

LCA Process Data Function

January 5th, 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 Out to a Banner LCA lights via an IO-Link Master from a Siemens PLC. The function covers parsing and display of the LCA sensor Process Data Out.

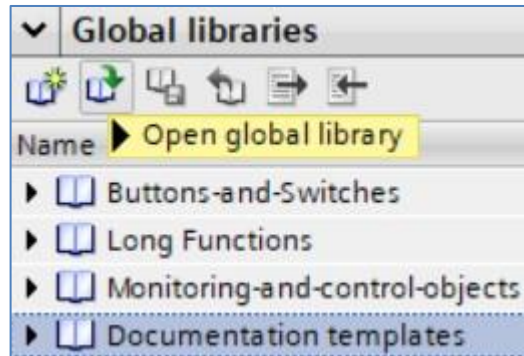
Components

Banner LCA Library v16.zal16

There are two methods for 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 manufacturers' IO-Link masters.

Installation Instructions

1. Open a project.
2. Go Global Libraries and select the Open Global Library option.



3. Select the Banner LCA Library v16. Click Open.
4. The library is now accessible in the Libraries tab.
5. Go to page 3 for Banner IO-Link Masters, to page 8 for all other IO-Link Masters.

Setup of LCA with a Banner DXMR

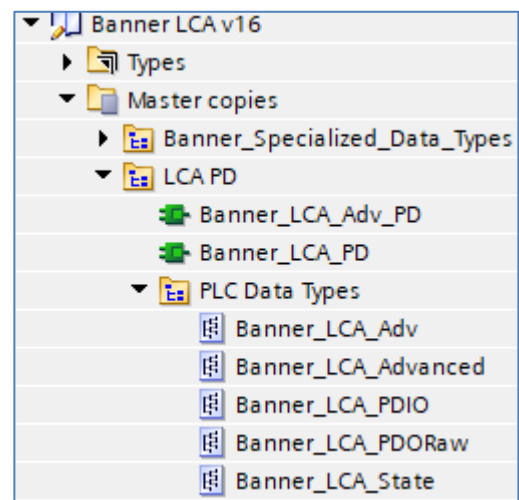
1. Go to Device and Networks to configure the DXMR. Add the DXMR if it has yet to be added to the system.
2. Add Banner IO-Link Master Info to Slot 1. This sets the DXMR for IO-Link mode.

Banner IO-Link Master Info_1	0	1	1...9	Banner IO-Link Master Info
------------------------------	---	---	-------	----------------------------

3. Open the IO-Link Generic Devices and select the proper module. The **32/32 byte** is required for **LCA**. Make note of the input %I10 address for Slot 2 which represents Port 1. Slot 2 starts are %Q1 for outputs.

IO-Link In/Out 32/32 Byte + Status_1	0	2	10...45	1...46	IO-Link In/Out 32/32 Byte + Status
--------------------------------------	---	---	---------	--------	------------------------------------

4. Drag the necessary tags from "Banner_Specialized_Data_Types". The tags used in this example is "Banner_32Out" and "Banner_32In". These tags represent the full raw process data along with port information.
5. Drag the necessary files from the "LCA PD" folder.
 - a. Move "Banner_LCA_Adv", "Banner_LCA_Advanced", "Banner_LCA_PDIO", "Banner_LCA_PDORaw", and "Banner_LCA_State" to the PLC Data Types area.
 - b. Move "Banner_LCA_Adv_PD" and "Banner_LCA_PD" to the Program Blocks area.
6. Go to PLC Tags. Create four tags. One set of tags is for the full data structure while the second set creates tags to represent the raw Process Data from the IO-Link Master. In this example, Tag table_1 was created, then the tag "LCA IOLM1 01 PDO" was created using a Data Type of "Banner_32Out". This naming convention calls out the type of device in question as well as the specific IO-Link Master and port number to which 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 "Q" address found in step 3 (%Q1) is tied to this new tag. The tag that represents the raw data is "LCA IOLM1 01 outRaw" and uses the "Q" address found in step 3 (%Q3). Tags "LCA IOLM1 01 PDI" (%I10) and "K50 IOLM1 01 inRaw" (%IW14) are created for the inputs also. This is the tag that will be used in the Function block.



