

S15C-MGN Process Data Function

January 15th, 2026

This document covers the installation and use of a function for Siemens's TIA Portal software package. This function handles cyclic IO-Link Process Data In and Process Data Out from a Banner S15C-MGN sensor via an IO-Link Master to a Siemens PLC. The function covers parsing and display of the S15C-MGN sensor Process Data In and Process Data Out.

Components

Banner S15C R45C Library v16.zal16

There are two methods for the process data. The first is used when creating a connection to Banner's IO-Link masters. The second set of instructions are for systems using other manufacturer's IO-Link masters.

Installation Instructions

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



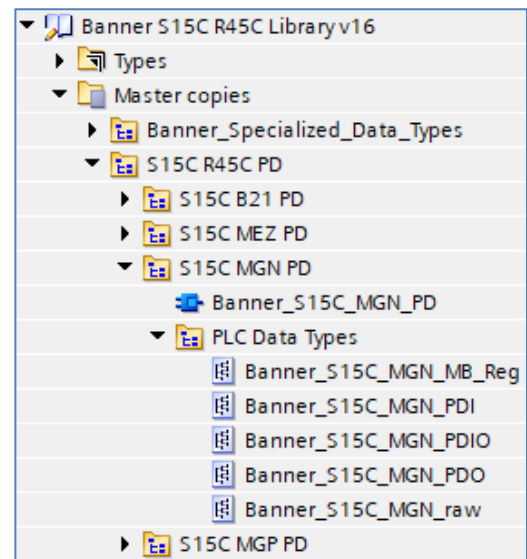
3. Switch the “Files of type” to Compressed libraries. Go to the location of the compressed library.
4. Press the Open button and the library will be uncompressed and opened.
5. The library is now accessible in the Libraries tab in v16 or greater.

Setup of S15C-MGN 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 32/32 byte option has been selected for port 1. Make note of the I address for the Slot 2 which represents Port 1. Slot 2 starts are 10. The other number needed is I14. The data for the port start at that point (I14). The previous four bytes represents Port Status, Process Data In Size, and Process Data Out Size.

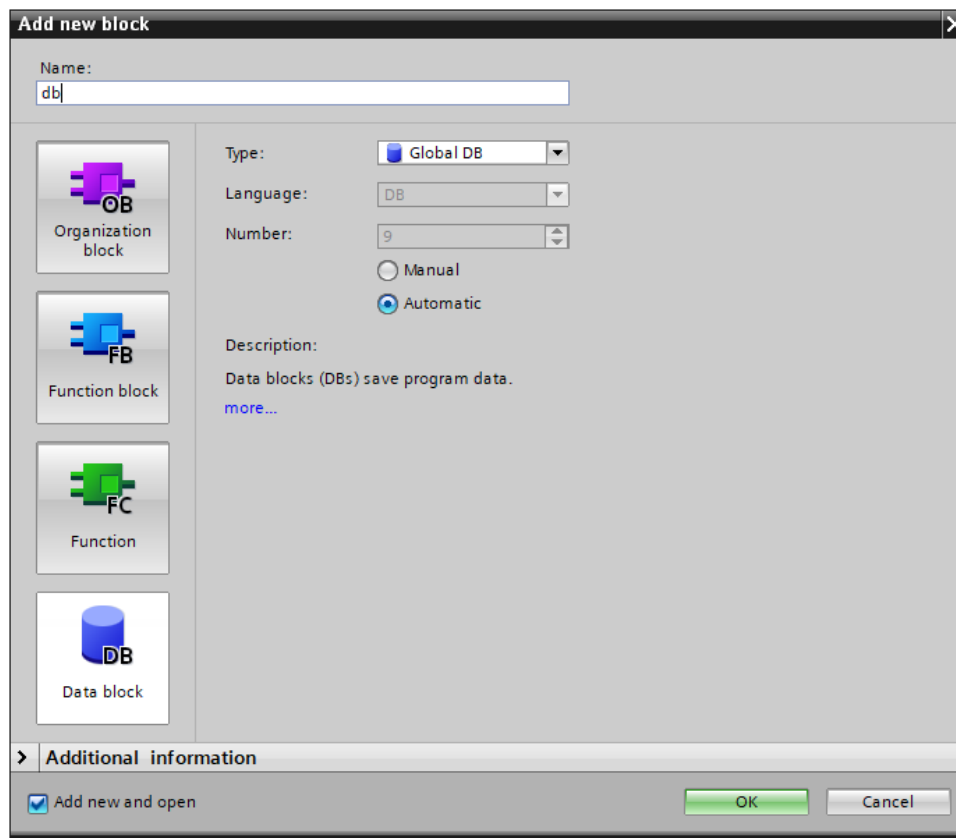
Module	Rack	Slot	I address	Q address	Type
▼ dxm	0	0			1-port Device
▶ Interface	0	0 X1			dxm
Banner IO-Link Master Info_1	0	1	1...9		Banner IO-Link Master Info
IO-Link In/Out 32/32 Byte + Status_1	0	2	10...45	1...46	IO-Link In/Out 32/32 Byte + Status

