IND360
Dynamic Application EtherNet/IP Note

METTLER TOLEDO

FCC Notice

This device complies with Part 15 of the FCC Rules and the Radio Interference Requirements of the Canadian Department of Communications. Operation is subject to the following conditions: (1) this device may not cause harmful interference, and (2) this device must accept any interference received, including interference that may cause undesired operation.

This equipment has been tested and found to comply with the limits for a Class A digital device, pursuant to Part 15 of FCC Rules. These limits are designed to provide reasonable protection against harmful interference when the equipment is operated in a commercial environment. This equipment generates, uses, and can radiate radio frequency energy and, if not installed and used in accordance with the instruction manual, may cause harmful interference to radio communications. Operation of this equipment in a residential area is likely to cause harmful interference in which case the user will be required to correct the interference at his or her expense.

- Declaration of Conformity is available at
  http://glo.mt.com/global/en/home/search/compliance.html/compliance/.

Contents

This Engineering Note demonstrates the integration of the METTLER TOLEDO IND360's Dynamic Application with an EtherNet/IP PLC. Go to www.mt.com/ind-ind360-downloads to download all the necessary files and documents.

NOTICE
NOTE: THE CONFIGURATION USED IN THIS SAMPLE CODE IS BASED ON THE DEFAULT SETTINGS：

Rockwell Studio5000:   Version 24
PLC: 1769-L30ER
SAI DATA FORMAT: 2-BLOCK FORMAT (DEFAULT), 8-BLOCK FORMAT
IND360 IP ADDRESS: 192.168.0.2
EDS FILE: MT_IND360_EIP_V1.1_20200728
IND360 DEVICE FIRMWARE VERSION: V2.01.0000

It is recommended to integrate one IND360 into the PLC EtherNet/IP network and go through the sample code to understand the functionality of each Add-On Instruction (AOI).

1   Setup of Project Development Environment

1.1.   Confirm EDS Installation

This sample code project utilizes an EDS file for the IND360. These files can be found on www.mt.com/ind-ind360-downloads.

To confirm installation of IND360 AOP file:

1. In any Studio 5000 project, right click on **Ethernet** within the I/O Configuration folder in the controller organizer.

2. Select **New Module...**.
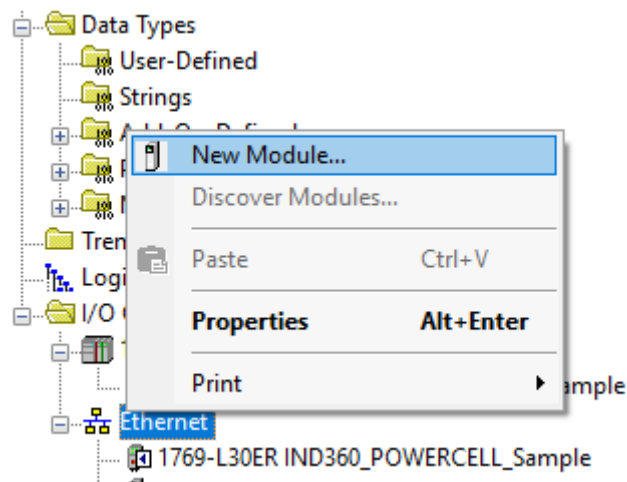
Figure 1-1: Try to add a new module to confirm EDS is installed

3. Search IND360

Figure 1-2: Search for IND360

If the EDS is installed, there should be an option for MT-IND360. If the search returns no results, follow these steps to install the AOP:

