Digitized Automation for a Changing World

DIADesigner - AX Series Standard Instructions Manual

www.deltaww.com

Revision History

Version	Revision	Date
1st	The first version was published.	10/30/2020
2nd	1. Updated CH3: ◦ Added supported products AX-364E & AX-324N for DFB_Capture, DFB_Compare, DFB_HCnt, DFB_HTmr, DFB_PresetValue, DFB_Sample ◦ Updated library DL_BuiltInIO_AX3.library for DFB_Capture, DFB_Compare, DFB_HCnt, DFB_HTmrr, DFB_PresetValue, DFB_Sample ◦ Updated “dwTimerValue” unit 2. Updated CH4: ◦ Added supported product AX-364E for DFB_EcGetAllSlaveAddr, DFB_EcGetSlaveCount, DFB_EtherCATLink_Diag, DFB_GetAllECATSlaveInfo, DFB_GetECATMasterError, DFB_GetECATMasterState, DFB_ResetECATMaster, DFB_ResetECATSlave 3. Updated CH6: ◦ Added supported products AX-364E, AX-300, and AX-324N for DFB_From, DFB_To 4. Updated CH7: ◦ Added supported products AX-364E,AX-300, and AX-324N for DFB_ModbusComChannel, DFB_ModbusRequest, DFB_ModbusRequest2 “DFB_MR_ERROR“ update ◦ Added DFB_SLAVE_DEVICE_FAILURE, DFB_ACKNOWLEDGE, DFB_SLAVE_DEVICE_BUSY, DFB_GATEWAY_PATH_UNAVAILABLE, DFB_GATEWAY_DEVICE_FAILED_TO_RESPOND ◦ Removed DFB_NO_MDBSCOM_CONFIG ◦ Updated DFB_ILLEGAL_DATA_VALUE 5. Updated CH8: ◦ Added supported products AX-364E, AX-300, and AX-324N for DFB_TCP_Client, DFB_TCP_Server, DFB_UDP_Socket, DFB_ModbusTCPChannel, DFB_ModbusTCPRequest "DFB_MR_ERROR“ update ◦ Added DFB_SLAVE_DEVICE_FAILURE, DFB_ACKNOWLEDGE, DFB_SLAVE_DEVICE_BUSY, DFB_GATEWAY_PATH_UNAVAILABLE, DFB_GATEWAY_DEVICE_FAILED_TO_RESPOND,	02/28/2021

Version	Revision	Date
	DFB_INVALID_FUNCTION_CODE, DFB_NO_ETHERNET_CONFIG ◦ Removed DFB_NO_MDBSETH_CONFIG ◦ Updated DFB_ILLEGAL_DATA_VALUE, DFB_MEMORY_NOT_ENOUGH 6. Added CH10 High Speed Output Instructions 7. Updated CH11: ◦ Added supported products AX-364E, AX-300, and AX-324N for DFC_LogGetSize, DFB_LogDump	
3rd	1. Updated Section 1.1: Added Setting Value range of wNum 2. Updated Section 2.5: Updated input pin note 3. Updated Section 4.3: Updated DFB_EtherCATLink_Diag library: DL_EtherCAT_Diag.library 4. Added Section 6.3–Section 6.19: DFB_DLCCAL, DFB_DLCWEI, DFB_DPUCONF, DFB_PUSTAT, DFB_DPUPLS, DFB_DPUPLS, DFB_DPUDRI, DFB_DPUDRA, DFB_DPUZRN, DFB_DPUJOG, DFB_DPUCNT, DFB_DMPID library: DL_ASModuleAPI_AX3.library, DFB_DHCCNT, DFB_DHCCAP, DFB_HCDO, DFB_DHCCMP, DFB_DHCCMPT, DFB_DHCMEAS 5. Updated Section 7.3: Added Notes 4 & 5 6. Updated Section 7.4: Added Notes 1 & 2 7. Updated Sections 8.1, 8.2, and 8.3: Updated example program 8. Updated Sections 8.1–8.5: Added Function notes	07/31/2021
4th	1. Updated Section 5.1–5.3: ◦ Corrected output Error_ID data type ◦ Adjusted dwLen Input to wLen ◦ Updated Input pSrc range and added notes ◦ Updated error code description 2. Added Section 6.20–6.21: DFB_DADLOG, DFB_DADPEAK 3. Updated Section 6.22: Added DFB_DADLOG, DFB_DADPEAK error codes 4. Updated Section 9.2: Updated Input FileInfo data type 5. Updated Section 9.3: Updated error code content 6. Updated Section 11.3: Updated error code content, added DFB_FTPClient	06/01/2022

Version	Revision	Date
1.4.0	1. Updated Section 3.1: Added bEdgeSelect and bCycle. 2. Updated Section 3.2: Added OutputAction, diTablePosition, diTableSize. 3. Updated Section 3.3: Added xUD_Select, bTriggerMode.	03/31/2023
1.5.0	1. Updated Section 3.1: Updated the bEdgeSelect description. 2. Added general description of instructions to Chapter 3, Chapter 4, Chapter 6, and Chapter 10. 3. Updated Chapter 6: Updated supported products. 4. Updated Section 6.13: Updated the bUpdate description. 5. Updated Chapters 7 and 8: Updated the description of the command library. 6. Updated Section 8.6: Added a new instruction DFB_SetIPConfig. 7. Added Sections 12.1 and 12.2: SNTPGetUTCTime and SNTPServer instructions. 8. Added Chapter 13: Added util library–related instructions. 9. Added Chapter 14: Added instructions related to the MEMUtils library.	09/30/2023
1.6.0	1. Updated the supported products and libraries of instructions. 2. Added Section 1.3: Added DFC_Clear instruction. 3. Updated Section 1.4: Added DFC_Clear error codes and troubleshooting. 4. Updated Section 6.21: Added AS08AD_B and AS08AD_C modules. 5. Added Section 6.22: Added DFB_SCMRS instruction. 6. Updated Section 6.23: Added DFB_SCMRS error codes and troubleshooting. 7. Updated Section 8.6: Updated DFB_SetIPConfig. 8. Updated Section 11.2: Changed the image of Example. 9. Removed Section 11.3: Removed DFB_FTPClient instruction. 10. Added Section 14.2: Added MemSet instruction. 11. Added Chapter 15: Added SysTimeRtc library instructions. 12. Added Chapter 16: Added DL_SysInfo library instructions.	03/29/2024
1.7.0	1. Updated Section 3.1: Updated the note of bEdgeSelect and bCycle. 2. Updated Section 3.2: Updated the function block figure and the setting values of OutputAction and diTableSize. 3. Updated Section 3.4: Update the figure of the programming example. 4. Updated Section 3.5: Update the figure of the programming example. 5. Updated Section 3.7: Updated the original section 3.7 Error Code and Troubleshooting to section 3.9. 6. Added Section 3.7 and Section 3.8: Added DFB_Capture_EncoderAxis and DFB_Compare_EncoderAxis instructions.	09/30/2024

Version	Revision	Date
	7. Updated Section 6.1–6.2, 6.4, 6.20–6.21: Updated the notes. 8. Updated Section 6.3: Updated the note and the timing figure and description. 9. Updated Section 6.22: Updated the description of DFB_SCMRS_RXMODE and updated the programming example and the figure. 10. Updated Section 8.1: Updated the note. 11. Updated Section 8.2: Add the note. 12. Updated Section 8.5: Updated the output scope of uiReadLen and uiWriteLen. 13. Updated Section 8.7: Updated a description. 14. Updated Section 10.1: Added a new example. 15. Updated Section 14.1–14.2: Updated the programming examples. 16. Updated Section 15.1–15.2: Updated the programming examples. 17. Updated Section 16.1–16.2: Updated the programming examples.	
1.8.0	1. Updated Section 4.1–4.8: Added a parameter ECAT_Master and updated example content. 2. Added Section 6.23–6.24: Added DFB_IOLINKR and DFB_IOLINKW instructions. 3. Updated Section 6.23: Renumbered 6.23 to 6.25 and added IOLINK error codes and troubleshooting methods. 4. Added Chapter 17: Added DFB_PIDE instruction. 5. Added Chapter 18: Added CODESYS basic instructions.	04/28/2025

TOC

P1 Preface

P1.1 Introduction

This manual introduces Delta self–developed function blocks and functions for customers to perform PLC application development.

P1.1.1 Applicable Products

This manual applies to the following products:

- AX-3 series
- AX-8 series
- AX-C series

P1.1.2 Associated Manuals

1. **DIADesigner–AX User Manual**

   Includes the information of software operation, programming languages (Ladder Diagram, Sequential function charts, ST (Structured Text) and function blocks), concept of POU and Task, as well as motion control programming.

2. **AX-3 Series Operation Manual**

   Introduces the concept of motion control system, while gives the information of hardware and software configuration, motion control programming framework, troubleshooting, analog input–output module and temperature measurement module.

3. **AX-8 Series Operational Manual**

   Introduces the concept of motion control system, while gives the information of hardware and software configuration, motion control programming framework, troubleshooting, analog input–output module and temperature measurement module.

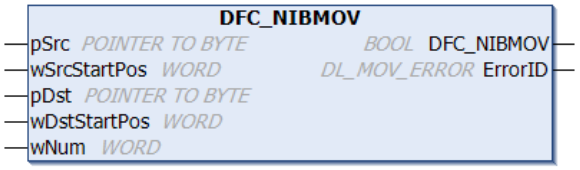
4. **AX-C Series Operational Manual**

   Introduces the concept of motion control system, while gives the information of hardware and software configuration, motion control programming framework, troubleshooting, analog input–output module and temperature measurement module.

Chapter 1: Move Instructions

1.1. DFC_NIBMOV

DFC_NIBMOV shifts the variable values.

FB/FC	Instruction	Graphic Expression	ST Language
FC	DFC_NIBMOV		DFC_NIBMOV(pSrc:= , wSrcStartPos:= , pDst:= , wDstStartPos:= , wNum:= , ErrorID=>);

- **Inputs**

Name	Function	Data Type	Setting Value (Default value)
pSrc	Memory address of source variables	POINTER TO BYTE	Memory address (0)
wSrcStartPos	Start address for source variable shift (Unit: Nibble)	WORD*	Positive integer (0)
pDst	Memory address of target variables	POINTER TO BYTE	Memory address (0)
wDstStartPos	Start address for storing target variables (Unit: Nibble)	WORD*	Positive integer (0)
wNum	The data length for data shift (Unit: Nibble)	WORD*	1–256 Positive integer (0)

***Note**: The variable types BYTE and WORD can be used for inputs.

- **Outputs**

Name	Function	Data Type	Output Range (Default value)
DFC_NIBMOV	Instruction running result (The parameter is returned)	BOOL	TRUE/FALSE (FALSE)
ErrorID	Error code	DL_MOV_ERROR	DL_MOV_ERROR (DFC_NO_ERROR)

- **Function**

  After running this Function, the value of variable1 (*pSrc*) will be copied to variable2 (*pDst*), while the length of copied data is determined by *wNum* input (Unit: Nibble).

- **Programming Example**

  - **Example 1**

    In this example, FC instruction (DFC_NIBMOV) is used for shifting the content of wVar0 (*pSrc*) to the variable wVar1 (*pDst*).

```
PROGRAM PLC_PRG
VAR
    bVar0: BOOL;
    Length: WORD:=2;
    wVar0: WORD : 16#1234;
```

```
            wVarl: WORD : 16#FFFF;
END_VAR

IF bVar0 THEN
   DFC_NIBMOV(
       pSrc:= ADR(wVar0),
       wSrcStartPos:= 2,
       pDst:= ADR(wVar1),
       wDstStartPos:= 0,
       wNum: = Length,
       ErrorID=> );
    bVarO:=FALSE;
END_IF;
```

Since *wSrcStartPos*=2, *wNum*=2, and *wDstStartPos*=0, two consecutive Nibbles (Length=2), which start from Nibble2 of variable wVar0 (*pSrc*), are shifted to the address Nibble0 inside the memory of wVar1 (*pDst*).

Variable	Nibble3	Nibble2	Nibble1	Nibble0
wVar0 [16#1234]	Memory content			
	1	2	3	4
wVar1 [16#FFFF]	Memory content (before running FC)			
	F	F	F	F
wVar1 [16#FF12]	Memory content (after running FC)			
	F	F	1	2

- **Example 2**

  In this example, FC instruction (DFC_NIBMOV) is used for shifting the content of ar_wVar0 (*pSrc*) to the variable ar_wVar1 (*pDst*).

```
PROGRAM POU
VAR
   bVar0: BOOL;
   Length: WORD:=2;
   ar_wVar0: ARRAY[0..1] OF WORD:= [16#0123,16#4567];
   ar_wVar1: ARRAY[0..1] OF WORD:= [16#FFFF,16#FFFF];
END_VAR

IF bVar0 THEN
   DFC_NIBMOV(
   pSrc:= ADR(ar_wVar0),
   wSrcStartPos:= 3,
   pDst:= ADR(ar_wVarl),
   wDstStartPos:= 0,
   wNum:= Length,
   ErrorID=> );
bVarO:=FALSE;
END_IF;
```

Since *wSrcStartPos*=3, *wNum*=2, and *wDstStartPos*=0, two consecutive Nibbles (Length=2), which start from Nibble3 of variable ar_wVar0 (*pSrc*), are shifted to the address Nibble0 inside the memory of ar_wVar1 (*pDst*).

Variable	Nibble7	Nibble6	Nibble5	Nibble4	Nibble3	Nibble2	Nibble1	Nibble0
ar_wVar0 [16#0123,16#4567]	Memory Content							
	4	5	6	7	0	1	2	3

Variable	Nibble7	Nibble6	Nibble5	Nibble4	Nibble3	Nibble2	Nibble1	Nibble0
ar_wVar1 [16#FFFF,16#FFFF]	**Memory Content**							
	F	F	F	F	F	F	F	F
ar_wVar1 [16#FF70,16#FFFF]	**Memory Content (after running FC)**							
	F	F	F	F	F	F	7	0

- **Supported Devices**
  - AX series
- **Library**
  - DL_Mov.library

1.2. DFC_XCH

DFC_XCH exchanges between two variable values.

FB/FC	Instruction	Graphic Expression	ST Language
FC	DFC_XCH		DFC_XCH(pSrc1:= , pSrc2:= , dwNum:= , ErrorID=>);

- **Inputs**

Name	Function	Data Type	Setting Value (Default value)
pSrc1	Memory address of variable1	POINTER TO BYTE	Memory address (0)
pSrc2	Memory address of variable2	POINTER TO BYTE	Memory address (0)
dwNum	The length of data for exchange (Unit: Byte)	DWORD*	1–65535 positive integer (0)

***Note**: The BYTE, WORD, and DWORD variable types can be used for dwNum input.

- **Outputs**

Name	Function	Data Type	Output Range (Default value)
DFC_XCH	Instruction running result (This parameter is returned)	BOOL	TRUE/FALSE (FALSE)
ErrorID	Error code	DL_MOV_ERROR	DL_MOV_ERROR (DFC_NO_ERROR)

- **Function**

  After running this Function, the values of variable1 (*pSrc1*) and variable2 (*pSrc2*) will be exchanged, while the length of exchanged data is determined by *dwNum* input.

- **Programming Example**

  In this example, Function (DFC_XCH) is used for exchanging contents of two variables.

```
PROGRAM PLC_PRG
VAR
   bVar0,bVarl:BOOL;
   Length:DWORD:=1;
   wVar0:WORD:=16#1234;
   wVarl:WORD:=16#5678;
END_VAR

IF bVar0 THEN
   bVarl:=DFC_XCH(pSrcl:= ADR(wVar0), pSrc2:=ADR(wVarl) , dwNum:=Length , ErrorID=> )
   bVar0:=FALSE;
END_IF;
```

Since the data length for data exchange is set to one Byte (Length=1), low–byte of variable1 and 2 will be exchanged after running the Function (DFC_XCH).

Before ruuning the Function				After running the Function		
Variable	wVar0	wVar1		Variable	wVar0	wVar1
Content	16#1234	16#5678		Content	16#1278	16#5634

- **Supported Devices**
  - AX series
- **Library**
  - DL_Mov.library

1.3. DFC_Clear

DFC_Clear clears the variable values.

FB/FC	Instruction	Graphic Expression	ST Language
FC	DFC_Clear		DFC_Clear(In:= , ErrorID=>);

- **Inputs**

Name	Function	Data Type	Setting Value (Default value)
In	Clear the value.	Any	–

- **Outputs**

Name	Function	Data Type	Output Range (Default value)
DFC_Clear	Instruction running result (This parameter is returned.)	BOOL	TRUE/FALSE (FALSE)
ErrorID	Error code	DL_MOV_ERROR	DL_MOV_ERROR (DFC_NO_ERROR)

- **Function**
  - This function is supported by DL_Mov V1.0.6.1 or later.
  - When this instruction is running, the value of the variable (*In*) is cleared to 0 instead of the initial value declared by the variable.
  - Supported variable types:

	BOOL	Bit strings				Integers								Real Numbers		Times, durations, dates, and text strings					POINTER	REFERENCE
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING		
In	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	x	x
	Support an enumeration, array, array element, struct, or struct member.																					

- **Programming Example**

  This example uses DFC_Clear to clear the variable value.

1. Declare Structure

```
TYPE Struct1 :
STRUCT
   rl :REAL := 0.11;
   dil :DINT := –11;
   udil :UDINT := 11;
   DT1 :DT := DT#2024–1–12–1:42:50;
END_ STRUCT
END_ TYPE
```

2. When bCallFunction is TRUE, clear the value entered by DFC_Clear of the variable.

```
PROGRAM Use_Lib
VAR
   arvalue: ARRAY [1..10] OF DWORD;
   structValue: Structl;
   bCallFunction: BOOL;
   dwReturn: BOOL;
   TmpErrorld: DL_Mov.DL_M0V_ERR0R;
END_VAR

IF bCallFunction THEN
   dwReturn := DFC_Clear(In :=arValue, ErrorlD => TmpErrorld);
   dwReturn := DFC_Clear{In := structValue, ErrorlD => TmpErrorld);
   bCallFunction:=FALSE;
END_IF
```

- **Supported Devices**
  - AX series
- **Library**
  - DL_Mov.library

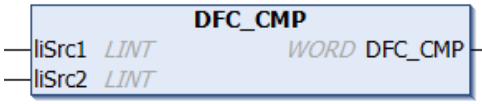
1.4. Error Code and Troubleshooting

Description	Reasons for error	Troubleshooting
DFC_NIBMOV_ERR_PARAMETER	Incorrect value of *wNum*	Check if the value of *wNum* is bigger than 0.
DFC_XCH_ERR_PARAMETER	Incorrect value of *dwNum*	Check if the value of *dwNum* is bigger than 0.
DFC_XCH_ERR_NOMEMORY	Not enough memory space in the controller	Check if the size of downloaded program exceeds the limit, then reboot the controller.
DFC_CLEAR_ERR_PAR	Invalid input	Check the input.
DFC_CLEAR_ERR_DA	Unsupported variable data type	Check the variable data type of Input.

Chapter 2: Comparison Instructions

2.1. DFC_CMP

DFC_CMP compares LINT variable values.

FB/FC	Instruction	Graphic Expression	ST Language
FC	DFC_CMP		DFC_CMP(liSrc1:= , liSrc2:=)

- **Inputs**

Name	Function	Data Type	Setting Value (Default value)
liSrc1	Variable1	LINT*	LINT:-2^{63}–2^{63}–1 (0)
liSrc2	Variable2	LINT*	LINT:-2^{63}–2^{63}–1 (0)

***Note**: The variable types SINT, INT, DINT, and LINT can be used for inputs.

- **Outputs**

Name	Function	Data Type	Output Range (Default value)
DFC_CMP	Instruction running result (The parameter is returned.)	WORD	1:liSrc1 = liSrc2 2:liSrc1 < liSrc2 3:liSrc1 > liSrc2 (0)

- **Function**

  The FC instruction is used to compare the values in variable 1 with that in variable 2.

- **Programing Example**

  This example use FC instruction (DFC_CMP) to compare between two variable values.

```
PROGRAM PLC_PRG
VAR
   liVar0: LINT :=1000;
   liVarl: LINT :=2000;
   wVarO: WORD;
END_VAR

wVar0:=DFC_CMP(liSrcl:=liVar0 , liSrc2:=liVarl );
```

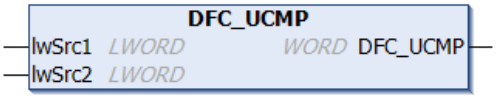
  Since variable1 (liVar0) is smaller than variable2 (liVar1), the calculation result (wVar0) will be 2.

- **Supported Devices**
  - AX series

- **Library**
  - DL_Comparison.library

2.2. DFC_UCMP

DFC_UCMP compares ULINT variable values.

FB/FC	Instruction	Graphic Expression	ST Language
FC	DFC_UCMP		DFC_UCMP(lwSrc1:= , lwSrc2:=)

- **Inputs**

Name	Function	Data Type	Setting Value (Default Value)
lwSrc1	Variable 1	ULINT/LWORD*	ULINT/LWORD:0~2^{64}–1 (0)
lwSrc2	Variable 2	ULINT/LWORD*	ULINT/LWORD:0~2^{64}–1 (0)

***Note**: The variable types USINT, UINT, UDINT, ULINT, BYTE, WORD, DWORD, and LWORD can be used for inputs.

- **Outputs**

Name	Function	Data Type	Output Range (Default Value)
DFC_UCMP	Instruction running result (The parameter is returned.)	WORD	1:lwSrc1 = lwSrc2 2:lwSrc1 < lwSrc2 3:lwSrc1 > lwSrc2 (0)

- **Function**

  The FC instruction is used to compare the values in variable 1(*lwSrc1*) with that in variable 2(*lwSrc2*).

- **Programming Example**

  This example use FC instruction (DFC_UCMP) to compare between two variable values.

```
PROGRAM PLC_PRG
VAR
   uiVar0: UINT :=1000;
   uiVarl: UINT :=2000;
   wVar0: WORD;
END_VAR

wVar0:=DFC_UCMP(lwSrcl:=uiVar0, lwSrc2:=uiVarl );
```

  Since variable1 (uiVar0) is smaller than variable2 (uiVar1), the calculation result (wVar0) will be 2.

- **Supported Devices**

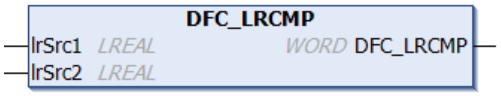
  - AX series

- **Library**

  - DL_Comparison.library

2.3. DFC_LRCMP

DFC_LRCMP compares LREAL variable values.

FB/FC	Instruction	Graphic Expression	ST Language
FC	DFC_LRCMP		DFC_LRCMP(lrSrc1:= , lrSrc2:=)

- **Inputs**

Name	Function	Data Type	Setting Value (Default Value)
lrSrc1	Variable 1	LREAL*	LREAL:–1.7976931348623157E+308¯1.7976931348623157E+308 (0)
lrSrc2	Variable 2	LREAL*	LREAL:–1.7976931348623157E+308¯1.7976931348623157E+308 (0)

***Note**: The variable types REAL and LREAL can be used for inputs.

- **Outputs**

Name	Function	Data Type	Output Range (Default Value)
DFC_LRCMP	Instruction running result (The parameter is returned)	WORD	1:lrSrc1 = lrSrc2 2:lrSrc1 < lrSrc2 3:lrSrc1 > lrSrc2 (0)

- **Function**

  The FC instruction is used to compare the values in variable 1(lrSrc1) with that in variable 2(lrSrc2).

- **Programming Example**

  This example use FC instruction (DFC_LRCMP) to do comparison between two variable values.

```
PROGRAM PLC_PRG
VAR
   lrVar0: LREAL :=10.5;
   lrVarl: LREAL :=33.5;
   wVar0: WORD;
END_VAR

wVar0:=DFC_LRCMP(lrSrcl:=lrVar0 , lrSrc2:=lrVarl );
```

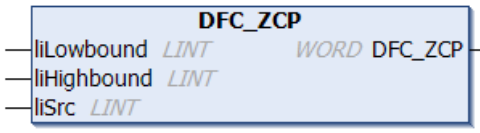
  Since variable1 (lrVar0) is smaller than variable2 (lrVar1), the calculation result (wVar0) will be 2.

- **Supported Devices**
  - AX series

- **Library**
  - DL_Comparison.library

2.4. DFC_ZCP

DFC_ZCP compares the LINT variable value with a range of values.

FB/FC	Instruction	Graphic Expression	ST Language
FC	DFC_ZCP		DFC_ZCP(liLowbound:= , liHighbound:= , liSrc:=);

- **Inputs**

Name	Function	Data Type	Setting Value (Default Value)
liLowbound	Lower value	LINT*	LINT: -2^{63}–$2^{63}-1$ (0)
liHighbound	Upper value	LINT*	LINT: -2^{63}–$2^{63}-1$ (0)
liSrc	Variable	LINT*	LINT: -2^{63}–$2^{63}-1$ (0)

***Note**: The variable type SINT, INT, DINT and LINT can be used for inputs.

- **Outputs**

Name	Function	Data Type	Output Range (Default Value)
DFC_ZCP	Instruction running result (The parameter is returned.)	WORD	1: liSrc < Lower value 2: Lower value < liSrc < Upper value 3: liSrc > Upper value (0)

- **Function**

  This FC instruction is used to compare the values in variable (*liSrc*) with the high bound and low bound values.

- **Programming Example**

  This example uses FC instruction (DFC_ ZCP) to compare variable values with the high bound and low bound values.

```
PROGRAM PLC_PRG
VAR
   L_liVar: LINT :=1;
   H_liVar: LINT :=99;
   liVar0: LINT :=120;
   wVar0: WORD;
END_VAR

wVar0:=DFC_ZCP(liLowbound:=L_liVar , liHighbound:=H_liVar , liSrc:=liVar0 );
```

  Since the value in variable (*liVar0*) is larger than the upper value (H_liVar), the calculation result (wVar0) is 3.

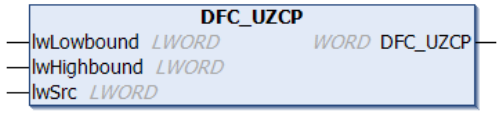
- **Supported Devices**

  - AX series

- **Library**
  - DL_Comparison.library

2.5. DFC_UZCP

DFC_UZCP compares the ULINT variable value with a range of values.

FB/FC	Instruction	Graphic Expression	ST Language
FC	DFC_UZCP		DFC_UZCP(lwLowbound:= , lwHighbound:= , lwSrc:=);

- **Inputs**

Name	Function	Data Type	Setting Value (Default Value)
lwLowbound	Lower value	ULINT/LWORD*	ULINT/LWORD: 0–2^{64}–1 (0)
lwHighbound	Upper value	ULINT/LWORD*	ULINT/LWORD: 0–2^{64}–1 (0)
lwSrc	Variable	ULINT/LWORD*	ULINT/LWORD: 0–2^{64}–1 (0)

***Note**: The variable type USINT, UINT, UDINT, ULINT, BYTE, WORD, DWORD, and LWORD can be used for inputs.

- **Outputs**

Name	Function	Data Type	Output Range (Default Value)
DFC_UZCP	Instruction running result (The parameter is returned)	WORD	1: lwSrc < Lower value 2: Lower value < lwSrc < Upper value 3: lwSrc > Upper value (0)

- **Function**

  The FC instruction is used to compare the values in variable (*lwSrc*) with the high bound and low bound values.

- **Programming Example**

  This example uses FC instruction (DFC_ UZCP) to compare variable values with the high bound and low bound values.

```
PROGRAM PLC_PRG
VAR
   L_ulVar: UINT :=1;
   H_ulVar: UINT :=99;
   ulVar0: ULINT :=120;
   wVar0: WORD;
END_VAR,

wVar0:=DFC_UZCP(lwLowbound:=L_ulVar , liHighbound:=H_ulVar , lwSrc:=ulVar0);
```

  Since the value in variable (*ulVar0*) is larger than the upper value (*H_ulVar*), the calculation result (*wVar0*) is 3.

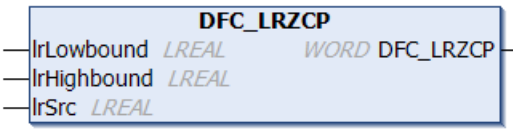
- **Supported Devices**

  - AX Series

- **Library**
  - DL_Comparison.library

2.6. DFC_LRZCP

DFC_LRZCP compares the LREAL variable value with a range of values.

FB/FC	Instruction	Graphic Expression	ST Language
FC	DFC_LRZCP		DFC_LRZCP(lrLowbound:= , lrHighbound:= , lrSrc:=);

- **Inputs**

Name	Function	Data Type	Setting Value (Default Value)
lrLowbound	Lower value	LREAL*	LREAL:–1.7976931348623157E+308~1.7976931348623157E+308 (0)
lrHighbound	Upper value	LREAL*	LREAL:–1.7976931348623157E+308~1.7976931348623157E+308 (0)
lrSrc	Variable	LREAL*	LREAL:–1.7976931348623157E+308~1.7976931348623157E+308 (0)

***Note**: The variable types REAL and LREAL can be used for inputs.

- **Outputs**

Name	Function	Data Type	Output Range (Default Value)
DFC_LRZCP	Instruction running result (The parameter is returned.)	WORD	1:lrSrc < Lower value 2: Lower value < lrSrc < Upper value 3:lrSrc > Upper value (0)

- **Function**

  The FC instruction is used to compare the values in variable (lrSrc) with the high bound and low bound values.

- **Programming Example**

  This example uses the FC instruction (DFC_LRZCP) to compare variable values with the high–bound and low–bound values.

```
PROGRAM PLC_PRG
VAR
   L_lrVar: LREAL :=1.5;
   H_lrVar: LREAL :=99.9;
   lrVar0: LREAL :=50.5;
   wVar0: WORD;
END_VAR

wVar0:=DFC_LRZCP(lrLowbound:=L_lrVar , lrHighbound:=H_lrVar , lrSrc:=lrVar0 );
```

  Since the value in variable (lrVar0) is smaller than the upper value (H_lrVar) and larger than the lower value (L_lrVar), the calculation result (wVar0) is 2.

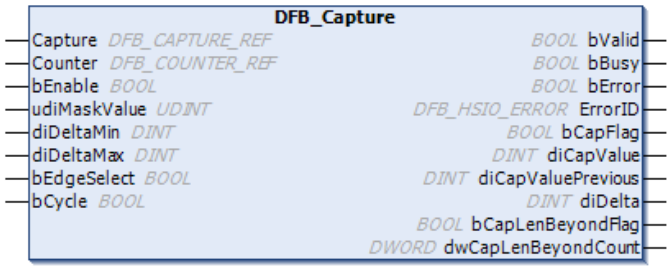
- **Supported Devices**
  - AX series
- **Library**
  - DL_Comparison.library

Chapter 3: Timers and Counters Instructions

FB/FC	Description	Library
DFB_Capture	Triggers and captures the commanded pulses of the specified high-speed counter according to the specified external signal source	DL_BuiltInIO_AX3 / DL_BuiltInIO
DFB_Compare	Compares the designated source and setting values and then sets or resets the desired device when the comparison result is TRUE.	DL_BuiltInIO_AX3 / DL_BuiltInIO
DFB_HCnt	Enables the high-speed counter and monitors the counter value.	DL_BuiltInIO_AX3 / DL_BuiltInIO
DFB_HTmr	Enables the high-speed timer to count the frequency wave bandwidth and monitors and timed value.	DL_BuiltInIO_AX3 / DL_BuiltInIO
DFB_PresetValue	The function block for high-speed counters. It can reset the current counter value to the default value.	DL_BuiltInIO_AX3 / DL_BuiltInIO
DFB_Sample	The function block for high-speed counters. It reads the increasing and decreasing number of the counter value during the sampling cycle.	DL_BuiltInIO_AX3 / DL_BuiltInIO

3.1. DFB_Capture

DFB_Capture captures the commanded pulses of the specified high-speed counter according to the designated external trigger.

FB/FC	Instruction	Graphic Expression	ST Language
FB	DFB_Capture		DFB_Capture_instance(Capture :=, Counter :=, bEnable :=, uiMaskValue :=, diDeltaMin :=, diDeltaMax :=, bEdgeSelect :=, bCycle :=, bValid =>, bBusy =>, bError =>, ErrorID =>, bCapFlag =>, diCapValue =>, diCapValuePrevious =>, diDelta =>, bCapLenBeyondFlag =>, dwCapLenBeyondCount =>);

- **Inputs**

Name	Function	Data Type	Setting Value (Default value)	Timing for Updating
Counter	Designate the source of the specified high-speed counter.	DFB_COUNTER_REF[*1]	DFB_COUNTER_REF (Cannot be null.)	–
bEnable	Run the instruction when *bEnable* changes to TRUE.	BOOL	TRUE/FALSE (FALSE)	–
uiMaskValue2	Define the mask range of Capture.	UINT	Positive number or 0 (0)	When *bEnable* shifts to TRUE, the setting parameters of *uiMaskValue* will be updated.
diDeltaMin[*3]	Define the minimum interval of each Capture[*2].	DINT	Positive number, negative number or 0 (0)	When *bEnable* shifts to TRUE and *bBusy* is FALSE

Name	Function	Data Type	Setting Value (Default value)	Timing for Updating
diDeltaMax[*3]	Define the maximum interval of each capture [*2].	DINT	Positive number, negative number or 0 (0)	When *bEnable* shifts to TRUE and *bBusy* is FALSE
bEdgeSelect[*4]	Define the rising edge or falling edge capture	BOOL	TRUE/FALSE (FALSE)	When *bEnable* shifts to TRUE, update the parameter of *bEdgeSelect*.
bCycle[*4]	Define whether capture values continuously	BOOL	TRUE/FALSE (FALSE)	When *bEnable* shifts to TRUE, update the parameter of *bCycle.*

***Note**:

1. DFB_Counter_REF (FB): This FB work as the driver interface of the high-speed counter . It calls the counter's parameters and the driver.
2. *uiMaskValue* is in the DL_BuiltInIO library, and UINT is changed to UDINT after v1.0.5.0.
3. Once *diDeltaMin* and *diDeltaMax* are set to 0, the system will not check whether the capture interval is within the correct range.
4. This parameter is only applicable to AX-332E.

- **Outputs**

Name	Function	Data Type	Output Range (Default value)
bValid	TRUE when the output value is valid	BOOL	TRUE (FALSE)
bBusy	TRUE when the instruction is enabled	BOOL	TRUE/FALSE (FALSE)
bError	TRUE if an error occurs	BOOL	TRUE/FALSE (FALSE)
ErrorID	Indicates the error code if an error occurs.	DFB_HSIO_ERROR*	DFB_HSIO_ERROR (DFB_HSIO_NO_ERR)
bCapFlag	Indicates that the current Capture is valid. (The flag shifts to TRUE for one scan cycle and will be reset immediately)	BOOL	TRUE/FALSE (FALSE)
diCapValue	The captured value	DINT	Positive number, negative number or 0 (0)
diCapValuePrevious	The previous captured value	DINT	Positive number, negative number or 0 (0)
diDelta	The difference between the previous and the current captured values.	DINT	Positive number, negative number or 0 (0)
bCapLenBeyondFlag	Indicates that a capture is failed. (The flag shifts to TRUE for one scan cycle and will be reset immediately)	BOOL	TRUE/FALSE (FALSE)

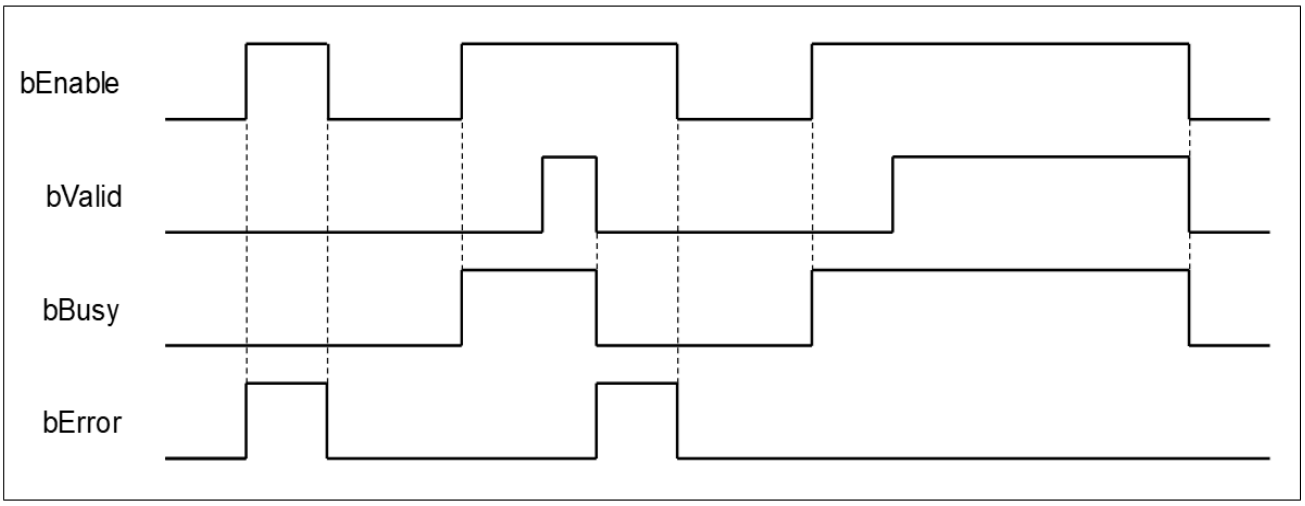
Name	Function	Data Type	Output Range (Default value)
dwCapLenBeyondCount	Counts the number of the failed Capture.	DWORD	Positive number or 0 (0)

***Note**: DFB_HSIO_ERROR: Enumeration (Enum)

- **Outputs Update Timing**

Name	Timing for shifting to TRUE	Timing for shifting to FALSE
bValid	When the values at the outputs are valid after *bEnable* being TRUE for one scan cycle	• When *bEnable* shifts to FALSE • When *bError* shifts to FALSE
bBusy	When *bEnable* is triggered by the rising edge	• When *bEnable* shifts to FALSE • When *bError* shifts to TRUE
bError	When an error occurs during the running or there are incorrect input values	When *bEnable* shifts from TRUE to FALSE (Error code is cleared)
ErrorID		
bCapFlag	Updates value continuously when *bValid* is TRUE.	Updates value continuously when *bValid* is TRUE.
diCapValue	Updates value continuously when *bValid* is TRUE.	Updates value continuously when *bValid* is TRUE.
diCapValuePrevious	Updates value continuously when *bValid* is TRUE.	Updates value continuously when *bValid* is TRUE.
diDelta	Updates value continuously when *bValid* is TRUE.	Updates value continuously when *bValid* is TRUE.
bCapLenBeyondFlag	Updates value continuously when *bValid* is TRUE.	Updates value continuously when *bValid* is TRUE.
dwCapLenBeyondCount	Updates value continuously when *bValid* is TRUE.	Updates value continuously when *bValid* is TRUE.

- **Timing Diagram**



- **Inputs/Outputs**

Name	Function	Data Type	Setting Value (Default value)	Timing for Updating
Capture	Refer to the source of the specified high-speed capture	DFB_CAPTURE_REF (FB)*	DFB_CAPTURE_REF (Cannot be null.)	When *bEnable* shifts to TRUE and *bBusy* is FALSE.

***Note**: DFB_CAPTURE_REF (FB): The I/O function block of the high-speed counter which contains parameter adjustment and the driver.

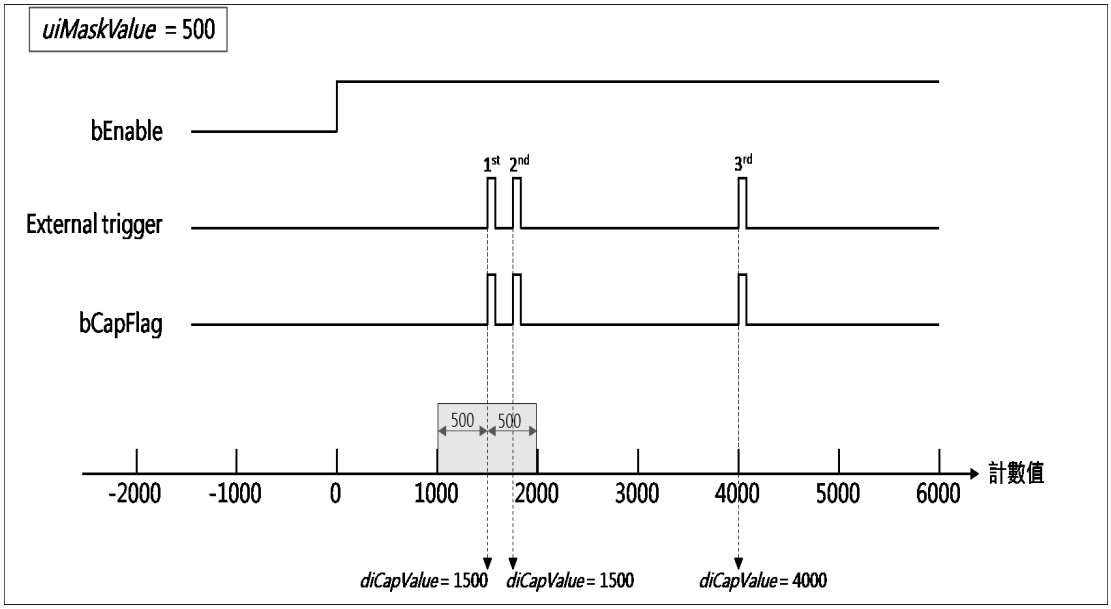
- **Function**
  - ***bEdgeSelect***

    This feature is supported on AX-3 series with firmware version V1.0.5.0 or later and with lilbrary DL_BuiltInIO V1.1.0.2 or later.

  - ***uiMaskValue***

    Refer to the following figure for the function description of *uiMaskValue* input.

    1. Set *uiMaskValue* to 500 and *bEnable* to TRUE, then Capture function is enabled. At the same time, the output *bValid* is TRUE and the first captured value will be the center value of the mask range. In addition, the next capture action will be invalid if the next captured value is within the mask range.
    2. In the figure below, the 1st capture happens in a –500–500 range and the captured value changes from 0 to 1500.
    3. The captured value 1500 becomes the new center of mask range. Therefore, the next captured value which locates from 1000 to 2000 (1000 < *diCapValue* < 2000) will be invalid. So when the 2nd capture is triggered (in the mask range), the captured value will remain as 1500.
    4. Since the 3rd capture is triggered outside of the mask range, the captured value will be updated to 4000.

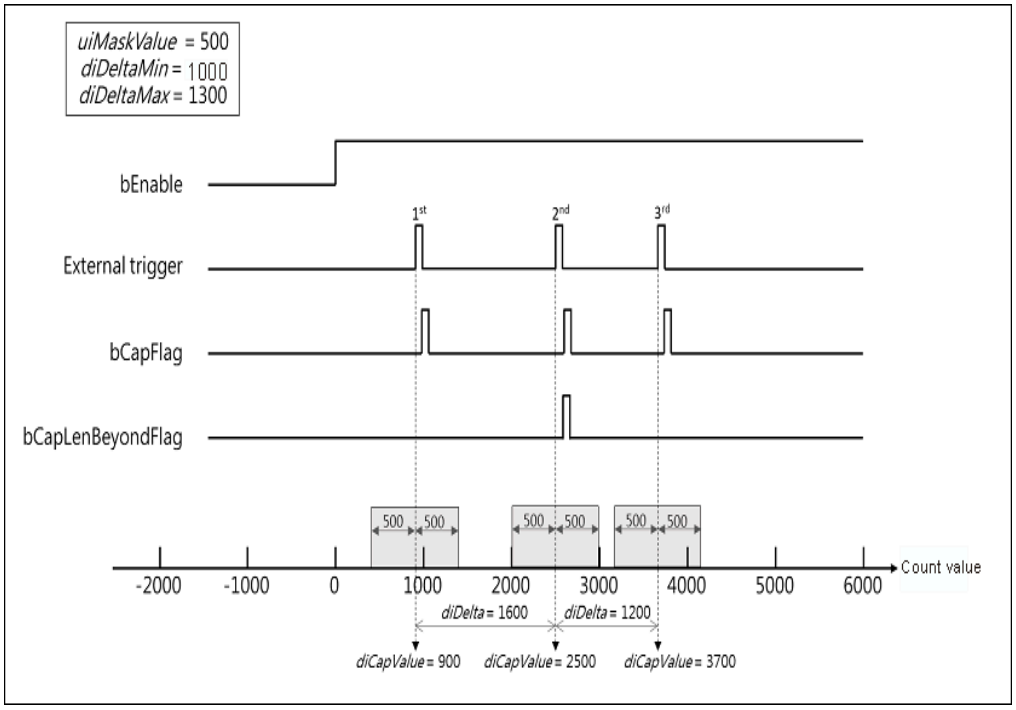
    

  - **diDeltaMin，diDeltaMax，bCapLenBeyondFlag，dwCapLenBeyondCount**

    DeltaMin/DeltaMax define the minimum and maximum distance between each Capture, while CapLenBeyondFlag and CapLenBeyondCount represent the error flag and the number of the failed Capture.

1. The function of *diDeltaMin*/*diDeltaMax* is to judge if a trigger mark is missed and the Capture is not Run. For example, if the value of DeltaMin is 1000 and DeltaMax is 1300, when the detected distance between 2 Capture exceeds 1000–1300, the system will flag this situation as trigger mark missing.
2. When a mark missing condition occurs, *bCapLenBeyondFlag* shifts to Ture for one scan cycle and will be reset immediately. At the same time *dwCapLenBeyondCount* counts 1.
3. Refer to the below diagram for the explanation of these inputs and outputs:

    - The mask range is between –500–500 and the 1st Capture occurs at 900.
    - The 2nd Capture occurs at 2500. Because DeltaMax is set to 1300 and DeltaMin is set to 1000 (1000–1300), the detected distance between two captures has exceeded the range of 1000–1300. Therefore, a trigger mark missing condition is flagged for a scan cycle, while *bCapLenBeyondFlag* remains as TRUE.
    - The 3rd Capture occurs at 3700. Because the difference between 3700 and the previous captured value 2500 is 1200, which is within the range of 1000–1300 (DeltaMin/DeltaMax), also 3700 is out of the mask range 2000–3000, the captured value changes to 3700 in this case, and *bCapLenBeyondFlag* will not change to TRUE.

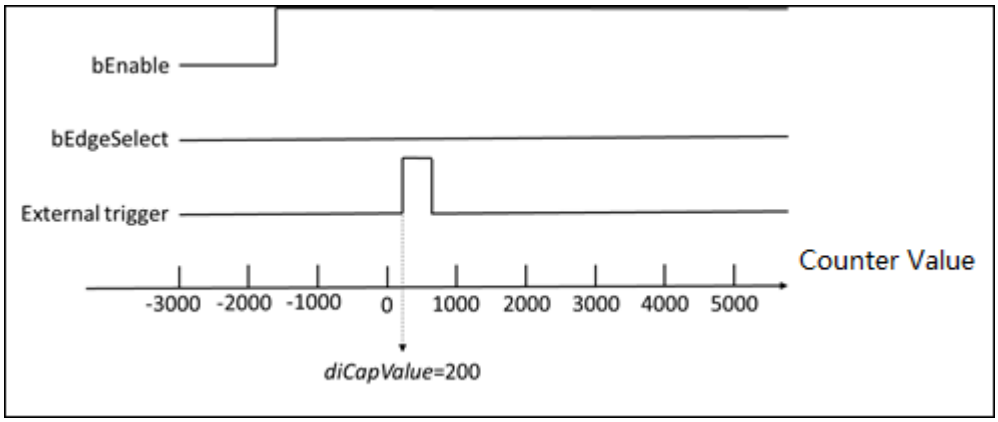


- ***bEdgeSelect***

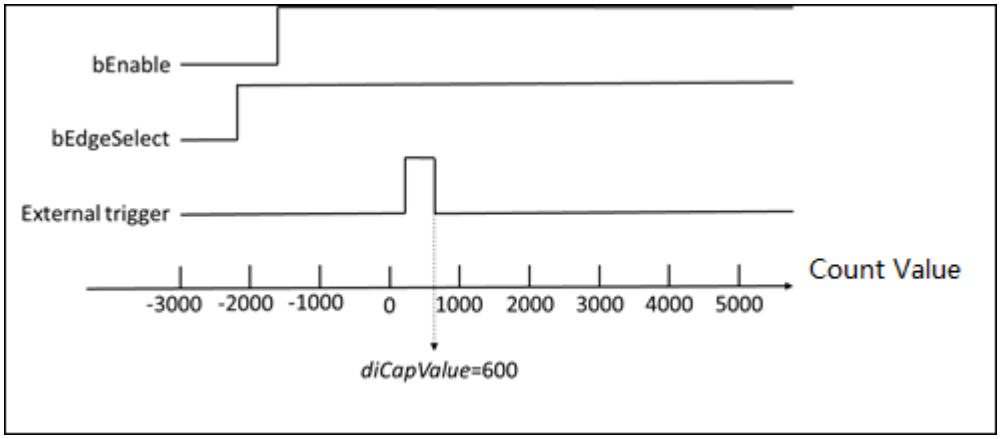
  bEdgeSelect defines whether the DI signal is triggered by the rising edge or the falling edge to capture the signal. This function is only applicable to AX-332E models.

  1. When *bEdgeSelect*=FALSE, the capture is triggered when DI generates the rising edge signal, and when *bEdgeSelect*=TRUE, the capture is triggered when DI generates the fallingr edge signal.
  2. Refer to the below diagram for the explanation of these inputs and outputs:

     When *bEdgeSelect*=FALSE:



When *bEdgeSelect*=TRUE:

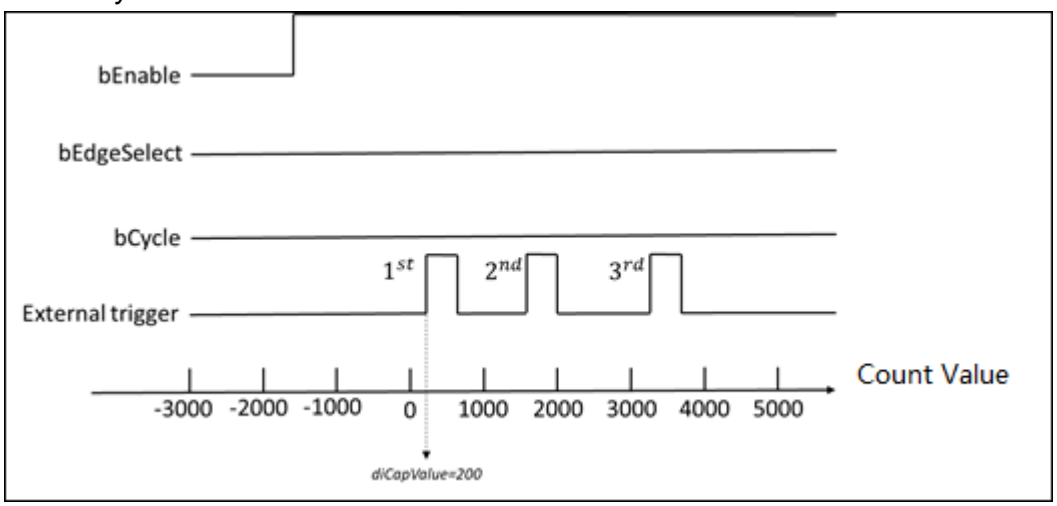


- ***bCycle***

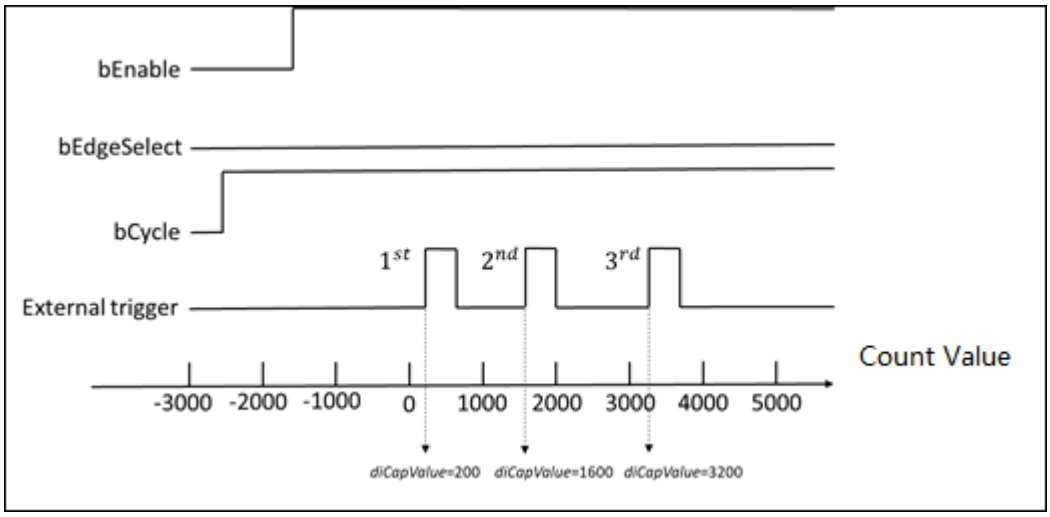
  The *bCycle* setting determines whether the Capture function block will capture values continuously. This function is only applicable to the AX-332EI.

  1. When *bCycle*=FALSE, the Capture function block can only capture the value once triggered by DI. When *bCycle*=TRUE, the Capture function block can capture multiple values for multiple times.
  2. Refer to the below diagram for the explanation of these inputs and outputs:

     When *bCycle*=FALSE



     When *bCycle*=TRUE



3. Only support AX-332E v1.0.4.2 or later.

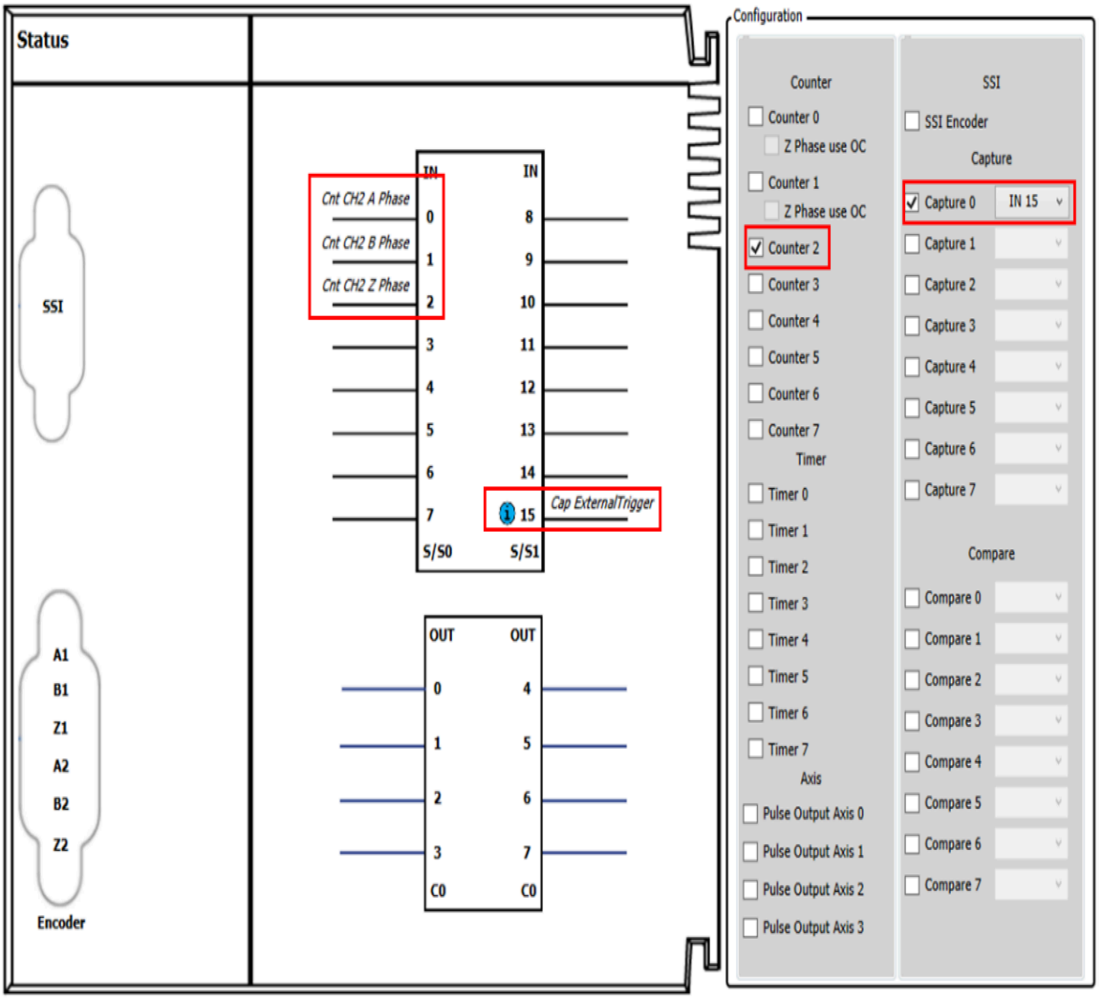
- **Troubleshooting**

  If an error occurs during the running of the instruction, *bError* will change to TRUE and Capture will stop. You can refer to ErrorID (Error Code) to address the problem.

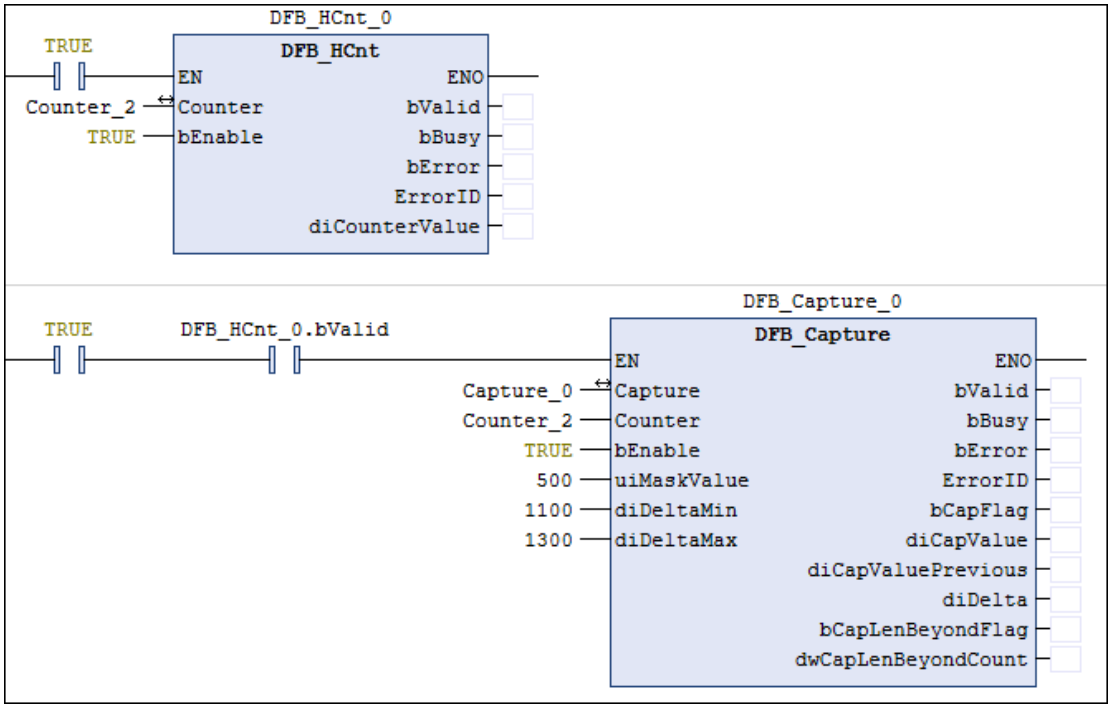
- **Programming Example**

  - This example uses DFB_HCnt and DFB_Capture in AX-308E to perform the Capture function.

    1. As the following figure shows, select a Counter and a Capture for Hardware IO Configuration in BuiltIn_IO and set the trigger source of Capture to a signal input on the hardware (e.g. IN15).



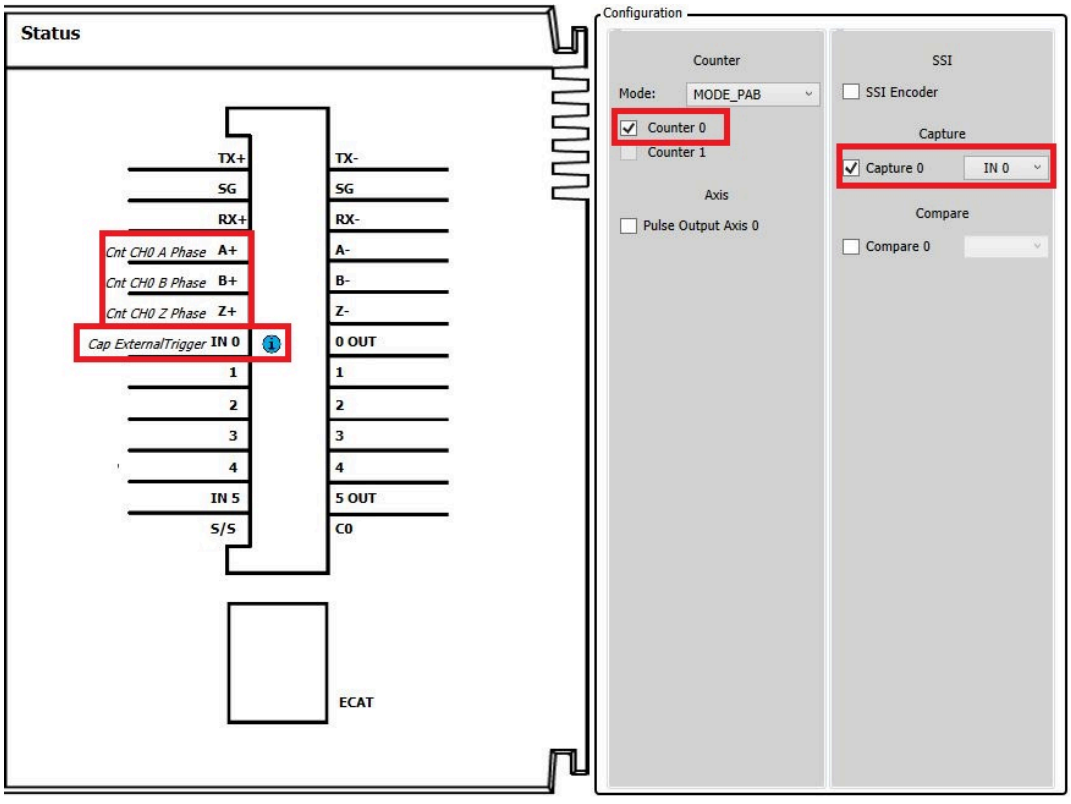
2. Enable the FB DFB_Capture (*bEnable* = TRUE) after using the FB DFB_HCnt to activate the high-speed counter (*bEnable* = TRUE) in the POU, then the present counter value will be captured and shown on the *diCapValue* output of DFB_Capture after the external signal (IN15) being triggered.



3. Refer to *AX-3 Series Operation Manual* for more details related to the settings and operation of **Hardware IO Configuration**.

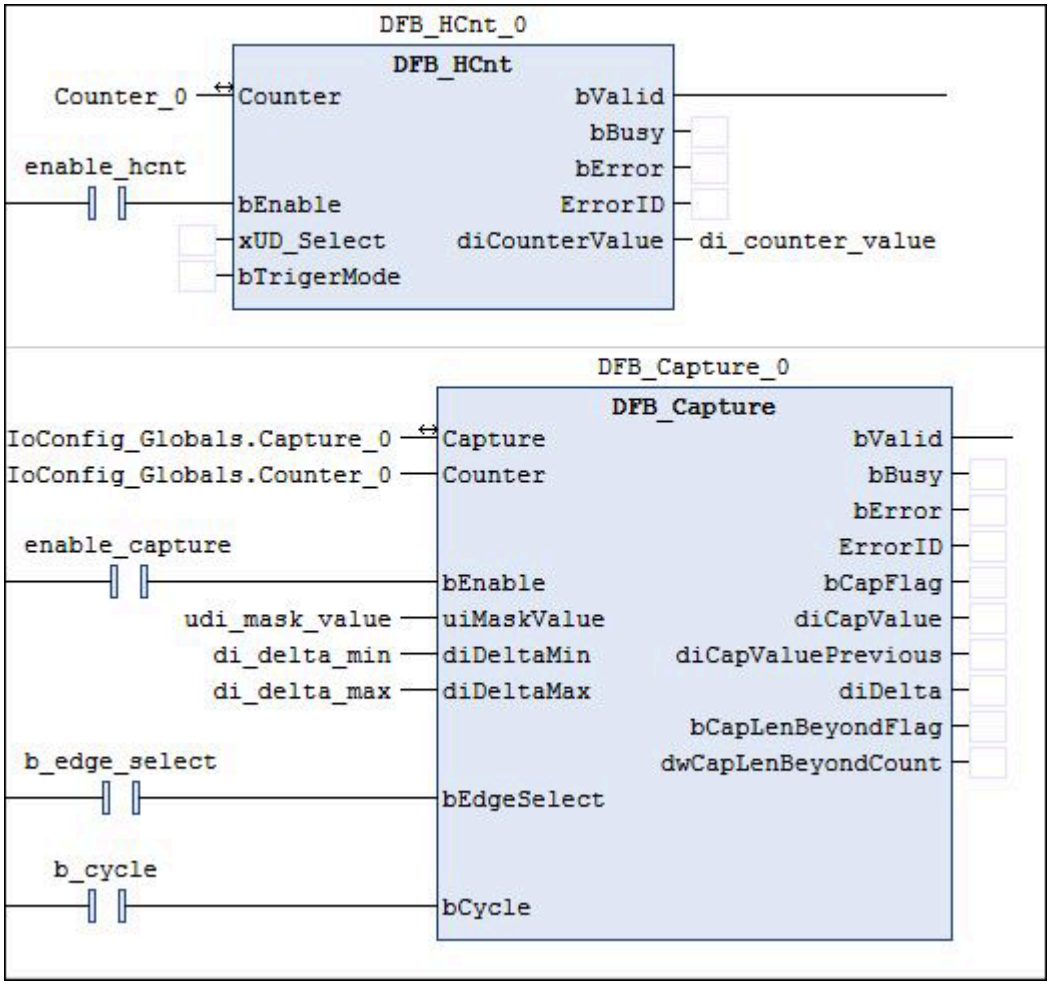
- This example uses DFB_HCnt and DFB_Capture in AX-308E to capture the high-speed counter.

1. As the following figure shows, select a Counter and a Capture for Hardware IO Configuration in BuiltIn_IO and set the trigger source of Capture to a signal input on the hardware (e.g. IN0).



2. Enable the FB DFB_Capture (*bEnable* = TRUE) after using the FB DFB_HCnt to activate the high-speed counter (*bEnable* = TRUE) in the POU, then the present counter value will be

captured and shown on the *diCapValue* output of DFB_Capture after the external signal (IN0) being triggered.



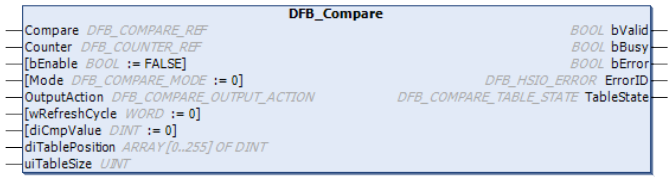
3. Refer to *AX-3 Series Operation Manual* for more details related to the settings and operation of **Hardware IO Configuration**.

- **Supported Devices**
  - AX-3 series (except for AX-300), AX-C
- **Library**
  - DL_BuiltInIO.library

    Note: From version 1.0.5.0, library DL_BuiltInIO_AX3 is changed to DL_BuiltInIO.

3.2. DFB_Compare

DFB_Compare compares the designated source value and the setting value and then sets or resets the device when the comparison result is TRUE.

FB/FC	Instruction	Graphic Expression	ST Language
FB	DFB_Compare		DFB_Compare_instance(Compare :=, Counter :=, bEnable :=, Mode :=, OutputAction :=, wRefreshCycle :=, diCmpValue :=, diTablePosition :=, uiTableSize :=, bValid =>, bBusy =>, bError =>, ErrorID => TableState=>);

- **Inputs**

Name	Function	Data Type	Setting Value (Default value)	Timing for Updating
Counter	Designates the source of high-speed counter.	DFB_COUNTER_REF*1	DFB_COUNTER_REF (Cannot be null.)	–
bEnable	Run the instruction when *bEnable* changes to TRUE.	BOOL	TRUE/FALSE (FALSE)	–
OutputAction*3	Define the output mode.	DFB_COMPARE_OUTPUT_ACTION	0: SET 1: RESET (SET)	When *bEnable* shifts to TRUE and *bBusy* is FALSE.
Mode	Comparison condition	DFB_COMPARE_MODE*2	0: Equal(=) 1: Bigger_Equal(≧) 2: Smaller_Equal(≦) (Equal)	When *bEnable* shifts to TRUE and *bBusy* is FALSE.

