

IC70 Process Data Function

January 9th, 2026

This document covers the installation and use of a function for Siemen's TIA Portal software package. This function handles cyclic IO-Link Process Data In from a Banner IC70 via an IO-Link Master to a Siemens PLC. The function covers parsing and display of the IC70 Process Data.

Components

Banner IC70 v16.zal16

Installation Instructions

1. Open a project.
2. Go to the Open Global Library option in the Libraries tab in TIA Portal v16 or greater.



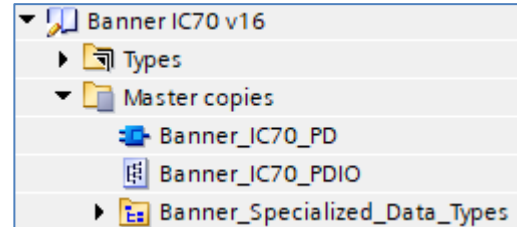
3. Switch the “Files of type” to Compressed libraries. Go to the location of the compressed library.
4. Press the Open button and the library will be uncompressed and opened.
5. The library is now accessible in the Libraries tab in v16 or greater.

Setup of IC70 with a Banner DXMR

1. Go to Device and Networks to configure the DXMR. Add the DXMR if it has yet to be added to the system.
2. Open the IO-Link Generic Devices and select the proper module. The 2/2 byte option has been selected for port 1. Make note of the I and Q address for Slot 2 which represents Port 1. Slot 2 starts at I1 and Q1.

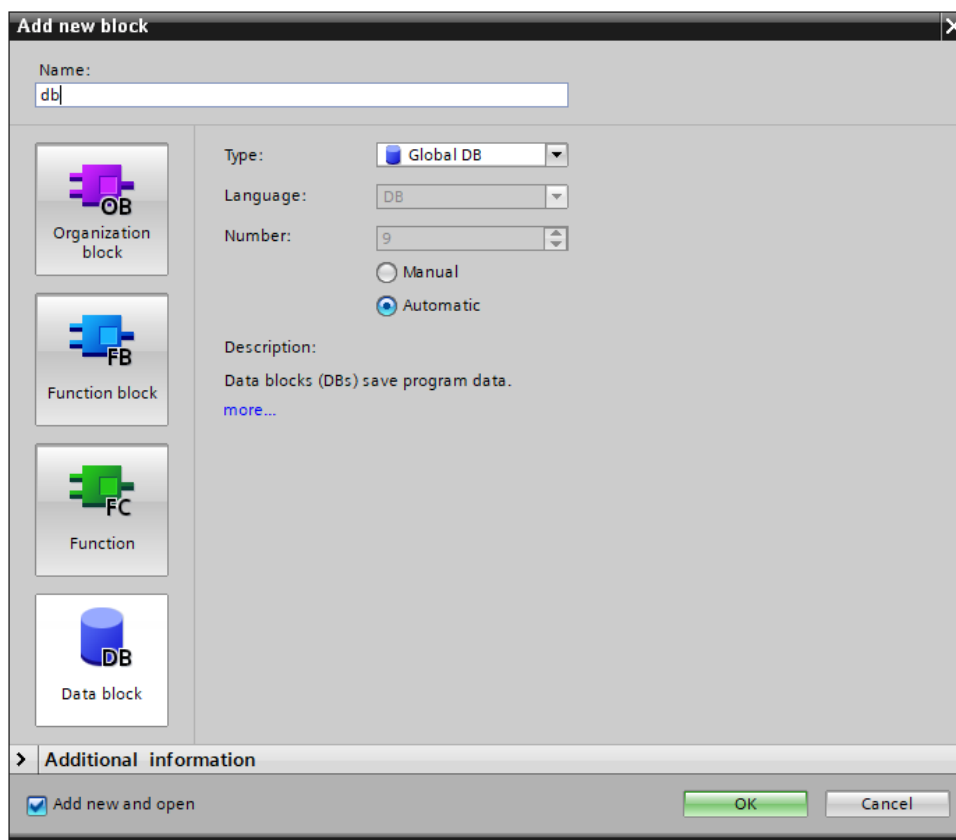
Module	Rack	Slot	I address	Q address
▼ dxm	0	0		
▶ Interface	0	0 X1		
Banner IO-Link Master Info_1	0	1	101...109	
IO-Link In/Out 2/ 2 Byte + Status_1	0	2	1...6	1...16