1. Go to the IND360 download page: www.mt.com/ind-IND360-downloads

2. Click the EDS file to begin the download.

3. Once the download is complete, unzip the folder

4. Use the EDS installation tool in Studio5000 to install the EDS.

Figure 1-3: Use the EDS Hardware Installation Tool to complete installation

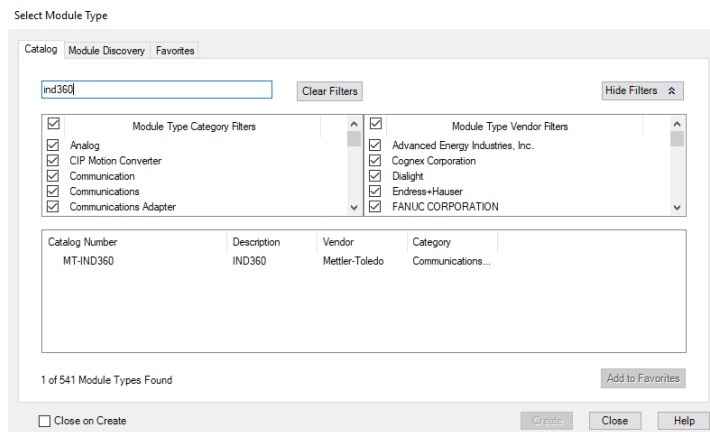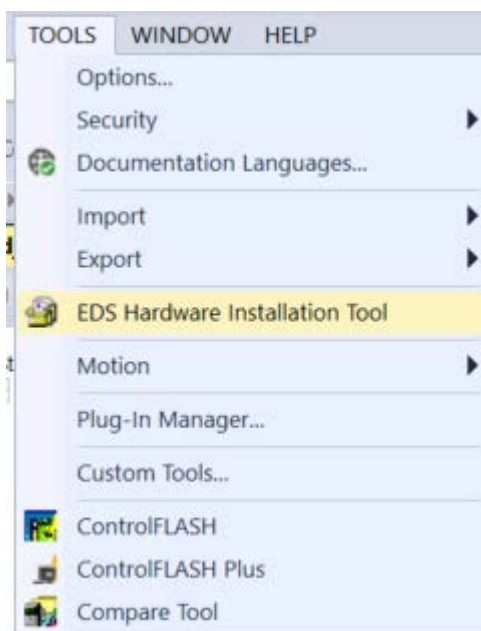## 1.2. Import Example as a New Project

To import the examples, Studio5000 V24 or above is required.

1. To import the project to Studio5000, click **File-> Open.**
2. Select the .ACD file and click open. The project will load.

Figure 1-4: Import Project

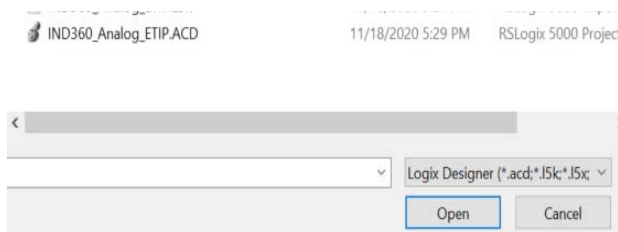## 1.3. Import an Example to an Existing Project

1. Add an MT-IND360 to the I/O Configuration in the existing project. See the first steps of Section 3 for more information on how to complete this. Using the name **IND360** and the IP Address **192.168.0.2** will require no changes to the sample code. If a different name or IP address is required, steps explaining what changes to make are provided below.
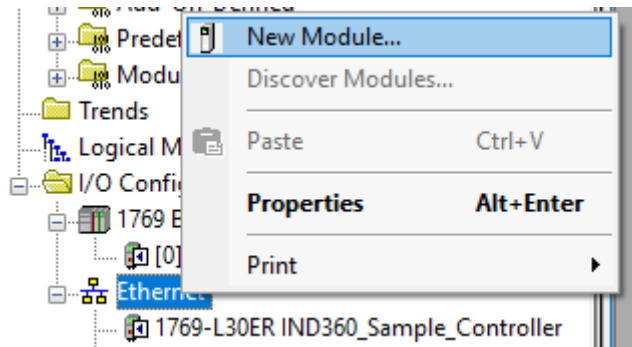
Figure 1-5: Add IND360 to the existing project

2. Copy the Add-On Instructions from the Add-On Instructions folder in the
    Controller Organizer of the sample project and paste in the same location in the
    existing project.
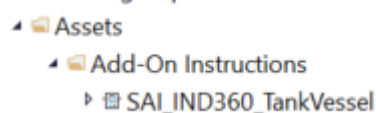
Figure 1-6: Copy/Paste AOIs

3. Copy the controller tags from the sample code project and paste in the controller
    tags of the existing project. Make sure not to copy the **IND360:I** and **IND360:O**
    tags since those are already present in the existing project.