Name	Data type	Address
▶ LCA IOLM1 01 PDI	"Banner_32In"	%I10.0
LCA IOLM1 01 inRaw	UInt	%IW14
▶ LCA IOLM1 01 PDO	"Banner_32Out"	%Q1.0
▶ LCA IOLM1 01 outRaw	"Banner_LCA_PDORaw"	%Q3.0

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 Output for our LCA. The tag name again calls out the type of sensor, the IO-Link Master, and the port number. Use the data type "Banner_LCA_PDIO" for the new tag.

▼ LCA IOLM01 01 PD	"Banner_LCA_PDIO"
■ ▶ 0-State	"Banner_LCA_State"
■ ▶ 1-Advanced	"Banner_LCA_Advanced"

9. Add the "Banner_LCA_PD" function to an OB ladder. Link the "PDO" to the raw process data variable from step 6. The tag name again calls out the type of device, IO-Link Master, and the port number. Use the variable called "LCA IOLM1 01 outRaw" in this example. Link the "PDI" to the raw process data variable from step 5. Use the variable called "LCA IOLM1 01 inRaw" from step 6. The "LCA PD" needs to be linked to the variable created in step 7. It was called "LCA IOLM1 01 PD" for this example.

The last variable, "Operational Mode", allow the function to correctly interpret the Process Data Out. In the case of the LCA, there are two user-selected modes for the Process Data Out. This function needs to know what choice has been made in the LCA for this Operational Mode variable.

There are two ways to achieve this goal. We can simply type in the correct number for Operational Mode (see Fig. 1), or we can link this LCA Process Data Function to the LCA Parameter Data Function Block (see Fig. 2). See Appendix A for more information about LCA Process Data Out.

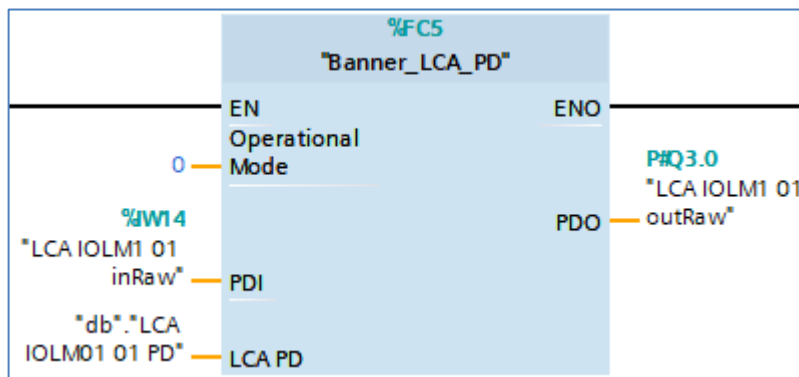


Figure 1: Hand type correct number for Operational Mode

NOTE: if you type in the incorrect number (i.e. it does not match the light's current Operational Mode) you will get incorrectly displayed Process Data Out information.

Operational Mode: the options here are "0" (State Mode) and "1" (Advanced Mode). The default is "0".

