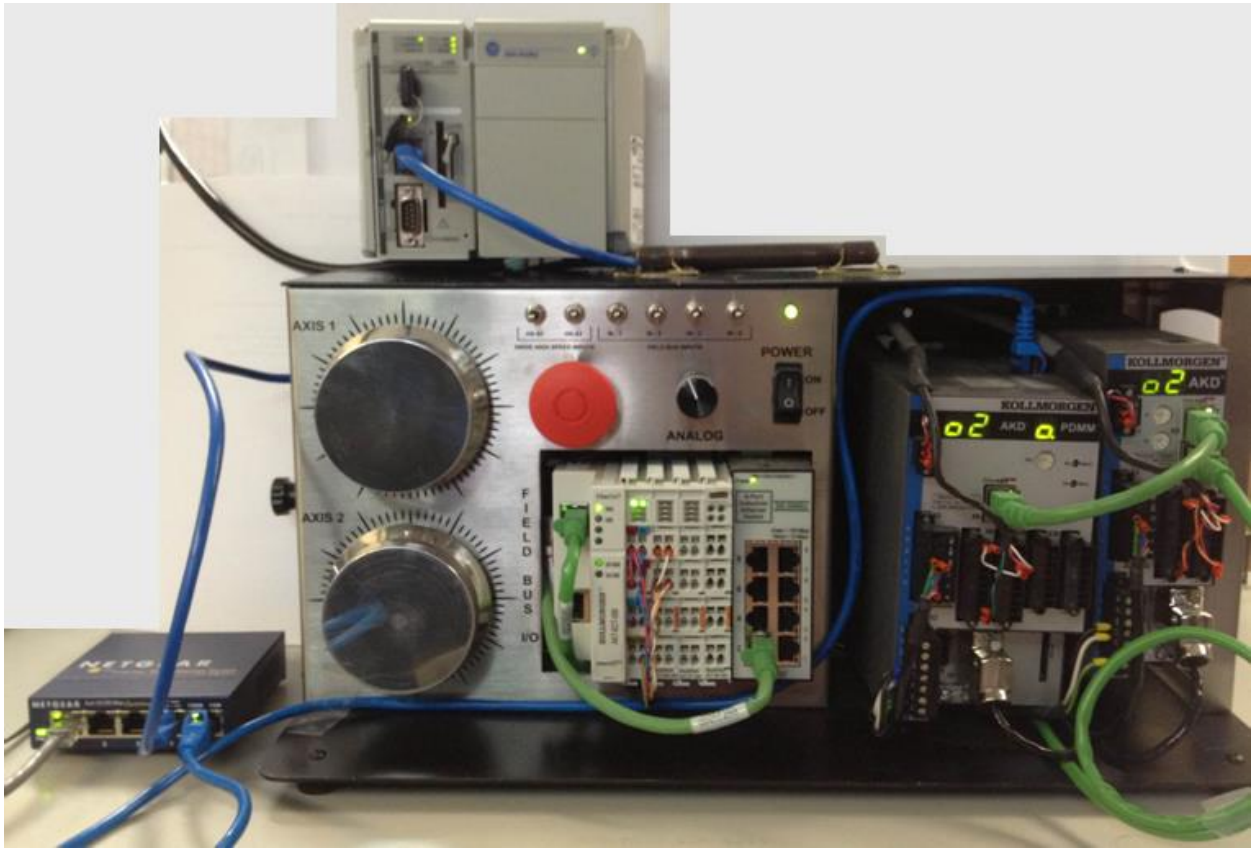
KOLLMORGEN

Because Motion Matters™

Using EtherNet/IP with RSLogix for Kollmorgen Automation Suite (KAS)

This application note has two parts. Chapter 1 is an in-depth, step-by-step “how to” of adding Ethernet/IP communications to a KAS project as well as adding KAS support to a new or existing RSLogix5000 project. Chapter 2 is a hands-on quick start guide that explains how to upload the KAS EIP sample projects (.kas & .ACD files) in order to demonstrate correct setup / working Ethernet/IP communications.

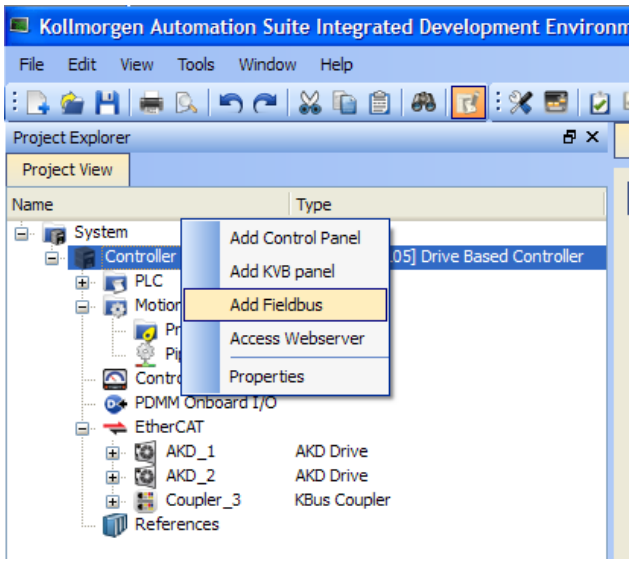
Chapter 1: Adding KAS Support to a New or Existing RSLogix5000 Project



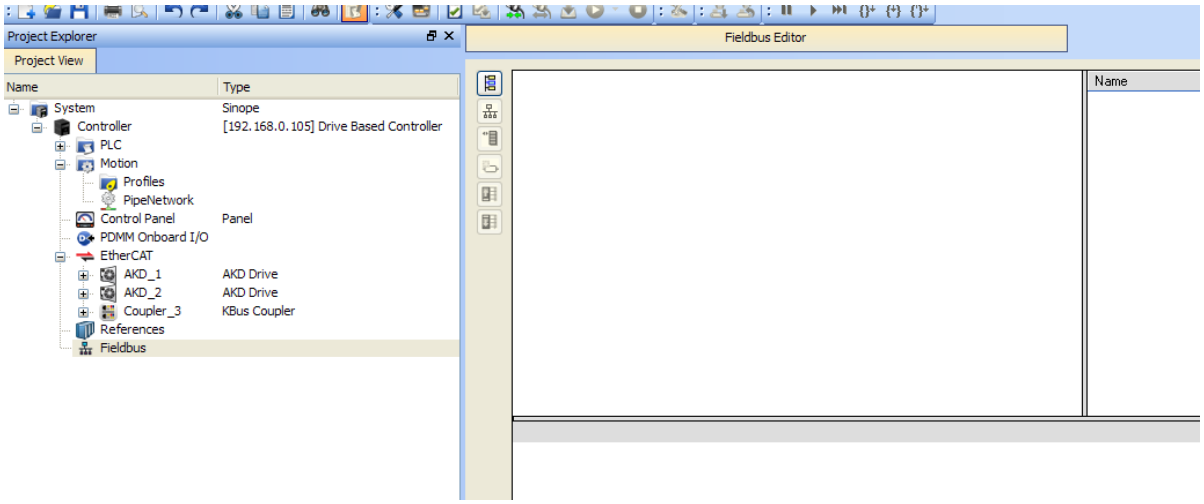
PDMM 2-axis demo (p/n KAS-DEMO-100) with CompactLogix L32E controller

Part 1 – KAS Fieldbus Configuration

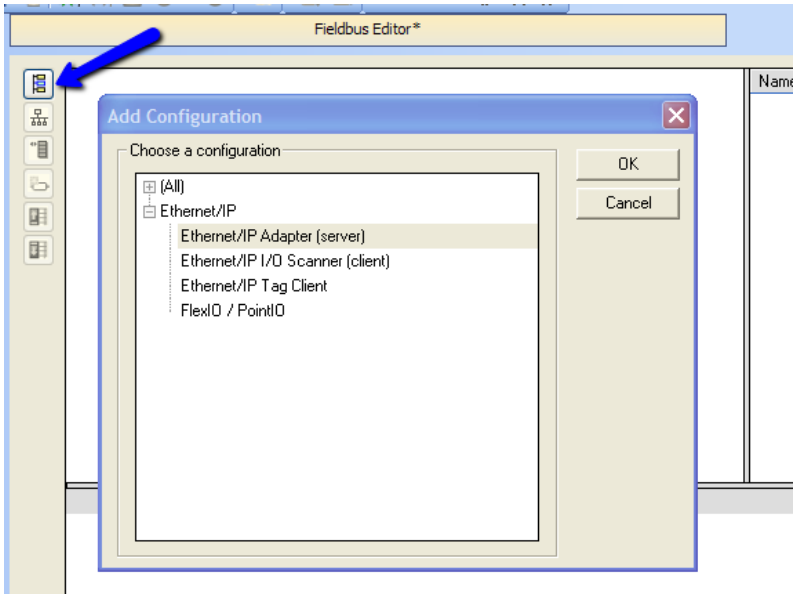
1. Start KAS IDE and open the project with which you want to use EtherNet/IP. Note this example uses the PDMM Controller.
2. Right click on the Controller in the Project Explorer and select Add Fieldbus.



