

Sending data between Beckhoff PLC and Kuka Robot Controller(KRC4)

Espen Teigen

March 2019



Github: <https://github.com/EspenTeigen/Kuka-KR-C4-commissioning>

Contents

1	Setting up the PLC to send and receive data with controller	3
1.1	Opening a project and get to know the main PLC components .	3
1.1.1	KukaPLC Project	6
1.1.2	POUs	7
1.1.3	PLCTask	8
1.1.4	KukaPLC instance	9
1.2	Creating a new program	10
1.3	Connecting the program to the PLC-task	12
1.4	Linking of variables to bridge	14
1.5	Downloading the program to the PLC	16
2	Configuration of the robot controller to send and receive values to and from the PLC	17
2.1	Get Work Visual up and running and brief overview	17
2.2	Linking the inputs from the PLC	21
2.3	Linking the outputs to the PLC	26
2.4	Give the output and input usable names	30

1 Setting up the PLC to send and receive data with controller

1.1 Opening a project and get to know the main PLC components

First you have to open Twincat XAE. you will get this window

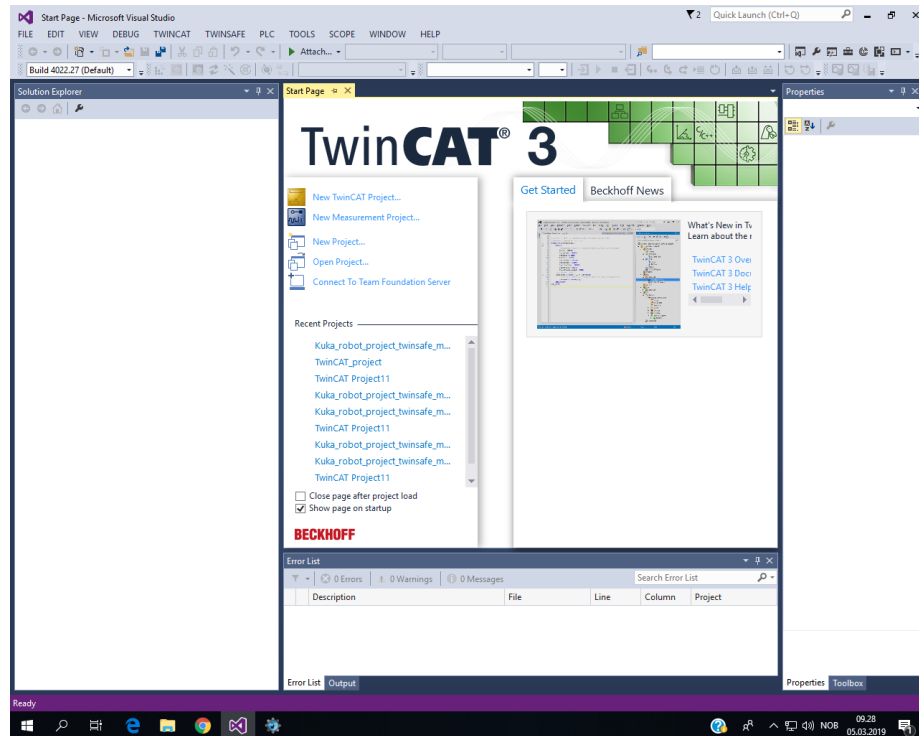


Figure 1: Twincat startscreen

Select open project and in the folder TwinCAT project, you will find the microsoft visual studio solution **Kuka_robot_project_twinsafe_modbus**. If the project is not found, you can get a copy from [github](#)

You will get this view

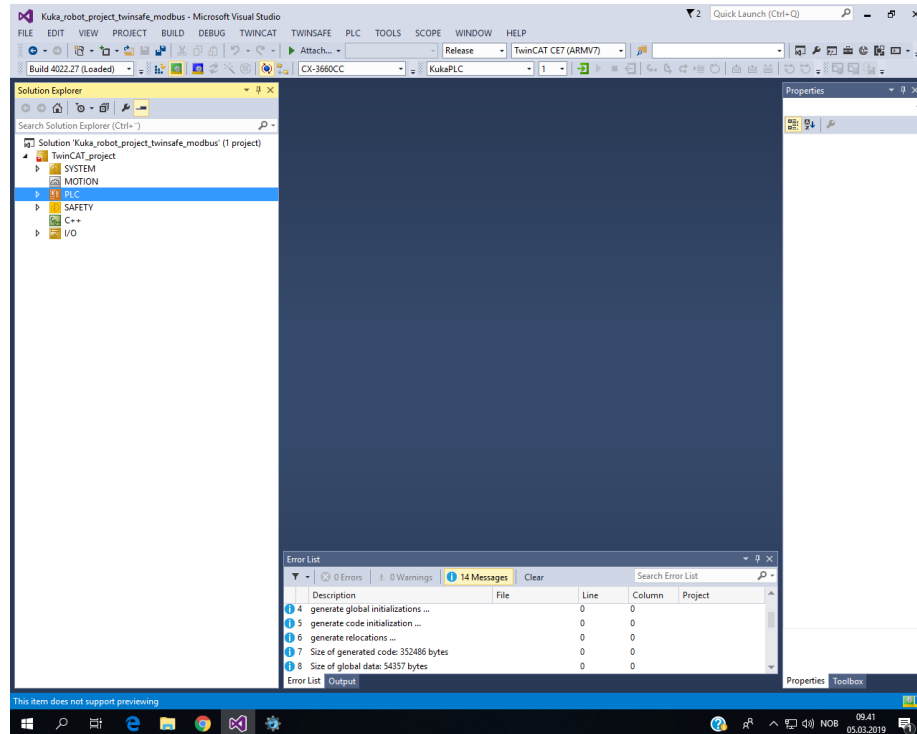


Figure 2: Project view

The most interesting part off the project tree is the PLC, so expand that part.

We can now take a look at the different parts that are relevant for data transmission.

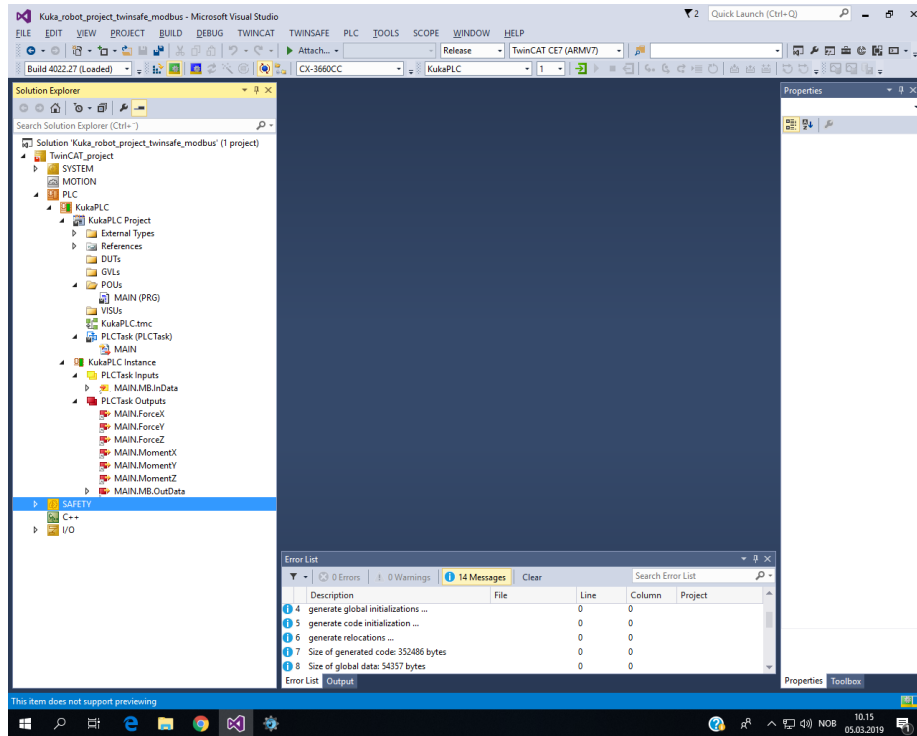


Figure 3: Caption

1.1.1 KukaPLC Project

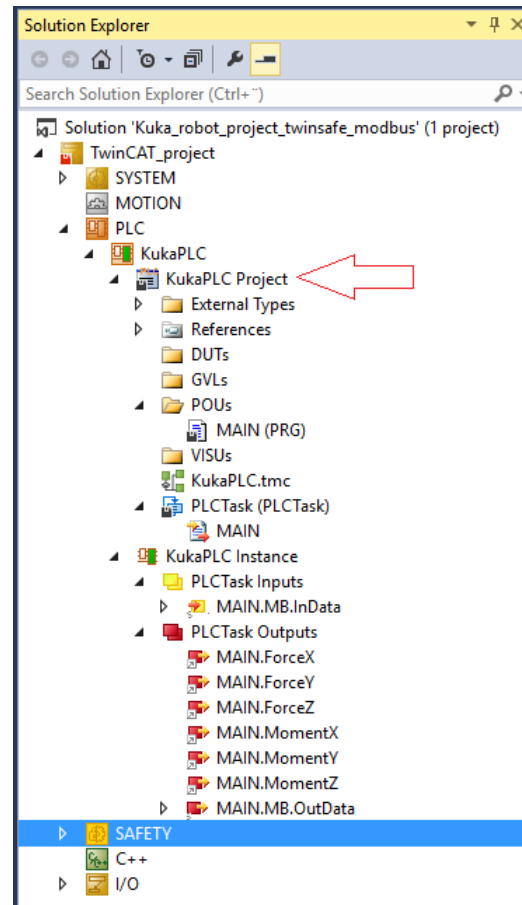


Figure 4: Caption

This is the entire PLC project, and are pre-configured to transfer data from the force-torque sensor to the robot controller. It is in this project we will set up our data transfer.