Name	Function	Data Type	Setting Value (Default value)	Timing for Updating
wRefreshCycle	Define the cycle time to refresh the status of the output device.	WORD	Positive number or 0 (0)	When *bEnable* shifts to TRUE and *bBusy* is FALSE.
diCmpValue*5	Specifies the compare value	DINT	Positive number, negative number or 0 (0)	When *bEnable* shifts to TRUE and *bBusy* is FALSE.
diTablePosition*4	Specify the compare value array	ARRAY OF DINT		When *bEnable* is rising edge and *bBusy* is FALSE.
diTableSize*4	Specify the compare value array scope	DINT	Positive number 0 to 255 (0)	When *bEnable* is rising edge and *bBusy* is FALSE.

***Note**:

1. DFB_Counter_REF (FB): This FB work as the driver interface of the high-speed counter . It calls the counter's parameters and the driver.
2. DFB_COMPARE_MODE: Enumeration (Enum). AX-332E only supports the 0: Equal (=) mode.
3. Except for AX-332E, other AX-3 series with DL_BuiltInIO_AX3 version V1.0.4.2 or later also supports this function.
4. This parameter is only applicable to AX-332E.
5. In AX-332E, *diCmpValue* will not refer to the value of this variable for comparison.

- **Outputs**

Name	Function	Data Type	Output Range (Default value)
bValid	TRUE when the output value is valid	BOOL	TRUE/FALSE (FALSE)
bBusy	TRUE when the instruction is enabled	BOOL	TRUE/FALSE (FALSE)
bError	TRUE if an error occurs	BOOL	TRUE/FALSE (FALSE)
ErrorID	Indicates the error code if an error occurs.	DFB_HSIO_ERROR*	DFB _HSIO_ERROR (DFB_HSIO_NO_ERR)
TableState	The hight–speed comparator state table	DFB_COMPARE_TABLE_STATE	–

***Note**: DFB _HSIO_ERROR: Enumeration (Enum)

- **DFB_COMPARE_TABLE_STATE**

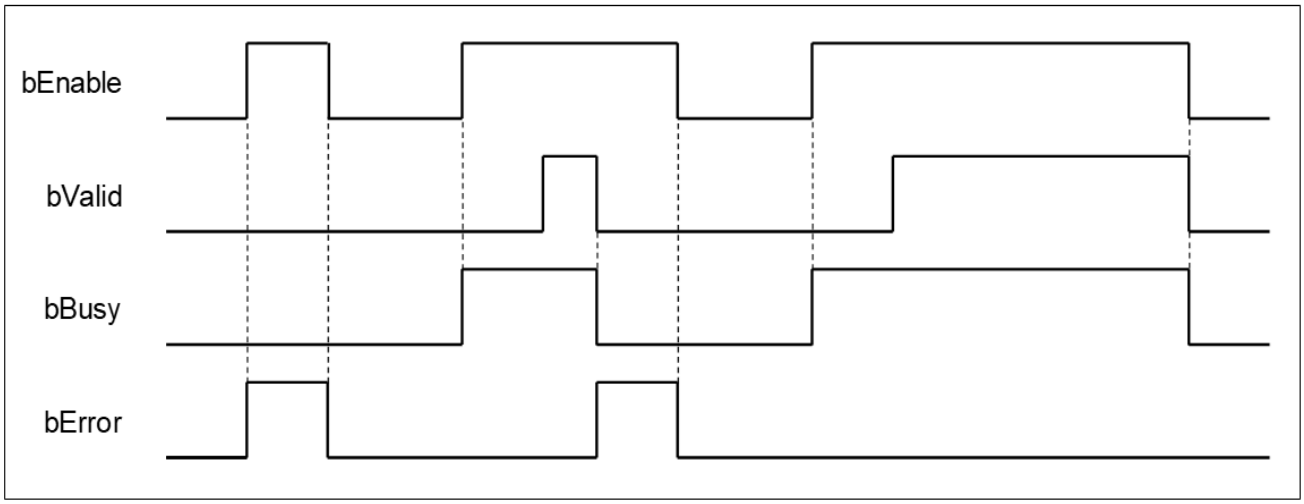
Name	Function	Data Type
xEmpty	The comparing table has not been filled in with the comparison items or all the high-speed comparison conditions are valid.	BOOL
xFull	The comparing table was filled with 256 comparison items and the high-speed compare was not established.	BOOL
xAlmostEmpty	The comparing table is almost empty.	BOOL

Name	Function	Data Type
uiDataCounter	The number of comparison items in the comparing table.	UINT
udiOSTriCounter	Count the number of times a high-speed comparison is triggered.	UDINT

- **Output Updating Timing**

Name	Timing for shifting to TRUE	Timing for shifting to FALSE
bValid	When *bEnable*is TRUE and the output values are valid after one scan cycle	• When *bEnable* shifts to FALSE • When *bError* shifts to TRUE
bBusy	When *bEnable* is triggered by the rising edge	• When *bEnable* shifts to FALSE • When *bError* shifts to TRUE
bError	When an error occurs during the running or there are incorrect input values	When *bEnable* shifts from TRUE to FALSE (Error code is cleared)
ErrorID		
TableState	Continuously updated when *bValid* is TRUE.	Continuously updated when *bValid* is TRUE.

- **Timing Diagram**



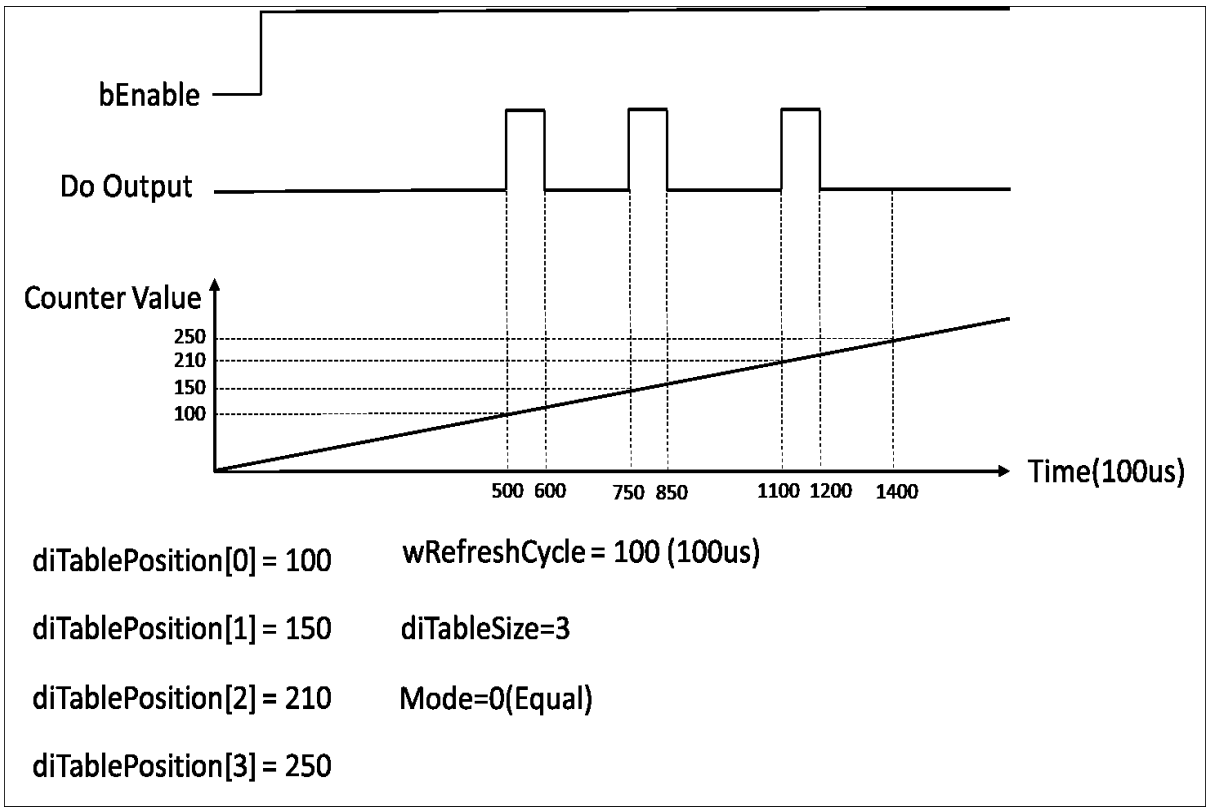
- **Inputs/Outputs**

Name	Function	Data Type	Setting Value (Default value)	Timing for Updating
Compare	Reference to the source of high-speed comparator.	DFB_COMPARE_REF (FB)*	DFB_COMPARE_REF (Cannot be null.)	When *bEnable* shifts to TRUE and *bBusy* is FALSE

***Note**: DFB_COMPARE_REF (FB): This FB work as the driver interface of the high-speed comparator . It calls the comparator's parameters and the driver.

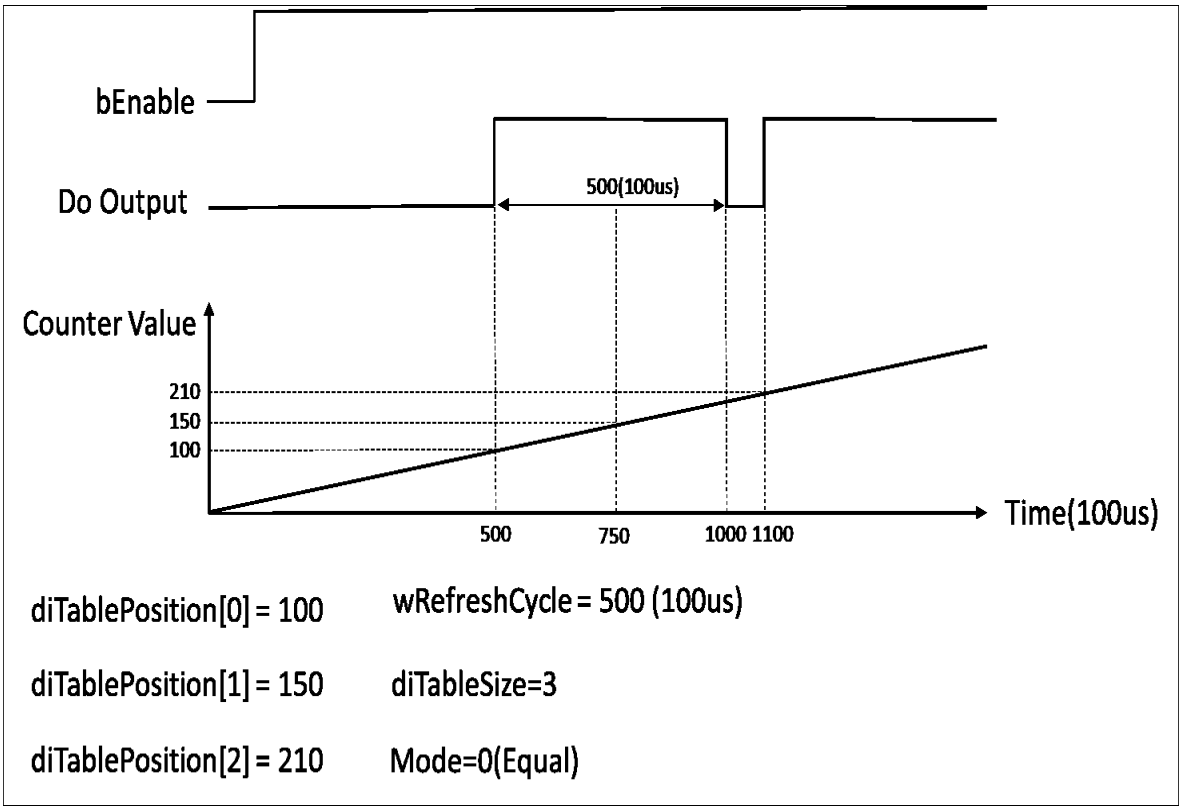
- **Function**
  - Cannot run this function block in **Ethercat_Task** in AX-332, or it will affect the sync cycle of other devices in Ethercat.
  - When the comparison result is TRUE (Counter Value = *diCmpValue*), DFB_Compare will output the results according to the settings of Hardware IO Configuration in BuiltIn IO.
  - When *bValid* output of DFB_Compare is TRUE, the comparator will continue to compare on the high-speed counter values. In case that the comparison condition is fulfilled and the output result is given according to the settings, the device will remain at a high–level signal and will not retrigger the output (TRUE → FALSE → TRUE) after the condition is fulfilled once again. If you need to reset the output device and change the high–level signal to low, find the following methods.
    - Define the variable at the output of Compare via I/O mapping in DIO, then set the output variable to falling–edge in the POU programming area so as to reset the output device.
    - Use the setting of *wRefreshCycle* to change the high–level signal to low automatically after the PLC keeps it at a high–level signal for a period of time.

      Either ways, the purpose of changing from high–level signals to low can be reached. However, the comparison conditions must be fulfilled again if you intend to make the output back to high–level.
  - The output device status can be refreshed by using the input *wRefreshCycle*. For example, set the value of *wRefreshCycle* to 10000(Unit: 0.1ms), then the designated output device will be pulled to a low level by the controller after the condition is fulfilled and remains a high–level output for one second. If *wRefreshCycle* is set to zero, the output device will keep at a high level without being reset.
  - *diTablePosition* is the numerical array to be compared, and *diTableSize* is the number of values to be compared. For example, the size of the *diTablePosition* array is 256, if *diTableSize*=100 is set, the 0–99 elements of the array will be set as the values to be compared. The timing diagram is as follows:

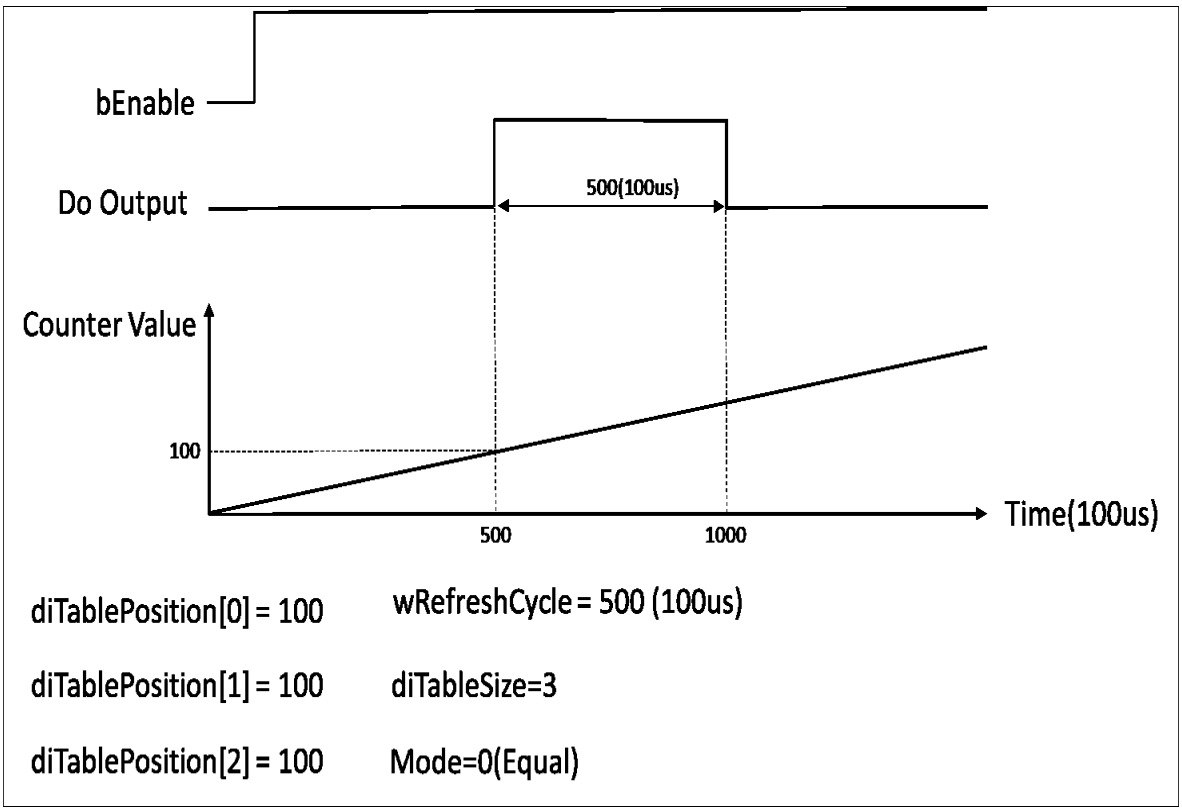


If comparing multiple value at the same time, the following two situations will affect the ouput of Do.

Case1: In the process of Do output, other value of *diTablezpodtion* is compared, the value will not generate Do output. The output will be the Do output of the current value. Refer the following diagram for the timing and parameter:



Case2: When comparing multiple values of *diTablePosion* at the same time, the defined Do output time is the time of of one cycle of *wRefreshCycle*. Refer the following diagram for the timing and parameter:



- *OutputAction* is an output variable that define the action of the output device once the comparison condition is met. When it is set to SET, the ouput device will be set to high level. When it is set to RESET, the output device will be set to low level.

- Only support AX-332E version 1.0.4.2 or later.

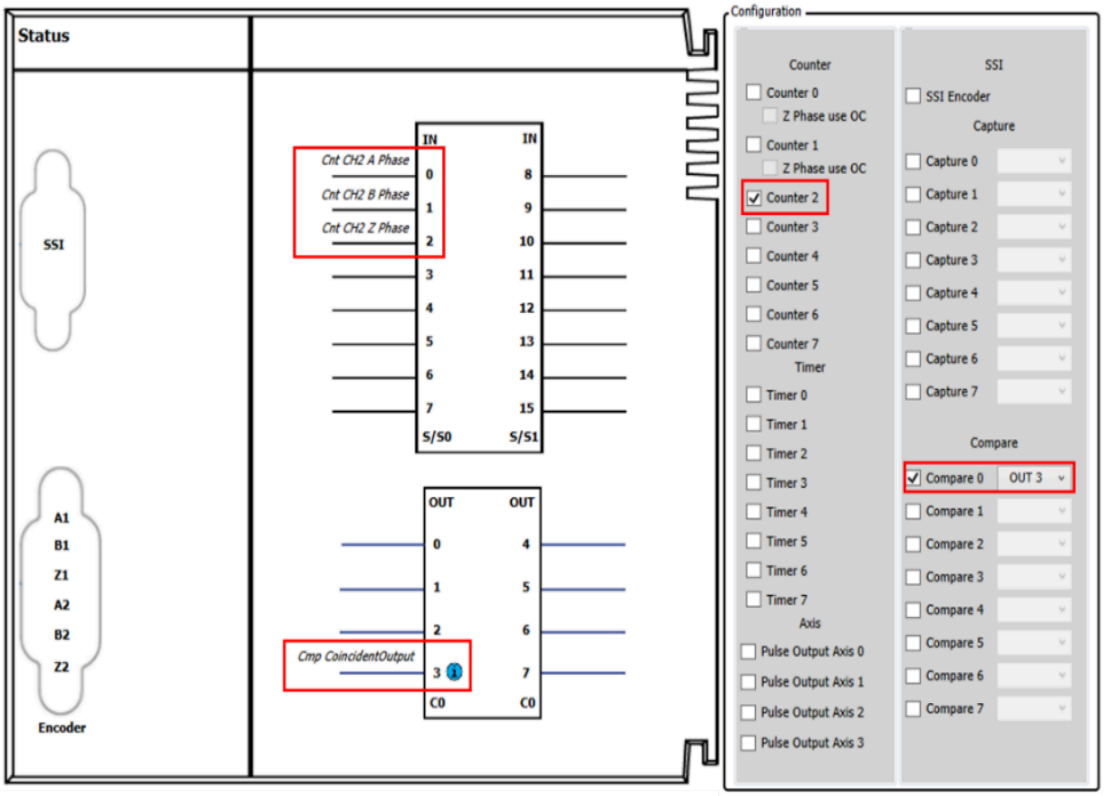
- **Troubleshooting**

  If an error occurs while running the instruction, *bError* will change to TRUE and Capture will stop. You can refer to ErrorID (Error Code) to troubleshoot the problem.

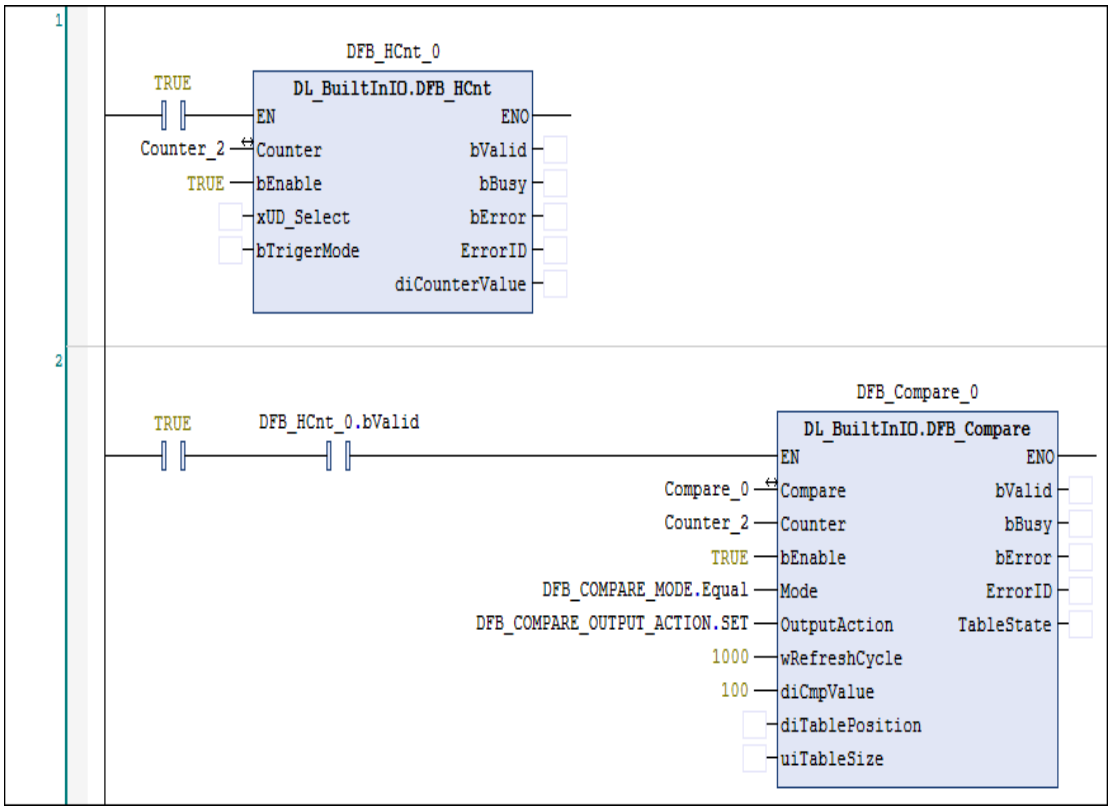
- **Programming Example**

  - This example uses DFB_HCnt and DFB_Compare in AX-308E to perform the Compare function.

    1. As the following figure shows, select a Counter and a Compare for **Hardware IO Configuration** in **BuiltIn_IO** and set a signal output on the hardware as the output device of Compare (e.g. **OUT 3**).



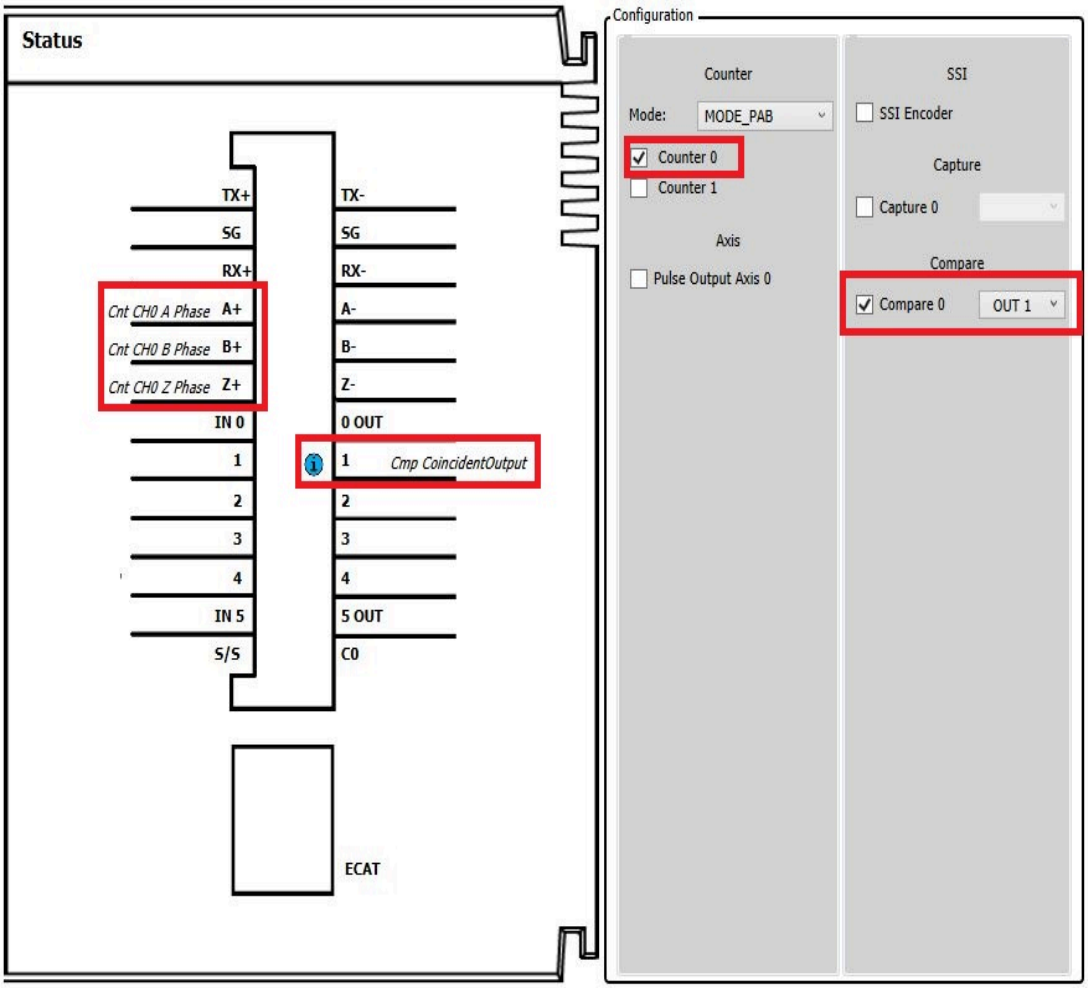
    2. Run the function block DFB_Compare after enable the high-speed counter by using DFB_HCnt in the POU as shown in follows. At the same time, the output device (OUT 3) will output the signal once the comparison condition is fulfilled (*DFB_HCnt_0.diCounterValue* = *DFB_Compare_0.diCmpValue*).



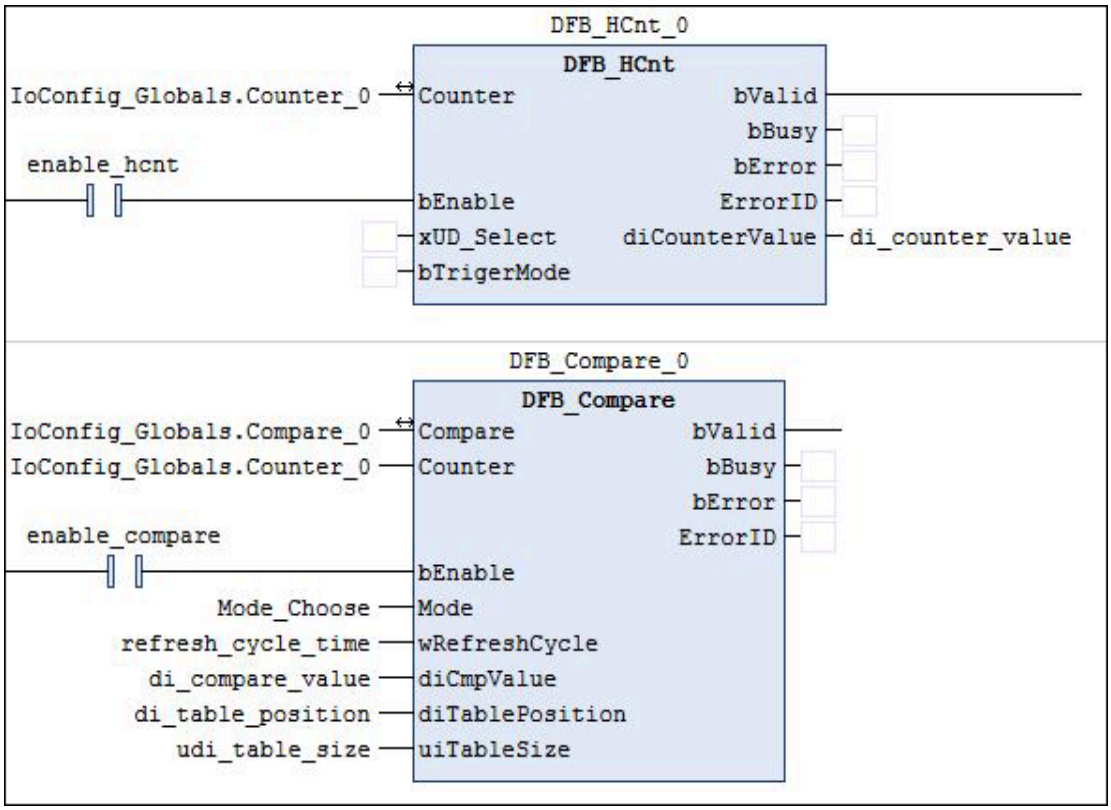
3. Refer to *AX-3 Series Operation Manual* for more details related to the settings and operation of **Hardware IO Configuration**.

- This example uses DFB_HCnt and DFB_Compare in AX-332E to perform the Compare function.

1. As the following figure shows, select a Counter and a Compare for **Hardware IO Configuration** in **BuiltIn_IO** and set a signal output on the hardware as the output device of Compare (e.g. **OUT 1**).



2. Run the function block DFB_Compare after enable the high-speed counter by using DFB_HCnt in the POU as shown in follows. At the same time, the output device (OUT 1) will output the signal once the comparison condition is fulfilled (*DFB_HCnt_0.diCounterValue* = *DFB_Compare_0.diCmpValue*).



3. Refer to *AX-3 Series Operation Manual* for more details related to the settings and operation of **Hardware IO Configuration**.

- **Supported Devices**
  - AX-3 series (except for AX-300), AX-C
- **Library**
  - DL_BuiltInIO_AX3.library

    Note: From version 1.0.5.0, library DL_BuiltInIO_AX3 is changed to DL_BuiltInIO.

3.3. DFB_HCnt

DFB_HCntL enables the high-speed counter and monitors the counter value.

FB/FC	Instruction	Graphic Expression	ST Language
FB	DFB_HCnt		DFB_HCnt_instance(Counter :=, bEnable :=, bTrigerMode :=, bValid =>, bBusy =>, bError =>, ErrorID =>, diCounterValue =>);

- **Inputs**

Name	Function	Data Type	Setting Value (Default value)	Timing for Updating
bEnable	Run the instruction when *bEnable* changes to TRUE.	BOOL	TRUE/FALSE (FALSE)	–
xUD_Select*2	Define the high-speed counter to count up or down	BOOL	TRUE/FALSE (FALSE)	When *bEnable* is rising edge, update parameter of *xUD_Select*.
bTriggerMode*2	Define the capture is triggered by rising edge or falling edge	BOOL	TRUE/FALSE (FALSE)	When *bEnable* is rising edge, update parameter of *bTrigerMode*.

***Note**:

1. DFB_HSIO_ERROR: Enumeration (Enum)
2. Only applicable to AX-332.

- **Outputs**

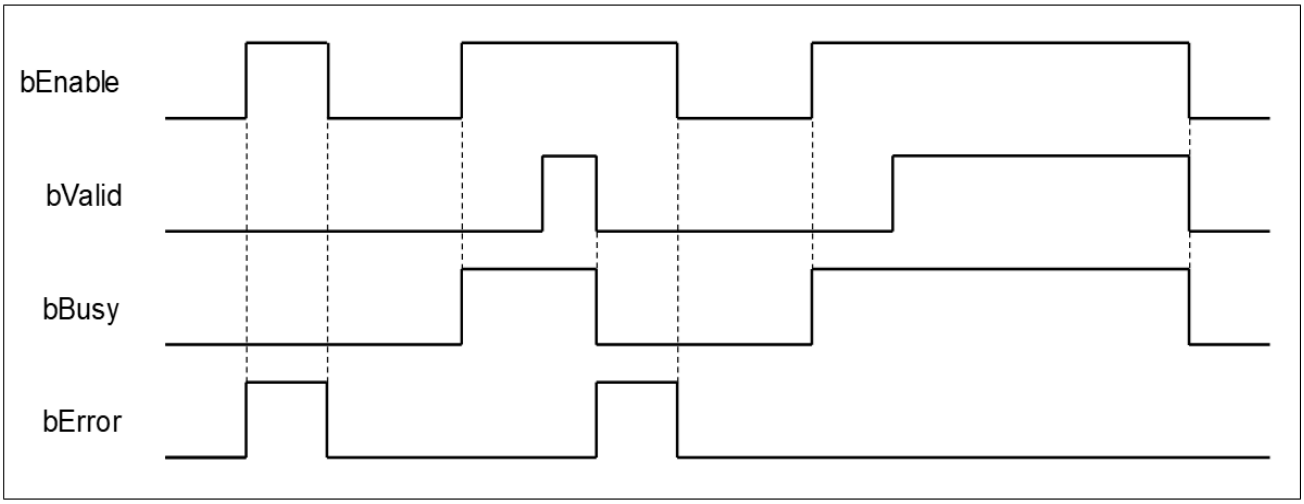
Name	Function	Data Type	Output Range (Default value)
bValid	TRUE when the output value is valid.	BOOL	TRUE/FALSE (FALSE)
bBusy	TRUE when the instruction is enabled.	BOOL	TRUE/FALSE (FALSE)
bError	TRUE if an error occurs.	BOOL	TRUE/FALSE (FALSE)
ErrorID	Indicates the error code if an error occurs.	DFB_HSIO_ERROR*	DFB_HSIO_ERROR (DFB_HSIO_NO_ERR)
diCounterValue	The present counter value of the counter	DINT	Positive number, negative number or 0 (0)

***Note**: DFB_HSIO_ERROR: Enumeration (Enum)

- **Output Updating Timing**

Name	Timing for shifting to TRUE	Timing for shifting to FALSE
bValid	When *bEnable*is TRUE and the output values are valid after one scan cycle	• When *bEnable* shifts to FALSE • When *bError* shifts to TRUE
bBusy	When *bEnable* is triggered by the rising edge	• When *bEnable* shifts to FALSE • When *bError* shifts to TRUE
bError	When an error occurs during the running or there are incorrect input values	When *bEnable* shifts from TRUE to FALSE (Error code is cleared)
ErrorID		
diCounterValue	Updates value continuously when *bValid* is TRUE.	Updates value continuously when *bValid* is TRUE.

- **Timing Diagram**



- **Inputs/Outputs**

Name	Function	Data Type	Setting Value (Default value)	Timing for Updating
Counter	Reference to the source of specified high-speed counter.	DFB_COUNTER_REF (FB)*	DFB_COUNTER_REF (Cannot be null.)	When *bEnable* shifts to TRUE and *Busy* is FALSE.

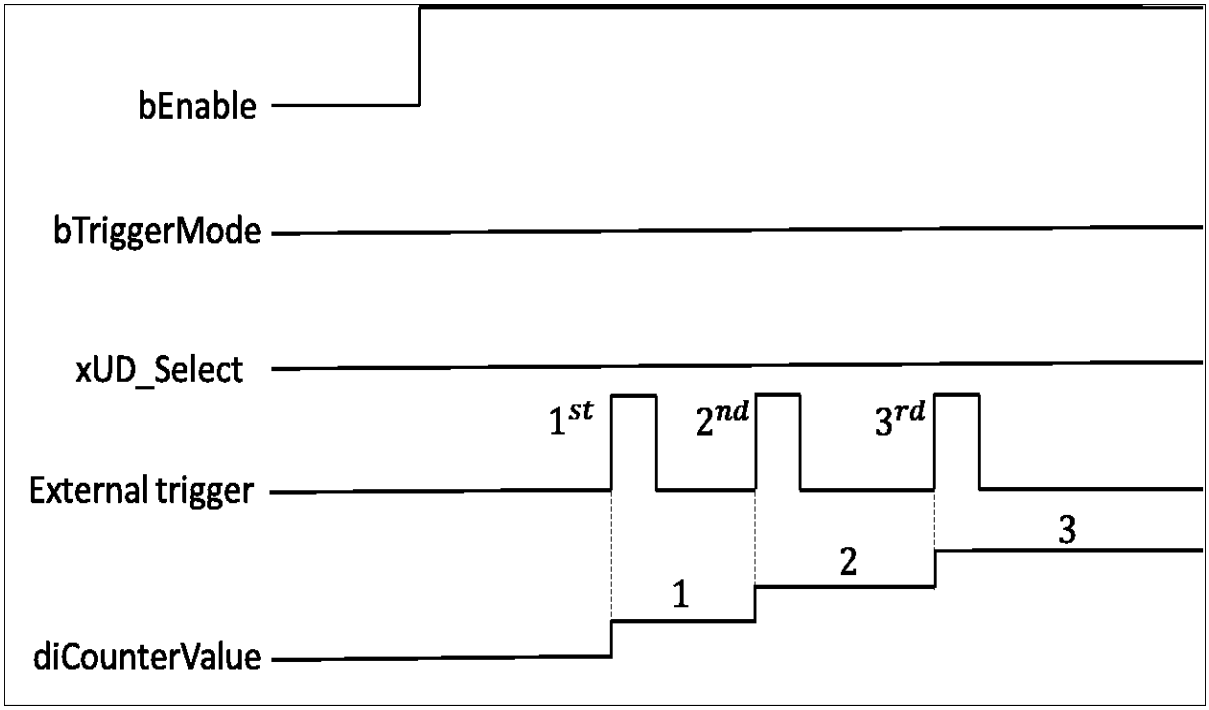
***Note**: DFB_Counter_REF (FB): This FB work as the driver interface of the high-speed counter . It calls the counter's parameters and the driver.

- **Function**

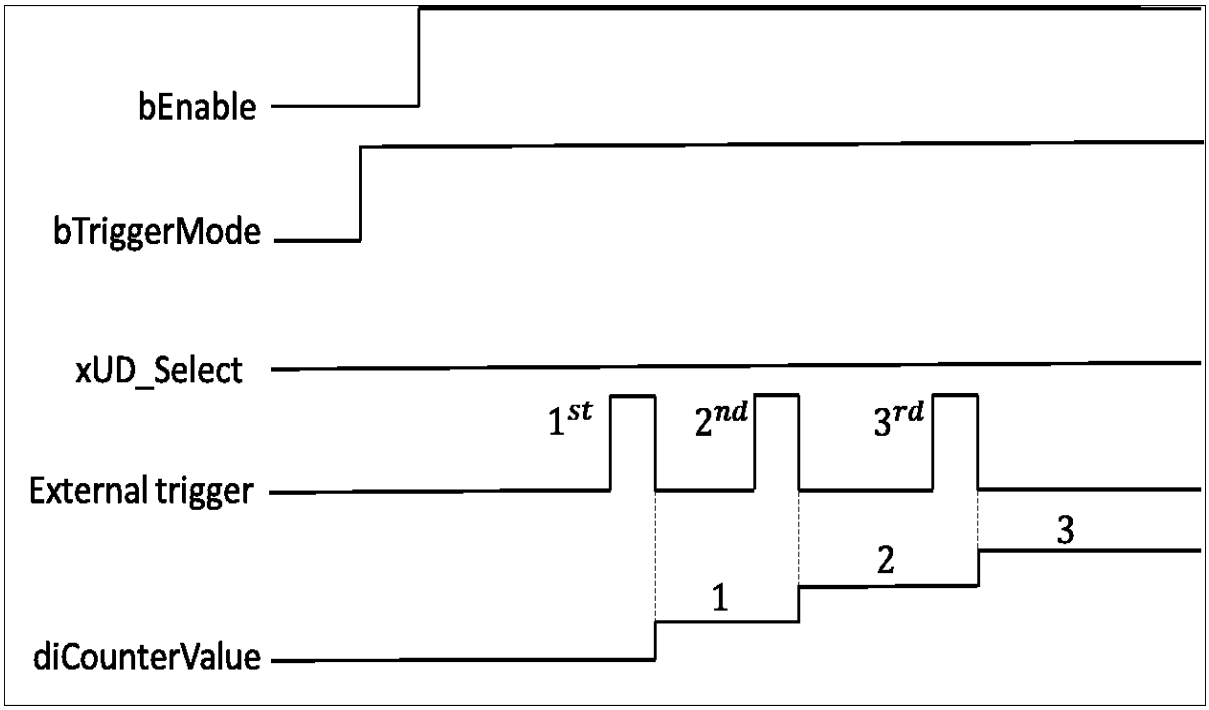
1. When the input *bEnable* is TRUE, the counter will start calculating pulses to the corresponding input points based on the Counter configuration of Hardware IO Configuration in BuiltIn IO.
2. The counter value is given through the output *diCounterValue* during the counting process.
3. *bTriggerMode* defines whether the DI signal is triggered by the rising or falling edge for high-speed counting. It is only applicable to AX-332E. When EdgeSelect=FALSE, high-speed counting is triggered when DI generates the rising edge signal; when EdgeSelect=TRUE, high-speed counting

is triggered when DI generates the falling edge signal. This function is only available when the corresponding input Counter mode is MODE_UD/ MODE_UDR/ MODE_UDRE/MODE_UDRE2.

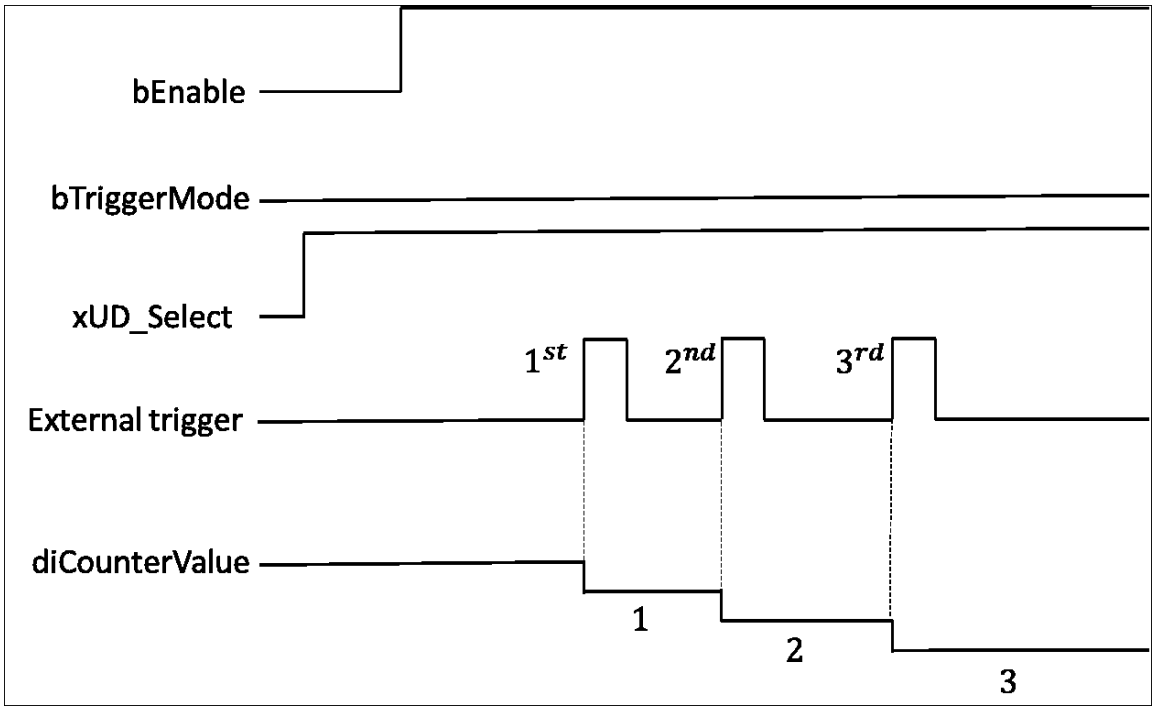
4. xUD_Select defines whether to count up or count down when the external DI signal triggers counting. It is only applicable to AX-332E. When xUD_Select=FALSE, the DI signal is triggered to count up, and when xUD_Select=TRUE, the DI signal is triggered to count down. This function is only available when the corresponding input Counter mode is MODE_UD/ MODE_UDR/ MODE_UDRE.
   Refer to the following timing diagram for *bTriggerMode* and *xUD_Select*:
   bTriggerMode=FALSE
   xUD_Select=FALSE



bTriggerMode=TRUE
xUD_Select=FALSE

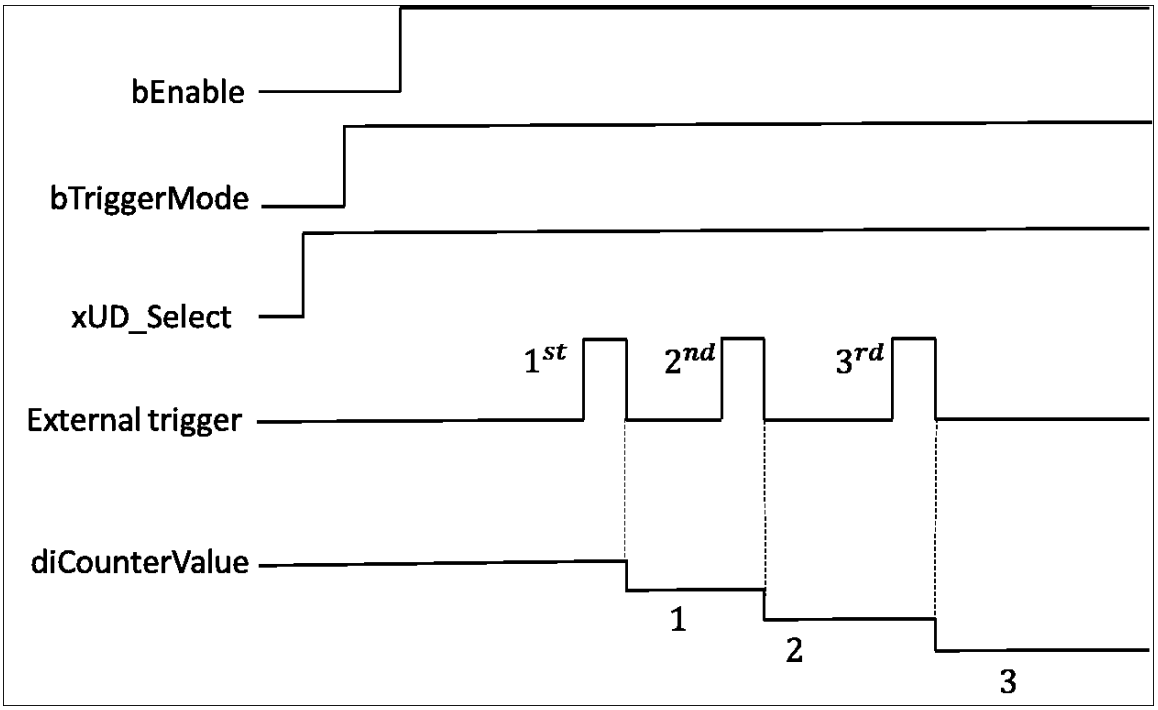


bTriggerMode=FALSE
xUD_Select=TRUE



bTriggerMode=TRUE

xUD_Select=TRUE



5. Only supports AX-332 v1.0.4.2 or later.

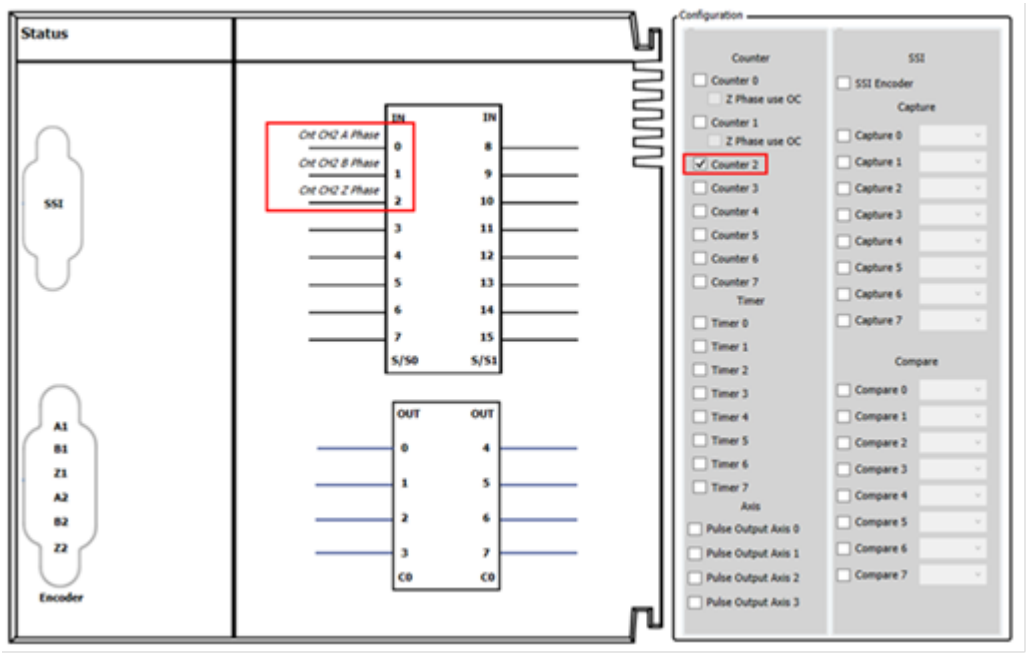
- **Troubleshooting**

  If an error occurs while running the instruction, *bError* will change to TRUE and Capture will stop. You can refer to ErrorID (Error Code) to troubleshoot the problem.

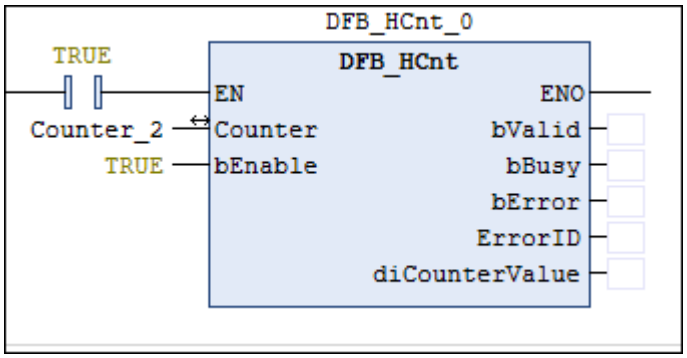
- **Programming Example**

  - This example uses DFB_HCnt in AX-308 to perform the Count function.

    1. As the following figure shows, select a Counter (**Counter 2**) in **Hardware IO Configuration** and you will see the input points (e.g. IN 0, IN 1, IN 2) matched to the corresponding encoder A, B, Z phase outputs, which the wiring should follow the configuration to perform the normal function of high-speed counting.



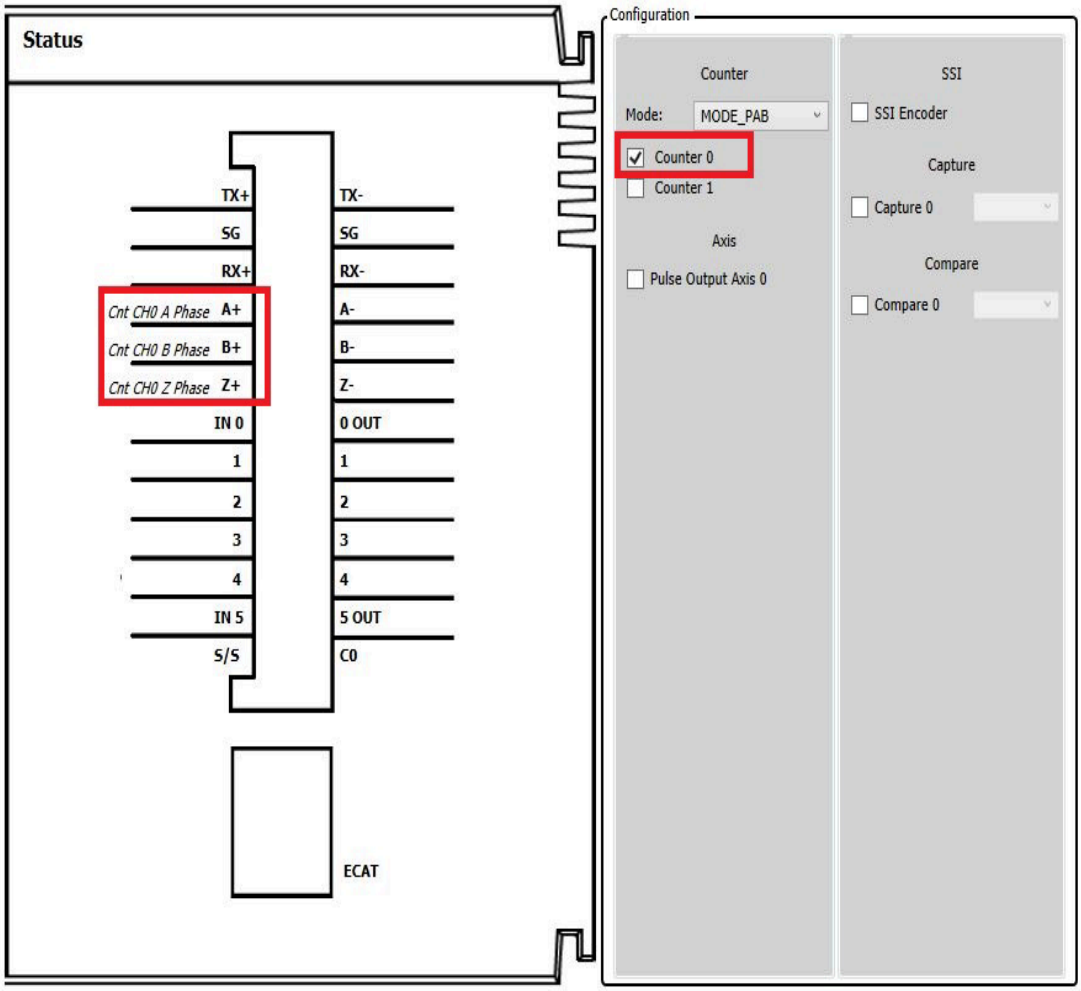
2. After using the FB DFB_HCnt in the POU to activate the high-speed counter (*bEnable* = TRUE), it starts receiving and counting the pulses from the external signals (IN 0, IN 1) based on the counting mode set in **Counter Configuration**, then the counter value will be displayed in the output *diCounterValue*. In addition, you should make sure that the mode of sending pulses from the external signal source matches the counting mode to get the correct counter values.

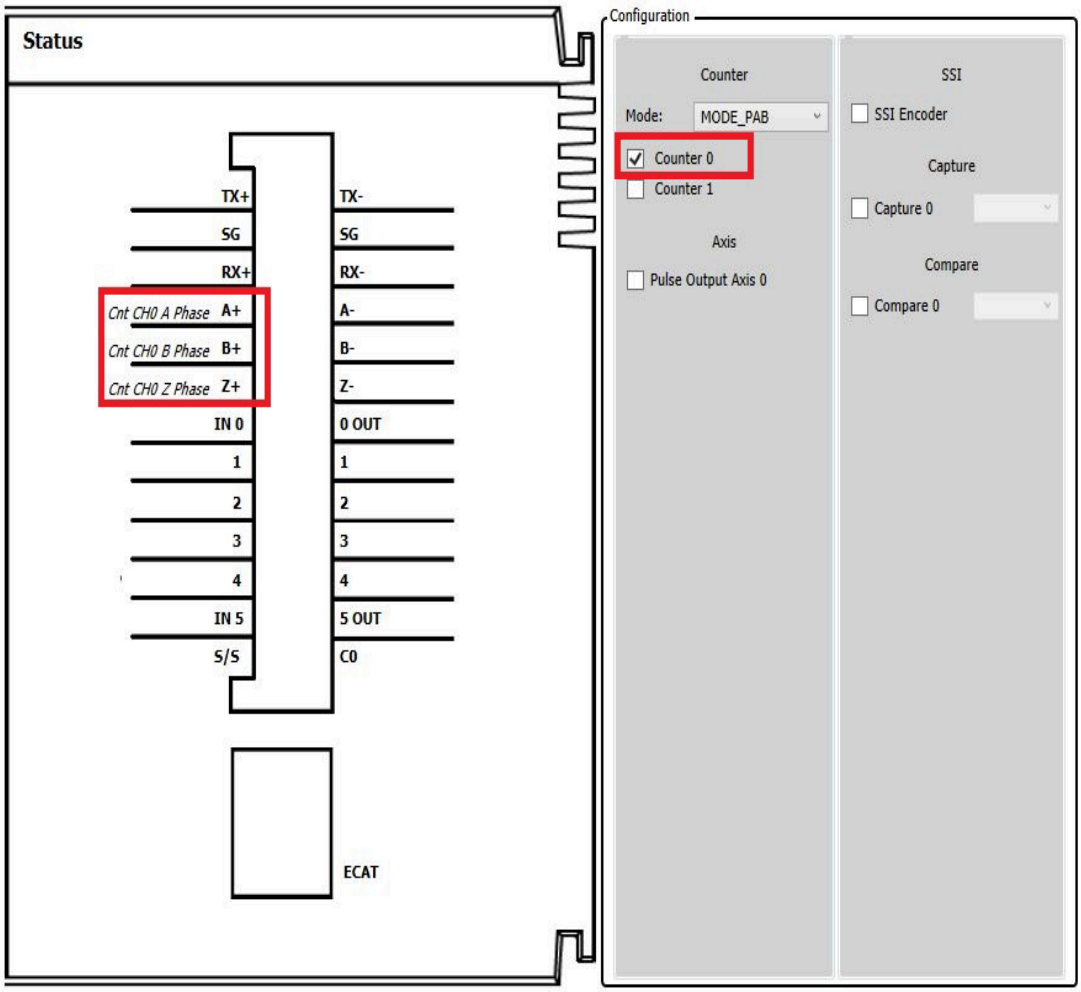


3. Refer to *AX-3 series Operation Manual* for more details related to the settings and operation of **Hardware IO Configuration** and **Counter Configuration**.

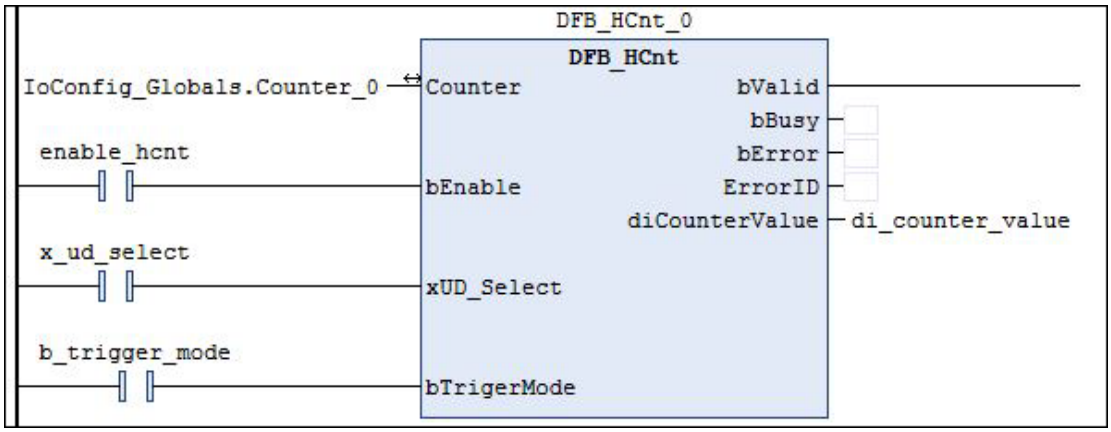
- This example uses DFB_HCnt in AX-332E to perform the Count function.

1. As the following figure shows, select a Counter (**Counter 0**) in **Hardware IO Configuration** and you will see the input points (e.g. A+A–/B+B–/C+C–)) matched to the corresponding encoder A, B, Z phase outputs, which the wiring should follow the configuration in order to run the high-speed counting properly.


Status
TX+
SG
RX+
Cnt CH0 A Phase A+
Cnt CH0 B Phase B+
Cnt CH0 Z Phase Z+
IN 0
1
2
3
4
IN 5
S/S
TX-
SG
RX-
A-
B-
Z-
0 OUT
1
2
3
4
5 OUT
C0
ECAT
Configuration
Counter
Mode:
MODE_PAB
Counter 0
Counter 1
Axis
Pulse Output Axis 0
SSI
SSI Encoder
Capture
Capture 0
Compare
Compare 0


Status
TX+
SG
RX+
Cnt CH0 A Phase A+
Cnt CH0 B Phase B+
Cnt CH0 Z Phase Z+
IN 0
1
2
3
4
IN 5
S/S
TX-
SG
RX-
A-
B-
Z-
0 OUT
1
2
3
4
5 OUT
C0
ECAT
Configuration
Counter
Mode:
MODE_PAB
Counter 0
Counter 1
Axis
Pulse Output Axis 0
SSI
SSI Encoder
Capture
Capture 0
Compare
Compare 0

2. After using the FB DFB_HCnt in the POU to activate the high-speed counter (*bEnable* = TRUE), it starts receiving and counting the pulses from the external signals(A+A–/B+B–) based on the counting mode set in **Counter Configuration**, then the counter value will be displayed in the output *diCounterValue*. In addition, you should make sure that the mode of sending pulses from the external signal source matches the counting mode so as to get the correct counter values.



3. Refer to *AX-3 Series Operation Manual* for more details related to the settings and operation of **Hardware IO Configuration** and **Counter Configuration**.

- **Supported Devices**
  - AX-3 series (except for AX-300), AX-C
- **Library**
  - DL_BuiltInIO_AX3.library

3.4. DFB_HTmr

DFB_HTmr enables the specified high-speed timer channel according to the specified parameters and monitors and timed value.

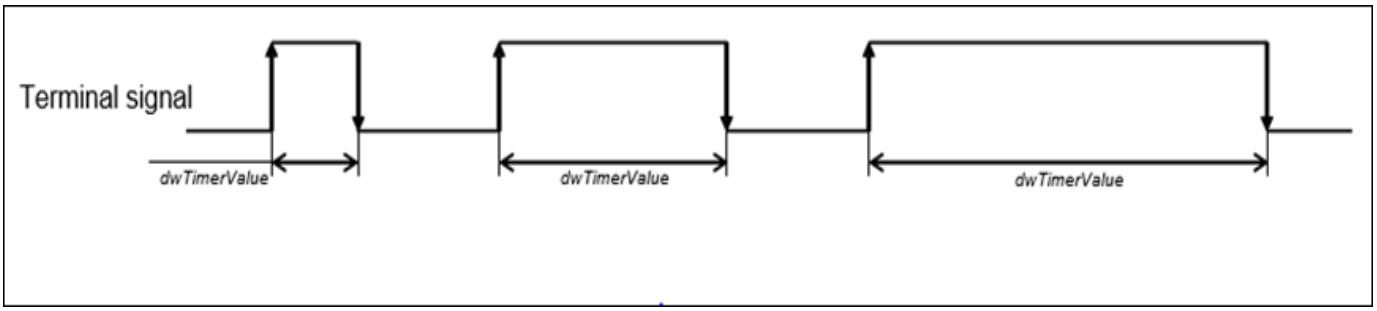
FB/FC	Instruction	Graphic Expression	ST Language
FB	DFB_HTmr		DFB_HTmr_instance(Timer :=, bEnable :=, TriggerMode :=, bValid =>, bBusy =>, bError =>, ErrorID =>, dwTimerValue =>);

- **Inputs**

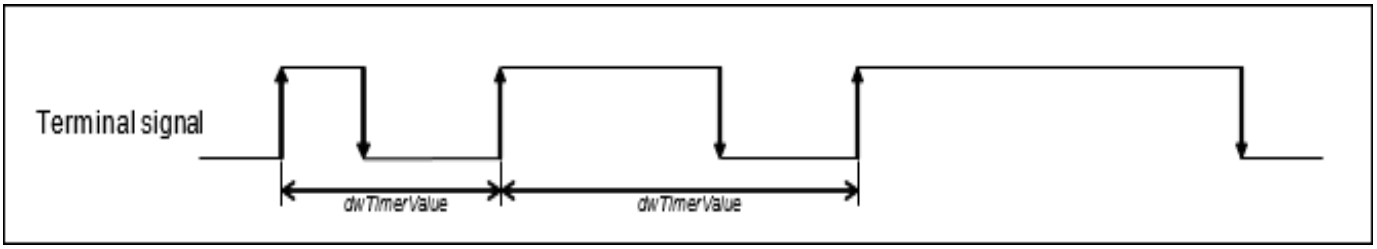
Name	Function	Data Type	Setting Value (Default value)	Timing for Updating
bEnable	Run the instruction when *bEnable* changes to TRUE.	BOOL	TRUE/FALSE (FALSE)	–
TriggerMode	Set the timing mode.	DFB_TIMER_MODE *	0: UP_DOWN 1: UP_UP (UP_DOWN)	When *bEnable* shifts to TRUE and *bBusy* is FALSE.

***Note**: DFB_TIMER_MODE: Enumeration (Enum)

Up–Down mode:



Up–Up mode:



- **Outputs**

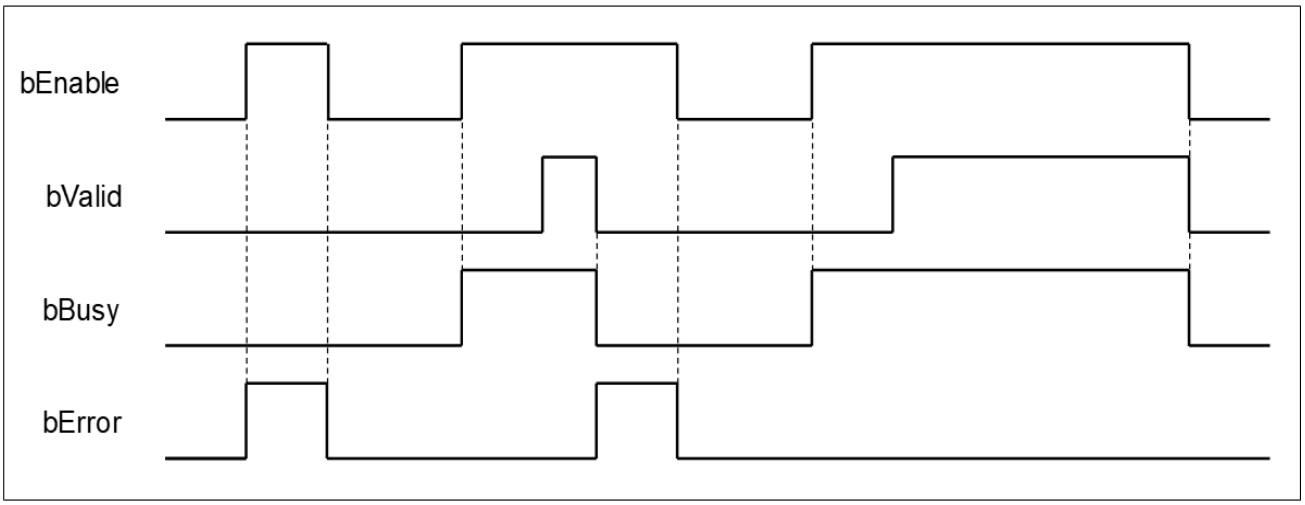
Name	Function	Data Type	Output Range (Default value)
bValid	TRUE when the output value is valid.	BOOL	TRUE/FALSE (FALSE)
bBusy	TRUE when the instruction is enabled.	BOOL	TRUE/FALSE (FALSE)
bError	TRUE if an error occurs.	BOOL	TRUE/FALSE (FALSE)
ErrorID	Indicates the error code if an error occurs.	DFB_HSIO_ERROR*	DFB_HSIO_ERROR (DFB_HSIO_NO_ERR)
dwTimerValue	Timed value (Unit: 0.01us)	DWORD	Positive number or 0(0)

***Note**: DFB_HSIO_ERROR: Enumeration (Enum)

- **Output Updating Timing**

Name	Timing for shifting to TRUE	Timing for shifting to FALSE
bValid	When *bEnable*is TRUE and the output values are valid after one scan cycle	• When *bEnable* shifts to FALSE • When *bError* shifts to TRUE
bBusy	When *bEnable* is triggered by the rising edge	• When *bEnable* shifts to FALSE • When *bError* shifts to TRUE
bError	When an error occurs during the running or there are incorrect input values	When *bEnable* shifts from TRUE to FALSE (Error code is cleared)
ErrorID		
dwTimerValue	Updates value continuously when *bValid* is TRUE.	Updates value continuously when *bValid* is TRUE.

