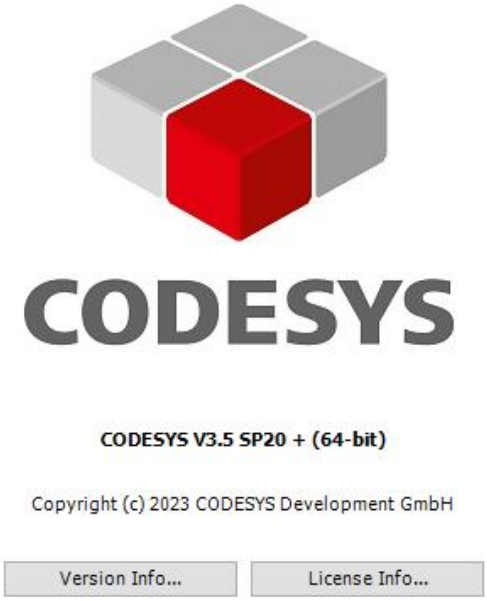
Codesys Ethernet/IP Implicit and Explicit Messaging to the Yaskawa YRC1000 Robot Controller



*Program was made using Codesys V3.5 SP20 + (64bit)

*Program was ran using Codesys Control Win V3 x64

*Achieved on a YRC1000 with the following version information

```
VERSION INFORMATION
SYSTEM    : YAS4.90.00A(JP/EN)-00
PARAMETER : 4.28/4.28
MODEL     : 1-06VXH12-A0*(AR1440)
APPLI     : ARC WELD
LANGUAGE  :  4.90-00-00/ 4.90-00-00

CPU       SYSTEM ROM(CPU1/CPU2) BOOT ROM   OS/FPGA
ACP01      4.90.00                         1.05-10
AIF01      --------                        16092200
PP         3.00-00              2.11-00     1.14
ASF01#0    2.26-00 /  2.26-00    1.00       16122200
ASF01#1   DummyMsf / DummyMsf   DummyMsf    15010100
ASF01#2   DummyMsf / DummyMsf   DummyMsf    15010100
ASF01#3   DummyMsf / DummyMsf   DummyMsf    15010100
ASF01#4   DummyMsf / DummyMsf   DummyMsf    15010100
```

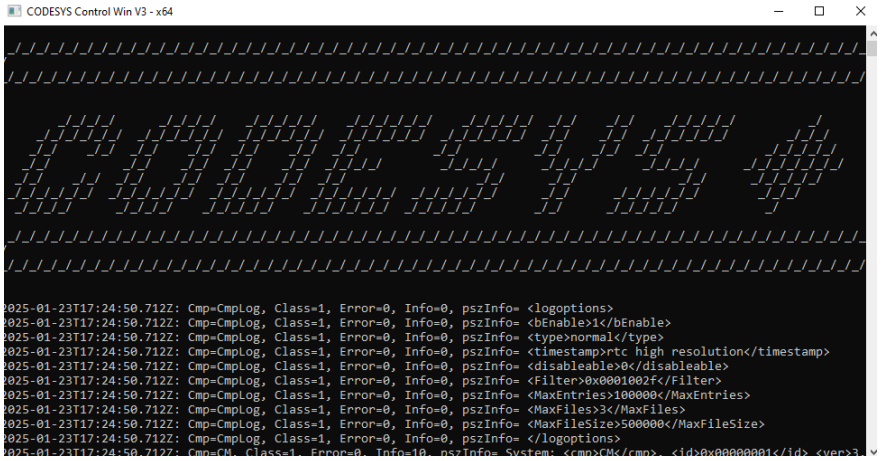
Michael Allen Michael.allen@motoman.com 01/23/2025

Overview

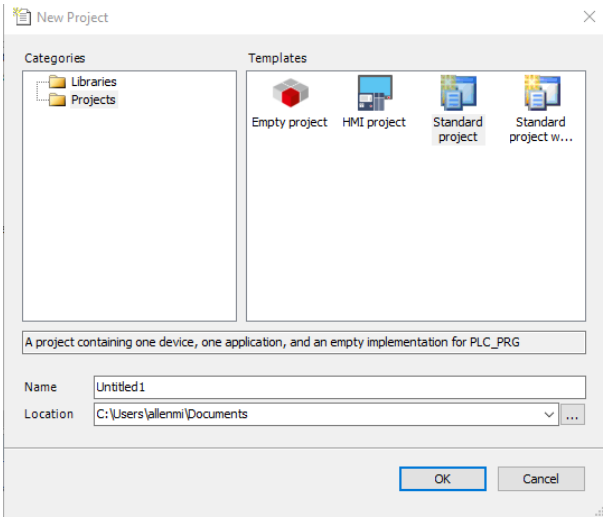
This document shows how to configure a Codesys Windows Soft PLC to communicate over EIP using both implicit and explicit messaging to a Yaskawa YRC1000 Robot Controller. In this example 8 bytes of input and output data was created and a position variable was transferred over the explicit connection.

PLC Setup

1. Download Codesys Control Win V3 x64 and install it and then run it. This is the “PLC”, you will download projects to it. Once installed, run it from the start menu, you will see a popup that will look like the image below, this indicates that the PLC is running, you cannot close this.

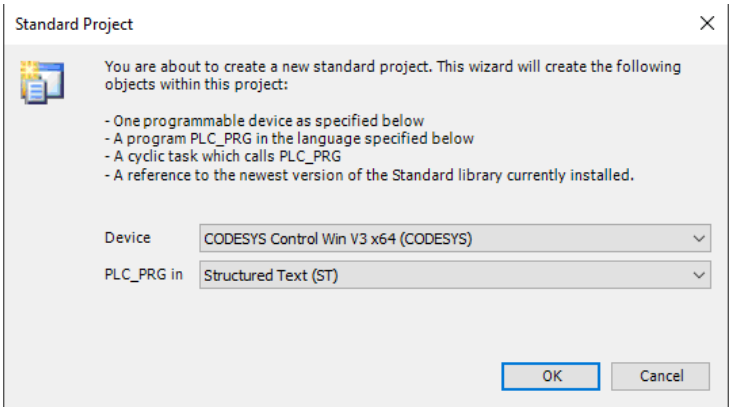


2. Download, install and run Codesys V3.5 SP20, this is the programming software. Select File->New Project.
3. Click standard project, give it a name and then press OK.

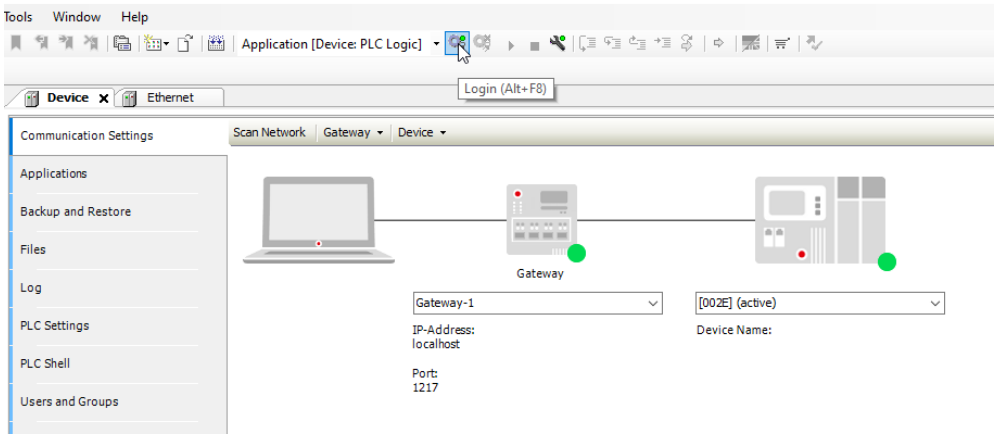


4. It will then ask you the device type and the programming type you would like to use. You can select whichever device type you would like to use however in the example I