3. Open the Master Copies folder in the IC70 Library.
4. Drag the Banner_IC70_PDIO to the PLC Data Types area under your PLC.
5. Drag the Banner_IC70_PD to the Program Blocks area.
6. Drag the necessary tags from Banner_Specialized_Data_Types. The tags used in this example is “Banner_2in” and “Banner_2out”. These tags represent the full raw process data along with port status information.
7. Go to PLC Tags. Create four tags. Two of the tags are for the full data structure while the second set represents the raw Process Data from the IO-Link Master. In this example, Tag table_1 was created, the tags “IC70 IOLM1 01 PDI” and “IC70 IOLM1 01 PDO” was created using a Data Type of “Banner_2In” and “Banner_2Out”. This naming convention calls out the type of sensor in question as well as the specific IO-Link Master and port number where the sensor is connected. A different IO-Link Master might be named IOLM2 or IOLM3, for instance, and other specific sensors may be connected to different port numbers. The “I” address found in step 2 is tied to this new tag. The second set of tags use “IC70 IOLM1 01 iRaw” with “Word” and “IC70 IOLM1 01 oRaw” with “Word” . These are the tags that will be used in the Function block.



Name	Data type	Address
▶ IC70 IOLM1 01 PDO	"Banner_2Out"	%Q1.0
IC70 IOLM1 01 oRaw	Word	%QW3
▶ IC70 IOLM1 01 PDI	"Banner_2In"	%I1.0
IC70 IOLM1 01 iRaw	Word	%IW5

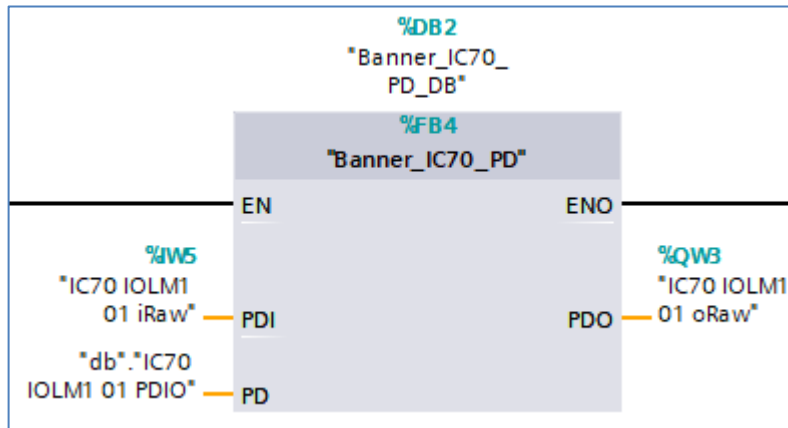
8. Go to Program blocks. Add a new Data block if necessary. In this example the new data block is named "db".



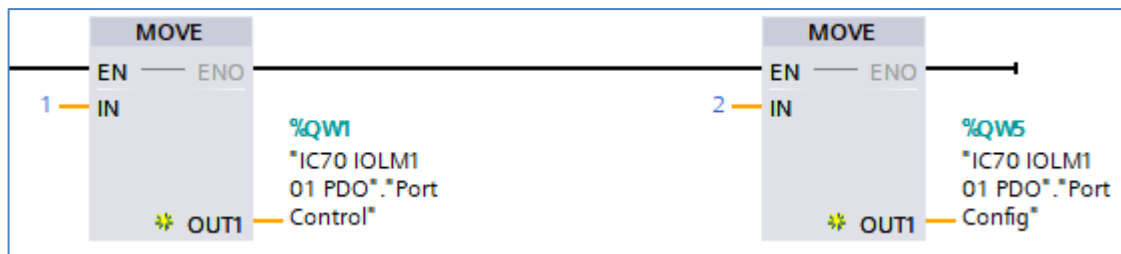
9. In the new data block, create a new tag to represent the parsed Process Data for the IC70. The tag name again calls out the type of sensor, the IO-Link Master, and the port number. Use the data type "Banner_IC70_PDIO" for the new tag.