- **Timing Diagram**



- **Inputs/Outputs**

Name	Function	Data Type	Setting Value (Default value)	Timing for Updating
Timer	Reference to the source of the specified high-speed timer.	DFB_TIMER_REF (FB)*	DFB_TIMER_REF (Cannot be null.)	When *bEnable* shifts to TRUE and *Busy* is FALSE

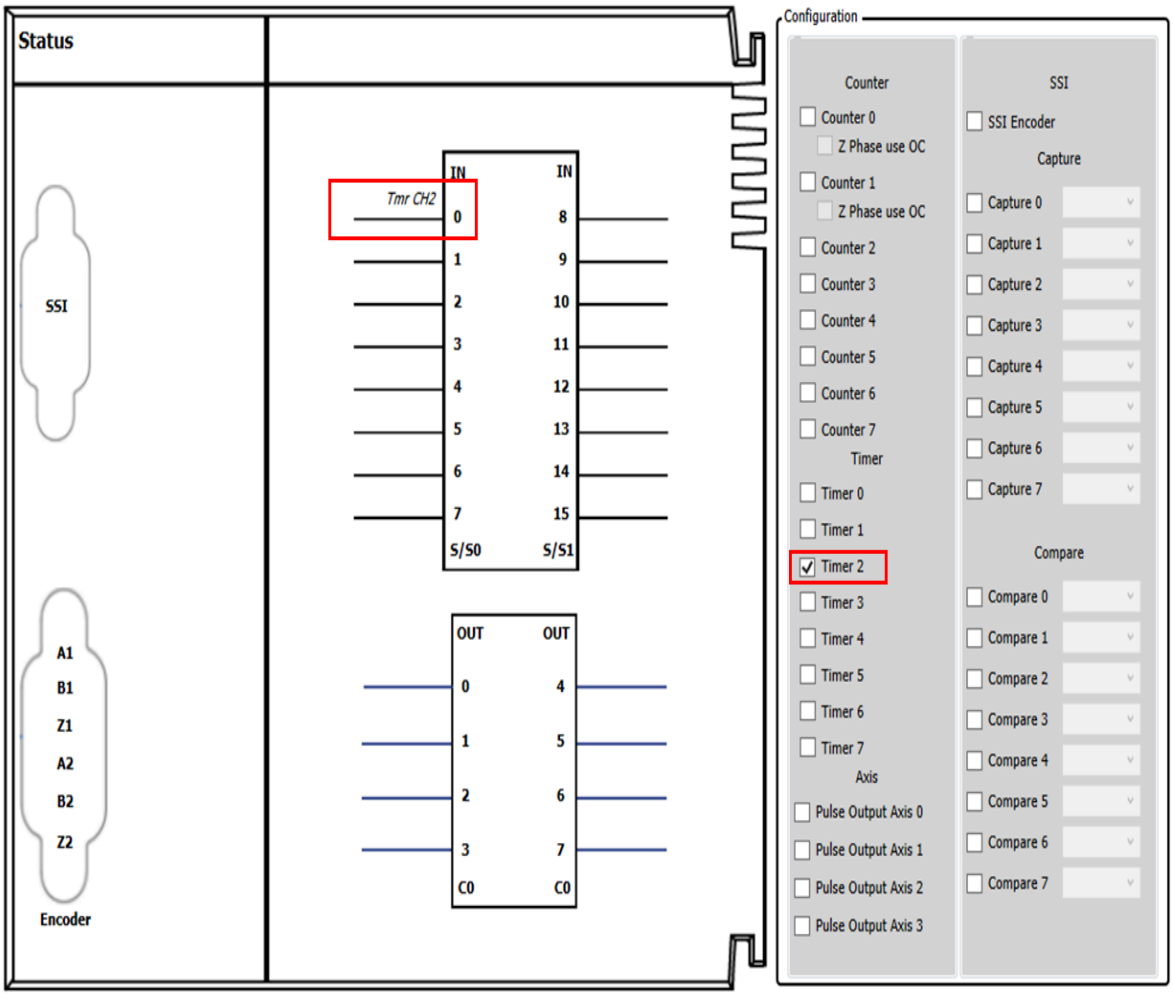
***Note**: DFB_TIMER_REF (FB): This FB work as the driver interface of the high-speed timer . It calls the timer's parameters and the driver.

- **Function**
  - When the input *bEnable* is TRUE, the timer will start calculating pulses to the corresponding input points based on the Timer configuration of HW IO configuration in BuiltIn IO.
  - The counter value is given through the output *dwTimerValue* during the counting process.
- **Troubleshooting**

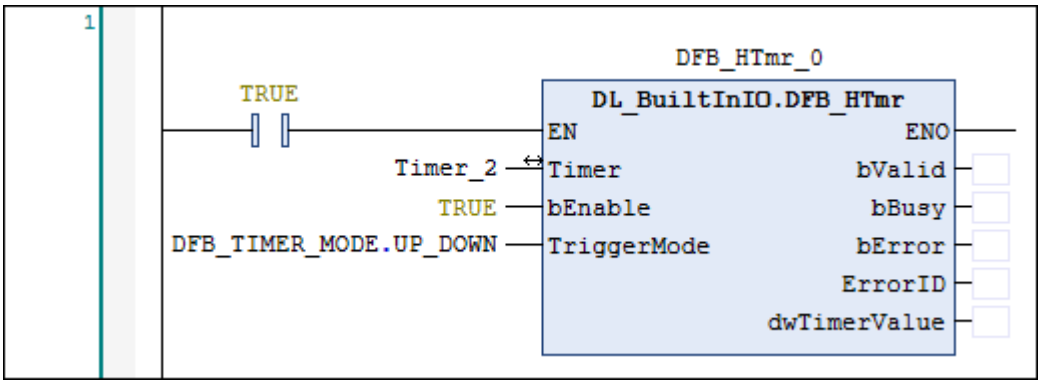
  If an error occurs during the running of the instruction, *bError* will change to TRUE and Capture will stop. You can refer to *ErrorID* (Error Code) to address the problem.
- **Programming Example**

  This example demonstrates the function performed by DFB_HTmr.

  1. As the following figure shows, select a Timer (**Timer 2**) in **Hardware IO Configuration** and you will see the input point (IN 0) matched to the corresponding timer input channel, which the wiring should follow the configuration so as to perform the normal function of high-speed timing.



2. After using the FB DFB_HTmr in the POU to activate the high-speed timer (*bEnable* = TRUE), it starts receiving and counting the pulses from the external signals (IN 0) based on the timing mode set in Timer Configuration, then the timed value will be displayed in the output *dwTimerValue*.



3. Refer to *AX-3 Series Operation Manual* for more details about **Hardware IO Configuration**.

- **Supported Devices**
  - AX-3 series (except for AX-300)
- **Library**
  - DL_BuiltInIO.library

    Note: From version 1.0.5.0, library DL_BuiltInIO_AX3 is changed to DL_BuiltInIO.

3.5. DFB_PresetValue

DFB_PresetValue presets the current counter value to the default value. It is application function block for high-speed counters.

FB/FC	Instruction	Graphic Expression	ST Language
FB	DFB_PresetValue		DFB_PresetValue_instance(Counter :=, bExecute :=, TriggerType :=, diPresetValue :=, bDone =>, bBusy =>, bCommandAborted =>, bError =>, ErrorID =>);

- **Inputs**

Name	Function	Data Type	Setting Value (Default value)	Timing for Updating
bExecute	Run the instruction when *bEnable* changes to TRUE.	BOOL	TRUE/FALSE (FALSE)	–
TriggerType	Define when the default value will be preset.	DFB_PRESET_TRIGGER_TYPE *	0:RUN_TRIGGER 1:EXTERNAL_TRIGGER (RUN_TRIGGER)	When *bExecute* shifts to TRUE and *bBusy* is FALSE.
diPresetValue	The preset counter value for high-speed counters.	DINT	Positive number, negative number or 0 (0)	When *bExecute* shifts to TRUE and *bBusy* is FALSE

***Note**: DFB_PRESET_TRIGGER_TYPE: Enumeration (Enum)

1. RUN_TRIGGER: Set the default value right after the input *bExecute* shifts to TRUE.
2. EXTERNAL_TRIGGER: Set the default value right after the external signal of high-speed counter being triggered.

- **Outputs**

Name	Function	Data Type	Output Range (Default value)
bDone	The default value of the counter has been changed.	BOOL	TRUE/FALSE (FALSE)
bBusy	TRUE when the instruction is enabled	BOOL	TRUE/FALSE (FALSE)

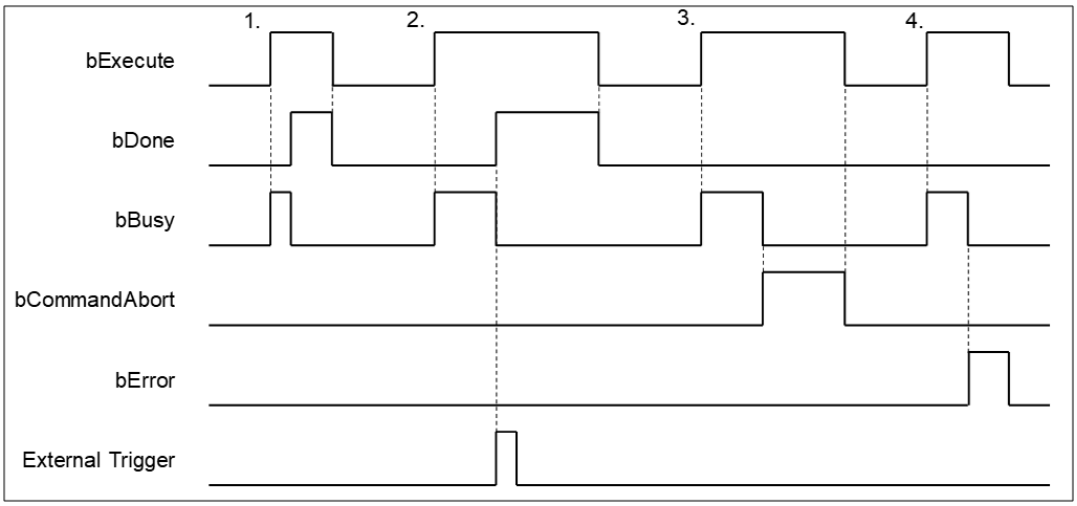
Name	Function	Data Type	Output Range (Default value)
bCommandAborted	TRUE when the instruction is interrupted before it's completed	BOOL	TRUE/FALSE (FALSE)
bError	TRUE when an error occurs	BOOL	TRUE/FALSE (FALSE)
ErrorID	Error codes	DFB_HSIO_ERROR*	DFB_HSIO_ERROR (DFB_HSIO_NO_ERR)

***Note**: DFB_HSIO_ERROR: Enumeration (Enum)

- **Output Updating Timing**

Name	Timing for shifting to TRUE	Timing for shifting to FALSE
bDone	TRUE when the counter value has been set back to default.	• When *bExecute* shifts to FALSE • When *bError* shifts to TRUE
bBusy	When *bExecute* shifts to TRUE	• When *bExecute* shifts to FALSE • When *bError* shifts to TRUE
bCommandAborted	TRUE when the FB is interrupted	When *bExecute* shifts to FALSE
bError	When an error occurs during the running or there are incorrect input values	When *bEnable* shifts from TRUE to FALSE. (Error code is cleared)
ErrorID		

- **Timing Diagram**



1. *TriggerType* = 0 (RUN_TRIGGER)
2. *TriggerType* = 1 (EXTERNAL_TRIGGER)
3. *bCommandAborted* = TRUE
4. *bError* = TRUE

- **Inputs/Outputs**

Name	Function	Data Type	Setting Value (Default value)	Timing for Updating
Counter	Reference to the source of high-speed counter.	DFB_COUNTER_REF (FB)*	DFB_COUNTER_REF (Cannot be null.)	When *bExecute* shifts to TRUE and *bBusy* is FALSE

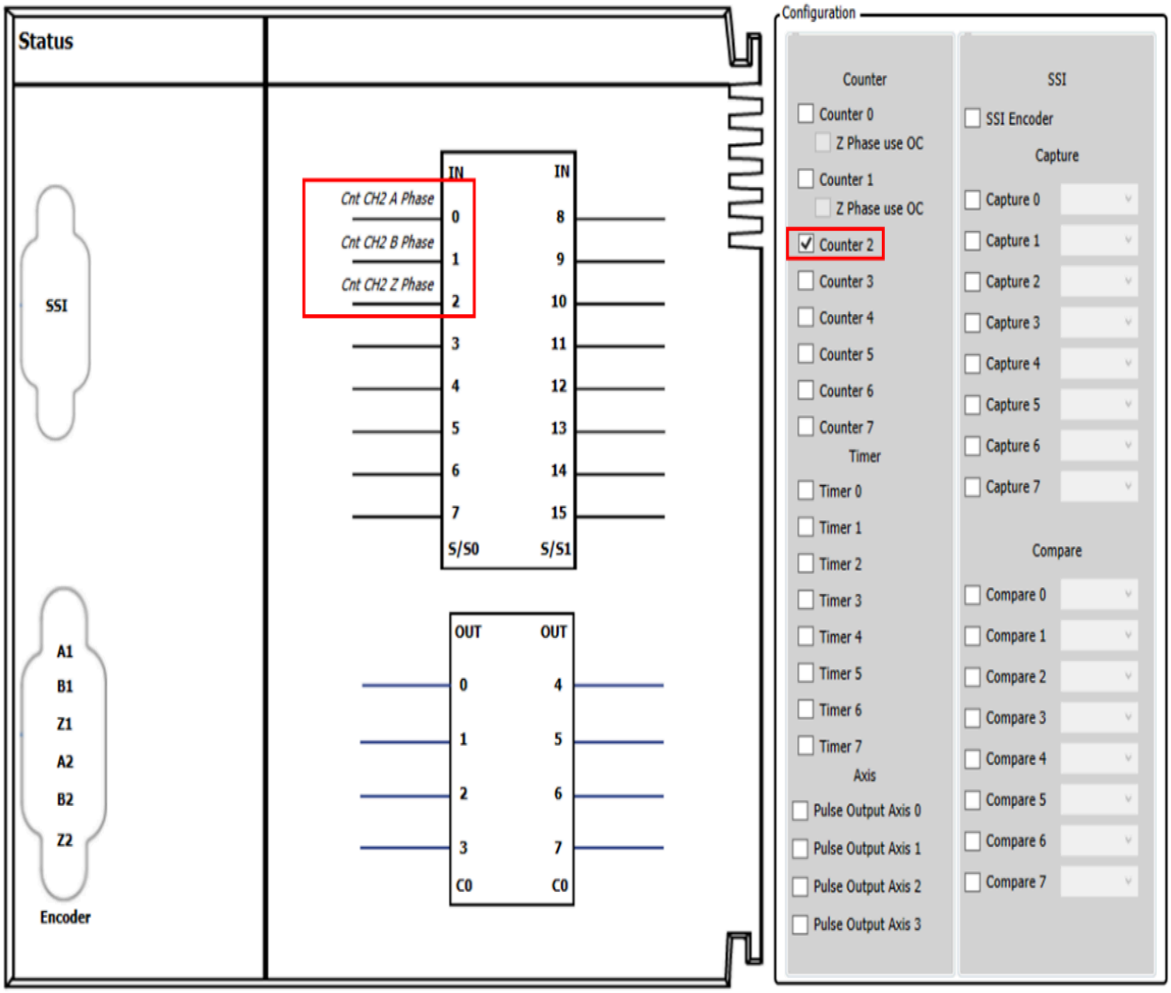
***Note**: DFB_COUNTER_REF (FB): This FB work as the driver interface of the high-speed counter. It calls the counter's parameters and the driver.

- **Function**
  - When *TriggerType* = RUN_TRIGGER, the counter value will be set back to the default value right after activating the function block.
  - When *TriggerType* = EXTERNAL_TRIGGER, the counter value will not be set back to the default until the Z phase signal of the counter rises.
- **Troubleshooting**

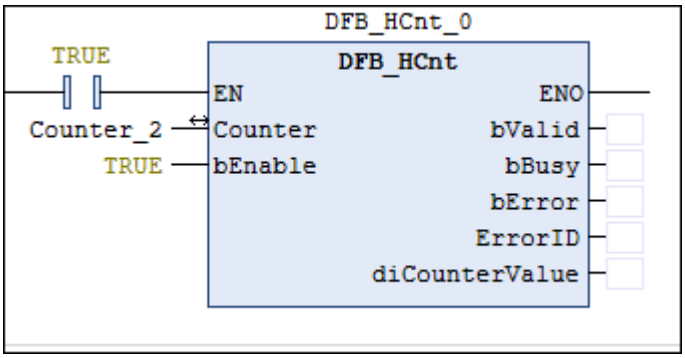
  If an error occurs during the running of the instruction, *bError* will change to TRUE and Capture will stop. You can refer to ErrorID (Error Code) to address the problem.
- **Programming Example**

  This example demonstrates the function performed by DFB_HCnt and DFB_PresetValue.

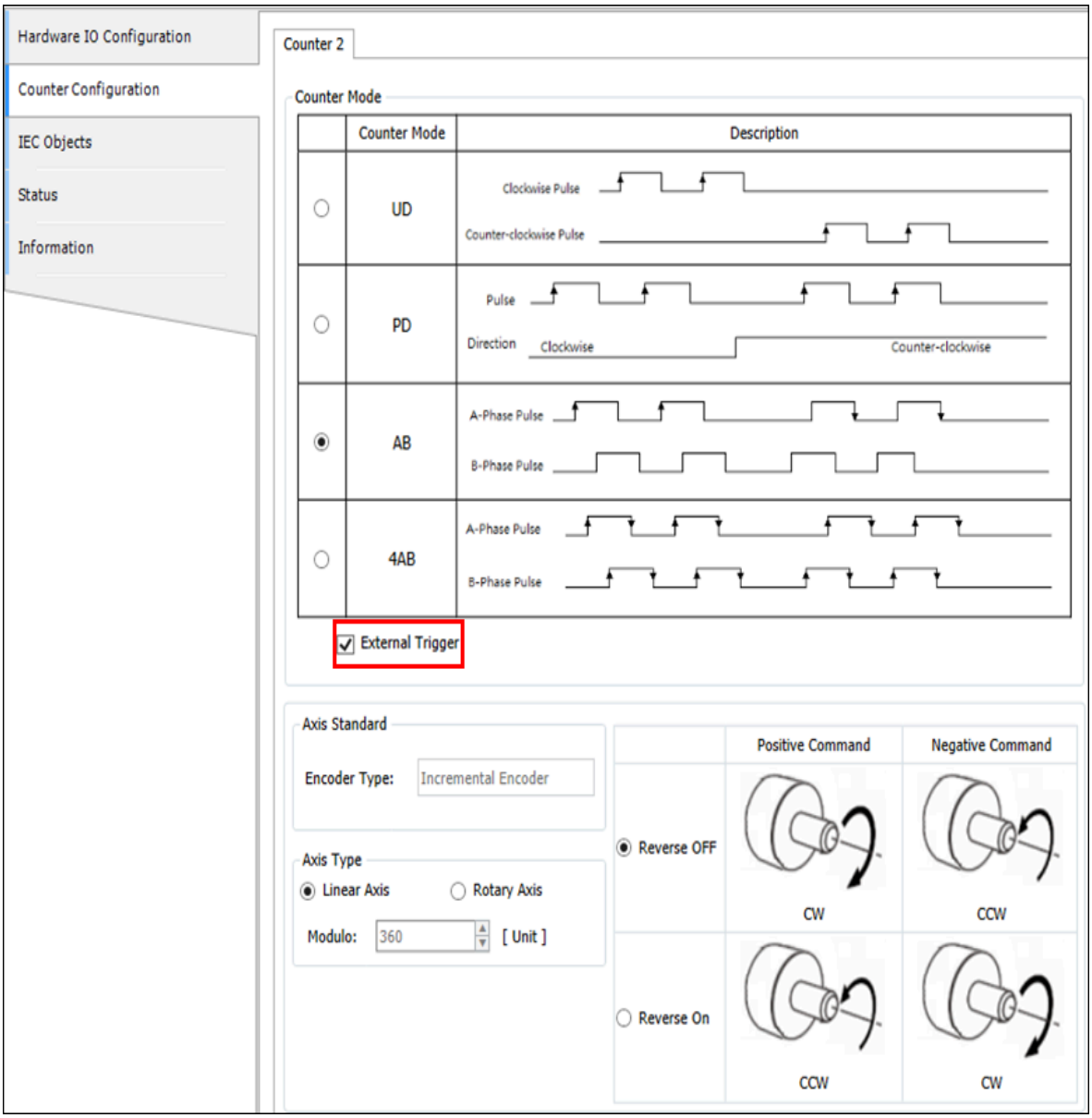
  1. As the following figure shows, select a Counter (**Counter 2**) in **Hardware IO Configuration** and you will see the input points (e.g. IN 0, IN 1, IN 2) matched to the corresponding encoder A, B, Z phase outputs, which the wiring should follows the configuration so as to perform the normal function of high-speed counting.



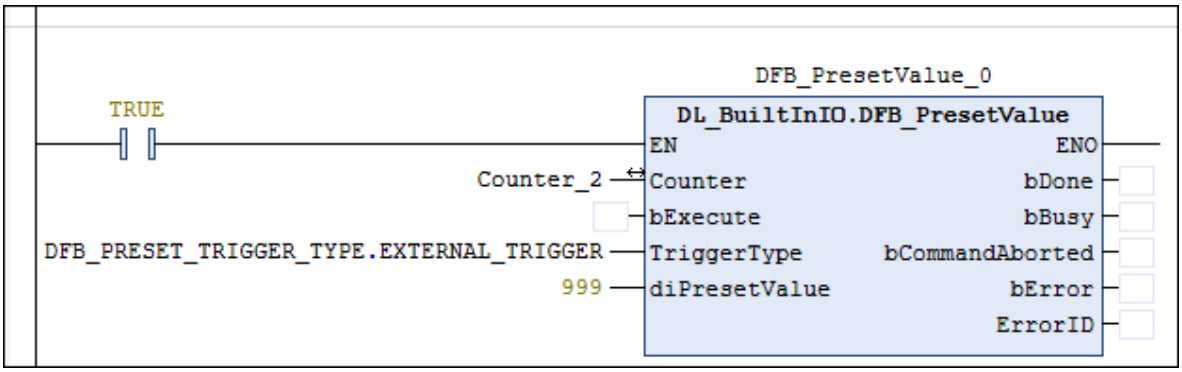
2. After using the FB DFB_HCnt in the POU to activate the high-speed counter(*bEnable* = TRUE), it starts receiving and counting the pulses from the external signals(IN 0, IN 1) based on the counting mode set in **Counter Configuration**, then the counter value will be displayed in the output *diCounterValue*. In addition, you should make sure that the mode of sending pulses from the external signal source matches the counting mode so as to get the correct counter values.



3. If you want to use external signal as the trigger, select **External trigger** in **Counter Configuration** as the following figure shows.



4. Then the input *bExecute* of DFB_PresetValue shifts to TRUE and the FB DFB_PresetValue will wait for the Z phase of high-speed counter to trigger the Default value function. After the counter value being set to the default (*DFB_HCnt.diCounterValue* = fb), the output *bDone* will shift from FALSE to TRUE.



5. Refer to *AX-3 Series Operation Manual* for more details of **Counter Configuration**.

- **Supported Devices**
  - AX-3 series (except for AX-300), AX-C
- **Library**

- DL_BuiltInIO.library

  Note: From version 1.0.5.0, library DL_BuiltInIO_AX3 is changed to DL_BuiltInIO.

3.6. DFB_Sample

DFB_Sample reads the increasing and decreasing number of the counter value during the sampling period. It is the application function block for high-speed counters.

FB/FC	Instruction	Graphic Expression	ST Language
FB	DFB_Sample		DFB_Sample_instance(Counter :=, bEnable :=, wSampleTime :=, bValid =>, bBusy =>, bError =>, ErrorID => diSampleValue =>);

- **Inputs**

Name	Function	Data Type	Setting Value (Default value)	Timing for Updating
bEnable	Run the instruction when *bEnable* changes to TRUE.	BOOL	TRUE/FALSE (FALSE)	–
wSampleTime	Sampling period (Unit: 1ms)	WORD	10–65535 (0)	When *bEnable* shifts to TRUE and *bBusy* is FALSE

- **Outputs**

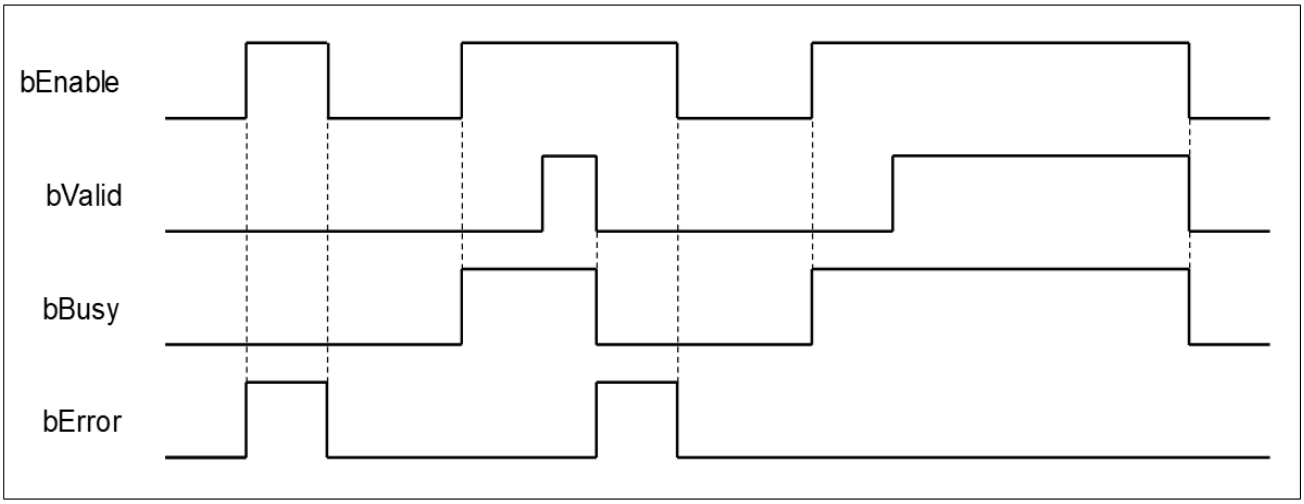
Name	Function	Data Type	Output Range (Default value)
bValid	TRUE when the output value is valid	BOOL	TRUE/FALSE (FALSE)
bBusy	TRUE when the instruction is enabled	BOOL	TRUE/FALSE (FALSE)
bError	TRUE if an error occurs	BOOL	TRUE/FALSE (FALSE)
ErrorID	Indicates the error code if an error occurs.	DFB_HSIO_ERROR*	DFB_HSIO_ERROR (DFB_HSIO_NO_ERR)
diSampleValue	Increasing number of the counter value during each sampling period	DINT	Positive number, negative number or 0 (0)

***Note**: DFB_HSIO_ERROR: Enumeration (Enum)

- **Output Updating Timing**

Name	Timing for shifting to TRUE	Timing for shifting to FALSE
bValid	When *bEnable*is TRUE and the output values are valid after one scan cycle	• When *bEnable* shifts to FALSE • When *bError* shifts to TRUE
bBusy	When *bEnable* shifts to TRUE	• When *bEnable* shifts to FALSE • When *bError* shifts to TRUE
bError	When an error occurs during the running or there are incorrect input values	When *bEnable* shifts from TRUE to FALSE (Error code is cleared)
ErrorID		
diSampleValue	Updates value continuously when *bValid* is TRUE.	Updates value continuously when *bValid* is TRUE.

- **Timing Diagram**



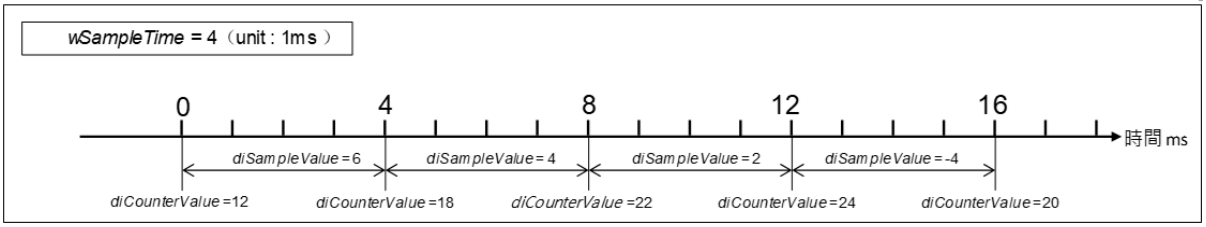
- **Inputs/Outputs**

Name	Function	Data Type	Setting Value (Default value)	Timing for Updating
Counter	Reference to the source of high-speed counter.	DFB_COUNTER_REF (FB)*	DFB_COUNTER_REF (Cannot be null.)	When *bEnable* shifts to TRUE and *bBusy* is FALSE.

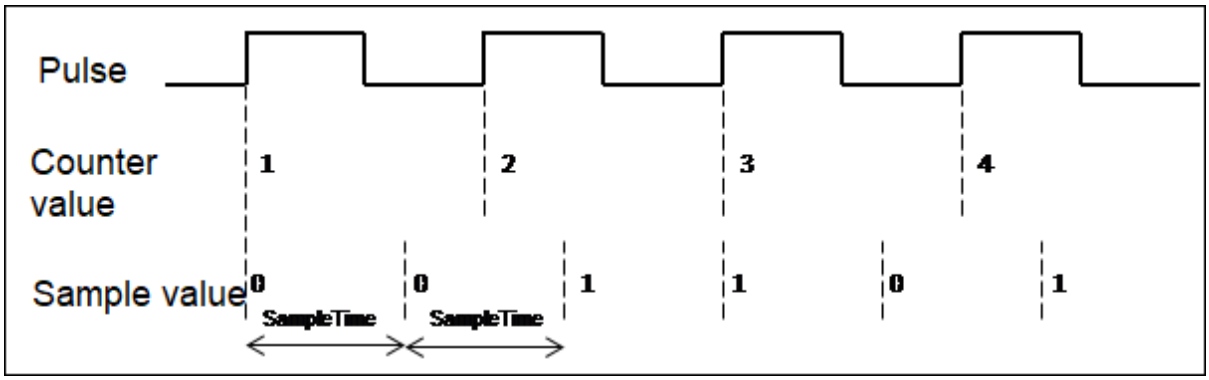
***Note**: DFB_COUNTER_REF (FB): As the I/O interface of the high-speed counter to perform actions include parameter adjustment and the driver.

- **Function**

  - DFB_Sample counts incoming pulses during a specified sampling period (*wSampleTime*).



- When *wSampleTime* is shorter than the pulse period, the increasing number (*diSampleValue*) will be shown between 0 and 1 for each SampleTime.



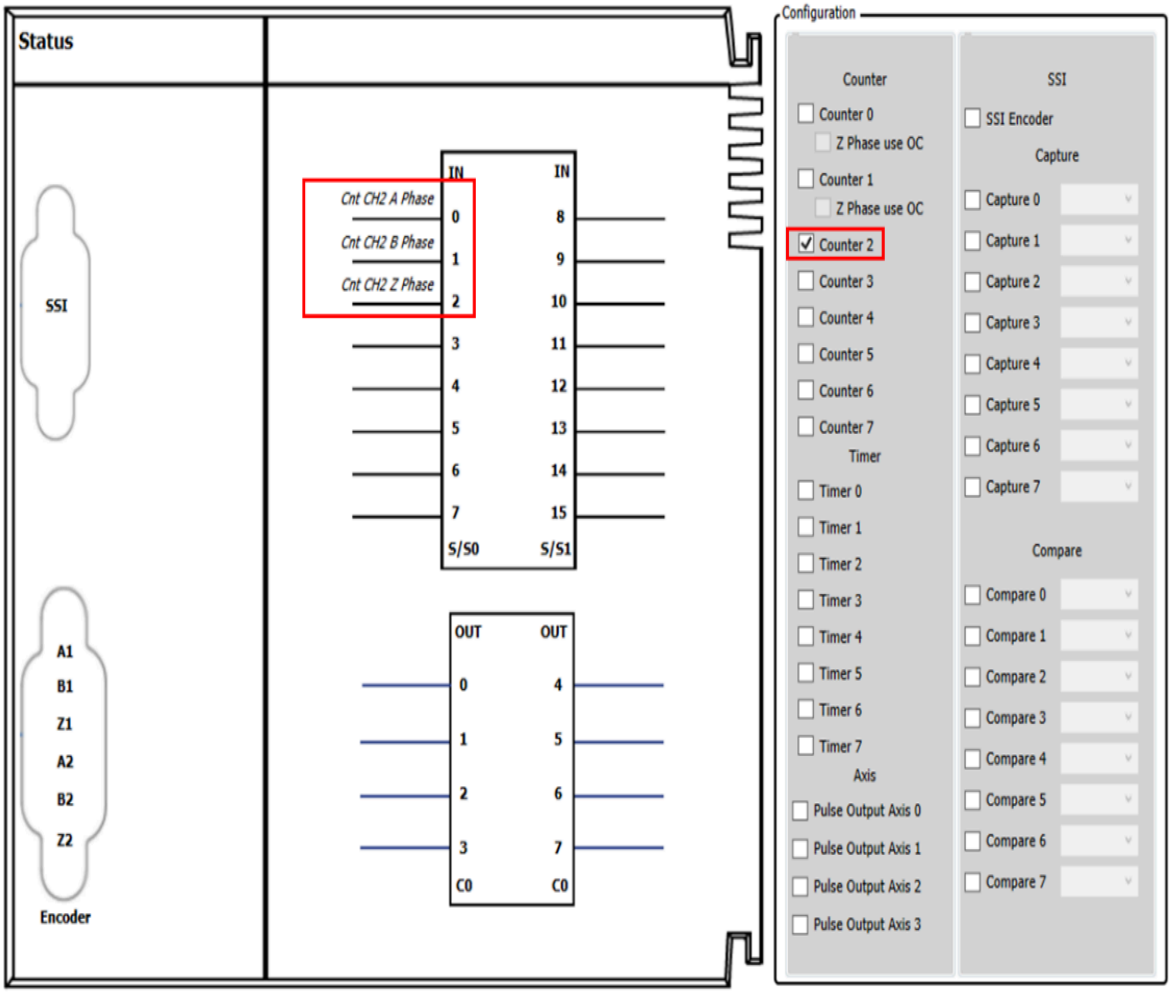
- **Troubleshooting**

  If an error occurs during the running of the instruction, *bError* will change to TRUE and Capture will stop. You can refer to ErrorID (Error Code) to address the problem.

- **Programming Example**

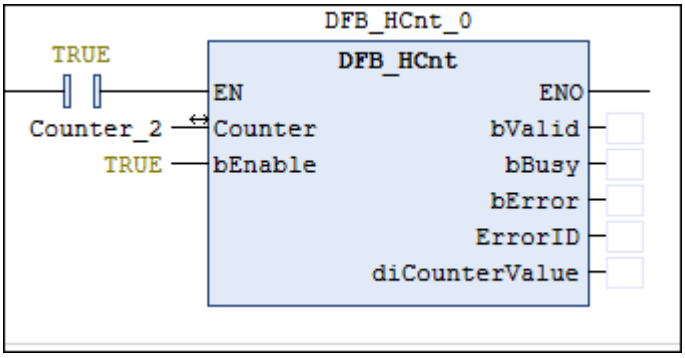
  This example uses DFB_HCnt and DFB_Sample to perform pulse counting during the sampling period.

  1. As the following figure shows, select a Counter (**Counter 2**) in **Hardware IO Configuration** and you will see the input points (e.g. IN 0, IN 1, IN 2) matched to the corresponding encoder A, B, Z phase outputs, which the wiring should follows the configuration so as to perform the normal function of high speed counting.

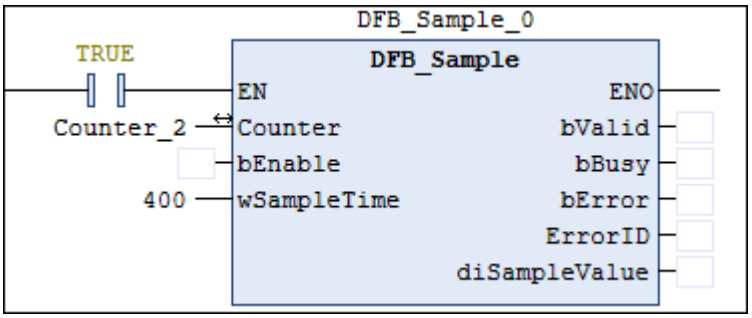


  2. After using the FB DFB_HCnt in the POU to activate the high-speed counter (*bEnable* = TRUE), it starts receiving and counting the pulses from the external signals(IN 0, IN 1) based on the counting mode set in **Counter Configuration**, then the counter value will be displayed in the

output diCounterValue. In addition, you should make sure that the mode of sending pulses from the external signal source matches the counting mode so as to get the correct counter values.



3. After enabling DFB_Sample in the POU (bEnable = TRUE), the FB starts counting the increasing number of the pulse counter value during each sampling period.



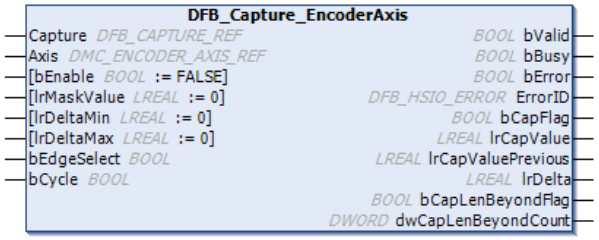
4. Refer to *AX-3 Series Operation Manual* for more details related to the settings and operation of **Counter Configuration**.

- **Supported Devices**
  - AX-3 series (except for AX-300)
- **Library**
  - DL_BuiltInIO.library

    Note: From version 1.0.5.0, library DL_BuiltInIO_AX3 is changed to DL_BuiltInIO.

3.7. DFB_Capture_EncoderAxis

DFB_Capture_EncoderAxis triggers and captures the encoder command value of the specified high-speed counter based on the external signal.

FB/FC	Instruction	Graphic Expression	ST Language
FB	DFB_Capture_EncoderAxis		DFB_Capture_EncoderAxis_0_instance (Capture:= , Axis:= , bEnable:= , lrMaskValue:= , lrDeltaMin:= , lrDeltaMax:= , bEdgeSelect:= , bCycle:= , bValid=> , bBusy=> , bError=> , ErrorID=> , bCapFlag=> , lrCapValue=> , lrCapValuePrevious=> , lrDelta=> , bCapLenBeyondFlag=> , dwCapLenBeyondCount=>);

- **Inputs**

Name	Function	Data Type	Setting Value (Default value)	Timing for Updating
Axis	Designate the source of the specified high-speed counter.	DMC_ENCODER_AXIS_REF*1	DMC_ENCODER_AXIS_REF (cannot be null)	–
bEnable	Run the instruction when *bEnable* changes to TRUE.	BOOL	TRUE/FALSE (FALSE)	–
lrMaskValue	Define the mask range of Capture.	LREAL	Positive number or 0 (0)	When *bEnable* shifts to TRUE, update the parameter of uiMaskValue

Name	Function	Data Type	Setting Value (Default value)	Timing for Updating
lrDeltaMin*2	Define the minimum difference between each Capture*2.	LREAL	Positive number, negative number or 0 (0)	When *bEnable* shifts to TRUE and *Busy* is FALSE
lrDeltaMax*2	Define the maximum difference between each Capture*2.	LREAL	Positive number, negative number or 0 (0)	When *bEnable* shifts to TRUE and *Busy* is FALSE
bEdgeSelect*3	Define the rising edge or falling edge capture	BOOL	TRUE/FALSE (FALSE)	When bEnable shifts to TRUE
bCycle*3	Define whether capture values continuously	BOOL	TRUE/FALSE (FALSE)	When bEnable shifts to TRUE

*** Note**:

1. DMC_ENCODER_AXIS_REF (FB): This function block works as the driver interface of the encoder, which contains the parameter calls of the encoder axis and the driver.
2. Once *diDeltaMin* and *diDeltaMax* are set to 0, the system will not check whether the capture range is appropriate.
3. This parameter is only applicable to AX-332E.

- **Outputs**

Name	Function	Data Type	Output Range (Default value)
bValid	TRUE when the output value is valid	BOOL	TRUE/FALSE (FALSE)
bBusy	TRUE when the instruction is enabled	BOOL	TRUE/FALSE (FALSE)
bError	TRUE if an error occurs	BOOL	TRUE/FALSE (FALSE)
ErrorID	Indicates the error code if an error occurs.	DFB_HSIO_ERROR*	DFB_HSIO_ERROR (DFB_HSIO_NO_ERR)
bCapFlag	Indicates that the current Capture is valid. (The flag shifts to TRUE for one scan cycle and will be reset immediately)	BOOL	TRUE/FALSE (FALSE)
lrCapValue	The captured value	DINT	Positive number, negative number or 0 (0)
lrCapValuePrevious	The previous captured value	DINT	Positive number, negative number or 0 (0)

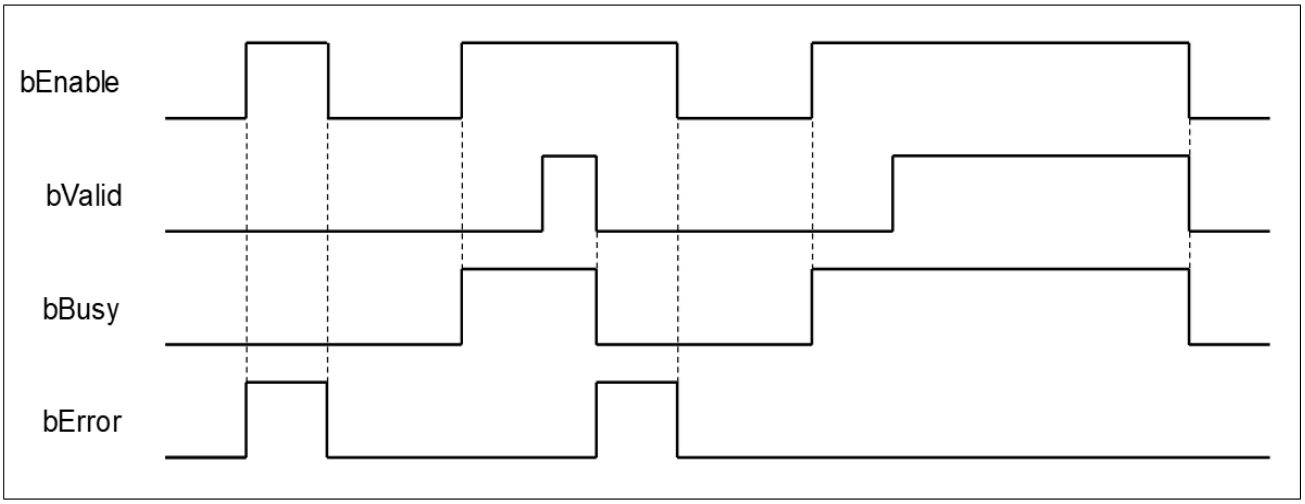
Name	Function	Data Type	Output Range (Default value)
lrDelta	The difference between the previous and the current captured values.	DINT	Positive number, negative number or 0 (0)
bCapLenBeyondFlag	Indicates that a capture is failed. (The flag shifts to TRUE for one scan cycle and will be reset immediately)	BOOL	TRUE/FALSE (FALSE)
dwCapLenBeyondCount	Counts the number of the failed Capture.	DWORD	Positive number or 0 (0)

***Note**: DFB_HSIO_ERROR: Enumeration (Enum)

- **Output Updating Timing**

Name	Timing for shifting to TRUE	Timing for shifting to FALSE
bValid	When *bEnable*is TRUE and the output values are valid after one scan cycle	• When *bEnable* shifts to FALSE • When *bError* shifts to TRUE
bBusy	When *bEnable* shifts to TRUE	• When *bEnable* shifts to FALSE • When *bError* shifts to TRUE
bError	When an error occurs during the running or input values are incorrect	When *bEnable* shifts from TRUE to FALSE (Error code is cleared)
ErrorID		
bCapFlag	Continuously update when *bValid* is TRUE	Continuously update when *bValid* is TRUE
lrCapValue	Continuously update when *bValid* is TRUE	Continuously update when *bValid* is TRUE
lrCapValuePrevious	Continuously update when *bValid* is TRUE	Continuously update when *bValid* is TRUE
lrDelta	Continuously update when *bValid* is TRUE	Continuously update when *bValid* is TRUE
bCapLenBeyondFlag	Continuously update when *bValid* is TRUE	Continuously update when *bValid* is TRUE
dwCapLenBeyondCount	Continuously update when *bValid* is TRUE	Continuously update when *bValid* is TRUE

- **Timing Diagram**



- **Inputs/Outputs**

Name	Function	Data Type	Setting Value (Default value)	Timing for Updating
Capture	Reference to the source of high speed compture	DFB_CAPTURE_REF (FB)*	DFB_CAPTURE_REF (Cannot be null.)	When *bEnable* shifts to TRUE and *bBusy* is FALSE.

***Note**: DFB_CAPTURE_REF (FB): This function block works as the driver interface of the high-speed Capture, which contains the parameter calls of the Capture and the driver.

- **Function**

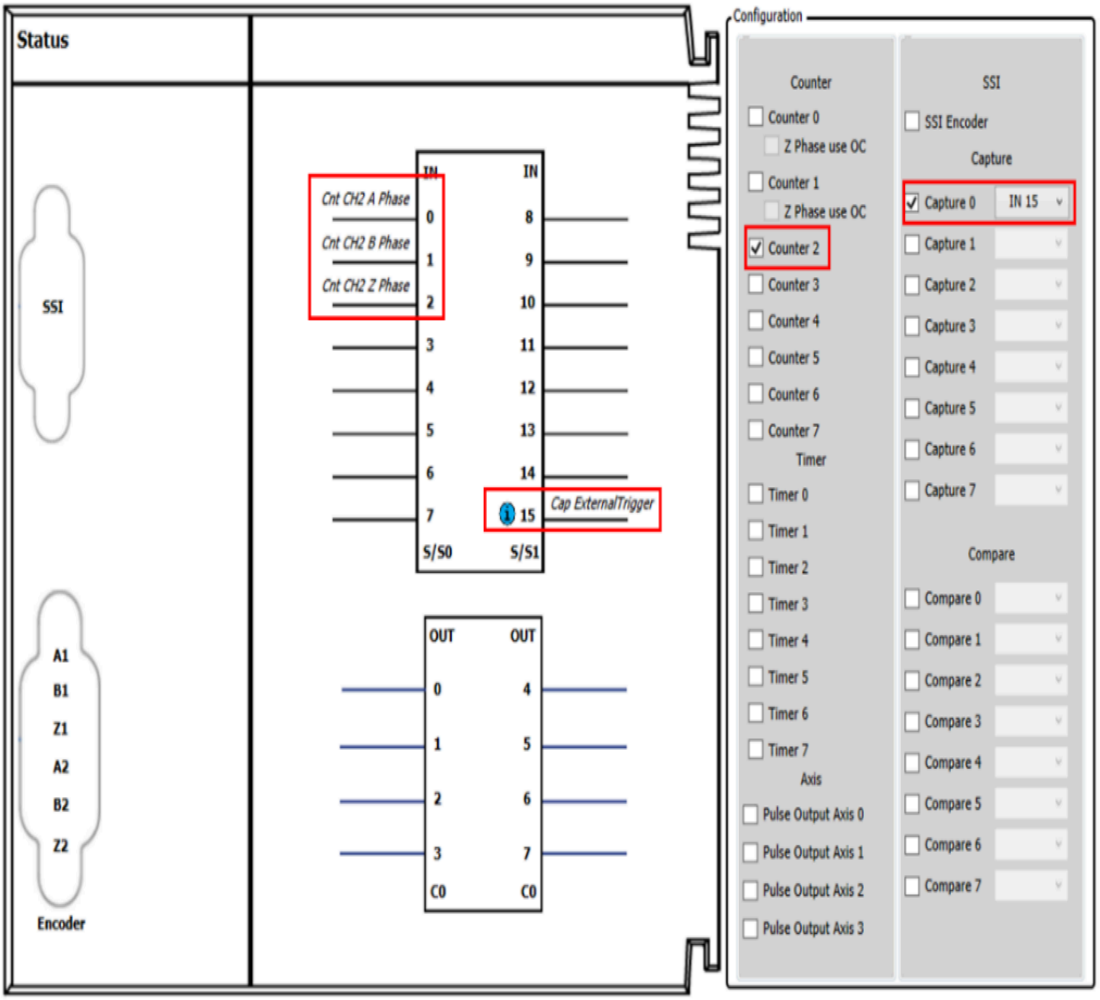
  This function is supported in DL_BuiltInIO V1.1.0.5 or later.

  Refer to the DFB_Capture function for more descriptions.

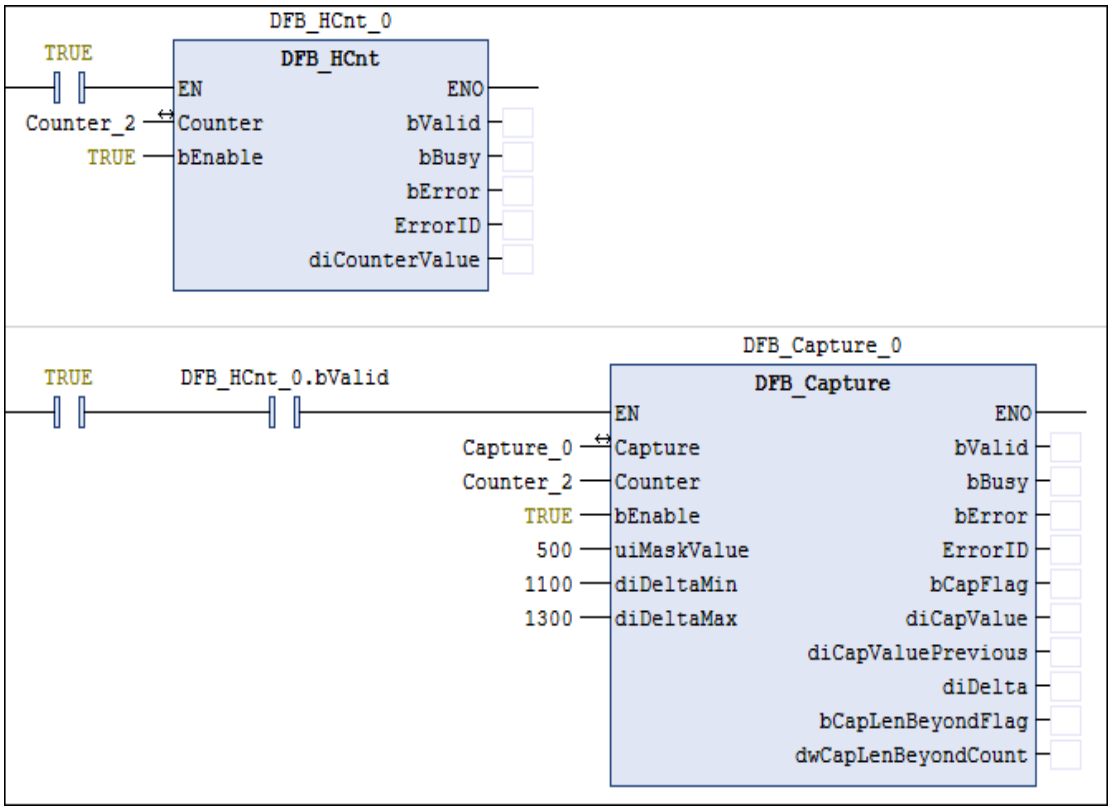
- **Troubleshooting**

  If an error occurs while running the instruction, *bError* will change to TRUE, and Capture will stop. You can refer to ErrorID (Error Code) to address the problem.

- **Programming Example**
  - This example uses DFB_HCnt and DFB_Compare in AX-308E to perform the Capture function.
    1. As the following figure shows, select a **Counter** and a **Capture** for **Hardware IO Configuration** in BuiltIn_IO and set the trigger of Capture to a signal input on the hardware (e.g., IN15).



    2. Enable the FB DFB_Capture (*bEnable* = TRUE) after using the FB DFB_HCnt to activate the high-speed counter (*bEnable* = TRUE) in the POU, then the current counter value will be captured and shown on the *diCapValue* output of DFB_Capture after the external signal (IN15) being triggered.



3. Refer to section 7.7.7.4 Pulse Encoder Settings of *AX-3 Series Operation Manual* for more details about the settings and operation of **Hardware IO Configuration**.

- **Supported Devices**
  - AX-3 series (except for AX-300), AX-C
- **Library**
  - DL_BuiltInIO.library

3.8. DFB_Compare_EncoderAxis

DFB_Compare_EncoderAxis compares the specified encoder and settings. When the comparison result is TRUE, output or reset the device.

FB/FC	Instruction	Graphic Expression	ST Language
FB	DFB_Capture_EncoderAxis		DFB_Compare_EncoderAxis_instance (Compare:= , Axis:= , bEnable:= , Mode:= , OutputAction:= , wRefreshCycle:= , lrCmpValue:= , lrTablePosition:= , uiTableSize:= , bValid=> , bBusy=> , bError=> , ErrorID=> , TableState=>);

- **Inputs**

Name	Function	Data Type	Setting Value (Default value)	Timing for Updating
Axis	Specify the built–in encoder	DMC_ENCODER_AXIS_REF*1	DMC_ENCODER_AXIS_REF (cannot be null)	–
bEnable	Run the Instruction when bEnable changes to TRUE.	BOOL	TRUE/FALSE (FALSE)	–
Mode	Specify the comparison criteria	DFB_COMPARE_MODE*2	0：Equal(=) 1：Bigger_Equal(≧) 2：Smaller_Equal(≦) (Equal)	When *bEnable* shifts to TRUE and *Busy* is FALSE
OutputAction*3	Set the output mode	DFB_COMPARE_OUTPUT_ACTION	0: SET 1: RESET (SET)	When *bEnable* shifts to TRUE and *Busy* is FALSE
wRefreshCycle	Define the cycle time to refresh	WORD	Positive number or 0 (0)	When *bEnable* shifts to TRUE and *Busy* is FALSE

Name	Function	Data Type	Setting Value (Default value)	Timing for Updating
	the status of the output device.			
lrCmpValue	Specifies the compare value	LREAL	Positive number, negative number or 0 (0)	When *bEnable* shifts to TRUE and *Busy* is FALSE
lrTablePosition*4	Specify the compare value array	ARRAY OF LREAL	Positive number 0–255	When *bEnable* shifts to TRUE and *Busy* is FALSE
diTableSize*4	Specify the compare value array scope	DINT	Positive number 0–255	When *bEnable* shifts to TRUE and *Busy* is FALSE

***Note**:

1. DMC_ENCODER_AXIS_REF (FB): This function block works as the driver interface of the encoder, which contains the parameter calls of the encoder axis and the driver.
2. DFB_COMPARE_MODE: Enumeration (Enum).
3. Except for AX-332E, other AX-3 series with DL_BuiltInIO_AX3 version V1.0.7.5 or later supports this function.
4. This parameter is only applicable to AX-332E.

- **Outputs**

Name	Function	Data Type	Output Range (Default value)
bValid	TRUE when the output value is valid	BOOL	TRUE/FALSE (FALSE)
bBusy	TRUE when the instruction is enabled	BOOL	TRUE/FALSE (FALSE)
bError	TRUE if an error occurs	BOOL	TRUE/FALSE (FALSE)
ErrorID	Indicates the error code if an error occurs.	DFB_HSIO_ERROR*	DFB _HSIO_ERROR (DFB _HSIO_NO_ERR)
TableState	The hight–speed comparator state table	DFB_COMPARE_TABLE_STATE	–

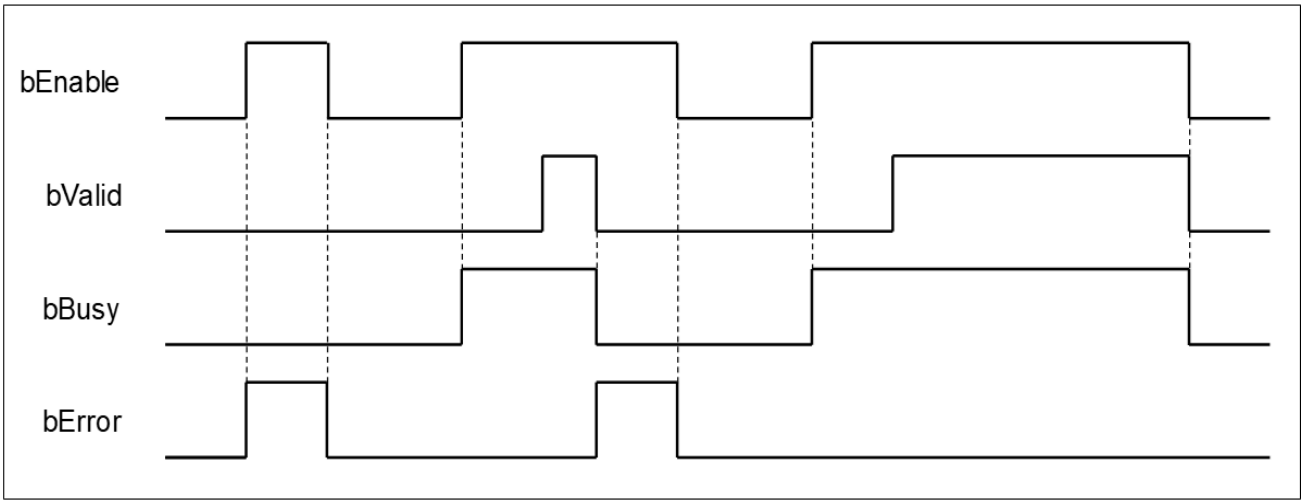
- **DFB_COMPARE_TABLE_STATE**

Name	Function	Data Type
xEmpty	The comparing table has not been filled in with the comparison items or all the high-speed comparison conditions are valid.	BOOL
xFull	The comparing table was filled with 256 comparison items and the high-speed compare was not established.	BOOL
xAlmostEmpty	The comparing table is almost empty.	BOOL
uiDataCounter	The number of comparison items in the comparing table.	UINT
udiOSTriCounter	Count the number of times a high-speed comparison is triggered.	UDINT

- **Output Updating Timing**

Name	Timing for shifting to TRUE	Timing for shifting to FALSE
bValid	When the values at the outputs are valid after *bEnable* being TRUE for one scan cycle	• When *bEnable* shifts to FALSE • When *bError* shifts to TRUE
bBusy	When *bEnable* is triggered by the rising edge	• When *bEnable* shifts to FALSE • When *bError* shifts to TRUE
bError	When an error occurs during the running or there are incorrect input values	When *bEnable* shifts from TRUE to FALSE (Error code is cleared)
ErrorID		
TableState	Continuously updated when *bValid* is TRUE.	Continuously updated when *bValid* is TRUE.

- **Timing Diagram**



- **Inputs/Outputs**

Name	Function	Data Type	Setting Value (Default value)	Timing for Updating
Compare	Reference to the source of high-speed comparator.	DFB_COMPARE_REF (FB)*	DFB_COMPARE_REF (Cannot be null.)	When *bEnable* shifts to TRUE and *bBusy* is FALSE

***Note**: DFB_COMPARE_REF (FB): As the I/O interface of the high-speed comparator to perform actions include parameter adjustment and the driver.

- **Function**

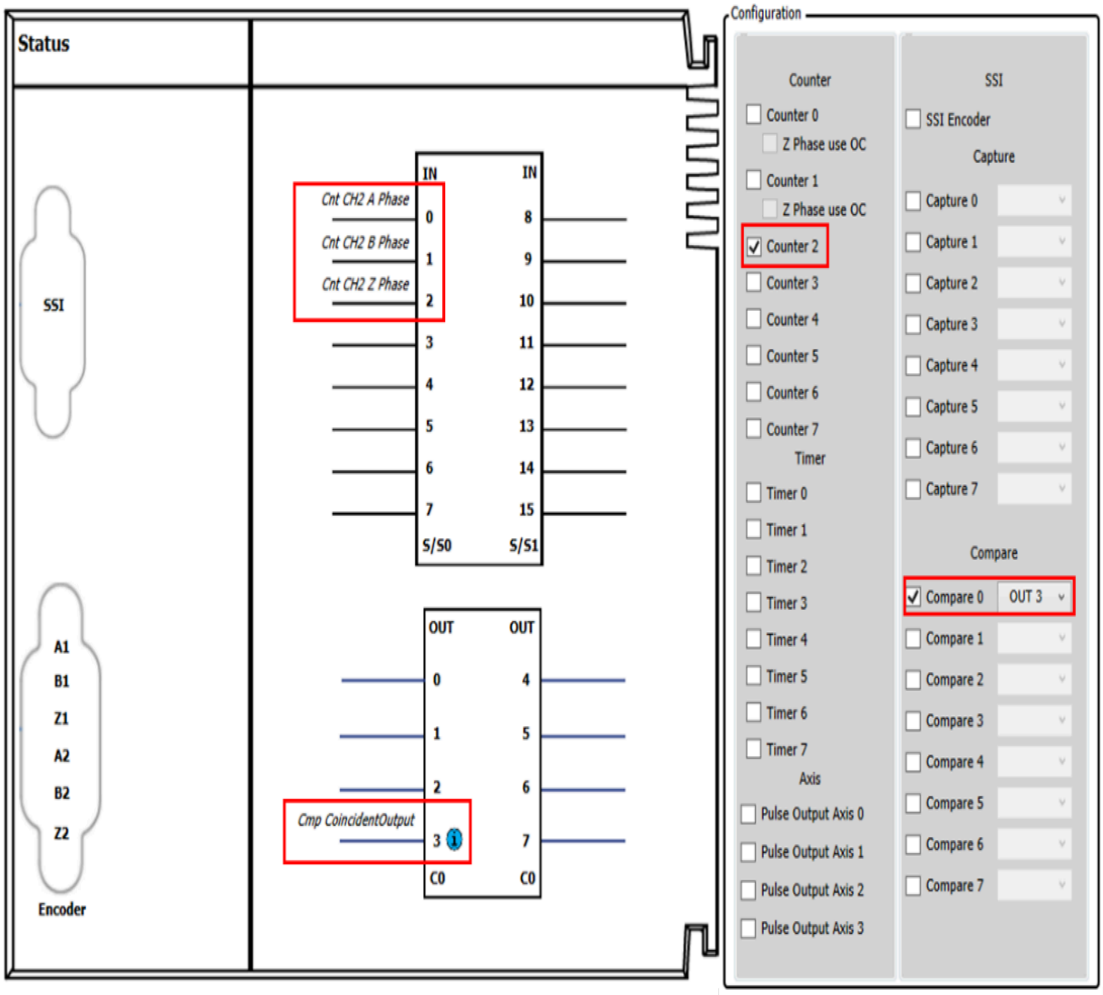
  This function is supported in DL_BuiltInIO V1.1.0.5 or later.

  Refer to the DFB_Compare function for more descriptions.

- **Troubleshooting**

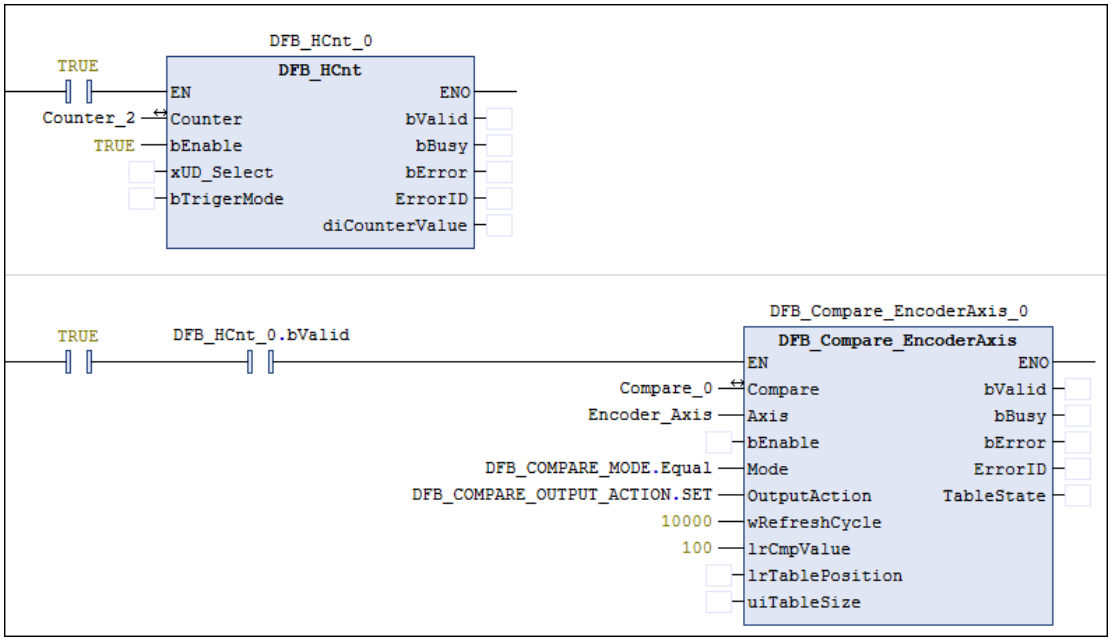
  If an error occurs while running the instruction, *bError* will change to TRUE and Campare will stop. You can refer to ErrorID (Error Code) to address the problem.

- **Programming Example**
  - This example uses DFB_HCnt and DFB_Compare in AX-308E to perform the Capture function.
    1. As the following figure shows, select a Counter and a Compare for Hardware IO Configuration in BuiltIn_IO and set the trigger of Compare to an output signal on the hardware (e.g. OUT3).



    2. Enable the FB DFB_Capture (*bEnable* = TRUE) after using the FB DFB_HCnt to activate the high-speed counter (*bEnable* = TRUE) in the POU, then start the DFB_Compare_EncoderAxis function block (bEnable=TRUE). After the comparison is comparasion is started (Encoder_Axis.fActPosition =

DFB_Compare_EncoderAxis_0.lrCmpValue), the output device (OUT3) will output the signal.



3. Refer to section 7.7.7.4 Pulse Encoder Settings of *AX-3 Series Operation Manual* for more details related to the settings and operation of Hardware IO Configuration.

- **Supported Devices**
  - AX-3 series (except for AX-300), AX-C
- **Library**
  - DL_BuiltInIO.library

3.9. Error Codes and Troubleshooting

The following table lists the error codes of the FBs and the actions to correct the error.