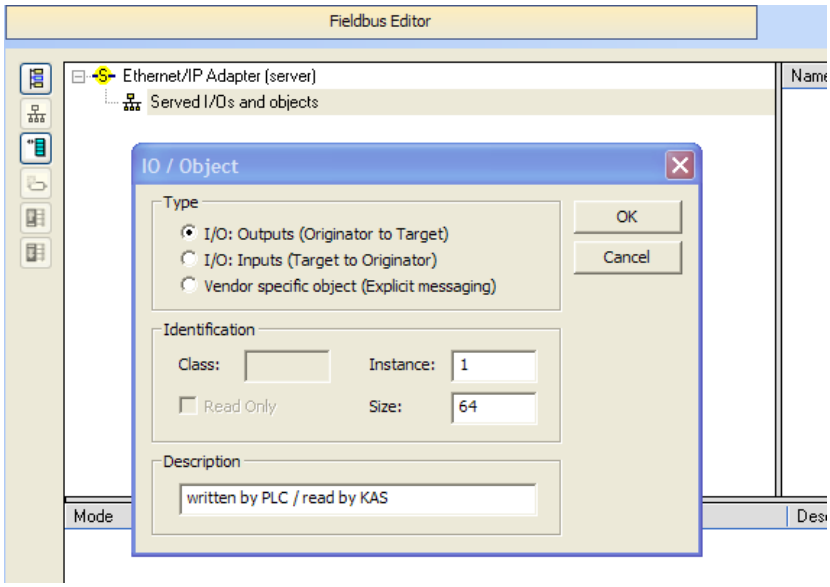
3. Fieldbus will appear in the Project Explorer tree and the Fieldbus Editor screen will display.



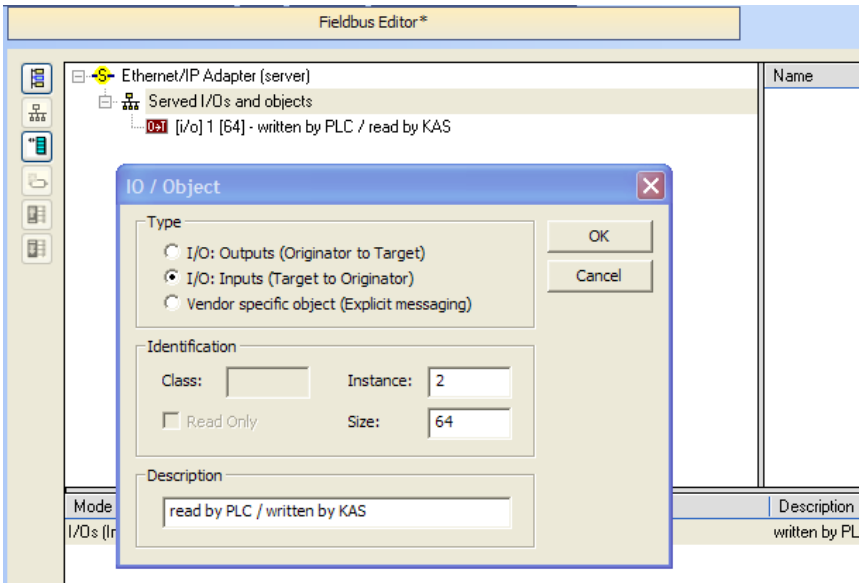
4. Click on the Insert Network button  . Select the Ethernet/IP Adapter (server) configuration and click OK.



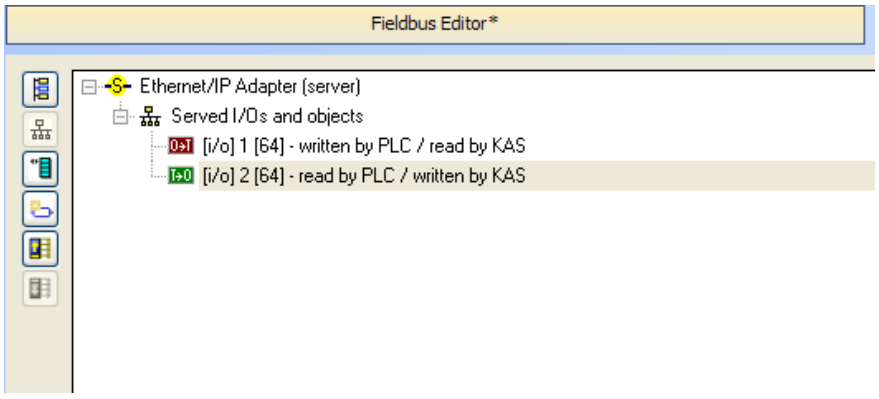
5. Click on the Insert Slave button  to add an I/O Object. Select Type: I/O Outputs (Originator to Target). Set Identification Instance to 1 and Size to 64 (bytes). Suggested Description: “written by PLC / read by KAS.” Click OK.



6. Click on the Insert Slave button  to add an I/O Object. Select Type: I/O Inputs (Target to Originator). Set Identification Instance to 2 and Size to 64 (bytes). Suggested Description: "read by PLC / written by KAS." Click OK.



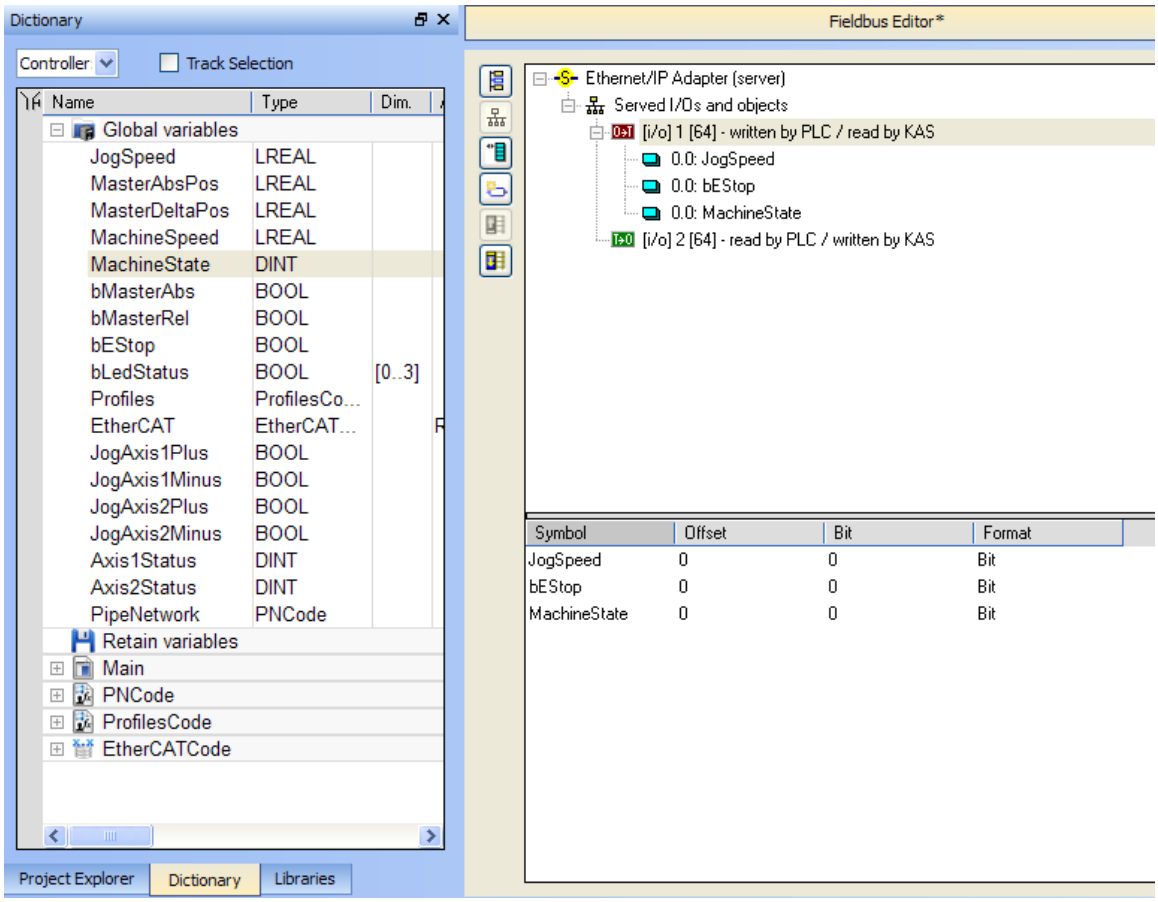
7. The I/O Assemblies for Implicit Messaging (Assembly Messaging) should now be configured and display in the Fieldbus Editor.