Figure 1-7: Copy/Paste Controller Tags

4. Copy the Main Program local tags from the sample project and paste in the tags
    for the existing project.

Figure 1-8: Copy/Paste Main Program Local Tags

5. Copy the **MT_IND_Application** routine from the sample project and paste in the
    existing project.
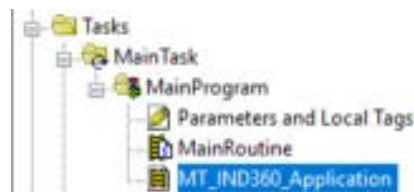
Figure 1-9: Copy/Paste the Routine

6. Make sure something in the existing project calls the MT_IND_Application. Any
    AOIs that automatically monitor weight conditions will not run if nothing calls
    this routine.

METTLER TOLEDO IND360fill/dose Application EtherNet/IP Engineering
Notes

7. If a name other than **IND360** was used as the name of the transmitter in the project, replace every use of **IND360** in the AOI instances with the name given to the transmitter in the project.
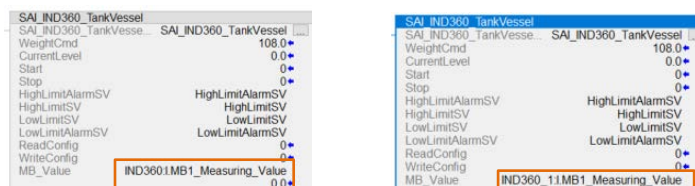
Figure 1-10: Example of name "IND360_1" used in project

1.4. Configure Controller Type

Please note that this is only necessary if using the sample code as the basis for the PLC project. If importing the routine and AOIs into an already existing project, this is unnecessary.

Right-click the project's controller, select **Properties**, and set the controller type.

Figure 1-11: Configure controller type

Download the project to the controller and test.

2   Add-On Instructions (AOI)

Please refer to the **IND360 Dynamic Application Software User's Guide** for a complete description of the application's functionality. This application user manual can be downloaded from the IND360 download page (www.mt.com/ind-ind360-downloads).

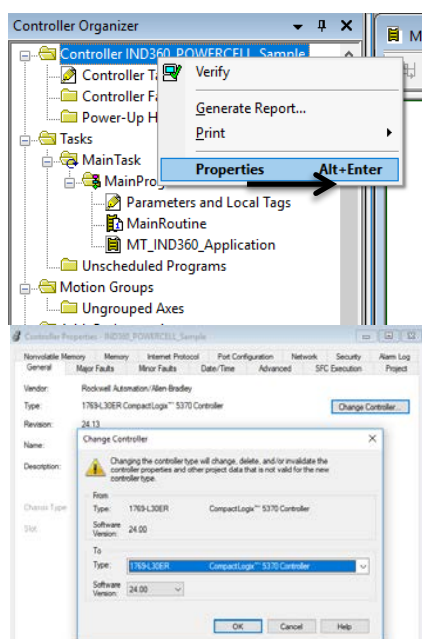## 2.1.   Dynamic Application

This AOI is used to read and write the configuration parameters, control the running state, and view process status of the Dynamic application. In this sample code, the AOI uses Measuring Block 1 from the IND360 (referred to as **IND360:I.MB1** or **IND360:O.MB1** in the controller tags). If the PLC communication is set up as SAI 8 Block format, any Measuring Block (1-7) can be used.

Before using this AOI, make sure the Dynamic application is enabled under the PAC Management setup menu in the IND360.

Please note that this sample code is for reference only. You must thoroughly test any PLC code in your own controller and production environment to make sure it functions as you expect.
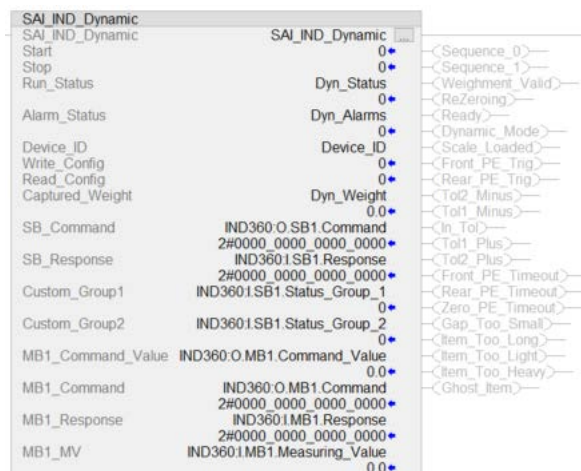
Figure 2-1: SAI_IND360_Dynamic AOI

Parameters	Data Type	Description
Input Parameters		
Start	BOOL	Set = 1 to start the application. Bit will reset to 0 after command is sent.
Stop	BOOL	Set = 1 to stop the application. Bit will reset to 0 after the command is sent.