Description	Cause of Error	Corrective Action
DFB_HSIO_NO_ERR	No error messages	
DFB_CAP_INVALID_CAPTURE_REF	The variable type set for the FB input is not Capture_REF.	After selecting Capture in IO Configuration, input the variable of IEC Object to the *Capture* input of DFB_Capture.
DFB_CAP_INVALID_COUNTER_REF	The variable type set for the FB input is not Counter_REF	After selecting Counter in IO Configuration, input the variable of IEC Object to the "Counter" input of DFB_Capture.
DFB_CAP_INVALID_VALUE_SETTING	The mask range of DFB_Capture (*uiMaskValue*) exceeds the rotation range of the axis.	Reset the input value of *uiMaskValue* to be in the rotation range of encoder axis. [0–EncoderAxis.Modulo Value]
DFB_CAP_INVALID_DELTARANGE	When a rotary axis is used as the encoder axis, the min/max difference between each Capture exceeds the rotation range.	Reset the input value of *diDeltaMax* or *diDeltaMin* to be in the rotation range of encoder axis. [0–EncoderAxis.Modulo Value]
DFB_CAP_CAPTURE_ALREADY_ENABLE	The high-speed capture device has been activated.	Check if this Capture device is currently being used by another DFB_Capture.
DFB_CAP_DRIVE_ERROR	Errors occur in the Capture device or Count device driver.	Check the error message on the BuiltIn_IO page and refer to the AX-3 operational manual to troubleshoot the errors.
DFB_CMP_INVALID_COMPARE_REF	The variable type set for the FB input is not Compare_REF.	After selecting Compare in IO Configuration, input the variable of IEC Object to the *Compare* input of DFB_Compare.
DFB_CMP_INVALID_COUNTER_REF	The variable type set for the FB input is not Counter_REF.	After selecing Counter in IO Configuration , input the variable of IEC Object to the Counter input of DFB_Capture.
DFB_CMP_INVALID_CMPVALUE	When a rotary axis is used as the encoder axis, the input "diCompareValue" exceeds the rotation range.	Reset the input value of diCompareValue to be in the rotation range of encoder axis. [0–EncoderAxis.Modulo Value]
DFB_CMP_INVALID_REFRESHCYCLE	The input *wRefreshCycle* exceeds the range of –30000 (Unit: 0.1us).	Set the value of *wRefreshCycle* within the range of 0–30000.
DFB_CMP_COMPARE_ALREADY_ENABLE	The high-speed comparator has been activated.	Check if this Compare device is currently being used by another DFB_Compare.
DFB_CMP_DRIVE_ERROR	Errors occur in the Compare device or Count device driver.	Check the error message on the BuiltIn_IO page and refer to

Description	Cause of Error	Corrective Action
		the AX-3 operational manual to troubleshoot the errors.
DFB_HC_INVALID_COUNTER_REF	The variable type set for the FB input is not Counter_REF.	After selecting Counter in IO Configuration, input the variable of IEC Object to the *Counter* input of DFB_Hcnt.
DFB_HC_COUNTER_ALREADY_ENABLE	The high-speed counter has been activated.	Check if this Counter device is currently being used by another DFB_HCnt.
DFB_HC_COUNTER_REF_CHANGED_DURING_OPERATION	The input value of “Counter” is changed while the FB is running.	Check if the value of the input Counter changes after the FB DFB_HCnt running.
DFB_HC_COUNTER_DRIVE_ERROR	Errors occur in the Count device driver.	Check the error message on the BuiltIn_IO page and refer to the AX-3 operational manual to troubleshoot the errors.
DFB_HT_INVALID_TIMER_REF	The variable type set for the FB input is not Timer_REF.	After selecting Timer in IO Configuration, input the variable of IEC Object to the *Timer* input of DFB_HTmr.
DFB_HT_TIMER_ALREADY_ENABLE	The high-speed timer has been activated.	Check if this Timer device is currently being used by another DFB_HTmr.
DFB_HT_TIMER_REF_CHANGED_DURING_OPERATION	The input value of “Timer” is changed while the FB is running.	Check if the value of the input Timer changes after the FB DFB_HTmr running.
DFB_HT_TIMER_DRIVE_ERROR	Errors occur in the Timer device driver.	Check the error message on the BuiltIn_IO page and refer to the AX-3 operational manual to troubleshoot the errors.
DFB_PV_INVALID_COUNTER_REF	The variable type set for the FB input is not Counter_REF.	After selecting Counter in IO Configuration, input the variable of IEC Object to the *Counter* input of DFB_PresetValue.
DFB_PV_NOT_ENABLE_EXTERNAL_TRIGGER	“External Trigger” in Counter mode configuration is not selected while the input *TriggerType* of DFB_PresetValue is set to EXTERNAL_TRIGGER.	Check the box of External Trigger on the **Counter configuration** page.
DFB_PV_PREVIOUS_PRESET_NOT_DONE	The preset value function of the counter has been used by other DMC_PresetValue FBs.	Run the function block after the DFB_PresetValue function block being used by the Counter has completed running.
DFB_PV_CANNOT_PRESET_WHEN_SAMPLING	The counter is running DFB_Sample.	Disable DFB_Sample of the counter to disable the Sample function in this counter.

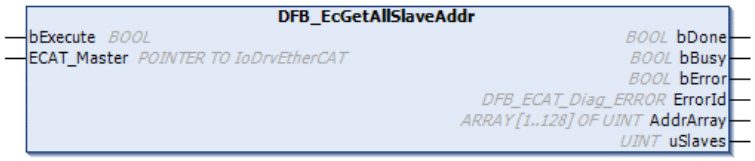
Description	Cause of Error	Corrective Action
DFB_PV_SETRING_NOT_DONE	The counter is running DFB_SetRing and not completed.	Run the DFB_PresetValue after the DFB_SetRing counter has completed running.
DFB_PV_INVALID_PRESET_VALUE	When a rotary axis is used as the encoder axis, the input *diPresetValue* exceeds the rotation range.	Reset the input value of *diPresetValue* to be in the rotation range of the encoder axis. [0–EncoderAxis.Modulo Value]
DFB_PV_COUNTER_REF_CHANGED_DURING_OPERATION	The input value of “Counter” is changed while the FB is running.	Check if the value of the input Counter changes after the FB DFB_PresetValue runs
DFB_PV_COUNTER_DRIVE_ERROR	Errors occur in the Timer device driver.	Check the error message on the BuiltIn_IO page and refer to the AX-3 operational manual to troubleshoot the errors.
DFB_SP_INVALID_COUNTER_REF	The variable type set for the FB input is not Counter_REF.	After make selecting Counter in IO Configuration, input the variable of IEC Object to the “Counter” input of DFB_Sample.
DFB_SP_COUNTER_NOT_ENABLE	DFB_Counter has not enabled the high-speed counter.	Ensure the counter device has been enabled by DFB_HCnt before running the FB DFB_Sample.
DFB_SP_ALREADY_SAMPLING	The counter is running DFB_Sample.	Check if this counter device is currently being used by another DFB_Sample.
DFB_SP_PRESET_NOT_DONE	The counter is running DFB_PresetValue and not completed.	Run DFB_Sample after completing running the DFB_PresetValue function block being used by the counter.
DFB_SP_INVALID_SAMPLE_TIME	The input *wSampleTime* of DFB_Sample exceeds the range of 10–65535.	Reset the input value of *wSampleTime* in the range of 10–65535.
DFB_SP_COUNTER_REF_CHANGED_DURING_OPERATION	The input value of Counter is changed while the FB is running.	Check if the value of the input Counter changes after the FB DFB_Sample running.
DFB_SP_COUNTER_DRIVE_ERROR	Errors occur in the Counter device driver.	Check the error message on the BuiltIn_IO page and refer to the AX-3 operational manual to troubleshoot the errors.

Chapter 4: EtherCAT Network Instructions

FB/FC	Description	Library
DFB_EcGetAllSlaveAddr	Gets all slave addresses.	DL_EtherCAT_Diag
DFB_EcGetSlaveCoun	Gets the number of slaves in the tree.	DL_EtherCAT_Diag
DFB_EtherCATLink_Diag	Displays the online status of all EtherCAT slaves.	DL_EtherCAT_Diag
DFB_GetAllECATSlaveInfo	Gets all the slave information.	DL_EtherCAT_Diag
DFB_GetECATMasterError	Gets the error code for the EtherCAT network connection error.	DL_EtherCAT_Diag
DFB_GetECATasterState	Gets the EtherCAT master online status.	DL_EtherCAT_Diag
DFB_ResetECATMaster	Resets the EtherCAT master with an abnormal online status.	DL_EtherCAT_Diag
DFB_ResetECATSlave	Resets the slave with an abnormal online status.	DL_EtherCAT_Diag

4.1. DFB_EcGetAllSlaveAddr

DFB_EcGetAllSlaveAddr gets all the slave addresses.

FB/FC	Instruction	Graphic Expression	ST Language
FB	DFB_EcGetAllSlaveAddr		DFB_EcGetAllSlaveAddr (bExecute :=, ECAT_Master :=, bDone =>, bBusy =>, bError =>, ErrorID =>, AddrArray =>, uSlaves =>,);

- **Inputs**

Name	Function	Data Type	Setting Value (Default value)	Timing for Updating
bExecute	Run the instruction when *bExecute* changes to TRUE.	BOOL	TRUE/FALSE (FALSE)	–
ECAT_Master	Name of the master to be used	POINTER TO IoDrvEtherCAT	(0)	When *bExecute* shifts to TRUE and *bBusy* is FALSE

- **Outputs**

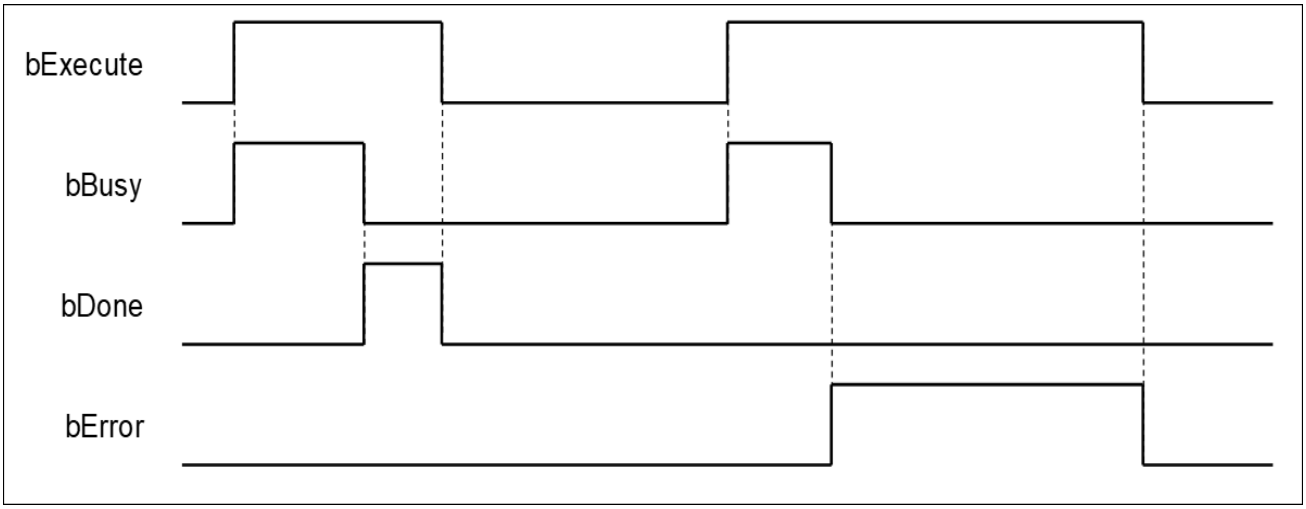
Name	Function	Data Type	Output Range (Default value)
bDone	The running of FB is completed.	BOOL	TRUE/FALSE (FALSE)
bBusy	TRUE when the instruction is enabled	BOOL	TRUE/FALSE (FALSE)
bError	TRUE if an error occurs	BOOL	TRUE/FALSE (FALSE)
ErrorID	Indicates the error code if an error occurs.	DFB_ECAT_Diag_ERROR*	DFB_ECAT_Diag_ERROR (DFB_ECAT_Diag_NO_ERROR)
AddrArray	Slave address array	UINT[1..128]	(0)
uSlaves	The number of slaves	UINT	0–128 (0)

***Note**: DFB_ECAT_Diag_ERROR: Enumeration (Enum)

- **Output Updating Timing**

Name	Timing for shifting to TRUE	Timing for shifting to FALSE
bDone	When the running of FB is completed	• When *bExecute* shifts to FALSE • If *bExecute* is FALSE and *bDone* shifts to TRUE, *bDone* will be TRUE for only one period and immediately shift to FALSE.
bBusy	When *bExecute* shifts to TRUE	• When *bDone* shifts to TRUE • When *bError* shifts to TRUE
bError	When an error occurs in the running conditions of the instruction	When *bExecute* shifts from TRUE to FALSE (Error code is cleared)
ErrorID		
AddrArray	When *bExecute* shifts to TRUE	When *bExecute* shifts to FALSE
uSlaves	When *bExecute* shifts to TRUE	When *bExecute* shifts to FALSE

- **Timing Diagram**



- **Function**

  When *bExecute* shifts to TRUE, the output *AddrArray* gives the addresses of all the EtherCAT slaves in the project tree, which supports up to 128 stations. Therefore, the maximum number of slave addresses output by *AddrArray* will be 128 given by the output *uSlaves*.

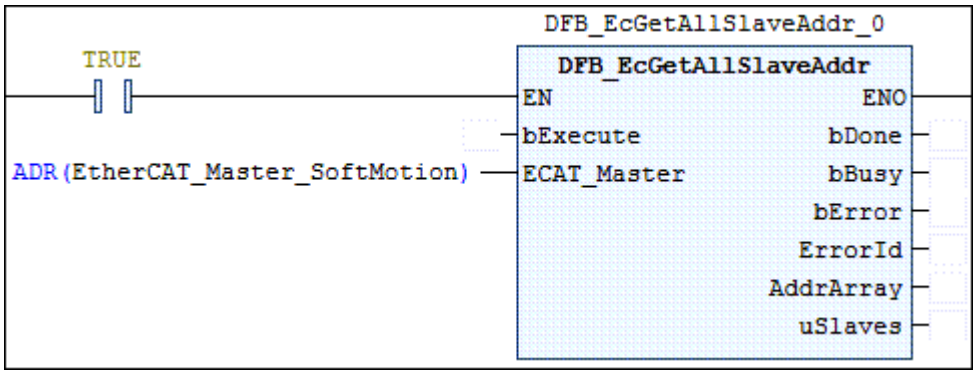
  The ECAT_Master parameter applies to DL_EtherCAT_Diag version 1.3.2.0 or later. The name of the master to be used must be specified. If ECAT_Master is not specified, the first EtherCAT master from top to bottom in the project will be used as the master.

- **Troubleshooting**

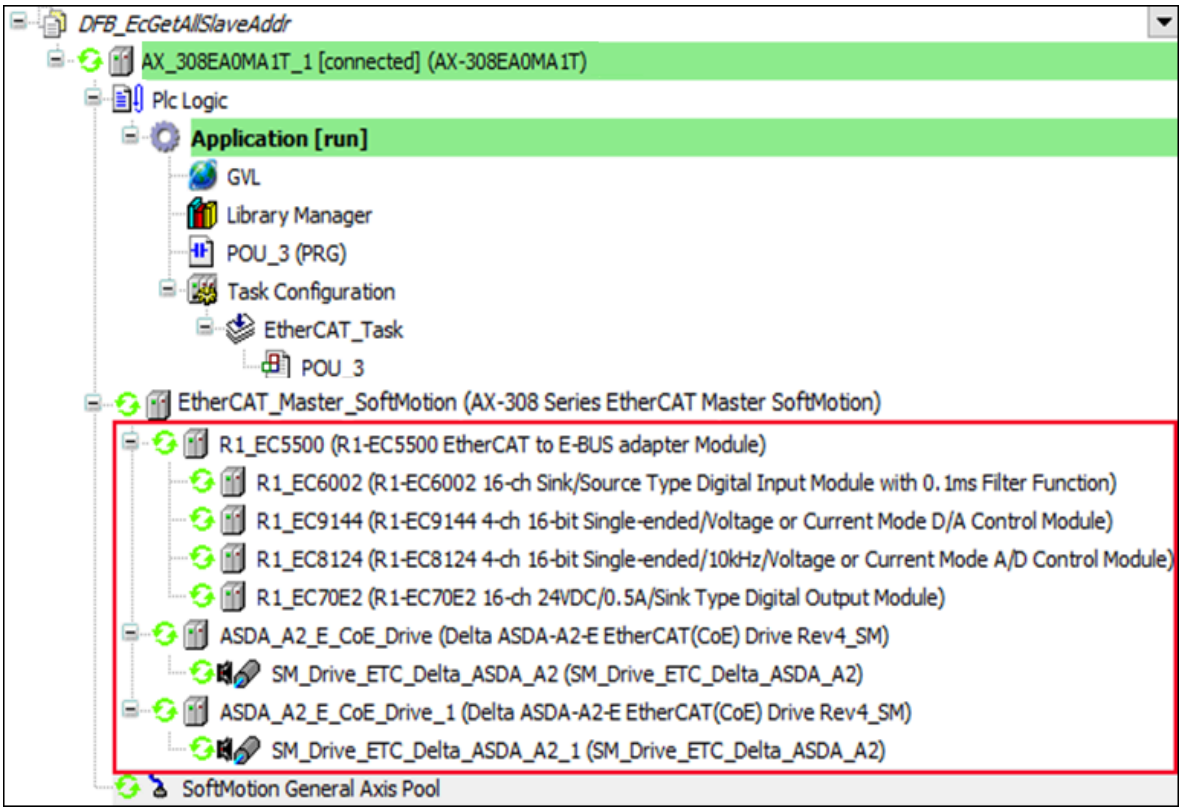
  If an error occurs while the running the instruction, *bError* will change to TRUE and Capture will stop. You can refer to *ErrorID* (Error Code) to troubleshoot the problem.

- **Programming Example**

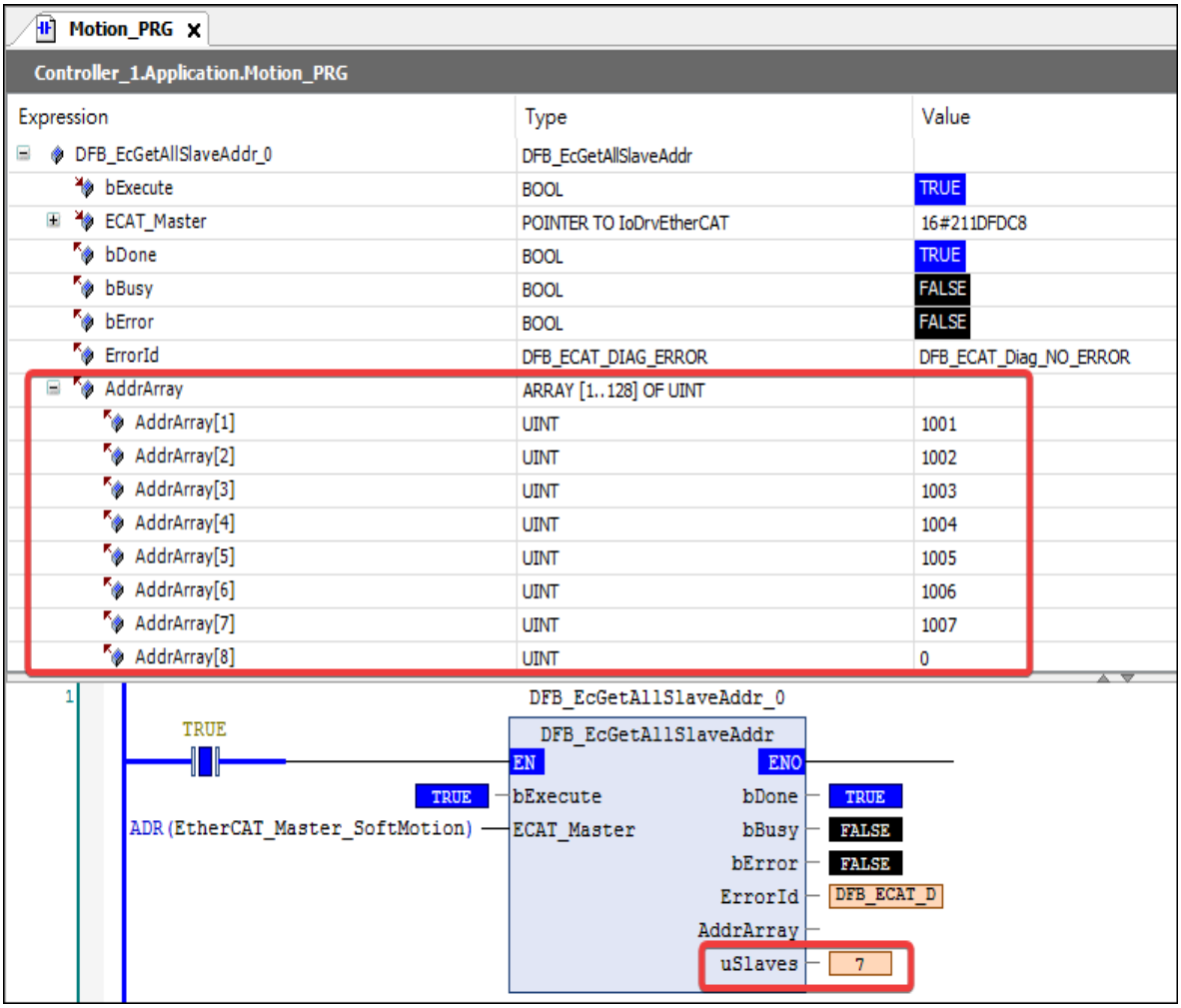
  The following example demonstrates the behavior of DFB_EcGetAllSlaveAddr.



1. There’re 7 EtherCAT slaves in the category EtherCAT_Master_SoftMotion.



2. After the input *bExecute* of DFB_EcGetAllSlaveAddr *bExecute* shifts to TRUE, the output of *AddrArray* is shown as below and the output value of *uSlaves* is 7.



- **Supported Devices**
  - AX-3 series motion controller, AX-8, AX-C
- **Library**
  - DL_EtherCAT_Diag.library

4.2. DFB_EcGetSlaveCount

DFB_EcGetSlaveCount gets the number of slaves that are connected to the master.

FB/FC	Instruction	Graphic Expression	ST Language
FB	DFB_EcGetSlaveCount		DFB_EcGetSlaveCount (bExecute :=, ECAT_Master :=, bDone =>, bBusy =>, bError =>, ErrorID =>, uSlaves =>,);

- **Inputs**

Name	Function	Data Type	Setting Value (Default value)	Timing for Updating
bExecute	Run the instruction when *bExecute* changes to TRUE.	BOOL	TRUE/FALSE (FALSE)	–
ECAT_Master	Name of the master to be used	POINTER TO IoDrvEtherCAT	(0)	When *bExecute* shifts to TRUE and *bBusy* is FALSE

- **Outputs**

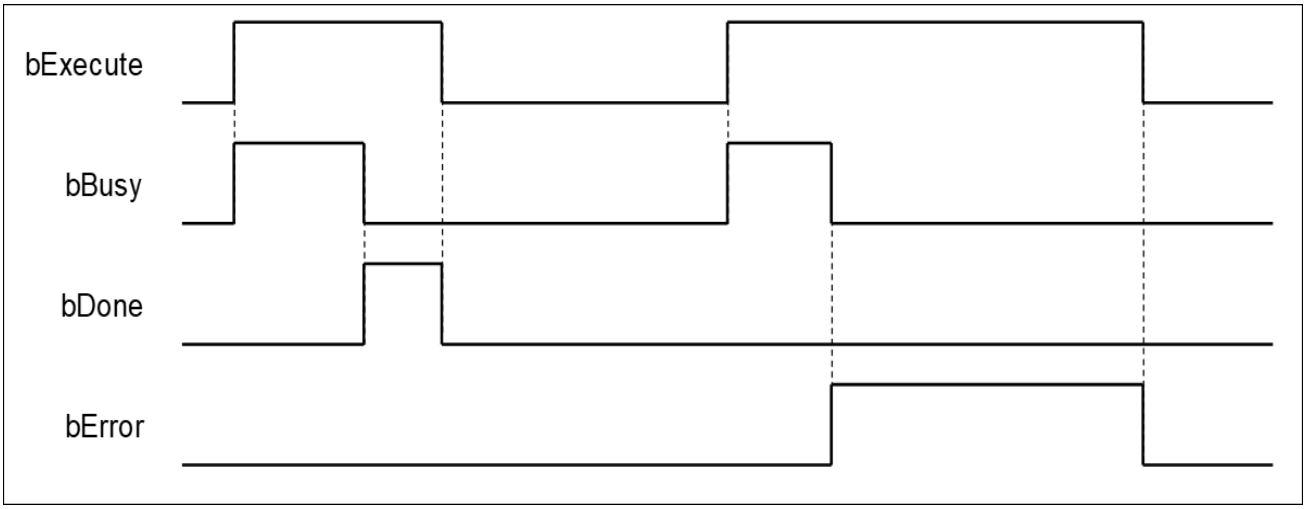
Name	Function	Data Type	Output Range (Default value)
bDone	The running of FB is completed.	BOOL	TRUE/FALSE (FALSE)
bBusy	TRUE when the instruction is enabled	BOOL	TRUE/FALSE (FALSE)
bError	TRUE if an error occurs	BOOL	TRUE/FALSE (FALSE)
ErrorID	Indicates the error code if an error occurs.	DFB_ECAT_Diag_ERROR*	DFB_ECAT_Diag_ERROR (DFB_ECAT_Diag_NO_ERROR)
uSlaves	The number of slaves	UINT	0–128 (0)

***Note**: DFB_ECAT_Diag_ERROR: Enumeration (Enum)

- **Output Updating Timing**

Name	Timing for shifting to TRUE	Timing for shifting to FALSE
bDone	When the running of FB is completed	• When *bExecute* shifts to FALSE • If *bExecute* is FALSE and *bDone* shifts to TRUE, *bDone* will be TRUE for only one period and immediately shift to FALSE.
bBusy	When *bEnable* shifts to TRUE	• When *bDone* shifts to TRUE • When *bError* shifts to TRUE
bError	When an error occurs in the running conditions of the instruction	When *bExecute* shifts from TRUE to FALSE (Error code is cleared)
ErrorID		
uSlaves	When *bExecute* shifts to TRUE	When *bExecute* shifts to FALSE

- **Timing Diagram**



- **Function**

  When *bExecute* shifts to TRUE, the output *uSlaves* gives the number of EtherCAT slaves in the project tree.

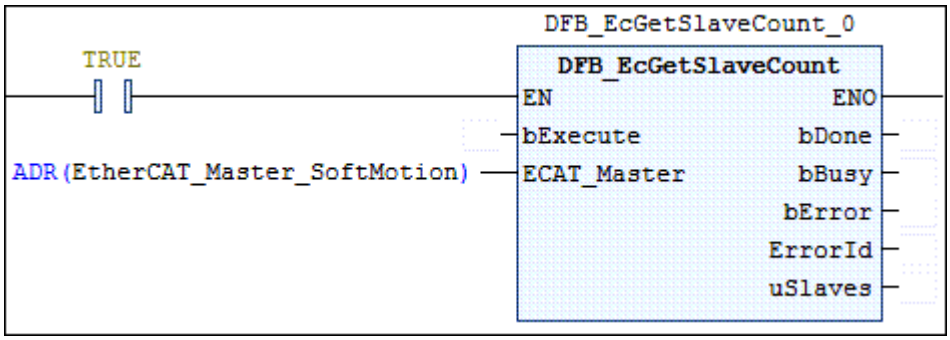
  The ECAT_Master parameter applies to DL_EtherCAT_Diag version 1.3.2.0 or later. The name of the master to be used must be specified. If ECAT_Master is not specified, the first EtherCAT master from top to bottom in the project will be used as the master.

- **Troubleshooting**

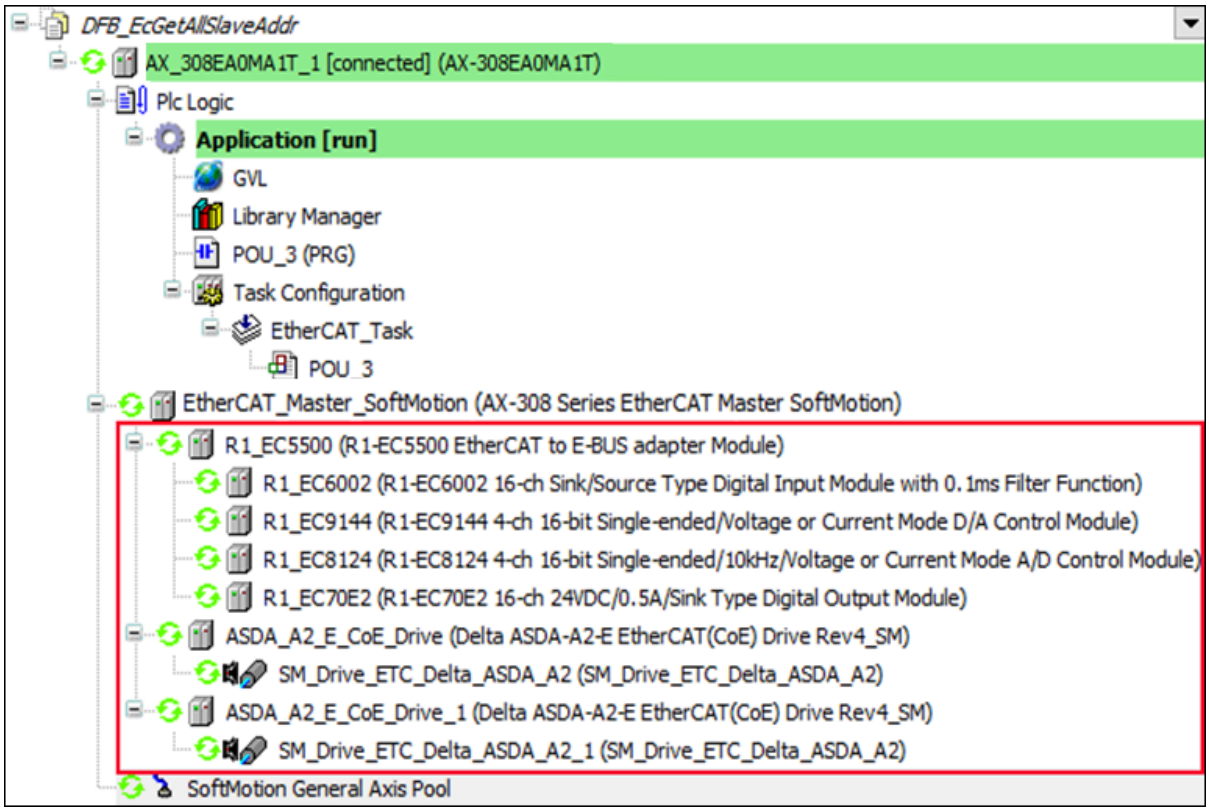
  If an error occurs during the running of the instruction, *bError* will change to TRUE and Capture will stop. You can refer to *ErrorID* (Error Code) to troubleshoot the problem.

- **Programming Example**

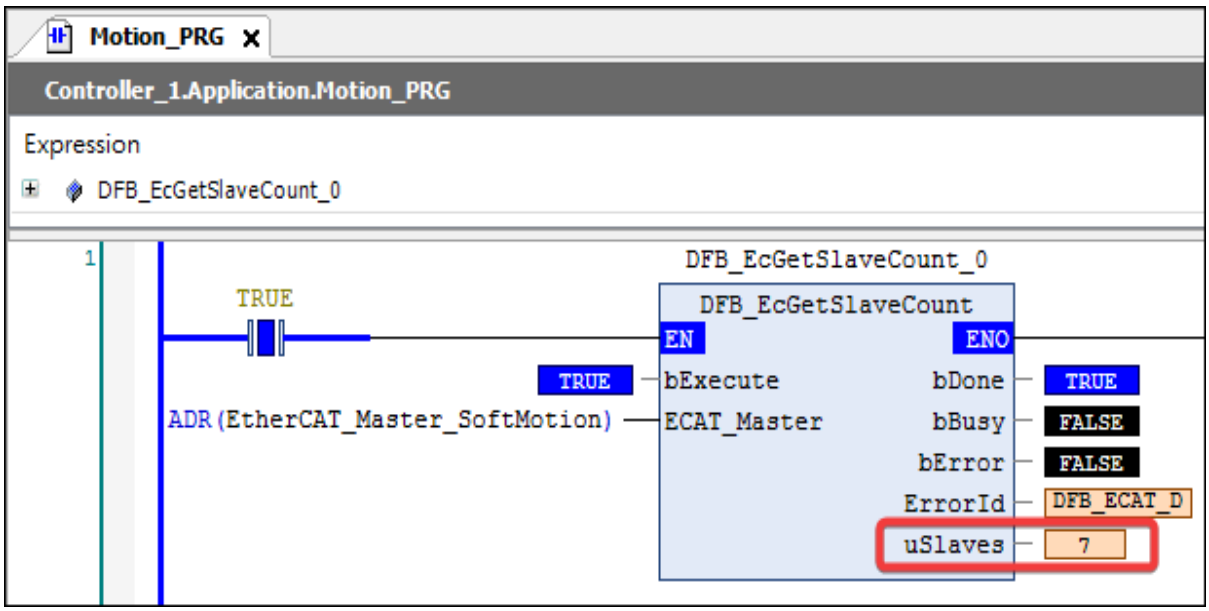
  The following example demonstrates the behavior of DFB_EcGetSlaveCount.



1. There're a total of 7 EtherCAT slaves in the category EtherCAT_Master_SoftMotion.



2. When the input *bExecute* of DFB_EcGetSlaveCount shifts to TRUE, the output value of *uSlaves* is 7.

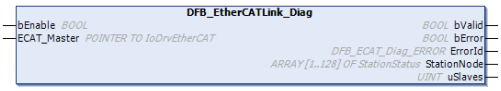


- **Supported Devices**
  - AX-3 series motion controller, AX-8, AX-C

- **Library**
  - DL_EtherCAT_Diag.library

4.3. DFB_EtherCATLink_Diag

DFB_EtherCATLink_Diag displays all the EtherCAT slave diagnostics.

FB/FC	Instruction	Graphic Expression	ST Language
FB	DFB_EtherCAT Link_Diag		DFB_EtherCATLink_Diag (bEnable :=, ECAT_Master :=, bValid =>, bError =>, ErrorID =>, StationNode => uSlaves =>);

- **Inputs**

Name	Function	Data Type	Setting Value (Default value)	Timing for Updating
bEnable	Run the instruction when *bEnable* changes to TRUE.	BOOL	TRUE/FALSE (FALSE)	–
ECAT_Master	Name of the master to be used	POINTER TO IoDrvEtherCAT	(0)	When *bEnable* shifts to TRUE and *bValid* is FALSE

- **Outputs**

Name	Function	Data Type	Output Range (Default value)
bValid	TRUE when the instruction is running.	BOOL	TRUE/FALSE (FALSE)
bError	TRUE if an error occurs.	BOOL	TRUE/FALSE (FALSE)
ErrorID	Indicates the error code if an error occurs.	DFB_ECAT_Diag_ERROR*1	DFB_ECAT_Diag_ERROR (DFB_ECAT_Diag_NO_ERROR)
StationNode	Slave addresses and structure array of slave status.	StationStatus [1..128] *2 *3	StationStatus
uSlaves	Number of Slaves	UINT	0–65535

***Note**:

1. DFB_ECAT_Diag_ERROR: Enumeration (Enum)
2. StationStatus Structure (STRUCT)

Name	Function	Data Type	Setting Value (Default value)
StationAddress	Slave station address	UINT	(0)

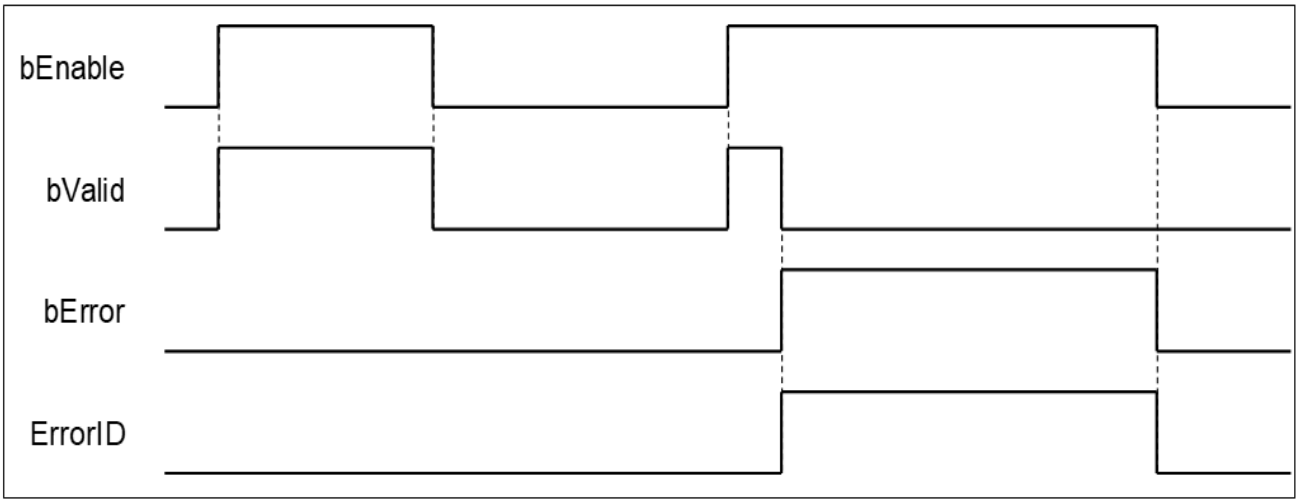
Name	Function	Data Type	Setting Value (Default value)
Node	Connection status of slave stations	BOOL	TRUE: Connected and functioning properly. FALSE: Abnormal connection status. (FALSE)
LinkStatus	Slave physical connection status	BOOL	TRUE: Physical connection status normal FALSE: Physical connection status abnormal (FALSE)

3. The array includes all the slave addresses and connection status, which starts from the first slave station. (Supports up to 128 stations) In addition, if the *StationAddress* value is shown as 0 in the struct array after *bEnable* is triggered by the rising edge, it indicates that the slave station does not exist.

- **Outputs Updating Time**

Name	Timing for shifting to TRUE	Timing for shifting to FALSE
bValid	When *bEnable* shifts to TRUE	• When *bEnable* shifts to FALSE • When *bError* shifts to TRUE
bError	When an error occurs in the running conditions of the instruction	When *bEnable* shifts from TRUE to FALSE (Error code is cleared)
ErrorID		
StationNode	When *bEnable* shifts to TRUE	When *bEnable* shifts to FALSE
uSlaves	When *bEnable* shifts to TRUE	When *bEnable* shifts to FALSE

- **Timing Diagram**



- **Function**

When *bEnable* shifts to TRUE, *StationAddress* and *Node* output from *StationAddress* are in array type to show all the slave addresses and status with the support up to 128 slave stations. If the value of *StationAddress* is shown as 0 in the struct array after *bEnable* is triggered by the rising edge, it indicates that the slave station does not exist. An error will be reported by the function block if EtherCAT master is not found when *bEnable* shifts to TRUE.

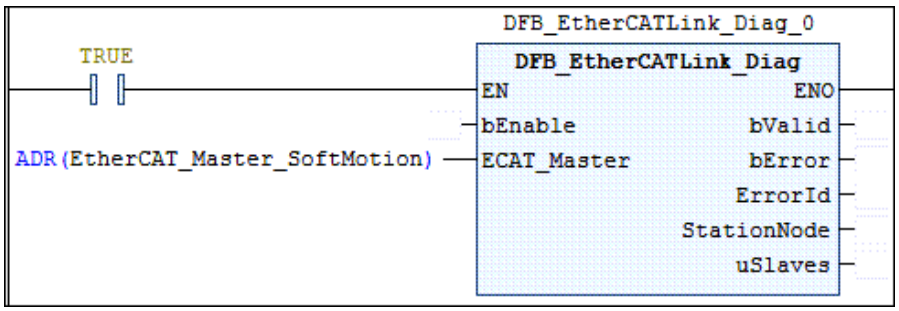
The ECAT_Master parameter applies to DL_EtherCAT_Diag version 1.3.2.0 or later. The name of the master to be used must be specified. If ECAT_Master is not specified, the first EtherCAT master from top to bottom in the project will be used as the master.

- **Troubleshooting**

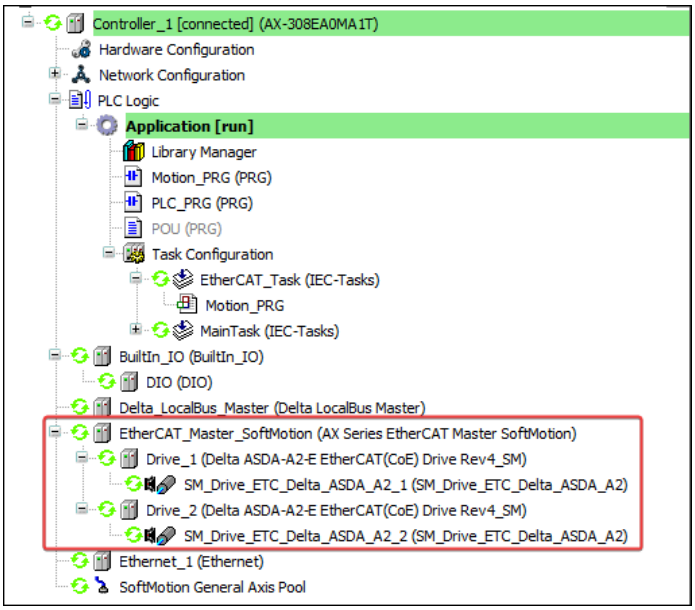
  If an error occurs during the running of the instruction, *bError* will change to TRUE and Capture will stop. You can refer to *ErrorID* (Error Code) to troubleshoot the problem.

- **Programming Example**

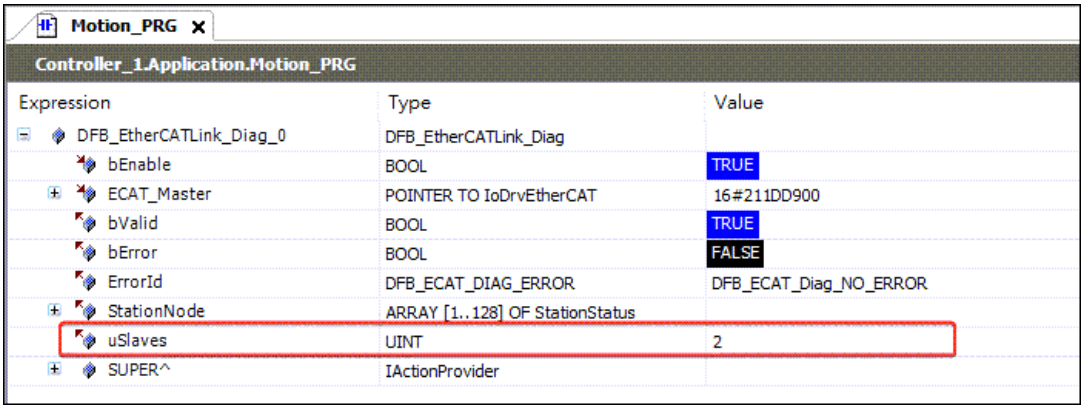
  The following example demonstrates the behavior of DFB_EtherCATLink_Diag.

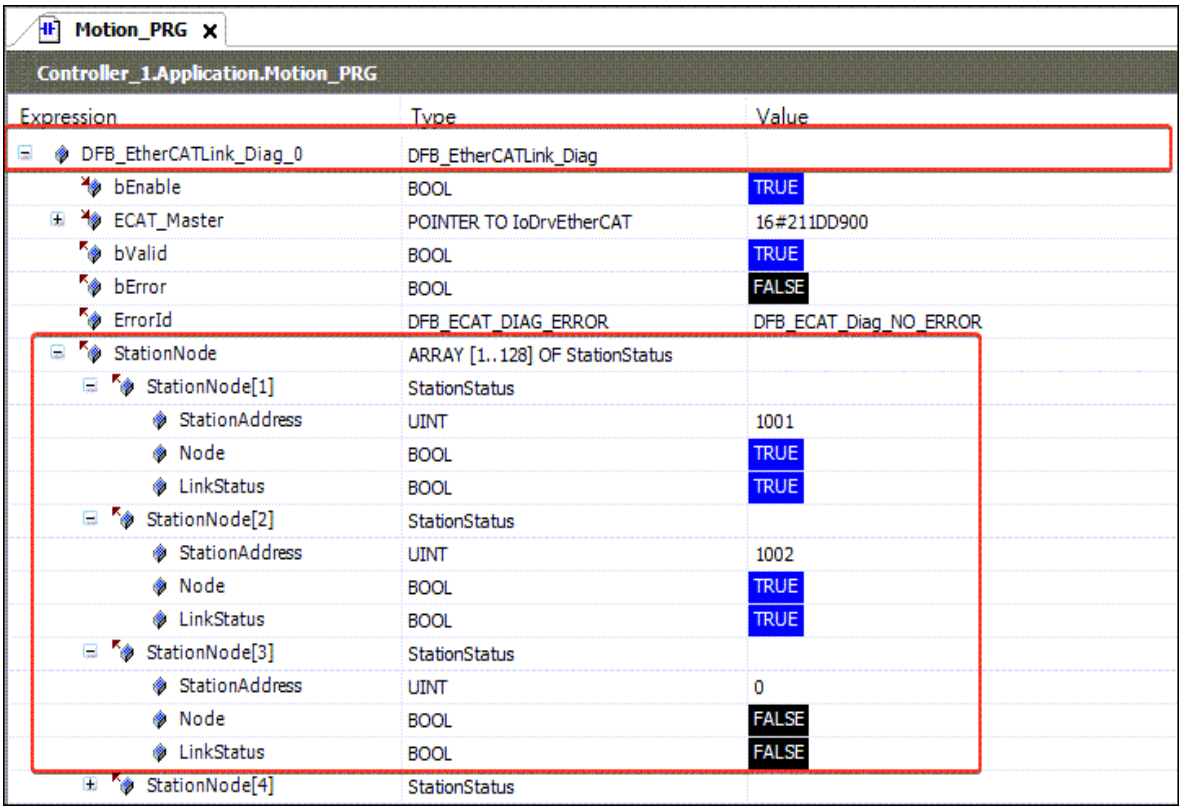


1. There's a total of two EtherCAT slave stations in the Device tree and the connection status shows PASS.

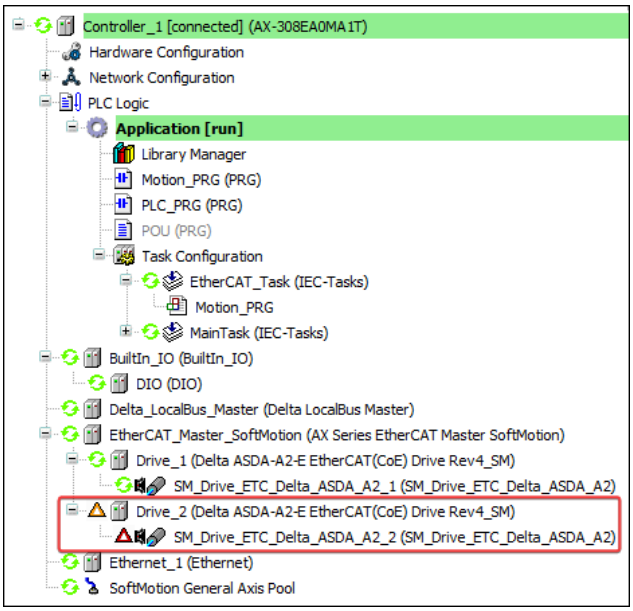


2. After *bEnable* of DFB_EtherCATLink_Diag shifts to TRUE, the output *uSlave* shows 2, indicating two EtherCAT slaves in total. The array of *StationNode* index 1 to 2 shows the two slave addresses, statuses, and physical connection statuses, and the *StationAddress* of *StationNode* shows 0 from index 3.

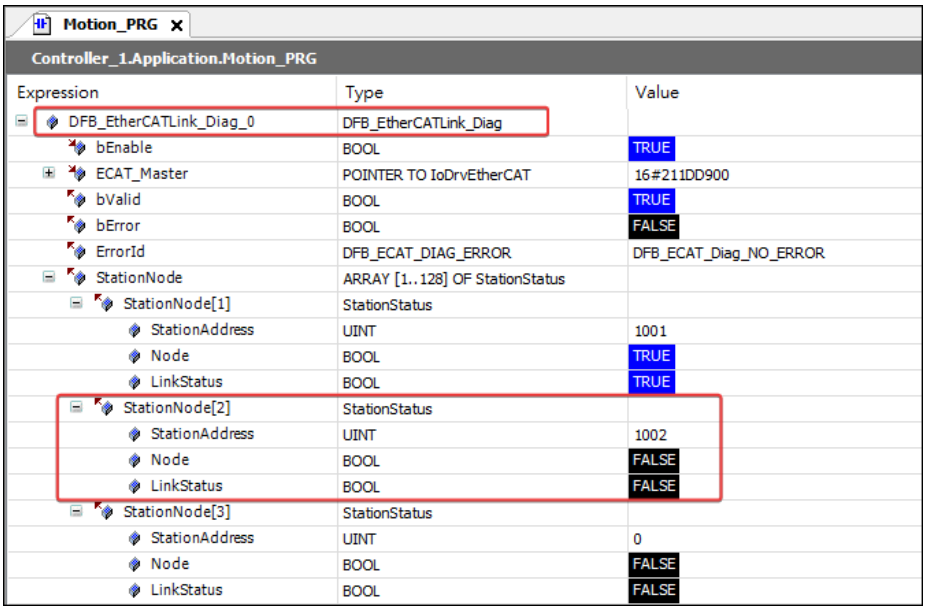




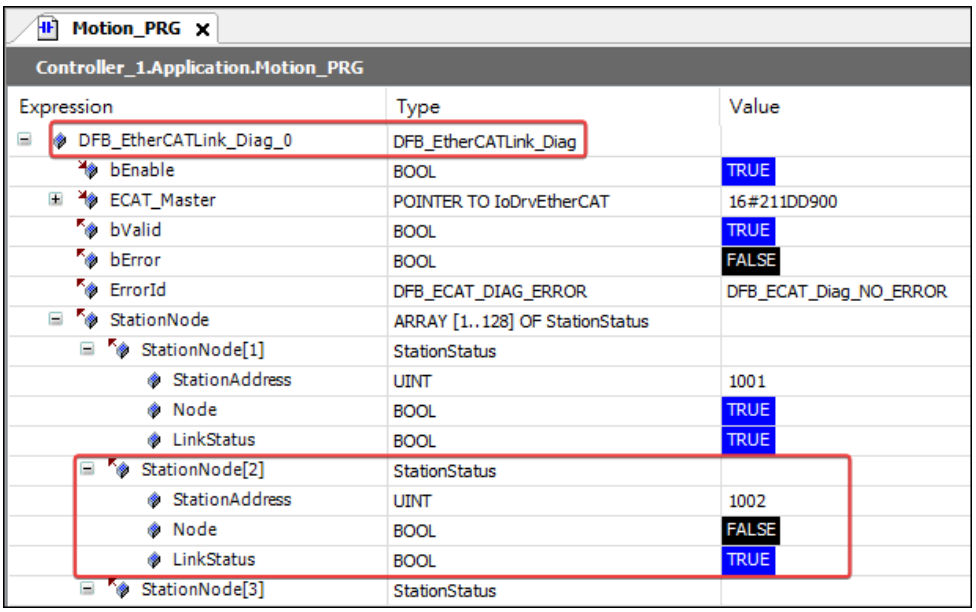
3. Disconnect the cable for internet connection between slave stations 1 and 2, and you can see the status of slave 2 is shown to be Fail in the device tree.



4. Now, the *StationNode* index is 2. *Node* and *LinkStatus* of DFB_EtherCATLink_Diag are FALSE.



5. If you connect the physical network of Slave 1 and Slave 2, *StationNode* index 2 *Node* value will be FALSE and *LinkStatus* will be TRUE.



- **Supported Devices**
  - AX-3 series motion controller, AX-8, AX-C
- **Library**
  - DL_EtherCAT_Diag.library

4.4. DFB_GetAllECATSlaveInfo

DFB_GetAllECATSlaveInfo gets all the slaves' information.

FB/FC	Instruction	Graphic Expression	ST Language
FB	DFB_GetAllECATSlaveInfo		DFB_GetAllECATSlaveInfo (bExecute :=, ECAT_Master :=, bDone =>, bBusy =>, bError =>, ErrorID =>, slaveInfoArray =>, uSlaves =>,);

- **Inputs**

Name	Function	Data Type	Setting Value (Default value)	Timing for Updating
bExecute	Run the instruction when *bExecute* changes to TRUE.	BOOL	TRUE/FALSE (FALSE)	–
ECAT_Master	Name of the master to be used	POINTER TO IoDrvEtherCAT	(0)	When *bExecute* shifts to TRUE and *bBusy* is FALSE.

- **Outputs**

Name	Function	Data Type	Output Range (Default value)
bDone	The running of FB is completed.	BOOL	TRUE/FALSE (FALSE)
bBusy	TRUE when the instruction is enabled	BOOL	TRUE/FALSE (FALSE)
bError	TRUE if an error occurs	BOOL	TRUE/FALSE (FALSE)
ErrorID	Indicates the error code if an error occurs.	DFB_ECAT_Diag_ERROR*[1]	DFB_ECAT_Diag_ERROR (DFB_ECAT_Diag_NO_ERROR)
slaveInfoArray	Slave information array.	ECATSlaveInfo [1..128] *[2]	ECATSlaveInfo
uSlaves	The number of slaves	UINT	0–128 (0)

***Note**:

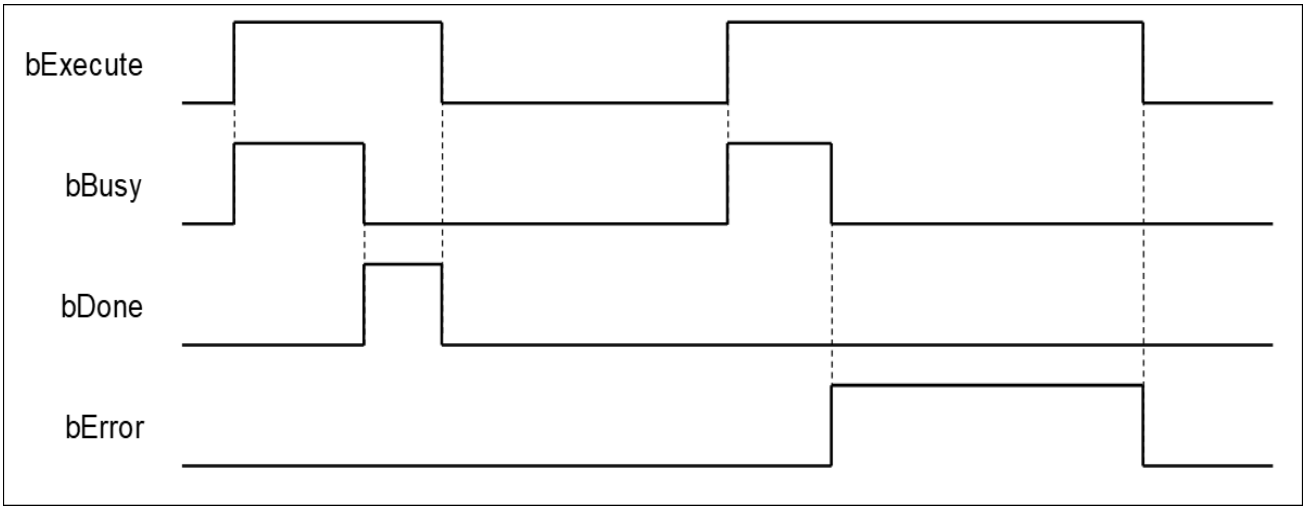
1. DFB_ECAT_Diag_ERROR: Enumeration (Enum)
2. *slaveInfoArray*: Structure (STRUCT)

Name	Function	Data Type	Output Range (Default value)
vendorId	Slave vendor id	UDINT	(0)
productCode	Slave product code	UDINT	(0)
revisionNo	Slave revision number	UDINT	(0)
serialNo	Slave serial number	UDINT	(0)

- **Output Updating Timing**

Name	Timing for shifting to TRUE	Timing for shifting to FALSE
bDone	When the running of FB is completed	• When *bExecute* shifts to FALSE • If *bExecute* is FALSE and *bDone* shifts to TRUE, *bDone* will be TRUE for only one period and immediately shift to FALSE.
bBusy	When *bExecute* shifts to TRUE	• When *bDone* shifts to TRUE • When *bError* shifts to TRUE
bError	When an error occurs in the running conditions of the instruction	When *bExecute* shifts from TRUE to FALSE (Error code is cleared)
ErrorID		
slaveInfoArray	When *bExecute* shifts to TRUE	When *bExecute* shifts from TRUE to FALSE
uSlaves	When *bExecute* shifts to TRUE	When *bExecute* shifts from TRUE to FALSE

- **Timing Diagram**



- **Function**

  When *bExecute* shifts to TRUE, *slaveInfoArray* gives the information of all the EtherCAT slaves in the device tree, which includes vendor id, product code, revision number and serial number. Support up to 128

stations as well as the maximum number of slaves and the corresponding information output from *uSlaves* and *slaveInfoArray*.

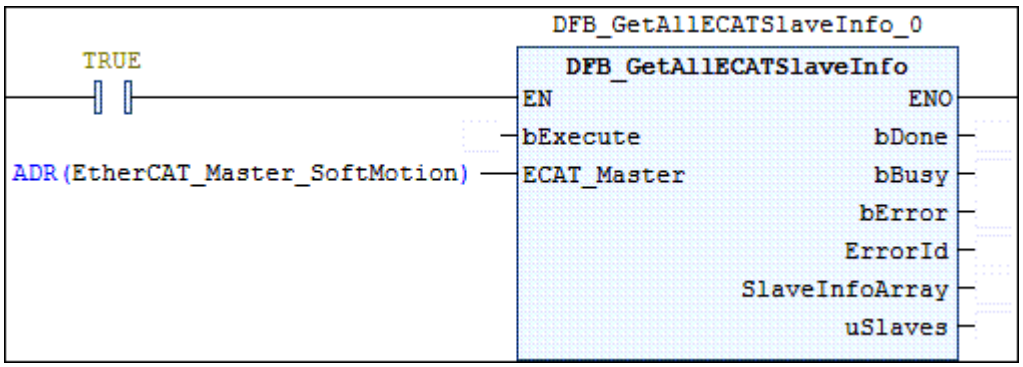
ECAT_Master applies to DL_EtherCAT_Diag version 1.3.2.0 or later. The name of the master to be used must be specified. If ECAT_Master is not specified, the first EtherCAT master from top to bottom in the project will be used as the master.

- **Troubleshooting**

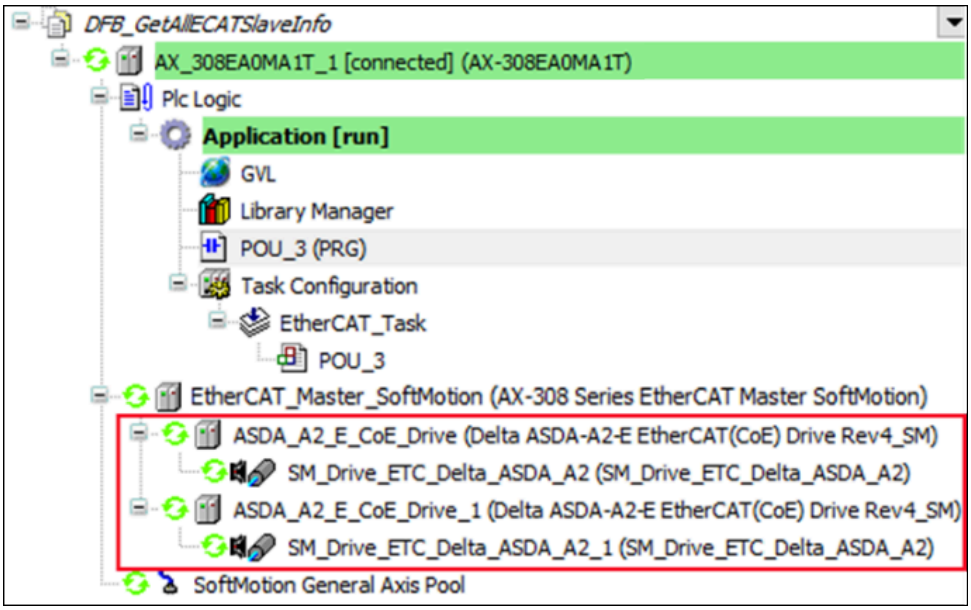
  If an error occurs during the running of the instruction, *bError* will change to TRUE and Capture will stop. You can refer to *ErrorID* (Error Code) to troubleshoot the problem.

- **Programming Example**

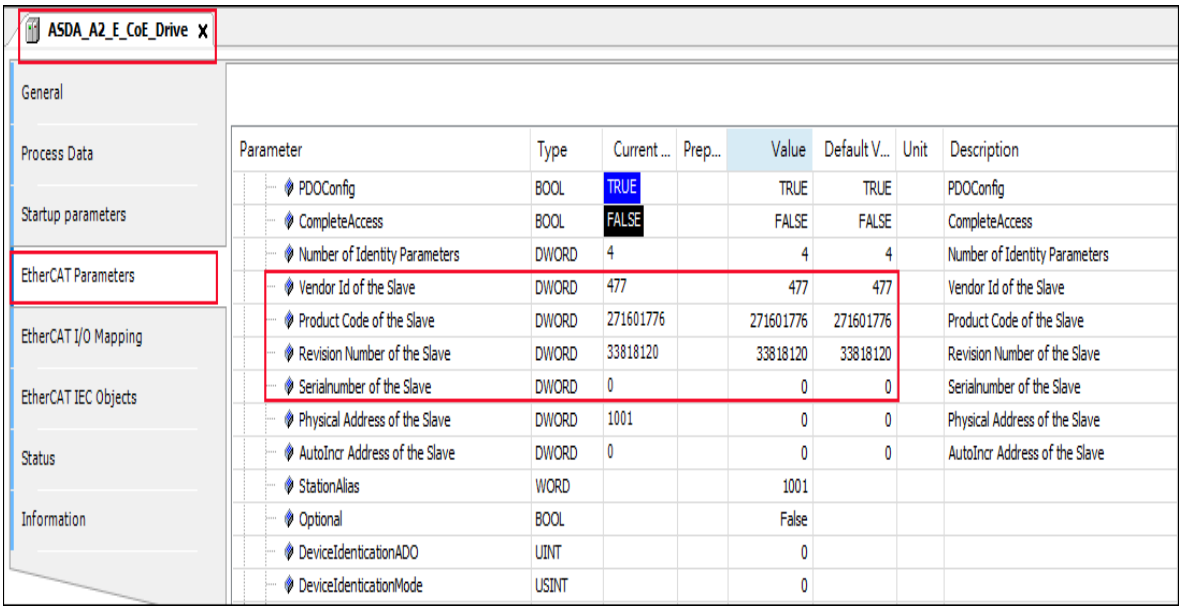
  The following example demonstrates the behavior of DFB_GetAllECATSlaveInfo.



1. There’re two EtherCAT slave stations in the device tree, both are ASDA_A2.



2. Double–click on the target ASDA_A2 in the device tree to view its slave information.

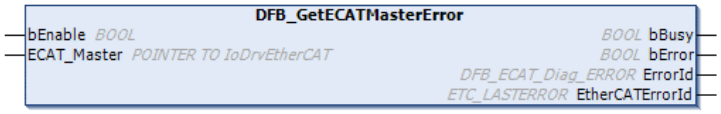


3. The input *bExecute* of DFB_GetAllECATSlaveInfo *bExecute* shifts to TRUE, then the output of *slaveInfoArray* is shown as below and the output value of *uSlaves* is 2.

- **Supported Devices**
  - AX-3 series motion controller, AX-8, AX-C
- **Library**
  - DL_EtherCAT_Diag.library

4.5. DFB_GetECATMasterError

DFB_GetECATMasterError gets the error code of failed EtherCAT network connection.

FB/FC	Instruction	Graphic Expression	ST Language
FB	DFB_GetECATMasterError		DFB_GetECATMasterError (bEnable :=, ECAT_Master :=, bBusy =>, bError =>, ErrorID =>, EtherCATErrorId =>);

- **Inputs**

Name	Function	Data Type	Setting Value (Default value)	Timing for Updating
bEnable	Run the instruction when *bEnable* changes to TRUE.	BOOL	TRUE/FALSE (FALSE)	–
ECAT_Master	Name of the master to be used	POINTER TO IoDrvEtherCAT	(0)	When *bEnable* shifts to TRUE and *bBusy* is FALSE

- **Outputs**

Name	Function	Data Type	Output Range (Default value)
bBusy	TRUE when the instruction is enabled	BOOL	TRUE/FALSE (FALSE)
bError	TRUE if an error occurs.	BOOL	TRUE/FALSE (FALSE)
ErrorID	Indicates the error code if an error occurs.	DFB_ECAT_Diag_ERROR*1	DFB_ECAT_Diag_ERROR (DFB_ECAT_Diag_NO_ERROR)
EtherCATErrorId	EtherCAT error codes	ETC_LASTERROR*2	ETC_LASTERROR (NO_ERROR)

***Note**:

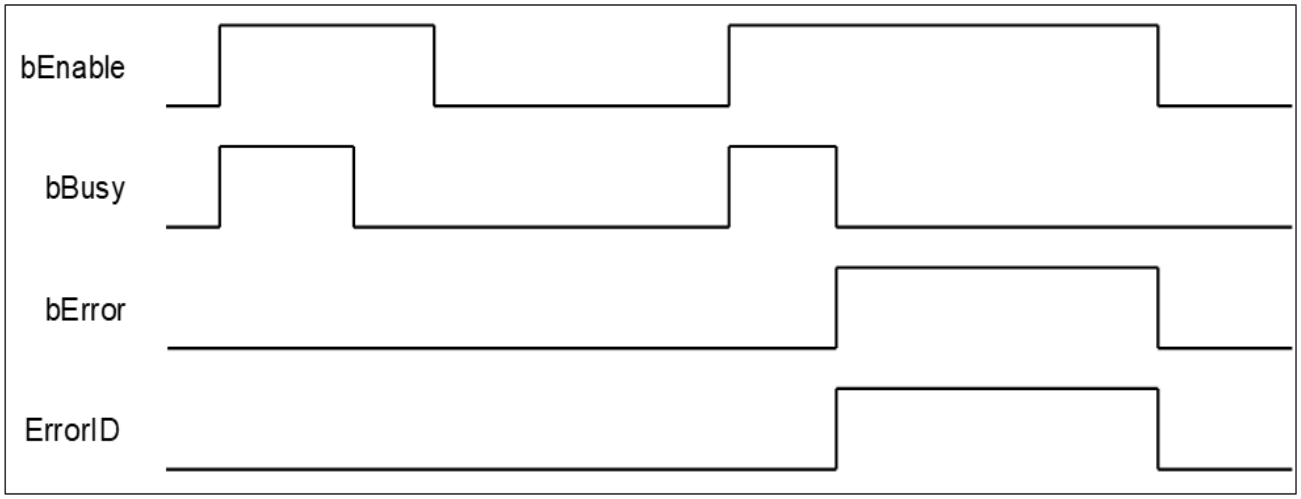
1. DFB_ECAT_Diag_ERROR: Enumeration (Enum)
2. ETC_LASTERROR: Enumeration (Enum)

  - **Outputs Updating Time**

Name	Timing for shifting to TRUE	Timing for shifting to FALSE
bBusy	When *bEnable* shifts to TRUE	• When *bEnable* shifts to FALSE • When *bError* shifts to TRUE

Name	Timing for shifting to TRUE	Timing for shifting to FALSE
bError	When an error occurs in the running conditions of the instruction	When *bEnable* shifts from TRUE to FALSE (Error code is cleared)
ErrorID		
EtherCATErrorId	When an error occurs in the EtherCAT connection	When *bEnable* shifts to FALSE

- **Timing Diagram**



- **Function**

  When *bEnable* shifts to TRUE, the output *EtherCATErrorId* gives the error codes of failed EtherCAT network connection during each cycle. If there's no error, the output will be displayed as NO_ERROR. For more details of error codes, Refer to the content of ETC_LASTERROR_STATE in the Library.

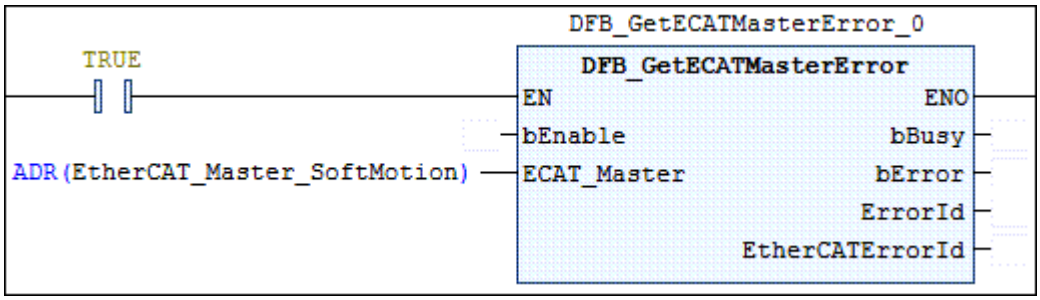
  The ECAT_Master parameter applies to DL_EtherCAT_Diag version 1.3.2.0 or later. The name of the master to be used must be specified. If ECAT_Master is not specified, the first EtherCAT master from top to bottom in the project will be used as the master.

- **Troubleshooting**

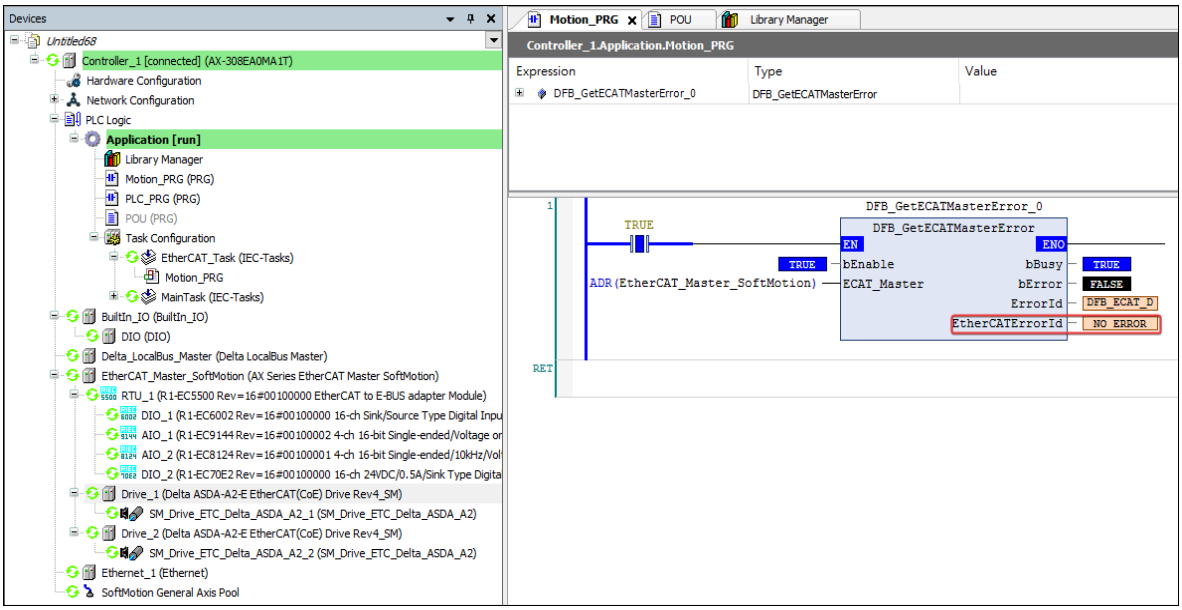
  If an error occurs during the running of the instruction, *bError* will change to TRUE and Capture will stop. You can refer to *ErrorID* (Error Code) to address the problem.

