Exercise 6

Contents

Introduction

In this exercise you will create a game with the limited input and output possibilities the plc rig offers.

Exercises

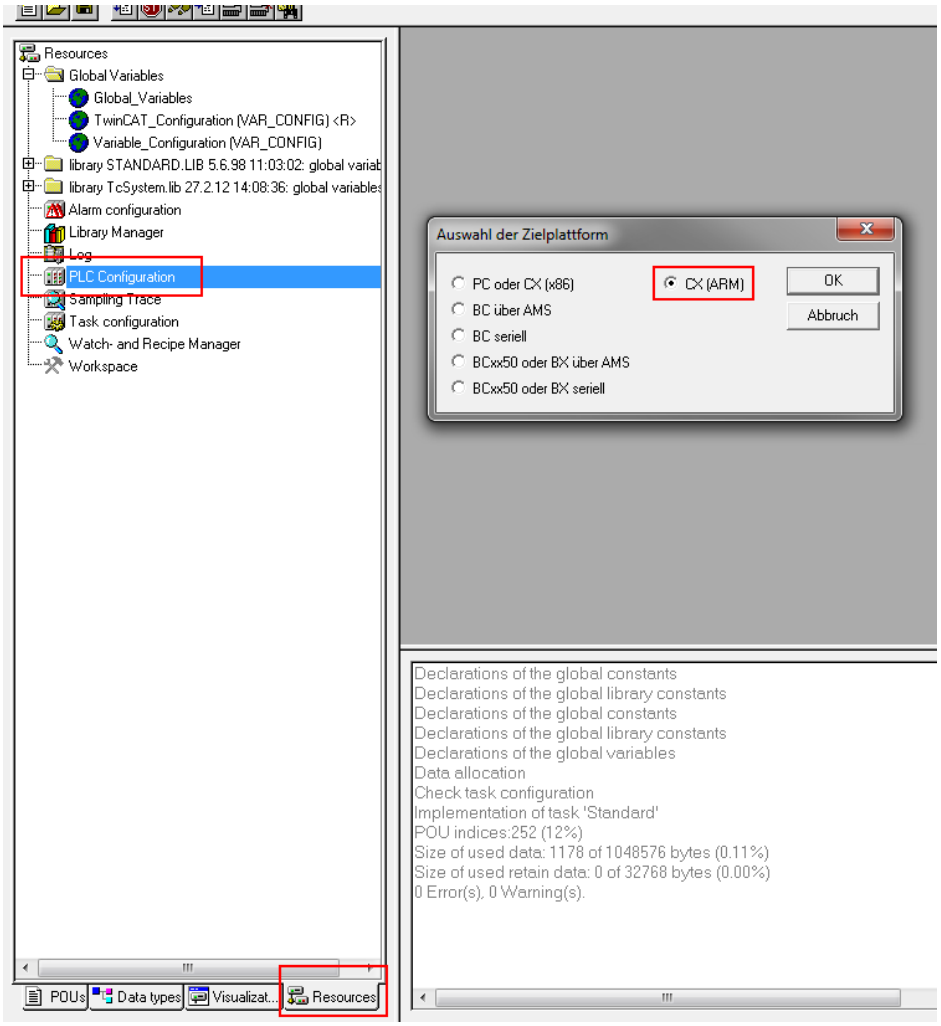
1. Connect PC and PLC

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

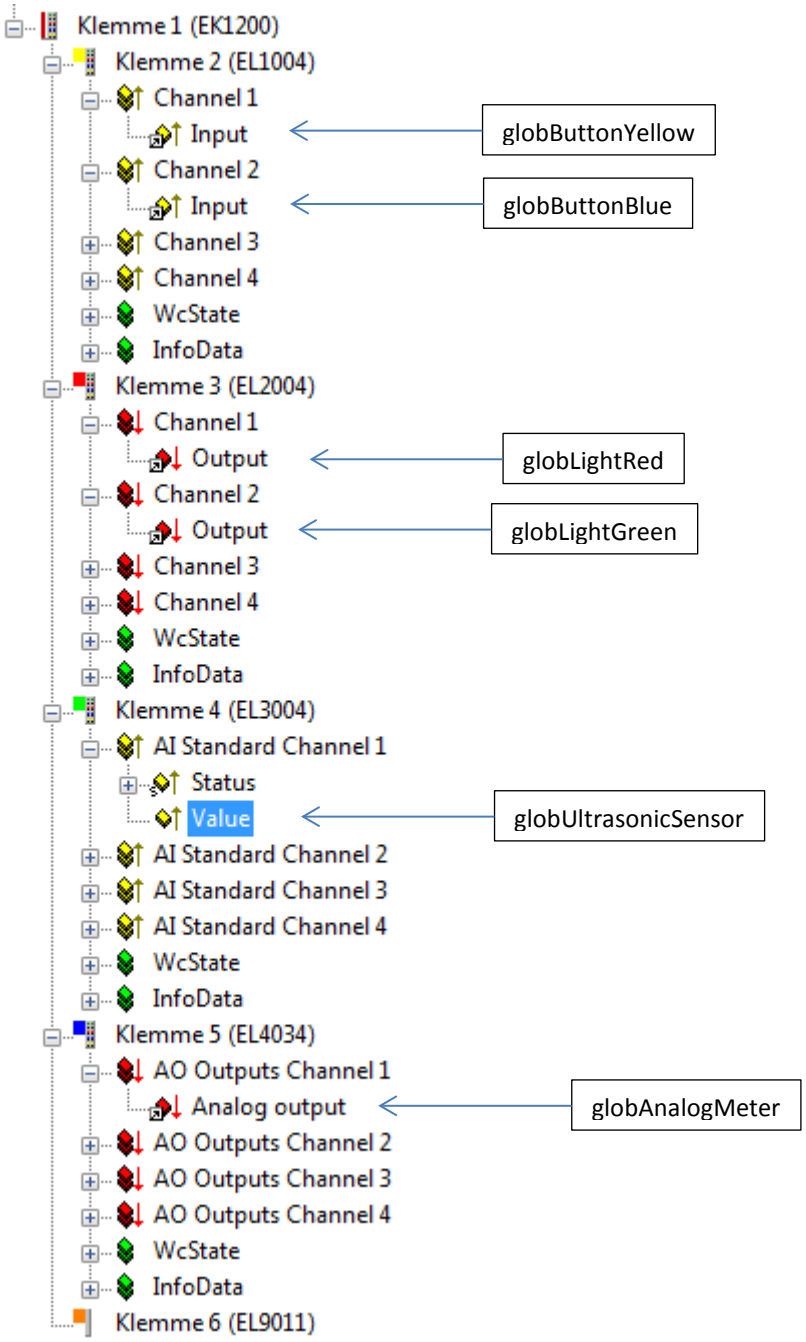
1. Download and save the exercise template project called Ex6_template.pro
2. Open the template project in "PLC Control".