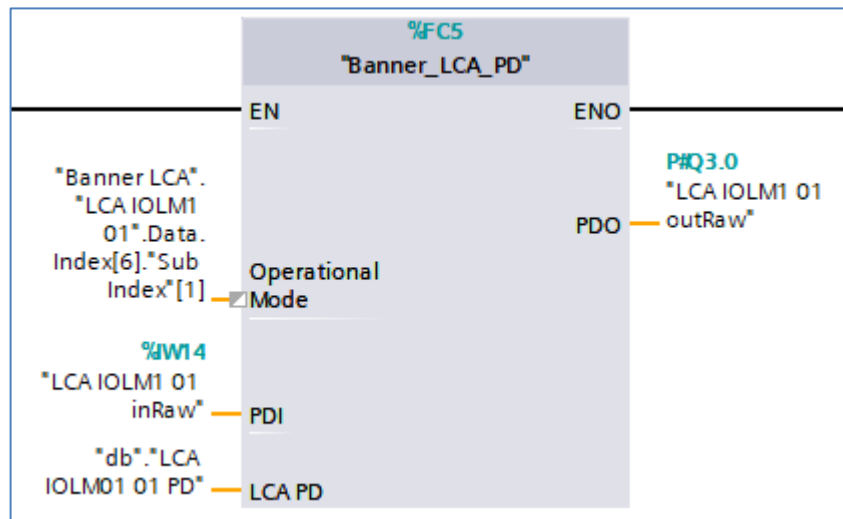
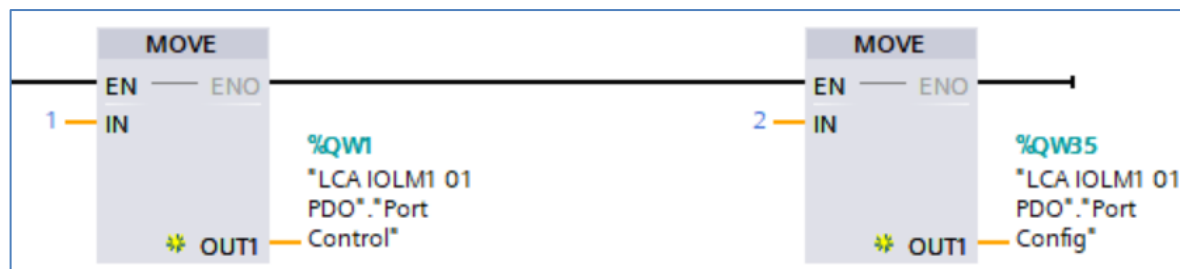


Figure 2: Linking Operational Mode variable to LCA Parameter Data Function Block

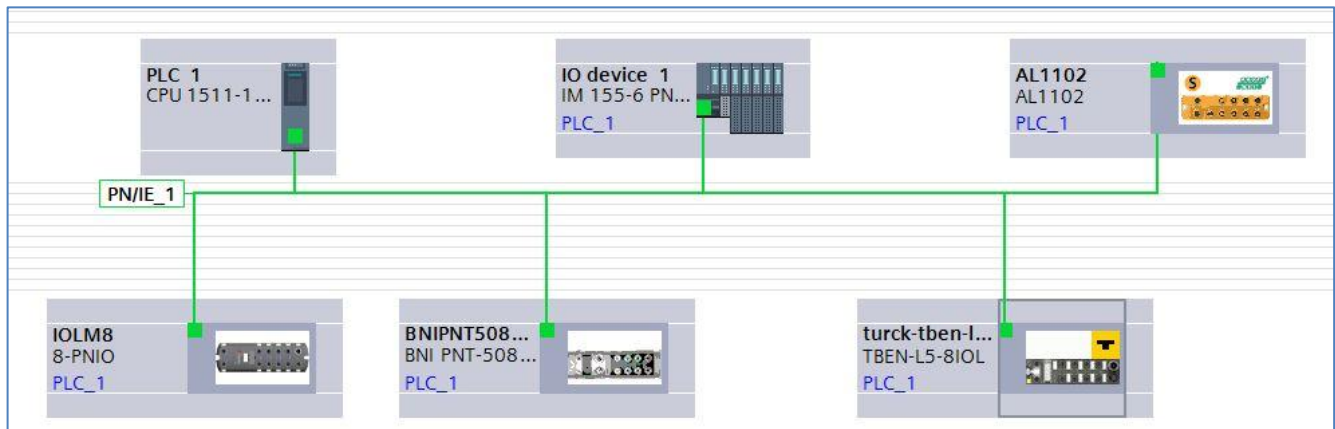
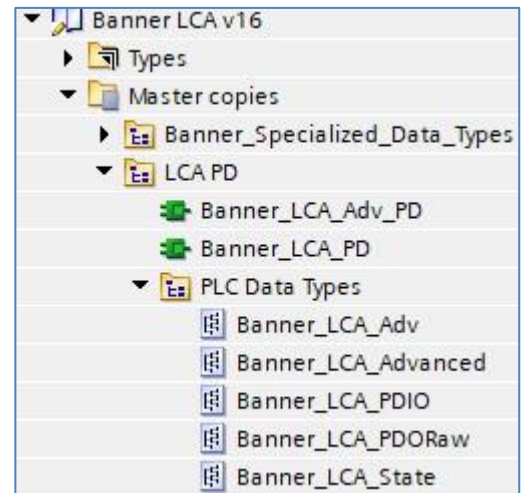
10. The final step is to configure the IO-Link output control. This is done by sending a 1 to Port Control and a 2 to Port Config. Both parameters are part of the tag created in step 6 "LCA IOLM1 01 PDO".