3. Drag the Banner_S15C_MGN_MB_Reg, Banner_S15C_MGN_PDI, Banner_S15C_MGN_PDIO, Banner_S15C_MGN_PDO, and Banner_S15C_MGN_raw to the PLC Data Types area under your PLC.
4. Drag the Banner_S15C_MGN_PD to the Program Blocks area.
5. Drag the necessary tags from Banner_Specialized_Data_Types. The tags used in this example is "Banner_32in" and "Banner_32out". These tags represent the full process data along with port status information.
6. 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 "S15C MGN IOLM1 01 PDI" and "S15C MGN IOLM1 01 PDO" was created using a Data Type of "Banner_32In" and "Banner_32Out". 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 "S15C_MGN_IOLM1_01_inRaw" and "S15C_MGN_IOLM1_01_outRaw" using "Banner_S15C_MGN_raw" data type. These are the tags that will be used in the Function block.



Name ▼	Data type	Address
▶ S15C MGN IOLM1 01 PDO	"Banner_32Out"	%Q1.0
▶ S15C MGN IOLM1 01 PDI	"Banner_32In"	%I10.0
▶ S15C MGN IOLM1 01 outRaw	"Banner_S15C_MGN_raw"	%Q3.0
▶ S15C MGN IOLM1 01 inRaw	"Banner_S15C_MGN_raw"	%I14.0

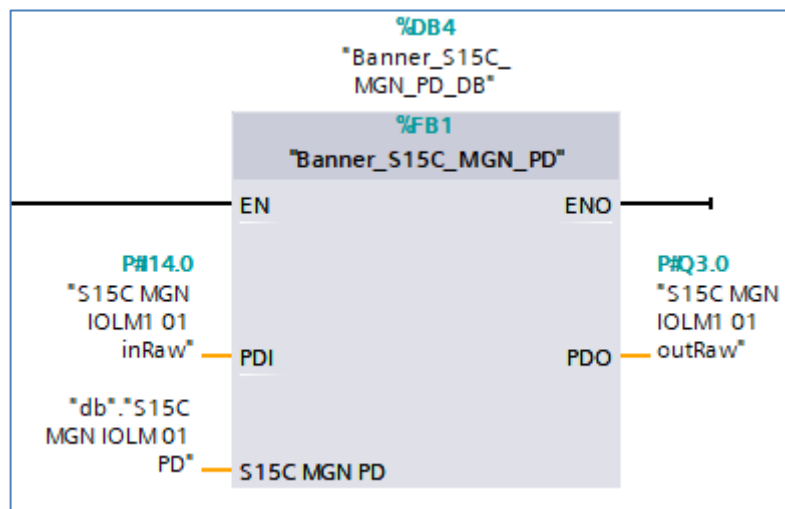
7. Go to Program blocks. Add a new Data block if necessary. In this example the new data block is named "db".