1.1.2 POU's

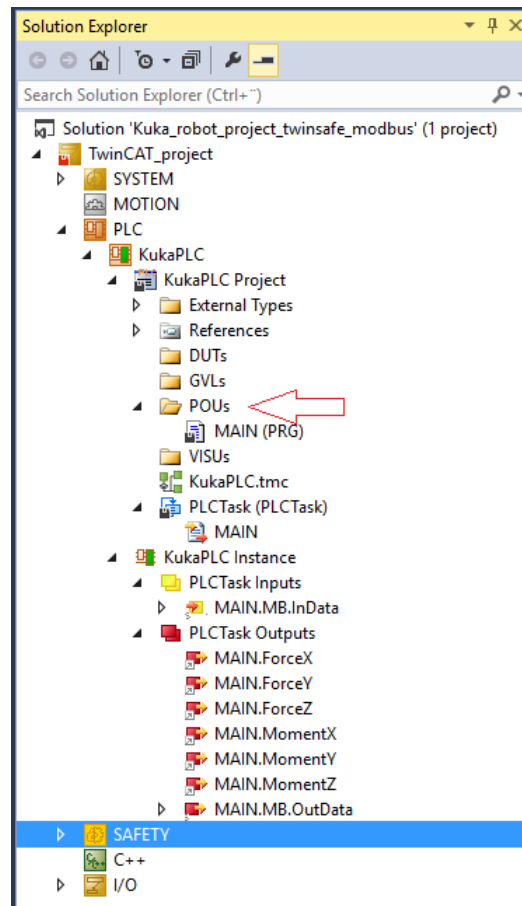


Figure 5:

POUs are where at the programs, function blocks and functions are placed. You can chose between all the IEC 61131-3 standard programming languages. I will be using structured text.

1.1.3 PLCTask

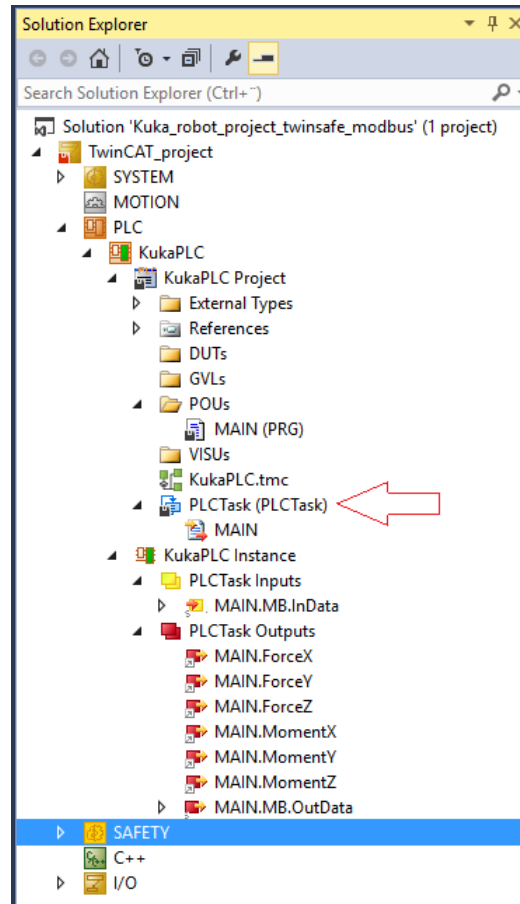


Figure 6:

To understand this, I have to explain what a task is. A task is controlling the timing of how often the program should be run. You can also configure the priority of how the program should be run. If the program block is of utmost importance, and must be completed within a certain time limit, you will give it a suitable priority.

PLCTask is a task that I made, and is used for this PLC. Every program that is going to be run in the PLC is timed by this task, and has to be added to it.

1.1.4 KukaPLC instance

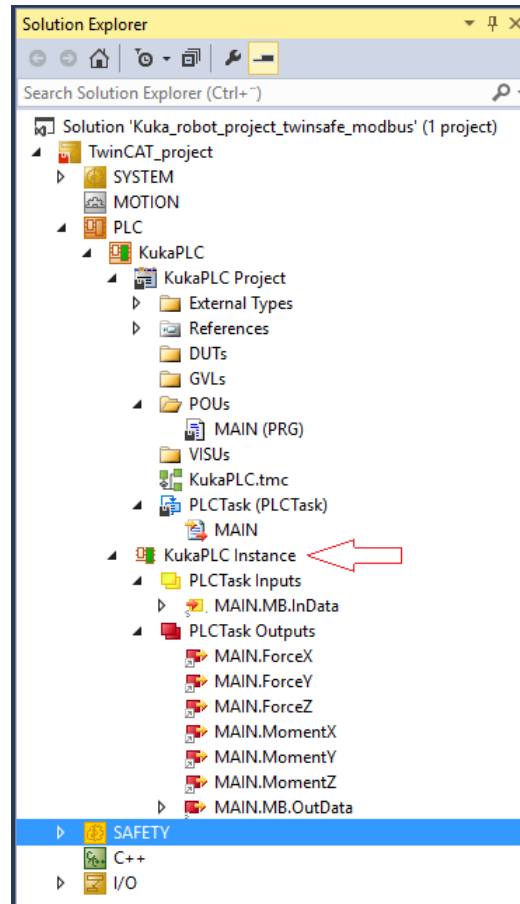


Figure 7:

This is where you tie you're inputs and outputs to be transferred to and from the robot controller. When you create inputs and outputs in the PLC-program, they will end up here, and you have to link them to the module that transfer the data. Do not fret, I will describe this in a better manner later in the document.

1.2 Creating a new program

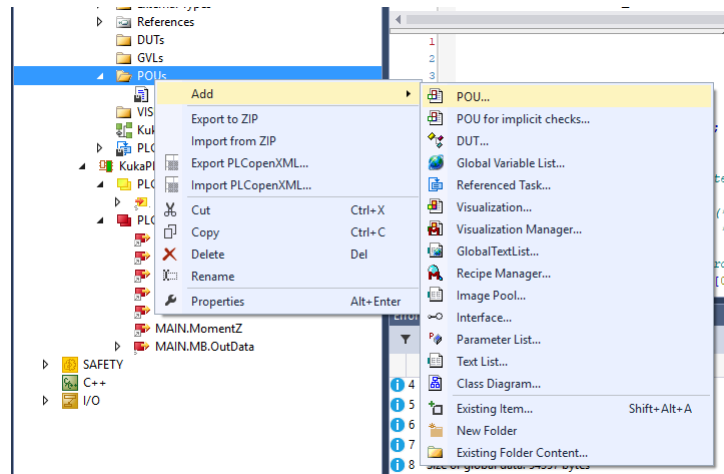


Figure 8: Start by right-clicking on the POU's-folder and select add-POU

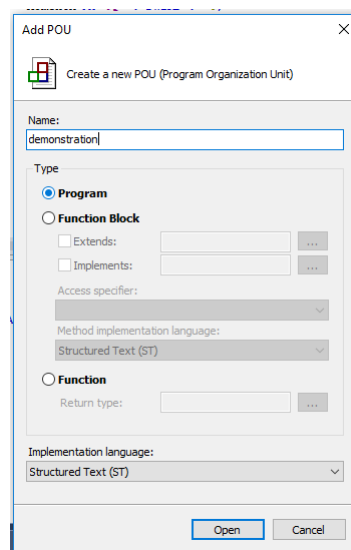


Figure 9: Give the program a name, and make sure you create a program, and that it is structured text

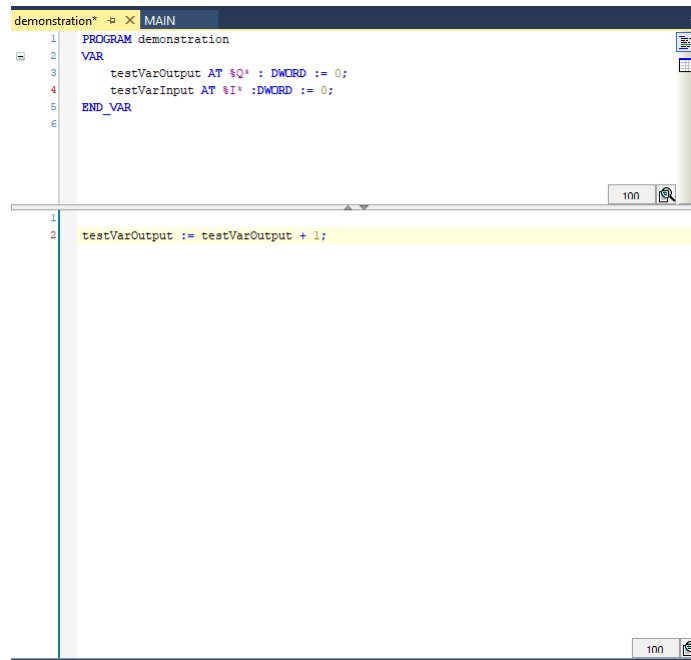


Figure 10: Now we are ready to write a program

PROGRAM demonstration is the name of the program

VAR and **END_VAR** denotes where you declare variables

testVarOutput **AT %Q*** : **DWORD** := 0;
testVarOutput is the variable name, **AT** tells the compiler that this should be attached to something. **%Q** tells the compiler that it should be an output, and the asterix * is telling the compiler that it can choose where in memory the data should be saved.