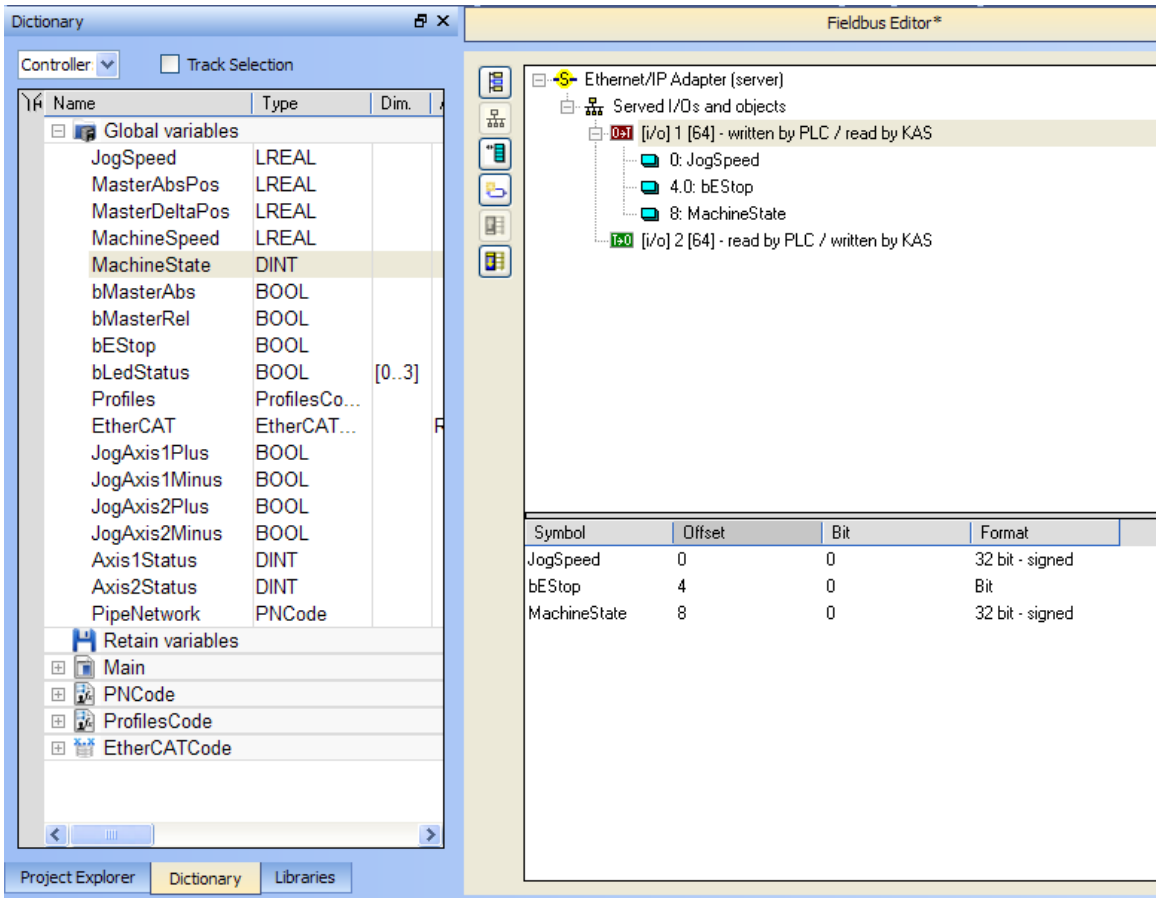
Note: In the KAS I/O Object definition, the Assembly size was set to 64 bytes or 64 x 8-bit. In the Rockwell I/O Object definition, notice the Assembly size is set to 16 x 32-bit. The Assemblies are the same size: 512 bits total. As is typically the case when integrating two manufacturer's products, each manufacturer has their own way of defining things. It is important to be mindful of this difference in definition.

The next step is to add user variables to be passed back and forth between the PLC and KAS. In this example, Boolean (BOOL), double integer (DINT) and long real (LREAL) type variables are shown. The user is responsible for dimensioning these.

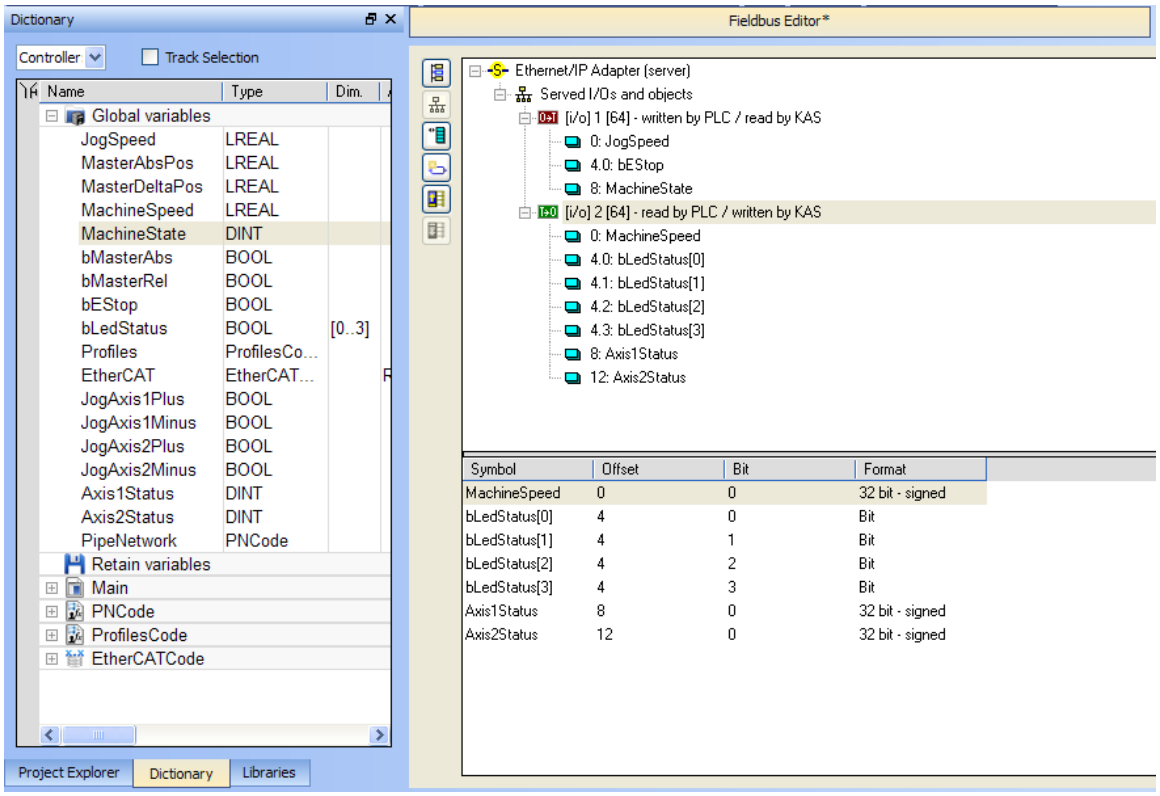
8. Highlight “[i/o] 1 [64] – written by PLC / read by KAS” in the main window. Drag variables from the Dictionary into the lower window with the columns labeled “Symbol, Offset, Bit, Format.” Notice when the variables are initially dropped into the I/O table, they are all assigned address (Offset) 0 and are Bit Format. The user must dimension the variables to the correct memory location.



In KAS, each variable will have an offset of 4 bytes from the previous variable because in RSLogix all memory locations are 32-bit.



9. Highlight “[i/o] 2 [64] – read by PLC / written by KAS” in the main window. Drag variables from the Dictionary into the lower window with the columns labeled “Symbol, Offset, Bit, Format.” The user must dimension these variables as well. Since a Boolean variable is only 1 bit, 32 Boolean variables are dimensioned into one Offset (memory location). See the example of bLedStatus in the screen shot below. For this Boolean variable, the Offset is always 4, but the Bit value changes for each bit of the variable.