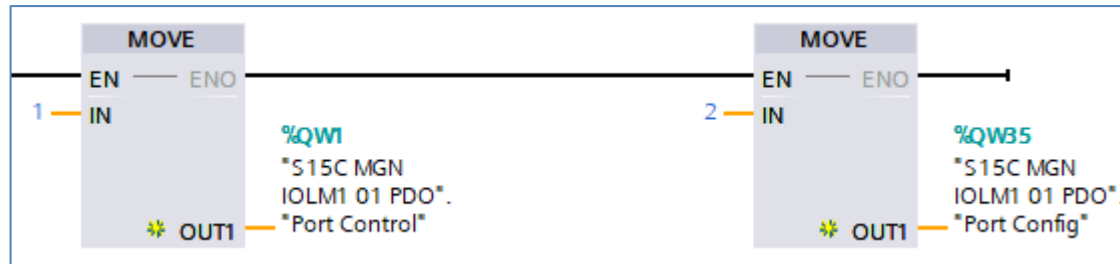
8. In the new data block, create a new tag to represent the parsed Process Data In for our LE. The tag name again calls out the type of sensor, the IO-Link Master, and the port number. Use the data type "Banner_S15C_MGN_PDI" for the new tag.

Name	Data type
▼ Static	
■ ▼ S15C MGN IOLM01 PD	"Banner_S15C_MGN_PDIO"
■ ▼ PDI	"Banner_S15C_MGN_PDI"
■ Register Set to Read	USInt
■ Register Read Successful	Bool
■ Register Write Successful	Bool
■ Register Verify Successful	Bool
■ Counter Value	USInt
■ ▶ Read Register Set	"Banner_S15C_MGN_MB_Reg"
■ ▼ PDO	"Banner_S15C_MGN_PDO"
■ Register Set to Read	USInt
■ ▶ Write Register Data	"Banner_S15C_MGN_MB_Reg"

9. Add the “Banner_S15C_MGN_PD” function to an OB ladder. Link the “PDI” and “PDO” to the raw Process Data variable from step 5. Link the “S15C PD” to the parsed Process Data variable from step 7.



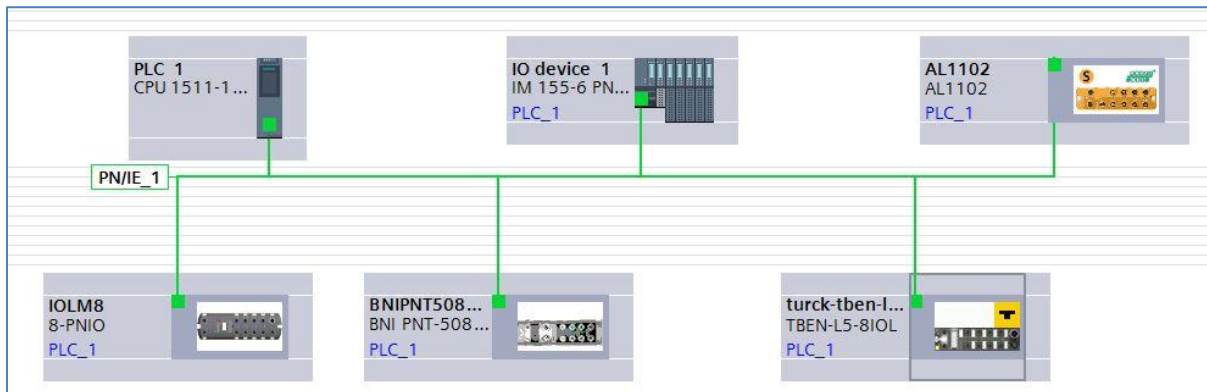
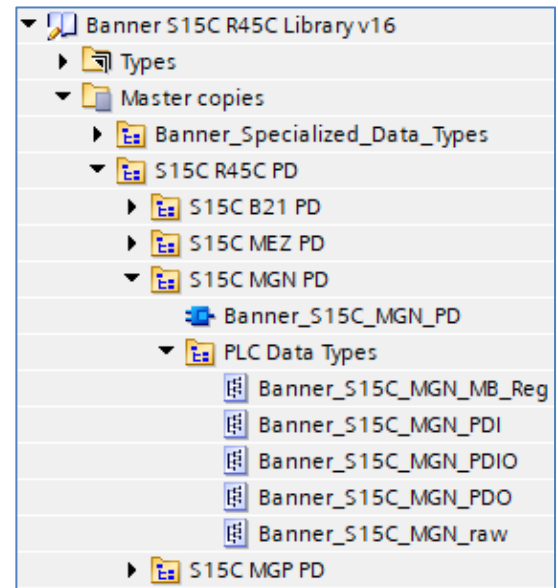
10. The final step is to configure the IO-Link output control. This is done by sending a 1 to Port Control and a 2 to Port Config. Both parameters are part of the tag created in step 6 “S15C MGN IOLM1 01 PDO”.