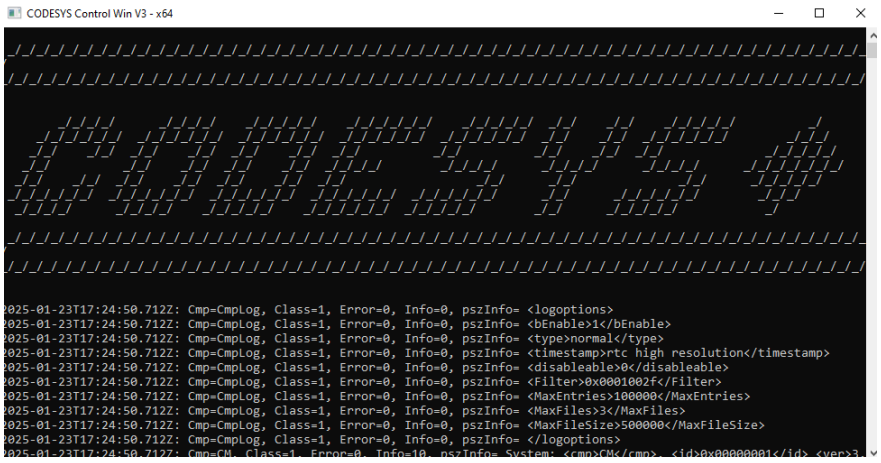
used Codesys Control Win V3 x64 which is the Windows runtime that can operate on any windows computer. I wanted to be able to test this on my laptop which is why I used this as the device. Choose Structured Text for the program type and press OK.



5. At the top of the screen press the little gear with a plug to go online with the PLC. Make sure you have started Codesys Control Win V3 x64 and you see the command line like page running.

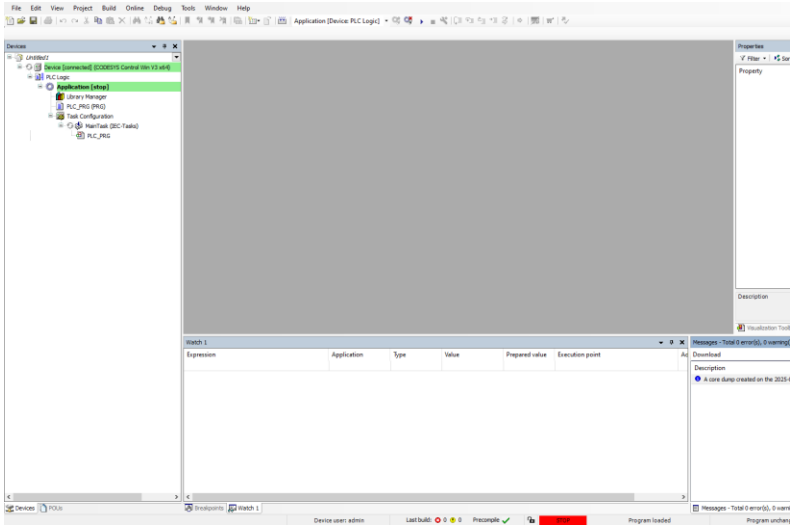


You should see this screen open on your computer.

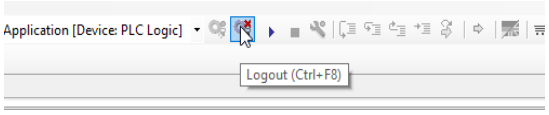


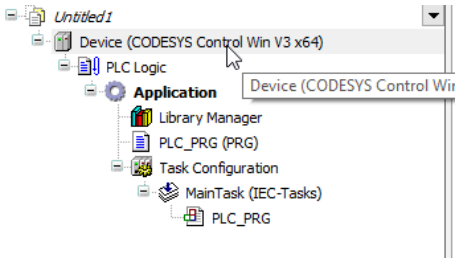
6. Once you press the gear it will ask you if you want to go online. Press yes and if required put in a password when running the PLC the first time. Your screen should

look like the image below with a green strip over the device and a red bar at the bottom saying it is in Stop mode.

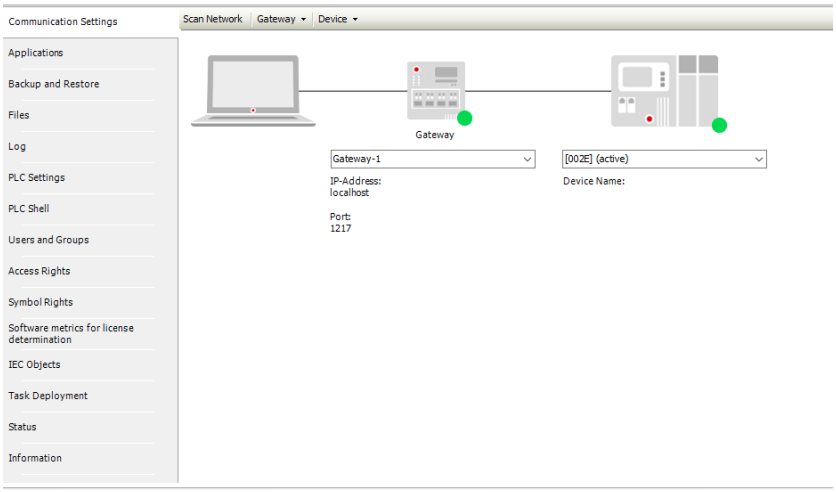


7. No go offline by pressing the gear beside the go online gear, then double click the Device to get to the device screen.

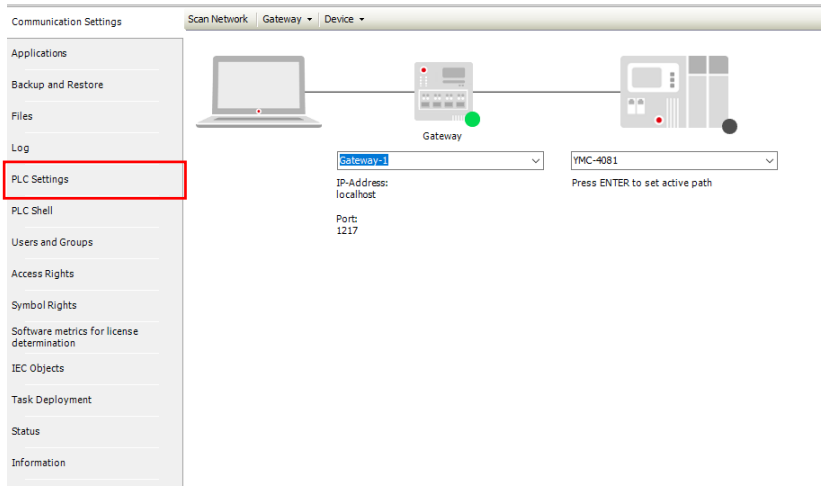




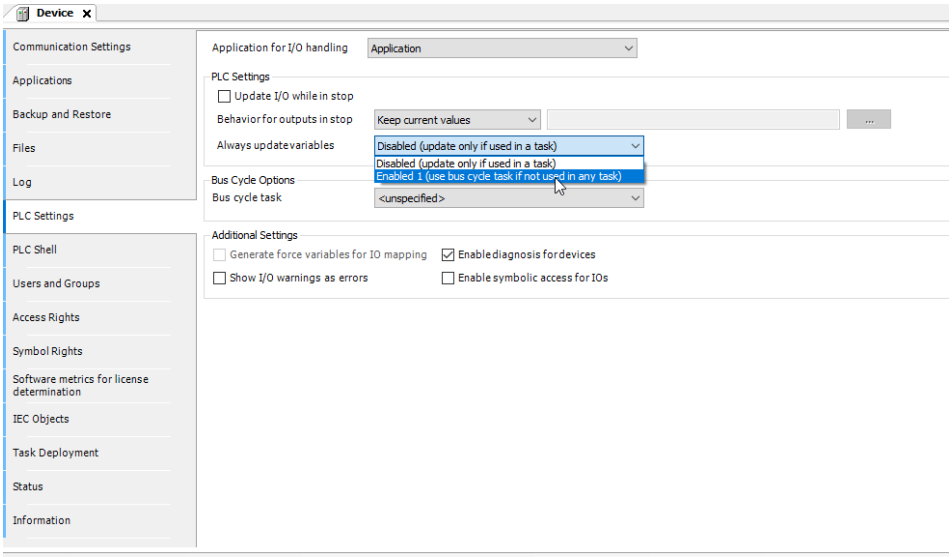
8. On the device screen you should see the gateway and the PLC should have (active) beside the PLC name, in my case it is 002E (active).