The reason we are using **DWORD** is because the ethercat bridge is configured to use it. The use of anything else will give a compiler error.

For the variable testVarInput it is the same, except we are using **I** to tell the compiler that it is an input.

Lastly I have just written a short code that will iterate through all the values until the variable overflows and start at zero again. After we have linked the output to the bridge, the value will be transferred to the robot controller every iteration.

1.3 Connecting the program to the PLC-task

Now we have to connect our program to the PLC-task, otherwise it will not be executed.

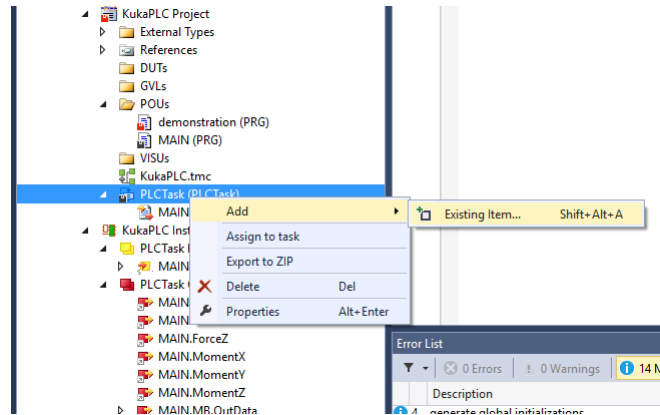


Figure 11: Adding the program to the PLC-task

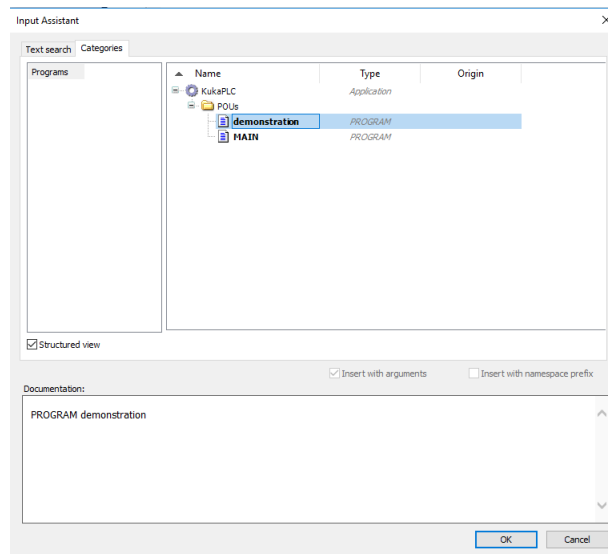


Figure 12: Select our program to be added to task and click OK

Ok, just a few more steps. To get our input and output added to the instances, we have to build the PLC-program.

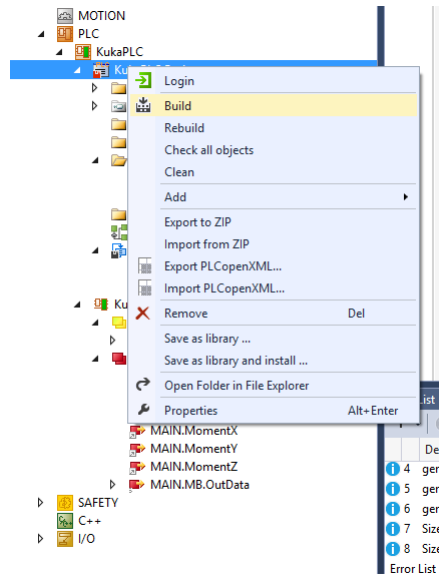


Figure 13: Right click on KukaPLC project and select build

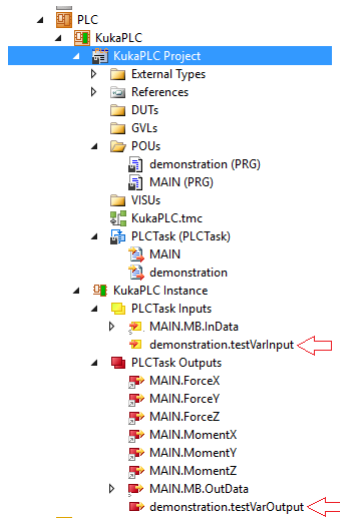


Figure 14: And now we can see the input and output we created in the KukaPLC instance

1.4 Linking of variables to bridge

And to get the data transferred, we have to link this instances to the ethercat-bridge inside the controller. Double-click the demonstration.testVarInput.

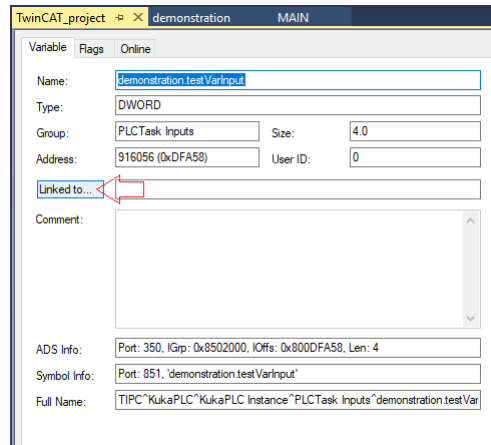


Figure 15: We start by linking the input. Click the link to button

A new window opens, this shows everything in the system you can link this variable to. If the variable type was BOOL, we could have linked this to a regular input or output. But we are linking this to the ethercat-bridge in the controller. This is the Box 9 (KRC4 secondary EL6695-1001).

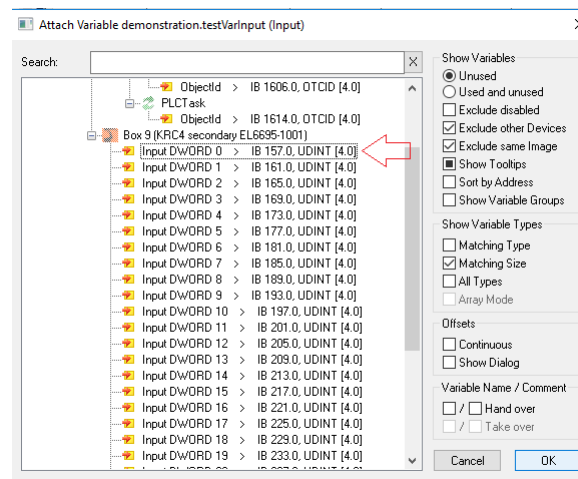


Figure 16: I am choosing the first DWORD, but you could use anyone

We have successfully(I hope) linked the variable to the ethercat-bridge, and after upload to the PLC we can pull the values from the controller from this variable.

Next step is to link the output variable to the ethercat-bridge so we can send values to the robot controller.

Start by double-clicking demonstration.TestVarOut located under PLC Task Outputs and click the Link to button(like you did when linking the input).

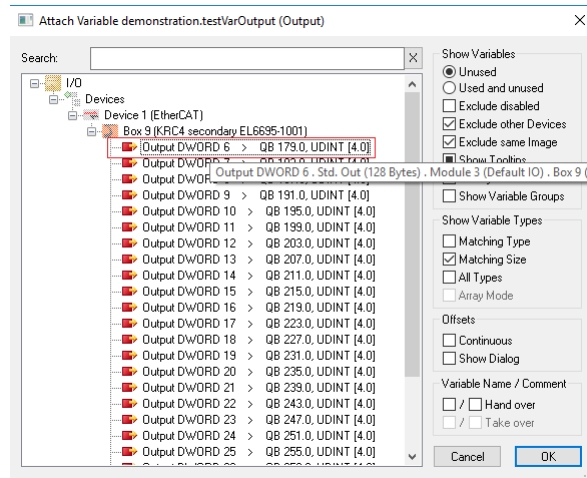


Figure 17: Since I already have used the six first outputs to the data transfer from the force-torque sensor, we have to select the 6 output.

Press OK, and we have also linked our output variable to be sent to the controller. The last step is to download the program to the PLC.

1.5 Downloading the program to the PLC

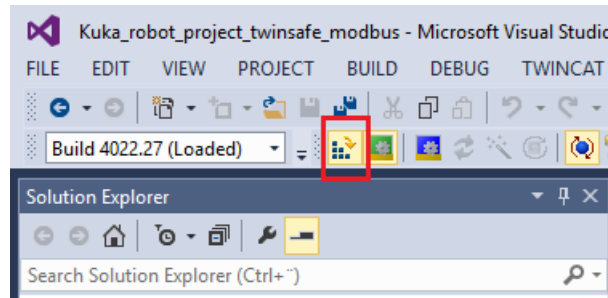


Figure 18: Up in the left area of TwinCAT, you can see this button. Push it, just do it, i dare you

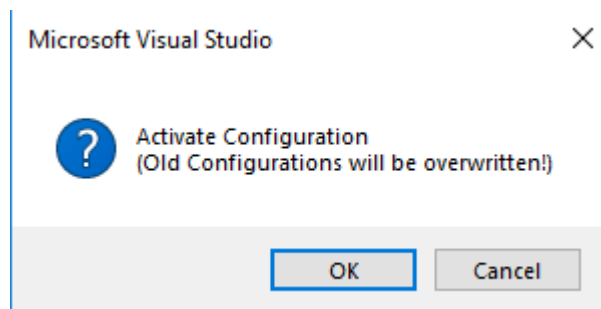


Figure 19: Press OK

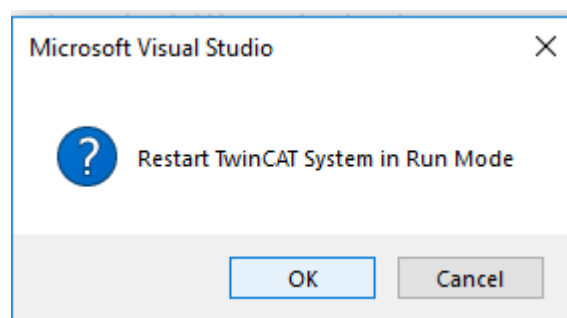


Figure 20: And press OK again

You have now downloaded the program to the PLC, the next step is to configure Work Visual so the controller can receive the values we are sending to it.