Parameters	Data Type	Description
Read_Config	BOOL	Trigger this bit to read Dynamic configuration parameters from the IND360. Configuration parameters used in sample code are minimal, but the infrastructure is in place for more commands to be added and triggered by this bit.
Write_Config	BOOL	Trigger this bit to write Dynamic configuration parameters to the IND360. Configuration parameters used in sample code are minimal, but the infrastructure is in place for more commands to be added and triggered by this bit.
MB1_MV	REAL	This should always be set to the MB1_Measuring_Value of the IND360. This will provide measuring data for the AOI.
MB1_Response	REAL	This should always be set to MB1_Response value of the IND360. Once a cyclic command is successfully executed, MB_Response = MB_Command. The AOI uses this information to detect if a command has been executed successfully or if an error has occurred.
Custom_Group1	DINT	This should always be set to the SB1_Status_Group_1 of the IND360. This will provide status information for the Application.
Custom_Group2	DINT	This should always be set to the SB1_Status_Group_2 of the IND360. This will provide alarm information for the Application.
SB_Response	DINT	This should always be set to SB1_Response value of the IND360. Once a cyclic command is successfully executed, SB_Response = SB_Command. The AOI uses this information to detect if a command has been executed successfully or if an error has occurred.
In/Out Parameters		
Device_ID	REAL	Used to read and write the device ID value
Output Parameters		
Sequence_0	BOOL	These two bits work together to count up in binary each time a dynamic weighment is captured. Value rolls over to "00" after "11"
Sequence_1	BOOL	
Weighment_Valid	BOOL	Is set = 1 if the last dynamic weighment was valid
ReZeroing	BOOL	Is set = 1 if in the process of re-zeroing the scale
Ready	BOOL	Is set = 1 if the application is ready to run
Dynamic_Mode	BOOL	Is set = 1 if application is running
Scale_Loaded	BOOL	Is set = 1 if weight is detected on the scale
Front_PE_Trig	BOOL	Is set = 1 if the front photoeye is currently triggered
Rear_PE_Trig	BOOL	Is set = 1 if the rear photoeye is currently triggered
Tol2_Minus	BOOL	Is set = 1 if the dynamic weighment was below the (-)Tol2 setting for checkweighing
Tol1_Minus	BOOL	Is set = 1 if the dynamic weighment was below the (-)Tol1 setting for checkweighing
In_Tol	BOOL	Is set = 1 if the dynamic weighment was within the tolerance settings for checkweighing

Parameters	Data Type	Description
Tol1_Plus	BOOL	Is set = 1 if the dynamic weighment was above the (+)Tol1 setting for checkweighing
Tol2_Plus	BOOL	Is set = 1 if the dynamic weighment was above the (+)Tol2 setting for checkweighing
Front_PE_Timeout	BOOL	Is set = 1 if the front photoeye has been triggered for too long. This could indicate that the photoeye is blocked
Rear_PE_Timeout	BOOL	Is set = 1 if the rear photoeye has been triggered for too long. This could indicate that the photoeye is blocked
Zero_PE_Timeout	BOOL	Is set = 1 if the zero photoeye has been triggered for too long. This could indicate that the photoeye is blocked
Gap_Too_Small	BOOL	Is set = 1 if not enough time elapsed between successive items on the scale based on application settings
Item_Too_Long	BOOL	Is set = 1 if the item is too long for the scale based on application parameters. In dual-photoeye mode, this could indicate the rea photoeye was triggered while the front photoeye was still triggered
Item_Too_Light	BOOL	Is set = 1 if the item was measured to be too light
Item_Too_Heavy	BOOL	Is set = 1 if the item was measured to be too heavy
Ghost_Item	BOOL	Is set = 1 if the rear photoeye is triggered, but the front photoeye is not during dual photoeye mode
Run_Status	DINT	Status of the application. Broken out into Boolean values elsewhere in the AOI
Alarm_Status	BOOL	Status of the application alarms. Broken out into Boolean values elsewhere in the AOI
Captured_Weight	REAL	Captured weight value from the last successful dynamic weighment
MB1_Command_Value	REAL	This should always be set to MB1_Command_Value tag of the IND360. Parameter sent along with the cyclic command to the IND360.
MB1_Command	REAL	This should always be set to MB1_Command tag of the IND360. Value of the last cyclic command sent to the IND360. Once successfully executed, MB1_Response = MB1_Command.
SB_Command	DINT	This should always be set to SB1_Command tag of the IND360. Value of the last status command sent to the IND360. Once successfully executed, SB_Response = SB_Command.