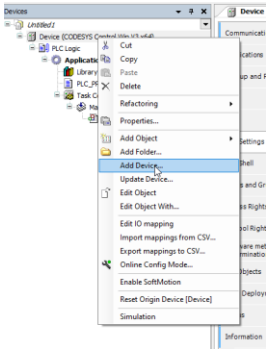
9. On the device screen press PLC Settings.



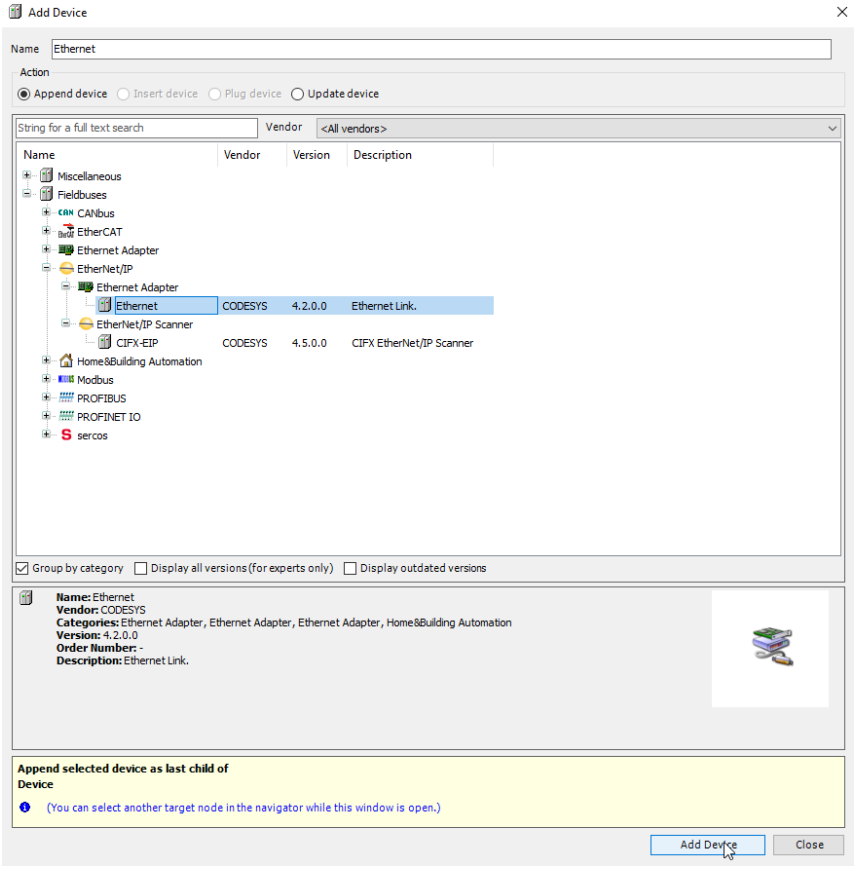
10. **Change "Always Update Variables" from disabled to Enabled 1 so you can see the bits being transferred between the robot and the PLC without needing to use them in the program. Otherwise if you just try and toggle them on in the watch table they will not work!**



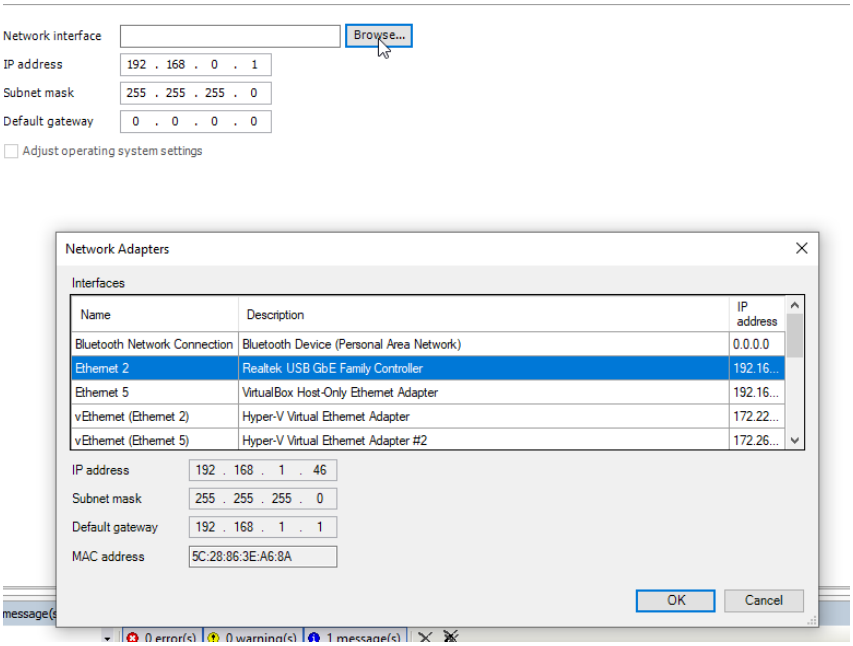
11. Right click on the device again and select add device.



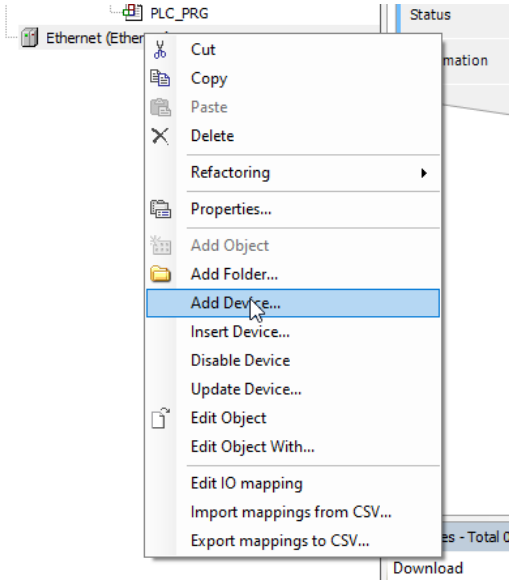
12. Click Fieldbuses, Ethernet Adapter, Ethernet and then click Add Device on the bottom right hand corner of the page.