2 Configuration of the robot controller to send and receive values to and from the PLC

So. Lets open Work Visual, and get started with the configuration to send and receive data.

2.1 Get Work Visual up and running and brief overview

This is the first window Work Visual greet's you with.

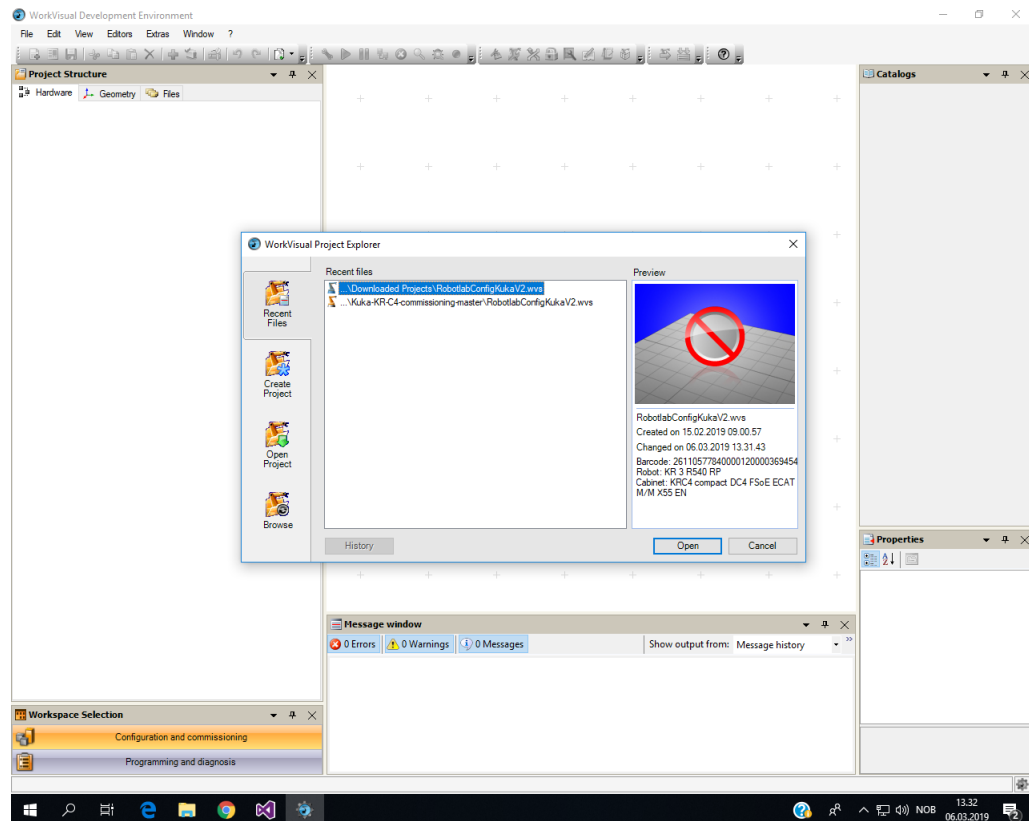


Figure 21: Start by clicking browse

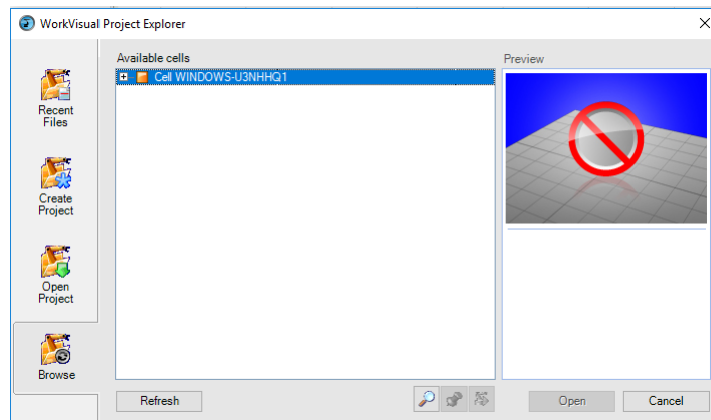


Figure 22: You are now looking at the files that are on the controller
Expand the tree and you will see this

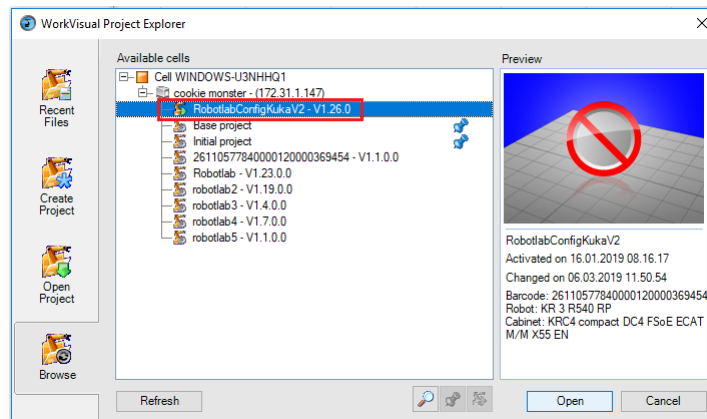


Figure 23: Notice that the file marked in red has a green arrow. This denotes that it is the active project

Click open

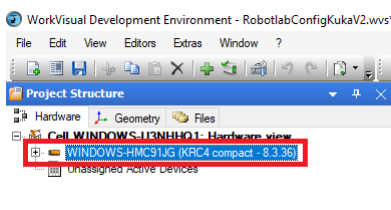


Figure 24: Double-click the project to make it active

On the left you will see the project tree for the robot configuration.

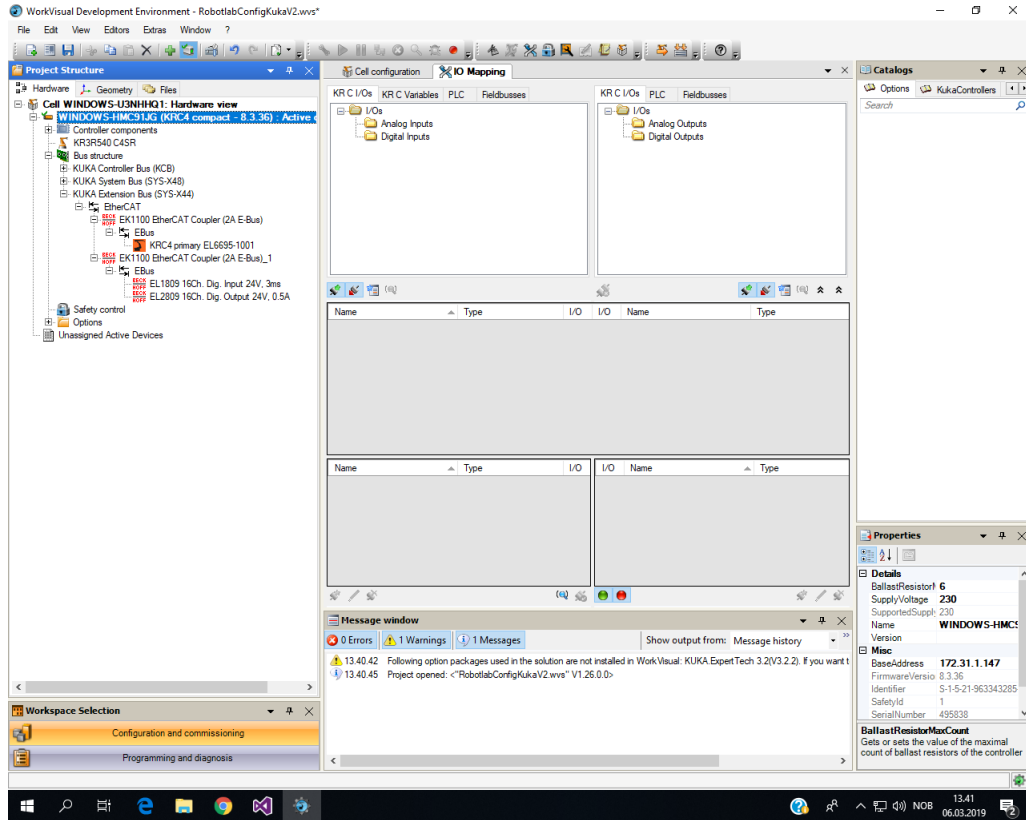


Figure 25: Expand the Kuka extension bus(SYS-X44)

Do not do any configuration with control or system bus if you don't know what you are doing: This will do major damage to the configuration file, and is thoroughly hard to unfuck.