2.2.   Alibi Memory

This AOI can be used to read transactions stored in the alibi memory. Alibi memory is often used in legal for trade weighing applications and is a requirement in some regions of the world.

The Transaction ID of this AOI must first be set to the number of the transaction to be read. Once the Read_Alibi bit is set to 1, the stored data from the specified transaction is stored in the various output tags of this AOI.
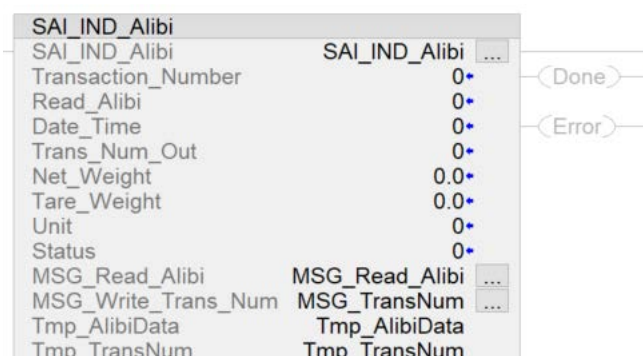
Figure 2-2 SAI_IND_Alibi AOI

Input Parameters	Data Type	Description
Transaction_Number	DINT	Enter the transaction number to be read from alibi memory
Read_Alibi	DINT	Set = 1 to read the alibi data of the transaction number
Output Parameters	Data Type	Description
Trans_Num_Out	DINT	The read transaction number from alibi memory. This can be used as a sort of handshake to confirm the transaction number read matches the transaction number requested.
Date_Time	DINT	IND360 system date and time when the alibi record was written. UTC timestamp
Net_Weight	REAL	The net weight of the scale when the alibi record was written
Tare_Weight	REAL	The tare weight of the scale when the alibi record was written
Unit	SINT	0 = g, 1 = kg, 2 = lb, 3 = t, 4 = ton
Status	SINT	Status when the dynamic weighment was captured
Done	BOOL	Latched high when the alibi read successfully completes. Unlatched when alibi read is triggered again.
Error	BOOL	Latched high if an error occurred and the alibi read could not complete. Unlatched when alibi read is triggered again. Check the errors of the messages for this AOI to troubleshoot
In/Out Parameters	Data Type	Description
MSG_Read_Alibi	Message	**Message Type:** CIP Generic **Service Type:** Get Attribute Single **Class:** 412 (Hex) **Instance:** 1 **Attribute:** 5 (Hex) **Destination Element:** Tmp_AlibiData

		Communication -> Path: Browse for the appropriate IND360
MSG_TransNum	Message	**Message Type:** CIP Generic **Service Type:** Set Attribute Single **Class:** 412 (Hex) **Instance:** 1 **Attribute:** 4 (Hex) **Source Element:** Tmp_TransNum **Source Length:** 4 (Bytes) **Communication -> Path:** Browse for the appropriate IND360
Tmp_AlibiData	SINT[24]	Temporary value to hold all 24 bytes from the alibi record
Tmp_TransNum	DINT	Temporary value to send the transaction number to the CIP message

3 Steps to Add New IND360s

Because EtherNet/IP uses IP addresses to distinguish between devices, when multiple IND360s are networked the default IP address of each terminal must first be modified.