- **Programming Example**

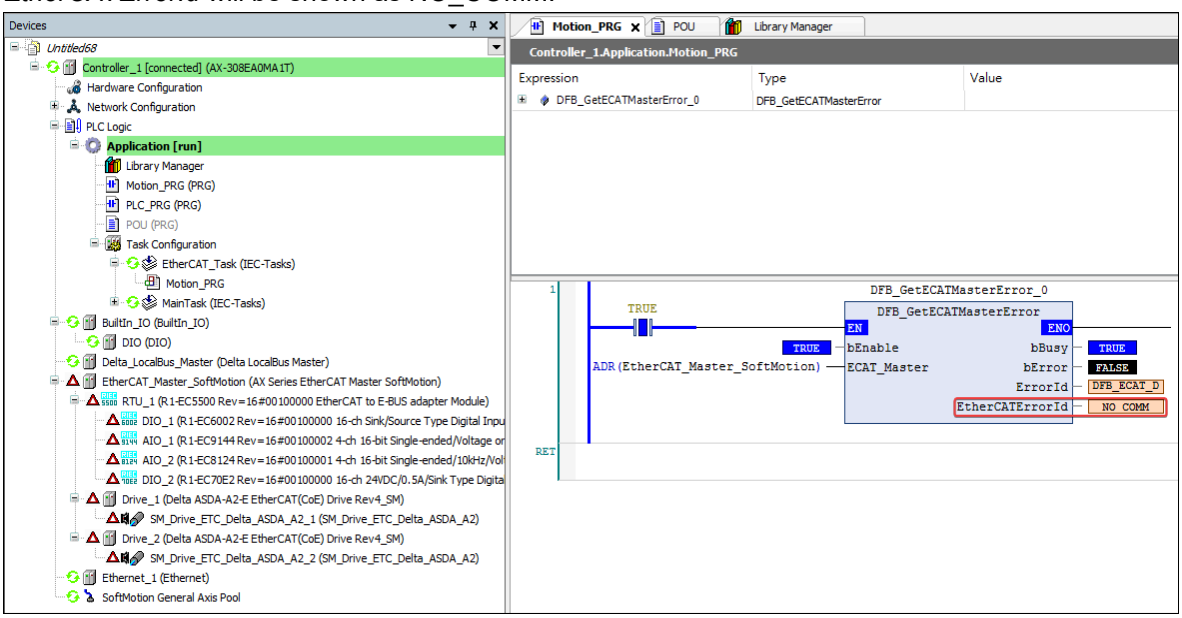
  The following example demonstrates the behavior of DFB_GetECATMasterError.



1. If the EtherCAT connection is normal without any existing errors, the output content of *EtherCATErrorId* will be shown as NO_ERROR after the input *bEnable* of DFB_GetECATMasterError shifts to TRUE.



2. Remove the network connection between the master and the slave, then the output content of *EtherCATErrorId* will be shown as NO_COMM.



- **Supported Devices**
  - AX-3 series motion controller, AX-8, AX-C
- **Library**
  - DL_EtherCAT_Diag.library

4.6. DFB_GetECATMasterState

DFB_GetECATMasterState gets the connection status of EtherCAT Master.

FB/FC	Instruction	Graphic Expression	ST Language
FB	DFB_GetECAT MasterState		DFB_GetECATMasterState (bEnable :=, ECAT_Master :=, bBusy =>, bError =>, ErrorID =>, bStatus =>,);

- **Inputs**

Name	Function	Data Type	Setting Value (Default value)	Timing for Updating
bEnable	Run the instruction when *bEnable* changes to TRUE.	BOOL	TRUE/FALSE (FALSE)	–
ECAT_Master	Name of the master to be used	POINTER TO IoDrvEtherCAT	(0)	When *bEnable* shifts to TRUE and *bBusy* is FALSE

- **Outputs**

Name	Function	Data Type	Output Range (Default value)
bBusy	TRUE when the instruction is enabled	BOOL	TRUE/FALSE (FALSE)
bError	TRUE if an error occurs	BOOL	TRUE/FALSE (FALSE)
ErrorID	Indicates the error code if an error occurs.	DFB_ECAT_Diag_ERROR*	DFB_ECAT_Diag_ERROR (DFB_ECAT_Diag_NO_ERROR)
bStatus	EtherCAT master communication status	BOOL	TRUE/FALSE (FALSE)

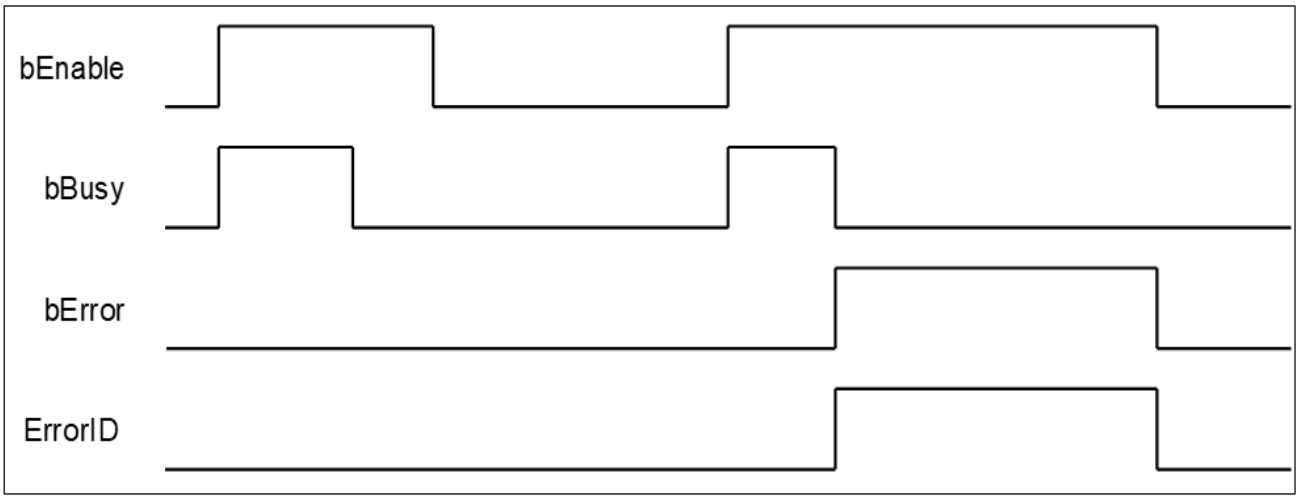
***Note**: DFB_ECAT_Diag_ERROR: Enumeration (Enum)

- **Output Updating Timing**

Name	Timing for shifting to TRUE	Timing for shifting to FALSE
bBusy	When *bEnable* shifts to TRUE	When *bError* shifts to TRUE
bError	When an error occurs in the running conditions of the instruction	When *bEnable* shifts from TRUE to FALSE (Error code is cleared)
ErrorID		

Name	Timing for shifting to TRUE	Timing for shifting to FALSE
bStatus	When the EtherCAT master connection is normal	• When *bEnable* shifts to FALSE • When *bError* shifts to TRUE • When the connection is abnormal

- **Timing Diagram**



- **Function**

  When *bEnable* shifts to TRUE, the function block performs cyclical status updates of EtherCAT master communication.

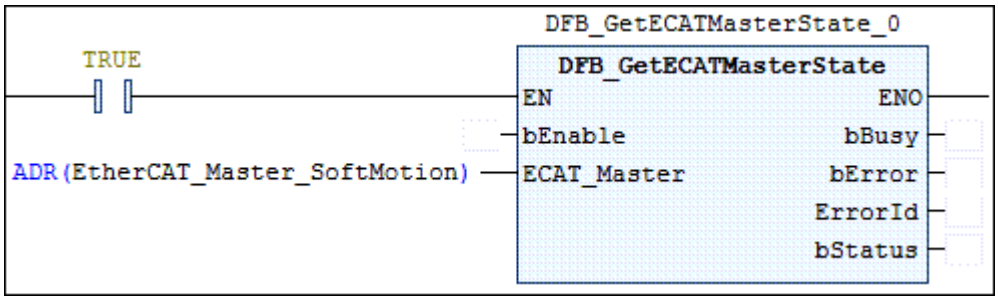
  The ECAT_Master parameter applies to DL_EtherCAT_Diag version 1.3.2.0 or later. The name of the master to be used must be specified. If ECAT_Master is not specified, the first EtherCAT master from top to bottom in the project will be used as the master.

- **Troubleshooting**

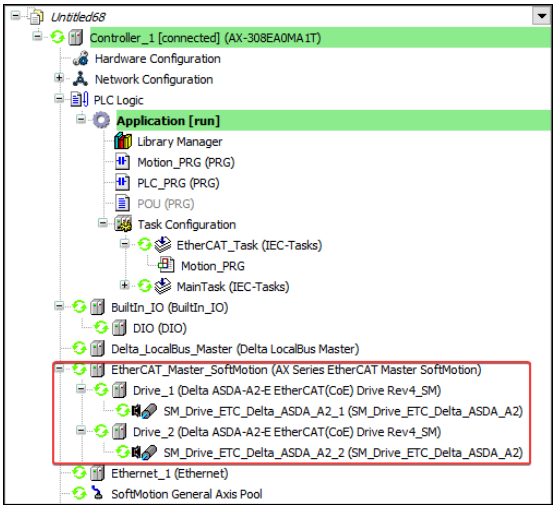
  If an error occurs during the running of the instruction, *bError* will change to TRUE and Capture will stop. You can refer to *ErrorID* (Error Code) to troubleshoot the problem.

- **Programming Example**

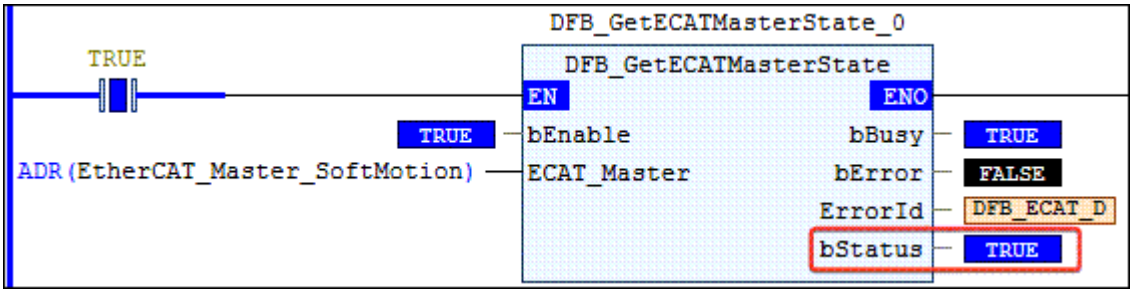
  The following example demonstrates the behavior of DFB_GetECATMasterState.



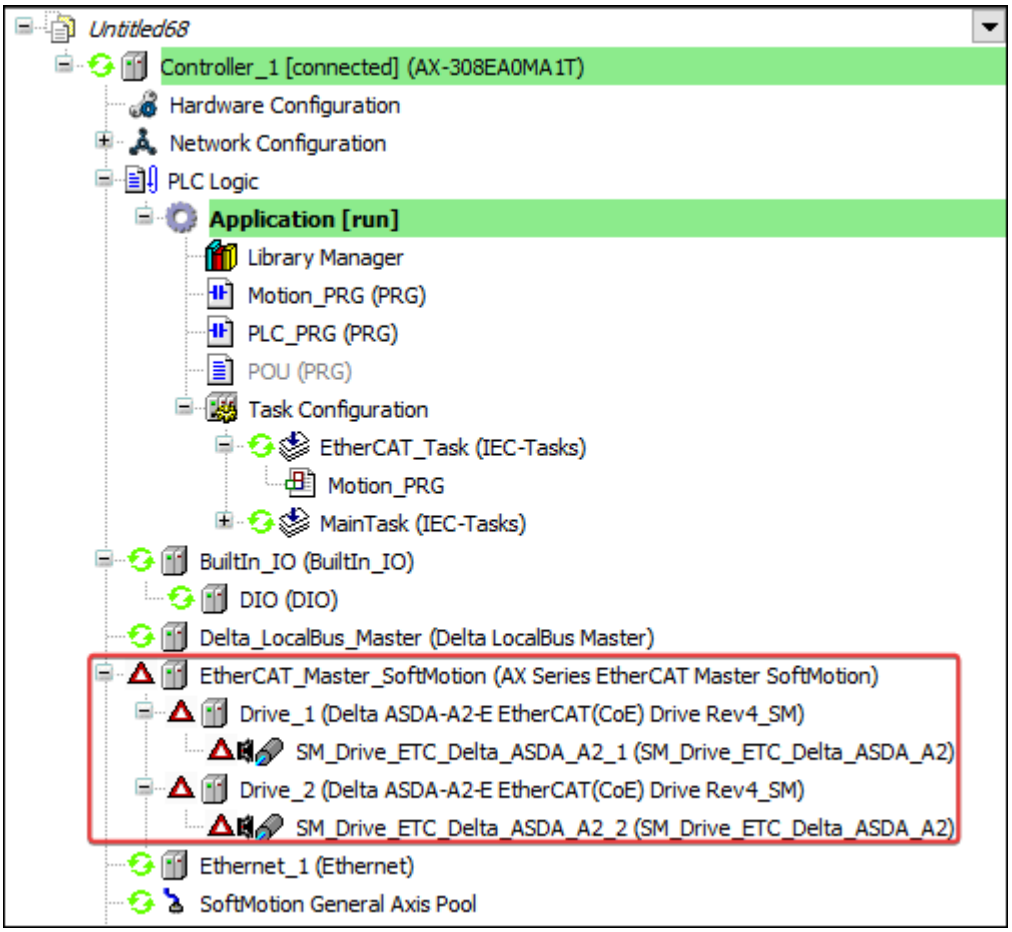
1. The connection status of EtherCAT master shows PASS in the device tree.



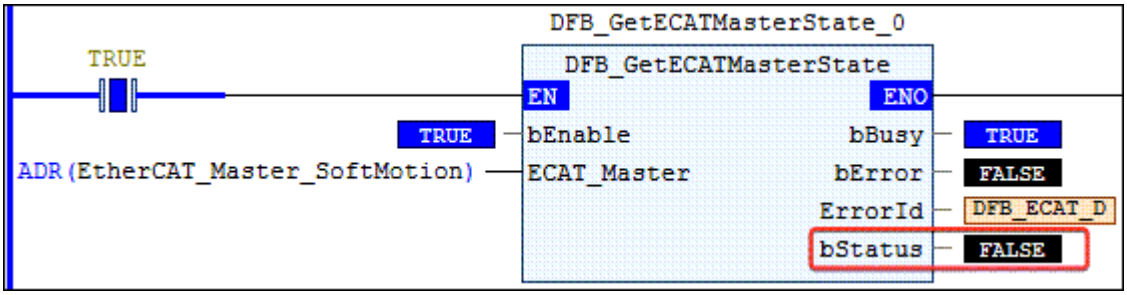
2. When the input *bEnable* of DFB_GetECATMasterState shifts to TRUE, the output of *bStatus* is displayed as TRUE.



3. Remove the network connection between the master and the slave, and the current connection status of EtherCAT master will show Fail in the device tree.



4. The output *bStatus* of DFB_GetECATMasterState is displayed as FALSE.



- **Supported Devices**
  - AX-3 series motion controller, AX-8, AX-C
- **Library**
  - DL_EtherCAT_Diag.library

4.7. DFB_ResetECATMaster

DFB_ResetECATMaster resets the EtherCAT master with connection errors.

FB/FC	Instruction	Graphic Expression	ST Language
FB	DFB_ResetECATMaster		DFB_ResetECATMaster (bExecute :=, ECAT_Master :=, bDone =>, bBusy =>, bError =>, ErrorID =>,);

- **Inputs**

Name	Function	Data Type	Setting Value (Default value)	Timing for Updating
bExecute	Run the instruction when *bExecute* changes to TRUE.	BOOL	TRUE/FALSE (FALSE)	–
ECAT_Master	Name of the master to be used	POINTER TO IoDrvEtherCAT	(0)	When *bExecute* shifts to TRUE and *bBusy* is FALSE

- **Outputs**

Name	Function	Data Type	Output Range (Default value)
bDone	The running of FB is completed.	BOOL	TRUE/FALSE (FALSE)
bBusy	TRUE when the instruction is enabled	BOOL	TRUE/FALSE (FALSE)
bError	TRUE if an error occurs	BOOL	TRUE/FALSE (FALSE)
ErrorID	Indicates the error code if an error occurs.	DFB_ECAT_Diag_ERROR*	DFB_ECAT_Diag_ERROR (DFB_ECAT_Diag_NO_ERROR)

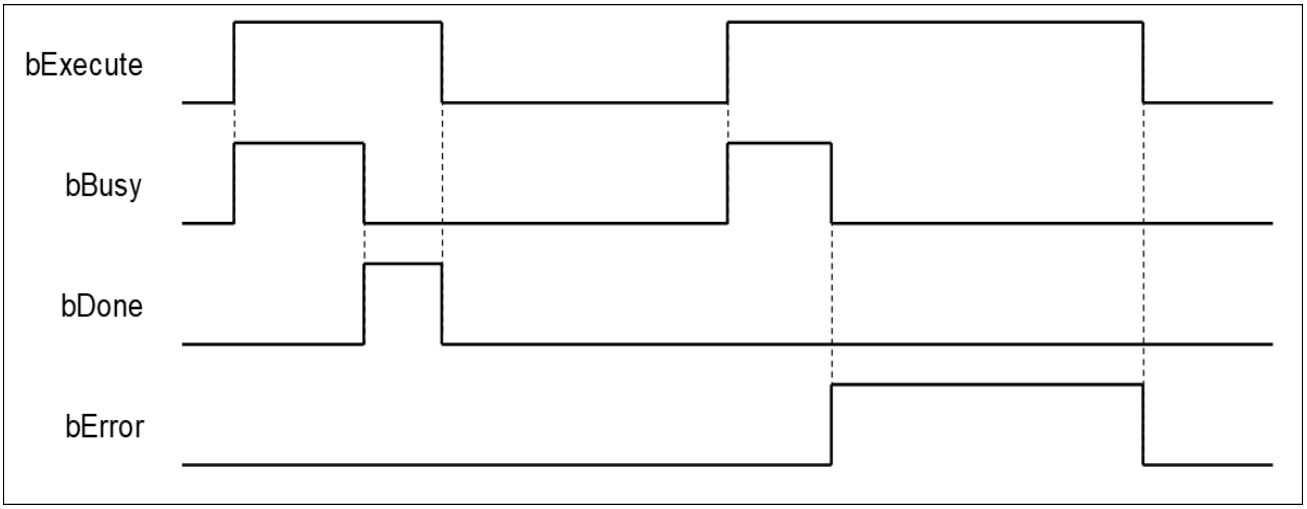
***Note**: DFB_ECAT_Diag_ERROR: Enumeration (Enum)

- **Output Updating Timing**

Name	Timing for shifting to TRUE	Timing for shifting to FALSE
bDone	When the running of FB is completed.	• When *bExecute* shifts to FALSE • If *bExecute* is FALSE and *bDone* shifts to TRUE, *bDone* will be TRUE for only one period and immediately shift to FALSE.

Name	Timing for shifting to TRUE	Timing for shifting to FALSE
bBusy	When *bExecute* shifts to TRUE	• When *bDone* shifts to TRUE • When *bError* shifts to TRUE
bError	When an error occurs in the running conditions of the instruction	When *bExecute* shifts from TRUE to FALSE (Error code is cleared)
ErrorID		

- **Timing Diagram**



- **Function**

  When *bExecute* shifts to TRUE and the connection status of EtherCAT master shows Fail, the function block will perform reset action.

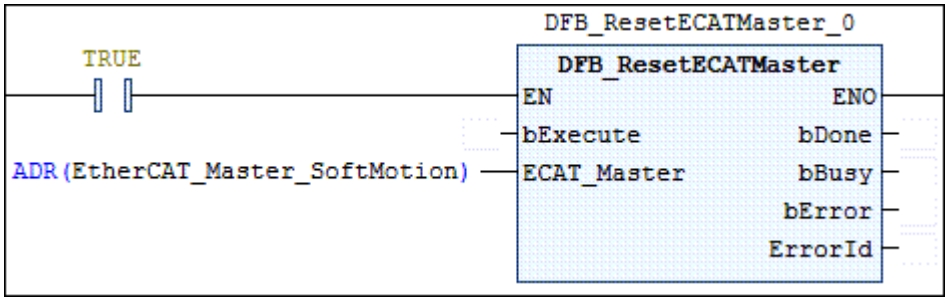
  The ECAT_Master parameter applies to DL_EtherCAT_Diag version 1.3.2.0 or later. The name of the master to be used must be specified. If ECAT_Master is not specified, the first EtherCAT master from top to bottom in the project will be used as the master.

- **Troubleshooting**

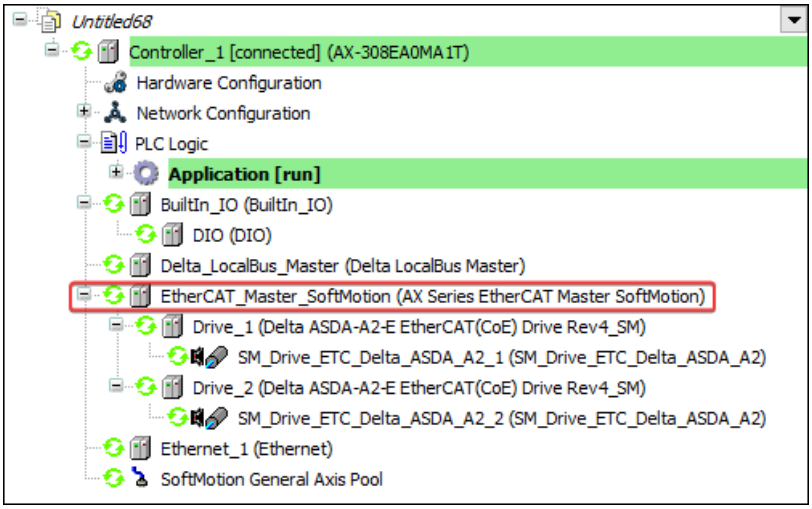
  If an error occurs during the running of the instruction, *bError* will change to TRUE and Capture will stop. You can refer to *ErrorID* (Error Code) to address the problem.

- **Programming Example**

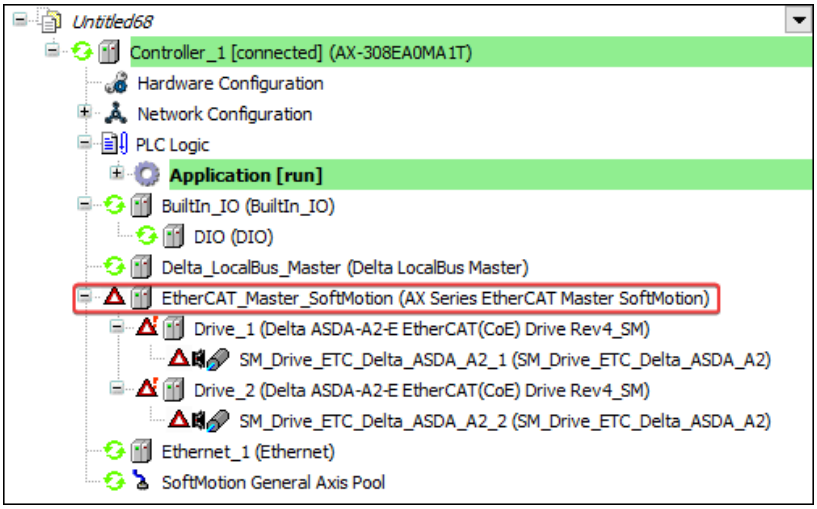
  The following example demonstrates the behavior of DFB_ResetECATMaster.



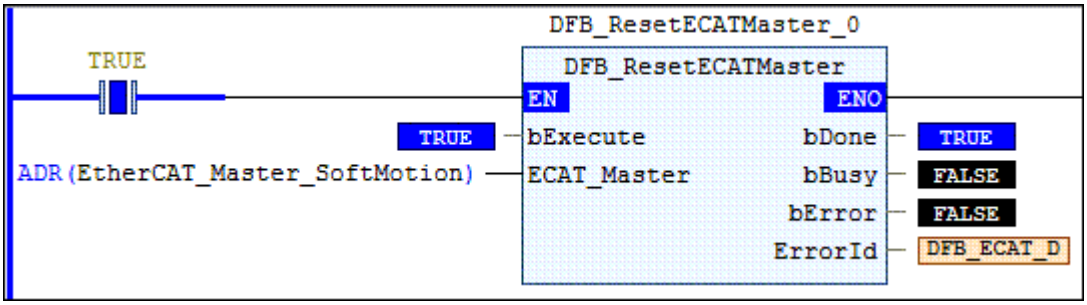
1. The connection status of EtherCAT master shows PASS in the device tree.



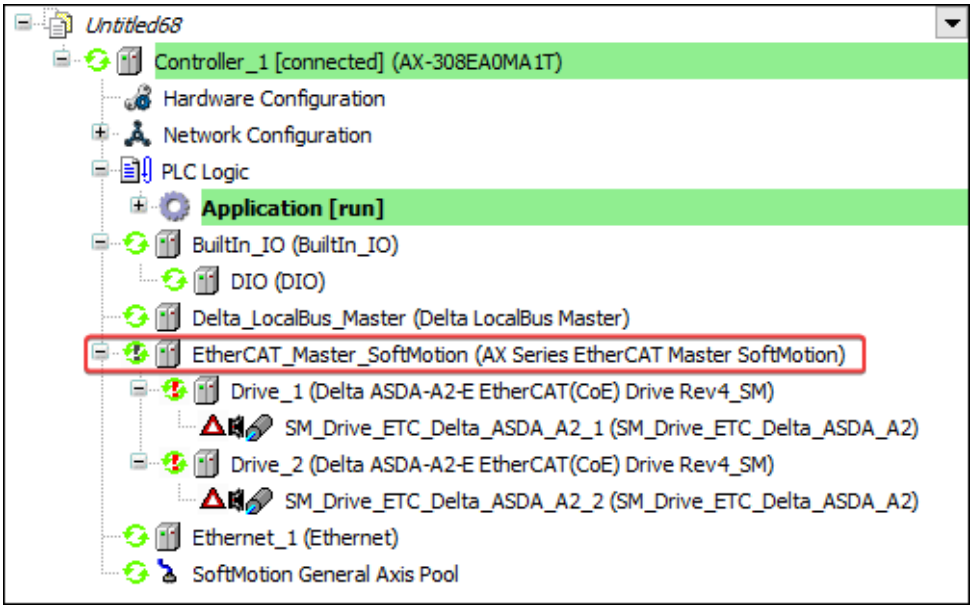
2. Remove the network connection between the master and the slave, and the current connection status of EtherCAT master will show Fail in the device tree.



3. To restore the network connection between the master and the slave, shift the input *bExecute* of DFB_ResetECATMaster to TRUE.



4. The network connectivity has been recovered after the output bDone shifting to TRUE.



- **Supported Devices**
  - AX-3 series motion controller, AX-8, AX-C
- **Library**
  - DL_EtherCAT_Diag.library

4.8. DFB_ResetECATSlave

DFB_ResetECATSlave resets the EtherCAT slave with connection errors.

FB/FC	Instruction	Graphic Expression	ST Language
FB	DFB_ResetECATSlave		DFB_ResetECATSlave(bExecute :=, uiSlaveAddr :=, tTimeout :=, ECAT_Master :=, bDone =>, bBusy =>, bError =>, ErrorID =>,);

- **Inputs**

Name	Function	Data Type	Setting Value (Default value)	Timing for Updating
bExecute	Run the instruction when *bExecute* changes to TRUE.	BOOL	TRUE/FALSE (FALSE)	–
uiSlaveAddr	Reset the slave address.	UINT	Positive number (0)	When *bExecute* is triggered by the rising edge and *bBusy* is FALSE
tTimeout	Slave resets the time–out.	TIME	Positive number (0)	When *bExecute* is triggered by the rising edge and *bBusy* is FALSE
ECAT_Master	Name of the master to be used	POINTER TO IoDrvEtherCAT	(0)	When *bExecute* shifts to TRUE and *bBusy* is FALSE

- **Outputs**

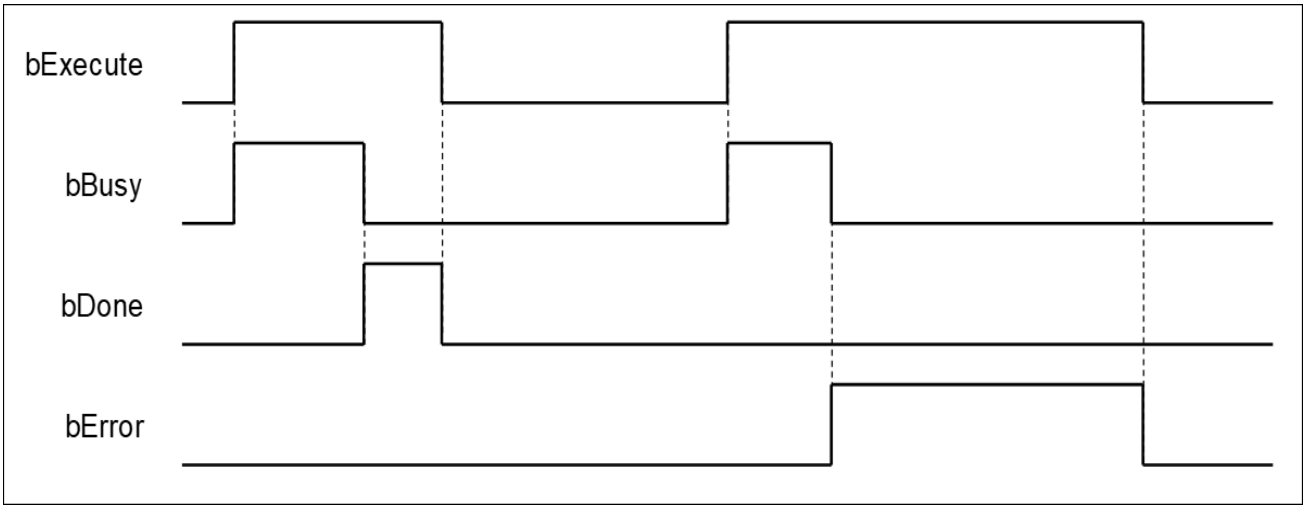
Name	Function	Data Type	Output Range (Default value)
bDone	The running of FB is completed.	BOOL	TRUE/FALSE (FALSE)
bBusy	TRUE when the instruction is enabled	BOOL	TRUE/FALSE (FALSE)
bError	TRUE if an error occurs	BOOL	TRUE/FALSE (FALSE)
ErrorID	Indicates the error code if an error occurs.	DFB_ECAT_Diag_ERROR*	DFB_ECAT_Diag_ERROR (DFB_ECAT_Diag_NO_ERROR)

***Note**: DFB_ECAT_Diag_ERROR: Enumeration (Enum)

- **Output Updating Timing**

Name	Timing for shifting to TRUE	Timing for shifting to FALSE
bDone	When the running of FB is completed.	• When *bExecute* shifts to FALSE • If *bExecute* is FALSE and *bDone* shifts to TRUE, *bDone* will be TRUE for only one period and immediately shift to FALSE.
bBusy	When *bExecute* is shifts to TRUE	• When *bDone* shifts to TRUE • When *bError* shifts to TRUE
bError	When an error occurs in the running conditions of the instruction	When *bExecute* shifts from TRUE to FALSE (Error code is cleared)
ErrorID		

- **Timing Diagram**



- **Function**

  When *bExecute* shifts to TRUE, the function block starts searching for the target slave station and resets the EtherCAT slave, if the status of target slave shows Fail. If the input value of *uiSlaveAddr* is 0, the function block will reset all the slave stations which have errors in connection.

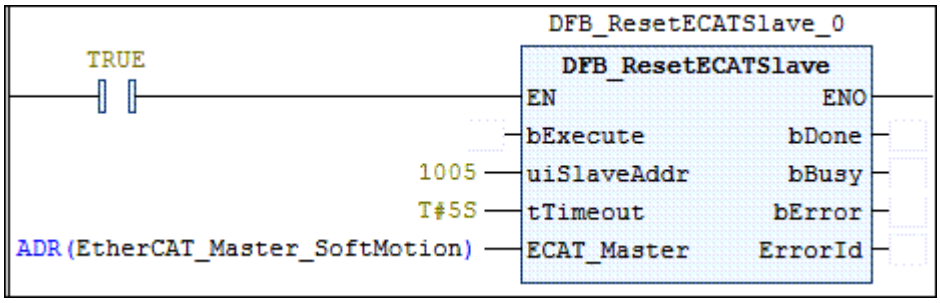
  The ECAT_Master parameter applies to DL_EtherCAT_Diag version 1.3.2.0 or later. The name of the master to be used must be specified. If ECAT_Master is not specified, the first EtherCAT master from top to bottom in the project will be used as the master.

- **Troubleshooting**

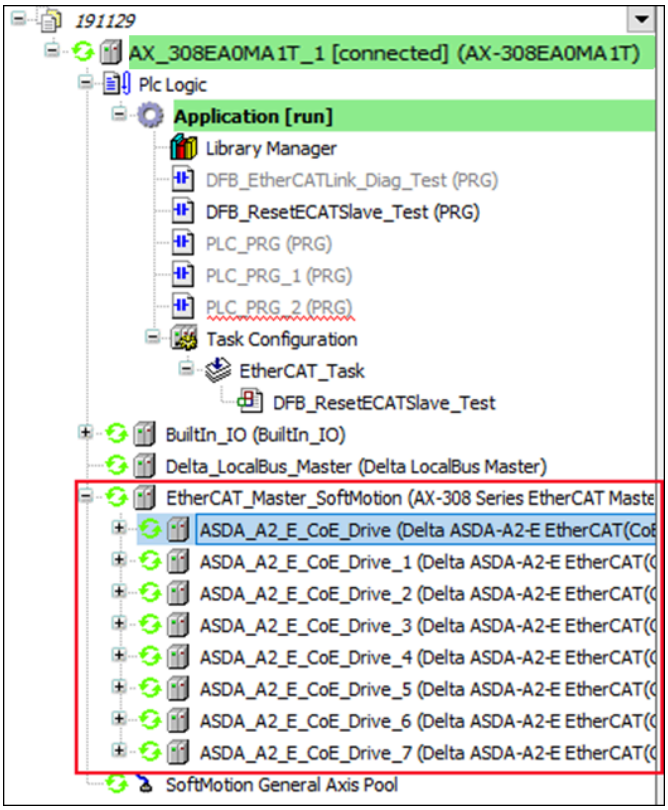
  If an error occurs during the running of the instruction, *bError* will change to TRUE and Capture will stop. You can refer to *ErrorID* (Error Code) to troubleshoot the problem.

- **Programming Example**

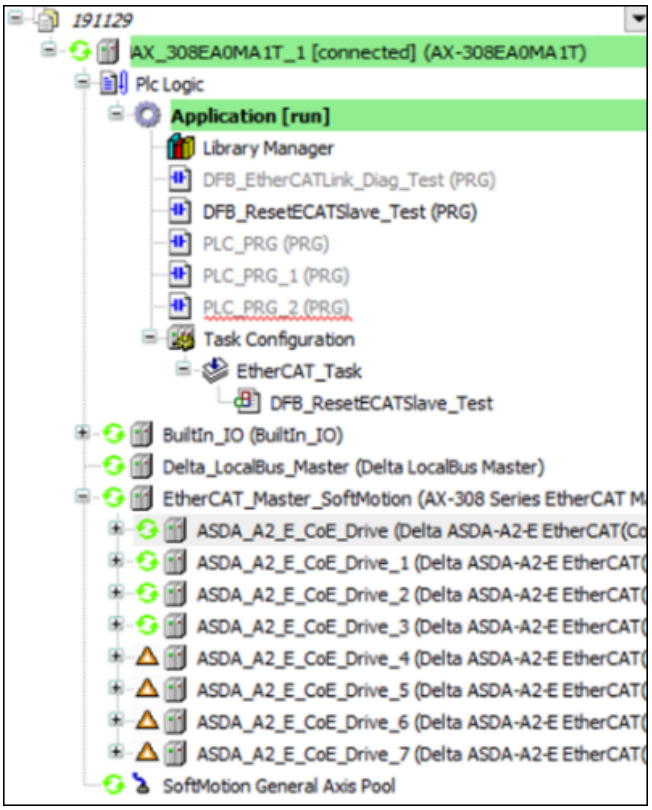
  The following example demonstrates the behavior of DFB_ResetECATSlave.



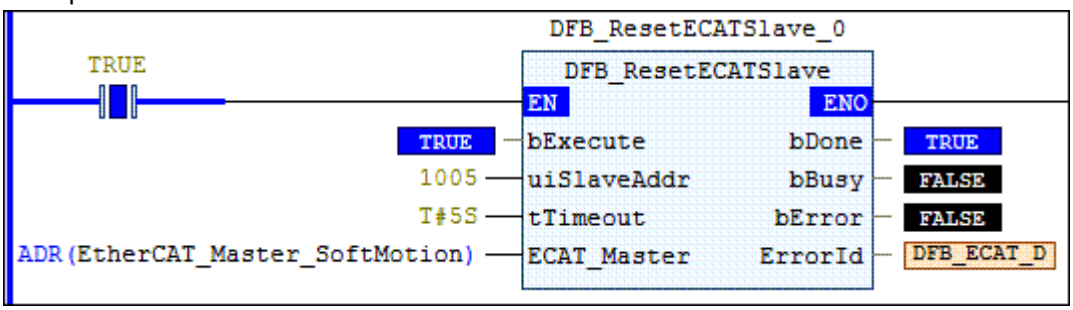
1. There's a total of 8 EtherCAT slave stations in EtherCAT_Master_SoftMotion and all their connection status shows PASS.



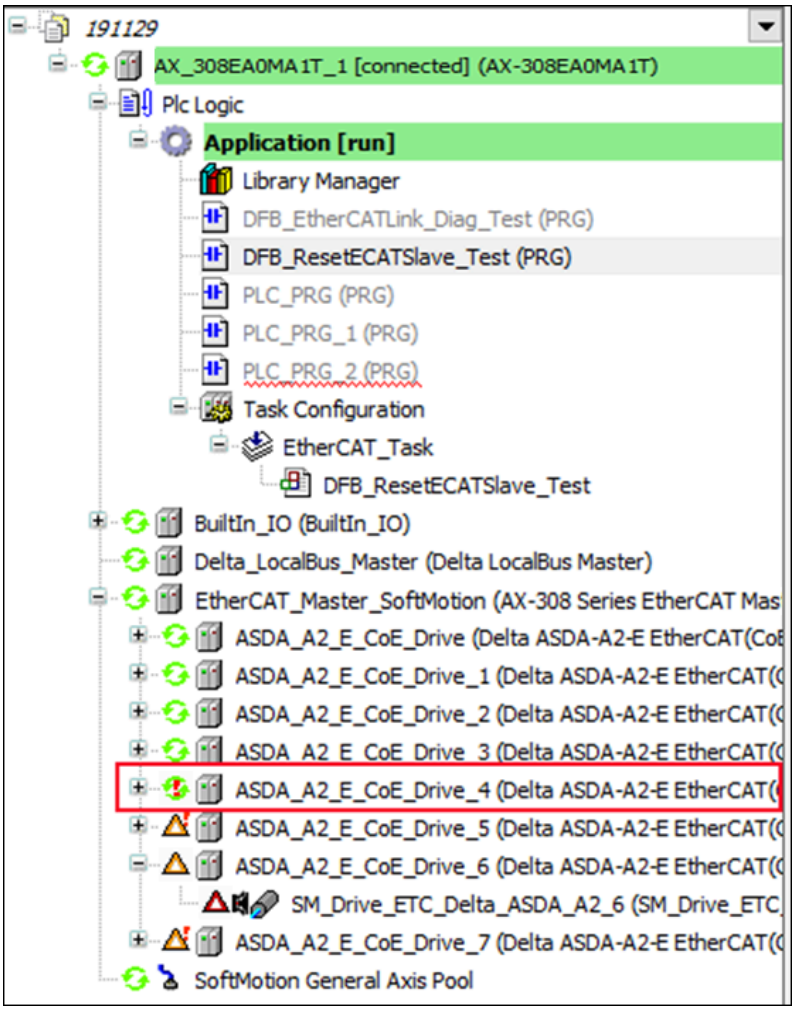
2. Remove the network connection of slave 4 and 5, and the current connection status of EtherCAT slave starting from slave 5 will show Fail in the device tree.



3. To restore the network connection of slave 4 and 5, enter 1005 to the input *uiSlaveAddr* and shift the input *bExecute* to TRUE.



4. The network connectivity of slave 5 has been recovered after the output bDone of the FB shifting to TRUE.



5. All the slave stations will be reset if you enter 0 to the input *uiSlaveAddr*.

- **Supported Devices**
  - AX-3 series motion controller, AX-8, AX-C
- **Library**
  - DL_EtherCAT_Diag.library

4.9. Error Codes and Troubleshooting

The following table lists the error codes corresponding to the FBs and the contents of the errors:

Description	Cause of Error	Corrective Action
DFB_ECAT_Diag_MASTER_CANT_BE_FOUND	EtherCAT master cannot be found.	The EtherCAT master is not in the mapping. Make sure the configuration of the EtherCAT master is correct.
DFB_ECAT_Diag_MASTER_ERROR	Diagnostics of EtherCAT master state is wrong.	Troubleshoot the errors in EtherCAT master before running the FB.
DFB_ECAT_Diag_SLAVE_CANT_BE_FOUND	EtherCAT slave cannot be found.	The EtherCAT master is not in the mapping. Make sure the EtherCAT master address is correct.
DFB_ECAT_Diag_MASTER_RESTART_TIME OUT	Time out occurs when restart EtherCAT master.	Check if the time–out is too short or the internet has been lost.
DFB_ECAT_Diag_SLAVE_RESTART_TIMEOUT	Time out occurs when restart EtherCAT slave.	Check if the time–out is too short or the internet has been lost.
DFB_ECAT_Diag_MASTER_DISABLE	EtherCAT master is disabled.	Check whether to turn on the EtherCAT master.

Chapter 5: Checksum Instructions

5.1. DFC_LRC8

DFC_LRC8 calculates the LRC (8–bit) checksum.

FB/FC	Instruction	Graphic Expression	ST Language
FC	DFC_LRC8		DFC_LRC8(pSrc:= , wLen:= , ErrorID=>);

- **Inputs**

Name	Function	Data Type	Setting Value (Default value)
pSrc	The start address for LRC calculation	POINTER TO BYTE	Memory address 1–256 (0)
wLen	The data length for LRC calculation	DWORD*	(0)

***Note 1**: The BYTE and WORD data types can be used for dwLen input.

***Note 2**: The memory address given by the *pSrc* exceeds the usable range, it may cause controller exceptions, such as %I, %Q and %M.

- **Outputs**

Name	Function	Data Type	Output Range (Default value)
DFC_LRC8	LRC checksum (The parameter is returned)	BYTE	(0)
ErrorID	Error codes	DL_LRC_ERROR	DL_LRC_ERROR(DFC_NO_ERROR)

- **Function**

  After running the FC instruction, begins to calculate LRC (8–bit) checksum from the start memory address (*pSrc*). The calculation scope is determined by the input *wLen*.

- **Programming Example**

  The example uses FC instruction (DFC_LRC8) to perform calculating the LRC (8–bit) checksum.

```
PROGRAM PLC_PRG
VAR
   bVar0: BOOL;   byVar0: BYTE;
   ar_byVar0: ARRAY [0..5] OF BYTE := [16#30,16#31,16#30,16#30,16#40,16#30];
END_VAR

IF bVar0 THEN
   byVar0:=DFC_LRC8(pSrc:=ADR(ar_byVar0[0]) , dwLen:=6 , ErrorID=> );
   bVar0:=FALSE;
END_IF;
```

  The checksum calculation scope is 6(dwLen = 6), therefore, the FC instruction(DFC_LRC8) will starts calculating checksums of six consecutive BYTE data from the memory address input to *pSrc*(ar_byVar0[0]) and will result in a checksum value of 16#CF.

  Note: In library version 1.0.0.1, change the input from dwLen to wLen.

- **Supported Devices**
  - AX series
- **Library**
  - DL_LRC.library

5.2. DFC_LRC16

DFC_LRC16 calculates the LRC (16–bit) checksum.

FB/FC	Instruction	Graphic Expression	ST Language
FC	DFC_LRC16		DFC_LRC16(pSrc:= , dwLen:= , ErrorID=>);

- **Inputs**

Name	Function	Data Type	Setting Value (Default value)
pSrc	The start address for LRC calculation	POINTER TO BYTE	Memory address 1–512 (0)
wLen	The data length for LRC calculation	DWORD*	(0)

***Note 1**: The variable type BYTE and WORD can be used for dwLen input.

***Note 2**: The memory address given by the *pSrc* exceeds the usable range, it may cause controller exceptions, such as %I, %Q and %M.

- **Outputs**

Name	Function	Data Type	Output Range (Default value)
DFC_LRC16	LRC checksum (The parameter is returned.)	WORD	(0)
ErrorID	Error codes	DL_LRC_ERROR	DL_LRC_ERROR (DFC_NO_ERROR)

- **Function**

  After running the FC instruction, begins to calculate LRC (16–bit) checksum from the start memory address *pSrc*. The calculation scope is determined by the input *wLen*.

- **Programming Example**

  The example uses FC instruction (DFC_LRC16) to perform calculating the LRC (16–bit) checksum.

```
PROGRAM PLC_PRG
VAR
   bVar0: BOOL;
   wVar0: WORD;
   ar_wVar0: ARRAY [0..5] OF WORD := [16#3031,16#3132,16#3233,16#3334,16#3435,16#3536];
END_VAR

IF bVar0 THEN
   wVar0:=DFC_LRC16(pSrc:=ADR(ar_wVar0[0]) , dwLen:=6 , ErrorID=> );
   bVar0:=FALSE;
END_IF;
```

  The checksum calculation scope is 6(dwLen = 6), therefore, the FC instruction(DFC_LRC16) will starts calculating checksums of six consecutive BYTE data from the memory address input to *pSrc*(ar_byVar0[0]) and will result in a checksum value of 16#CFCB.

  Note: In library version 1.0.0.1, change the input from dwLen to *wLen*.

- **Supported Devices**
  - AX series
- **Library**
  - DL_LRC.library

5.3. DFC_LRC32

DFC_LRC32 calculates the LRC (32–bit) checksum

FB/FC	Instruction	Graphic Expression	ST Language
FC	DFC_LRC32		DFC_LRC32(pSrc:= , wLen:= , ErrorID=>);

- **Inputs**

Name	Function	Data Type	Setting Value (Default value)
pSrc	The start address for LRC calculation	POINTER TO BYTE	Memory address 1–512 (0)
dwLen	The data length for LRC calculation	DWORD*	(0)

***Note** [1]: The variable type BYTE and WORD can be used for dwLen input.

***Note** [2]: The memory address given by the *pSrc* exceeds the usable range, it may cause controller exceptions, such as %I, %Q and %M.

- **Outputs**

Name	Function	Data Type	Output Range (Default value)
DFC_LRC32	LRC checksum (This parameter is returned)	DWORD	(0)
ErrorID	Error codes	DL_LRC_ERROR	DL_LRC_ERROR (DFC_NO_ERROR)

- **Function**

  After running this FC instruction, begins to calculate LRC (32–bit) checksum from the start memory address *pSrc*. The calculation scope is determined by the input wLen.

- **Programming Example**

  The example uses FC instruction (DFC_LRC32) to perform calculating the LRC (32–bit) checksum.

```
PROGRAM PLC_PRG
VAR
   bVar0: BOOL;
   dwVar0: DWORD;
   ar_dwVar0: ARRAY [0..3] OF DWORD := [16#30313233,16#31323334,16#32333435,16#33343536];
END_VAR

IF bVar0 THEN
   dwVar0:=DFC_LRC32(pSrc:=ADR(ar_dwVar0[0]) , dwLen:=4 , ErrorID=> );
   bVar0:=FALSE;
END_IF;
```

  The checksum calculation scope is 4(dwLen = 4), therefore, the FC instruction(DFC_LRC32) will starts calculating checksums of four consecutive BYTE data from the memory address input to *pSrc*(ar_byVar0[0]) and will result in a checksum value of 16#3935312E.

  Note: From library version 1.0.0.1, change the input from dwLen to wLen.

- **Supported Devices**
  - AX series
- **Library**
  - DL_LRC.library

5.4. Error Codes and Troubleshooting

Description	Cause of Error	Corrective Action
DFC_LRC_ERR_PARAMETER	The value of wLen is incorrect.	Make sure the wLen value is greater than zero and does not exceed the upper limit.

Chapter 6: Module Read–write Instructions

The instructions in this chapter applies only to the AX-3 expansion module (AS Series module). The instructions can be used to read or set module parameters, or perform specific functions of the module.

FB/FC	Description	Library
DFB_From	Reads the CR data in the module.	DL_ASModuleAPI_AX3
DF_To	Writes a value to the CR data in the module.	DL_ASModuleAPI_AX3
DFB_DLCCAL	Calibrates the weight for the AS02LC load cell module.	DL_ASModuleAPI_AX3
DFB_DLCWEI	Measures the weight for the AS02LC load cell module.	DL_ASModuleAPI_AX3
DFB_DPUCONF	Sets the output parameter of the PU module.	DL_ASModuleAPI_AX3
DFB_PUSTAT	Reads the PU module output status.	DL_ASModuleAPI_AX3
DFB_DPUPLS	Outputs the PU module pulse (No acceleration/deceleration).	DL_ASModuleAPI_AX3
DFB_DPUDRI	Outputs the PU module relative positioning (With acceleration/deceleration).	DL_ASModuleAPI_AX3
DFB_DPUDRA	Outputs the PU module absolute positioning (With acceleration/deceleration).	DL_ASModuleAPI_AX3
DFB_DPUZRN	Returns the PU module to the homing position.	DL_ASModuleAPI_AX3
DFB_DPUJOG	Outputs the PU module inching.	DL_ASModuleAPI_AX3
DFB_DMPID	Operates the PID of the RTD/TC module.	DL_ASModuleAPI_AX3
DFB_DPUCNT	Activates or deactivates the PU module high-speed counter .	DL_ASModuleAPI_AX3
DFB_DHCCAP	Starts or stops the capture for HC counting.	DL_ASModuleAPI_AX3
DFB_HCDO	Controls the HC module output point.	DL_ASModuleAPI_AX3
DFB_DHCCMP	Compares the outputs of for the HC module.	DL_ASModuleAPI_AX3
DFB_DHCCMPT	Compares the count value of the HC module with the matrix table, and output the action when the condition is valid.	DL_ASModuleAPI_AX3
DFB_DHCMEAS	Measures the HC module frequency and rotation speed.	DL_ASModuleAPI_AX3
DFB_DADLOG	Records the analog input module data	DL_ASModuleAPI_AX3
DFB_DADPEAK	Records the Analog input module peak.	DL_ASModuleAPI_AX3
DFB_SCMRS	Sends and receives communication data via the COM port.	DL_ASModuleAPI_AX3

6.1. DFB_From

DFB_From reads the CR data in the module.

FB/FC	Instruction	Graphic Expression	ST Language
FB	DFB_ From		DFB_From(bExecute:= , byRemoteID:= , byLocalID:= , wCRAddr:= , iLength:= , pVal:= , bDone=> , bBusy=> , bError=> , ErrorID=>);

- **Inputs**

Name	Function	Data Type	Setting Value (Default value)
bExecute	Run the instruction when *bExecute* changes to TRUE.	BOOL	TRUE/FALSE (FALSE)
byRemoteID*	The CPU or remote module ID	BYTE	0: CPU 1–15: Remote module (0)
byLocalID	Expansion module ID	BYTE	0–31
wCRAddr	The CR data positon in the module.	WORD	(0)
iLength	The CR data length	INT	1–8 (0)
pVal	The CR data length	POINTER TO WORD	

***Note**: Only support mode 0.

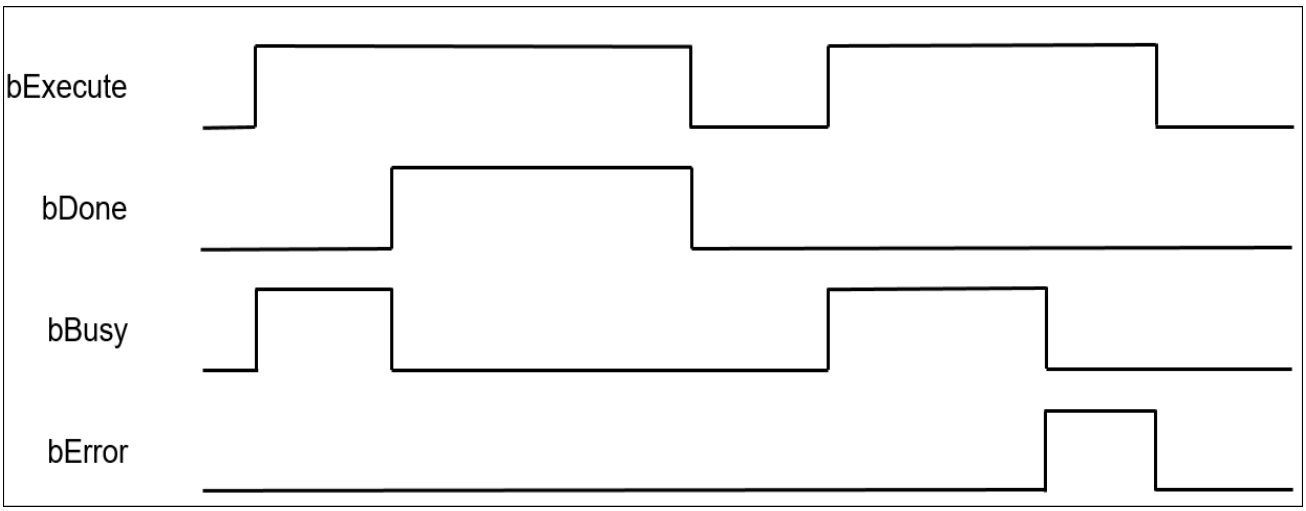
- **Outputs**

Name	Function	Data Type	Output Range (Default value)
bDone	TRUE when the running of the instruction is completed	BOOL	TRUE/FALSE (FALSE)
bBusy	TRUE when the instruction is running	BOOL	TRUE/FALSE (FALSE)
bError	TRUE if an error occurs	BOOL	TRUE/FALSE (FALSE)
ErrorID	Indicates the error code if an error occurs.	DFB_AS_MODULE_API_ERROR	DFB_AS_MODULE_API_ERROR (DFB_NO_ERROR)

- **Output Updating Timing**

Name	Timing for shifting to TRUE	Timing for shifting to FALSE
bDone	When the running of FB is completed	When *bExecute* shifts to FALSE
bBusy	When the running of FB starts	• When the running of FB is completed • When *bExecute* shifts to FALSE
bError	When an error occurs during the running or there are incorrect input values	When *bExecute* shifts to FALSE
ErrorID		

- **Timing Diagram**



- **Function**
  - We suggest placing this instruction under Main Task.
  - The function block DFB_From reads the CR data in the module.

- **Programming Example**

This example uses DFB_From to read the value of CR1 in the second module on the right side of the CPU and store the value in the variable (wVar) of the controller.

```
PROGRAM PLC_PRG
VAR
   bVar0: BOOL :=TRUE;
   bExecute_Var,bDone_Var,bBusy_Var,bError_Var: BOOL;
   byRemoteID_Var,byLocalID_Var: BYTE;
   wCRAddr_Var: WORD;
   iLength_Var: INT;
   ErrorID_Var: DFB_AS_MODULE_API_ERROR
   wVar0: WORD;
   FB0: DFB_From;
END_VAR

IF bVar0 THEN
   bExecute_Var:=TRUE;
   byRemoteID_Var:=0;
   byLocalID_Var:=2;
   wCRAddr_Var:=l;
   iLength_Var:=1;
   bVar0:=FALSE;
END_IF
IF bDone_Var THEN
   bExecute_Var:=FALSE;
```

```
END_IF
FB0(
   bExecute:=bExecute_Var ,
   byRemotelD:=byRemoteID_Var ,
   byLocallD:=byLocalID_Var ,
   wCRAddr:=wCRAddr_Var ,
   iLength:=iLength_Var ,
   pVal:=ADR(wVar0) ,
   bDone=>bDone_Var ,
   bBusy=>bBusy_Var ,
   bError=>bError_Var ,
   ErrorID=>ErrorID_Var );
```

- **Supported Devices**
  - AX-3 series
- **Library**
  - DL_ASModuleAPI_AX3

6.2. DFB_To

DFB_To writes a value to the CR data in the module.

FB/FC	Instruction	Graphic Expression	ST Language
FB	DFB_ To		DFB_To(bExecute:= , byRemoteID:= , byLocalID:= , wCRAddr:= , iLength:= , pVal:= , bDone=> , bBusy=> , bError=> , ErrorID=>);

- **Inputs**

Name	Function	Data Type	Setting Value (Default value)
bExecute	Run the instruction when *bExecute* changes to TRUE.	BOOL	TRUE/FALSE (FALSE)
byRemoteID*	The CPU or remote module ID	BYTE	0: CPU 1–15: Remote module (0)
byLocalID	Expansion module ID	BYTE	0–31
wCRAddr	The CR data positon in the module	WORD	(0)
iLength	The CR data length	INT	1–8 (0)
pVal	The CR data to be written	POINTER TO WORD	

***Note**: Only support mode 0.

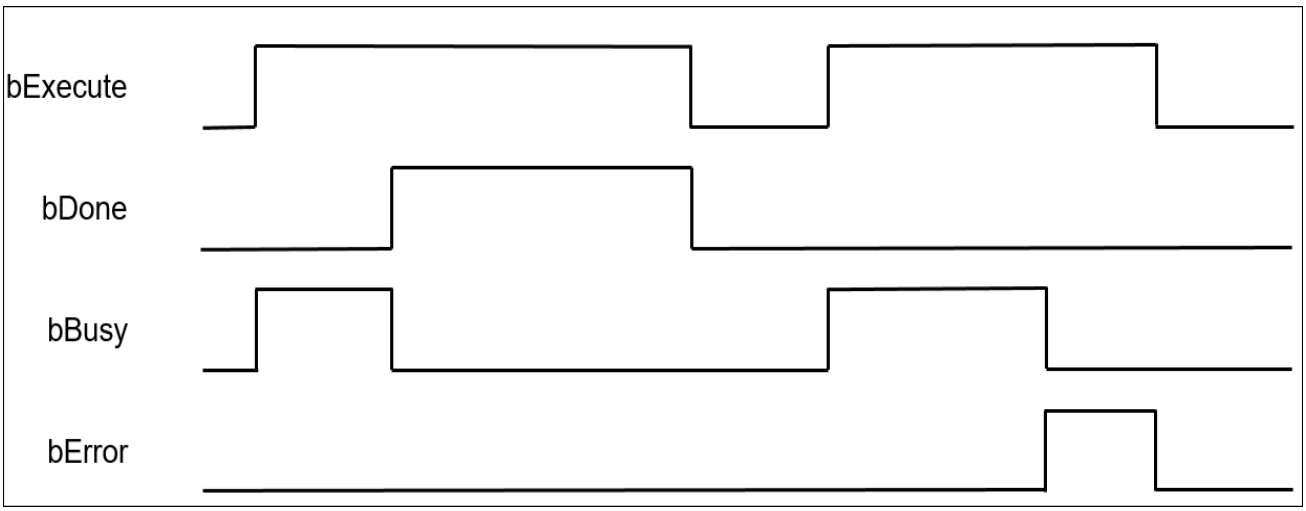
- **Outputs**

Name	Function	Data Type	Output Range (Default value)
bDone	TRUE when the running of the instruction is completed	BOOL	TRUE/FALSE (FALSE)
bBusy	TRUE when the instruction is running	BOOL	TRUE/FALSE (FALSE)
bError	TRUE if an error occurs	BOOL	TRUE/FALSE (FALSE)
ErrorID	Indicates the error code if an error occurs.	DFB_AS_MODULE_API_ERROR	DFB_AS_MODULE_API_ERROR (DFB_NO_ERROR)

- **Output Updating Timing**

Name	Timing for shifting to TRUE	Timing for shifting to FALSE
bDone	When the running of FB is completed	When *bExecute* shifts to FALSE
bBusy	When the running of FB starts	• When the running of FB is completed • When *bExecute* shifts to FALSE
bError	When an error occurs during the running or there are incorrect input values	When *bExecute* shifts to FALSE
ErrorID		

- **Timing Diagram**



- **Function**
  - It is suggested that this instruction be placed under Main Task.
  - The Function block DFB_To writes a value to the CR in the module.

- **Programming Example**

This example uses DFB_ To to write the value of variable (wVar) to CR1 in the second module on the right side of the CPU.

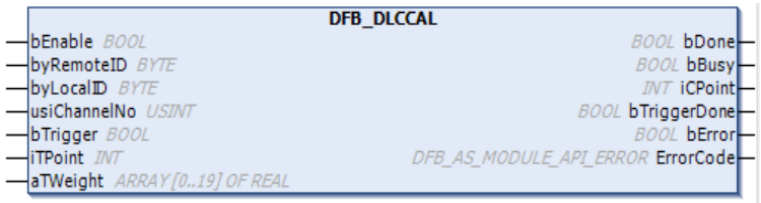
```
PROGRAM PLC_PRG
VAR
   bVar0: BOOL :=TRUE;
   bExecute_Var,bDone_Var,bBusy_Var,bError_Var: BOOL;
   byRemoteID_Var,byLocalID_Var: BYTE;
   wCRAddr_Var: WORD;
   iLength_Var: INT;
   ErrorID_Var: DFB_AS_MODULE_API_ERROR
   wVar0: WORD; =2;
   FB0: DFB_To;
END_VAR

IF bVar0 THEN
   bExecute_Var:=TRUE;
   byRemoteID_Var:=0;
   byLocalID_Var:=2;
   wCRAddr_Var:=l;
   iLength_Var:=1;
   bVar0:=FALSE;
END_IF
IF bDone_Var THEN
   bExecute_Var:=FALSE;
```

```
END_IF
FB0(
   bExecute:=bExecute_Var ,
   byRemotelD:=byRemoteID_Var ,
   byLocallD:=byLocalID_Var ,
   wCRAddr:=wCRAddr_Var ,
   iLength:=iLength_Var ,
   pVal:=ADR(wVar0) ,
   bDone=>bDone_Var ,
   bBusy=>bBusy_Var ,
   bError=>bError_Var ,
   ErrorID=>ErrorID_Var );
```

- **Supported Devices**
  - AX-3 series
- **Library**
  - DL_ASModuleAPI_AX3

6.3. DFB_DLCCAL

DFB_DLCCAL calibrates the weight for the AS02LC load cell module.

FB/FC	Instruction	Graphic Expression	ST Language
FB	DFB_DLCCAL		DFB_DLCCAL(bEnable:= , byRemoteID:= , byLocalID:= , usiChannelNo:= , bTrigger:= , iTPoint:= , aTWeight:= , bDone=> , bBusy=> , iCPoint=> , bTriggerDone=> , bError=> , ErrorCode=>);

- **Inputs**

Name	Function	Data Type	Setting Value (Default value)
bEnable	Run the function block	BOOL	TRUE/FALSE (FALSE)
byRemoteID*	The CPU or remote module ID	BYTE	0: CPU 1–15: Remote module (0)
byLocalID	Expansion module ID	BYTE	0–31 (0)
usiChannelNo	Specify channel number	USINT	1–2 (1)
bTrigger	Trigger single–point calibration	BOOL	TRUE/FALSE (FALSE)
iTPoint	Total number of calibration points	INT	2–20 (2)
aTWeight	Calibration weight value	ARRAY[0..19]OF REAL	1.0E–44–3.402823E+38 (0)

*__Note__: Only support mode 0.

- **Outputs**

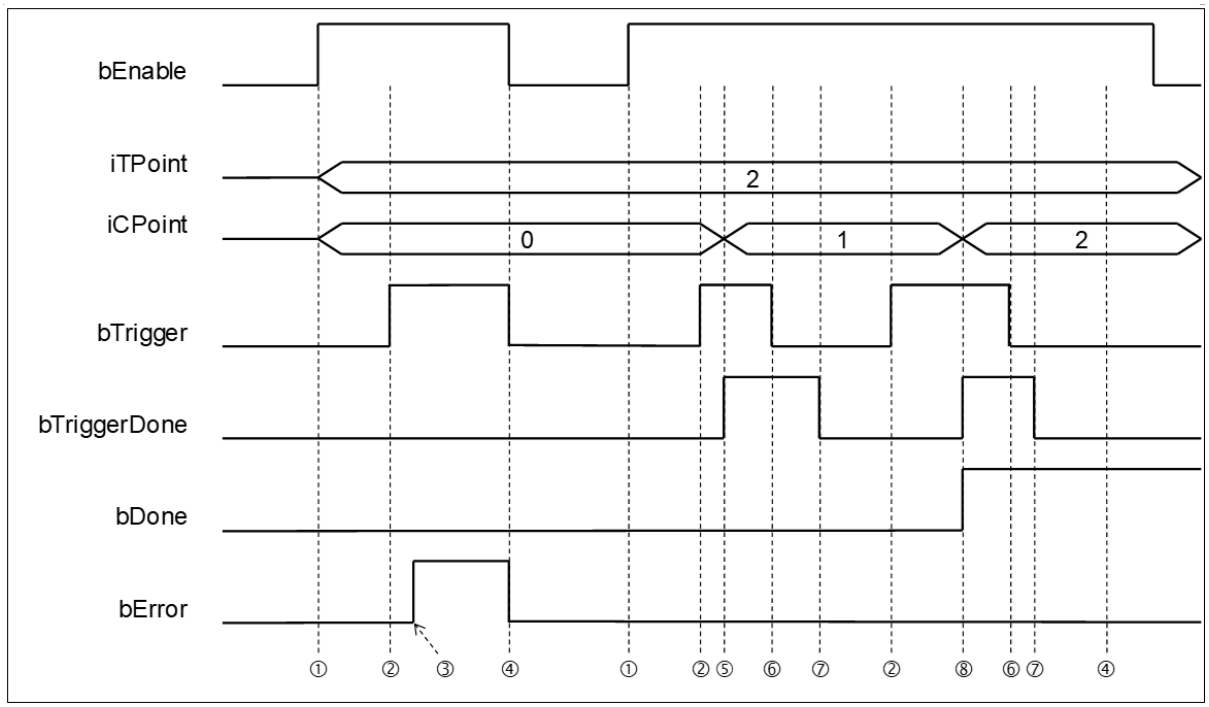
Name	Function	Data Type	Output Range (Default value)
bDone	All calibration is done.	BOOL	TRUE/FALSE (FALSE)
bBusy	Indicates that the function block is running.	BOOL	TRUE/FALSE (FALSE)

Name	Function	Data Type	Output Range (Default value)
iCPoint	Points number calibration is compeletd.	INT	0–20 (0)
bTriggerDone	Single calibration is done.	BOOL	TRUE/FALSE (FALSE)
bError	FB instruction error flag.	BOOL	TRUE/FALSE (FALSE)
ErrorCode	Error code	DFB_AS_MODULE_API_ERROR	DFB_AS_MODULE_API_ERROR (DFB_NO_ERROR)

- **Output Updating Timing**

Name	Timing for shifting to TRUE	Timing for shifting to FALSE
bDone	When all calibration is done	When *bExecute* shifts to FALSE
bBusy	When *bEnable* shifts to TRUE.	When *bEnable* shifts to FALSE
iCPoint	Add one when each calibration is done.	When *bEnable* shifts to FALSE, clear to zero.
bTriggerDone	Every time when the calibration is done	When *bEnable* shifts to FALSE
bError	When an error occurs during the running or there are incorrect input values	When *bEnable* shifts to FALSE
ErrorCode		

- **Timing Diagram**



The timing points in the timing diagram are described below:

① → Instruction starts.