Lets take a closer look at the project tree

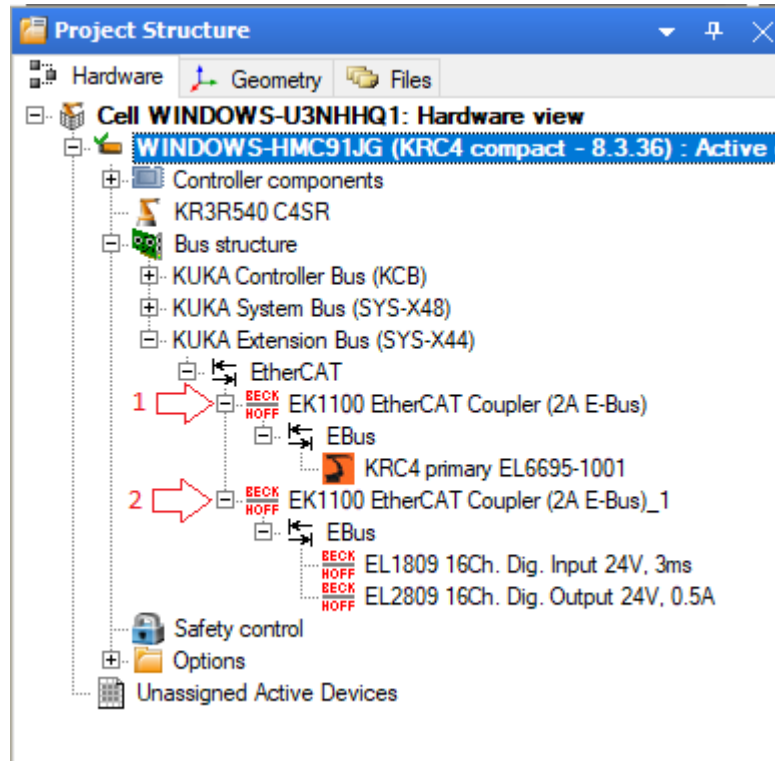


Figure 26: 1 is the bus our ethercat bridge is connected to, 2 is the bus we can connect our IO's and modules to

I do not think there should be any need in the near future to do any changes to the settings of the ethercat-bridge. I have configured it to handle 128 bytes of data on each transmission, and I think that should be enough for future use.

Now, let's have a look at how we would go about to configure the sending and receiving of data to and from the PLC.

2.2 Linking the inputs from the PLC

We are now going to connect the data being sent from the PLC with the controllers internal IO-handling system.

In the center pane, you will find this

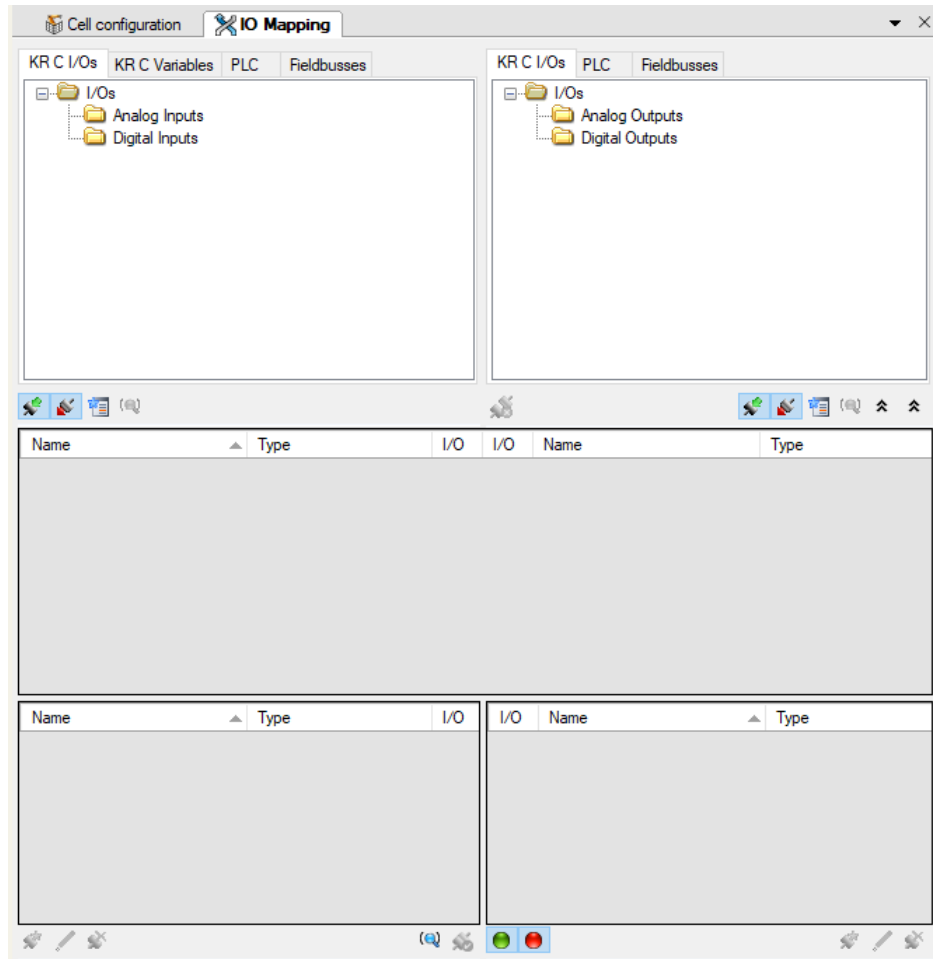


Figure 27: Make sure you are on the IO-maping tab

In the left pane, you are looking at the controllers internal IO's. We are going to connect this ones to the IO's comming from the ethercat brdige that are attached to the field-bus. So, in the right pane, select the fieldbusses pane.

Under sys-x44 you will find the EL6695-1001 bridge, select this in the right pane, and inputs ind the left pane.

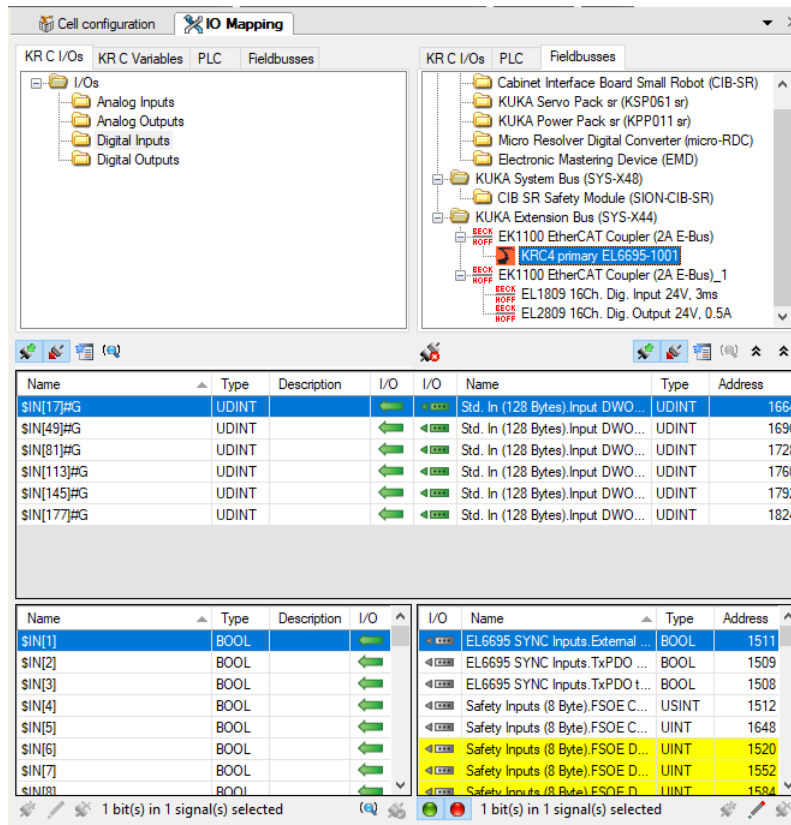


Figure 28: We are first adding the values sent from the PLC

in this picture, you will see that in the center horizontal pane, there is already some connections. This are the values that are from the force-torque sensor. As mentioned earlier, it is six values, and this matches up with the amount of connections we have.

The bottom right pane is the inputs from the ethercat bridge, and the bottom left, are the inputs on the controller we can connect to. The green arrow denotes the ones that are already connected.

One import thing to know and remember her, is that the controller treats every input as a boolean value, but fortunately, we can put several boolean inputs in to bytes, words and so on.

The data we are receiving are UDINT(Unsigned double integer), and as everyone knows, this is 32 bits. That means that we have to select 32 of the inputs on the bottom left pane to have the space for the data we are receiving from the PLC.