█ Each IND360 must have a different IP address.

1. Click **Communication-> Industrial Ethernet -> IP Address** in the IND360 Advanced Service Mode in order to modify the IP address.

Figure 3-1: IND360 IP Address Menu

2. Add an MT-IND360 to **I / O Configuration-> Ethernet** in Studio5000.
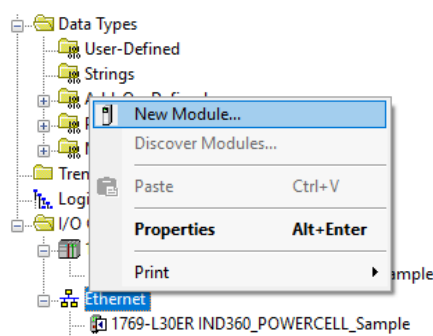

Figure 3-2: Add a device

3. Configure the name and IP address. Each device must have a unique name and IP address. Once the device is configured, click **Change**.
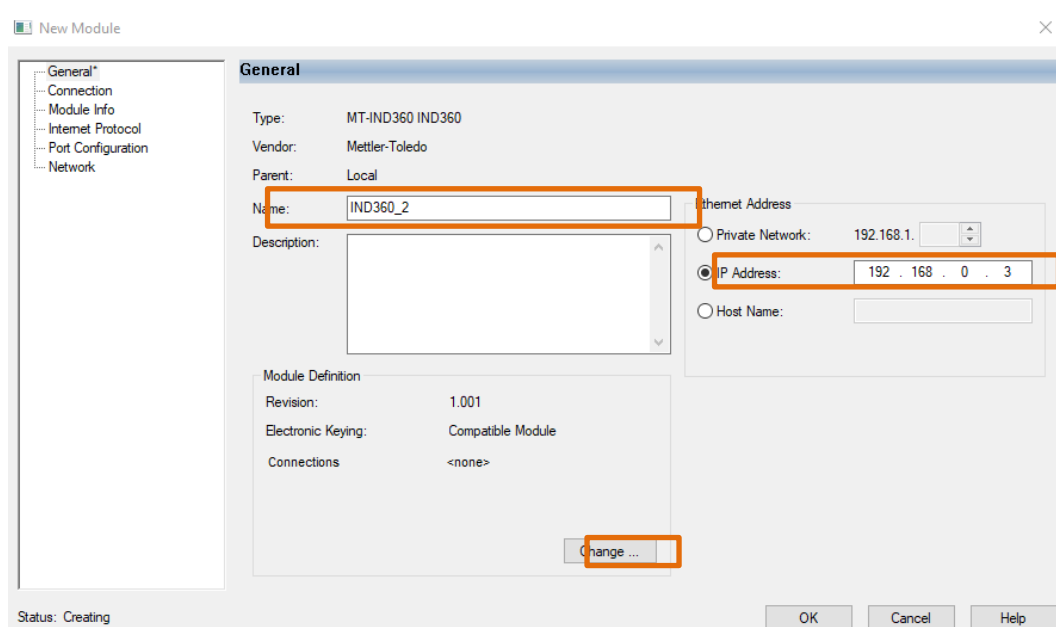

Figure 3-3: Configure name and IP address

4. Select **I/O 2 Block Format** to make the sample code function with minimal changes. Select 8 Block if it is necessary to receive multiple pieces of cyclic data simultaneously. For example, if it is required to read the gross weight, net weight and target weight at one time, 8 Block can easily accomplish this.
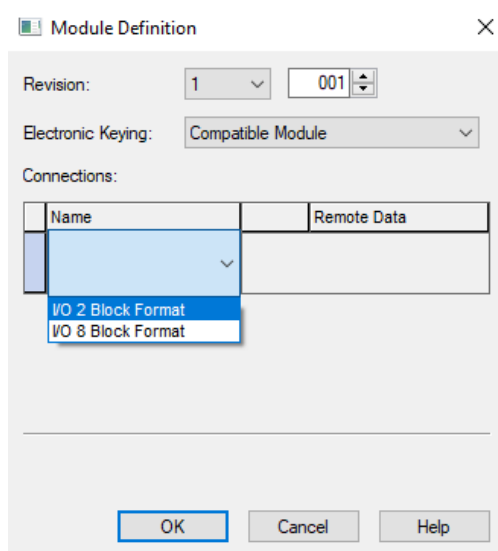
Figure 3-4: Module Definition Configuration

5. Copy the controller tags relating to the sample code and paste in the same location in order to create a duplicate set of tags. Please note that since all tags end with "_1", Studio 5000 will create duplicates that all end with "_2" instead.