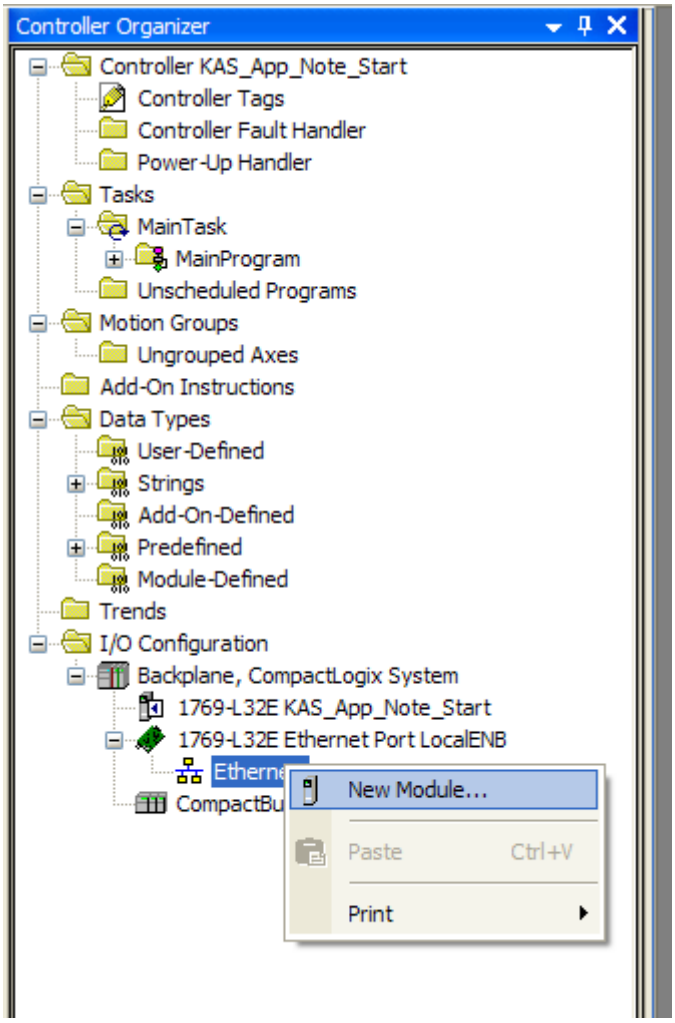


10. Now the KAS Ethernet/IP configuration is complete.

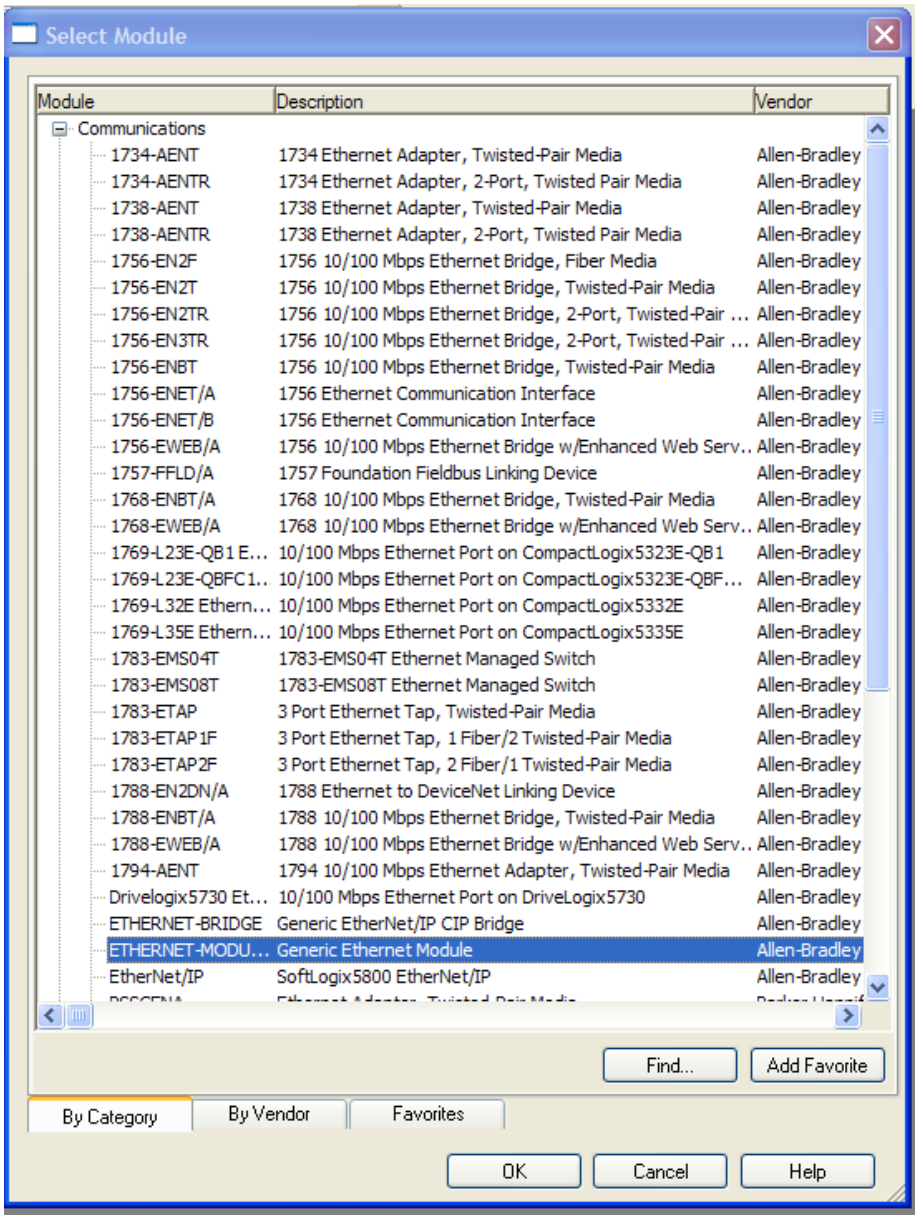
Part 2 – RSLogix5000 Configuration

11. Start RSLogix5000 and open the project with which you want to use KAS.

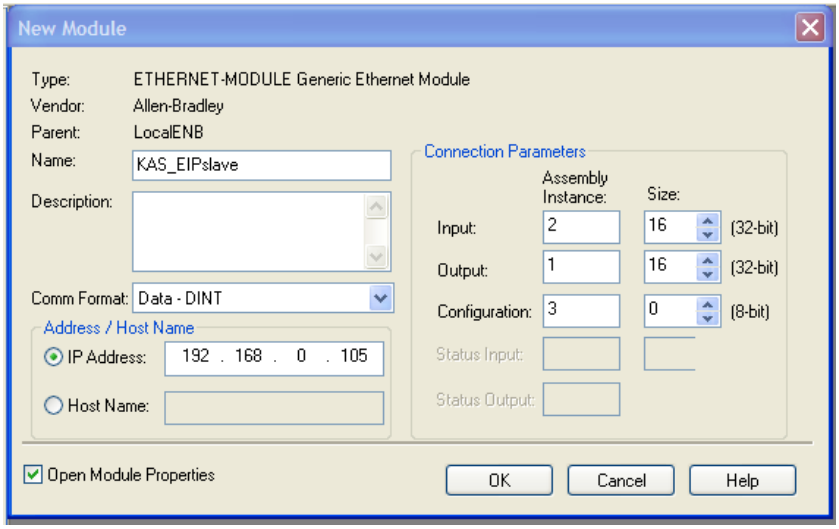
12. Right click on the Ethernet port in the I/O Configuration and select “New Module…”



13. Select “ETHERNET-MODULE” under Communications and click OK.

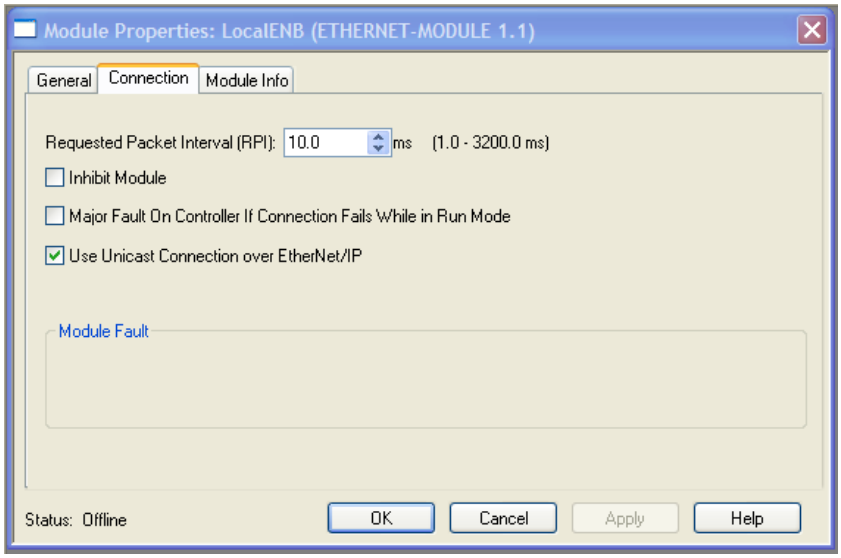


14. Enter the settings for the new module as described below, verify the “Open Module Properties” checkbox is checked and click OK.

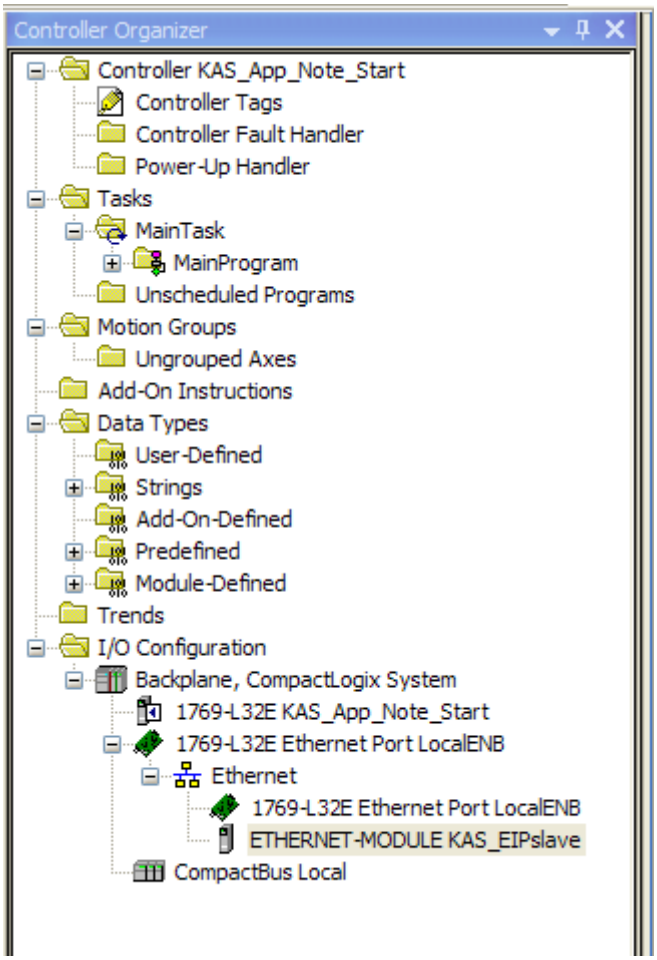


15. The “New Module” window displays as a “Module Properties: Local ENB” window with the Connection tab selected. Set the Requested Packet Interval (RPI) value to 10.0ms.