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. The LCA can be controlled now.

Setup of LCA with other IO-Link Masters

1. The Banner LCA library will now be in the Global Library List. Expand the Master copies section.
2. Drag the necessary files from the “LCA PD” folder.
 - a. Move “Banner_LCA_Adv”, “Banner_LCA_Advanced”, “Banner_LCA_PDIO”, “Banner_LCA_PDORaw”, and “Banner_LCA_State” to the PLC Data Types area.
 - b. Move “Banner_LCA_Adv_PD” and “Banner_LCA_PD” to the Program Blocks area.
3. 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.



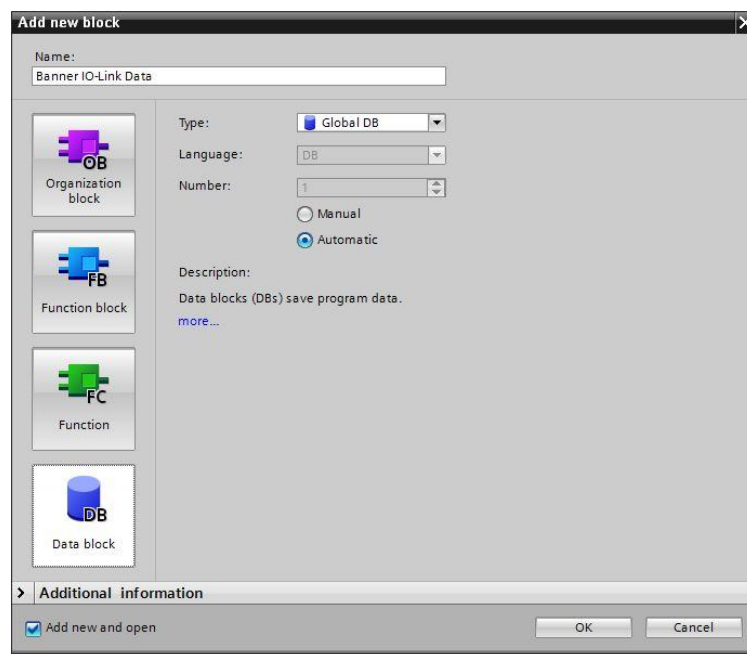
4. Click on the relevant device and configure the IO-Link Master as necessary. Refer to the documentation for the IO-Link Master. The **LCA** uses **32in/32out byte**.
5. Record the “I” and “Q” addresses where this LCA Process Data is to be stored, as these addresses will be required in the next step.