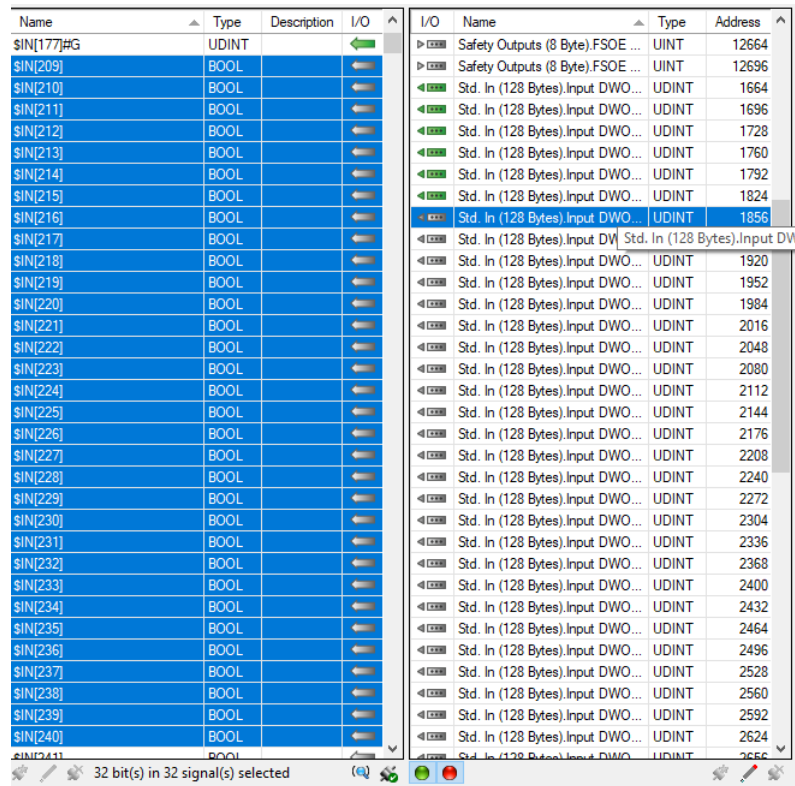


Figure 29: I have expanded the two bottom windows so we can see all the bits i have chosen

Notice that in the right window, that i have chosen std.In(128 Bytes.input) DWORD 6. This correlate to where we put the data when we were doing the linking in the PLC.

Now we can connect the input to the bits

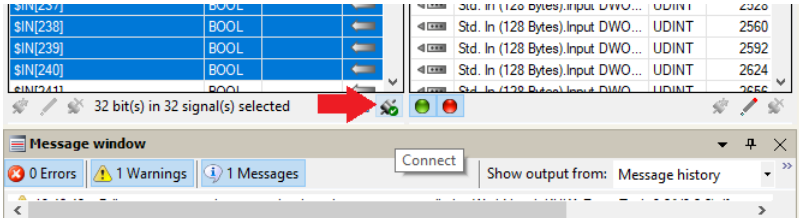


Figure 30: By clicking the connect button. Fortunately, I have marked this button with a big red arrow

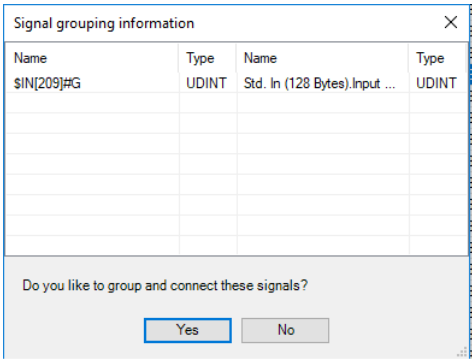


Figure 31: Click Yes

We should now have made this connection

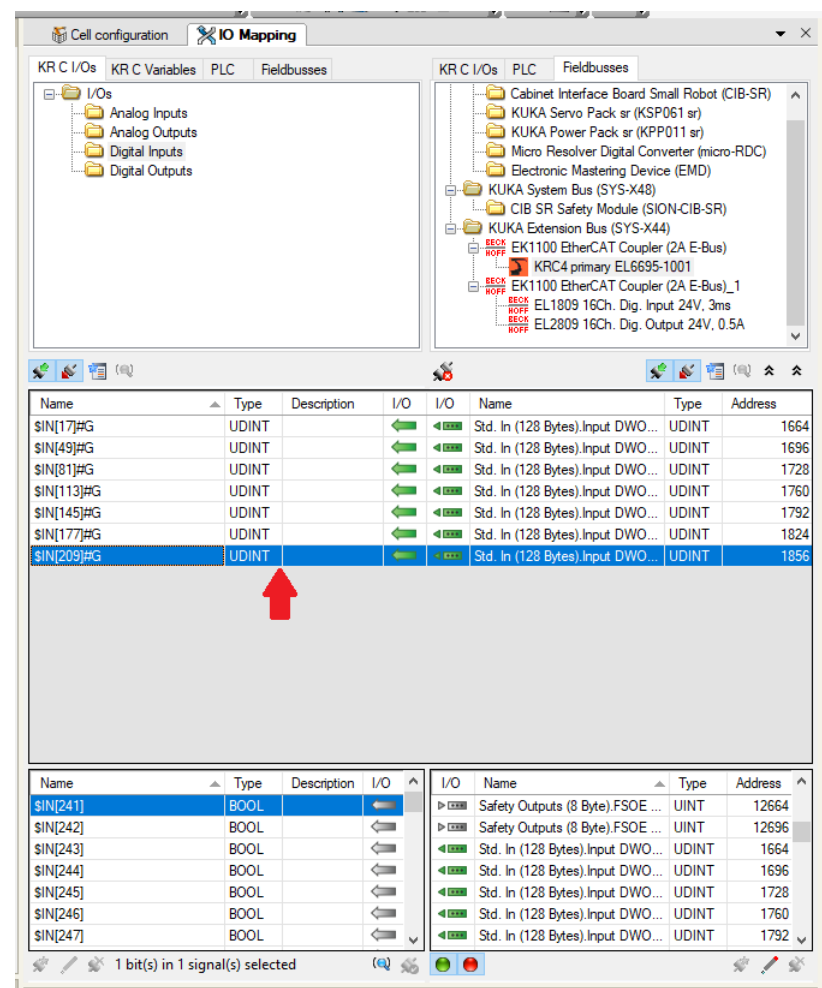


Figure 32: Again we have a big red arrow that show us the change. Take a note of the number to the left, this will be useful later

2.3 Linking the outputs to the PLC

The process of linking the outputs to the PLC is essentially the same, but since I am a nice guy, and I already made the section name I will walk you through this too.

Instead of choosing digital input in the upper left pane, we are now going to choose output

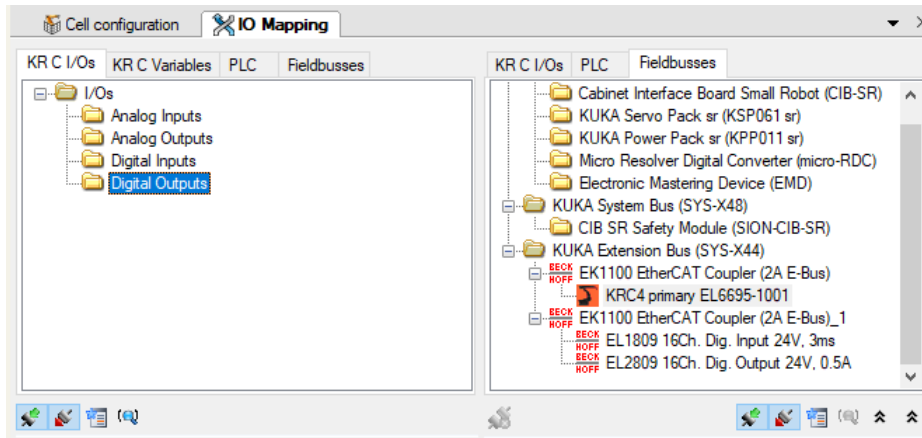


Figure 33: Select the digital output in the left pane, but stay on the EL6695-1001 fieldbus

The process is going to be the same as last time. In the bottom pane, available outputs are gray, and linked outputs are green. The green one's, are the digital outputs in the cabinet the robot controls.

Name	Type	Description	I/O
\$OUT[16]	BOOL		
\$OUT[17]	BOOL		
\$OUT[18]	BOOL		
\$OUT[19]	BOOL		
\$OUT[20]	BOOL		
\$OUT[21]	BOOL		
\$OUT[22]	BOOL		
\$OUT[23]	BOOL		
\$OUT[24]	BOOL		
\$OUT[25]	BOOL		
\$OUT[26]	BOOL		
\$OUT[27]	BOOL		
\$OUT[28]	BOOL		
\$OUT[29]	BOOL		
\$OUT[30]	BOOL		
\$OUT[31]	BOOL		
\$OUT[32]	BOOL		
\$OUT[33]	BOOL		
\$OUT[34]	BOOL		
\$OUT[35]	BOOL		
\$OUT[36]	BOOL		
\$OUT[37]	BOOL		
\$OUT[38]	BOOL		
\$OUT[39]	BOOL		
\$OUT[40]	BOOL		
\$OUT[41]	BOOL		
\$OUT[42]	BOOL		
\$OUT[43]	BOOL		
\$OUT[44]	BOOL		
\$OUT[45]	BOOL		
\$OUT[46]	BOOL		
\$OUT[47]	BOOL		
\$OUT[48]	BOOL		
\$OUT[49]	BOOL		
\$OUT[50]	BOOL		

Figure 34: Notice that i made an error, i linked it to output 1 instead. You should use the same as the data you linked in the PLC-program.

Select 32 available outputs and link them to output 0, as we chose in the PLC-program.

Push yes on the popup, and it should look something like this.