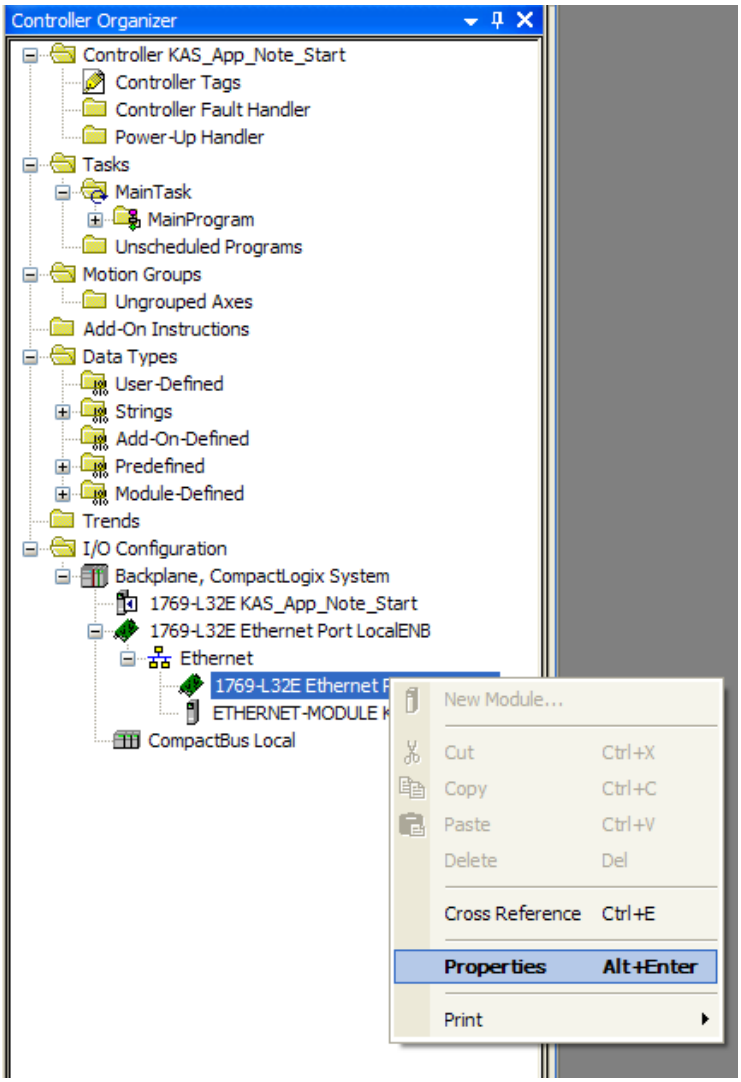
Note: Depending on the amount of data being transferred between the two controllers, the RPI may need to be reduced to 20.0ms. If option “Use Unicast Connection over EtherNet/IP” is visible, make sure it is checked. Click OK.



16. The KAS controller should now be configured and will show up under the Ethernet Port.



17. Make sure the Ethernet port for the Rockwell controller is setup with a compatible IP address on the same subnet as the KAS controller IP address. Right click on “Ethernet Port LocalENB” and select Properties.

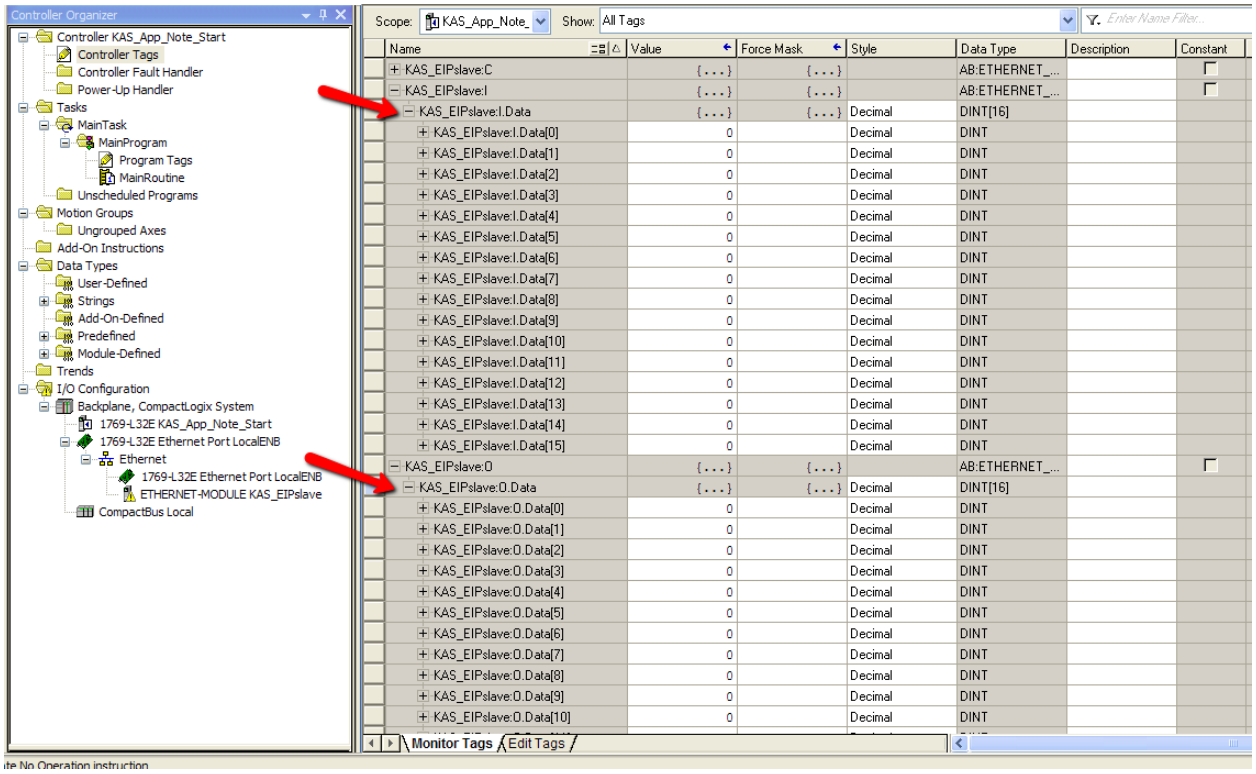


Verify the Rockwell controller IP address is on the same subnet as the KAS controller. In this example, 192.168.0.xxx is correct.

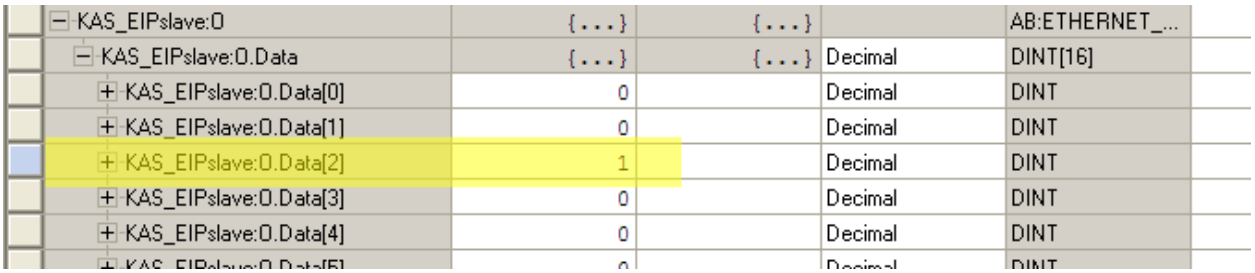
The RSLogix5000 configuration is complete.