6. Go to PLC Tags. Create 2 tags for LCA. In this example, Tag table_1 was created, then the tag “LCA IOLM1 01 outRaw” was created using a Data Type of “Banner_LCA_PDORaw”. This naming convention calls out the type of device in question as well as the specific IO-Link Master and port number to which 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 “Q” address found in step 3 (%Q3) is tied to this new tag. Tags “LCA IOLM1 01 inRaw” (%IW14). This is the tag that will be used in the Function block.

Name	Data type	Address
LCA IOLM1 01 inRaw	UInt	%IW14
▶ LCA IOLM1 01 outRaw	"Banner_LCA_PDORaw"	%Q3.0

LCA Example

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



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 our LCA. The tag name again calls out the type of sensor, the IO-Link Master, and the port number. Use the data type “Banner_LCA_PDIO” for the new tag.

▼ LCA IOLM01 01 PD	"Banner_LCA_PDIO"
▶ 0-State	"Banner_LCA_State"
▶ 1-Advanced	"Banner_LCA_Advanced"

10. Add the “Banner_LCA_PD” function to an OB ladder. Link the “PDO” to the raw process data variable from step 6. The tag name again calls out the type of device, IO-Link Master, and the port number. Use the variable called “LCA IOLM1 01 outRaw” in this example. Link the “PDI” to the raw process data variable from step 5. Use the variable called “LCA IOLM1 01 inRaw” from step 6. The “LCA PD” needs to be linked to the variable created in step 9. It was called “LCA IOLM1 01 PD” for this example.

The last variable, “Operational Mode”, allow the function to correctly interpret the Process Data Out. In the case of the LCA, there are four user-selected modes for the Process Data Out. This function needs to know what choice has been made in the LCA for this Operational Mode variable.

There are two ways to achieve this goal. We can simply type in the correct number for Operational Mode (see Fig. 1), or we can link this LCA Process Data Function to the LCA Parameter Data Function Block (see Fig. 2). See Appendix A for more information about LCA Process Data Out.

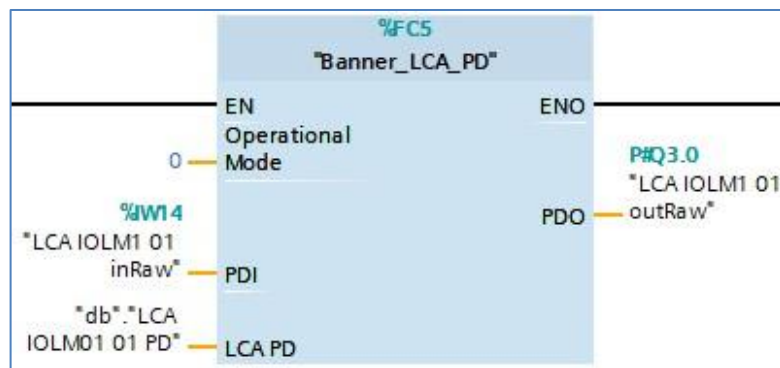


Figure 3: Hand type correct number for Operational Mode

NOTE: if you type in the incorrect number (i.e. it does not match the light’s current Operational Mode) you will get incorrectly displayed Process Data Out information.

Operational Mode: the options here are “0” (State Mode) and “1” (Advanced Mode). The default is “0”.