▼ IC70 IOLM1 01 PDIO	"Banner_IC70_PDIO"
■ ► Channel In	Array[1..16] of Bool
■ ► Channel Out	Array[1..16] of Bool

10. Add the “Banner_IC70_PD” function to an OB ladder. Link the “PDI” and “PDO” to the raw Process Data variable from step 7. Link the “PD” to the parsed Process Data variable from step 9.



8. The final step is to configure the IO-Link output control. This is done by sending a 1 to Port Control and a 2 to Port Config. Both parameters are part of the tag created in step 6 “IC70 IOLM1 01 PDO”.



11. Process Data setup is complete.

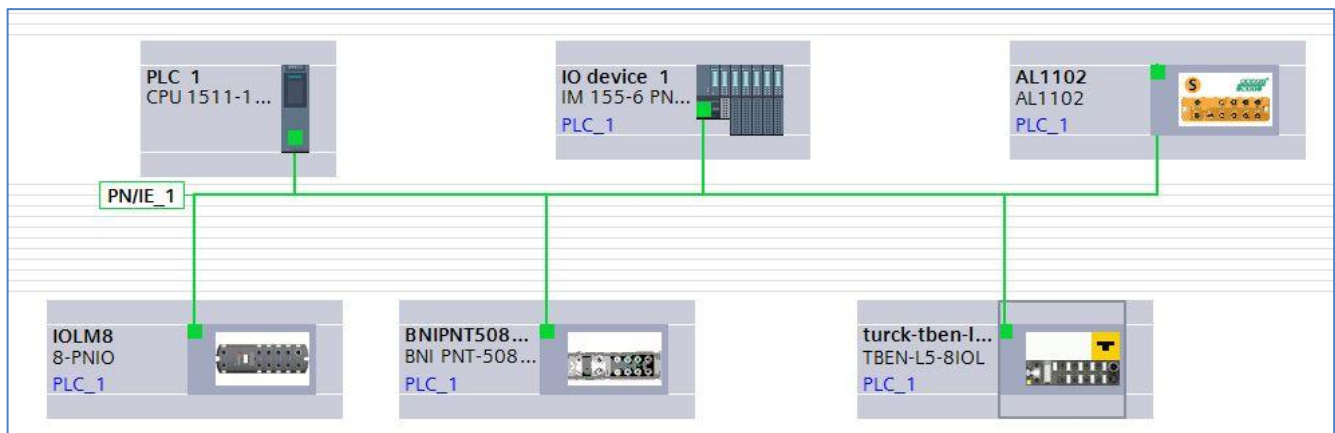
12. Compile and download the configuration to the PLC, then go online. Open the “db” data block and click Monitor all. You should see parsed Hub Process Data In.

▼ IC70 IOLM1 01 PDIO	"Banner_IC70_PDIO"	
■ ► Channel In	Array[1..16] of Bool	
■ ▼ Channel Out	Array[1..16] of Bool	
■ Channel Out[1]	Bool	TRUE
■ Channel Out[2]	Bool	FALSE
■ Channel Out[3]	Bool	FALSE
■ Channel Out[4]	Bool	FALSE
■ Channel Out[5]	Bool	TRUE
■ Channel Out[6]	Bool	FALSE
■ Channel Out[7]	Bool	FALSE
■ Channel Out[8]	Bool	FALSE

▼ IC70 IOLM1 01 PDIO	"Banner_IC70_PDIO"	
■ ▼ Channel In	Array[1..16] of Bool	
■ Channel In[1]	Bool	TRUE
■ Channel In[2]	Bool	FALSE
■ Channel In[3]	Bool	FALSE
■ Channel In[4]	Bool	FALSE
■ Channel In[5]	Bool	TRUE
■ Channel In[6]	Bool	FALSE
■ Channel In[7]	Bool	FALSE
■ Channel In[8]	Bool	FALSE

Setup of IC70 with other IO-Link Masters

1. The Banner IC70 Library will now be in the Global Library List. Expand the Master copies section.
2. Drag Banner_IC70_PD to the Program Blocks area under your PLC.
3. Drag the Banner_IC70_PDIO to the PLC Data Types area under your PLC.
4. Go to Devices and networks to configure the system as necessary. Below is an example of what a configuration might look like. This example shows 5 different IO-Link Masters connected to the same PLC.