Part 3 – Utilizing Ethernet/IP Communications

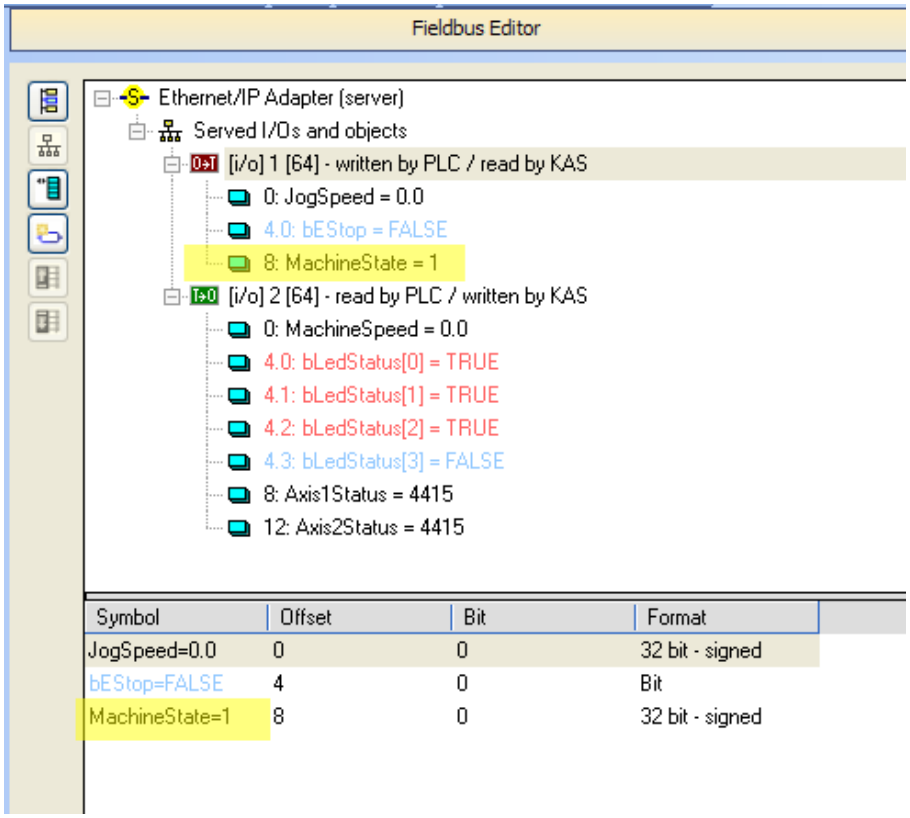
18. To monitor the Ethernet/IP communications, click on Controller Tags in the Controller Organizer tree in RSLogix5000. KAS_EIPslave:I data is the Input Assembly "[i/o] 2 [64] – read by PLC / written by KAS." KAS_EIPslave:O is the Output Assembly "[i/o] 1 [64] – written by PLC / read by KAS."



19. To edit the Output Assembly data, simply click in the Value field of the memory location to edit. For example, the variable MachineState is Memory Location [2] in the Output Assembly (KAS_EIPslave:O.Data[2]). Keep in mind the note regarding how the two manufacturers define the Assemblies. In KAS IDE, MachineState is Offset 8 [bytes]. Divide 8 by 4 (since Rockwell defines the memory locations as 4 bytes or 32-bits each) to get memory location 2, KAS_EIPslave:O.Data[2].

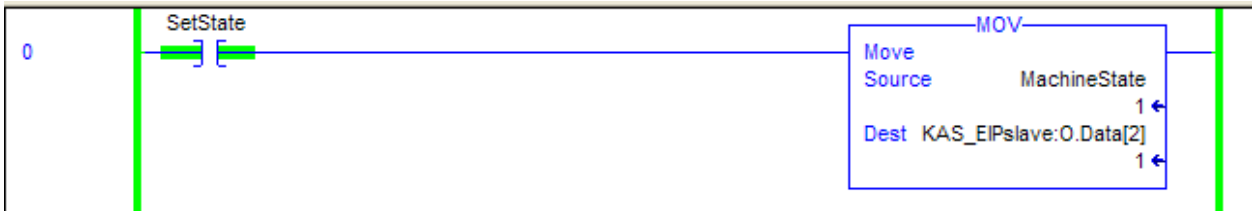


KAS_EIPslave:O	{...}	{...}		AB:ETHERNET_...
KAS_EIPslave:O.Data	{...}	{...}	Decimal	DINT[16]
KAS_EIPslave:O.Data[0]	0		Decimal	DINT
KAS_EIPslave:O.Data[1]	0		Decimal	DINT
KAS_EIPslave:O.Data[2]	1		Decimal	DINT
KAS_EIPslave:O.Data[3]	0		Decimal	DINT
KAS_EIPslave:O.Data[4]	0		Decimal	DINT

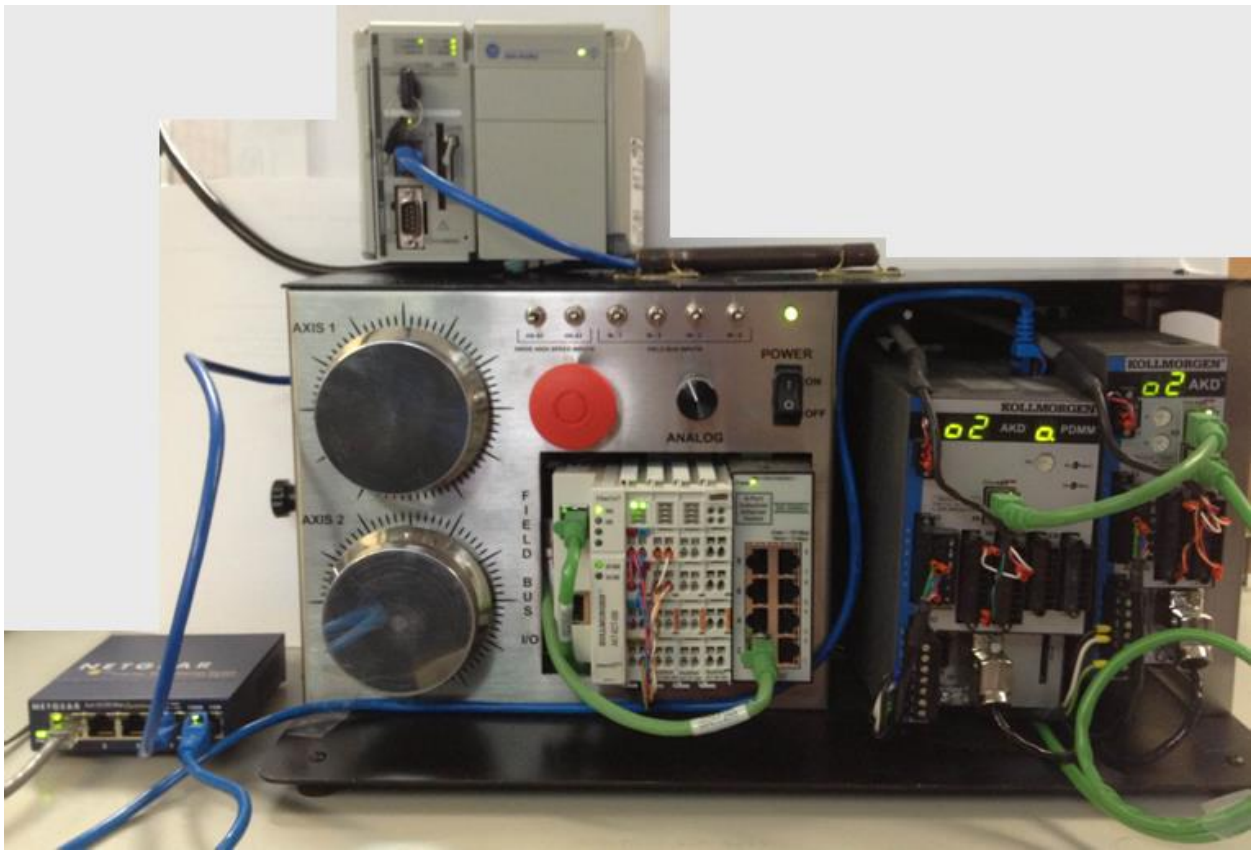


As shown – when the value is changed to a 1 in RSLogix5000, it automatically updates the Fieldbus Editor in KAS IDE.

Alternatively, add code to the RSLogix5000 ladder to change the value of the KAS_EIPslave:O.Data[2] from the ladder.



Chapter 2: Quick Start with the KAS EIP Sample Projects



The primary purpose of the sample projects is to demonstrate the correct setup of the EtherNet/IP communication channel between a PDMM and a CompactLogix L32E controller. RSLogix_KAS_SampleProject.ACD for RSLogix5000 paired with KAS_EIP_SampleProject.kas for KAS IDE demonstrates the correct setup of each controller.