3. Have a look at the global variables in the ressources tab of the project. These variables will be connected to the hardware.
4. If you have an ARM PLC you need to change the PLC Configuration to CX (ARM)



5. 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



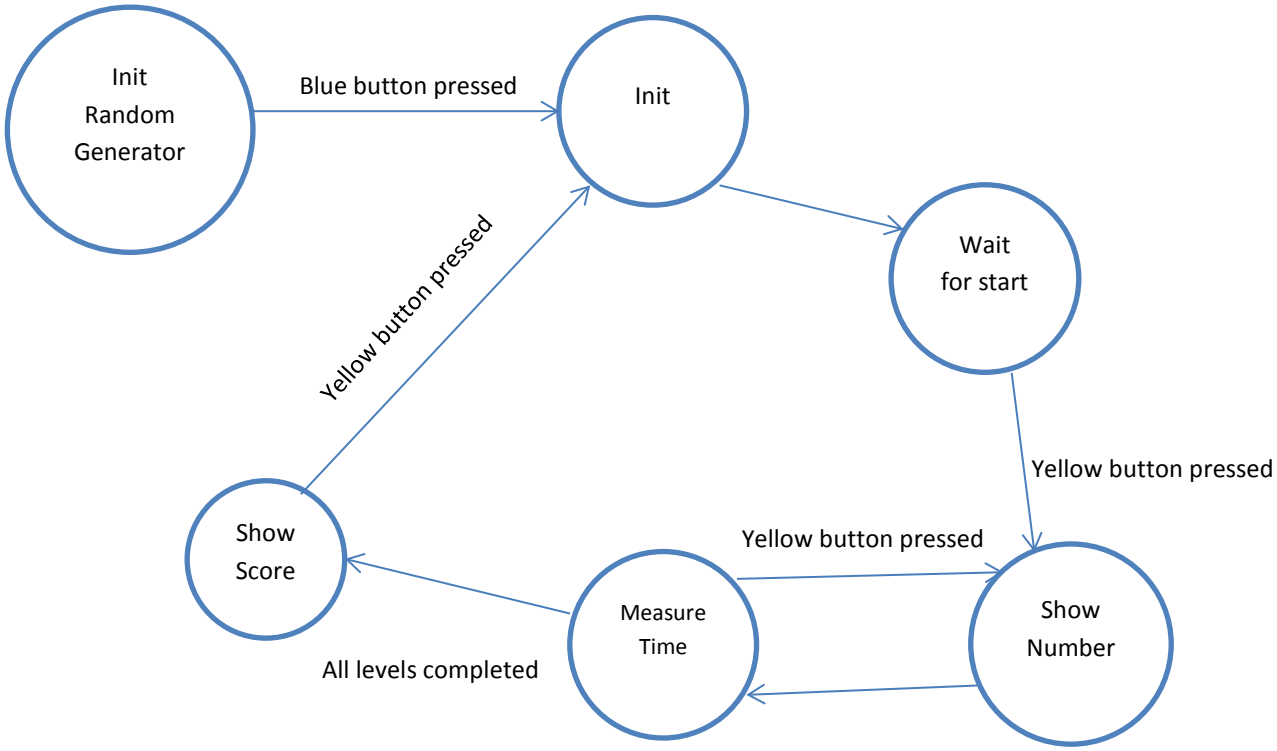
    c. Activate configuration
4. In "PLC Control" run the project and with the global variables test if you can read the buttons, the ultrasonic sensor, control the lights and the analog meter.