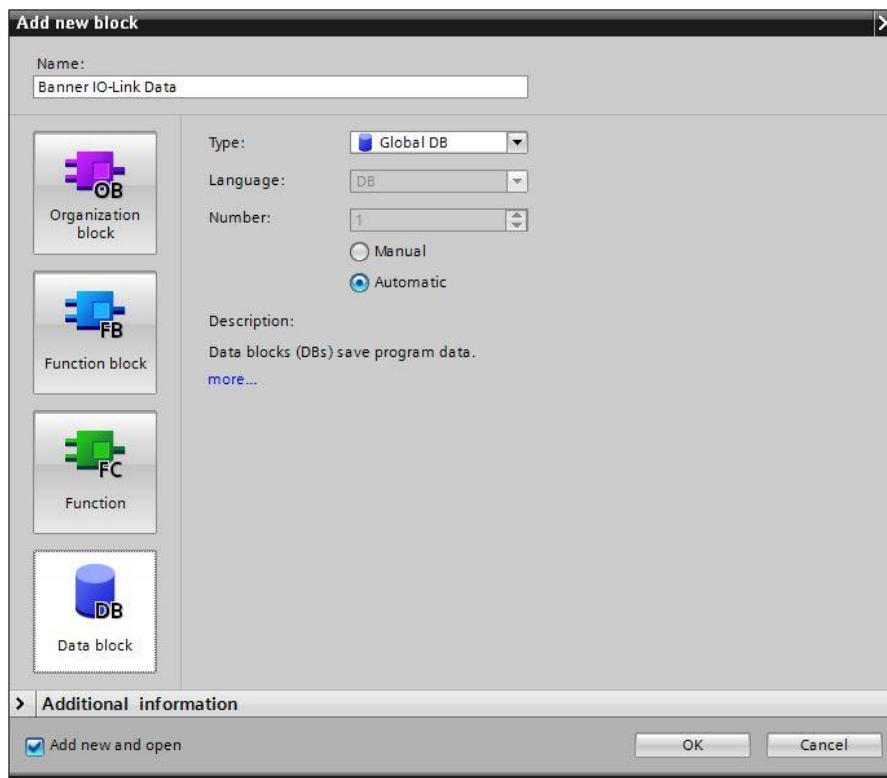


5. Click on the relevant device and configure the IO-Link Master as necessary. Refer to the documentation for the IO-Link Master. Recall that a I70 requires 2 bytes of space for the Process Data.
6. Record the "I" address and "Q" address where this IC70 Process Data is to be stored, as the addresses will be required in the next step. In this example, 2 bytes of Process Data In for port 1 on the IO-Link Master will be stored in I68, I69, Q68, and Q69.

7. Go to PLC Tags. Add a new tag table, then create a new tag to represent the raw Process Data from the IO-Link Master. In this example, Tag table_1 was created, then the tag “IC70 IOLM1 01 oRaw” and “IC70 IOLM1 01 iRaw” was created using Data Type of “Word”. This naming convention calls out the type of sensor in question as well as the specific IO-Link Master and port number where the sensor is connected. A different IO-Link Master might be named IOLM2 or IOLM3, for instance, and other specific sensors may be connected to different port numbers. The “I” and “Q” address found in step 6 is tied to this new tag.