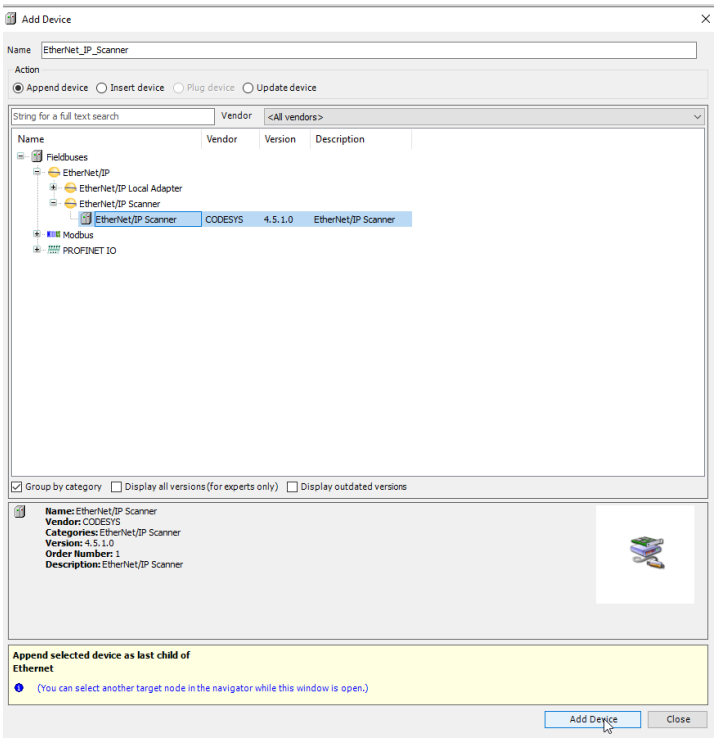
13. Double click the ethernet device and under the general tab select “Browse..” beside the Network Interface to select the network interface you would like to use. In my case I want Ethernet 2 so I select it and press OK.



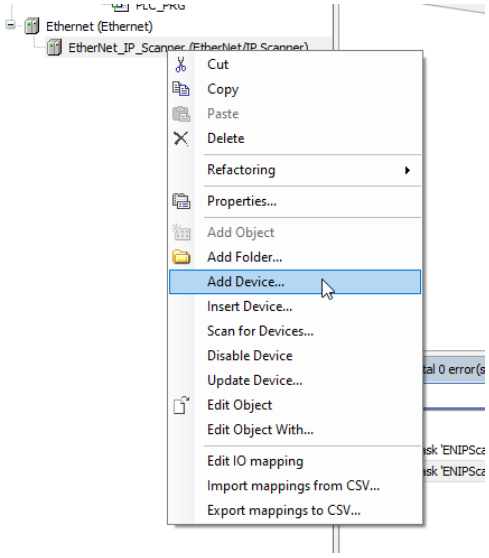
14. Right click the Ethernet device and click add device.



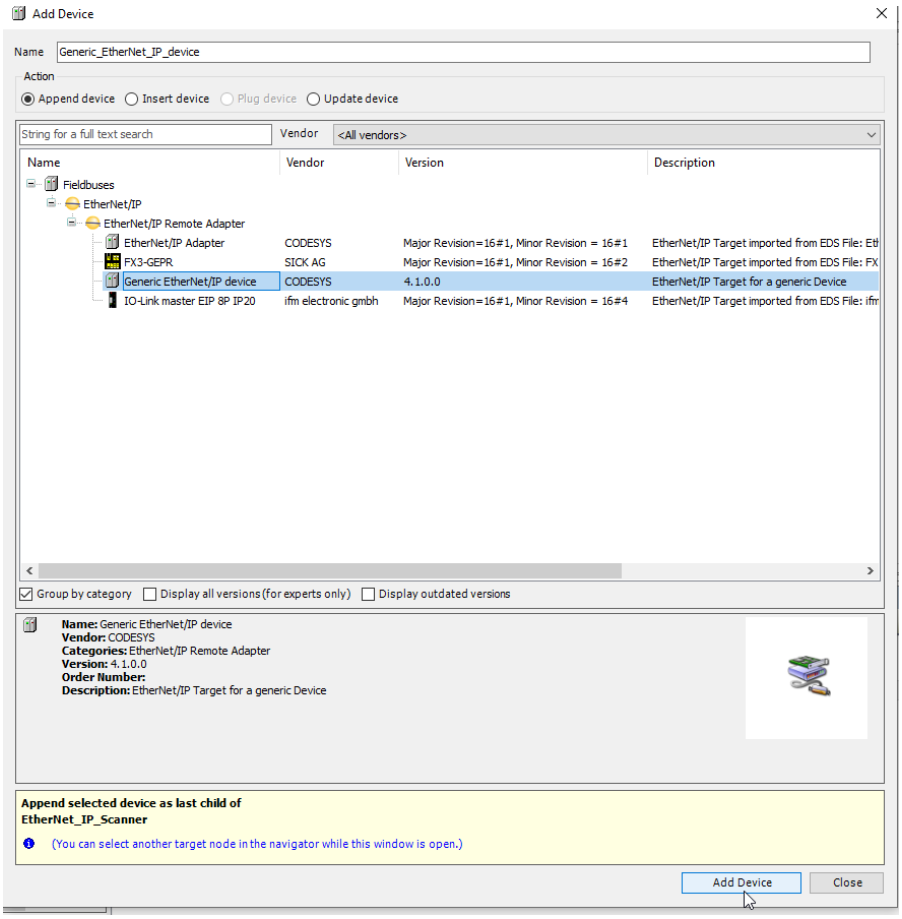
15. Under Fieldbuses, EtherNet/IP, EtherNet/IP Scanner select EtherNet/IP Scanner and then press Add Device.



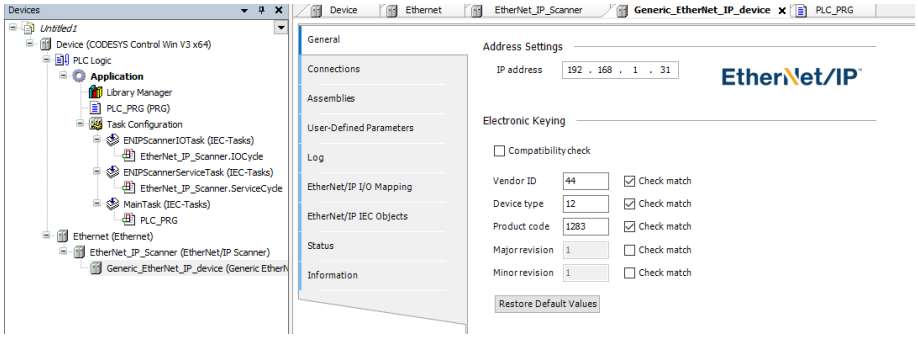
16. Rick click the EtherNet_IP_Scanner and press Add Device just like we did for the scanner.



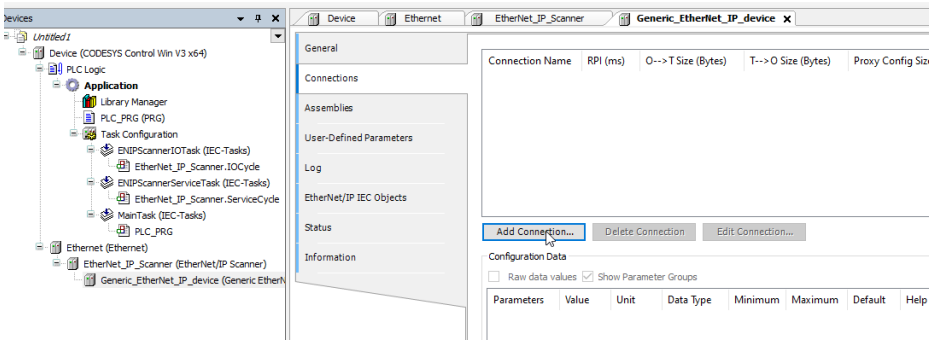
17. Under Fieldbuses, EtherNet/IP, EtherNet/IP Remote Adapter select Generic EtherNet/IP device and then select add device. Double click on the newly added device.