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 S15C MGN Process Data In.

Setup of S15C-MGN with other IO-Link Masters

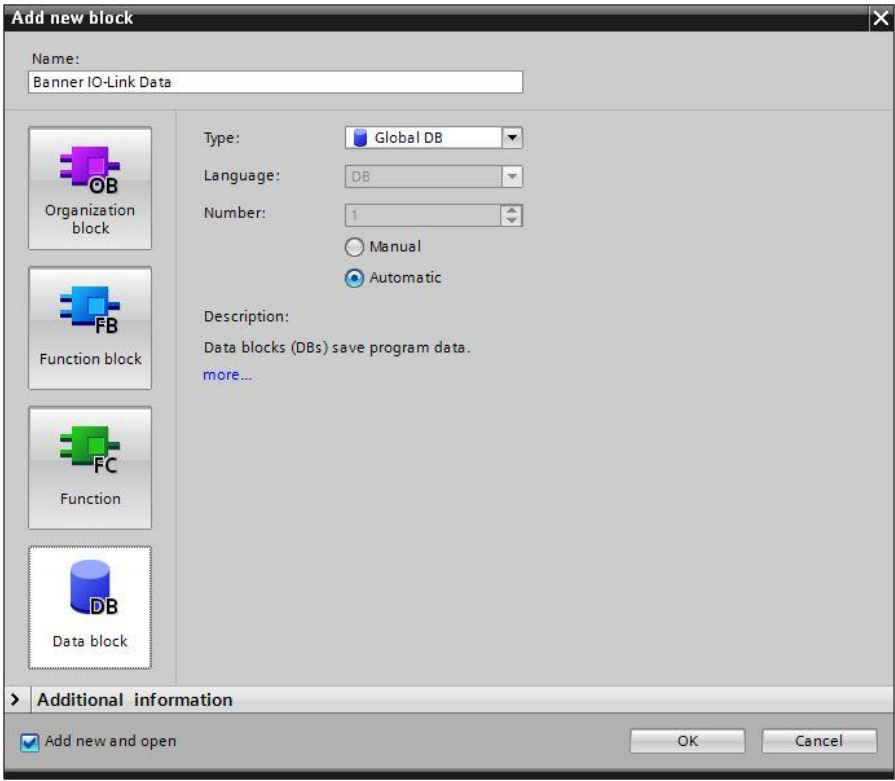
1. The Banner IO-Link Library will now be in the Global Library List. Expand the Master copies section.
2. Drag Banner_S15C_MGN_PD to the Program Blocks area under your PLC.
3. Drag Banner_S15C_MGN_MB_Reg, Banner_S15C_MGN_PDI, Banner_S15C_MGN_PDIO, Banner_S15C_MGN_PDO, and Banner_S15C_MGN_raw to the 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 S15C-MGN requires 32 bytes of space for the Process Data In and 1 byte for the Process Data Out.
6. Record the "I" address where this S15C-MGN Process Data In is to be stored, as the address will be required in the next step. In this example, 32 bytes of Process Data In for port 1 on the IO-Link Master will be stored starting at I68. The 32 bytes of Process Data Out starts at Q68.
7. Go to PLC Tags. Add a new tag table, if desired, then create a new tag to represent the raw Process Data from the IO-Link Master. In this example the tag "S15C MGN raw PDI" was created using a Data Type of "Banner_S15C_MGN_raw". Another tag is created called "S15C MNG raw PDO". This one is a Byte data type and is linked to the other memory address found in step 6.