IC70 IOLM1 01 oRaw	Word	%QW68
IC70 IOLM1 01 iRaw	Word	%IW68

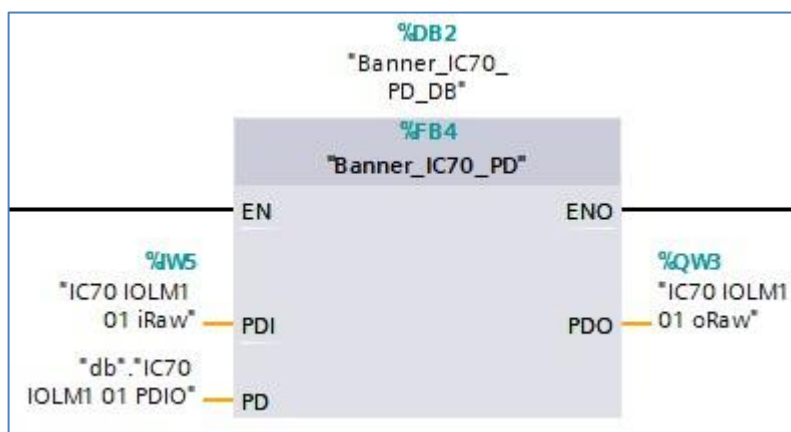
8. Go to Program blocks. Add a new Data block if necessary. In this example the new data block is named “Banner IO-Link Data”.



9. In the new data block, create a new tag to represent the parsed Process Data for the IC70. The tag name again calls out the type of sensor, the IO-Link Master, and the port number. Use the data type “Banner_IC70_PDIO” for the new tag.

▼ IC70 IOLM1 01 PDIO	"Banner_IC70_PDIO"
■ ▶ Channel In	Array[1..16] of Bool
■ ▶ Channel Out	Array[1..16] of Bool

10. Add the “Banner_IC70_PD” function to an OB ladder. Link “PDI” and “PDO” to the raw Process Data variables from step 7. Link “PD” to the parsed Process Data variable from step 9.



11. Process Data setup is complete.

12. Compile and download the configuration to the PLC, then go online. Open the “db” data block and click Monitor all. You should see parsed IC70 Process Data, like that shown below.

▼ IC70 IOLM1 01 PDIO	"Banner_IC70_PDIO"	
■ ► Channel In	Array[1..16] of Bool	
■ ▼ Channel Out	Array[1..16] of Bool	
■ Channel Out[1]	Bool	TRUE
■ Channel Out[2]	Bool	FALSE
■ Channel Out[3]	Bool	FALSE
■ Channel Out[4]	Bool	FALSE
■ Channel Out[5]	Bool	TRUE
■ Channel Out[6]	Bool	FALSE
■ Channel Out[7]	Bool	FALSE
■ Channel Out[8]	Bool	FALSE

▼ IC70 IOLM1 01 PDIO	"Banner_IC70_PDIO"	
■ ▼ Channel In	Array[1..16] of Bool	
■ Channel In[1]	Bool	TRUE
■ Channel In[2]	Bool	FALSE
■ Channel In[3]	Bool	FALSE
■ Channel In[4]	Bool	FALSE
■ Channel In[5]	Bool	TRUE
■ Channel In[6]	Bool	FALSE
■ Channel In[7]	Bool	FALSE
■ Channel In[8]	Bool	FALSE

Appendix A

IC70 Process Data

The IC70 has 2 bytes of Process Data In and Out, as shown below.

ProcessDataIn "Process Data Input" id=PD_ProcessDataIn

bit length: 16

data type: 16-bit Record (subindex access not supported)