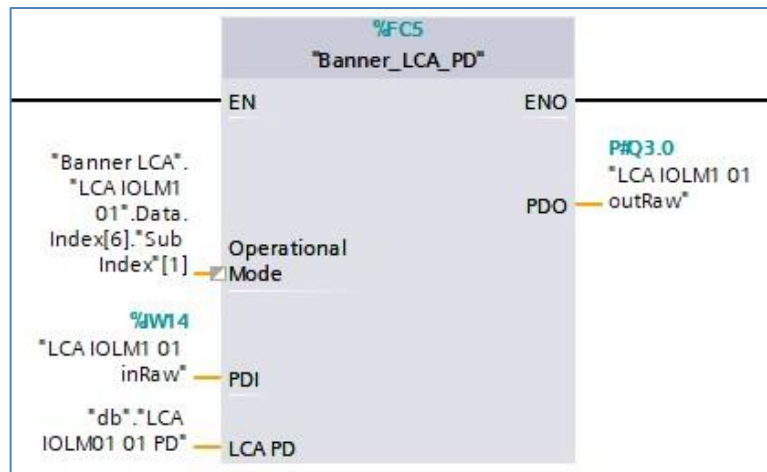


Figure 4: Linking Operational Mode variable to LCA Parameter Data Function Block

11. Process Data Setup is complete.

Compile and download the configuration to the PLC, then go online. Open the “db” data block and click Monitor all. The LCA can be controlled now.

Using LCA Process Data

Below each of the process data modes will be described for the LCA.

a. State Mode (0)

- i. PDI Output State: gives the status of the touch buttons.
- ii. PDO Output Control: allows control of the light on the LCA.

▼ PDI Output State	Array[1..5] of USInt	▼ PDO Output Control	Array[1..5] of USInt
■ PDI Output State[1]	USInt	■ PDO Output Control[1]	USInt
■ PDI Output State[2]	USInt	■ PDO Output Control[2]	USInt
■ PDI Output State[3]	USInt	■ PDO Output Control[3]	USInt
■ PDI Output State[4]	USInt	■ PDO Output Control[4]	USInt
■ PDI Output State[5]	USInt	■ PDO Output Control[5]	USInt

b. Advanced Mode (1)

- i. PDI Output State: gives the status of the touch buttons.
- ii. PDO Button Control: Gives complete control over button and light.

▼ PDI Output State	Array[1..5] of USInt
■ PDI Output State[1]	USInt
■ PDI Output State[2]	USInt
■ PDI Output State[3]	USInt
■ PDI Output State[4]	USInt
■ PDI Output State[5]	USInt

▼ PDO Button Control	Array[1..5] of *Banner_LCA_Adv*
■ ▶ PDO Button Control[1]	*Banner_LCA_Adv*
■ ▶ PDO Button Control[2]	*Banner_LCA_Adv*
■ ▶ PDO Button Control[3]	*Banner_LCA_Adv*
■ ▶ PDO Button Control[4]	*Banner_LCA_Adv*
■ ▶ PDO Button Control[5]	*Banner_LCA_Adv*

▼ PDO Button Control[1]	*Banner_LCA_Adv*
■ Animation Type	USInt
■ Output	USInt
■ Pattern	USInt
■ Speed	USInt
■ Color 1	USInt
■ Color 1 Intensity	USInt
■ Color 2	USInt
■ Color 2 Intensity	USInt

Appendix A

LCA Process Data

The LCA has 2 bytes of Process Data In and 20 bytes of Process Data Out. There are two modes for this data, as shown below. This Process Data is mapped to a specific group of PROFINET addresses. This function intelligently parses this Process Data into its component pieces.

The first is mode 0, "State".

ProcessDataIn "Process Data In" id=V_Pd_InState

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	2-bit UInteger	0 = Reserved, 1 = Off, 2 = On, 3 = Hold					Output 1 State	Output State. Related parameters defined in output and touch settings parameter data.
2	10	2-bit UInteger	0 = Reserved, 1 = Off, 2 = On, 3 = Hold					Output 2 State	Output State. Related parameters defined in output and touch settings parameter data.
3	12	2-bit UInteger	0 = Reserved, 1 = Off, 2 = On, 3 = Hold					Output 3 State	Output State. Related parameters defined in output and touch settings parameter data.
4	14	2-bit UInteger	0 = Reserved, 1 = Off, 2 = On, 3 = Hold					Output 4 State	Output State. Related parameters defined in output and touch settings parameter data.
5	0	2-bit UInteger	0 = Reserved, 1 = Off, 2 = On, 3 = Hold					Output 5 State	Output State. Related parameters defined in output and touch settings parameter data.