▶ S15C MGN IOLM1 01 outRaw	"Banner_S15C_MGN_raw"	%Q68.0
▶ S15C MGN IOLM1 01 inRaw	"Banner_S15C_MGN_raw"	%I68.0

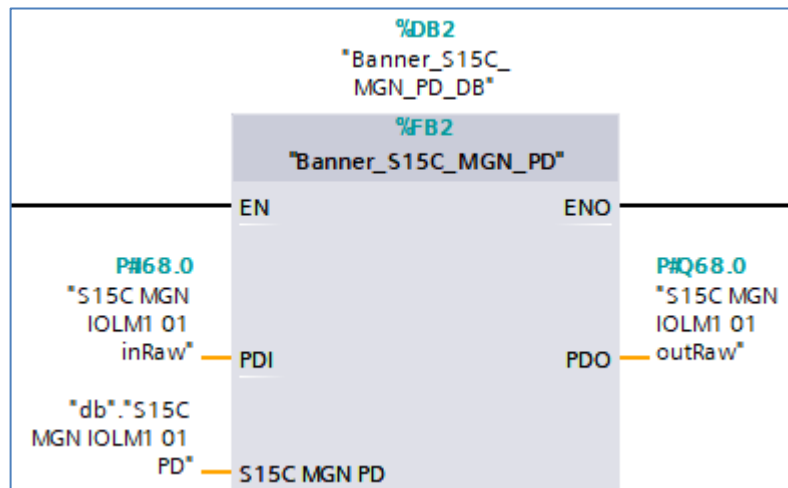
8. Go to Program blocks. Add a new Data block if necessary. In this example the new data block is named “Banner IO-Link Data”.



9. In the new data block, create a new tag to represent the parsed Process Data for our S15C-MGN. The tag name again calls out the type of sensor, the IO-Link Master, and the port number. Use the data type “Banner_S15C_MGN_PDIO” for the new tag.

▼ S15C MGN IOLM1 01 PD	"Banner_S15C_MGN_PDIO"
■ ▼ PDI	"Banner_S15C_MGN_PDI"
■ Register Set to Read	USInt
■ Register Read Successful	Bool
■ Register Write Successful	Bool
■ Register Verify Successful	Bool
■ Counter Value	USInt
■ ▶ Read Register Set	"Banner_S15C_MGN_MB_Reg"
■ ▼ PDO	"Banner_S15C_MGN_PDO"
■ Register Set to Read	USInt
■ ▶ Write Register Data	"Banner_S15C_MGN_MB_Reg"

10. Add the “Banner_S15C_MGN_PD” function to an OB ladder. Link the “PDI” and “PDO” to the raw Process Data variables from step 7. Link the “S15C MGN 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 “Banner IO-Link Data” data block and click Monitor all. Expand “S15C MGN IOLM1 01 PD”. This shows all the Process Data for the device. Variable “Register Set” controls which of the three process data configurations is being displayed. The data is shown in either PDI 0, PDI 1, or PDI 2 based on the value of “Register Set”.

Appendix A

S15C-MGN Process Data

The S15C-MGN has 32 bytes of Process Data In and 31 bytes of Process Data Out, as shown below.