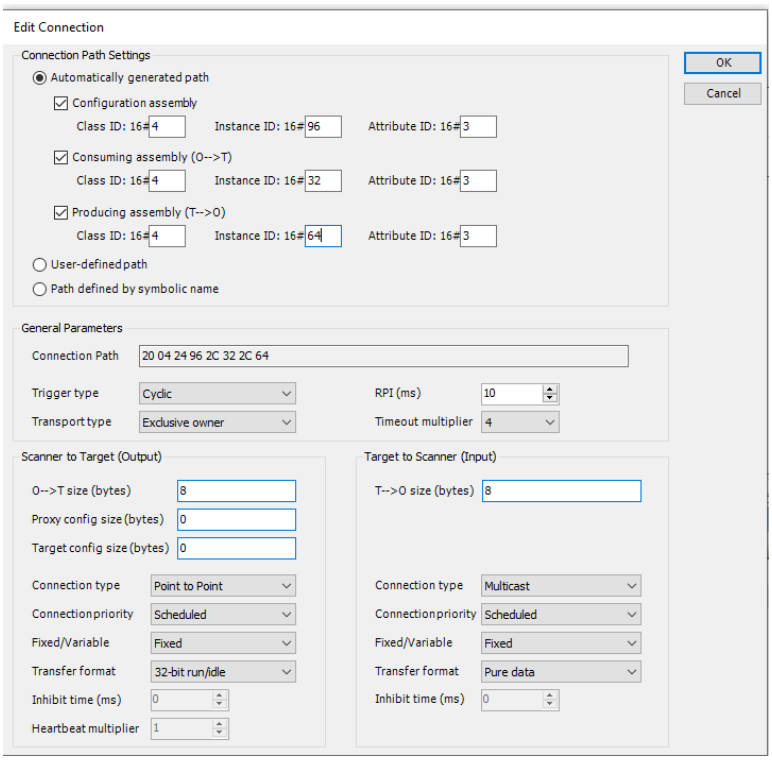
18. On the general tab of the Generic EtherNet IP Device change the IP address to the address of the robot, in my case it was 192.168.1.31. The Vendor ID is 44, Device Type is 12 and Product Code is 1283 for the YRC 1000.



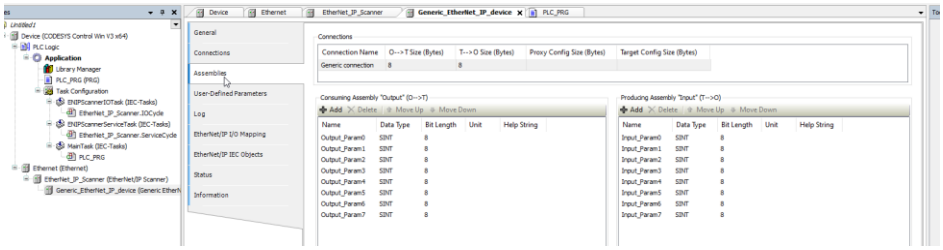
19. Click the Connections tab and then select Add Connection…



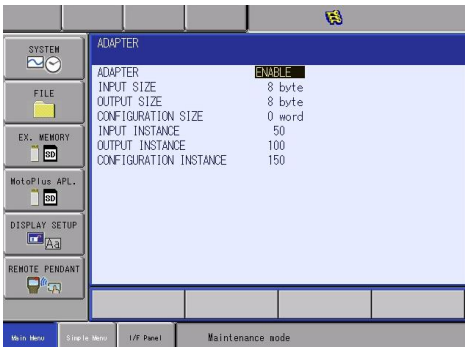
20. The Configuration Assembly Instance is 96. The Consuming Assembly Instance is 32 and the Producing Assembly Instance is 64. The Class ID is 4 for all instances and the Attribute ID is 3 for all instances. The O→T size and T→O sizes are set to 8 bytes and the RPI(ms) is set to 10.



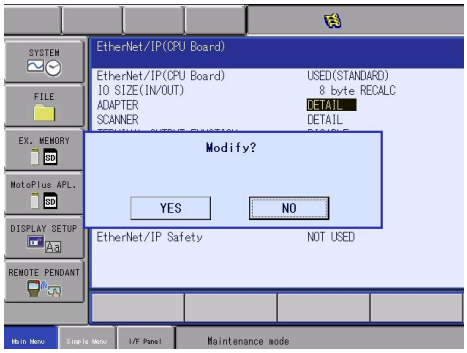
21. Click the assemblies tab and change the data types of the outputs and inputs to SINT from BYTE.



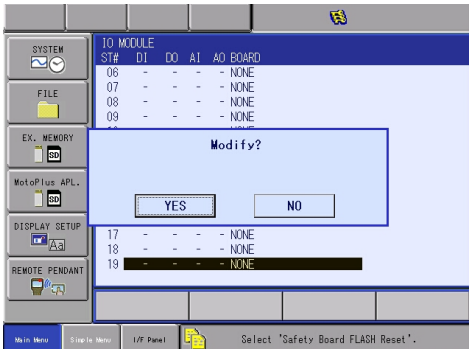
22. Now you need to complete the setup on the robot. Follow SSGW-509 for Ethernet IP setup. In Maintenance->System->Setup->OptionFunction->EtherNet/IP(CPU Board) press Detail. Press detail under Adapter. Enabled the Adapter and put in the settings below. Then complete the EtherNet/IP setup by pressing Enter and Modify and auto allocate the IO.



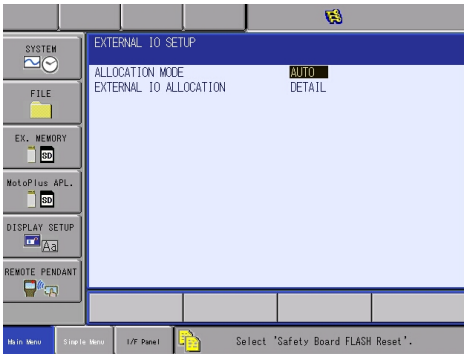
Enabled adapter and change instances and IO size, press enter.



Press enter and then Modify.

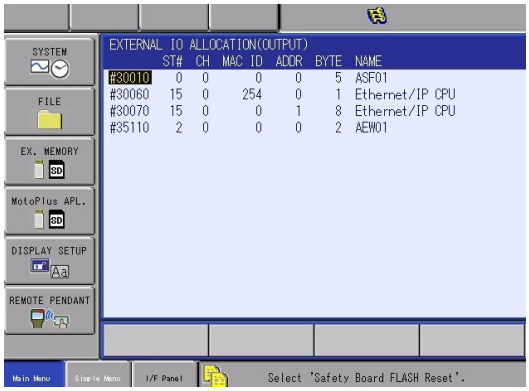


Press enter twice and then Modify.

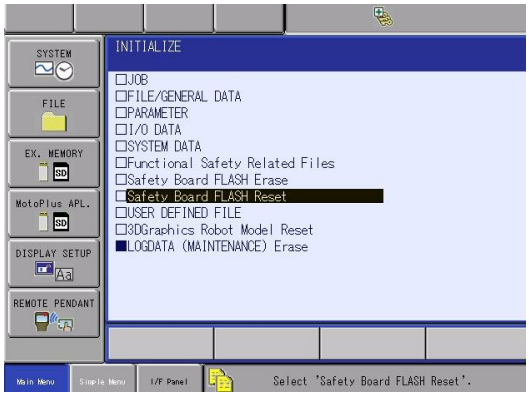


Set Allocation Mode to AUTO and press enter.

Take note where the 8 bytes of input data get mapped to, in this case they start at #20070, press enter.