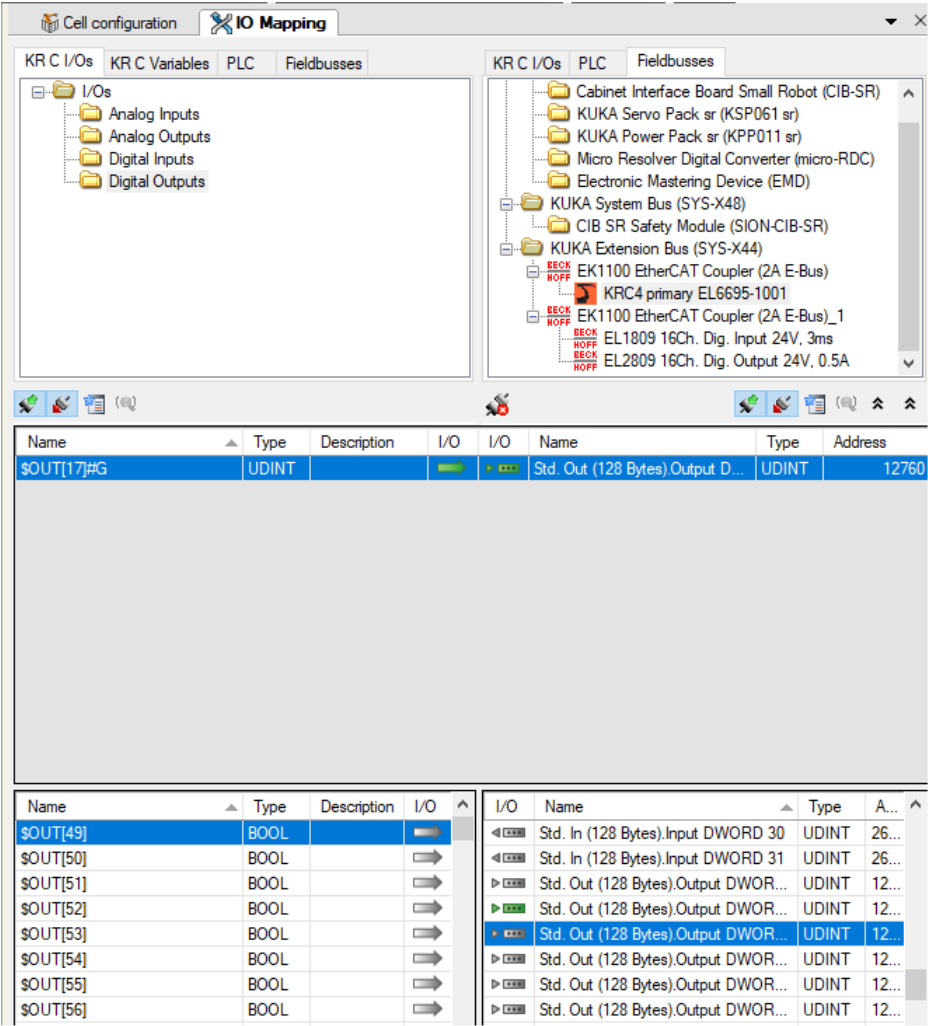


Figure 35: Success! Note: The error i made is still there

And of course, we need to deploy the changes.

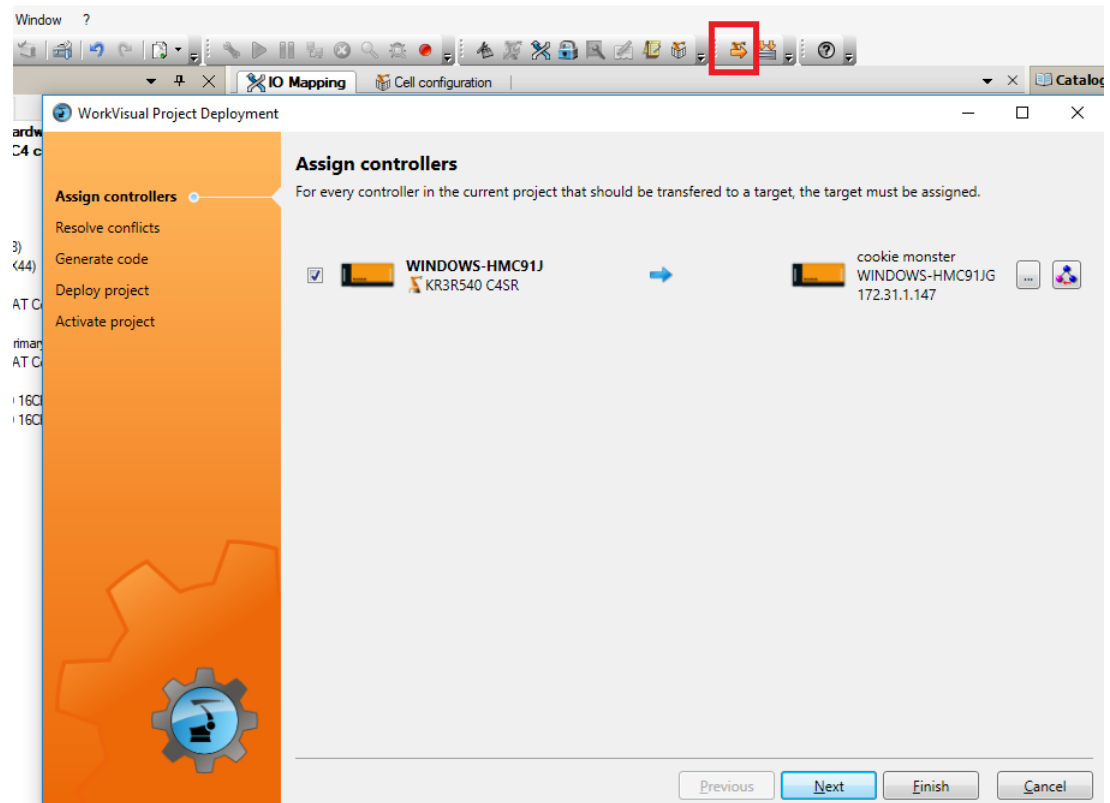


Figure 36: Click the button and follow the instructions

2.4 Give the output and input usable names

Since we are nice people, we would like to give the input and output names that we can remember. It can be hard to remember the data value we receive from the PLC is linked to input n to $n + 31$.

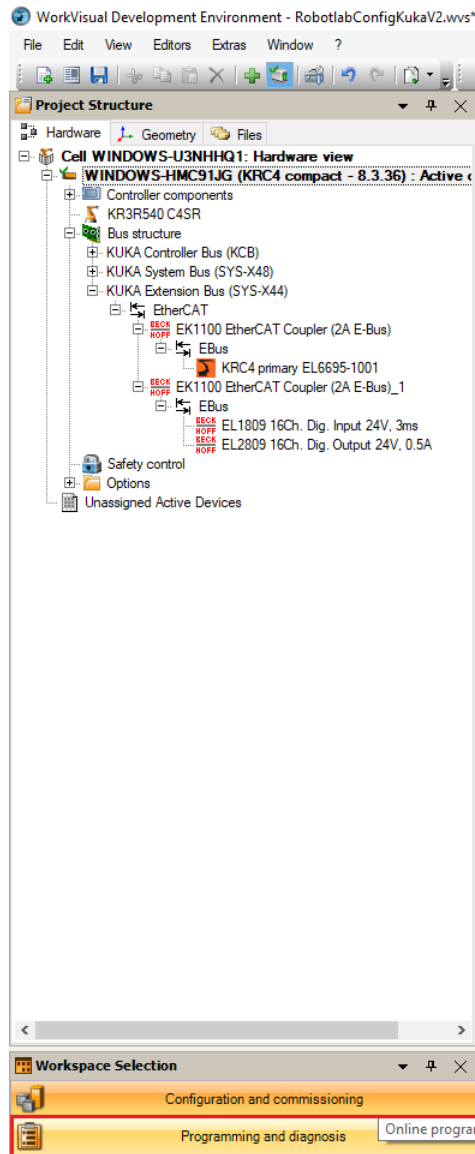


Figure 37: In the bottom left of the screen, click the programming and diagnosis button

This gives you a new window. This window allows you to navigate the files and folders on the controller.

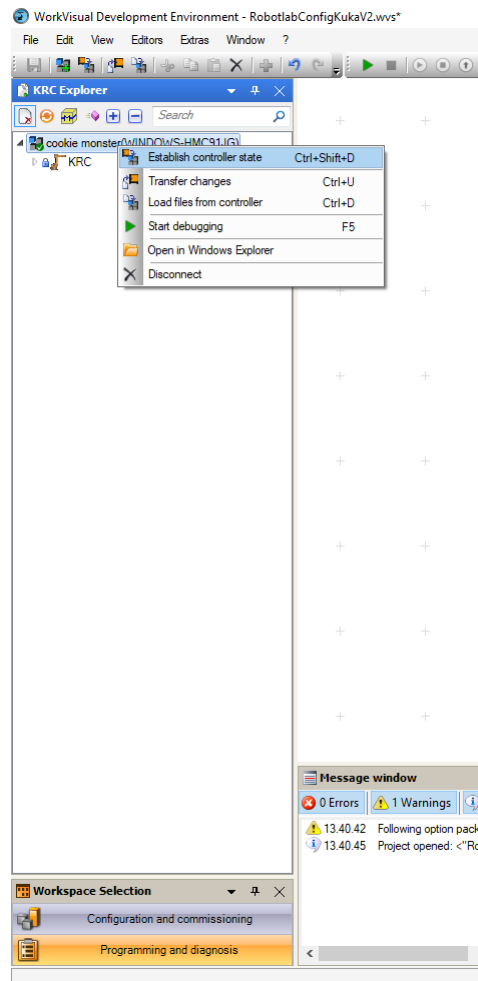


Figure 38: Right click the cookie monster and choose establish controller state to connect to the controller

Expand R1 and then system, double click config.dat and the center pane will give you a window with some code.