② → The calibration flag is triggerred.

③ → Module number error is found.

④ → The instruction is closed. The *iCPoint* value and *bTrigger*, *bTriggerDone*, *bDone* & *bError* flags are automatically cleared.

⑤ → After being triggered, the LC module completes single–point calibration, *iCPoint* adds one, and bTriggerDone is set to ON.

⑥ → The *bTrigger* signal is cleared.

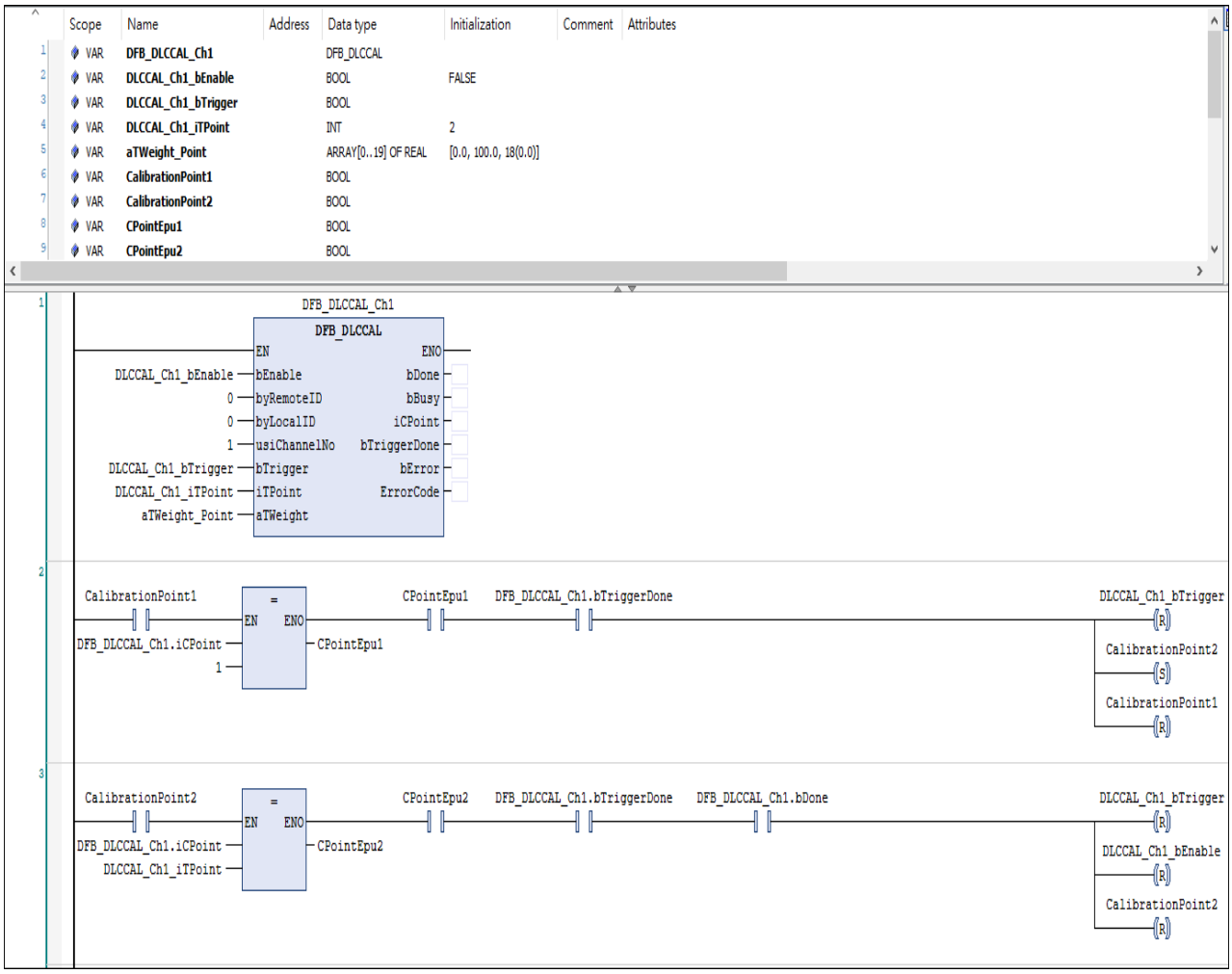
⑦ → The instruction follows to clear the *bTriggerDone* signal.

⑧ → After being triggered, LC modules are completely calibrated, *iCPoint* adds one, and *bTriggerDone* & *bDone* are set to ON.

- **Function**
  - It is suggested that this instruction be placed under Main Task.
  - This function is supported only when the AX-3 series firmware version is V1.0.1 and later.
  - This instruction is only used with the AS02LC load cell module, and supports AS02LC V1.04 and later.
  - This is a load cell module (AS02LC–A) instruction, and it is used to activate and deactivate the weight calibration function of the module.
  - *byRemoteID* specifies the group number which defines the load cell module should connect to the right of the controller or the right of the remote module. The controller number is 0, the first remote module number is 1, and so on. The maximum group number is 15.
  - *byLocalID* specifies module numbers. The first module on the right of the controller is 0, the number of the second module on the right of the controller is 1, and so on. All modules must be counted. The maximum number of modules is 32.
  - *usiChannelNo* specifies the channel numbers. The number of channel one is 1, and the number of channel two is 2.
  - *bTrigger* triggers the single–point calibration. When the *bTrigger* status changes from OFF to ON, LC module will be notified to perform a single calibration, and *bTriggerDone* flag will be set to ON after completion. If the calibrations of all calibrated points are done, the bDone flag will also be ON. Before proceeding the next calibration point, you need to check the *bTriggerDone* flag is ON, and change *bTrigger* to OFF. The instruction will also monitor the *bTrigger* flag when it turns from ON to OFF, and the *bTriggerDone* flag will be automatically cleared.
  - *iTPoint* is the total number of points for this calibration. After you activate the instruction, this value cannot be changed because this *iTPoint* has been sent to the LC module for calibration at the first activation.
  - *aTWeight* is the calibration weight value for each calibrated point, and the maximum number of calibration points for the LC module is 20. After the instruction is activated, you cannot change this value because this *aTWeight* has sent data to the LC module for calibration at the first activation. The first point calibration weight value must be 0. If it is not 0, the *bError* flag will be set to ON. For example, the total points of *iTPoint* calibration is 3, *aTWeight* gives [0.0, 100.0, 200.0, 17(0.0)] a total of 20 REAL–type ARRAY, of which 17(0.0) means there are 17 0.0.

- **Programming Example**

  This example uses the FB instruction (DFB_DLCCAL) to calibrate the first channel in the first module on the right of CPU.



1. Set CalibrationPoint1 to ON.
2. Make sure that the weight platform is unloaded first, set DLCCAL_CH1_*bEnable* to ON, and then set DLCCAL_CH1_bTrigger to ON. When the *iCPoint* value becomes 1, and *bTriggerDone* becomes TRUE, it means that this first point calibration is done.
3. Place 100.0g weights on the weight platform and after ensuring the platform is steady, set DLCCAL_CH1_bTrigger to ON. Now, *iCPoint* becomes 2, and *bTriggerDone* & *bDone* are TRUE, which means that all calibrations are done. After making the third network established, deactivate DFB_DLCCAL function block to complete this calibration.

- **Supported Devices**
  - AX-3 series
- **Library**
  - DL_ASModuleAPI_AX3.library

6.4. DFB_DLCWEI

DFB_DLCWEI measures the weight for the AS02LC load cell module.

FB/FC	Instruction	Graphic Expression	ST Language
FB	DFB_DLCWEI		DFB_DLCWEI(bEnable:= , byRemoteID:= , byLocalID:= , usiChannelNo:= , rStable:= , bZeroS:= , bTareS:= , bBusy=> , rTareW=> , rWeight=> , iStatus=> , bError=> , ErrorCode=>);

- **Inputs**

Name	Function	Data Type	Setting Value (Default value)
bEnable	Run the function block	BOOL	TRUE/FALSE (FALSE)
byRemoteID*	The CPU or remote module ID	BYTE	0: CPU 1–15: Remote module (0)
byLocalID	Expansion module ID	BYTE	0–31 (0)
usiChannelNo	Specify channel number	USINT	1–2 (1)
rStable	Set the weight stability range	REAL	0.0–100000.0 (0)
bZeroS	Set the weight to zero flag	BOOL	TRUE/FALSE (FALSE)
bTareS	Set the tare weight flag	BOOL	TRUE/FALSE (FALSE)

*__Note__: Only support mode 0.

- **Outputs**

Name	Function	Data Type	Output Range (Default value)
bBusy	Indicates that the function block is running.	BOOL	TRUE/FALSE (FALSE)
rTareW	Tare weight	REAL	1.0E–44–3.402823E+38(0)
rWeight	Current weight	REAL	1.0E–44–3.402823E+38(0)

Name	Function	Data Type	Output Range (Default value)
iStatus*	LC module status code	INT	0–5(0)
bError	FB instruction error	BOOL	TRUE/FALSE (FALSE)
ErrorCode	Error code	DFB_AS_MODULE_ API_ERROR	DFB_AS_MODULE_ API_ERROR (DFB_NO_ERROR)

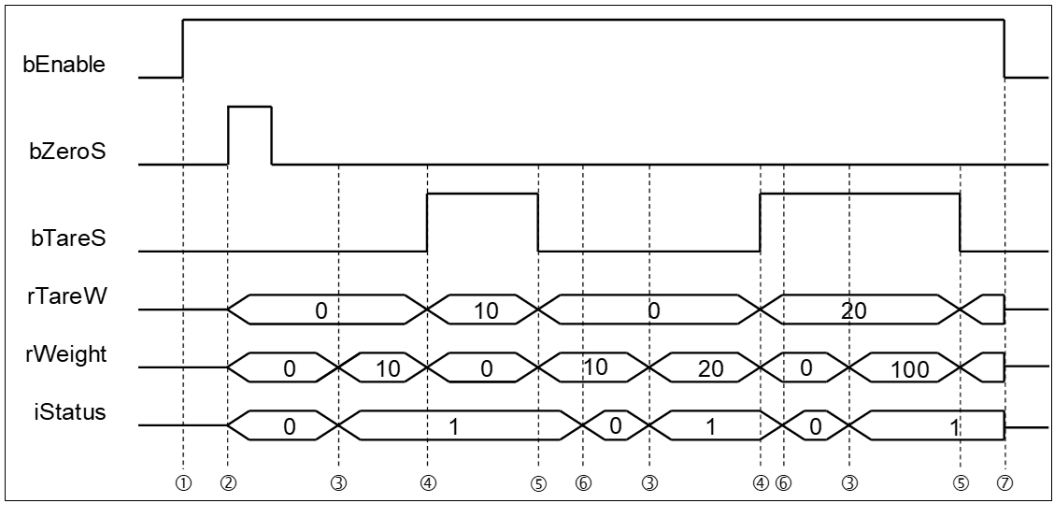
***Note**: *iStatus* integrates the LC module's common status code for this instruction. Its statuses are as follows:

Value	0	1	2	3	4	5
Description	Measuring or unloaded	Weight is steady.	Hardware/Calibration error	Calibrating	Weight is out of range.	Module number/Channel error

- **Output Updating Timing**

Name	Timing for shifting to TRUE	Timing for shifting to FALSE
bBusy	When *bEnable* shifts to TRUE	When *bEnable* shifts to FALSE
rTareW	Continuously update the value when *bEnable* is TRUE.	When *bEnable* shifts to FALSE, clear to zero.
rWeight	Continuously update the weight value when *bEnable* is TRUE.	When *bEnable* shifts to FALSE, clear to zero.
iStatus	Continuously update the status when bEnable is TRUE.	When *bEnable* shifts to FALSE, clear to zero.
bError	When an error occurs during the running or there are incorrect input values	When *bEnable* shifts to FALSE
ErrorCode		

- **Timing Diagram**



The timing points in the timing diagram are described below:

① → Activate the instruction

② → When you clear to 0, the instruction will clear the *rTareW* & rWeight values and the iStatus status.

③ → You place objects on the weight platform. When the weight values are steady, *iStatus* becomes 1, and *rWeight* weight value is shown.

④ → You set the tare weight to ON. The *rWeight* weight is transferred to *rTareW*, and then the *rWeight* weight is cleared.

⑤ → You clear the tare weight setting OFF. The *rTareW* weight is transferred back to *rWeight*, and then the *rTareW* weight is cleared.

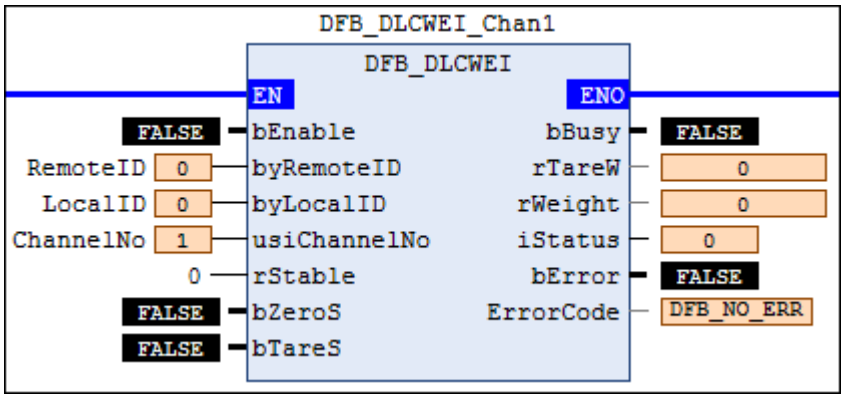
⑥ → You place another object on the weight platform again. Now, *iStatus* turns to measuring.

⑦ → Instruction is deactivated, and *rTareW*, *rWeight*, *iStatus* are cleared to 0.

- **Function**
  - It is suggested that this instruction be placed under Main Task.
  - This function is supported when the AX-3 series firmware version is V1.0.1 and later.
  - This instruction is only for use with the AS02LC weighting module, and the supported version is AS02LC V1.04 and later.
  - This is a load cell module (AS02LC–A) instruction, and it is used to activate and deactivate the weight measurement function of the module.
  - *byRemoteID* specifies the group number which defines the load cell module should connect to the right of the controller or the right of the remote module. The controller number is 0, the first remote module number is 1, and so on. The maximum group number is 15.
  - *byLocalID* specifies module numbers. The first module on the right of the controller is 0, the number of the second module on the right of the controller is 1, and so on. All modules must be counted. The maximum number of modules is 32.
  - *usiChannelNo* specifies the channel numbers. The number of channel one is 1, and the number of channel two is 2.
  - *rStable* is the weight stability range value. Its data type is REAL, and the inputable floating–point number value range is 0.0–100000.0. If the setting is out of range, the instruction will automatically set the value to the minimum/maximum values. The timing of the LC module parameter setting is when the instruction is started for the first time. To modify the value in the LC module later, you need to deactivate the instruction, set a new range value, and then open the instruction to reset.
  - After this instruction is activated, the specified channels will be automatically changed to “Net Weight” display mode. If you need to know the “Gross weight (total weight)” value, add *rTareW* and *rWeight*.
  - *bZeroS* is the flag that sets the current weight to 0. When this flag is from OFF to ON, *rTareW* and *rWeight* will be cleared to 0.
  - *bTareS* is the flag that sets the tare weight. When the bTareS flag is from OFF to ON, the current *rWeight* weight will be transferred to *rTareW*, and the *rWeight* value will be cleared to 0. When the bTareS flag is from ON to OFF, *rTareW* will be back to the *rWeight* current weight value, and the *rTareW* value will be cleared to 0.
  - *rWeight* is the weight value after deducting the tare weight. You can monitor if *rTareW* has a value to determine whether the tare function is activated. When the value is 0, it represents that the tare weight has not been set.

- **Programming Example**

  After completing DLCCAL calibration, the DLCWEI instruction can used to perform weight measurement.



1. Weight measurement: Place 500g weights on the weight platform. When *bEnable* is On, *rWeight* displays the current weight of 500.0.
2. Tare weight setting:
   - Place package material on the weight platform (e.g. 100 g). Now, *rWeight* shows that the current weight is 100.0.
   - When bTareS is ON, the *rWeight* weight will be transferred to *rTareW*, and then the *rWeight* weight will be cleared.
   - When *rWeight* = 0.0 (maybe a little unstable) and *rTareW* = 100.0, the tare weight setting is complete.
3. Clear tare weight setting:
   - When bTareS is OFF, the *rTareW* weight will be back to *rWeight*, and then the *rTareW* weight will be cleared.
   - When *rWeight* = 100.0 and *rTareW* = 0.0, the tare weight setting is cleared.
4. Weight stability range setting (Stability checking function):
   - Before *bEnable* is ON, set rStable = 10.0.
   - After placing 500g weights and setting *bEnable* to ON, when the measurement range is between 490–510g, *iStatus* = 1 (Weight is stable).

- **Supported Devices**
  - AX-3 series
- **Library**
  - DL_ASModuleAPI_AX3.library

6.5. DFB_DPUCONF

DFB_DPUCONF sets the output parameter of the PU module.

FB/FC	Instruction	Graphic Expression	ST Language
FB	DFB_DPUCONF		DFB_DPUCONF(bEnable:= , byLocalID:= , iAxis:= , iMode:= , iStartSpeed:= , iAccTime:= , iDecTime:= , diMaxSpeed:= , iZ_no:= , iOffset:= , bDone=> , bBusy=> , bError=> , ErrorCode=>);

- **Inputs**

Name	Function	Data Type	Setting Value (Default value)
bEnable	Run the function block	BOOL	TRUE/FALSE (FALSE)
byLocalID	Expansion module ID	BYTE	0–31 (0)
iAxis	Output axis number	INT	1–4 (1)
iMode	Output mode setting	INT	0–3 (1)
iStartSpeed	Start/End speed	INT	0–10000 Hz (100)
iAccTime	Acceleration time	INT	0–10000 ms (100)
iDecTime	Deceleration time	INT	0–10000 ms (100)
diMaxSpeed	Maximum output frequency	DINT	AS02PU: 100–200000 Hz (100K) AS04PU: 100–100000 Hz (100K)
iZ_no	Homing function and find the number of z–phase signals	INT	–100–100 times (0)
iOffset	Homing function is done, and the z–phase is found, then output the offset position.	INT	–10000–10000 numbers (0)

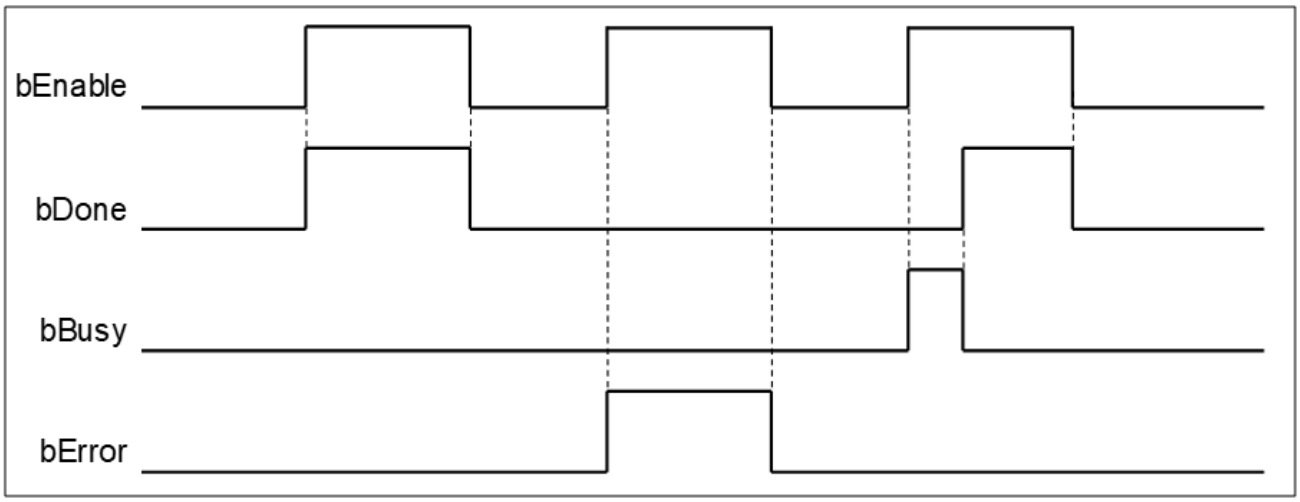
- **Outputs**

Name	Function	Data Type	Output Range (Default value)
bBusy	Indicates that the function block is running.	BOOL	TRUE/FALSE (FALSE)
bError	FB instruction error flag	BOOL	TRUE/FALSE (FALSE)
ErrorCode	Error code	DFB_AS_MODULE_API_ERROR	DFB_AS_MODULE_API_ERROR (DFB_NO_ERROR)
bDone	Parameter setting completion flag	BOOL	TRUE/FALSE (FALSE)

- **Output Updating Timing**

Name	Timing for shifting to TRUE	Timing for shifting to FALSE
bBusy	When *bEnable* shifts to TRUE	When *bEnable* shifts to FALSE
bError	When an error occurs during the running or there are incorrect input values	When *bEnable* shifts to FALSE
ErrorCode		
bDone	When the parameter setting is done	When *bEnable* shifts to FALSE

- **Timing Diagram**



- **Function**
  - It is suggested that this instruction be placed under Main Task.
  - This function is supported only when AX-3 series firmware version is V1.0.1 and later.
  - *byLocalID* specifies module numbers. The first module on the right of the controller is 0, the number of the second module on the right of the controller is 1, and so on. All modules must be counted. The maximum number of modules is 32. This PU dedicated instruction is only for the PU module on the right of the controller and is **not** for the PU module on the right of the remote module. If the specified module is not the PU module, the *bError* will be set to ON.
  - *iAxis* is the axis number of the specified output PU module. The input values 1–4 respectively represent the specified PU module axis 1–axis 4 output. If the PU module does not have this axis number, the *bError* flag will be set to ON. The combination of the axis number and the corresponding output points is as follows:

PU Module Name	Axis1 Combination	Axis2 Combination	Axis3 Combination	Axis4 Combination
AS04PU	Y0.0 / Y0.1	Y0.2 / Y0.3	Y0.4 / Y0.5	Y0.6 / Y0.7
AS02PU	Y0.0 / Y0.1	Y0.2 / Y0.3	NA	NA

- iModeselects selects parameters for the output mode of the output axis, and the setting values are as shown in the following table:

Output Mode Value	Mode	Note
N	Single–point pulse output (even–point output only)	For example: Y0.0 or Y0.2 output
1	Pulse (even points) + Direction (odd points)	For example: Y0.0 is pulse, Y0.1 is direction. When the direction is positive, Y0.1 is OFF; when the direction is negative, Y0.1 is ON.
2	CW (even points) + CCW (odd points)	For example: Y0.0 is CW (Positive direction), and Y0.1 is CCW (negative direction).
3	A–phase (even points) + B–phase (odd points)	For example: Y0.0 is A, and Y0.1 is B. When A is ahead of B, it represents positive direction output; when B is ahead of A, it represents negative direction output.
Others	Automatically change to mode 1 (Default)	–

- *iStartSpeed–iOffset*are not retain persistent value. If the value is out of range, *bError* will be reported.

- *bDone* is the output axis of the specified PU module, and is the parameter setting completion flag. When the flag is ON, it represents that parameter is set successfully. You can perform the subsequent positioning output function according to the flag status. You need to clear the bDone parameter by yourself. This instruction only sets this parameter after the setting is done.

- *bError* is the output axis of the specified PU module and the parameter error flag. Because most parameter ranges are automatically filtered by PLC, if this error flag occurs, it means that there is no specified PU module, the PU module number is incorrect, or the output axis number is incorrect.

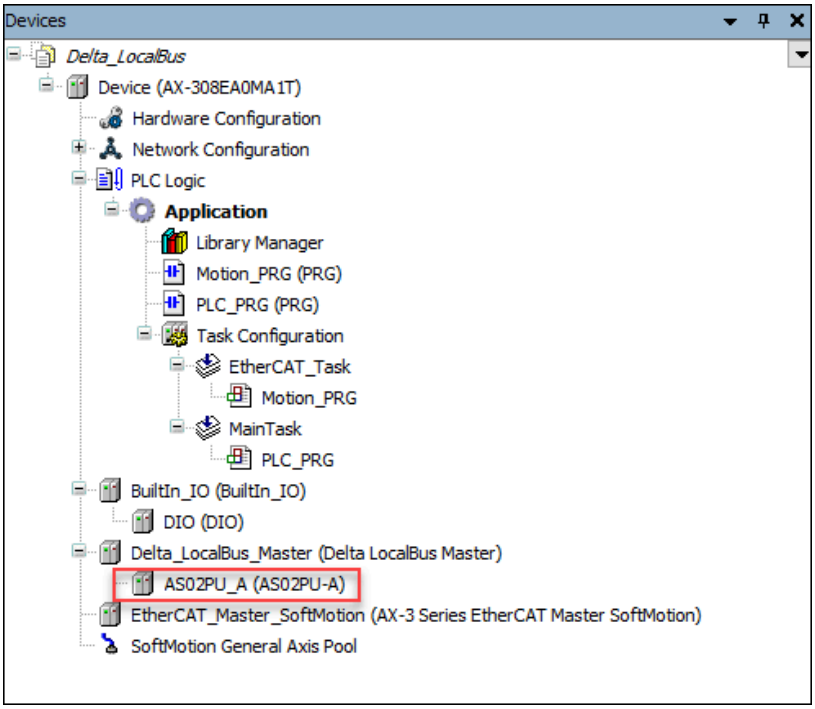
- This parameter setting instruction is pulse running instruction. Even if the A contact method is used for the conditional contact, this instruction will set the parameters of the PU module when it’s started. Therefore, when the axis parameters are changed, restart the instruction and reset the parameters.

- Because the parameter setting is executed through the module communication, check the *bDone* or *bError* results each time the parameters are modified, and then perform the related output action.

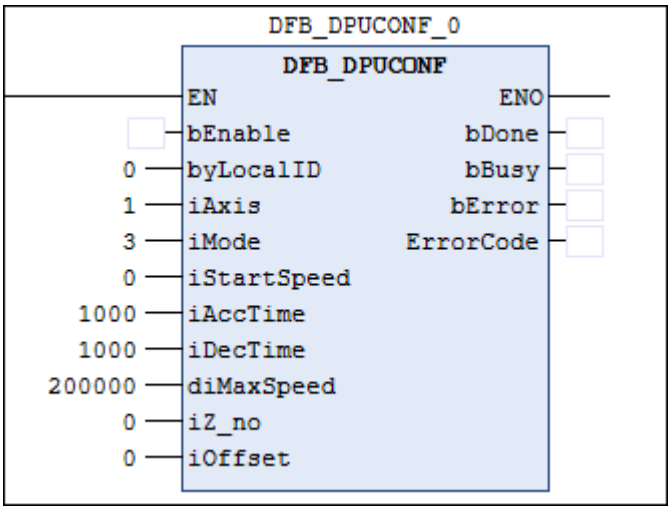
- **Programming Example**

  The following example shows how to Run the DFB_DPUCONF function block to set the parameter setting of the PU output module.

1. Delta_LocalBus_Master configures an AS02PU_A.



2. Use the DFB_DPUCONF function block to set the first axis parameter of the 02PU module on the right.



- **Supported Devices**
  - AX-3 series
- **Library**
  - DL_ASModuleAPI_AX3.library

6.6. DFB_PUSTAT

DFB_PUSTAT reads the PU module output status.

FB/FC	Instruction	Graphic Expression	ST Language
FB	DFB_ PUSTAT		DFB_PUSTAT(bEnable:= , byLocalID:= , iAxis:= , bZeroSet:= , bBusy=> , diCurrentPosi => , bMoving=> , bPause=> , bError=> , ErrorCode=>);

- **Inputs**

Name	Function	Data Type	Setting Value (Default value)
bEnable	Run the function block	BOOL	TRUE/FALSE (FALSE)
byLocalID	Expansion module ID	BYTE	0–31 (0)
iAxis	Output axis number	INT	1–4 (1)
bZeroSet	The current output location is cleared to 0.	BOOL	TRUE/FALSE (FALSE)

- **Outputs**

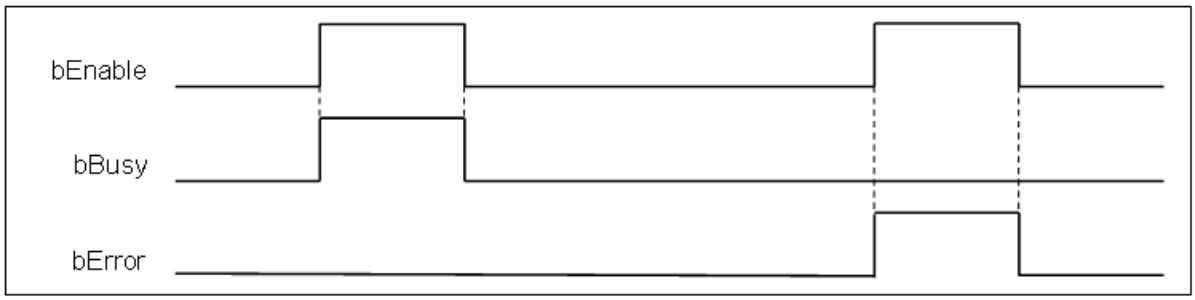
Name	Function	Data Type	Output Range (Default value)
bBusy	Indicates that the function block is running.	BOOL	TRUE/FALSE (FALSE)
diCurrentPosi	Current output position	DINT	0–2,147,483,648 (0)
bMoving	Output running flag	BOOL	TRUE/FALSE (FALSE)
bPause	Output pause flag	BOOL	TRUE/FALSE (FALSE)
bError	FB instruction error flag	BOOL	TRUE/FALSE (FALSE)
ErrorCode	Error code	DFB_AS_MODULE_API_ERROR	DFB_AS_MODULE_API_ERROR (DFB_NO_ERROR)

- **Output Updating Timing**

Name	Timing for shifting to TRUE	Timing for shifting to FALSE
bBusy	When *bEnable* shifts to TRUE	When *bEnable* shifts to FALSE

Name	Timing for shifting to TRUE	Timing for shifting to FALSE
diCurrentPosi	Continuously updating when *bEnable* shifts to TRUE.	When *bEnable* shifts to FALSE
bMoving	If the module is outputting pulse after *bEnable* shifts to TRUE.	When *bEnable* shifts to FALSE
bPause	If the module is not outputting pulse after *bEnable* shifts to TRUE.	When *bEnable* shifts to FALSE
bError	When an error occurs during the running or there are incorrect input values	When *bEnable* shifts to FALSE
ErrorCode		

- **Timing Diagram**

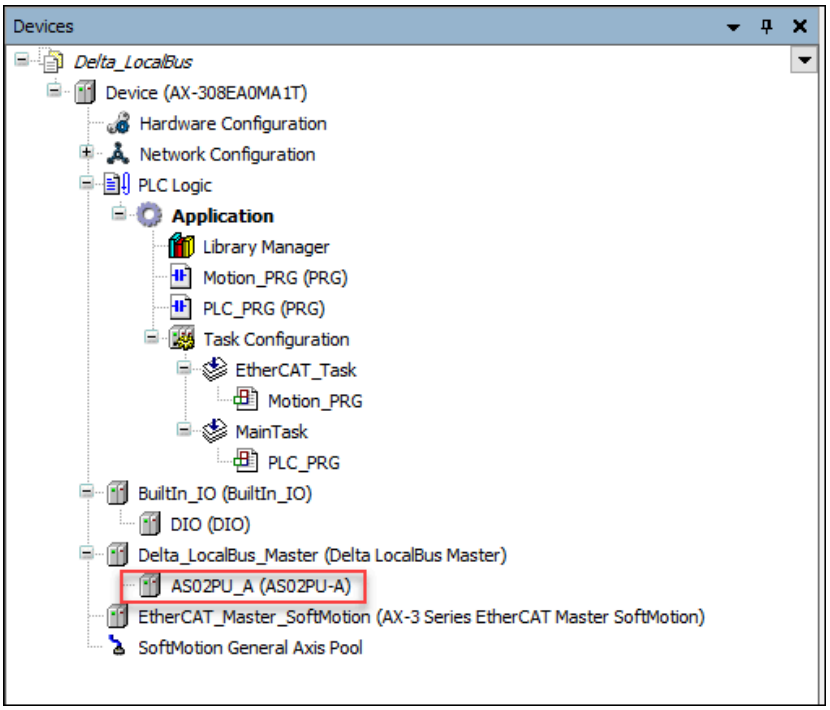


- **Function**
  - It is suggested that this instruction be placed under Main Task.
  - This function is supported only when AX-3 series firmware version is V1.0.1 and later.
  - *byLocalID* specifies module numbers. The first module on the right of the controller is 0, the number of the second module on the right of the controller is 1, and so on. All modules must be counted. The maximum number of modules is 32. This PU dedicated instruction is only for the PU module on the right of the controller and is **not** for the PU module on the right of the remote module. If the specified module is not the PU module, the *bError* will be set to ON.
  - *iAxis* is the axis number of the specified output PU module. The input values 1–4 respectively represent the specified PU module axis 1–axis 4 output. If the PU module does not have this axis number, the *bError* will be set to ON.
  - *diCurrentPosi* is the current position of the output axis of the specified PU module. This value is retain persistent, and is stored in the PU module. If you want to clear this value, clear the *bZeroSet* to 0 (OFF → ON) when the instruction is started.
  - *bMoving*(Read–only) represents the output axis of the PU module is running is the output running. When it is ON, it means that the output is in progress. When the flag is OFF, it means that the output axis is not being used and you can run the next output.
  - *bPause* (Read–only) represents the output aixs of the PU module is paused. When it is ON, the output is paused, the current speed is 0, and the current position has not yet reached the target position of the specified output. If you resume the output, it will be automatically cleared. **Note**: When the *bPause* flag is ON, the *bMoving* flag becomes OFF.
  - *bError*(Read–only) is the flag when the PU module has read or writer errors. When the error occurs, Refer to ErrorCode description.

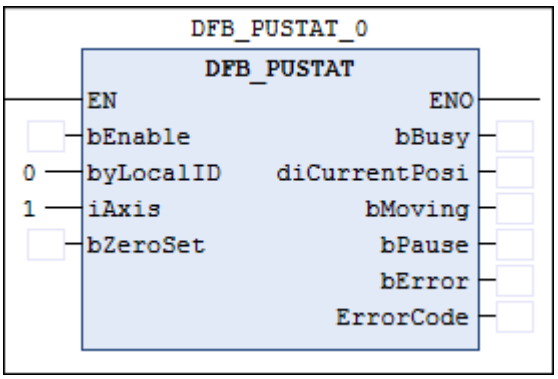
- **Programming Example**

  This example shows how to Run the DFB_PUSTAT function block to set the current output position of the first axis of the PU output module.

  1. Delta_LocalBus_Master configures an AS02PU_A.



  2. Use the DFB_PUSTAT function block to set the current output position of the first axis of the 02PU module.



- **Supported Devices**

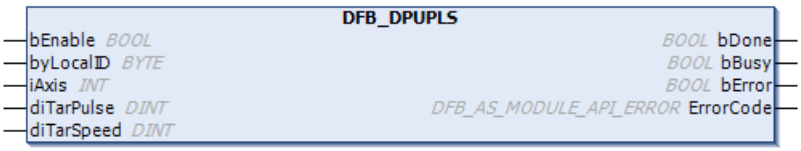
  - AX-3 series

- **Library**

  - DL_ASModuleAPI_AX3.library

6.7. DFB_DPUPLS

DFB_DPUPLS outputs the PU module pulse (No acceleration/deceleration).

FB/FC	Instruction	Graphic Expression	ST Language
FB	DFB_DPUPLS		DFB_DPUPLS(bEnable:= , byLocalID:= , iAxis:= , diTarPulse:= , diTarSpeed:= , bDone=> , bBusy=> , bError=> , ErrorCode=>);

- **Inputs**

Name	Function	Data Type	Setting Value (Default value)
bEnable	Run the function block	BOOL	TRUE/FALSE (FALSE)
byLocalID	Expansion module ID	BYTE	0–31 (0)
iAxis	Output axis number	INT	1–4 (1)
diTarPulse	Target output number	DINT	–2,147,483,648–2,147,483,648 (0)
diTarSpeed	Target output frequency (Unit: Hz)	DINT	AS02PU: –200K–200K (0) AS04PU: –100K–100K (0)

- **Outputs**

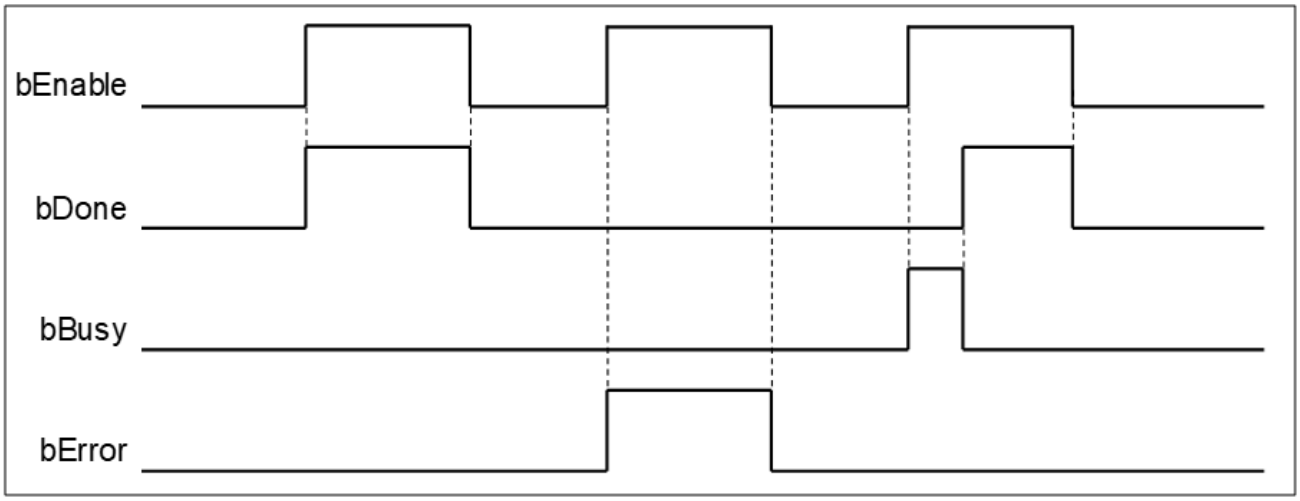
Name	Function	Data Type	Output Range (Default value)
bDone	Pulse output completion flag	BOOL	TRUE/FALSE (FALSE)
bBusy	Indicates that the function block is running.	BOOL	TRUE/FALSE (FALSE)
bError	FB instruction error flag.	BOOL	TRUE/FALSE (FALSE)
ErrorCode	Error code	DFB_AS_MODULE_API_ERROR	DFB_AS_MODULE_API_ERROR (DFB_NO_ERROR)

- **Output Updating Timing**

Name	Timing for shifting to TRUE	Timing for shifting to FALSE
bDone	When the pulse output is done	When *bEnable* shifts to FALSE
bBusy	When *bEnable* shifts to TRUE	When *bEnable* shifts to FALSE

Name	Timing for shifting to TRUE	Timing for shifting to FALSE
bError	When an error occurs during the running or there are incorrect input values	When *bEnable* shifts to FALSE
ErrorCode		

- **Timing Diagram**



- **Function**
  - It is suggested that this instruction be placed under Main Task.
  - This function is supported only when AX-3 series firmware version is v1.0.1 and later.
  - *byLocalID* specifies module numbers. The first module on the right of the controller is 0, the number of the second module on the right of the controller is 1, and so on. All modules must be counted. The maximum number of modules is 32. This PU dedicated instruction is only for the PU module on the right of the controller and is **not** for the PU module on the right of the remote module. If the specified module is not the PU module, the *bError* will be set to ON.
  - *iAxis* is the axis number of the specified output PU module. The input values 1–4 respectively represent the specified PU module axis 1–axis 4 output. If the PU module does not have this axis number, the *bError* flag will be set to ON.
  - *diTarPulse* is the pulse number of the specified output. The pulse number that can be input is the signed numbers 32–bit positive value. When the vale is 0, it means keeping outputting and the output numbers are not limited until the instruction is deactivated and the output stops; when the value is less than 0, PLC will automatically use the 2's complement method to convert to the number output of a positive integer number.
  - *diTarSpeed* is the target speed (Unit: Hz) of the specified output and the number that can be input is the signed numbers 32–bit value. You can modify the target frequency any time after the instruction starts output, and the PU module will switch to the latest target frequency after outputting a complete pulse. **Note**: Before changing the target frequency, consider if the change speed and the PLC scan time are appropriate. The corresponding *diTarSpeed* setting range of the module is as follows:

PU Module Name	diTarSpeed Setting Range
AS04PU	–100,000 (–100K)–100,000 (100K)
AS02PU	–200,000 (–200K)–200,000 (200K)

  - When the target speed of *diTarSpeed* is positive (>0), it means that the output point of "positive direction" is OFF; when the target speed of *diTarSpeed* is negative (<0), it means that the output

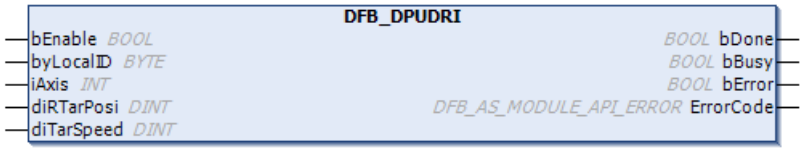
point of “negative direction” is ON; when the target speed of *diTarSpeed* is 0, it means that after outputting a complete running pulse, it enters the pause output status.

- This output instruction does not provide the acceleration/deceleration function. For acceleration/deceleration function requirement, use the DPUDRI instruction.
- This output instruction can be used for changing speed. When the instruction is running output, users can change the target frequency value of *diTarSpeed* to achieve the purpose of changing the output speed.
- When the output has reached the specified *diTarPulse* pulse number, the bDone completion flag will be set to ON. The bDone flag clearance needs to be run by users, and this instruction will set this flag once when the setting is done
- If any error situation occurs during output startup, the *bError* error flag will be set to ON. You can refer to ErrorCode for troubleshooting.

- **Supported Devices**
  - AX-3 series
- **Library**
  - DL_ASModuleAPI_AX3.library

6.8. DFB_DPUDRI

DFB_DPUDRI outputs the PU module relative positioning (With acceleration/deceleration).

FB/FC	Instruction	Graphic Expression	ST Language
FB	DFB_DPUDRI		DFB_DPUDRI(bEnable:= , byLocalID:= , iAxis:= , diRTarPosi:= , diTarSpeed:= , bDone=> , bBusy=> , bError=> , ErrorCode=>);

- **Inputs**

Name	Function	Data Type	Setting Value (Default value)
bEnable	Run the function block	BOOL	TRUE/FALSE (FALSE)
byLocalID	Expansion module ID	BYTE	0–31 (0)
iAxis	Output axis number	INT	1–4 (1)
diRTarPosi	Number of relative positioning outputs	DINT	–2,147,483,648– 2,147,483,648 (0)
diTarSpeed	Target output frequency (Unit: Hz)	DINT	AS02PU: –200K–200K (0) AS04PU: –100K–100K (0)

- **Outputs**

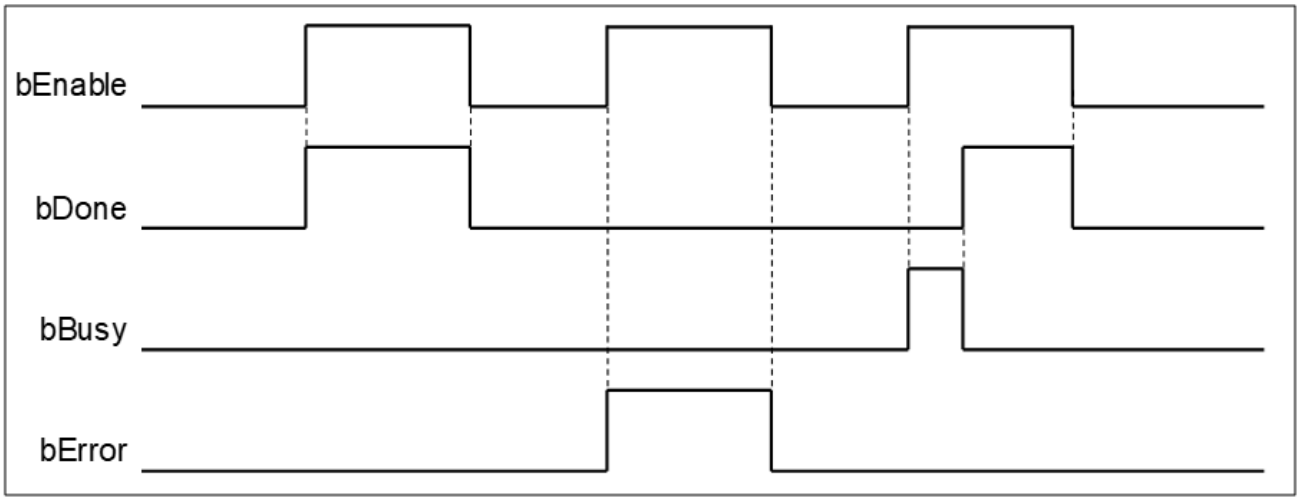
Name	Function	Data Type	Output Range (Default value)
bDone	Pulse output completion flag	BOOL	TRUE/FALSE (FALSE)
bBusy	Indicates that the function block is running.	BOOL	TRUE/FALSE (FALSE)
bError	FB instruction error flag	BOOL	TRUE/FALSE (FALSE)
ErrorCode	Error code	DFB_AS_MODULE_API_ERROR	DFB_AS_MODULE_API_ERROR (DFB_NO_ERROR)

- **Output Updating Timing**

Name	Timing for shifting to TRUE	Timing for shifting to FALSE
bDone	When the pulse output is done	When *bEnable* shifts to FALSE
bBusy	When *bEnable* shifts to TRUE	When *bEnable* shifts to FALSE

Name	Timing for shifting to TRUE	Timing for shifting to FALSE
bError	When an error occurs during the running or there are incorrect input values	When *bEnable* shifts to FALSE
ErrorCode		

- **Timing Diagram**

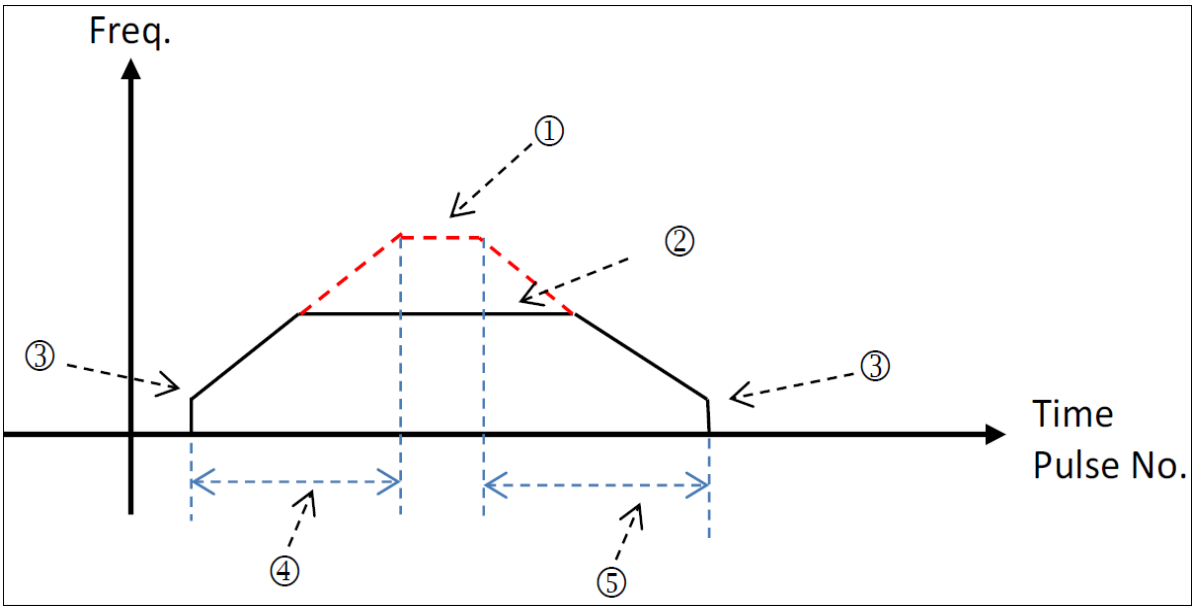


- **Function**

  - It is suggested that this instruction be placed under Main Task.
  - This function is supported only when AX-3 series firmware version is V1.0.1 and later.
  - *byLocalID* specifies module numbers. The first module on the right of the controller is 0, the number of the second module on the right of the controller is 1, and so on. All modules must be counted. The maximum number of modules is 32. This PU dedicated instruction is only for the PU module on the right of the controller and is **not** for the PU module on the right of the remote module. If the specified module is not the PU module, the *bError* will be set to ON.
  - *iAxis* is the axis number of the specified output PU module. The input values 1–4 respectively represent the specified PU module axis 1–axis 4 output. If the PU module does not have this axis number, the *bError* flag will be set to ON.
  - *diRTarPosi* is the position of the specified output relative positioning, and the pulse number that can be input is the signed numbers 32–bit value. When the value is greater than 0, the output is toward the positive direction (direction output point OFF); when the value is less than 0, the output is toward the negative direction (direction output point ON); when the value equals 0, this instruction will instantly set the bDone output completion flag to ON.
  - *diTarSpeed* is the target speed (Unit: Hz) of the specified output and the frequency value that can be input is the signed numbers 32–bit value. When the value is less than 0, the instruction will use the 2's complement method to convert to a positive integer number; when the values equals to 0, the instruction will inform the module to enter the pause mode. The actual output will decelerate according to deceleration slope until the output speed reaches 0, and users set the pause flag (refer to the PUSTAT instruction). The corresponding *diTarSpeed* setting range of the module is as follows:

PU Module Name	diTarSpeed Setting Range
AS04PU	–100,000 (–100K)–100,000 (100K)
AS02PU	–200,000 (–200K)–200,000 (200K)

- After starting the output, the target frequency can be changed anytime; however, when the frequency is actually changed, PLC will automatically change the frequency according to the acceleration/deceleration rate slope set by the DPUCONF instruction.
- When the output has reached the specified *diRTarPosi* relative positioning position, the bDone completion flag will be set to ON. The bDone flag clearance needs to be Run by users, and this instruction will set this flag once when the setting is done.
- During the output startup process, if any error situation occurs, the *bError* flag will be set to ON. You can refer to ErrorCode to perform troubleshooting.
- Acceleration/Deceleration curve of the PU module positioning output instruction is as follows:



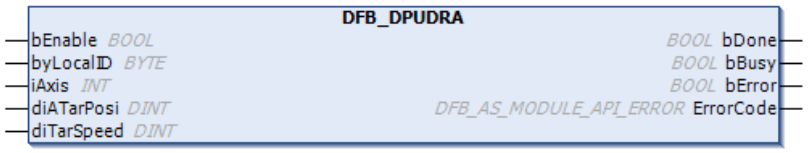
1. The value of the maximum output frequency. Refer to the DPUCONF instruction setting for this parameter.
2. The target frequency specified by the PU module output instruction. The target frequency cannot output the frequency that exceeds the maximum output frequency. The output will be limited to the maximum output frequency.
3. Start/End output frequency setting value. Refer to the DPUCONF instruction setting for this parameter.
4. Acceleration time value. Refer to the DPUCONF instruction setting for this parameter.
5. Deceleration time value. Refer to the DPUCONF instruction setting for this parameter.

   The acceleration/deceleration of the PU module is a fixed slope, so the actual acceleration/deceleration time will change according to the target frequency of the specified output. The acceleration/deceleration slope conversion formulas are (1) (maximum output frequency–startup frequency) / acceleration time, and (2) (maximum output frequency–end frequency) / deceleration time.

- **Supported Devices**
  - AX-3 series
- **Library**
  - DL_ASModuleAPI_AX3.library

6.9. DFB_DPUDRA

DFB_DPUDRA outputs the PU module absolute positioning (With acceleration/deceleration).

FB/FC	Instruction	Graphic Expression	ST Language
FB	DFB_DPUDRA		DFB_DPUDRI(bEnable:= , byLocalID:= , iAxis:= , diATarPosi:= , diTarSpeed:= , bDone=> , bBusy=> , bError=> , ErrorCode=>);

- **Inputs**

Name	Function	Data Type	Setting Value (Default value)
bEnable	Run the function block	BOOL	TRUE/FALSE (FALSE)
byLocalID	Expansion module ID	BYTE	0–31 (0)
iAxis	Output axis number	INT	1–4 (1)
diATarPosi	Number of relative positioning outputs	DINT	–2,147,483,648– 2,147,483,648 (0)
diTarSpeed	Target output frequency (Unit: Hz)	DINT	AS02PU: –200K–200K (0) AS04PU: –100K–100K (0)

- **Outputs**

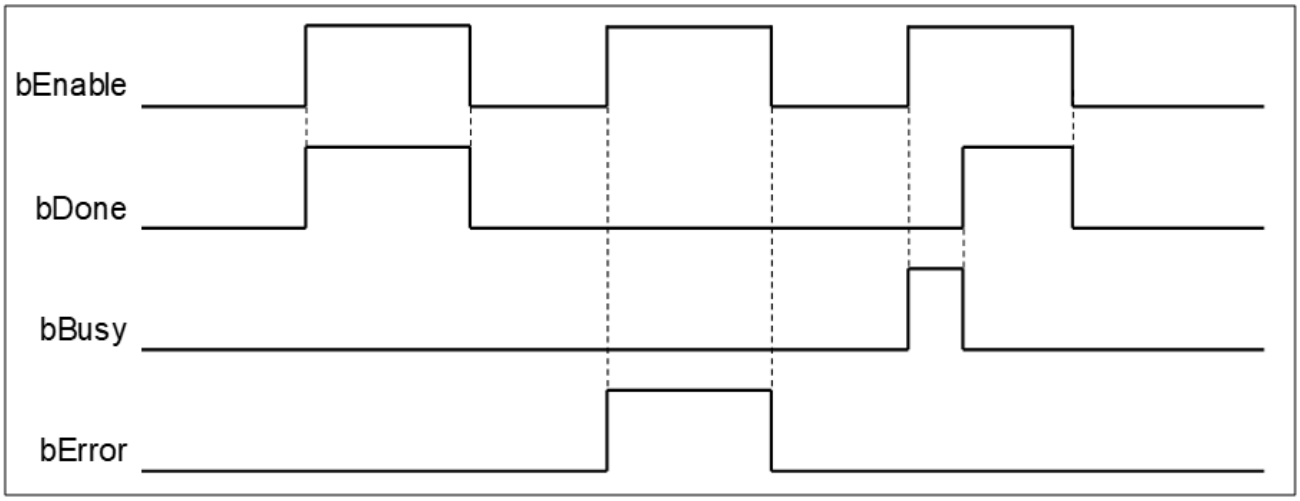
Name	Function	Data Type	Output Range (Default value)
bDone	Pulse output completion flag	BOOL	TRUE/FALSE (FALSE)
bBusy	Indicates that the function block is running.	BOOL	TRUE/FALSE (FALSE)
bError	FB instruction error flag	BOOL	TRUE/FALSE (FALSE)
ErrorCode	Error code	DFB_AS_MODULE_API_ERROR	DFB_AS_MODULE_API_ERROR (DFB_NO_ERROR)

- **Output Updating Timing**

Name	Timing for shifting to TRUE	Timing for shifting to FALSE
bDone	When pulse output is done	When *bEnable* shifts to FALSE
bBusy	When *bEnable* shifts to TRUE	When *bEnable* shifts to FALSE

Name	Timing for shifting to TRUE	Timing for shifting to FALSE
bError	When an error occurs during the running or there are incorrect input values	When *bEnable* shifts to FALSE
ErrorCode		

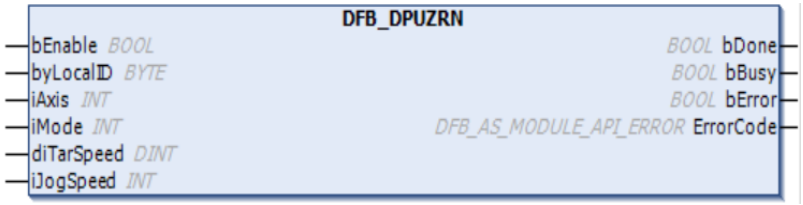
- **Timing Diagram**



- **Function**
  - It is suggested that this instruction be placed under Main Task.
  - This function is supported only when AX-3 series firmware version is V1.0.1 and later.
  - *diATarPosi* is the position of the absolute positioning of the specified output. The number of pulses that can be entered is a 32–bit number. The PU module will automatically compare the current position in the record. After comparison, if the value is greater than 0, it means that the output is toward the positive direction; if the value is less than 0, it means that the output is toward the negative direction; if the value is 0, this instruction will instantly set the *bDone* output completion to ON.
  - Refer to the DPUDRI instruction for the description of other parameters.
- **Supported Devices**
  - AX-3 series
- **Library**
  - DL_ASModuleAPI_AX3.library

6.10. DFB_DPUZRN

DFB_DPUZRN returns the PU module to the homing position.

FB/FC	Instruction	Graphic Expression	ST Language
FB	DFB_DPUZRN		DFB_DPUZRN(bEnable:= , byLocalID:= , iAxis:= , iMode:= , diTarSpeed:= , iJogSpeed:= , bDone=> , bBusy=> , bError=> , ErrorCode=>);

- **Inputs**

Name	Function	Data Type	Setting Value (Default value)
bEnable	Run the function block	BOOL	TRUE/FALSE (FALSE)
byLocalID	Expansion module ID	BYTE	0–31 (0)
iAxis	Output axis number	INT	1–4 (1)
iMode	Homing mode selection	INT	0–8, 255 (0)
diTarSpeed	Homing maximum output frequency	DINT	AS02PU:–200K to–100 and 100–200K (100) AS04PU: –100K to–100 100–100K (100)
iJogSpeed	Homing inching output frequency	INT	1–10000 (1)

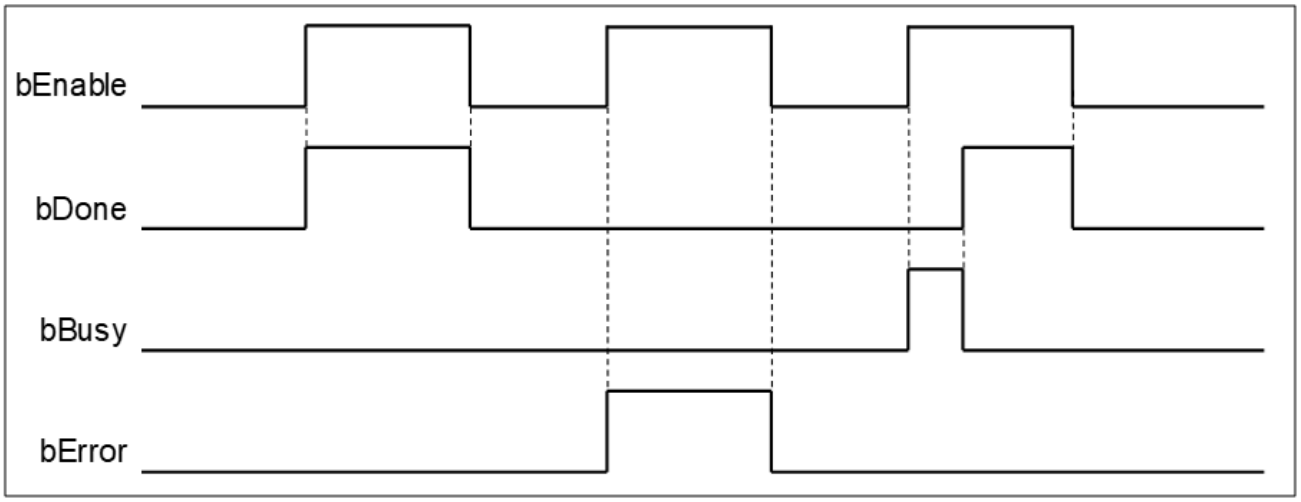
- **Outputs**

Name	Function	Data Type	Output Range(Default value)
bDone	Completion flag	BOOL	TRUE/FALSE (FALSE)
bBusy	Indicates that the function block is running.	BOOL	TRUE/FALSE (FALSE)
bError	FB instruction error flag	BOOL	TRUE/FALSE (FALSE)
ErrorCode	Error code	DFB_AS_MODULE_API_ERROR	DFB_AS_MODULE_API_ERROR (DFB_NO_ERROR)

- **Output Updating Timing**

Name	Timing for shifting to TRUE	Timing for shifting to FALSE
bDone	When the pulse output is done	When *bEnable* shifts to FALSE
bBusy	When *bEnable* shifts to TRUE	When *bEnable* shifts to FALSE
bError	When an error occurs during the running or there are incorrect input values	When *bEnable* shifts to FALSE
ErrorCode		

- **Timing Diagram**



- **Function**
  - It is suggested that this instruction be placed under Main Task.
  - This function is supported only when AX-3 series firmware version is V1.0.1 and later.
  - *byLocalID* specifies module numbers. The first module on the right of the controller is 0, the number of the second module on the right of the controller is 1, and so on. All modules must be counted. The maximum number of modules is 32. This PU dedicated instruction is only for the PU module on the right of the controller and is **not** for the PU module on the right of the remote module. If the specified module is not the PU module, the *bError* will be set to ON.
  - *iAxis* is the axis number of the specified output PU module. The input values 1–4 respectively represent the specified PU module axis 1–axis 4 output. If the PU module does not have this axis number, the *bError* flag will be set to ON.
  - *iMode* is the mode selection for homing. The mode description is as shown below:

Mode Parameter	Function	Select Matching Input Point (Parameter Setting of PU Module)	Remark
0	Directly clear the current position to 0.	None	
1	Stops when the negative direction leaves the origin.	DOG	
2	Stops when the positive direction leaves the origin.	DOG	
3	Look for z–phase times after mode 1 is done.	DOG and Z–phase input	Set Z–phase times with DPUCONF
4	Look for z–phase times after mode 2 is done.	DOG and Z–phase input	

Mode Parameter	Function	Select Matching Input Point (Parameter Setting of PU Module)	Remark
5	Output the offset position after mode 1 is done.	DOG	Set output offset position with DPUCONF
6	Output the offset position after mode 2 is done.	DOG	
7	After mode 1 is done, look for z–phase first, then output the offset position	DOG and Z–phase input	Set z–phase times and output the offset position with DPUCONF
8	After mode 2 is done, look for z–phase first, then output the offset position	DOG and Z–phase input	
255	Modify the current output position of the axis	None	Used with the **TarSpeed** parameter
Others	Reserved		

Note: If the mode selects the required input points that do not match the parameter setting of the PU module, the homing function may fail.

Note: For the above **iMode 1–4**, when the PU module firmware version is V1.02.00, the action is complete, the current output position of the axis will not be cleared to 0 when the action is complete, and it can be cleared using the PUSTAT instruction. For the PU module firmware version V1.02.10 and later, the current output position will be cleared to 0 when the action is complete .

Note: For the above **iMode 5–8**, when the PU module firmware version is v1.02.00, the current output position is the output result when the action is complete.The current position must be modified to the specified position with the mode 255. For the PU module firmware version V1.02.10 (included) and later, the axis current output position will be cleared to 0 when the action is complete.

- When *diTarSpeed*is in 1–8, it is the highest output speed to return to the homing position. This value is the signed numbers 32–bit value. Positive/Negative represents the default startup direction to fine the origin. For example, positive means that the instruction starts to look for the homing postion from the positive direction; the corresponding *diTarSpeed* setting range of the module is as follows: (If the mode parameter is specified as 255, the *diTarSpeed* will become the value of updating the PU module current position.)

PU Modul Name	When Mode is 1–8, diTarSpeed Setting Range
AS02PU	–200,000 to–100(Hz) and 100–200,000(Hz)
AS04PU	–100,000 to–100(Hz) and 100–100,000(Hz)

- *iJogSpeed*is the inching output speed when it touches the homing postion. This value is the signed number 16–bit value, and the setting value is 1–10,000 Hz.
- When the output has reached the specified homing postion, the *bDone* will be se5t to ON. You need to run *bDone* by yourself to clear. This instruction will set this flag once when the output is done.
- During the output startup process, if any error occurs, the *bError* flag will be set to ON. You can refer to ErrorCode to perform troubleshooting.

- **Supported Devices**
  - AX-3 series

- **Library**
  - DL_ASModuleAPI_AX3.library

6.11. DFB_DPUJOG

DFB_DPUJOG outputs the PU module inching.

FB/FC	Instruction	Graphic Expression	ST Language
FB	DFB_DPUJOG		DFB_DPUJOG(bEnable:= , byLocalID:= , iAxis:= , diJogSpeed:= , bBusy=> , bError=> , ErrorCode=>);

- **Inputs**

Name	Function	Data Type	Setting Value (Default value)
bEnable	Run the function block	BOOL	TRUE/FALSE (FALSE)
byLocalID	Expansion module ID	BYTE	0–31(0)
iAxis	Output axis number	INT	1–4(1)
diJogSpeed	Inching output frequency	DINT	AS02PU: –200K–200K(0) AS04PU: –100K–100K(0)

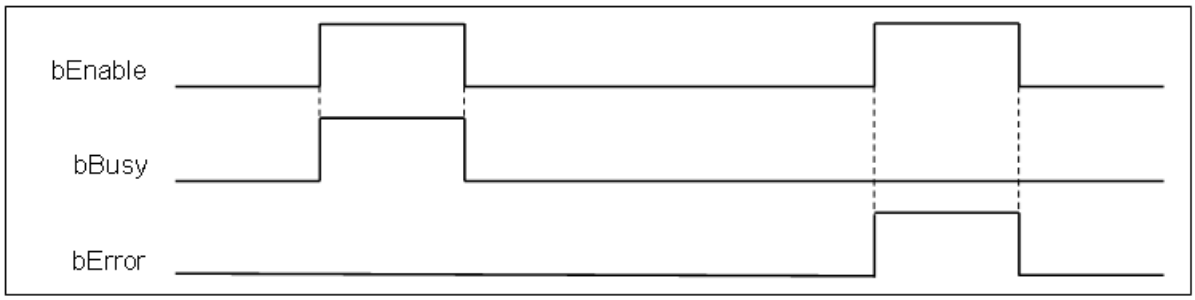
- **Outputs**

Name	Function	Data Type	Output Range (Default value)
bBusy	Indicates that the function block is running.	BOOL	TRUE/FALSE (FALSE)
bError	FB instruction error flag	BOOL	TRUE/FALSE (FALSE)
ErrorCode	Error code	DFB_AS_MODULE_ API_ERROR	DFB_AS_MODULE_API_ ERROR (DFB_NO_ERROR)

- **Output Updating Timing**

Name	Timing for shifting to TRUE	Timing for shifting to FALSE
bBusy	When *bEnable* shifts to TRUE	When *bEnable* shifts to FALSE
bError	When an error occurs during the running or there are incorrect input values	When *bEnable* shifts to FALSE
ErrorCode		

- **Timing Diagram**



- **Function**

  - It is suggested that this instruction be placed under Main Task.

  - This function is supported only when AX-3 series firmware version is V1.0.1 and later.

  - *byLocalID* specifies module numbers. The first module on the right of the controller is 0, the number of the second module on the right of the controller is 1, and so on. All modules must be counted. The maximum number of modules is 32. This PU dedicated instruction is only for the PU module on the right of the controller and is **not** for the PU module on the right of the remote module. If the specified module is not the PU module, the *bError* will be set to ON.