5. Game: Aim for the number

Using 2 buttons, 2 lights, 1 analog meter and 1 ultrasonic sensor you are going to develop simple game.
The gameplay works as follows:

1. The levels are initialized with random numbers
2. During the game the player has to count how many times the green light blinks (reference number)
3. The player uses the ultrasonic sensor to control the number shown on the analog meter
4. When the needle of the analog is close to the reference number the player needs to press the yellow button. Speed is important but precision is weighted double.
5. The lower the overall score the better.

The state machine looks as follows:



The places where you have to write some code are marked in the project template. The current game state is available over the global variable globCurrentGameState.

1. Open the exercise template project in "PLC Control".
2. Right click on the Data types directory and add an object called "Level". This is a STRUCT which will hold information about a game level, read about STRUCTs in the "ST Reference". In the STRUCT add:
    a. numberReference of type INT. The player tries to get as close as possible to this number.
    b. numberMeasured of type REAL. This is the number the player actually selected.
    c. timeMeasured of type REAL. This is the time the player took to complete the level.
3. For the game some more global variables are needed. Add global variables:
    a. ARRAY globLevels of type Level with indices from 1 to LEVELS_MAX_COUNT. This array holds all the levels of the game.
    b. globLevelsIndex of type INT. This index is used to navigate through all the levels.
    c. globCurrentLevel of type Level. A copy from the array of the current level.
4. To keep the game interesting a random number generator is used to initialize the levels. Else the player could memorize which number comes up next. This random number generator has to be initialized only once. Complete DoGameStateInitRandomGenerator:

   a. Go to state GameStateInit when the blue button was pressed (use globEdgeDetectorBlue.fallingEdge).
5. In DoGameStateInit the levels will be initialized:
   a. In the FOR loop initialize the levels of globLevels. Use the index variable to access the the array entries. See the ST Reference for how to access variables of a STRUCT.
   For each level set:
      i. timeMeasured to 0
      ii. numberMeasured to 0
      iii. numberReference to globRandomGenerator.number
   b. After the FOR loop:
      i. Set globLevelsIndex to 1
      ii. Set globCurrentLevel to the value in array globLevels at index globLevelsIndex
      iii. Go to state GameStateWaitForStart
6. In DoGameStateWaitForStart we wait for the player's start signal: When the yellow button was pressed (use globEdgeDetectorYellow.fallingEdge) go to state GameStateShowNumber.
7. In DoGameStateShowNumber: According to the number reference information in the current level the green light needs to blink as many times so the player knows which number he has to aim for. You can use globCurrentLevel here.
   a. Above the IF statement check if globCurrentLevel.numberReference is greater 0 and counterLight is 0. If this is TRUE set counterLight to INTERVAL (constant). Decrease globCurrentLevel.numberReference by 1.
   b. Below the IF statement check if globCurrentLevel.numberReference and counterLight equal 0. If so turn off the green light and go to state GameStateMeasureTime.
   c. At last check if counterLight is greater 0. If so decrease counterLight by 1.
8. In DoGameStateMeasureTime: The player has to use the ultrasonic sensor to reach the reference number as fast and precise as possible. We record the number she/he actually selects and how long it takes her/him. In the IF statement:
   a. Use globLevelsIndex to access the Level in globLevels and set the Level's number measured to globUltrasonicSensor / 3276.7. Set the Level's timeMeasured to stopWatch.
   b. Set stopWatch to 0
   c. Increase globLevelsIndex by 1
   d. If globLevelsIndex is smaller or equal LEVELS_COUNT_MAX the player has to finish more levels. Set globCurrentLevel to the level in globLevels at globLevelsIndex. Go to state GameStateShowNumber.
   e. If globLevelsIndex is larger LEVELS_COUNT_MAX the player has finished all levels. Go to state GameStateShowScore.
9. Before showing the score it needs to be calculated in CalculateScore. Pass a levels ARRAY named finishedLevels as an input. Use a FOR loop to navigate through all levels. Calculate the score according to the following formula:

$$Score = \sum_{i=1}^{LEVELS\_COUNT\_MAX} 2 * \left|num_{measured,i} - num_{reference,i}\right| + time_{measured,i}$$

10. In DoGameStateShowScore show the score the player has reached. In the IF statement:
   a. Pass globLevels into CalculateScore and store the result in variable score.
   b. Set tempDisplay to score / 100 * 32767.
   c. Limit temp display to 32767 if it is larger than that.
   d. Set globAnalogMeter to tempDisplay (use conversion).
11. Have fun playing!