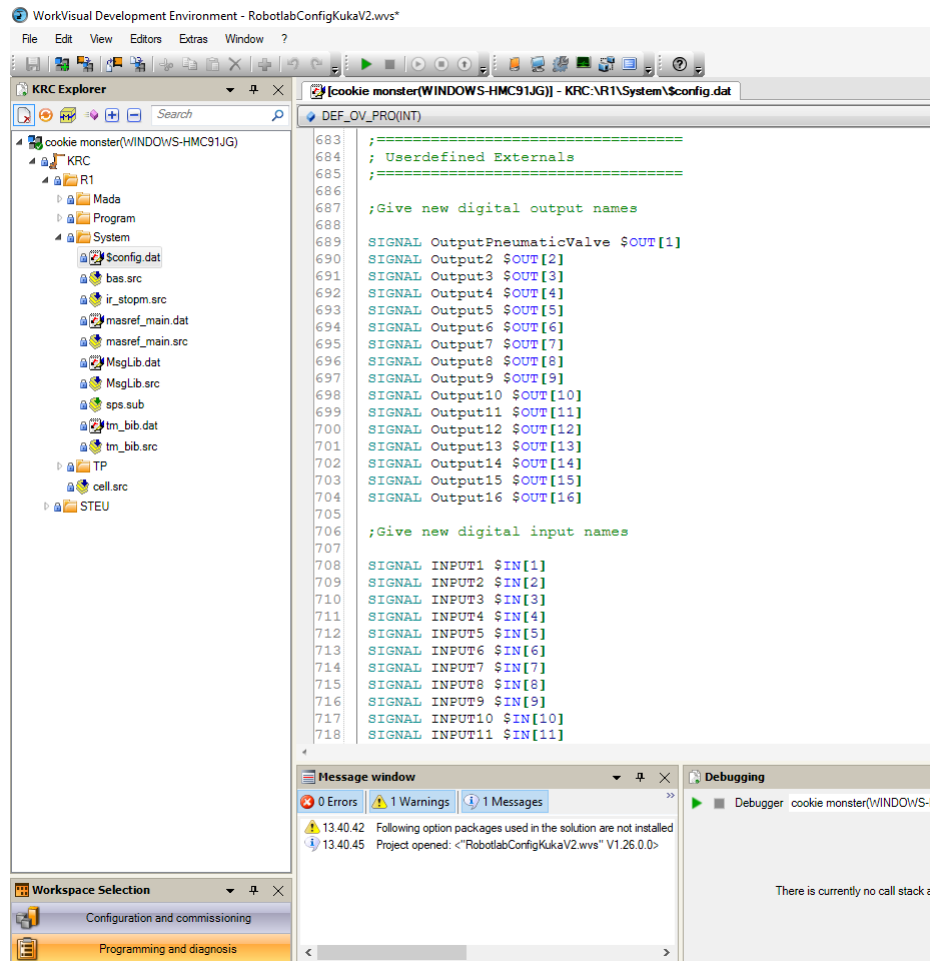
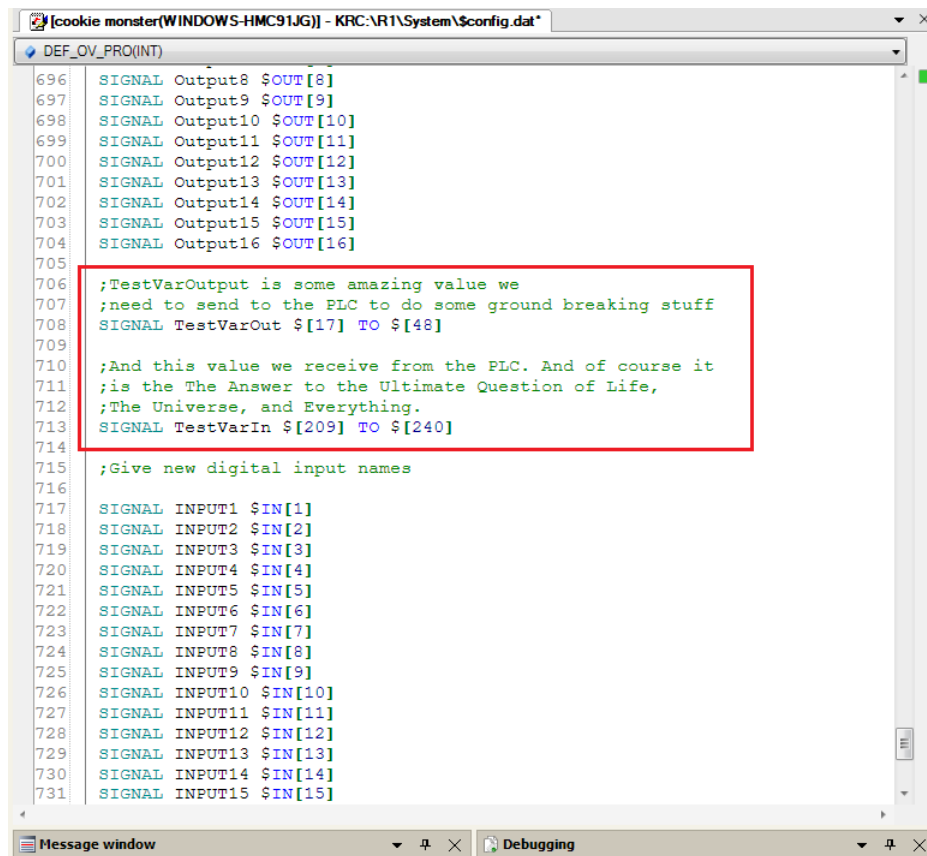


Figure 39: Double click config.dat and scroll to the bottom

Near the bottom you will find User defined externals. This is where we are going to give our input and output a name. And of course some nice comments, so it is easy to understand why they are there.



```
696 SIGNAL Output8 $OUT[8]
697 SIGNAL Output9 $OUT[9]
698 SIGNAL Output10 $OUT[10]
699 SIGNAL Output11 $OUT[11]
700 SIGNAL Output12 $OUT[12]
701 SIGNAL Output13 $OUT[13]
702 SIGNAL Output14 $OUT[14]
703 SIGNAL Output15 $OUT[15]
704 SIGNAL Output16 $OUT[16]
705
706 ;TestVarOutput is some amazing value we
707 ;need to send to the PLC to do some ground breaking stuff
708 SIGNAL TestVarOut $[17] TO $[48]
709
710 ;And this value we receive from the PLC. And of course it
711 ;is the The Answer to the Ultimate Question of Life,
712 ;The Universe, and Everything.
713 SIGNAL TestVarIn $[209] TO $[240]
714
715 ;Give new digital input names
716
717 SIGNAL INPUT1 $IN[1]
718 SIGNAL INPUT2 $IN[2]
719 SIGNAL INPUT3 $IN[3]
720 SIGNAL INPUT4 $IN[4]
721 SIGNAL INPUT5 $IN[5]
722 SIGNAL INPUT6 $IN[6]
723 SIGNAL INPUT7 $IN[7]
724 SIGNAL INPUT8 $IN[8]
725 SIGNAL INPUT9 $IN[9]
726 SIGNAL INPUT10 $IN[10]
727 SIGNAL INPUT11 $IN[11]
728 SIGNAL INPUT12 $IN[12]
729 SIGNAL INPUT13 $IN[13]
730 SIGNAL INPUT14 $IN[14]
731 SIGNAL INPUT15 $IN[15]
```

Figure 40: I have chosen to use the same variable name as i did on the PLC, but it is not something you must do

The last thing to do is to upload the changes

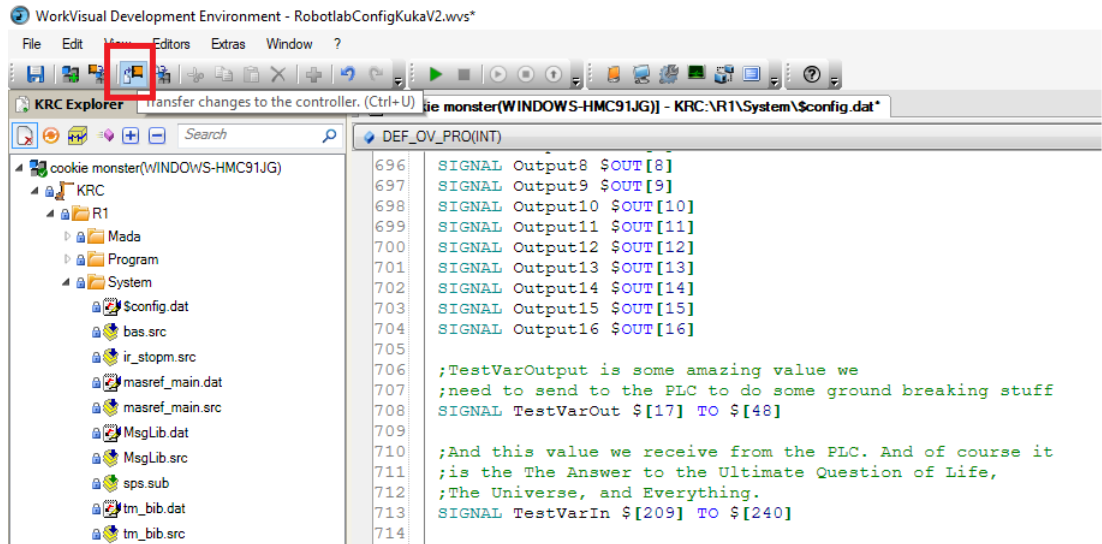


Figure 41: Just click the upload button

You will get a notification on the smart pad that an external user are trying to log in, and you have to be logged in on the smart pad as expert or higher

Congratulations, you have earned a black belt in data transmission between a Beckhoff PLC and Kuka robot controller 4.