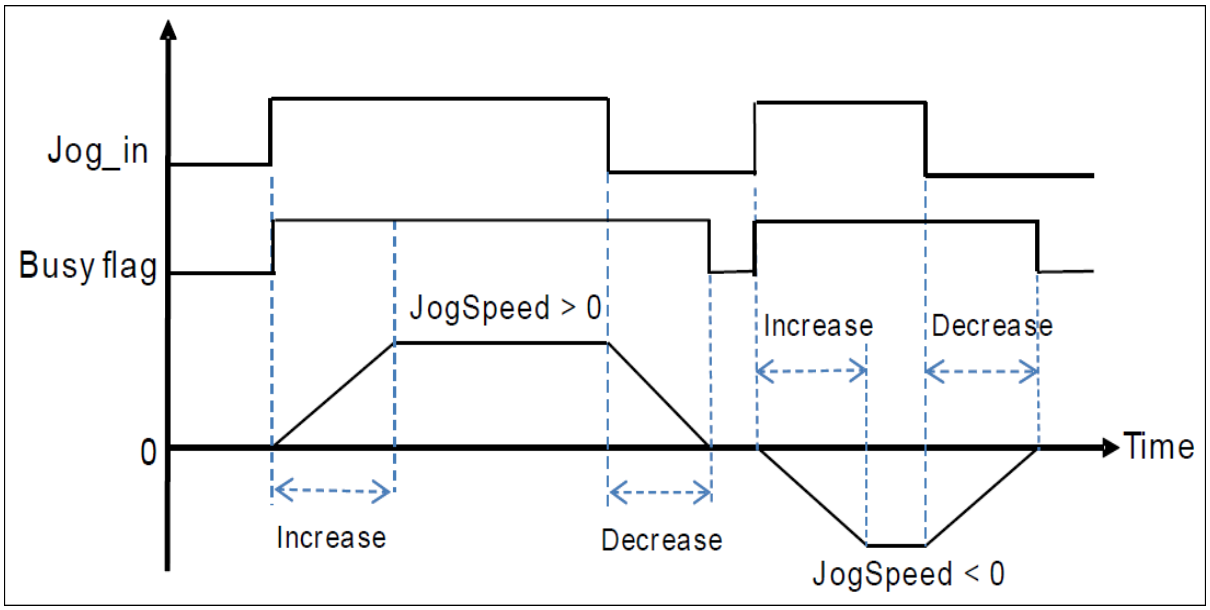
  - *iAxis* is the axis number of the specified output PU module. The input values 1–4 respectively represent the specified PU module axis 1–axis 4 output. If the PU module does not have this axis number, the *bError* flag will be set to ON.

  - *diJogSpeed* is the output speed of the specified inching, and this value is the signed number 32–bit value. When the value is greater than 0, it means that the output is toward the positive direction (Direction output point is OFF); when the value is less than 0, it means that the output is toward the negative direction (Direction output point is ON); when the value is 0, it means that the output stops. The corresponding *diJogSpeed* setting range of the module is as follows:

PU Module Name	JogSpeed Setting Range
AS02PU	–200,000(–200K)–200,000(200K)
AS04PU	–100,000(–100K)–100,000(100K)

  - If any error situation occurs during output startup, the *bError* error flag will be set to ON. You can refer to ErrorCode for troubleshooting.

  - The output timing diagram of the DPUJOG instruction is as follows: (In the timing diagram, Jog_in is the instruction startup switch, and *Busy* flag is the flag of *bBusy* runningning.)



- When the DPUJOG instruction is deactivated, you can perform other output control after the *bBusy* is OFF.

- **Supported Devices**
  - AX-3 series

- **Library**
  - DL_ASModuleAPI_AX3.library

6.12. DFB_DPUCNT

DFB_DPUCNT activates or deactivates the PU module high-speed counter .

FB/FC	Instruction	Graphic Expression	ST Language
FB	DFB_DPUCNT		DFB_DPUCNT(bEnable:= , byLocalID:= , iInputMode:= , iPeriod:= , bZeroSet:= , bBusy=> , diInputPulse=> , diInputSpeed => , bError=> , ErrorCode=>);

- **Inputs**

Name	Function	Data Type	Setting Value (Default value)
bEnable	Run the function block	BOOL	TRUE/FALSE (FALSE)
byLocalID	Expansion module ID	BYTE	0–31 (0)
iInputMode	Encoder input mode and counting frequency multiplication setting	INT	1–7 (4)
iPeriod	Speed fetch cycle time	INT	10–1000 (1000)
bZeroSet	Counter is cleared to 0.	BOOL	TRUE/FALSE (FALSE)

- **Outputs**

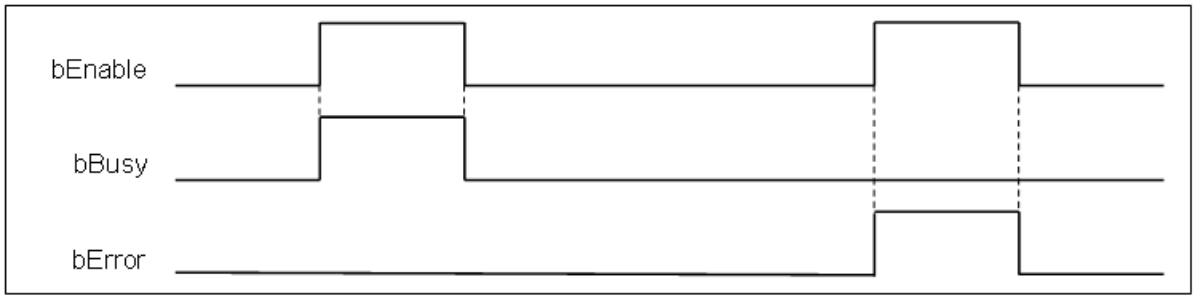
Name	Function	Data Type	Output Range (Default value)
bBusy	Indicates that the function block is running.	BOOL	TRUE/FALSE (FALSE)
diInputPulse	The entered number display	DINT	0– 2,147,483,648 (0)
diInputSpeed	Counting number of every cycle time	DINT	0– 2,147,483,648 (0)
bError	FB instruction error flag	BOOL	TRUE/FALSE (FALSE)
ErrorCode	Error code	DFB_AS_MODULE_API_ERROR	DFB_AS_MODULE_API_ERROR (DFB_NO_ERROR)

- **Output Updating Timing**

Name	Timing for shifting to TRUE	Timing for shifting to FALSE
bBusy	When *bEnable* shifts to TRUE	When *bEnable* shifts to FALSE

Name	Timing for shifting to TRUE	Timing for shifting to FALSE
diInputPulse	Continuously updating after *bEnable* shifts to TRUE.	When *bEnable* shifts to FALSE
diInputSpeed	Continuously updating after *bEnable* shifts to TRUE.	When *bEnable* shifts to FALSE
bError	When an error occurs during the running or there are incorrect input values.	When *bEnable* shifts to FALSE
ErrorCode		

- **Timing Diagram**



- **Function**
  - It is suggested that this instruction be placed under Main Task.
  - This function is supported by AX-3 series firmware version V1.0.1 and later.
  - This instruction is supported only by the AS02PU module.
  - *byLocalID* specifies module numbers. The number of the first module on the right of CPU is 0, the number of the second module on the right of CPU is 1, and so on. Regardless of any type of modules, all modules must be counted. The maximum number of modules is 32. This PU dedicated instruction is only for the PU module on the right of CPU and is **not** for the PU module on the right of the remote module. If the specified module is not the PU module, the *bError* flag will be set to ON.
  - *iInputMode* is the encoder source input mode and counting frequency multiplication setting, and its setting parameters are as follows:

    Note: When A–phase is ahead of B–phase, it represents the positive counting. When B–phase is ahead of A–phase, it represents the negative counting.

iInputMode Setting Description	
16#0000	Reserved
16#0001	1 time frequency multiplication A/B phase input
16#0002	2 times frequency multiplication A/B phase input
16#0003	Reserved
16#0004	4 times frequency multiplication A/B phase input (default value)
16#0005	Pulse + direction input (A+/A– is pulse input; B+/B– is direction input.) When B–phase is Off, it represents positive counting, and when it is On, it represents negative counting. A–phase uses rising edge–triggered counting.
16#0006	Pulse + direction input (A+/A– is pulse input; B+/B– is direction input.)

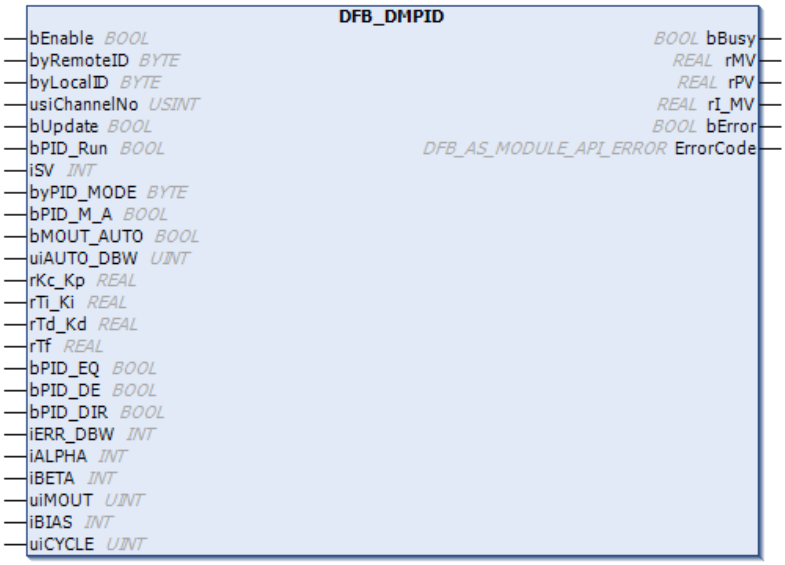
iInputMode Setting Description	
	When B–phase is On, it represents positive counting, and when it is Off, it represents negative counting. A–phase uses rising edge–triggered counting.
16#0007	Single–phase pulse input (A+/A– is pulse input.) A–phase uses rising edge–triggered counting.
Other value	Automatically transferred to (default value).
Value	Input Mode (Set the following reserved setting, and the module will Run in default values.)

- *iPeriod*is the cycle time setting value of speed fetching, and the settable range is 10ms–1000ms. If the value is out of range, it will be set to the minimum/maximum value.
- *diInputPulse* displays the entered pulse counting numbers (signed number 32–bit values), which are outage persistence value. If there is a need to clear it to 0, during the instruction startup, use the bZeroS setting flag (OFF → ON) to perform the clearance.
- *diInputSpeed* displays values that each *iPeriod* time counts (signed number 32–bit values). If users need to convert the values in Hz, use calculation formulas.
- When the instruction is activated/ deactivated, it means that the controller needs to inform the high-speed counter of the module to activate/deactivate. This instruction cannot be used with API1409 DPUMPG at the same time, otherwise the instructions may activate or deactivate the module counting with each other.

- **Supported Devices**
  - AX-3 series
- **Library**
  - DL_ASModuleAPI_AX3.library

6.13. DFB_DMPID

DFB_DMPID: Operates the PID of the RTD/TC module.

FB/FC	Instruction	Graphic Expression	ST Language
FB	DFB_DMPID		DFB_DMPID(bEnable:= , byRemoteID:= , byLocalID:= , usiChannelNo:= , bUpdate:= , bPID_Run:= , iSV:= , byPID_MODE:= , bPID_M_A:= , bMOUT_AUTO:= , uiAUTO_DBW:= , rKc_Kp:= , rTi_Ki:= , rTd_Kd:= , rTf:= , bPID_EQ:= , bPID_DE:= , bPID_DIR:= , iERR_DBW:= , iALPHA:= , iBETA:= , uiMOUT:= , iBIAS:= , uiCYCLE:= , bBusy=> , rMV=> , rPV=> , rI_MV=> , bError=> , ErrorCode=>);

- **Inputs**

Name	Function	Data Type	Setting Value (Default value)
bEnable	Run the function block	BOOL	TRUE/FALSE (FALSE)
byRemoteID[*1]	CPU or remote module number	BYTE	0: CPU

Name	Function	Data Type	Setting Value (Default value)
			1–15: remote module (0)
byLocalID	Expansion module ID	BYTE	0–31 (0)
usiChannelNo	Specified channel number	USINT	(0)
bPID_Run	Activate PID operation	BOOL	TRUE/FALSE (FALSE)
iSV	Target value	INT	–32768–32767 (0)
bPID_M_A	PID Auto/Manual mode	BOOL	TRUE: Manual. The MV will output the value according to the MOUT value. When PID_MODE is 1, this setting is invalid. FALSE: Auto. The MV will output the value according to the PID formula. (FALSE)
bMOUT_AUTO	Manual value (MOUT) automatic update mode	BOOL	TRUE: Auto. The MOUT value changes with the MV value. FALSE: Normal. The MOUT value does not change with the MV value. (FALSE)
uiAUTO_DBW*2	When it applies to Auto tuning, the SV ± dead band range does not operate.	USINT	0–32000 (0)
bPID_EQ	PID calculation formula selection	BOOL	TRUE/FALSE (FALSE)
bPID_DE	PID differential error calculation selection	BOOL	TRUE/FALSE (FALSE)
bPID_DIR	PID positive negative direction	BOOL	FALSE: heating operation (E=SV–PV) TRUE: cooling operation (E=PV–SV)
iERR_DBW	Deviation amount (E) ineffective range	INT	–32768–32767 (0)
iBIAS	Feedforward control output value	INT	–32768–32767 (0)

***Note**:

1. Only support mode 0.
2. If the PID parameter input value exceeds the parameter upper limit, only the upper limit value will be written to the module. If the PID parameter input value is below the parameter lower limit, only the lower limit value will be written to the module.

- **Inputs/Outputs**

Name	Function	Data Type	Setting/Output value range (Default value)
bUpdate	Update PID parameter flag	BOOL	TRUE/FALSE (FALSE) After *bEnable* is activated, if you want to change parameters, set parameters to new values, then set *bUpdate* to TRUE.

Name	Function	Data Type	Setting/Output value range (Default value)
			When the instruction is changed, the instruction will clear the *bUpdate* to FALSE.
byPIDMode	PID contol mode	BYTE	0: auto control function 1: auto adjust parameter function
rKc_Kp	scale coefficient (Kc or Kp, determin which coefficient to use accordingt o the *bPID_EQ* parameter)	REAL	Range of positive single-precision floating-point values
rTi_Ki	Integral coefficient (Ti or Ki, determin which coefficient to use accordingt o the *bPID_EQ* parameter)	REAL	Range of positive single-precision floating-point values (Unit: Ti = sec; Ki = 1/sec)
rTd_Kd	Differential coefficient (Td or Kd, determin which coefficient to use accordingt o the *bPID_EQ* parameter)	REAL	Range of positive single-precision floating-point values (Unit: sec)
rTf	Differential action time constant (Tf)	REAL	Range of positive single-precision floating-point values (Unit: sec)
iALPHA	Initial integral compensation parameter (heating)	INT	0–100(0) (Unit: 1%)
iBETA	Initial integral compensation parameter (cooling)	INT	0–100(0) (Unit: 1%)
uiMOUT*	MV manual value	UINT	0–1000(0)

***Note**: If the PID parameter input value exceeds the parameter upper limit, only the upper limit value will be written to the module. If the PID parameter input value is lower than the parameter lower limit, only the lower limit value will be written to the module.

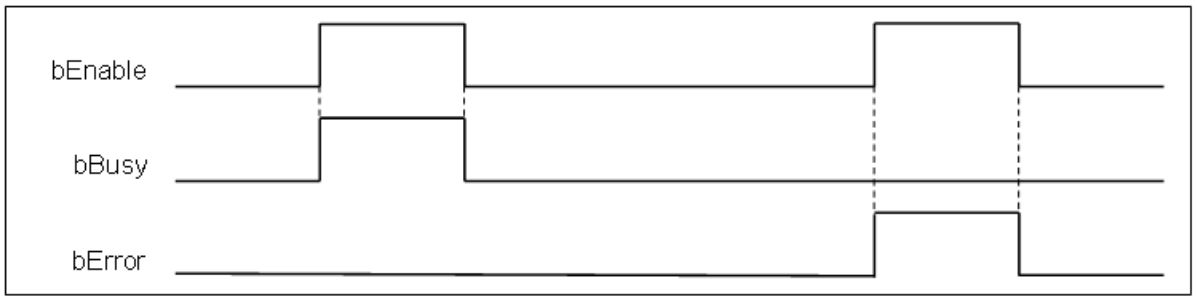
- **Outputs**

Name	Function	Data Type	Output Range (Default value)
bBusy	Instruction running flag	BOOL	TRUE/FALSE (FALSE)
lrMV	MV output value	LREAL	0.0–100.0(0) (Unit: 1%)
lrPV	Current value	LREAL	Positive/Negative number or 0 (0)
lrI_MV	Accumulated integral value	LREAL	Positive/Negative number or 0 (0)
bError	FB instruction error flag	BOOL	TRUE/FALSE (FALSE)
ErrorCode	Error code	DFB_AS_MODULE_API_ERROR	DFB_AS_MODULE_API_ERROR (DFB_NO_ERROR)

- **Output Updating Timing**

Name	Timing for shifting to TRUE	Timing for shifting to FALSE
bBusy	When *bEnable* shifts to TRUE	When *bEnable* shifts to FALSE.
lrMV	Continuously updating the record value when *bBusy* is TRUE	Clear to 0 when *bEnable* shifts to TRUE.
lrPV	Continuously updating the record value when *bBusy* is TRUE	Clear to 0 when *bEnable* shifts to TRUE.
lrI_MV	Continuously updating the record value when *bBusy* is TRUE	Continuously updating the record value when *bValid* is TRUE
bError	When an error occurs during the running or there are incorrect input values	When *bEnable* shifts to FALSE
ErrorCode		

- **Timing Diagram**



- **Function**

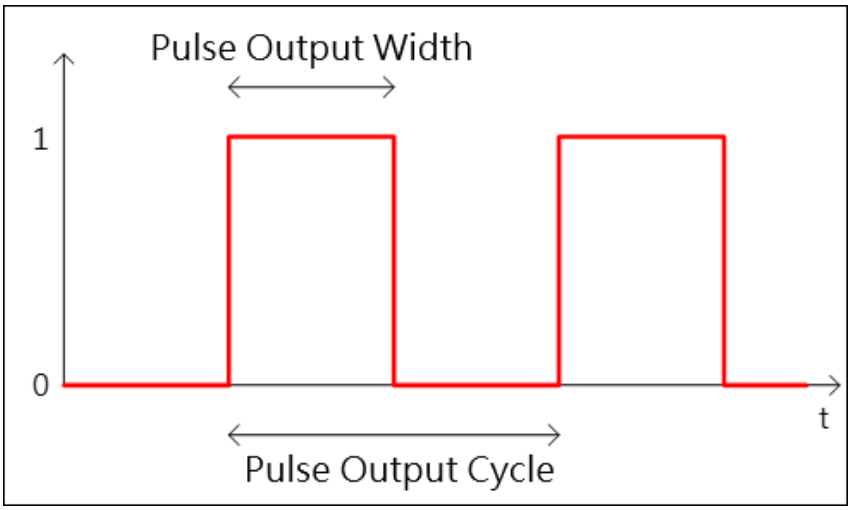
  - It is suggested that this instruction be placed under Main Task.
  - This function is supported only when AX-3 series firmware version is V1.0.1 and later.
  - The instruction is supported only by AS series temperature modules (supported version: AS04RTD–A V1.04 and later/AS06RTD–A V1.00 and later； AS04TC–A V1.04 and later /AS08TC–A V1.00 and later)
  - *byRemoteID* specifies that the analog input module should connect to the right of CPU or the right of the remote module group numbers. The CPU number is 0, the first remote module number is 1, and so on. The maximum group number is 15.
  - *byLocalID* specifies module numbers. The number of the first module on the right of CPU is 0, the number of the second module on the right of CPU is 1, and so on. Regardless of any type of modules, all modules must be counted. The maximum number of modules is 32.
  - *usiChannelNo* specifies the channel numbers. The number of channel one is 1, and the number of channel two is 2.
  - When PID_RUN is TRUE → FALSE, the *rMV* output value will be cleared to 0. If users want the *rMV* output value to be kept as the last *rMV* output value when PID is not working, make *bEnable* TRUE → FALSE to deactivate the instruction.
  - When *bUPDATE*'s *Enable* = TRUE, if you want to modify the parameters related to bPID_RUN–uiCYCLE, you need to set the *bUpdate* parameter to ON, and it will be automatically changed to FALSE after a cycle.
  - If the auto adjust parameter function is used (*byPID_MODE* =1), it will automatically enter the autocontrol mode when the adjustment is complete (*byPID_MODE* changes to 0 automatically), and

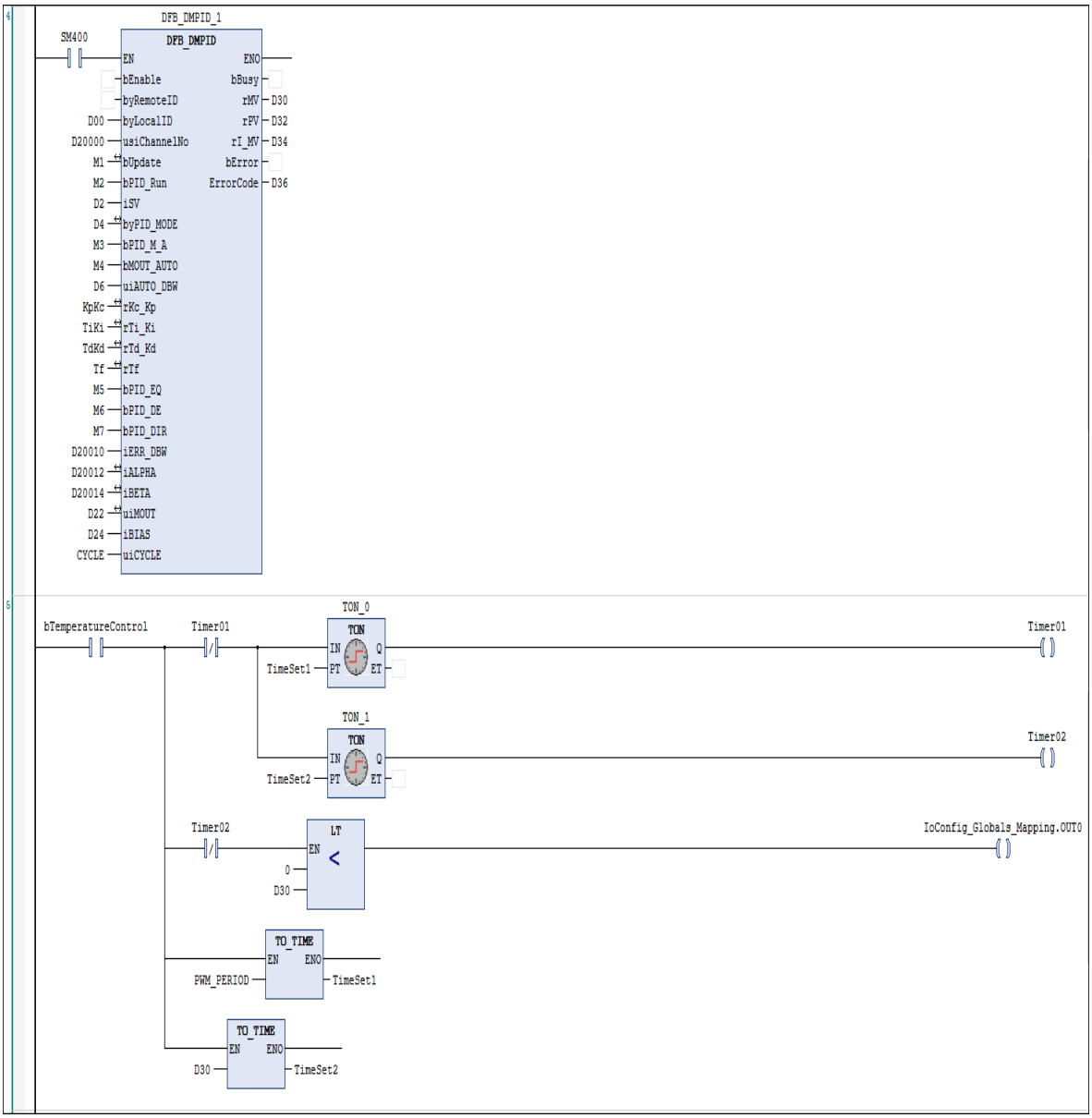
the debugged *rKc_Kp*, *rTi_Ki*, *rTd_Kd*, *rTf*, *iALPHA*, and *iBETA* parameters will be filled back to the corresponding parameter variables of the *DMPID* input. You can also use the outage persistence variables to meet the requirement of saving PID control parameters.

- **Programming Example**

1. When *bEnable*=TRUE, the instruction is running. When M2=TRUE, the DMPID instruction starts to operate. When M2=FALSE, MV value is 0 and is sent to D30. When *bEnable* is OFF, the instruction is not Run, and the parameter values in the original instruction does not change.
2. The *MV* output value range is 0.0–100.0. You need to convert the required control quantity per requirement. This application example converts to PWM output. Convert the *MV* output value to the pulse output width TimeSet2 with a range of 0%–100%, and then multiply the PWM cycle to convert to the time TimeSet1.
3. Input the pulse output width and pulse output cycle to the TON function block, PWM control can be realized in the specified pulse output device.

   Duty cycle = pulse output width/pulse output cycle





- **PID algorithm**:

  When the PID_MODE control mode is set to 0, it is the autocontrol mode. PID calculation formula is as follows:

- Independent Formula & Derivative of E (*bPID_EQ*=FALSE & *bPID_DE*=FALSE)

$$MV = K_P E + Ki\int_0^t Edt + K_d * \frac{dE}{dt} + BIAS$$

E = SV – PV or *E = PV – SV*

- Independent Formula & Derivative of PV (*bPID_EQ*=FALSE & *bPID_DE*=TRUE)

$$MV = K_P E + Ki\int_0^t Edt - K_d * \frac{dPV}{dt} + BIAS$$

E = SV – PV

Or

$$MV = K_P E + Ki\int_0^t Edt + K_d * \frac{dPV}{dt} + BIAS$$

E =PV – SV

- Dependent Formula & Derivative of E (*bPID_EQ*=TRUE & *bPID_DE*=FALSE)

$$MV = K_c\left[E + \frac{1}{T_i}\int_0^t Edt + T_d * \frac{dE}{dt}\right] + BIAS$$

E = SV – PV or *E = PV – SV*

- Dependent Formula & Derivative of PV (*bPID_EQ*=TRUE & *bPID_DE*=TRUE)

$$MV = K_c\left[E + \frac{1}{T_i}\int_0^t Edt - T_d * \frac{dE}{dt}\right] + BIAS$$

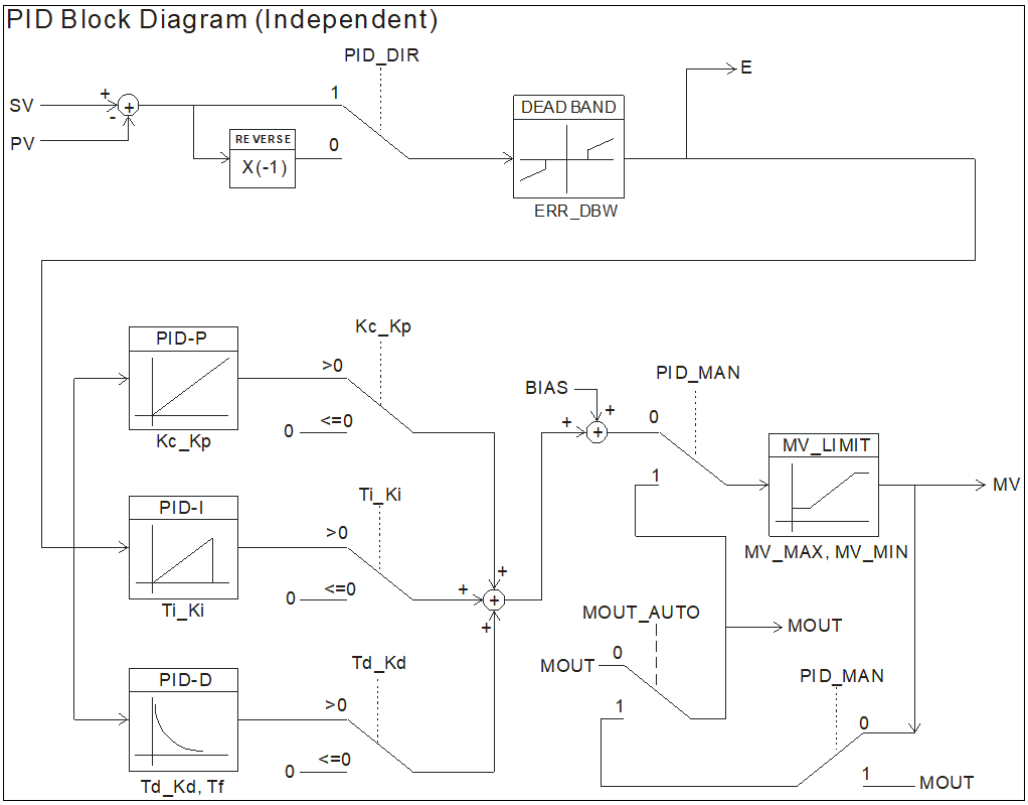
E = SV – PV

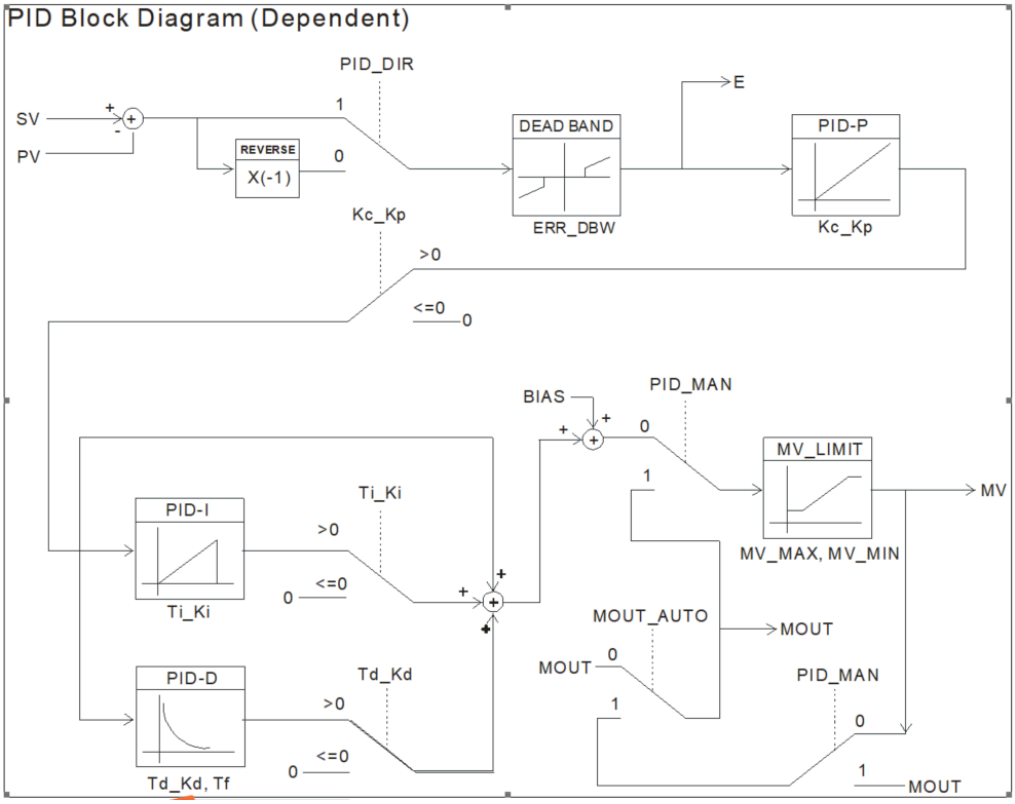
Or

$$MV = K_c\left[E + \frac{1}{T_i}\int_0^t Edt + T_d * \frac{dE}{dt}\right] + BIAS$$

E = PV – SV

- **PID Block Diagram**

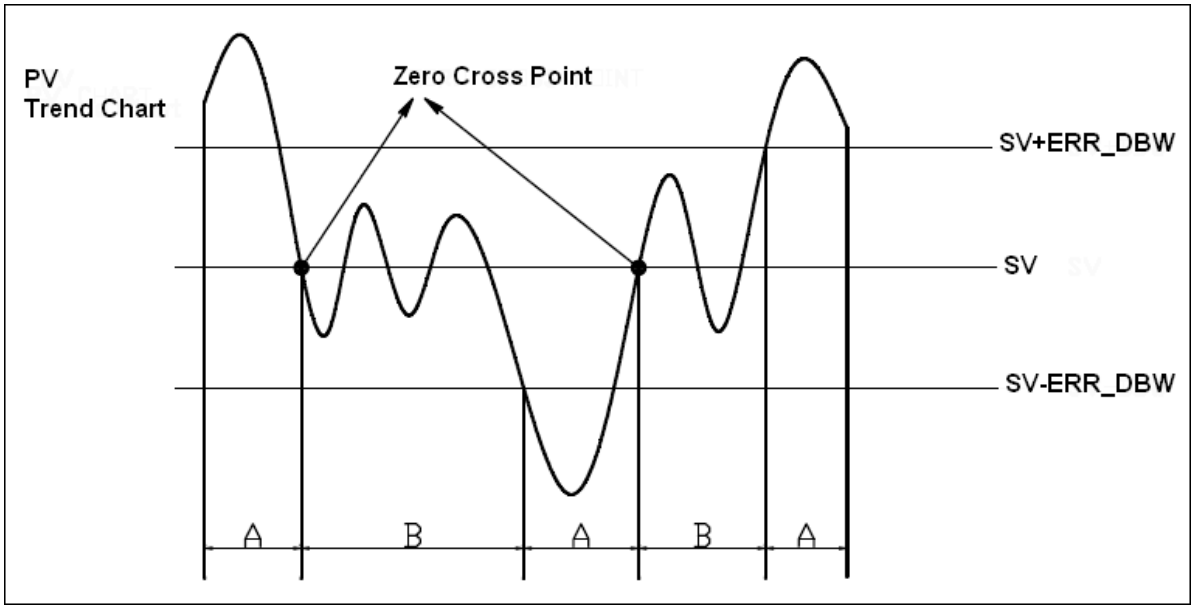




- **Deviation Ineffective Range**

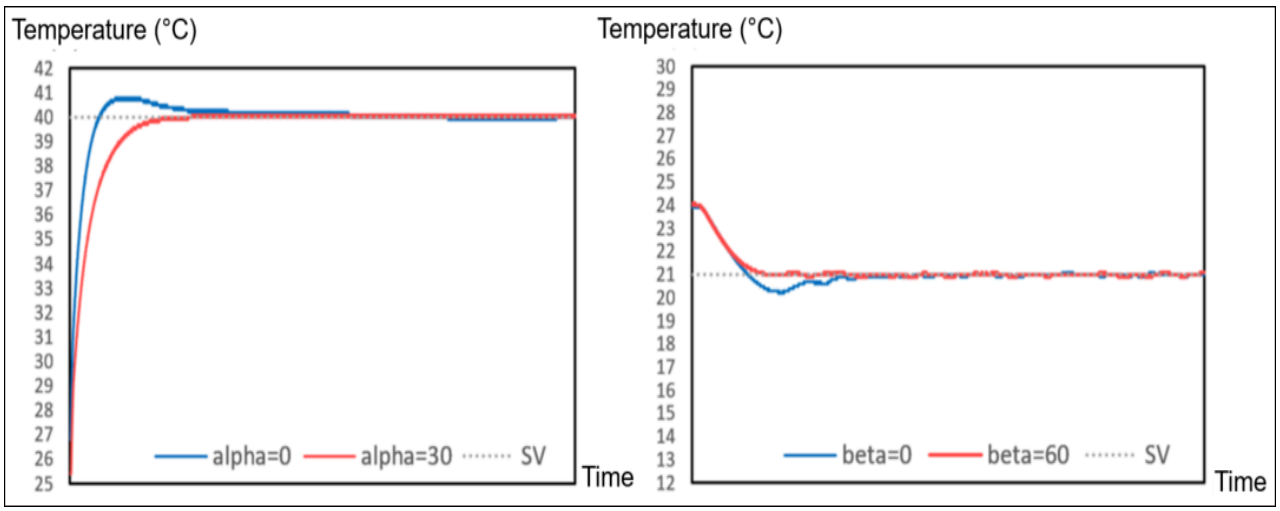
  When the PV value enters the *iERR_DBW* range, at first the CPU still performs the PID calculation according to the E value until PV crosses the SV value (Zero Cross Point), which means that Cross Status is established, and E = 0 will be substituted into the PID calculation. Then, when the PV value exceeds the *iERR_DBW* range, the E value will be resumed for the PID calculation. If *bPID_DE*=TRUE, it means that using the PV value to perform the differential calculation. After the Cross Status conditions are established, the CPU will take Delta PV as 0 to perform the PID differential calculation. (Delta PV= Current PV–previous PV).

For example, in the following PV trend diagram, CPU in A area will perform the normal PID calculation; however, CPU in B area will take E or Delta PV as 0 to perform the PID calculation.



- **α, β VALUE**

  ALPHA and BETA are used to compensate initial integral when PID starts and the SV target values changes, the aim of which is to reduce the overshoot phenomenon. As shown in the following figures, ALPHA parameter is used to slow down the rising overshoot; BETA parameter is used to slow down the falling overshoot.



- **Notes and recommendations**

  - When you adjust three main parameters, *Kc_Kp*, *Ti_Ki*, and *Td_Kd* (**PID_MODE**=0), adjust the *Kc_Kp* value first (based on the experience), and then set *Ti_Ki* and *Td_Kd* values to 0. When the adjustment is generally controllable, then adjust the Ti_Ki value (small → large) and the Td_Kd value (large → small) in sequence. When *Kc_Kp* = 1, it represents 100%, that is, the gain of the deviation value is 1. A value less than 100% will attenuate the deviation value; a value more than 100% will increase the deviation value.

  - The automatically adjusted parameters may not be applicable to each control environment, so you need to modify the adjusted parameters. It is recommended that only the *Ti_Ki* or *Td_Kd* value be modified.

  - The CYCLE parameter is the period that the PID calculates once and update the output value (MV).

  - Note that when the number of channels opened for measurement changes, the update time of the measurement value will change (for example, when opening only one channel for measurement,

the measurement value will be updated every 200 ms. When opening the other three channels for measurement, the measurement value will be updated every 800ms). The *Kc_Kp*, *Ti_Ki*, *Td_Kd* parameters may not be applicable.

- **Supported Devices**
  - AX-3 series
- **Library**
  - DL_ASModuleAPI_AX3.library

6.14. DFB_DHCCNT

DFB_DHCCNT: Starts or stops the counter and sets or modifys counter value. This instruction is only applicable for AS02HC-A.

FB/FC	Instruction	Graphic Expression	ST Language
FB	DFB_DHCCNT		DFB_DHCCNT(bEnable:= , byLocalID:= , usiChannelNo:= , bUpdate:= , usiAction:= , diActionValue:= , bBusy=> , diCurCnt=> , byCurSSI_SingleTurn=> , byCurSSI_MultiTurn=> , wSSIStatus=> , wRefCnt=> , bDir=> , uiCntStat=> , bError=> , ErrorCode=>);

- **Inputs**

Name	Function	Data Type	Setting Value (Default value)
bEnable	Run the function block	BOOL	TRUE/FALSE (FALSE)
byLocalID	Expand the module number	BYTE	0–31 (0)
usiChannelNo	Specify channel number	USINT	1–2 (1)
usiAction*	Counter Action mode setting	USINT	0–7 (0)
diActionValue	New counter value/offset	DINT	–32768–32767 (0)

***Note**: *usiAction* is the action mode setting of the counter, and *diActionValue* usage descriptions are as below:

usiAction	Function	Description
0	Keep the current mode and not to change.	–
1	Set/Change the current counter value *diCurCnt* to *diActionValue*	When setting the new counter value, set *diActionValue* value. **Note**: When using SSI encoder and the counter form is set as Absolute Position, this *usiAction* is invalid.
2	Set SSI encoder offset as *diActionValue*	When the counter form is set as Absolute Position, users can set the offset of the SSI encoder counter value. Set the offset through

usiAction	Function	Description
		diActionValue, and its counter value *diCurCnt* = encoder original counter value + *diActionValue*. **Note**: 1. *diActionValue* range: $-2^{(MT+ST^{length})}$ < *diActionValue* < $2^{(MT+ST^{length})}$ Setting is not allowed when it is out of range. 2. *diActionValue*range: $-2^{(MT+ST^{length})}$ < *diActionValue* < $2^{(MT+ST^{length})}$. Setting is not allowed when it is out of range. 3. When using SSI encoder and the counter form is set as Ring counter, this *diActionValue* is invalid. When the device area HC module configuration is re–downloaded, the offset will be cleared to 0.
3	Set/Change SSI encoder absolute position value to *diActionValue*	When the counter form is set as Absolute Position, the SSI encoder counter value will be automatically shifted to the *diActionValue* vale, and its counter value *diCurCnt* = *diActionValue*. **Note**: 1. *diActionValue* range: 0 < *diActionValue* < $2^{(MT+ST^{length})}$. Setting is not allowed when it is out of range. 2. When using SSI encoder and the counter form is set as Ring counter, this *usiAction* is invalid. Use *usiAction*=1 to change the counter value. 3. When the device configuration is re–downloaded, the offset will be cleared to 0.
4	Reset the current counter value *diCurCnt*	1. Reset *diCurCnt* as 0. 2. Reset *iCurrentNo* of table compare output instruction DFB_DHCCMPT. 3. Reset the *bMatch1* and bMatch2 flags of compare output instruction DFB_DHCCMP. **Note**: When using SSI encoder, and the counter form is set as Absolute Position, the counter value cannot be reset. However, the *iCurrentNo*, *bMatch1*, and *bMatch2* flags will be reset.
5	Reset *diCurCnt* current counter value *diCurCnt*, and reset table compare output instructions/specified Y output point of compare output instructions.	In addition to the above *usiAction*=4 reset content, compare output instructions/specified Y output point of compare output instructions (ON → OFF) are also reset.
6	Preset the current counter value *diCurCnt*	1. *diCurCnt* is specified to be modified to *diActionValue*. 2. Reset *iCurrentNo* of table compare output instruction DFB_DHCCMPT. Note that after Preset, DFB_DHCCMPT will wait the comparison to arrive from the first compare value; therefore, if the counter value is bigger than the first compare value after Preset, the table comparison cannot

usiAction	Function	Description
		be done correctly. Run Preset, then set the *bUpdate* flag of DFB_DHCCMPT to ON. 3. Reset the *bMatch1/2* flag of compare output instruction DFB_DHCCMP. **Note**: When using SSI encoder, and the counter form is set as Absolute Position, the counter value cannot be preset, but the *iCurrentNo*, *bMatch1*, and *bMatch2* flags will be reset.
7	Preset the current counter value *diCurCnt*, and reset table compare output instruction/ specified Y output point of compare output instructions.	In addition to the above *usiAction*=6 reset content, table compare output instructions/specified Y output point of compare output instructions (ON → OFF) are also reset.

- **Inputs/Outputs**

Name	Function	Data Type	Setting/Output value range (Default value)
bUpdate	Update DFB_DHCCNT parameter flag	BOOL	TRUE/FALSE (FALSE)

- **Outputs**

Name	Function	Data Type	Setting/Output value range (Default value)
bBusy	Instruction running flag	BOOL	TRUE/FALSE (FALSE)
diCurCnt	Specify the current counter value of module count channels	DINT	Positive number, negative number or 0 (0)
byCurSSI_SingleTurn	Absolute SSI encoder Single–Turn Data	BYTE	Positive number or 0 (0)
byCurSSI_MultiTurn	Absolute SSI encoder Multi–Turn Data	BYTE	Positive number or 0 (0)
wSSIstatus	Absolute SSI encoder status information	WORD	Positive number or 0 (0)
wRefCnt	Absolute SSI encoder data refresh counter	WORD	Positive number or 0 (0)
bDir	Counting direction display	BOOL	TRUE: Positive direction FALSE: Negative direction

Name	Function	Data Type	Setting/Output value range (Default value)
			(FALSE)
uiCntStat*	Counter status	UINT	Positive number or 0 (0)
bError	Instruction error flag	BOOL	TRUE/FALSE (FALSE)
ErrorCode	Error code	DFB_AS_MODULE_API_ERROR	DFB_AS_MODULE_API_ERROR (DFB_NO_ERROR)

Note: *uiCntStat* counts module action status, and its usage descriptions are as below:

Bit Number	Status Name	Status Description	Note
15–11	Reserved	Reserved	
10	SSI absolute position across zero point	0: Normal 1: Abnormal	The device configuration area starts or shuts detection, and the default is no detection. When the counter type is Absolute Position: When users have set the offset, the zero-crossing detection will be performed with the position value after the offset is adjusted. When the counter type is Ring Counter: Because the offset will be reset to 0, the zero-crossing detection will be performed with the encoder original position value. Cause of error: zero-crossing detection occurs. Clear method: DFB_DHCCNT performs resetting or presetting instructions, or the external input point performs resetting.
9	SSI communication anomaly	0: Normal 1: Abnormal	Cause of error: communication anomaly 5 times in a row. Clear method: communication is restored to normal.
8	SSI parity check error	0: Normal 1: Abnormal	Cause of error: parity check error Clear method: next reading value parity check is correct.
7	SSI data change size exceeds protection setting	0: Normal 1: Abnormal	Cause of error: The change size of two position data exceeds protection setting. Clear method: Next data change size is within a reasonable range.
6	Reserved	Reserved	
5	Ring counting Overflow	0: Normal 1: Abnormal	The device configuration area starts or shuts the detection, and the default is not to detect the cause of error: when hardware counter Overflow (> $2^{*32}-1$) or Underflow (< $-2^{*32}-1$) occurs, there will be no more counting. The counting will be restored after running the following: The DFB_DHCCNT instruction performs resetting or presetting instructions, or the external input point performs resetting.
4	Ring counting Underflow	0: Normal 1: Abnormal	

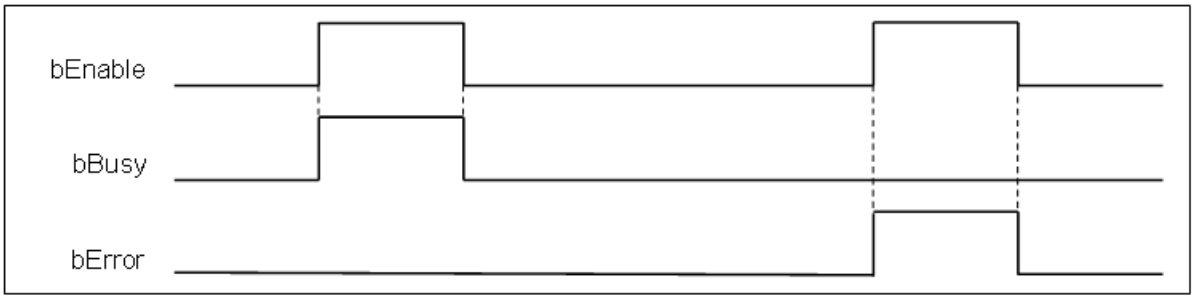
Bit Number	Status Name	Status Description	Note
3	Linear count is higher than the set limit	0: Normal 1: Abnormal	Cause of error: Linear count exceeds the user set range. Clear method: Linear counter value is back to the upper and lower limit. **Note**: When count exceeds the upper/lower limit, the counter value remains at the upper/lower limit. When counter value is back to the upper/lower limit range, the counting resumes.
2	Linear count is lower than the set limit	0: Normal 1: Abnormal	
1	Linear count Overflow	0: Normal 1: Abnormal	Cause of error: When linear count exceeds upper/lower limit, hardware counter continues counting. When hardware counter Overflow(> 2^{*32} − 1) or Underflow (< $–2^{*32}$ − 1) occurs, there will be no more counting. The counting will be restored after running the following: The DFB_DHCCNT instruction performs resetting or presetting instructions, or the external input point performs resetting.
0	Linear count Underflow	0: Normal 1: Abnormal	

- **Output Updating Timing**

Name	Timing for shifting to TRUE	Timing for shifting to FALSE
bBusy	When *bEnable* shifts to true and the function is enabled in the module	When *bEnable* shifts to false and the function is disabled in the module
diCurCnt	Continuously update the record when *bBusy* is TRUE.	–
byCurSSI_SingleTurn	Continuously update the record when *bBusy* is TRUE, and the counter form is Absolute Position.	–
byCurSSI_MultiTurn	Continuously update the record when *bBusy* is TRUE, and the counter form is Absolute Position.	–
wSSIstatus	Continuously update the record when *bBusy* is TRUE, and the counter form is Absolute Position.	–
wRefCnt	Continuously update the record when *bBusy* is TRUE, and the counter form is Absolute Position.	–
bDir	• Continuously update the record when *bBusy* is TRUE. • TRUE when the count direction is positive	FALSE when the count direction is negative

Name	Timing for shifting to TRUE	Timing for shifting to FALSE
uiCntStat	Continuously update the record when *bBusy* is TRUE.	• Clear the error when using *usiAction* clearance function. • When the status of the error counting module is cleared
bError	When an error occurs during the running or there are incorrect input values	When *bEnable* shifts to false
ErrorCode		

- **Timing Diagram**

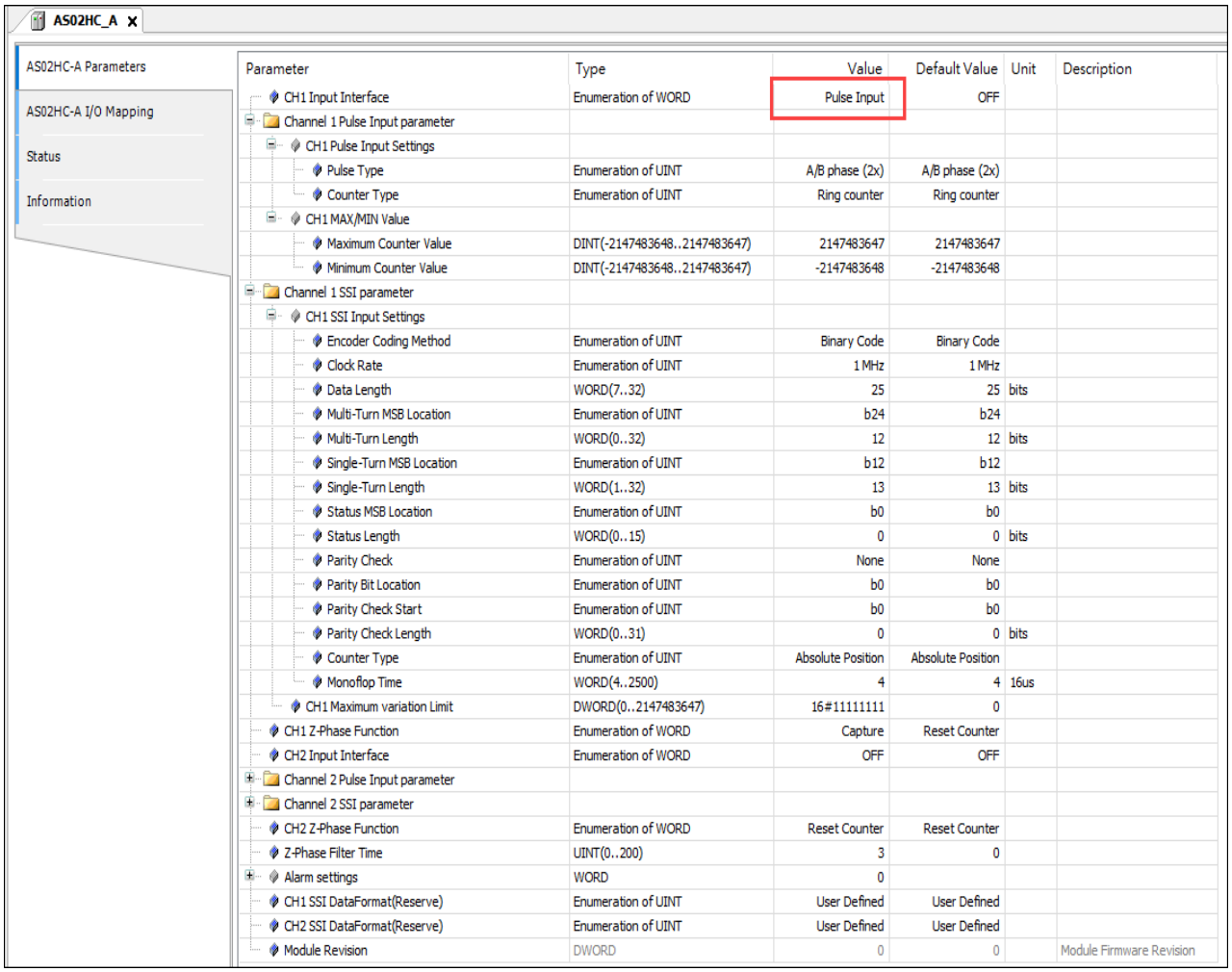


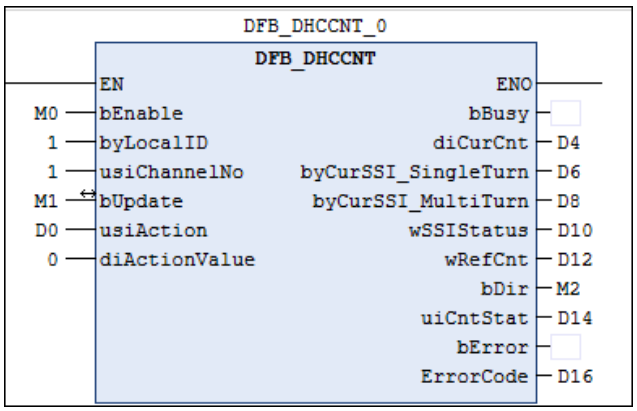
- **Function**
  - It is suggested that this instruction be placed under Main Task.
  - This function is supported with AX-3 firmware V1.0.2 and later.
  - This instruction is only supported by AS Series count module (The supported version is AS02HC-A V1.00 and above).
  - *byLocalID* specifies module numbers. The number of the first module on the right of CPU is 0, the number of the second module on the right of CPU is 1, and so on. Regardless of any type of modules, all modules must be counted. The maximum number of modules is 32.
  - *usiChannelNo* specifies the channel numbers. The number of channel one is 1, and the number of channel two is 2.
  - Complete the *usiAction* and *diActionValue* settings before running this instruction. When *bEnable* is started for the first time, *usiAction* and *usiAction* of HC module will be set once. When *usiAction* and *diActionValue* need to be re-changed during running, set *usiAction* and *diActionValue* as new values, and then set the *bUpdate* flag to On. When this instruction completes changes, the instruction will clear *bUpdate* as Off itself.
  - *diCurCnt* is to display the current counter value of the specified module count channel.
  - *byCurSSI_SingleTurn* is absolute SSI encoder Single-Turn Data display (When the counter form is set as Ring Position, *byCurSSI_SingleTurn* shows the value after deviation; when setting the counter as Ring counter, *byCurSSI_SingleTurn* will show the original SSI encoder value), when the channel mode is set as incremental encoder, the *byCurSSI_SingleTurn* will be 0.
  - *byCurSSI_MultiTurn* is absolute SSI encoder Multi-Turn Data display (When the counter form is set as Absolute Position, *byCurSSI_MultiTurn* shows the value after deviation; when setting the counter as Ring counter, *byCurSSI_MultiTurn* will show the original SSI encoder value), when the channel mode is set as incremental encoder, the *byCurSSI_MultiTurn* will be 0.

- *WSSIstatus* is absolute SSI encoder Status Data display. When the channel mode is set as incremental encoder, *wSSIstatus* will be 0.
- *wRefCnt* is absolute SSI encoder data refresh counter, and its length is 16 bits. When new SSI data is captured, refresh counter will be incremented by one, and when this counter overflows, it will re-accumulate from 0. When the channel mode is set as incremental encoder, *wRefCnt* will be 0.
- *bDir* is the count direction display. When it shows On, it represents going in positive direction; when it shows Off, it represents going in negative direction.
- If the instruction is closed, the specify channel will stop updating the values of the right half of the instruction.

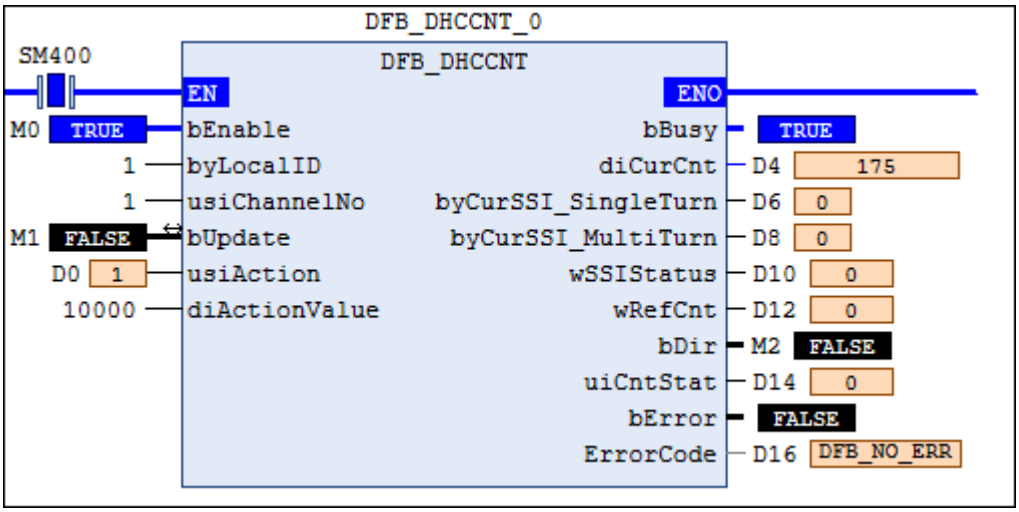
- **Programming Example**

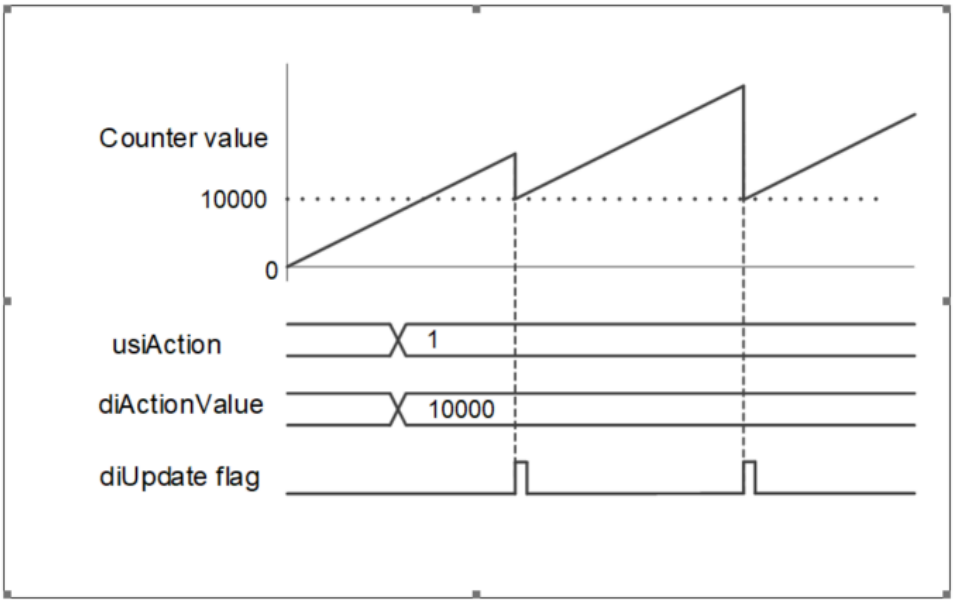
This example uses the FB instruction (DFB_DHCCNT) to read the first channel in the right module (AS02HC) of the host, and after setting the count parameter to the module, start updating counter value to a new variable (D4) through the function block.





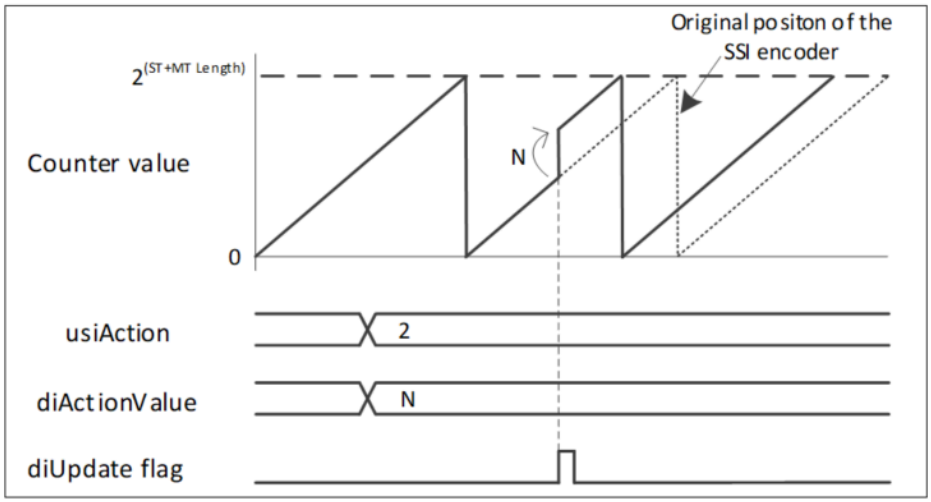
- **Example 1**: Set/Change the current counter value. (*usiAction* = 1)
  1. When setting M0 = ON, Counter starts to count.
  2. When setting *usiAction* parameter of the DHCCNT instruction as 1 and setting *diActionValue* parameter as 10000; when users set the *bUpdate* flag M1=ON, the current counter value CurNo will be changed to 10000.
  3. When setting *bUpdate* flag is complete, M1 will be cleared as OFF.





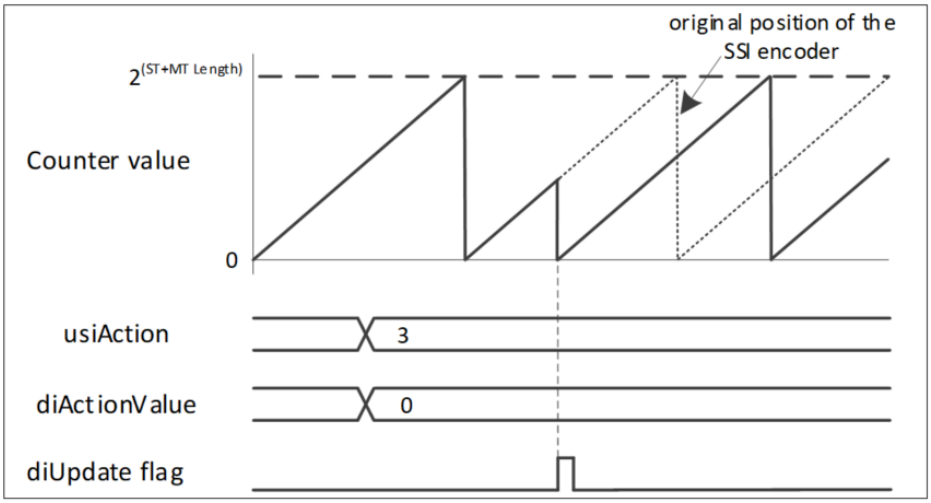
- **Example 2** – set absolute SSI encoder offset (*usiAction* = 2)
  1. Module device parameter sets the channel 1 as SSI Input, and select the counter form as Absolute Position.
  2. When M0=ON, counter starts to count.
  3. Set the *usiAction* parameter of the DHCCNT instruction as 2, and set *diActionValue* parameter as 500. Suppose the current counter value *diCurCnt* is 2500, and users set the *bUpdate* flag M1=ON, now the counter value will be changed to 3000.

4. After setting the *bUpdate* flag is complete, M1 will be clear to OFF automatically.



- **Example 3**: Set/Change SSI encoder absolute position value (Action = 3)

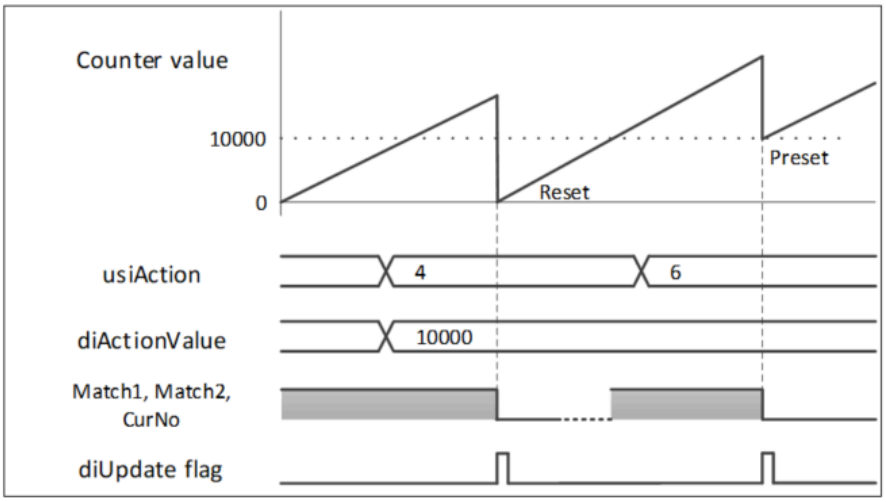
1. Module device parameter sets channel 1 as SSI Input, and select the counter form as Absolute Position.
2. When setting M0=ON, counter starts to count.
3. Set the *usiAction* parameter of the DHCCNT instruction to 3, and set the *diActionValue* parameter to 0. When users set the *bUpdate* flag M1=ON, the HC module will calculate the deviation amount automatically, and shift current counter value *diCurCnt* (encoder absolute position) to 0.
4. When setting the *bUpdate* flag is complete, M1 will be cleared to OFF.



- **Example 4**: Reset or preset the current counter value (*usiAction* = 4、6)

1. When setting M0=ON, counter starts to count.
2. Set the *usiAction* parameter of the DHCCNT instruction as 4. When users set the *bUpdate* flag M1=ON, the current counter value CurNo will be reset to 0. When setting the update flag is complete, M1 will be reset as OFF.
3. Set the *usiAction* parameter of the DHCCNT instruction to 6, and set the *diActionValue* parameter to 10000. When you set the *bUpdate* flag to M1=ON, the current counter value

diCurCnt will be preset as 10000. When setting the *bUpdate* flag is complete, M1 will be reset as OFF.



- **Supported Devices**
  - AX-3 series
- **Library**
  - DL_ASModuleAPI_AX3.library

6.15. DFB_DHCCAP

DFB_DHCCAP starts or stops the capture. This instruction is only applicable for HC counting.

FB/FC	Instruction	Graphic Expression	ST Language
FB	DFB_DHCCAP		DFB_DHCCAP(bEnable:= , byLocalID:= , usiChannelNo:= , byTrgSel:= , bBusy=> , diCapValue1=> , bCapValue1_Complete=> , diCapValue2 => , bCapValue2_Complete=> , bError=> , ErrorCode=>);

- **Inputs**

Name	Function	Data Type	Setting Value (Default value)
bEnable	Run the function block	BOOL	TRUE/FALSE (FALSE)
byLocalID	Expand the module number	BYTE	0–31(0)
usiChannelNo	Specify the channel number	USINT	1–2(1)
byTrgSel*	TRIGGER method selection	BYTE	0–1(0)

***Note**: *byTrgSel* is the trigger method selection captured by counter value. Its usage is as follows:

byTrgSel	Function	Note
0	Triggered by the digital input of the specified channel.	Capture function is selected for the external input point function with the HC device configuration.
1	Triggered by *bMatch1* and bMatch2 of another channel compare output instruction.	Example: When channel 2 starts DFB_DHCCAP and sets *byTrgSel*=1, and when instruction channel 1 is used for compare output instruction DFB_DHCCMP, when channel 1 compare arrives *bMatch1* and bMatch2, the current counter values captured by channel 2 will be triggered and saved in *diCapValue1*or *diCapValue2* . **Note**: When *byTrgSel*=1, even if the external input point function of the HC device configuration is selected as Capture, the external input point capture function will be considered invalid.