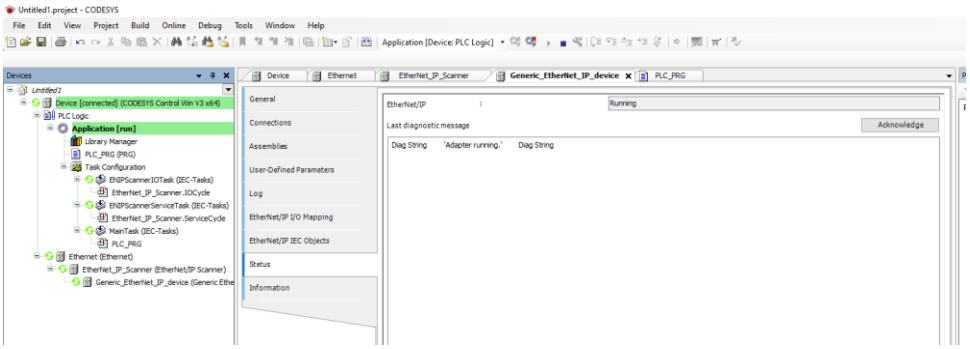


Take note where the 8 bytes of output data get mapped to, in this case they start at #30070, press enter and Modify.

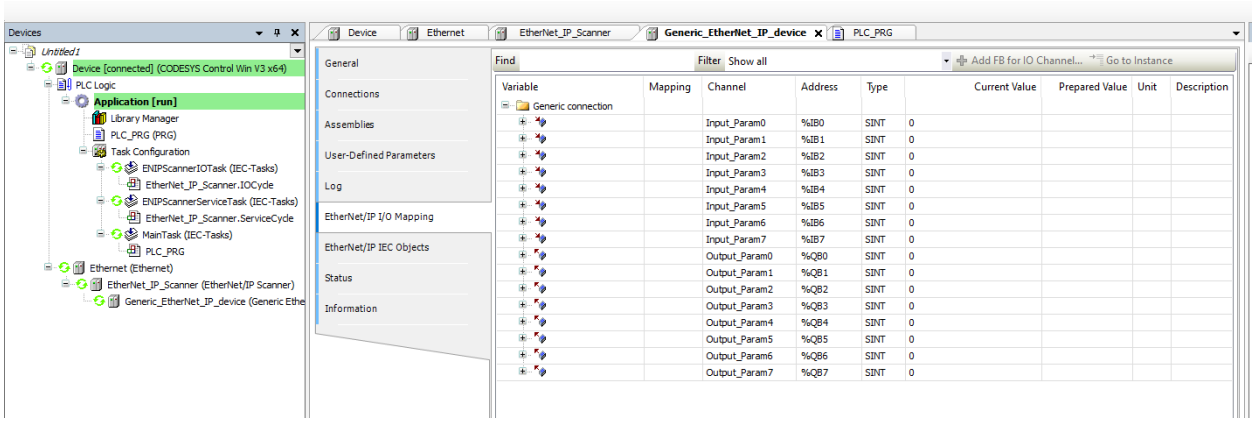


Perform safety flash reset and then reboot the controller in online mode.

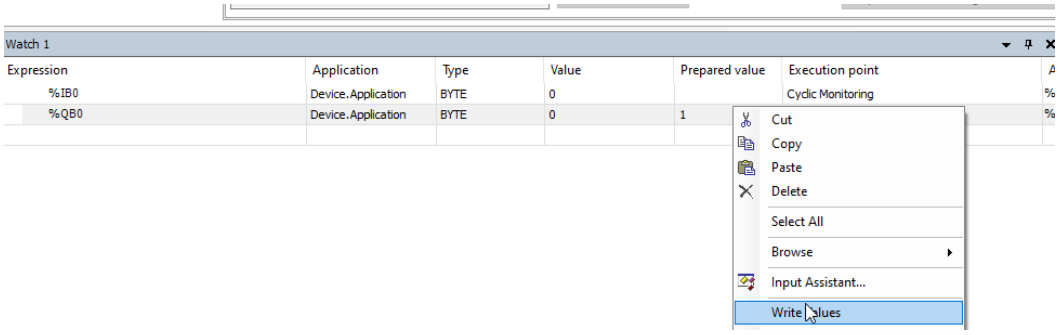
23. Once the robot boots back up you can go online with the PLC by pressing the gear and then downloading to the PLC. Then press the play button to start the PLC. Your screen should look like the image below. Note you should see green circles around all devices if connection



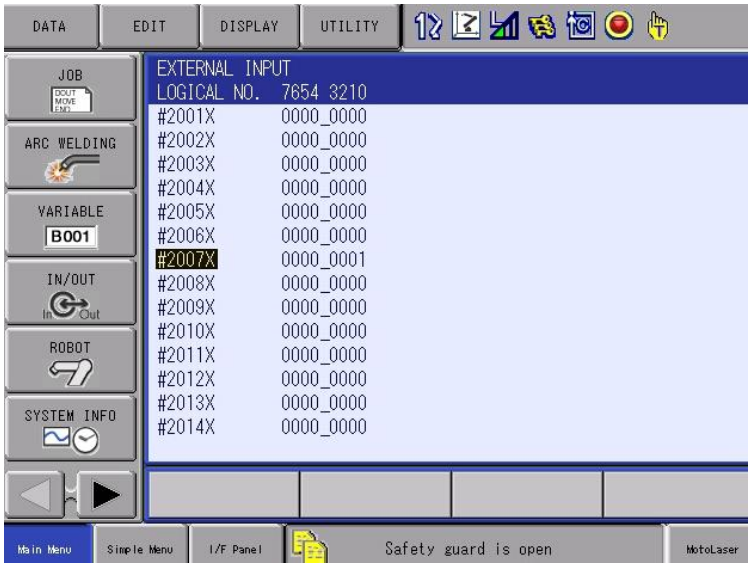
24. Now to test implicit connections we are going to turn on some bits on both the robot and PLC and make sure the other can see them. Click on the EtherNet/IP I/O Mapping tab and notice the first Input and first output addresses. Mine are located at %IB0 and %QB0.



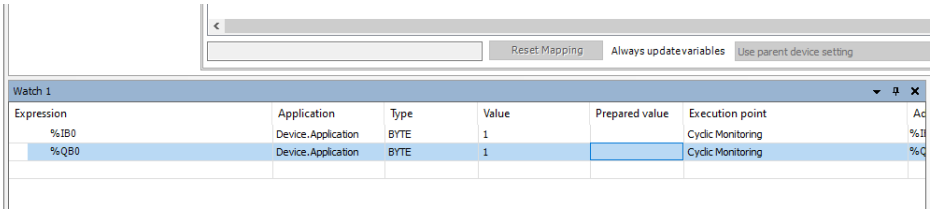
25. In the watch table add %IB0 and %QB0. Click on prepared value on %QB0 and change it to 1. Then right click and select Write Values, this will write a value of 1 to the first bit in the first byte of output data to the robot.



26. Now on the pendant click IN/OUT->External Input. Look at the first bit that was mapped when you did the auto allocation, in my case it was #20070. It should now be a 1 instead of a 0.



27. Now to test outputs go to IN/OUT->External Output. Click Display->Detail and then go to the start of your EtherNet/IP Outputs, which in my case was #30070. Hover over the box, press select to turn on Sim and then hover over the circle and press Interlock and Select at the same time to simulate the output. On the PLC program you should see %QB0 change from 0 to 1.



28. Now to test explicit communication we are going to read a position variable and then write it to another position variable. Please go offline with the PLC, copy the logic and variables to your PLC and then download and go back online with the PLC.

