Exercise 3

Contents

Introduction

In this exercise you will learn about working with some of the hardware connected to the PLC. The other big topics are state machines and arrays.

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. Create a project

1. In "PLC Control" create a new project.
2. In the "Ressources" tab you will find the global variables (variables which can be accessed from every place in the project). Create the following variables here:
   a. globButtonYellow of type BOOL
   b. globButtonBlue of type BOOL
   c. globLightGreen of type BOOL
   d. globLightRed of type BOOL

   To assign these variables to hardware in the "System Manager" they need to have the following signature for inputs:

   ```
   globButtonYellow AT %I* : BOOL;
   ```

   And for outputs:

   ```
   globLightGreen AT %Q* : BOOL;
   ```

3. Build the project.

3. Configure hardware

1. Change to "Config Mode"
2. Work through chapter "Automatic Terminal Configuration" in the "TwinCAT Guide".
3. Press the yellow and blue buttons of the "Control Box" and notice the LEDs lighting up on the second PLC terminal.
4. Work through chapter "Read inputs and write outputs" in the "TwinCAT Guide". Try to control the 2 lights on the "Control Box".
5. 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
   c. Activate configuration
6. Now if everything works correctly you are able to read the buttons and control the lights from within your program by using the 4 global variables.

4. Light toggeling

We are creating a small application that toggles the 2 lights.

1. In your program create the following variables:
    a. toggle of type BOOL.
    b. lightCounter of type INT.
2. Count up lightCounter and toggle the lights whenever lightCounter reaches 200.

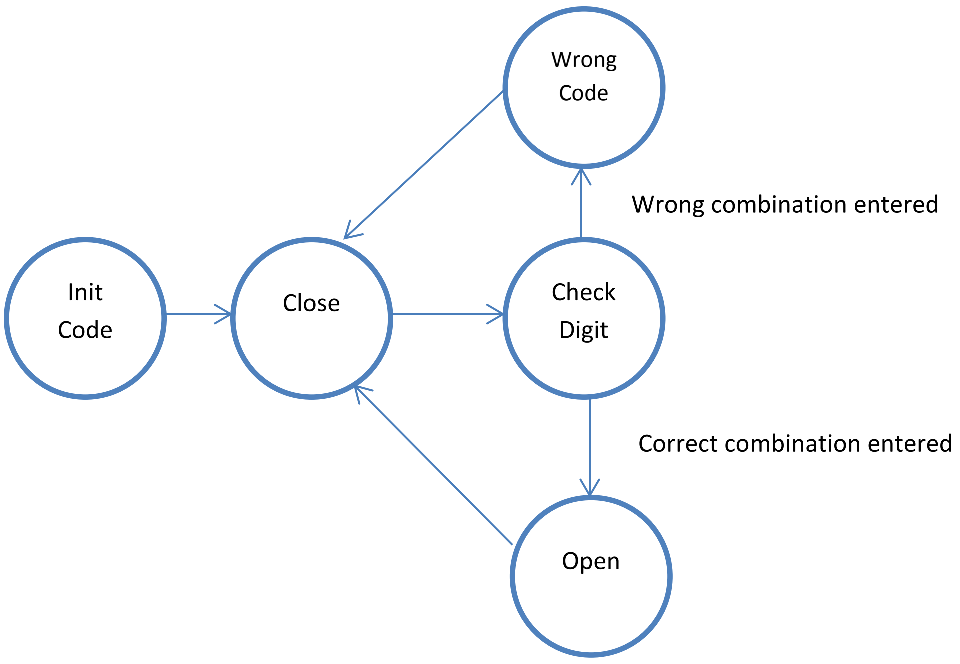
5. Safe

This application will be a safe which can be unlocked by pressing the right combination of the 2 buttons.

1. Create a Function Block called EdgeDetector. It will detect the rising and falling edges when pressing a button.
    a. Input:
        i. buttonValue of type BOOL
    b. Outputs:
        i. risingEdge of type BOOL
        ii. fallingEdge of type BOOL

    Detect the edges by comparing the current value of a button to it's previous one.
2. Create the ENUM "SafeState" with the following entries:
    a. SafeStateInitCode
    b. SafeStateInit
    c. SafeStateCheckDigit
    d. SafeStateWrongCode
    e. SafeStateOpen

    This will be used for the following state machine (http://en.wikipedia.org/wiki/Finite-state_machine) which will help to implement the safe's logic:



3. In MAIN create the following varibles:
    a. an instance of the EdgeDetector for the yellow button and one for the blue button.
    b. currentSafeState of type SafeState. The variable that holds the state of the state machine.
    c. Array codeArray of type BOOL with 4 entries. Read about arrays in the "ST Reference".
    d. currentIndex of type INT. Index used to navigate the codeArray.
    e. correctCounter of type INT. Counts how many digits of the code have been entered correctly.
    f. lightCounter of type INT. Used to turn on the red and green light.
4. With a CASE statement we handle states of the state machine:

```
CASE currentSafeState OF
      SafeStateInitCode:
            ...

      SafeStateClose:
            ...

      SafeStateCheckDigit:
            ...

      SafeStateWrongCode:
            ...

      SafeStateOpen:
            ...
END_CASE
```

5. In SafeStateInitCode fill the code array with the button combination. TRUE means yellow button, FALSE means blue button. For example TRUE, FALSE, FALSE, TRUE means the user has to press yellow, blue, blue, yellow to open the safe. Go to SafeStateClose after.
6. In SafeStateClose reset the counters and the lights. Go to SafeStateCheckDigit after.
7. In SafeStateCheckDigit only do something if a rising edge of button yellow or blue is detected. Check if the pressed button is the next in the code combination. If so increment the correctCounter. Keep on doing this until 4 buttons have been pressed. If the code was entered correctly go to SafeStateOpen else go to SafeStateWrongCode.
8. In SafeStateWrongCode turn on the red light for 5 seconds and go to SafeStateClose after.
9. In SafeStateOpen turn on the green light for 5 seconds and go to SafeStateClose after.

6. Safe with programmable lock

Extend the previous application so one can program a different code. For this the safe needs to be open and both the yellow and blue button need to be pressed together. Afterwards a new combination of length 4 can be entered.