The sample projects can help you learn:

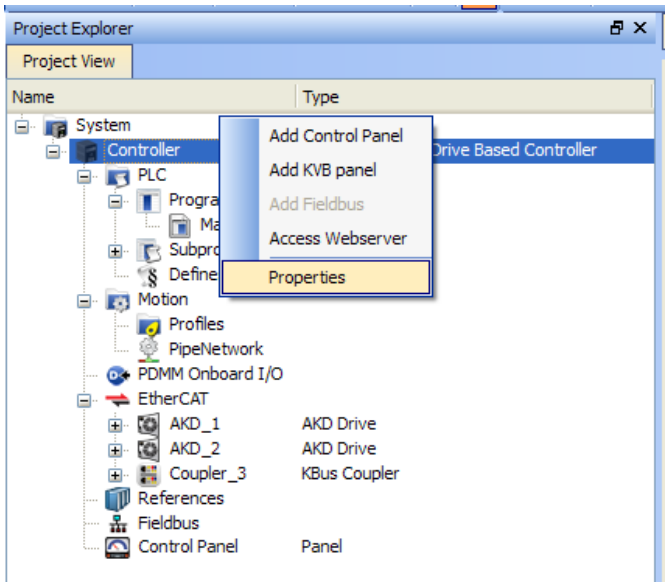
- How to setup cyclic Ethernet/IP communication in KAS IDE and RSLogix5000
- How to start motion from the CompactLogix controller
- How to change motion variables (such as machine acceleration and deceleration)
- How to read/write individual AKD axis drive parameters (such as current limits)
- How to read drive status (Faulted, Enabled, Disabled, etc.)

Part 1 — Setup Controllers for Cyclic EtherNet/IP Communications

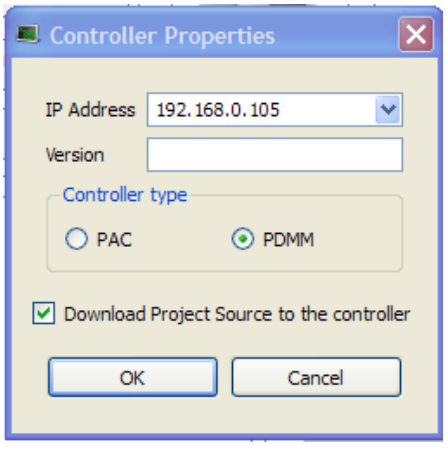
Note: These sample projects were designed for the KAS PDMM 2-axis demo kit, p/n KAS-DEMO-100.

1. Start KAS IDE and open the file KAS_EIP_SampleProject.kas.

2. You will most likely need to update the Controller Properties to match your specific installation. Right click on Controller at the top of the tree and select “Properties.”



3. Update the IP Address in order to match your specific hardware setup and then click OK to close the controller properties window.



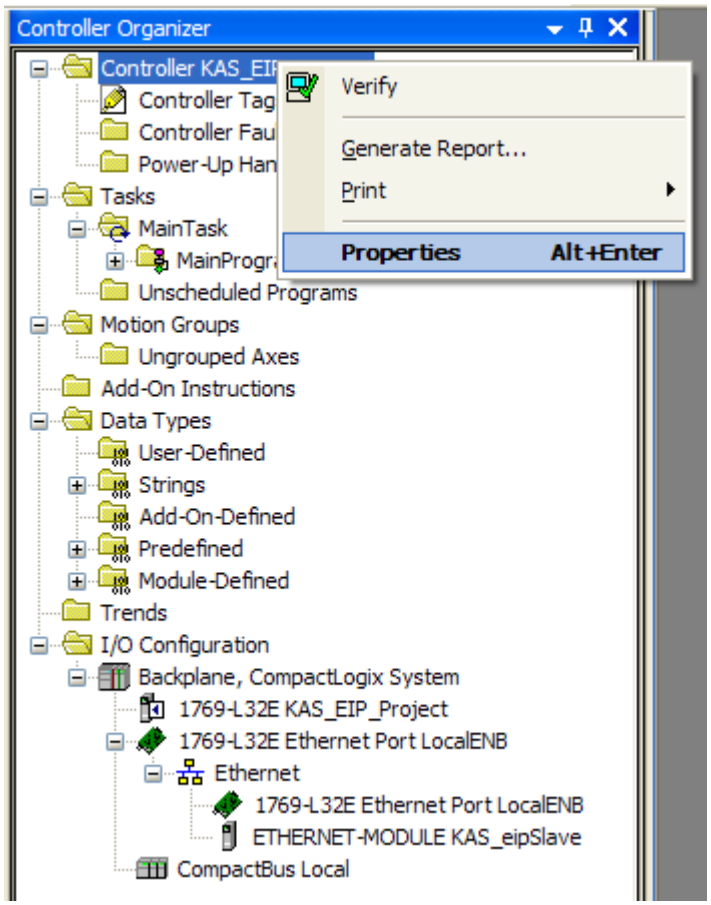
4. Next, open the EtherCAT Devices editor by double-clicking on EtherCAT in the tree.

5. Click the Scan Devices button on the far right. Verify that AKD_1 is mapped to AXIS1 and AKD_2 is mapped to AXIS2.

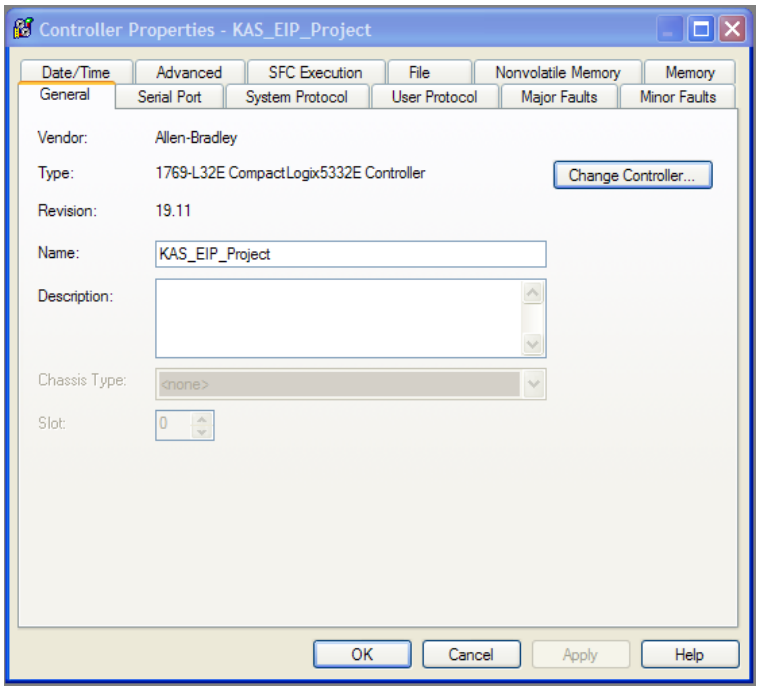
6. Once you have updated all of the configuration settings to match your specific hardware setup, you can compile and download  the program to the PDMM controller and then Run  the project.

7. Start RSLogix5000 and open the file RSLogix_KAS_SampleProject.ACD.

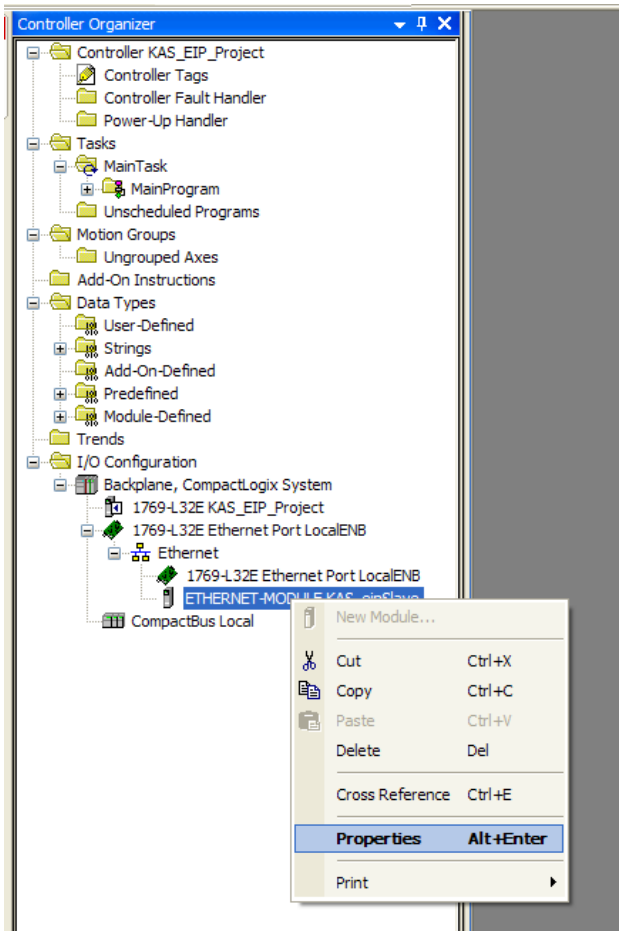
8. You will most likely need to update the controller properties to match your specific installation. Right click on the controller ("KAS_EIP_Project") at the top of the tree and select "Properties."