ProcessDataOut "Process Data Out" id=V_Pd_OutState

bit length: 160

data type: 160-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	152	2-bit UInteger	0 = No Override, 1 = Off, 2 = On, 3 = Hold					Output 1 State	Output State. Related parameters defined in output and touch settings parameter data.
2	144	2-bit UInteger	0 = No Override, 1 = Off, 2 = On, 3 = Hold					Output 2 State	Output State. Related parameters defined in output and touch settings parameter data.
3	136	2-bit UInteger	0 = No Override, 1 = Off, 2 = On, 3 = Hold					Output 3 State	Output State. Related parameters defined in output and touch settings parameter data.
4	128	2-bit UInteger	0 = No Override, 1 = Off, 2 = On, 3 = Hold					Output 4 State	Output State. Related parameters defined in output and touch settings parameter data.
5	120	2-bit UInteger	0 = No Override, 1 = Off, 2 = On, 3 = Hold					Output 5 State	Output State. Related parameters defined in output and touch settings parameter data.

The next mode, “1”, is “Advanced”.

ProcessDataIn "Process Data In" id=V_Pd_InAdvanced

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					Output 1 State	Output State. Related parameters defined in output and touch settings parameter data.
2	9	Boolean	false = Off, true = On					Output 2 State	Output State. Related parameters defined in output and touch settings parameter data.
3	10	Boolean	false = Off, true = On					Output 3 State	Output State. Related parameters defined in output and touch settings parameter data.
4	11	Boolean	false = Off, true = On					Output 4 State	Output State. Related parameters defined in output and touch settings parameter data.
5	12	Boolean	false = Off, true = On					Output 5 State	Output State. Related parameters defined in output and touch settings parameter data.

ProcessDataOut "Process Data Out" id=V_Pd_OutAdvanced

bit length: 160

data type: 160-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	152	4-bit UInteger	0 = Off, 1 = Steady, 2 = Flash, 3 = Two Color Flash, 4 = Intensity Sweep, 5 = Color Sweep					Button 1 Animation Type	The Animation type
2	156	2-bit UInteger	0 = Off, 1 = On, 2 = Pattern					Button 1 Output	The Output state
3	144	3-bit UInteger	0 = Flash, 1 = Strobe, 2 = Three Pulse, 3 = SOS, 4 = Random					Button 1 Animation Pattern	The pattern of Animation/Haptic Feedback
4	147	2-bit UInteger	0 = Slow, 1 = Medium, 2 = Fast, 3 = Custom					Button 1 Animation Speed	The speed of the Animation/Haptic Feedback
5	136	5-bit UInteger	0 = Green, 1 = Red, 2 = Orange, 3 = Amber, 4 = Yellow, 5 = Lime Green, 6 = Spring Green, 7 = Cyan, 8 = Sky Blue, 9 = Blue, 10 = Violet, 11 = Magenta, 12 = Rose, 13 = White, 14 = Custom1, 15 = Custom2					Button 1 Color 1	The main color of the Animation, Custom Colors are defined in Parameter data
6	141	3-bit UInteger	0 = High, 1 = Medium, 2 = Low, 3 = Off, 4 = Custom					Button 1 Color 1 Intensity	The Intensity of Color 1, Custom Intensity defined in Parameter Data
7	128	5-bit UInteger	0 = Green, 1 = Red, 2 = Orange, 3 = Amber, 4 = Yellow, 5 = Lime Green, 6 = Spring Green, 7 = Cyan, 8 = Sky Blue, 9 = Blue, 10 = Violet, 11 = Magenta, 12 = Rose, 13 = White, 14 = Custom1, 15 = Custom2					Button 1 Color 2	The secondary color of the Animation, Custom Colors are defined in Parameter data
8	133	3-bit UInteger	0 = High, 1 = Medium, 2 = Low, 3 = Off, 4 = Custom					Button 1 Color 2 Intensity	The Intensity of Color 2, Custom Intensity defined in Parameter Data

**Only Button 1 PDO data shown