Figure 3-5: Copy/Paste Tags to Create Duplicates

6. Copy and paste the Add-On Instructions and configure the instance name along with the input and output parameters. Each device must correspond to a unique instance of the AOI. As shown in the figure below, both devices call the AOI SAI_CheckAlive, but the corresponding instances are SAI_CheckAlive and SAI_CheckAlive_1. Notice that the Heartbeat parameter is also configured with different devices for these two instances. Refer to section 2, **Add-On Instructions (AOI)**, for information on configuring parameters for a particular AOI. Make sure that all tags for the second device for instance now end in "_2" as opposed to "_1" for the first device.
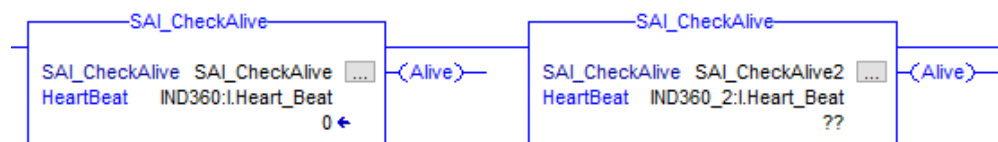

Figure 3-6: Two instances of the SAI_CheckAlive AOI for two IND360s

7. Repeat steps 1 to 6 until all devices are configured.

4 Steps to Use 8 Block Format Instead of 2 Block Format

The 8 Block Format for SAI is extremely powerful for viewing more information simultaneously, compared to the 2 Block Format. The sample code by default is configured for the 2 Block Format, but it is very simple to change the format:

1. Right click on the IND360 in the Controller Organizer.
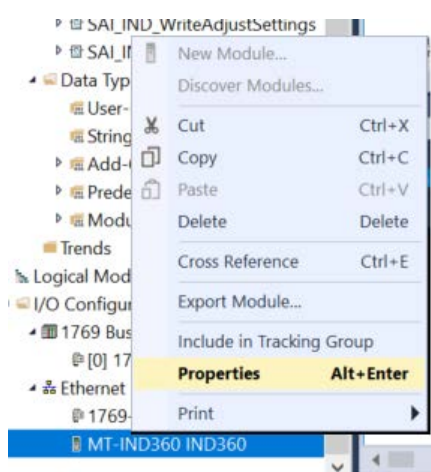
2. Click **Properties**.

Figure 4-1 Select Properties
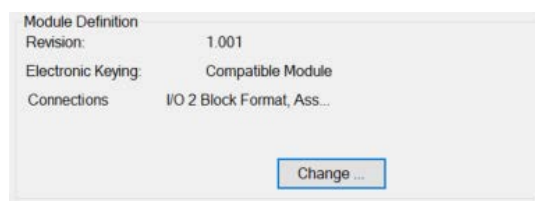
3. Click **Change** under the **Module Definition**.

Figure 4-2 Click "Change"

4. Select the drop-down arrow next to **I/O 2 Block Format** and Select **I/O 8 Block Format**.
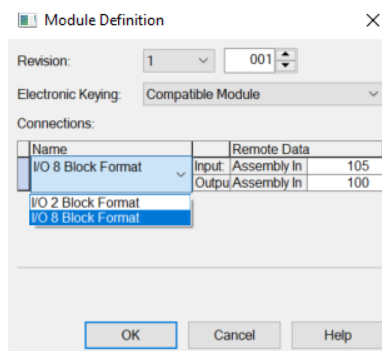
Figure 4-3 Select "I/O 8 Block Format"

At this point, the project has the IND360 configured for the 8 block format. No changes to the AOIs are required since the 8 Block Format simply expands upon the 2 Block format utilized by the AOIs. Now all references to IND360:I.MB1 and IND360:O.MB1in the sample code can be changed to IND360:I.MBx and IND360:O.MBx (where x is a value between 1 and 7) if it is necessary to use a specific Measuring Block for the Dynamic application. The final step is to confirm that the IND360 itself is configured for the 8 block format. This setting can be found in the IND360 setup menu, or via the web interface, at **Communication > Industrial Ethernet > Format**.