Device | Ethernet | EtherNet_IP_Scanner | Generic_EtherNet_IP_device | PLC_PRG

```
PROGRAM PLC_PRG
VAR
    reset               : ENIP.Reset;                   (* fb to reset a cip class *)
    getAttributeSingle  : ENIP.Get_Attribute_Single;    (* fb to get a single attribute from a cip class *)
    setAttributeSingle  : ENIP.Set_Attribute_Single;    (* fb to set a single attribute of a cip class *)
    getAttributeAll     : ENIP.Get_Attributes_All;      (* fb to get all attributes of a cip class *)
    setAttributeAll     : ENIP.Set_Attributes_All;      (* fb to set all attributes of a cip class *)
    udiReceivedData     : UDINT;
    done                : BOOL;
    busy                : BOOL;
    xError              : BOOL;
    startTransfer       : BOOL;
    axes                : ARRAY[0..13] OF DWORD;
END_VAR
```

Variable declaration

```
getAttributeAll(xExecute:= startTransfer,
    itfEtherNetIPDevice:= Generic_EtherNet_IP_Device,   (* instance of the device (instance is found in the I/O Image of the device) *)
    eClass:= 16#7F,                                     (* cip class which contains the desired attribute *)
    dwInstance:= 1,                                     (* value of 0 is class level, range from 1..x is instance level *)
    pData:= ADR(axes),                  (* data buffer *)
    udiDataSize:= SIZEOF(axes),         (* size of the data buffer *)                  (* attribute no. 6 of the tcp/ip interface object is the Hostna
    xDone=> done,
    xBusy=> busy,
    xError=> xError ,
    eError=> ,
    udiReceivedDataSize=> udiReceivedData);

setAttributeAll(
    xExecute:= startTransfer,
    itfEtherNetIPDevice:= Generic_EtherNet_IP_Device,   (* instance of the device (instance is found in the I/O Image of the device) *)
    eClass:= 16#7F,                                     (* cip class which contains the desired attribute *)
    dwInstance:= 2,                                     (* value of 0 is class level, range from 1..x is instance level *)
    pData:= ADR(axes),                  (* data buffer *)
    udiDataSize:= SIZEOF(axes),         (* size of the data buffer *)                  (* attribute no. 6 of the tcp/ip interface object is the Hostna
    xDone=> done,
    xBusy=> busy,
    xError=> xError ,
    eError=>
    );


    startTransfer := FALSE;
```

Program

29. In this program we are calling a getAttributeAll function that is in the standard EtherNet IP Library included with Codesys, if your program does not recognize it you may have to download it and install it. The CIP class for Position Variables is 16#7F as stated in the YRC1000 EtherNet/IP Manual. There is 13 x 4 Byte values that will be returned corresponding to the different axis and rotation values for the position. I made an array called Axes that will store all the values.

■ **Read and write a robot position-type variable (P)**

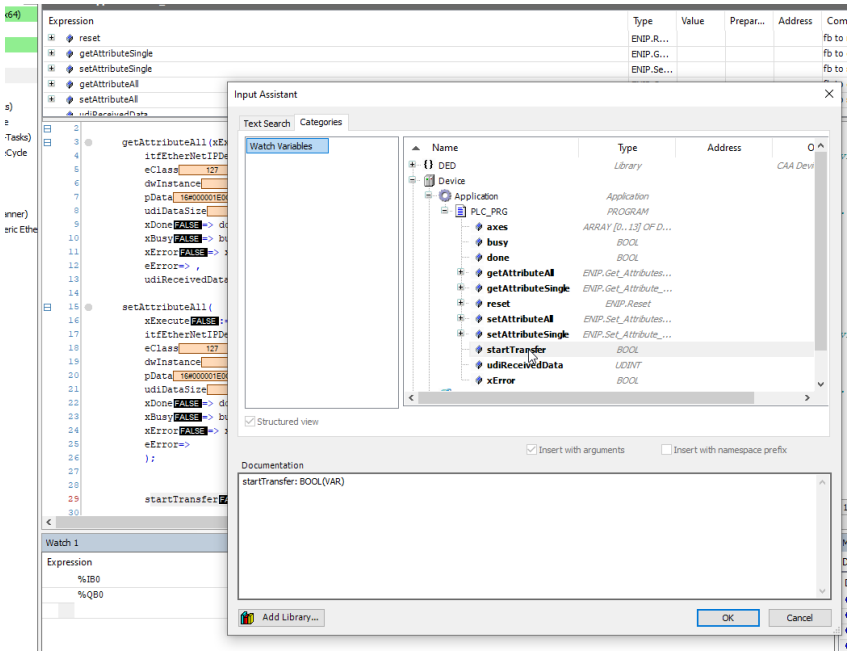
Table 5-16(a): Required Format

Division	Outline	Size	Data	Explanation
Header	Class	2 Byte	0x7F	Vendor-specific
	Instance	2 Byte	Specify the variable P to read/write From 0 or from 1	Specify the variable P number (Parameter: RS022=1) Specify the variable P number +1 (Parameter: RS022=0)
	Attribute	1 Byte	Specify the read data type 1 to 13	Specify the data number of the position information 1: Data type to 13: 8th axis data
	Service	1 Byte	Specify the data access method 0x01: Get Attribute All 0x0E: Get Attribute Single 0x02: Set Attribute All 0x10: Set Attribute Single	

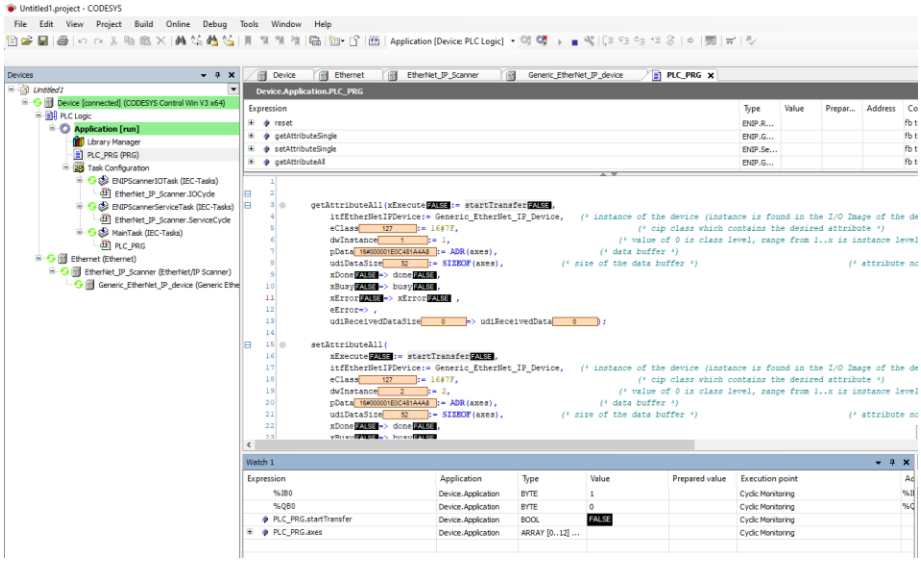
Table 5-16(b): Response Format (When Reading All Attributes/Omitting Headers)

Division	Outline	Size	Explanation
Data	Data type	4 Byte	The position data type will be output 0: Pulse 16: Base 17: Robot 18: Tool 19: User coordinates
	Configuration	4 Byte	The configuration will be output bit 0: Back bit 1: Lower arm bit 2: No flip bit 3: R axis ≥ 180° bit 4: T axis ≥ 180° bit 5: S axis ≥ 180°
	Tool number	4 Byte	
	User coordinate number	4 Byte	
	Extended configuration	4 Byte	Output the 7-axis robot extended configuration bit 0: θL ≥ 180° bit 1: θU ≥ 180° bit 2: θB ≥ 180° bit 3: θE ≥ 180° bit 4: θW ≥ 180°
	1st axis data	4 Byte	The following values will be output: For pulse: Each axis' pulse value For base: Length (μm) Angle (0.0001°) Data on each axis is set in ascending order. The value zero is set to a non-existing axis.
	2nd axis data	4 Byte	
	3rd axis data	4 Byte	
	4th axis data	4 Byte	
	5th axis data	4 Byte	
	6th axis data	4 Byte	
	7th axis data	4 Byte	
	8th axis data	4 Byte	

