic Terminal Configuration" in the "TwinCAT Guide".
3. Follow the instructions in the "TwinCAT Guide" to assign the variables to hardware:
    a. Append a PLC Project

b. Link SW variables and terminal IOs according to the description below



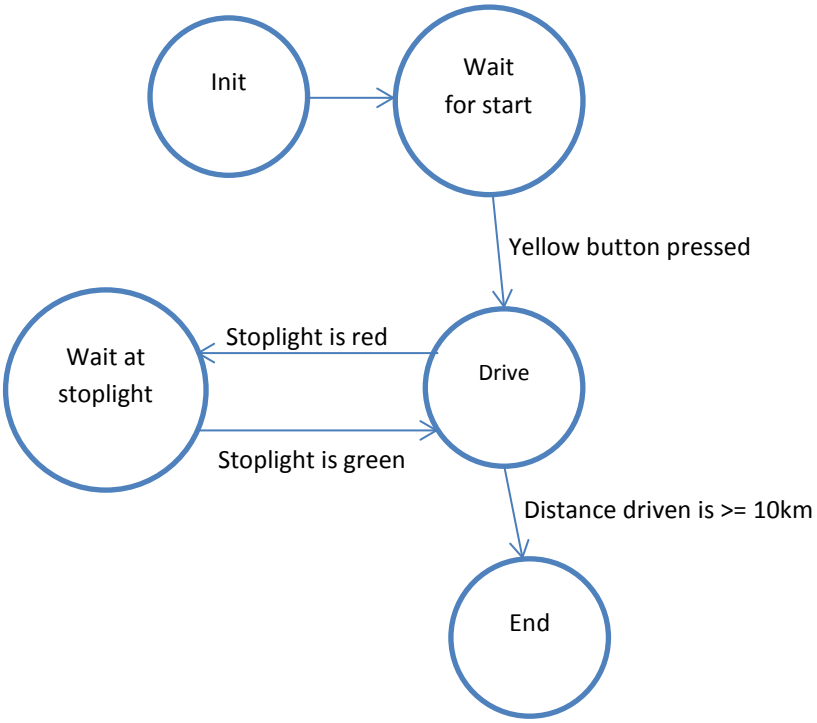
c. Activate configuration

4. In "PLC Control" run the project and with the global variables test if you can read the buttons, control the lights and the analog meter.

5. Electric car simulation

We simulate an electrical car driving 10km. It passes stoplights and needs to wait at them. The simulation is controlled by the previously configured 2 hardware buttons. The distance driven is displayed by the analog meter. During the simulation the lights show the state of the stoplight.

The state machine looks as follows:



Some of the work in "PLC Control" has already been done for you:
An enum for the car state exists where you add some more states later. In global variables there are 2 edge detectors globEdgeDetectorYellow and globEdgeDetectorBlue for the buttons. The state of the car is available over the global variable globCurrentCarState. The state of the stoplight (red or green) is available over the global variable globCurrentStopLightState.

1. Open the exercise template project in "PLC Control".
2. Add the missing states in the enum CarState (in tab Data types):
    a. CarStateWaitForStart
    b. CarStateDrive
    c. CarStateWaitAtStopLight
    d. CarStateEnd
3. Create a program for each state. This will make your code more readable and easier to understand. DoCarStateInit is already there, add:
    a. DoCarStateWaitForStart
    b. DoCarStateDrive
    c. DoCarStateWaitAtStopLight
    d. DoCarStateEnd
4. The program CarStateMachine needs to be completed. It executes the different state programs according to current car state which is available over the global variable globCurrentCarState.
    a. In the CASE statement inside the CarStateMachine add the missing states and their respective program calls. See the existing CarStateInit as an example.
5. Now you are ready to implement the actions and transitions within each state:
    a. In DoCarStateInit the initialization of the electric car happens. Set ...

   i. globDistanceDriven to 0
   ii. globAnalogMeter to 0
   iii. globCurrentCarState to CarStateWaitForStart (this will make the state machine switch to the next state)

b. In CarStateWaitForStart we wait for the user to start the simulation by pressing the yellow button.
   i. Set the green and red light both to TRUE (shows the user that simulation is ready)
   ii. Change to state CarStateDrive on the falling edge of the yellow button. The falling edge information is available in the global variable globEdgeDetectorYellow.fallingEdge

c. In CarStateDrive the car is driving. We use the red and green light to show the state of the stop light.
   i. Set the red and green light (global variables globLightRed and globLightGreen) according to the state of the stoplight (global variable globCurrentStopLightState which has 2 possible states StopLightStateRed and StopLightStateGreen).
   ii. Check if the car has already driven the 10km to reach it's destination. Use the global variable globDistanceDriven for this. If globDistanceDriven is >= 10km change the car state (globCurrentCarState) to CarStateEnd because the simulation has finished.
   iii. Check if the car is at a red stoplight. Use global variable globCurrentStopLightState for this. Change the car state to CarStateWaitAtStopLight if the stop light is red. If the stoplight is green increase the value of globDistanceDriven by 0.002 because the car is driving.
   iv. Display the distance driven on the analog meter (global variable globAnalogMeter). Here it is necessary to scale and convert the value of globDistanceDriven to INT because the terminal to which the analog meter is attached to only understands this datatype.
      1. Create a local variable tempAnalogMeterValue of type REAL. It is used to hold the conversion result temporarily.
      2. Set the temporary variable tempAnalogMeterValue to globDistanceDriven / 10 * 32767. This is the before mentioned scaling.
      3. Check if the value of tempAnalogMeterValue is > 32767. If yes, set it to 32767 because this is the biggest INT value possible.
      4. Now set the global variable globAnalogMeter to the result of REAL_TO_INT(tempAnalogMeterValue). This is the before mentioned conversion.

d. InCarStateWaitAtStopLight the car is waiting at a red stoplight. Once the stoplight is green change the car state to CarStateDrive.

e. InCarStateEnd the car has reached it's destination.
   i. Show it to the user by toggeling the red and green light at the same time.
   ii. If the user presses the yellow button (global variable globEdgeDetectorYellow.fallingEdge) change the car state to CarStateInit.

6. Display battery level (for fast students)

Extend the previous project so the user can show the battery level of the electric car instead of the distance driven. You are free in the way to achieve the following requirements:

- Introduce a new global variable for the current battery level.
- Initialize it to a 100% in the appropriate state.
- Decrease battery level by 0.004% when driving.
- Show battery state on analog meter while user presses the blue button. 100% equals 10 on the analog meter.