5 Frequently Asked Questions

- **Q**: How do I access the parameters in the AOI variables within my PLC program?

    - **A**:   You can use the format "instance_name.parameter" to access parameters in your PLC program. For example, if we create an instance of the SAI_CheckAlive AOI and name the instance "IND360_Comm", we can monitor the alive bit by looking at "IND360_Comm.Alive"
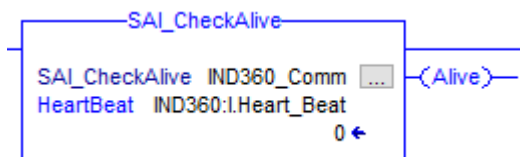

Figure 5-1: SAI_CheckAlive AOI with different instance name

- **Q**: Does my AOI instance always have to match the name of the AOI?

    - **A**:   No. The AOI instance can be named anything as long as the name is unique. They must be unique so that if we are using multiple of the same AOI, we can distinguish between them in the code. See Figure 4-1 for an example of an AOI instance name that does not match the AOI name but is still valid.

- **Q**: How do I know the source of the error in the SAI_IND360_Dynamic AOI?

A: Typical errors in this AOI include:

o **Dynamic PAC is not enabled in the IND360**: Using either the local display or the web interface, access **Application > PAC Management** in the IND360 menu system to enable to application.

- **Q**: An AOI is very close to what I want to do in my PLC logic, but I need to make a few changes. How can I do that?

- **A**: If it necessary to view or modify the logic of an AOI, simply use the Controller Organizer view in Studio 5000. Navigate to Add-On Instructions, expand the AOI you are interested in viewing, and double-click **Logic**. The Organizer view will show the ladder logic used in the AOI, and the logic can be changed as necessary for your particular application.
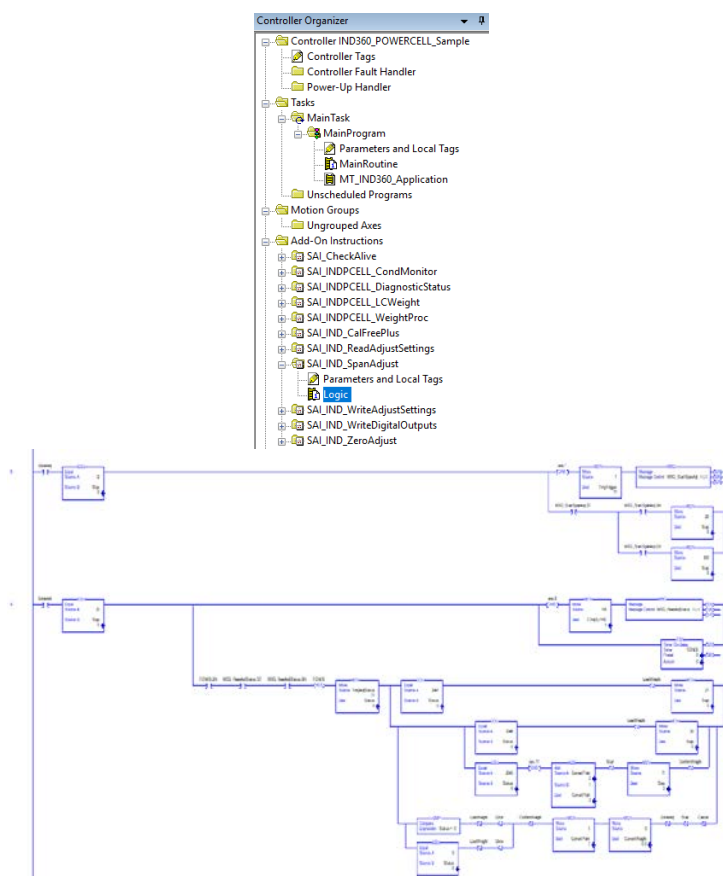

Figure 5-2: Example of AOI ladder logic