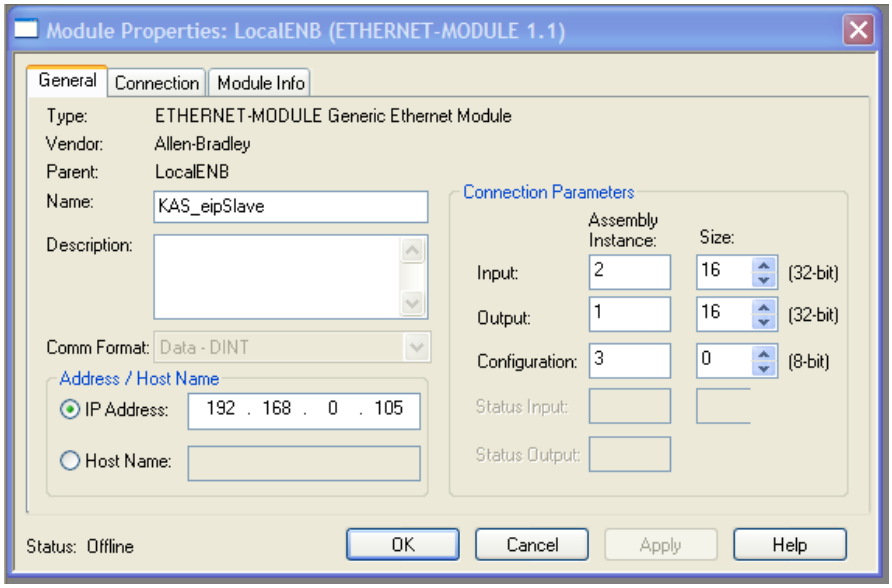
9. Update any controller properties in order for the controller to match your specific hardware setup, most notably any communication settings and/or the controller type, and then click Apply and close the controller properties window.



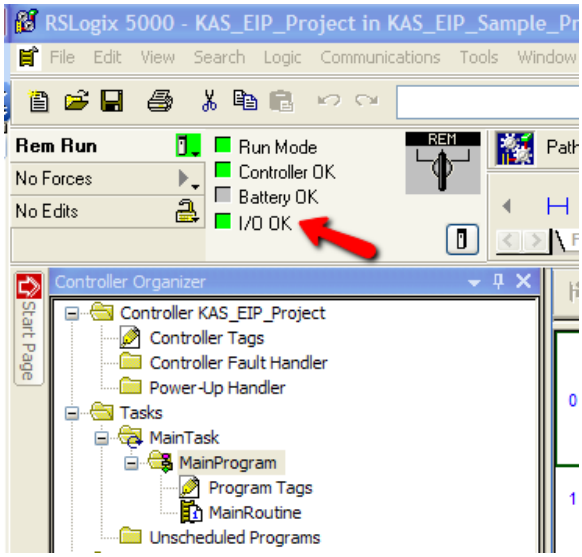
10. Next, open the Ethernet-Module setup for the axis communications by right clicking on “ETHERNET-MODULE KAS_eipSlave” under the Ethernet port in the I/O Configuration tree.



11. Update the IP Address in order to match your specific hardware setup and then click OK to close Module Properties.

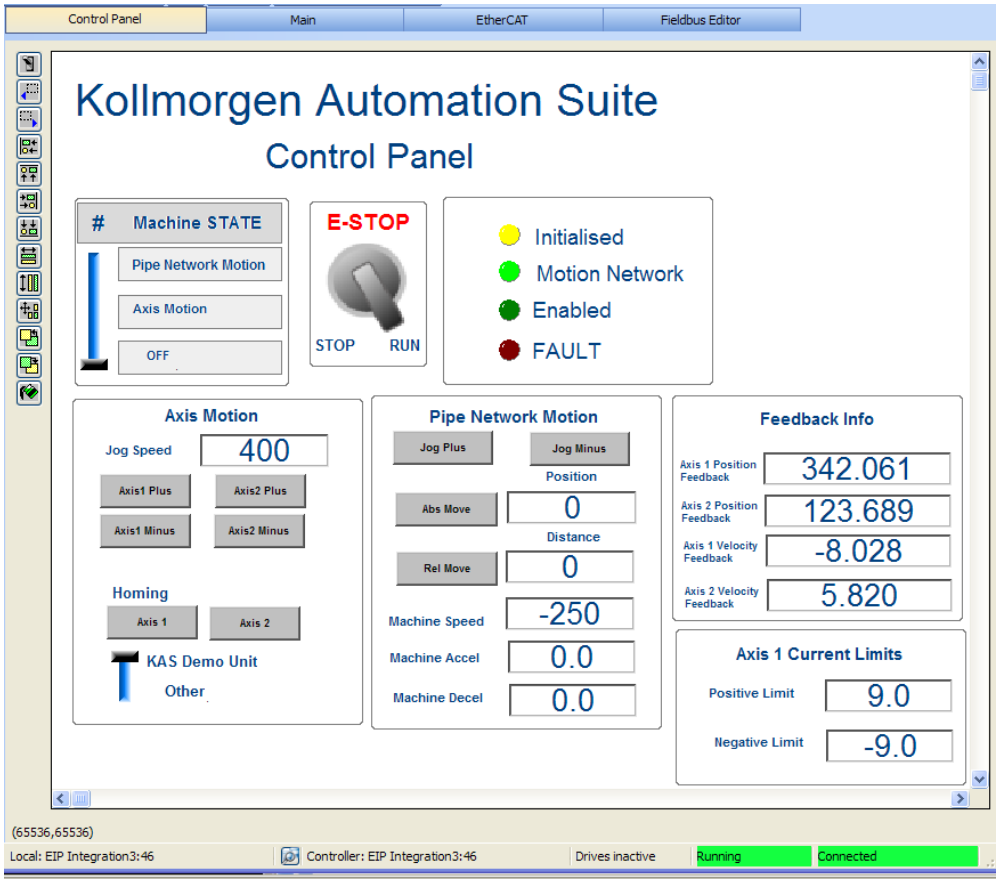


12. Once you had updated all of the configuration settings to match your specific hardware setup, you can download the program to the L32E controller and put the project in Run Mode. You should see the “I/O OK” LED show solid Green as pictured below.



Note: If a blinking green LED with the message “I/O Not Responding” displays, it is likely the KAS project is not running. Go to KAS IDE and click Run button  .

13. After both projects are running, you should see the Control Panel in the KAS IDE updating.



WARNING! *If you try to run the KAS project while the RSLogix5000 project is not running, you will encounter issues. All the variables that are mapped to be written by the PLC (listed under “O→T [i/o] 1 [64] – written by PLC”), cannot be changed / forced to a new value within the KAS IDE. They must be changed by PLC. If PLC is not running, you will not be able to change these variables.*

Explanation of the Sample Projects

The KAS sample project is a 2-axis structured text, Pipe Network program. It is based on the standard application template available in the KAS IDE. The program execution is dependent on the machine state [program variable MachineState – written by AB controller]. There are three possible machine states; OFF is 0 (default). Setting MachineState to 1 enables individual axis motion (jog plus and minus). Setting MachineState to 2 enables Pipe Network / Master axis motion (jogging or indexing moves).

For individual axis motion, the jog speed is a program variable [JogSpeed] written by the AB controller. To start clockwise (plus) or counter-clockwise (negative) motion for each axis a bit must be set from the AB controller [Boolean program variables JogAxis1Plus, JogAxis1Minus, JogAxis2Plus, JogAxis2Minus]. The Homing Slider bar is used to select the type of homing routine to execute. If the KAS Demo Unit hardware (p/n KAS-DEMO-100) is present, make this selection. When the Axis 1 button is clicked, the program physically homes the axis, moving the motor shaft to the home switch. If the KAS Demo Unit hardware is not present / the KAS Simulator is being used, select Other. If the Simulator is being used, when the Axis 1 button is clicked, the virtual motor is homed to “top-dead-center” position. If a real motor is being used, when the Axis 1 button is clicked the motor is homed to absolute zero.

For Pipe Network Motion, the machine speed, acceleration, and deceleration as well as relative move distance or absolute target position are written by the AB controller [program variables MachineSpeed, MachineAccel, MachineDecel, MasterAbsPos, MasterDeltaPos respectively].

For all modes of operation, velocity and position feedback data is read by the AB controller. Finally, the axis 1 current limits are an example of an AKD drive parameters (IL.LIMITP and IL.LIMITN) that are written by the AB controller.

The RSLogix_KAS_SampleProject.ACD is a very simplistic ladder that shows reading and writing values in the objects defined by the Input and Output Assemblies (Ethernet/IP cyclic communications channel). In this case, the program flow and control is defined by the structured text program in the PDMM controller. The RSLogix project is simply reading and setting variables in the PDMM program.

Appendix

Data Types KAS vs. Rockwell

BOOL	Boolean (Bit)	FALSE or TRUE - stored in 1 byte
SINT	Small signed integer in 8 bits	-128 to +127
INT	Signed integer in 16 bits	-32768 to +32767
DINT	Signed double precision integer in 32 bits	-2147483648 to +2147483647
REAL	Single precision floating point stored in 32 bits	-3.4E38 to 3.4E38 and -3.4E-38 to 3.4E-38 (6 to 7 significant digits of accuracy)
LREAL**	Double precision floating point stored in 32 bits	-1.7E308 to 1.7E308 and -1.7E-308 to 1.7E-308 (14 to 15 significant digits of accuracy)

KAS IDE supports all data types listed. RSLogix5000 does not have the LREAL data type.

Note: In the examples above, the Generic Ethernet Module (KAS I/O Module in RSLogix5000) Communications Format is defined as DINT (32 bit signed integers) so only integer values are passed.