ProcessDataIn "Process Data In" id=V_Pd_InData									
bit length: 256									
data type: 256-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	0	4-bit UInteger	0..1					Register Set To Read	Register Set To Read
2	4	Boolean	false = False, true = True					Register Read Successful	Register Read Successful
3	5	Boolean	false = False, true = True					Register Write Successful	Register Write Successful
4	6	Boolean	false = False, true = True					Register Verify Successful	Register Verify Successful
5	8	8-bit UInteger	0..255					Counter Value	Counter Value
6	16	16-bit UInteger	0..65535					Read Set Register 01 Value	Register Value. Data that was read from register.
7	32	16-bit UInteger	0..65535					Read Set Register 02 Value	Register To Read. Register to read from ModBus device.
8	48	16-bit UInteger	0..65535					Read Set Register 03 Value	Register Value. Data that was read from register.
9	64	16-bit UInteger	0..65535					Read Set Register 04 Value	Register To Read. Register to read from ModBus device.
10	80	16-bit UInteger	0..65535					Read Set Register 05 Value	Register Value. Data that was read from register.
11	96	16-bit UInteger	0..65535					Read Set Register 06 Value	Register To Read. Register to read from ModBus device.
12	112	16-bit UInteger	0..65535					Read Set Register 07 Value	Register Value. Data that was read from register.
13	128	16-bit UInteger	0..65535					Read Set Register 08 Value	Register Value. Data that was read from register.
14	144	16-bit UInteger	0..65535					Read Set Register 09 Value	Register Value. Data that was read from register.
15	160	16-bit UInteger	0..65535					Read Set Register 10 Value	Register Value. Data that was read from register.
16	176	16-bit UInteger	0..65535					Read Set Register 11 Value	Register Value. Data that was read from register.
17	192	16-bit UInteger	0..65535					Read Set Register 12 Value	Register Value. Data that was read from register.
18	208	16-bit UInteger	0..65535					Read Set Register 13 Value	Register Value. Data that was read from register.
19	224	16-bit UInteger	0..65535					Read Set Register 14 Value	Register Value. Data that was read from register.
20	240	16-bit UInteger	0..65535					Read Set Register 15 Value	Register Value. Data that was read from register.

Figure 1: PDI

ProcessDataOut "Process Data Out" id=V_Pd_OutData

bit length: 248

data type: 248-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	0	8-bit UInteger	0..3					Register Set To Read	Register Set To Read
2	8	16-bit UInteger	0..65535					Write Set Register 01 Value	Value To Write into register. Register value to write to ModBus device.
3	24	16-bit UInteger	0..65535					Write Set Register 02 Value	Register To Read. Register to read from ModBus device.
4	40	16-bit UInteger	0..65535					Write Set Register 03 Value	Value To Write into register. Register value to write to ModBus device.
5	56	16-bit UInteger	0..65535					Write Set Register 04 Value	Register To Read. Register to read from ModBus device.
6	72	16-bit UInteger	0..65535					Write Set Register 05 Value	Value To Write into register. Register value to write to ModBus device.
7	88	16-bit UInteger	0..65535					Write Set Register 06 Value	Register To Read. Register to read from ModBus device.
8	104	16-bit UInteger	0..65535					Write Set Register 07 Value	Value To Write into register. Register value to write to ModBus device.
9	120	16-bit UInteger	0..65535					Write Set Register 08 Value	Value To Write into register. Register value to write to ModBus device.
10	136	16-bit UInteger	0..65535					Write Set Register 09 Value	Value To Write into register. Register value to write to ModBus device.
11	152	16-bit UInteger	0..65535					Write Set Register 10 Value	Value To Write into register. Register value to write to ModBus device.
12	168	16-bit UInteger	0..65535					Write Set Register 11 Value	Value To Write into register. Register value to write to ModBus device.
13	184	16-bit UInteger	0..65535					Write Set Register 12 Value	Value To Write into register. Register value to write to ModBus device.
14	200	16-bit UInteger	0..65535					Write Set Register 13 Value	Value To Write into register. Register value to write to ModBus device.
15	216	16-bit UInteger	0..65535					Write Set Register 14 Value	Value To Write into register. Register value to write to ModBus device.
16	232	16-bit UInteger	0..65535					Write Set Register 15 Value	Value To Write into register. Register value to write to ModBus device.

Figure 2: PDO

This function intelligently parses this Process Data into its component pieces.