subindex	bit offset	data type	allowed values	default value	acc. restr.	mod. other var.	excl. from DS	name	description
1	8	Boolean	false = Inactive, true = Active					Channel 1 Input State	true (1) = Channel Input Active. Note - even if is configured as an output, the active state will be reflected at the input
2	9	Boolean	false = Inactive, true = Active					Channel 2 Input State	true (1) = Channel Input Active. Note - even if is configured as an output, the active state will be reflected at the input
3	10	Boolean	false = Inactive, true = Active					Channel 3 Input State	true (1) = Channel Input Active. Note - even if is configured as an output, the active state will be reflected at the input
4	11	Boolean	false = Inactive, true = Active					Channel 4 Input State	true (1) = Channel Input Active. Note - even if is configured as an output, the active state will be reflected at the input
5	12	Boolean	false = Inactive, true = Active					Channel 5 Input State	true (1) = Channel Input Active. Note - even if is configured as an output, the active state will be reflected at the input
6	13	Boolean	false = Inactive, true = Active					Channel 6 Input State	true (1) = Channel Input Active. Note - even if is configured as an output, the active state will be reflected at the input
7	14	Boolean	false = Inactive, true = Active					Channel 7 Input State	true (1) = Channel Input Active. Note - even if is configured as an output, the active state will be reflected at the input
8	15	Boolean	false = Inactive, true = Active					Channel 8 Input State	true (1) = Channel Input Active. Note - even if is configured as an output, the active state will be reflected at the input
9	0	Boolean	false = Inactive, true = Active					Channel 9 Input State	true (1) = Channel Input Active. Note - even if is configured as an output, the active state will be reflected at the input
10	1	Boolean	false = Inactive, true = Active					Channel 10 Input State	true (1) = Channel Input Active. Note - even if is configured as an output, the active state will be reflected at the input
11	2	Boolean	false = Inactive, true = Active					Channel 11 Input State	true (1) = Channel Input Active. Note - even if is configured as an output, the active state will be reflected at the input
12	3	Boolean	false = Inactive, true = Active					Channel 12 Input State	true (1) = Channel Input Active. Note - even if is configured as an output, the active state will be reflected at the input
13	4	Boolean	false = Inactive, true = Active					Channel 13 Input State	true (1) = Channel Input Active. Note - even if is configured as an output, the active state will be reflected at the input
14	5	Boolean	false = Inactive, true = Active					Channel 14 Input State	true (1) = Channel Input Active. Note - even if is configured as an output, the active state will be reflected at the input
15	6	Boolean	false = Inactive, true = Active					Channel 15 Input State	true (1) = Channel Input Active. Note - even if is configured as an output, the active state will be reflected at the input
16	7	Boolean	false = Inactive, true = Active					Channel 16 Input State	true (1) = Channel Input Active. Note - even if is configured as an output, the active state will be reflected at the input

ProcessDataOut "Process Data Output" id=PD_ProcessDataOut

bit length: 16

data type: 16-bit Record (subindex access not supported)

subindex	bit offset	data type	allowed values	default value	acc. restr.	mod. other var.	excl. from DS	name	description
1	8	Boolean	false = Off, true = On					Channel 1 Output State	true (1) = Channel Output Active
2	9	Boolean	false = Off, true = On					Channel 2 Output State	true (1) = Channel Output Active
3	10	Boolean	false = Off, true = On					Channel 3 Output State	true (1) = Channel Output Active
4	11	Boolean	false = Off, true = On					Channel 4 Output State	true (1) = Channel Output Active
5	12	Boolean	false = Off, true = On					Channel 5 Output State	true (1) = Channel Output Active
6	13	Boolean	false = Off, true = On					Channel 6 Output State	true (1) = Channel Output Active
7	14	Boolean	false = Off, true = On					Channel 7 Output State	true (1) = Channel Output Active
8	15	Boolean	false = Off, true = On					Channel 8 Output State	true (1) = Channel Output Active
9	0	Boolean	false = Off, true = On					Channel 9 Output State	true (1) = Channel Output Active
10	1	Boolean	false = Off, true = On					Channel 10 Output State	true (1) = Channel Output Active
11	2	Boolean	false = Off, true = On					Channel 11 Output State	true (1) = Channel Output Active
12	3	Boolean	false = Off, true = On					Channel 12 Output State	true (1) = Channel Output Active
13	4	Boolean	false = Off, true = On					Channel 13 Output State	true (1) = Channel Output Active
14	5	Boolean	false = Off, true = On					Channel 14 Output State	true (1) = Channel Output Active
15	6	Boolean	false = Off, true = On					Channel 15 Output State	true (1) = Channel Output Active
16	7	Boolean	false = Off, true = On					Channel 16 Output State	true (1) = Channel Output Active