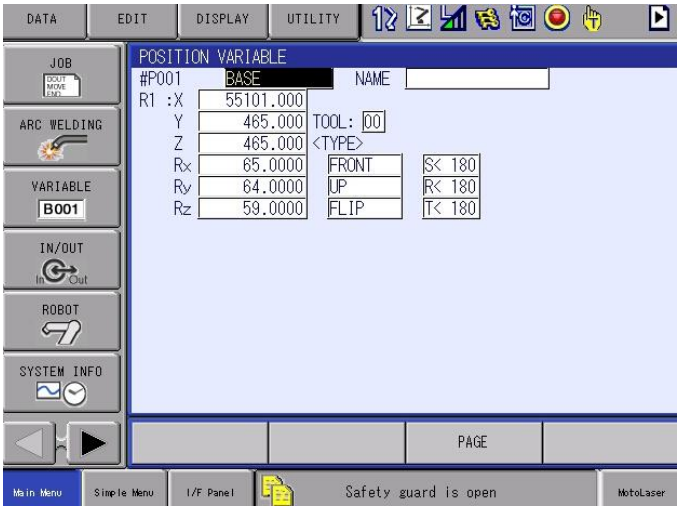
30. In the watch table click a new line, click the … and then select the Axes array variable and the Start Transfer bools to be added to the watch table. Press OK after highlighting both.



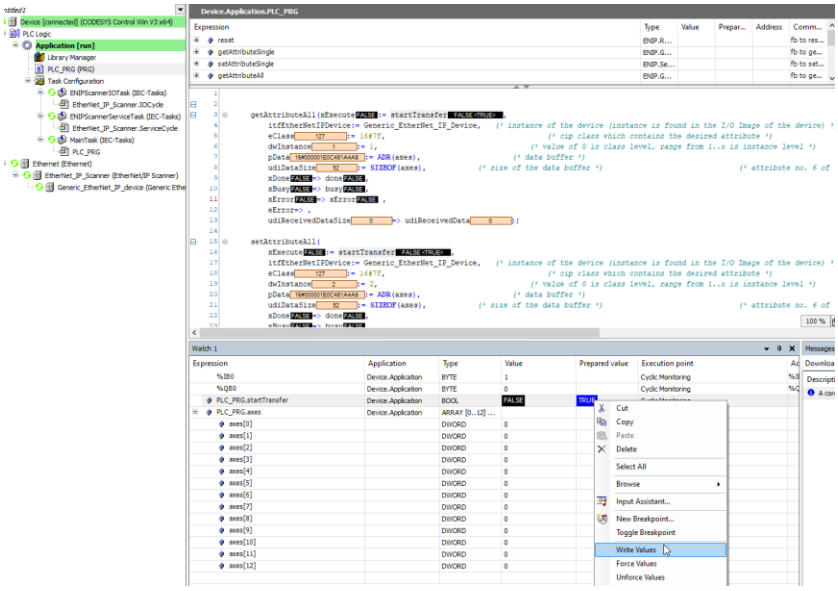
31. Go online by pressing the gear, download the program and then press the play button to run the PLC. Your screen should look like this after you are done.



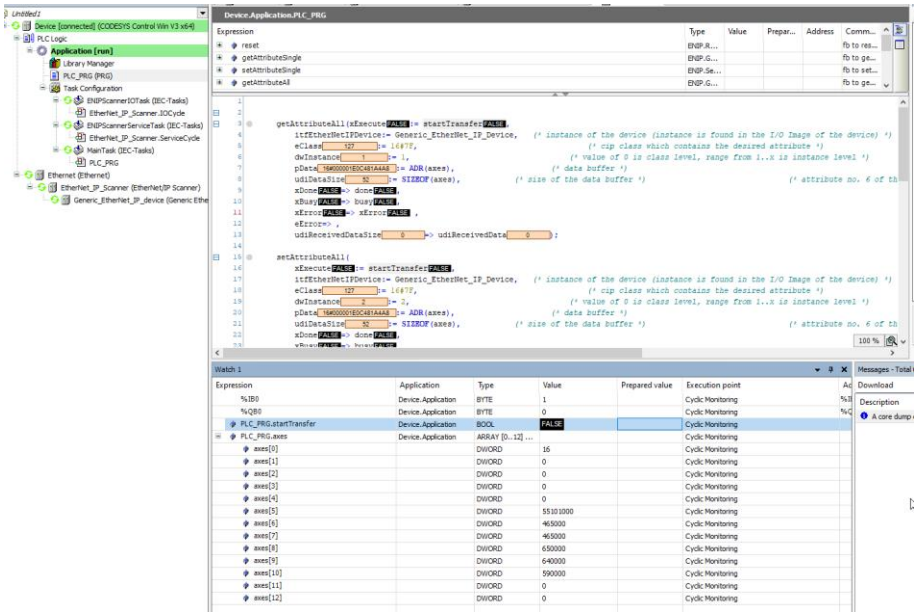
32. On the robot make sure you have a position value in #P1 other than all 0's so you know the value is read and then transferred. In my case I have the below position values.



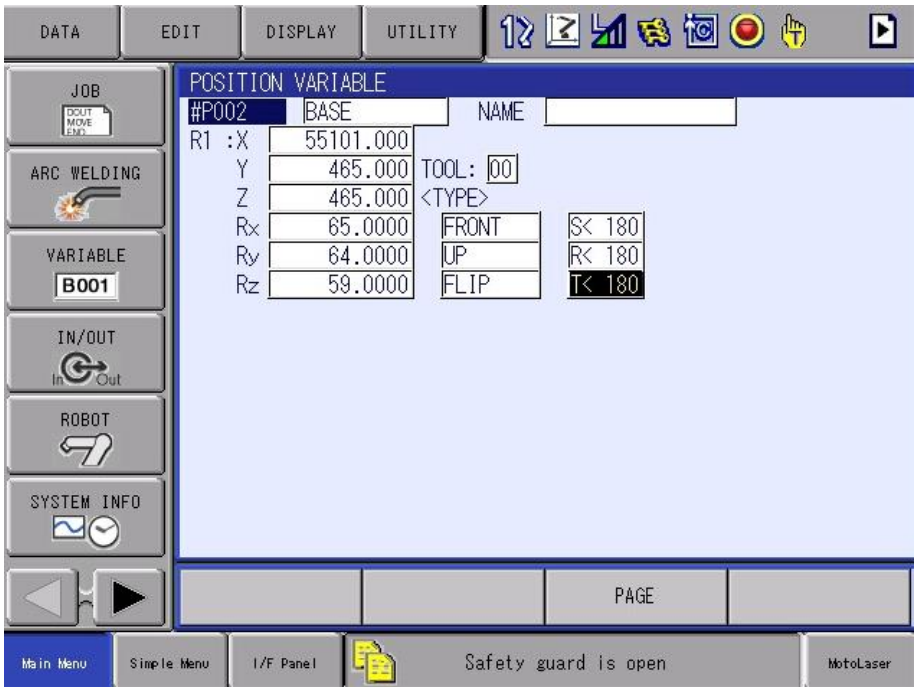
33. Click the box beside the Start Transfer bool under the prepared value heading until it changes to TRUE. Then right click and write value. This will run the program one time, it will read P1 and copy it into the Axes array and then copy it back to the robot and place it in P2.



34. You should see the axes array populate with your position and if you look at P2 on the robot you should see the values from P1



Note the values in axes array.



Note the values in P2.

35. Congratulations you have successfully set up Implicit and Explicit EtherNet/IP connections from a Codesys Control Win V3 X64 to a YRC1000. Codesys is the basis for a bunch of other PLC brands including but not limited to Wago, Festo, Lenze, Schneider, Eaton and Beckhoff so this documentation may help assist in setup for those PLC's as well.