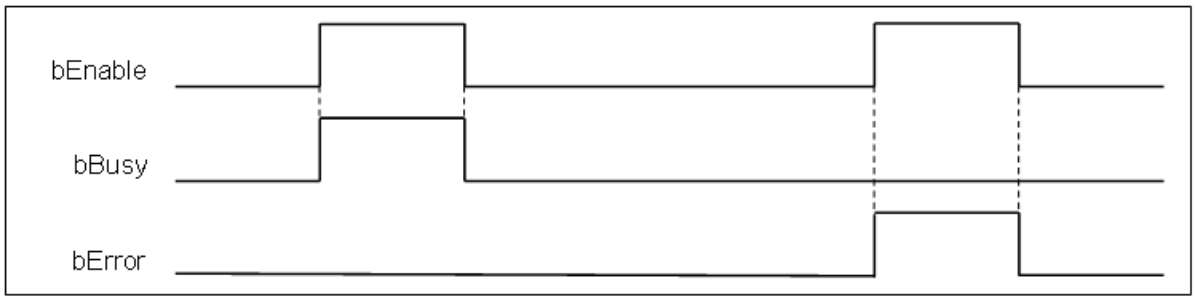
- **Outputs**

Name	Function	Data Type	Output Value Range (Default)
bBusy	Instruction running flag	BOOL	TRUE/FALSE (FALSE)
diCapValue1	Capture counter value1	DINT	Positive number, negative number or 0(0)
bCapValue1_Complete	*diCapValue1* capture completion flag	BOOL	TRUE/FALSE (FALSE)
diCapValue2	Capture counter value2	DINT	Positive number, negative number or 0 (0)
bCapValue2_Complete	*diCapValue2* capture completion flag	BOOL	TRUE/FALSE (FALSE)
bError	Instruction error flag	BOOL	TRUE/FALSE (FALSE)
ErrorCode	Error code	DFB_AS_MODULE_API_ERROR	DFB_AS_MODULE_API_ERROR (DFB_NO_ERROR)

- **Output Updating Timing**

Name	Timing for shifting to TRUE	Timing for shifting to FALSE
bBusy	When *bEnable* shifts to true and the function is enabled in the module	When *bEnable* shifts to false and the function is disabled in the module
diCapValue1	When *bBusy* is true and update the record when the capture signal is established.	–
bCapValue1_Complete	When *bBusy* is true and the capture value is updated to *diCapValue1*.	When *bEnable* shifts to false Cleared by users
diCapValue2	When *bBusy* is TRUE and updated the record when the capture signal is established.	–
bCapValue2_Complete	When *bBusy* is true and the capture value is updated to *diCapValue2*	• When *bEnable* shifts to false • Cleared by users
bError	When an error occurs during the running or there are incorrect input values	When *bEnable* shifts to false
ErrorCode		

- **Timing Diagram**

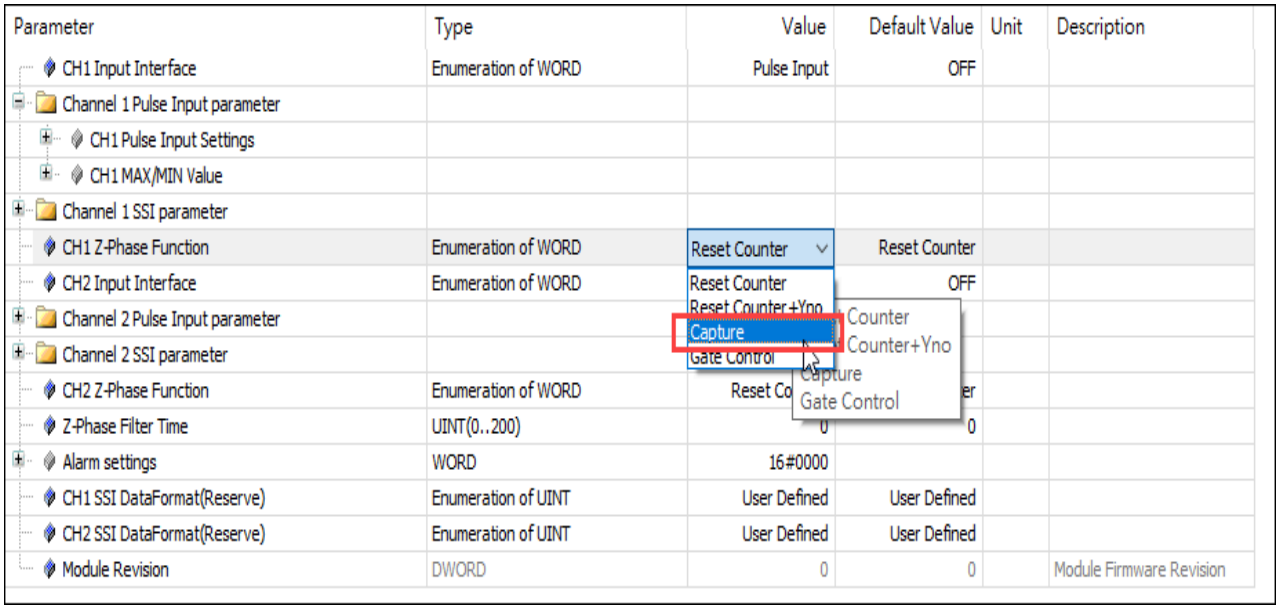


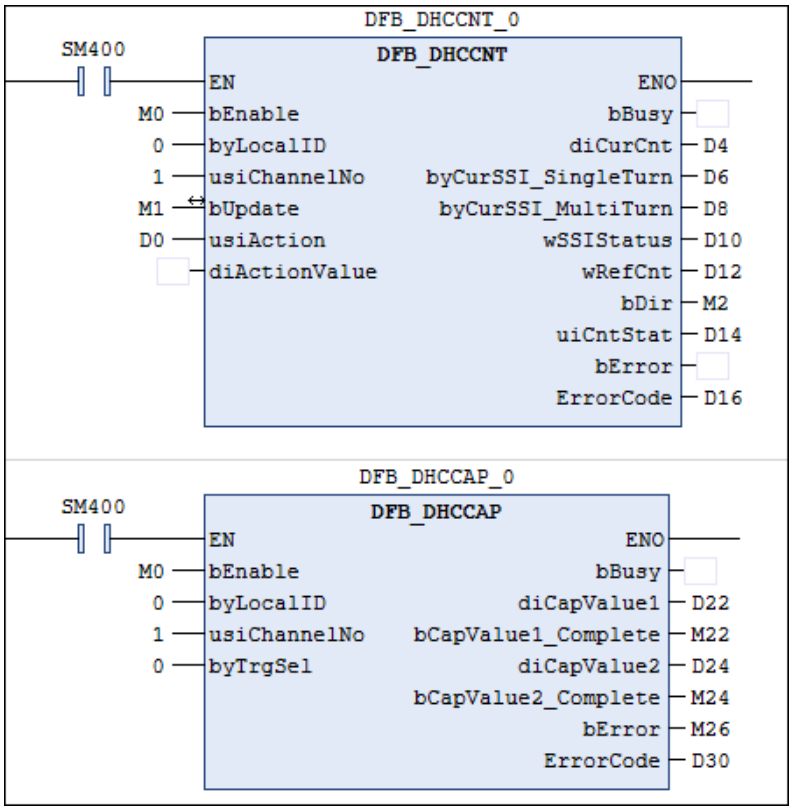
- **Function**

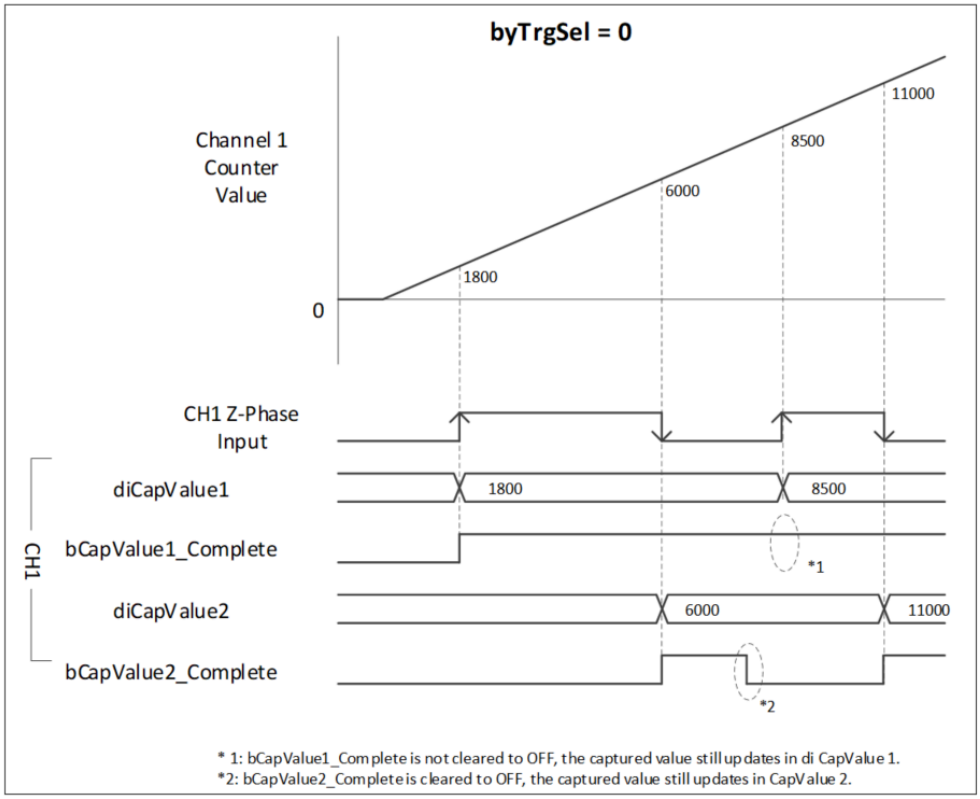
  - It is suggested that this instruction be placed under Main Task.
  - This function is supported with AX-3 firmware V1.0.2 and later.
  - This instruction is only supported by AS Series count module (The supported version is AS02HC-A V1.00 and above).
  - DFB_DHCCAP needs to be used with the DFN_DHCCNT instruction. Only when DFB_DHCCNT is enabled, the counter value will count according to the input signals, and the counter value captured by DFB_DHCCAP is valid. When DFB_DHCCNT is off, the counter value will stop receiving input signals and stop updating counter vale. At this time, the captured counter value will not change.
  - *byLocalID* specifies module numbers. The number of the first module on the right of the controller is 0, the number of the second module on the right of the controller is 1, and so on. All modules must be counted. The maximum number of modules is 32.
  - *usiChannelNo* specifies the channel numbers. The number of channel one is 1, and the number of channel two is 2.
  - Complete *byTrgSel* setting before running this instruction. When the En instruction starts for the first time, *byTrgSel* will be set once for the HC module. During running, if *byTrgSel* needs to be re–changed, disable the instruction and start again.
  - *diCapValue1* and *diCapValue2* are capture counter value 1 and capture counter value 2 respectively. When *byTrgSel*=0, *diCapValue1* is the counter value stored in external input point rising edge, and *diCapValue2* is the counter value stored in external input point falling edge. When *byTrgSel*=1, *diCapValue1* is the counter value that stored in another channel compare input instruction *bMatch1* from Off→On, and *diCapValue2* is the counter value that stored in another channel compare input instruction bMatch2 from Off→On.
  - *bCapValue1_Complete* and *bCapValue2_Complete* are the flags of capture completion counter value1 and capture completion counter value2. When *bCapValue1_Complete* or *bCapValue2_Complete* is Off–>On, it means that *diCapValue1* or *diCapValue2* are already the latest capture values, and users need to clear the *bCapValue1_Complete* or *bCapValue2_Complete* flags after reading capture values. When next *bCapValue1_Complete* and *bCapValue2_Complete* are Off–>On, there are new capture values. If users do not clear the *bCapValue1_Complete* and *bCapValue2_Complete* flags, the module latest capture values will keep updating until *diCapValue1* and *diCapValue2*.
  - If *bEnable* is from On to Off, it means that disabling the instruction Capture function. At this time, *diCapValue2* and *diCapValue2* content values will remain the same and will not be updated. However, *bCapValue1_Complete* and *bCapValue2_Complete* flags will be cleared.

- **Programming Example**

This example uses the FB instruction (DFB_DHCCNT) to start the first channel counting function in the right module (AS02HC) of the host, and use the FB instruction (DFB_DHCCAP) to capture the counter value.

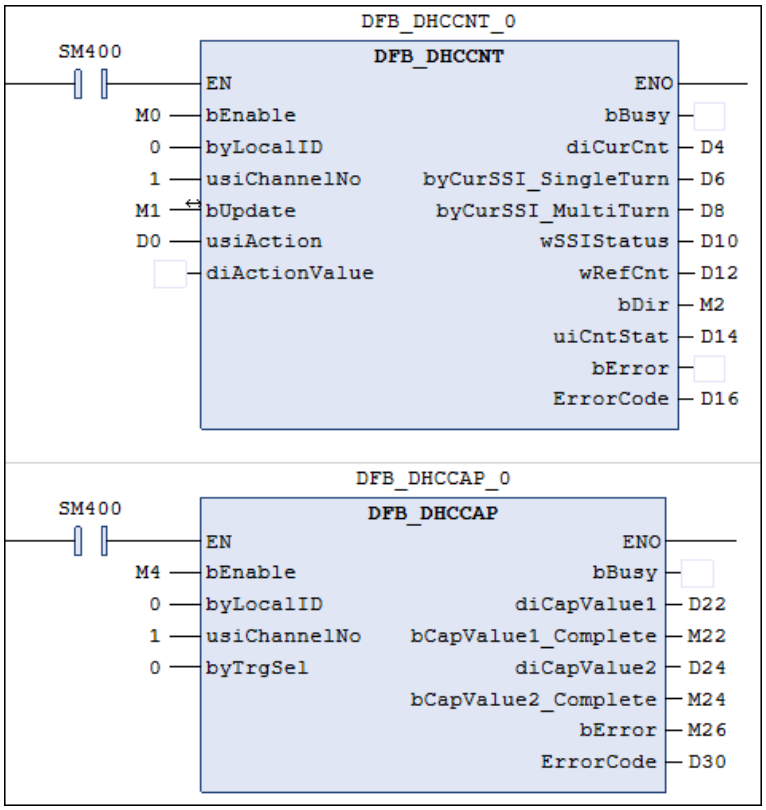


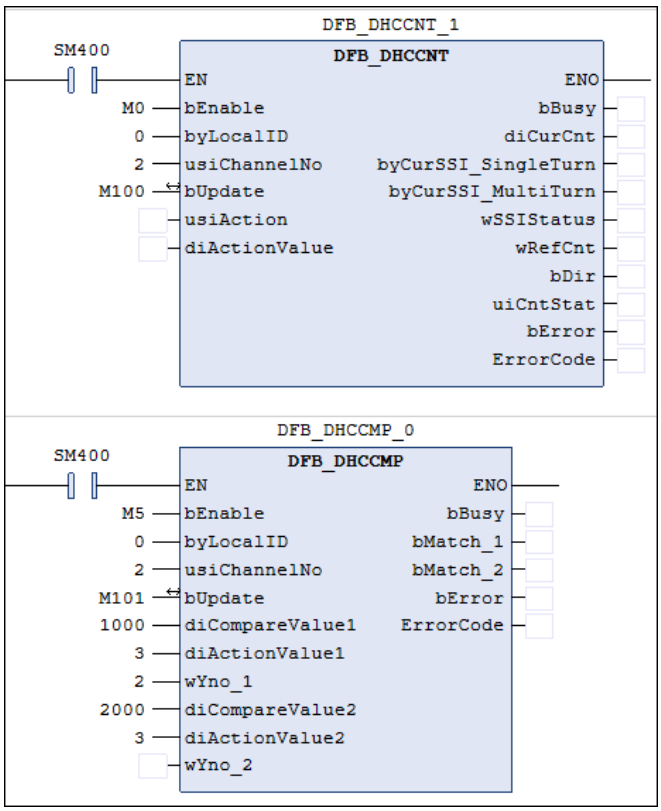




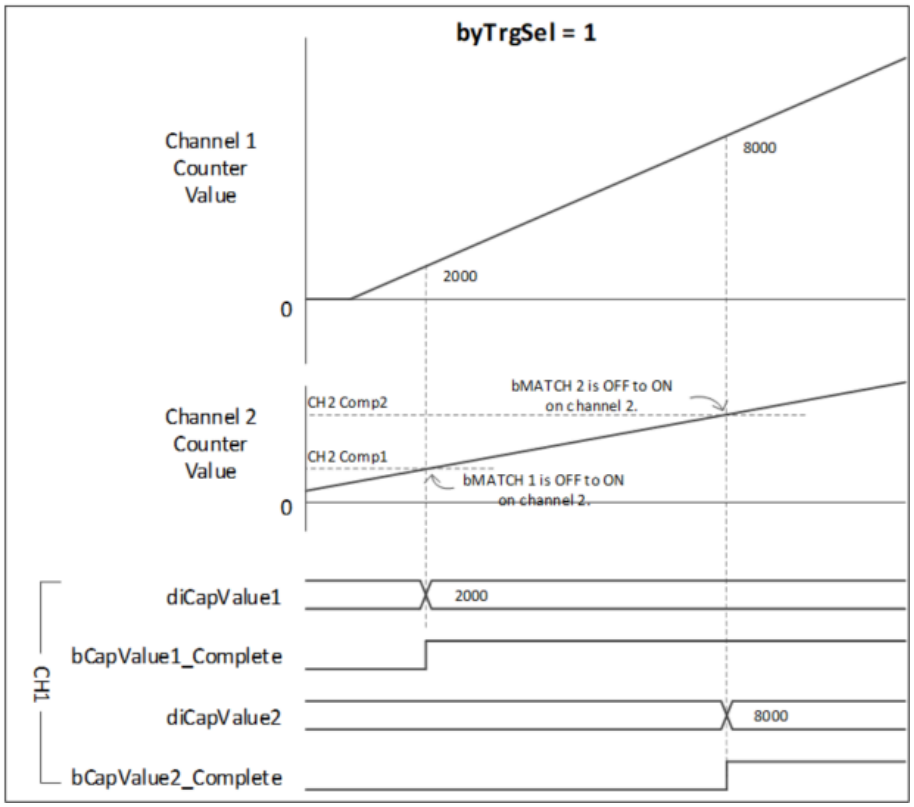
- **Programming example: Another channel DHCCMP compares arrival and captures the counter value.**
  1. Set the *byTrgSel*parameter of the DHCCAP instruction in channel 1. When M0=ON, the DHCCAP instruction will starts to wait for compare arrival of the other channel (channel 2).
  2. When channel 2 counter value arrives compare value *diCompareValue1*, and at this time channel 1 counter value 2000 will immediately output to *diCapValue1*, and the *bCapValue1_Complete* flag will be set to ON.
  3. When channel 2 counter value arrives compare value *diCompareValue2*, and at this time counter value 8000 will be immediately output to *diCapValue2*, and the *bCapValue2_Complete* flag will be set to ON.

4. Same as Example 1, even if the *bCapValue1_Complete* and *bCapValue2_Complete* flags are not cleared to OFF, the new captured values will still output to *diCapValue1 and diCapValue2* when channel 2 has a compare arrival event again.





- **Timing diagram**



- **Supported Devices**
  - AX-3 series
- **Library**
  - DL_ASModuleAPI_AX3.library

6.16. DFB_HCDO

DFB_HCDO controls the HC module output point.

FB/FC	Instruction	Graphic Expression	ST Language
FB	DFB_HCDO		DFB_HCDO(bEnable:= , byLocalID:= , bUpdate:= , iOutputSetting:= , bBusy=> , iOutputState=> , bError=> , ErrorCode=>);

- **Inputs**

Name	Function	Data Type	Setting Value (Default value)
bEnable	Run function block	BOOL	TRUE/FALSE (FALSE)
byLocalID	Expand module number	BYTE	0–31 (0)
bUpdate	Update parameter	BOOL	TRUE/FALSE (FALSE)
iOutputSetting	Output point motion setting	INT	0–15 (0)

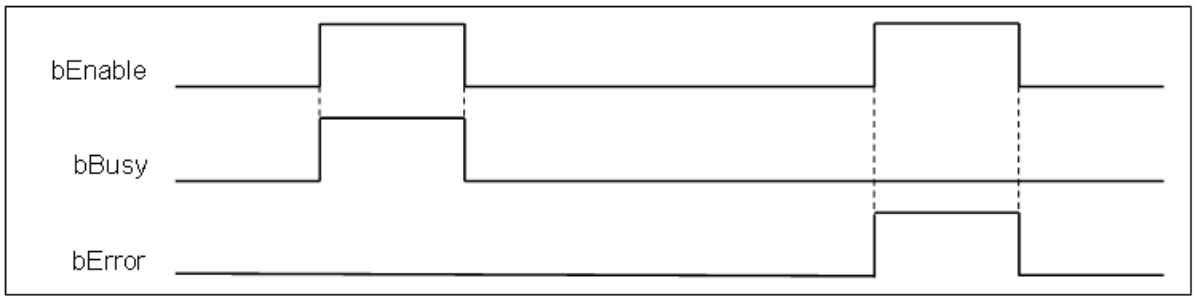
- **Outputs**

Name	Function	Data Type	Setting Value (Default value)
bBusy	The FB is running.	BOOL	TRUE/FALSE (FALSE)
iOutputState	Output point status	INT	0–15 (0)
bError	FB instruction error flag	BOOL	TRUE/FALSE (FALSE)
ErrorCode	Error code	DFB_AS_MODULE_API_ERROR	DFB_AS_MODULE_API_ERROR (DFB_NO_ERROR)

- **Output Updating Timing**

Name	Timing for shifting to TRUE	Timing for shifting to FALSE
bBusy	When *bEnable* shifts to true.	When *bEnable* shifts to false.
iOutputState	Continuously update after *bEnable*.	When *bEnable* shifts to false.
bError	When an error occurs during the running or there are incorrect input values.	When *bEnable* shifts to false.
ErrorCode		

- **Timing Diagram**



- **Function**

  - It is suggested that this instruction be placed under Main Task.
  - This function is supported by the AX-3 firmware V1.0.2 and later.
  - The output point Y0.0–Y0.3 in this instruction is the output point on the 02HC module.
  - The HCDO instruction is AS02HC-A dedicated instruction, and it can control the output point Y0.0–Y0.3 and show the status of the output point Y0.0–Y0.3.
  - Complete the *iOutputSetting* setting before running the instruction. When *bEnable* is started, *iOutputSetting* will be set once for the HC module, so the *iOutputSetting* set before starting will be taken as the initial state of the output point Y0.0–Y0.3. If users want to change the output point status during running, set *iOutputSetting* as a new value, and then set the *bUpdate* flag as On. After the instruction completes parameter change, the *bUpdate* flag will be cleared.
  - *byLocalID* specifies module numbers. The number of the first module on the right of CPU is 0, the number of the second module on the right of CPU is 1, and so on. Regardless of any type of modules, all modules must be counted. The maximum number of modules is 32.
  - *iOutputSetting* is the setting value of the output point action:

b15–b4	b3	b2	b1	b0
NA	Y0.3 action	Y0.2 action	Y0.1 action	Y0.0 action
NA	0：OFF 1：ON			

  - *iOutputState*is the status display of the output point

b15–b4	b3	b2	b1	b0
NA	Y0.3 status	Y0.2 status	Y0.1 status	Y0.0 status
NA	0：OFF 1：ON			

  - When the instruction is off, *iOutputState* changes to 0.
  - When compare output instruction DHCCMP or table compare output instruction DHCCMPT starts, output point cannot be changed by *iOutputSetting*. However, the display of the *iOutputState* status can be continuously updated.
  - If the compare output instruction needs to set the initial value to the output point, start the HCDO instruction first, and then start the DHCCMP or DHCCMPT instruction. If you need to change the

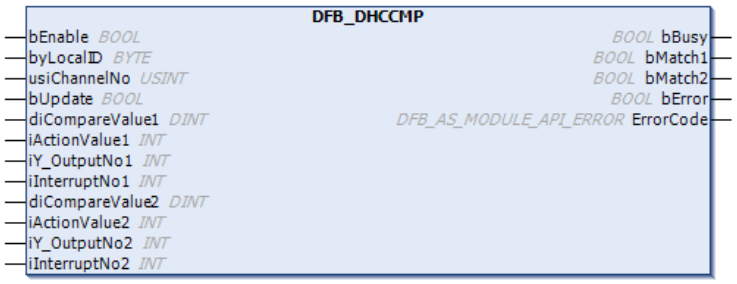
output point, stop the DHCCMP or DHCCMPT instruction, and then *IOutputSetting* of the HCDO instruction will update the latest output.

- When errors occur during startup, *bError*will be set to ON. Refer to error codes of ErrorCode for troubleshooting.

- **Supported Devices**
  - AX-3 series
- **Library**
  - DL_ASModuleAPI_AX3.library

6.17. DFB_DHCCMP

DFB_DHCCMP compares the outputs of the HC module.

FB/FC	Instruction	Graphic Expression	ST Language
FB	DFB_DHCCMP		DFB_DHCCMP(bEnable:= , byLocalID:= , usiChannelNo:= , bUpdate:= , diCompareValue1:= , iActionValue1:= , iY_OutputNo1:= , iInterruptNo1:= , diCompareValue2:= , iActionValue2:= , iY_OutputNo2:= , iInterruptNo2:= , bBusy=> , bMatch1=> , bMatch2=> , bError=> , ErrorCode=>);

- **Inputs**

Name	Function	Data Type	Setting Value (Default value)
bEnable	Run function block	BOOL	TRUE/FALSE (FALSE)
byLocalID	Expand module number	BYTE	0–31 (0)
usiChannelNo	Channel number	USINT	1–2 (1)
bUpdate	Update parameter flag	BOOL	TRUE/FALSE (FALSE)
diCompareValue1	Compare value1	DINT	–2, 147, 483, 648–2, 147, 483, 648 (0)
iActionValue1	Action of arrival compare value1	INT	0–8 (0)
iY_OutputNo1	Y output point number of arrival compare value1	INT	0–3(0)
iInterruptNo1	Arrive compare value1, External event interrupts number.	INT	0、400–431(0)
diCompareValue2	Compare value2	DINT	–2, 147, 483, 648–2, 147, 483, 648 (0)

Name	Function	Data Type	Setting Value (Default value)
iActionValue2	Action of arrival compare value2	INT	0–8 (0)
iY_OutputNo2	Y output point number of arrival compare value2	INT	0–3 (0)
iInterruptNo2	Arrive compare value2 External event interrupts number	INT	0, 400–431 (0)

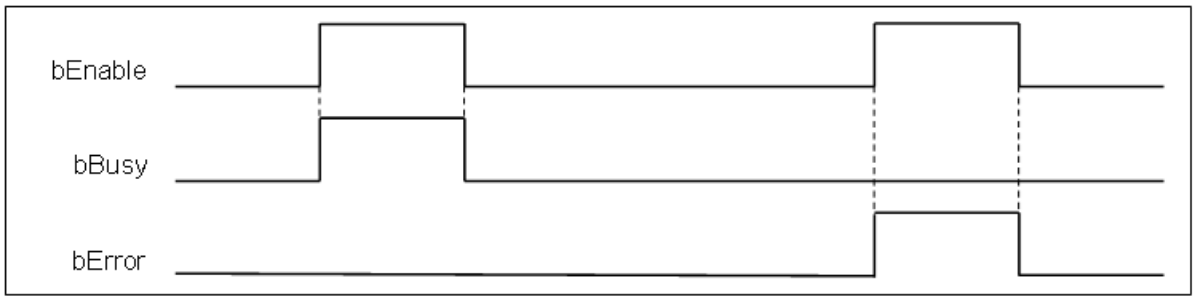
- **Outputs**

Name	Function	Data Type	Setting Value (Default value)
bBusy	The FB is running.	BOOL	TRUE/FALSE (FALSE)
bMatch1	Arrival comparison value1 flag	BOOL	TRUE/FALSE (FALSE)
bMatch2	Arrival comparison value 2 flag	BOOL	TRUE/FALSE (FALSE)
bError	FB instruction error flag	BOOL	TRUE/FALSE (FALSE)
ErrorCode	Error code	DFB_AS_MODULE_API_ERROR	DFB_AS_MODULE_API_ERROR (DFB_NO_ERROR)

- **Output Updating Timing**

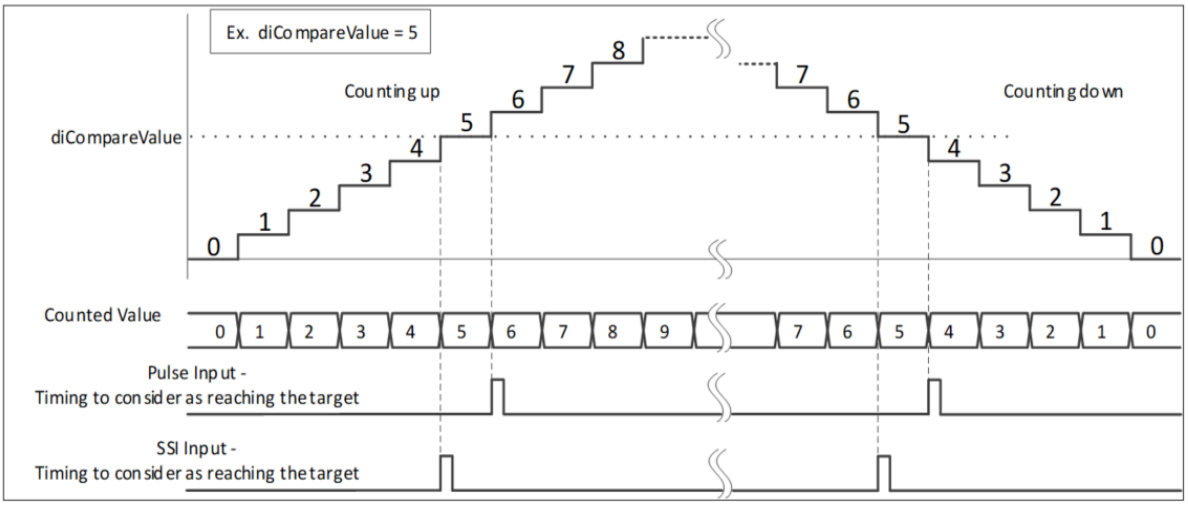
Name	Timing for shifting to TRUE	Timing for shifting to FALSE
bBusy	When *bEnable* shifts to true	When *bEnable* shifts to false
bMatch1	Arrival compare value1	When *bEnable* shifts to false
bMatch2	Arrival compare value2	When *bEnable* shifts to false
bError	When an error occurs during the running or there are incorrect input values	When *bEnable* shifts to false
ErrorCode		

- **Timing Diagram**



- **Function**
  - It is suggested that this instruction be placed under Main Task.
  - This function is supported by AX-3 firmware V1.0.2 and later.
  - The output point Y0.0–Y0.3 in this instruction is the output point on the 02HC module.

- The DHCCMP instruction is used for AS02HC-A. When the counting up/counting down value reaches the preset compare value, perform actions to output point and counter according to your settings.
- DHCCMP needs to be used with the DHCCNT instruction. Only when DHCCNT is started, counter value counts according to input signals and starts to compare.
- Complete setting *diCompareValue1*, *diCompareValue2*, iActionValue2, *iY_OutputNo1*, *iY_OutputNo2*, *iInterruptNo1*, *iInterruptNo2* parameters before running the instruction. When the instruction is enabled, the parameters of the AS02HC-A module will be set firstly.
- If you want to change parameters of *diCompareValue1*, *diCompareValue2*, *iActionValue1*, *iActionValue2*, *iY_OutputNo1*, *iY_OutputNo2*, *iInterruptNo1*, *iInterruptNo2* while running the instruction, set new values first, and then set *bUpdate* to On. When this instruction completes changing compare value, the instruction will clear *bUpdate* to Off, and clear *bMatch1* and *bMatch2* to Off.
- *byLocalID* specifies module numbers. The number of the first module on the right of the controller is 0, the number of the second module on the right of the controller is 1, and so on. All modules must be counted. The maximum number of modules is 32.
- *usiChannelNo* specifies the channel numbers. The number of channel one is 1, and the number of channel two is 2.
- *diCompareValue1* and *diCompareValue2* are compare value number 1 and compare value number 2 respectively (diCompareValue1 and *diCompareValue2* need to be different values). When the *usiChannelNo* channel counter value reaches the compare value of the number, *bMatch1* and *bMatch2* will be set to On, and the specified *iY_OutputNo1* and *iY_OutputNo2* output points will perform actions according to *iActionValue1* and *iActionValue2*.
  When selecting the counter form as Linear Counter, *diCompareValue1* and *diCompareValue2* need to meet “Minimum counter value < *diCompareValue* < Maximum counter value”. When using SSI encoder and set the counter value as Absolute Position, *diCompareValue1* and *diCompareValue2* need to meet “0 ≤ *diCompareValue* < $2^{(\mathrm{MT+ST^{length}})}$”.
- Note that compare arrival timing will be different according to the input interfaces “pulse input” or “SSI input” as shown in the following figure.



Take *diCompareValue* = 5 as an example, SSI input compare arrival timing occurs in the counter value 4→5 and 6→5 instantaneous. For pulse input, when counting up, compare arrival timing occurs in the counter value 5→6 instantaneous; when counting down, compare arrival timing occurs in the counter value 5→4 instantaneous.

- *iActionValue1*and *iActionValue2*are the specified running actions when compare arrival. The functions are as below:

iAction Value1	Function	iAction Value2	Function	Note
0	No action	0	No action	
1	*iY_OutputNo1*specifies output point Off.	1	*iY_OutputNo2* specifies output point Off.	
2	*iY_OutputNo1* specifies output point On.	2	*iY_OutputNo2* specifies output point On.	
3	*iY_OutputNo1*specifies output point Toggle output (alternating ON/OFF)	3	*iY_OutputNo2* specifies output point Toggle output (alternating ON/OFF)	
4	*iY_OutputNo1*specifies output point Off + clears the channel counter value.	4	*iY_OutputNo2* specifies output point Off + clears the channel counter value.	Since the *bMatch1*and *bMatch2*flags will be cleared when clearing the counter value, it is not suggested to judge the two bMatch flags when selecting these five modes.
5	*iY_OutputNo1* specifies output point On + clears the channel counter value.	5	*iY_OutputNo2* specifies output point On + clears the channel counter value.	
6	*iY_OutputNo1* specifies output point Toggle (alternating ON/OFF) + clears the channel counter value.	6	*iY_OutputNo2* specifies output point Toggle (alternating ON/OFF) + clears the channel counter value.	
7	Clear the channel counter value	7	Clear the channel counter value	
8	Clear the channel counter value + *iY_OutputNo1 I iY_OutputNo2* specifies output point Off.	8	Clear the channel counter value + *iY_OutputNo1 I iY_OutputNo2* specifies output point Off.	

Note: There will be a little delay between compare arrival occurrence and *iActionValue* running. The maximum delay time is 100us.

- *iY_OutputNo1*and *iY_OutputNo2* specify Y output point numbers for compare arrival number 1 and number 2 respectively. After compare instruction starts, the HCDO instruction will not be able to control output point:

iY_OutputNo1/iY_OutputNo2	Specified Output Point
0	Y0.0
1	Y0.1

iY_OutputNo1/iY_OutputNo2	Specified Output Point
2	Y0.2
3	Y0.3

- *iInterruptNo1*and *iInterruptNo2* are specified interrupt numbers. The functions are as below:

iInterruptNo	Action
0	Not to set interrupt
4□□	□□ = 00–31, **Interrupt_400_ModuleIN**0 to **Interrupt_431_ModuleIN31** of the corresponding **External event** The corresponding Task will be started when compare arrives.

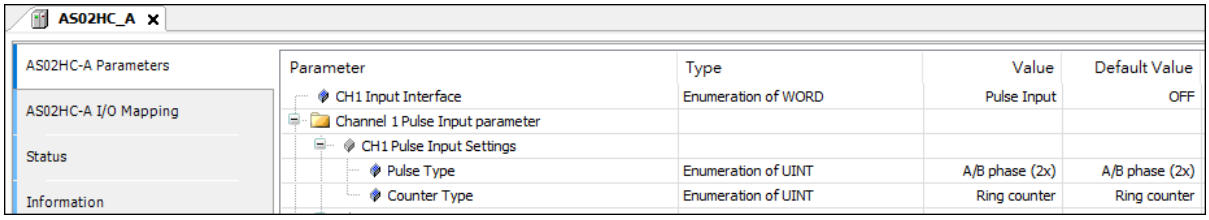
- *bMatch1* and *bMatch2*are the status flag display for the comparison values of compare arrival number1 and number2. When the counter value of the specified channel and number1 compare value & number compare value arrive, the corresponding *bMatch1*or *bMatch2*flag will be set to On. When the host STOP, clear counter, DHCCMP is OFF➔ON, or the *bUpdate*flag is set, the instruction will clear the *bMatch1*and *bMatch2* flags to Off.

- When the instruction is off, the relevant compare values and the output functions will not be updated.

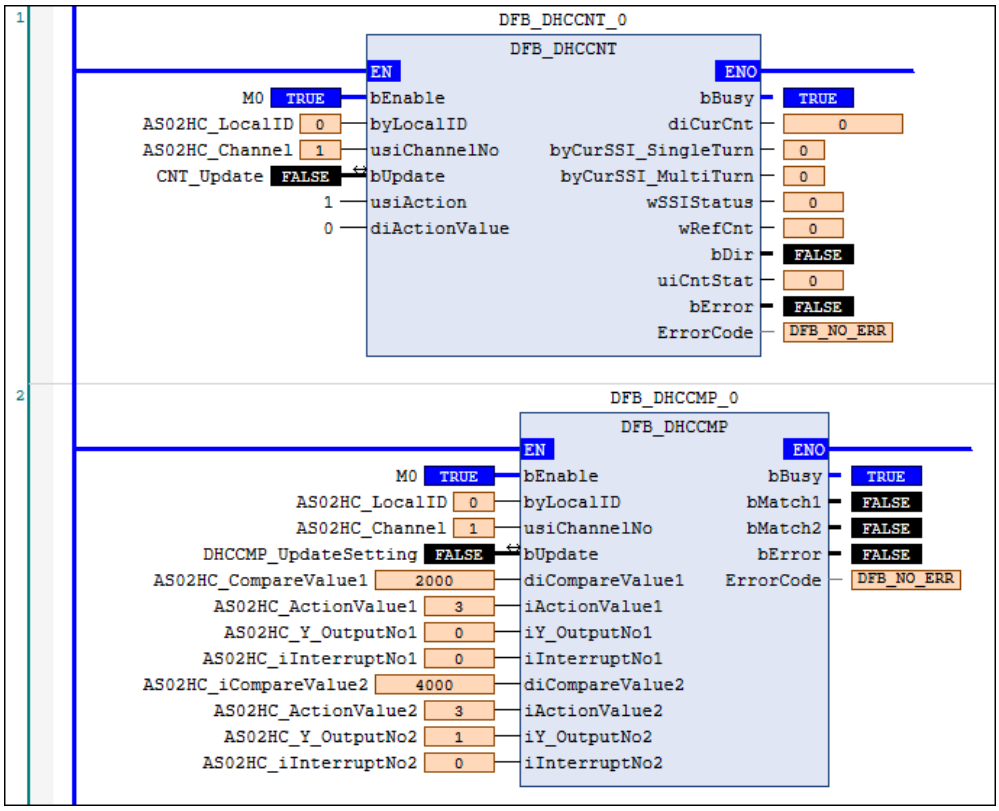
- If any error situation occurs during startup, the *bError* error flag will be set to ON. You can refer to ErrorCode for troubleshooting.

- **Programming Example 1: Control Y point output when compare arrives.**

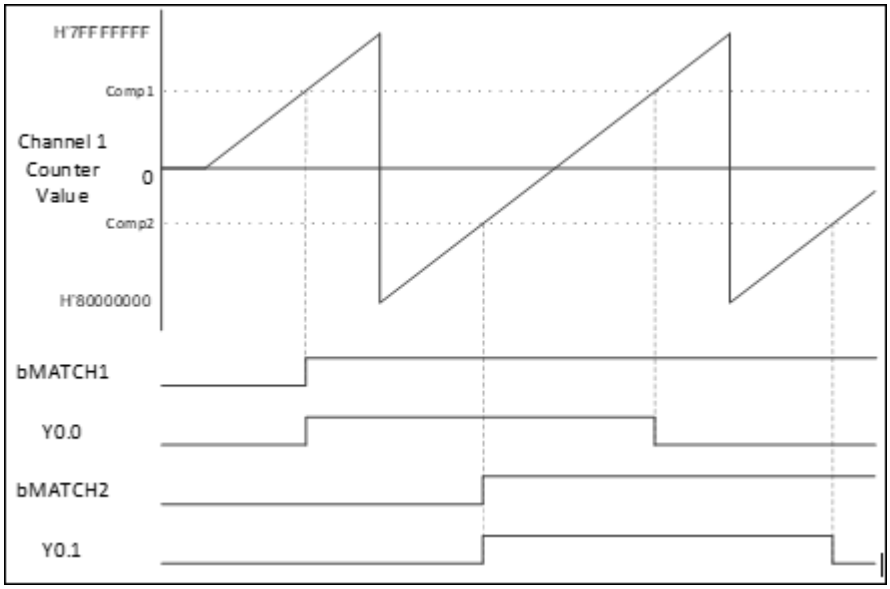
1. Set **CH1 Input Interface** in **AS02HC-A Parameters** as **Pulse Input**, set **Pulse Type** as **A/B phase(2x)**, and set **Counter Type** as **Ring counter**, as the following:



2. Set *diCompareValue1* to 2000, set *iActionValue1* to 3, set *iY_OutputNo1* to 0, and set *iInterruptNo1* to 0; set *diCompareValue2* to 4000, set *iActionValue2* to 3, set *iY_OutputNo2* to 1, and set *iInterruptNo2* to 0.
3. When setting M0 = ON, DHCCNT counter starts to count. At the same time, set the DHCCMP parameter for the module, and wait for the counter comparison to arrive
4. When counter values reach diCompareValue1, the *bMatch1* flag is set to ON, and Y0.0 is OFF➔ON.
5. Continuous counting until *diCompareValue2* is reached, and the *bMatch2* flag is set to ON, and Y0.1 is OFF➔ON.
6. Continuous counting until *diCompareValue1* is reached again, Y0.0 is ON➔OFF. Continuous counting until *diCompareValue2* is reached again, Y0.1 is ON➔OFF.

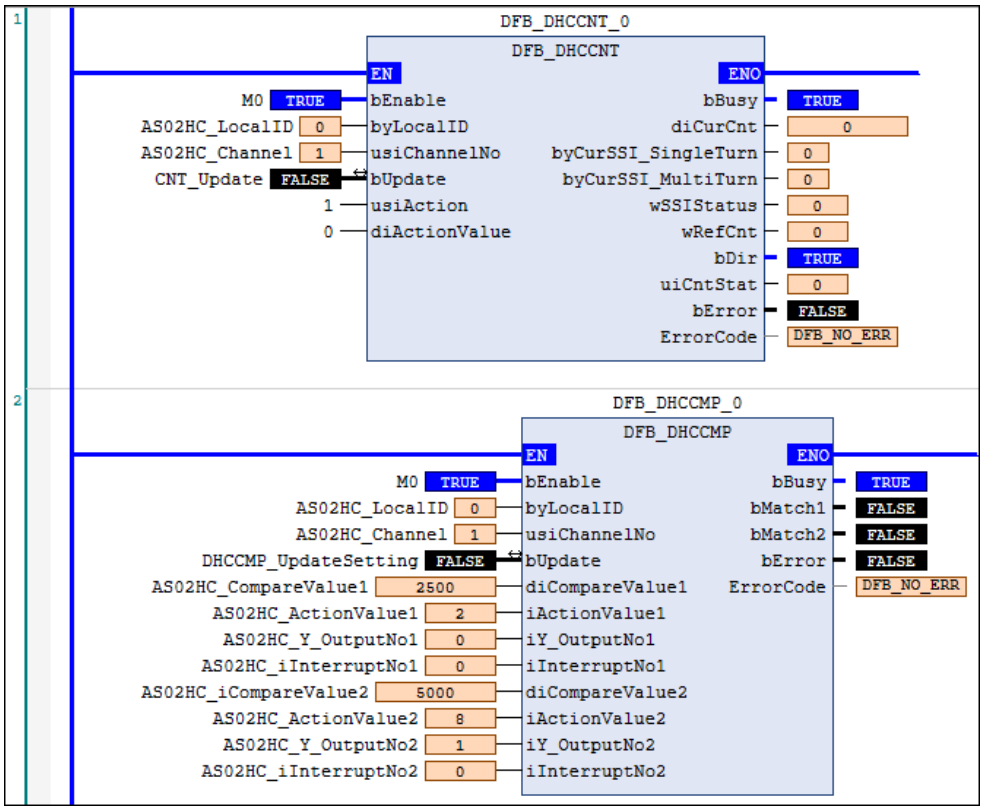


- **Timing Diagram**

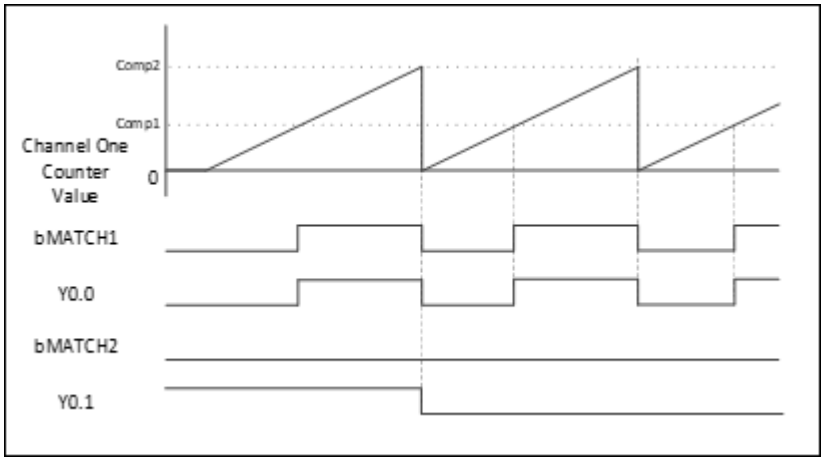


- **Programming Example 2: Clear the counter value when compare arrives**

1. Set *diCompareValue1* to 2500, set *iActionValue1* to 2, set *iY_OutputNo1* to 0 and set *iInterruptNo1* to 0; set *diCompareValue2* to 5000, set *iActionValue2* to 8, set *iY_OutputNo2* to 1, and set *iInterruptNo2* to 0.
2. When setting M0 = ON, DHCCNT counter starts to count. At the same time, set the DHCCMP parameter for the module, and start to wait for counter comparison to arrive.
3. When counter values reach *diCompareValue1*, the *bMatch1* flag is set to ON, and Y0.0 is set to ON.
4. Continuous counting until *diCompareValue2* is reached. Since the action of *iActionValue2* = 8 is "clear the channel counter value + clear *iY_OutputNo1* and *iY_OutputNo2* specified output point", the counter value is cleared to 0, *bMatch1* and *bMatch2* are cleared to FALSE, and Y0.0 & Y0.1 are cleared to OFF.
5. Continuous counting until *diCompareValue1* is reached again, the *bMatch1* flag will be set to on again, Y0.0 will be set to ON, and so on.



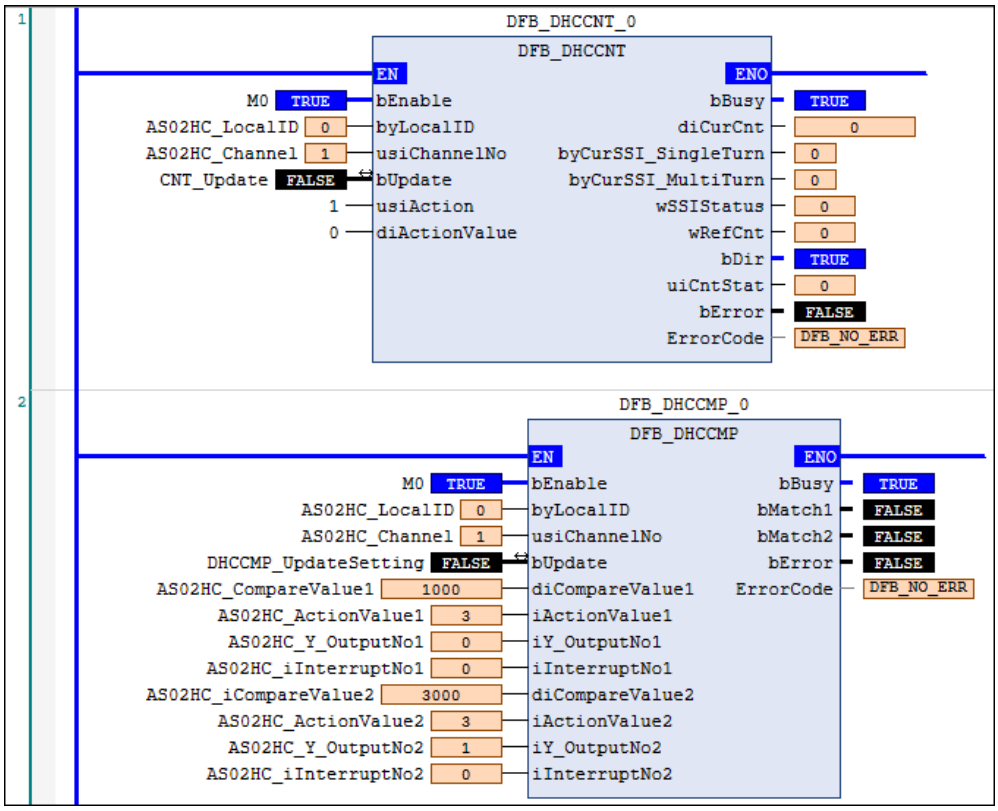
- **Timing Diagram**



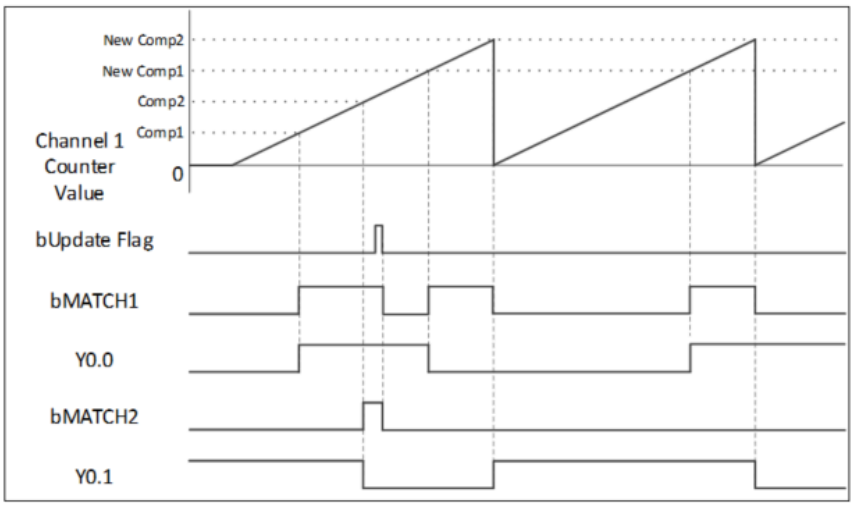
- **Programming Example 3: Update the compare value**

1. Set *diCompareValue1* to 1000, set *iActionValue1* to 3, set *iY_OutputNo1* to 0, and set *iInterruptNo1*to 0; set *diCompareValue2* to 3000, set *iActionValue2* to 3, set *iY_OutputNo2* to 1, and set *iInterruptNo2* to 0.
2. When setting M0 = ON, DHCCNT counter starts to count. At the same time, set the DHCCMP parameter for the module, and start to wait for counter comparison to arrive
3. When counter value reaches *diCompareValue1*, the *bMatch1* flag will be set to ON, and Y0.0 will be OFF➔ON.
4. When counter value reaches *diCompareValue2*, the *bMatch2* flag will be set to ON, and Y0.1 will be ON➔OFF.
5. Then, set new compare value: set *diCompareValue1* to 5000, set *iActionValue1* to 3, set *OutputNo1* to 0, set *iInterruptNo1* to 0, set *diCompareValue2* to 7000, set *iActionValue2* to 6, set *iY_OutputNo2* to 1, and set *iInterruptNo2* to 0.
6. Set the *bUpdate* flag to ON. When the setting completes, the *bUpdate* flag will be automatically cleared to OFF. At the same time, the *bMatch1* and *bMatch2* flags will be cleared to OFF.

7. When counter value reaches new compare value *diCompareValue1*, the *bMatch1* flag will be set to ON, and Y0.0 will be ON➔OFF.
8. When counter value reaches new compare value *diCompareValue2*, Y0.1 will be OFF➔ON. Because *iActionValue2* = 6 include Reset, counting value will be cleared to 0. At the same time, the *bMatch1* and *bMatch2* flags will be cleared to OFF.

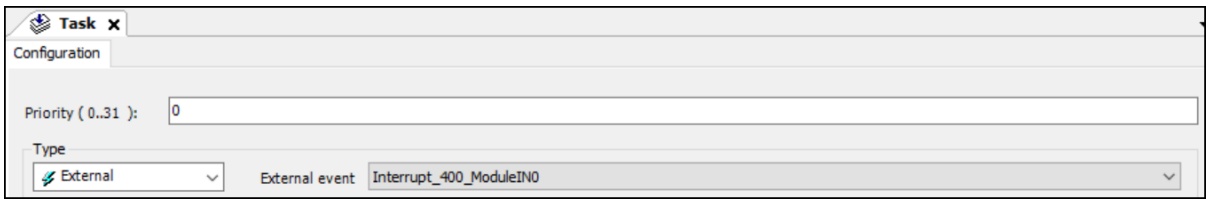


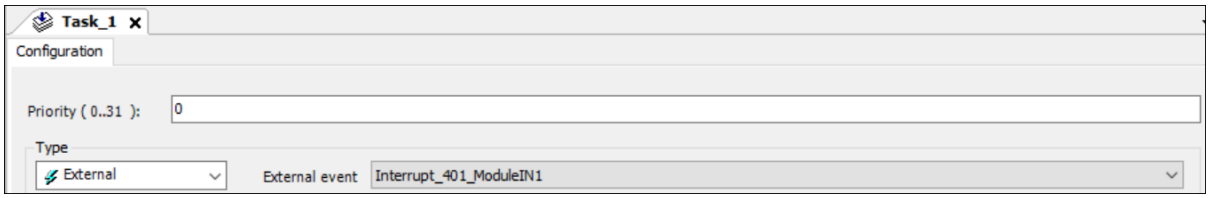
- **Timing Diagram**



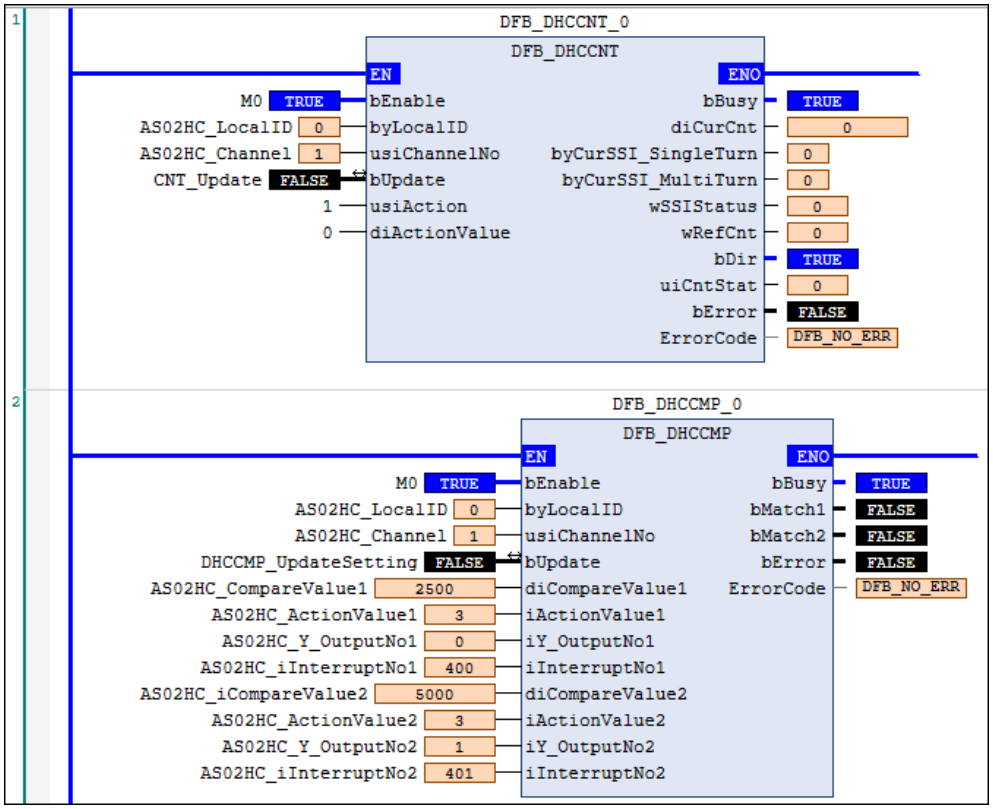
- **Programming Example4: Compare arrival interruption**

1. DHCCMP can also configure compare arrival interruption. Set *iInterruptNo1*and *iInterruptNo2*to 400 and 401 respectively, and configure two Tasks at the same time. In the tasks, select Interrupt_400_ModuleIN0 or Interrupt_401_ModuleIN1 for External event as shown in the following figures.





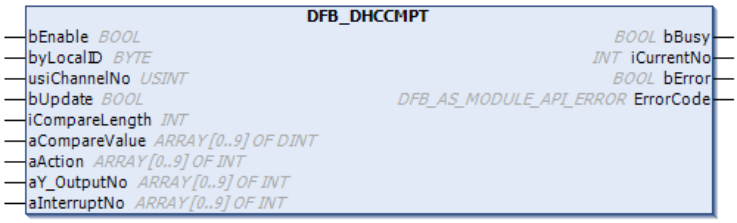
2. Set *diCompareValue1* to 2500, set *iActionValue1* to 3, set *iY_OutputNo1* to 0 set *iInterruptNo1* to 400; set *diCompareValue2* to 5000, set *iActionValue2* to 3, set *iY_OutputNo2* to 1, and set *iInterruptNo2* to 401.
3. When setting M0 = ON, DHCCNT counter starts to count. At the same time, set the DHCCMP parameter for the module, and start to wait for counter compare arrival.
4. When counter value reaches *diCompareValue1* , External event corresponding to *iInterruptNo1* will be Run, and Y0.0 will be toggled (alternating ON/OFF).
5. When counter value reaches *diCompareValue2* , External event corresponding to *iInterruptNo2* will be Run, and Y0.1 will be toggled (alternating ON/OFF).



- **Supported Devices**
  - AX-3 series
- **Library**
  - DL_ASModuleAPI_AX3.library

6.18. DFB_DHCCMPT

DFB_DHCCMPT compares the count value of the HC module with the matrix table, and output the action when the condition is valid.

FB/FC	Instruction	Graphic Expression	ST Language
FB	DFB_DHCCMPT		DFB_DHCCMPT(bEnable:= , byLocalID:= , usiChannelNo:= , bUpdate:= , iCompareLength := , aCompareValue := , aAction:= , aY_OutputNo:= , aInterruptNo:= , bBusy=> , iCurrentNo=> , bError=> , ErrorCode=>);

- **Inputs**

Name	Function	Data Type	Setting Value (Default value)
bEnable	Run function block	BOOL	TRUE/FALSE (FALSE)
byLocalID	Expand module number	BYTE	0–31 (0)
usiChannelNo	Channel number	USINT	1–2 (1)
bUpdate	Update parameter flag	BOOL	TRUE/FALSE (FALSE)
iCompareLength	The number of rows of the table	INT	2–10 (2)
aCompareValue	The storage source of the 32–bit value to be compared in the compare table	ARRAY[0..9]OF DINT	–2, 147, 483, 648–2, 147, 483, 648 (0)
aAction	The action storage source in the table when comparison is complete and the action condition is reached	ARRAY[0..9]OF INT	0–8 (0)
aY_OutputNo	The storage source of the Y output No. in the table when comparison is complete and the action condition is reached	ARRAY[0..9]OF INT	0–3 (0)
aInterruptNo	The storage source of the external event interrupt No. in the table when	ARRAY[0..9]OF INT	0, 400–431 (0)

Name	Function	Data Type	Setting Value (Default value)
	comparison is complete and the action condition is reached		

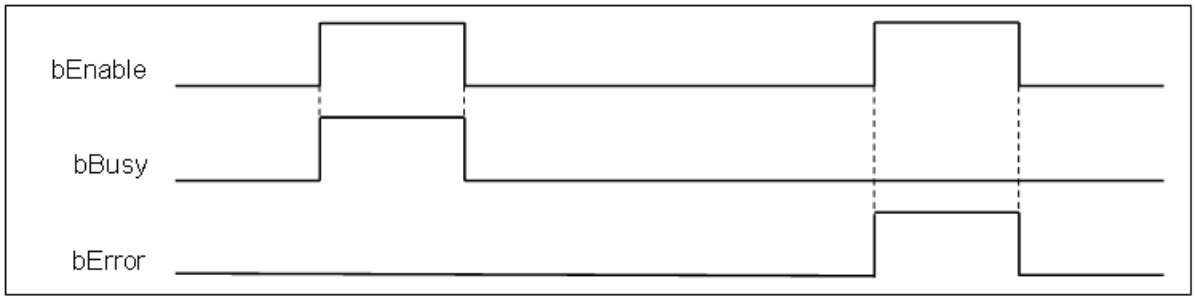
- **Outputs**

Name	Function	Data Type	Setting Value (Default value)
bBusy	The FB is running.	BOOL	TRUE/FALSE (FALSE)
iCurrentNo	The number of conditions and actions that has been compared so far.	INT	0–10 (0)
bError	FB instruction error flag	BOOL	TRUE/FALSE (FALSE)
ErrorCode	Error code	DFB_AS_MODULE_API_ERROR	DFB_AS_MODULE_API_ERROR (DFB_NO_ERROR)

- **Output Updating Timing**

Name	Timing for shifting to TRUE	Timing for shifting to FALSE
bBusy	When *bEnable* shifts to true	When *bEnable* shifts to false
iCurrentNo	Display according to the number of the group that currently has compare arrived.	When *bEnable* shifts to false
bError	When an error occurs during the running or there are incorrect input values.	When *bEnable* shifts to false
ErrorCode		

- **Timing Diagram**



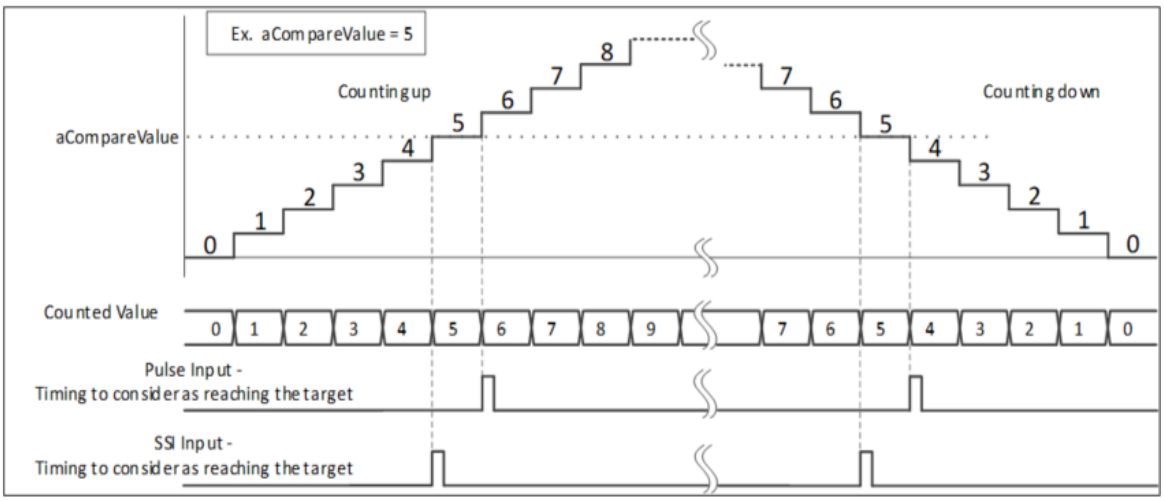
- **Function**
  - It is suggested that this instruction be placed under Main Task.
  - This function is supported by AX-3 firmware V1.0.2 and later.
  - The output point Y0.0–Y0.3 in this instruction is the output point on the 02HC module.
  - The DHCCMPT instruction is used for the AS02HC-A instruction. The order of comparison is cyclic according to the order set in the group. When the number of comparison reaches the last compare value, the next compare value will be reset as the first compare value. The DHCCMPT instruction is limited to be used when the counter counts in one direction (Changing the direction will cause incorrect action), and up to 10 compare values can be set. Compare values need to be sorted by incremental or decremental numbers. For incremental sorting, the counter needs to counting up, and only positive numbers for compare value. For decremental sorting, counter needs to counting down,

and only negative numbers for compare value. When the counting reaches the preset compare value, it will output pint and counter according to your setting, and the request the host to interrupt.

- DHCCMPT needs to be used with the DHCCNT instruction. Only when DHCCNT starts, counter value counts according to the input signal, and perform compare action.
- Complete *iCompareLength*, *aCompareValue*, *aAction*, *aY_OutputNo*, and *aInterruptNo* parameter setting before running the instruction. *bEnable* will write the parameter once when it is first started.
- The parameters of *iCompareLength*, *aCompareValue*, *aAction*, *aY_OutputNo*, and *aInterruptNo* can be changed during running. The changing methods is to set them as new values, and then set *bUpdate* to On. When the changing is complete, the instruction will clear the *bUpdate* flag to Off.
- *byLocalID* specifies module numbers. The number of the first module on the right of the controller is 0, the number of the second module on the right of the controller is 1, and so on. All modules must be counted. The maximum number of modules is 32.
- *usiChannelNo* specifies the channel numbers. The number of channel one is 1, and the number of channel two is 2.
- *iCompareLength* is the number of groups of compare tables. Its compare action is cyclical according to the set group numbers. The group number length can be set to 2–10. When the set value is out of this range, the instruction will not be run, and the error code shows.
- *aCompareValue* is the storage source of the compare vale, and its type is ARRAY[0..9] OF DINT. Compare values needs to be sorted by incremental or decremental numbers (each compare value needs to be different). For incremental sorting, counter needs to counting up, and only positive numbers for compare value. For decremental sorting, counter needs to counting down, and only negative numbers for compare value.When selecting the counter form as Linear Counter, each vale in *aCompareValue* needs to meet “Minimum counter value < *aCompareValue* < Maximum counter value”.

  When using SSI encoder and the counter form is set as Absolute Position, each value in *aCompareValue* needs to meet “0 ≤ *aCompareValue* ≤$2^{(\mathrm{MT}+\mathrm{ST}^{\mathrm{length}})}$”.

- Note that compare arrival timing will be different according to the input interface “Pulse input” or “SSI input” as shown in the following figure.



  Take *aCompareValue* = 5 as an example, SSI input compare arrival timing occurs in the counter value 4→5 and 6→5 instantaneous. For pulse input, when counting up, compare arrival timing occurs in the counter value 5→6 instantaneous; when counting down, compare arrival timing occurs in the counter value 5→4 instantaneous.

- *aAction*is the specified action code when compare arrives. The functions are as below:

aAction	Function	Note
0	No action	
1	*aY_OutputNo* specifies the output point Off.	
2	*aY_OutputNo* specifies the output point On.	
3	*aY_OutputNo* specifies the output point Toggle output (alternating ON/OFF).	
4	*aY_OutputNo* specifies the output point Off + clear the channel counter value.	In addition to clearing the channel counter value, clear *iCurrentNo* to 0.
5	*aY_OutputNo* specifies the output point On + clear the channel counter value.	
6	*aY_OutputNo* specifies the output point Toggle (alternating ON/OFF) + clear the channel counter value.	
7	Clear the channel counter value.	
8	Clear the channel counter value + all *aY_OutputNo* specify the output point Off.	Same as *aAction* = 7 clearance, and clear all specified *aY_OutputNo*output.

Note: There will be a little delay between compare arrival occurrence and Action running. The maximum delay time is 100us.

- *aY_OutputNo* is specified output point number. The function is as the following:

aY_OutputNo	Output Point
0	Y0.0
1	Y0.1
2	Y0.2
3	Y0.3

- *aInterruptNo* is specified interruption number. The function is as the following:

aInterruptNo	Action
0	Not to set interruption
4□□	□□ = 00–31, **Interrupt_400_ModuleIN0** to **Interrupt_431_ModuleIN31** of the corresponding **External event** The corresponding Task will be started when compare arrives.

- *iCurrentNo*displays the number of the group that currently has compare arrived. For example, when the counter value is 200, which is lower than the group number 1 in the following compare table, the *iCurrentNo* value is 0; when the counter value reaches 1000, which equals to the group number 1 in the following compare table, the *iCurrentNo* value is 1.

- Refer to the following table for the operator description of *aCompareValue*, *aAction*, *aY_OutputNo*, *aInterruptNo*, and *iCurrentNo*. In the table, the *iCompareLength* value is assumed as 6.

iCurrentNo Item Number	aCompareValue Source of Value to be compared	aActionSpecified Action after comparison and condition is reached	aY_OutputNoOutput Number	aInterruptNoInterruption Number
1	1000	2(On)	0(Y0.0)	400
2	2000	2(On)	1(Y0.1)	401
3	3000	3(Toggle)	0(Y0.0)	402
4	4500	2(On)	2(Y0.2)	403
5	5500	1(Off)	3(Y0.3)	404
6	6500	1(Off)	1(Y0.1)	405

- When the instruction is started (*bEnable*is from Off to On) and the *bUpdate* flag is set to On to change the parameter, the instruction will compare all the compare values in the compare table set by users according to the current counter vale. The group values that are smaller than the current counter value will Run the compare arrival action. For example, when the instruction starts with the counter value 3500, the actions numbered 1 –3 (compare value < current counter value) in the compare table will be Run once with the compare arrival action (Y0.0=On, Y0.1=On, Y0.0 Toggle (alternating ON/OFF)), and *iCurrentNo*will be set to 3.
- If *bEnable* is from On to Off, it means that shutting the table compare output function. *iCurrentNo* status display will return to 0. The output point state is not changed by the instruction being off. Counter remains counting, but no more comparison.
- If any error situation occurs during startup, the *bError* error flag will be set to ON. You can refer to ErrorCode for troubleshooting.

- **Programming Example**

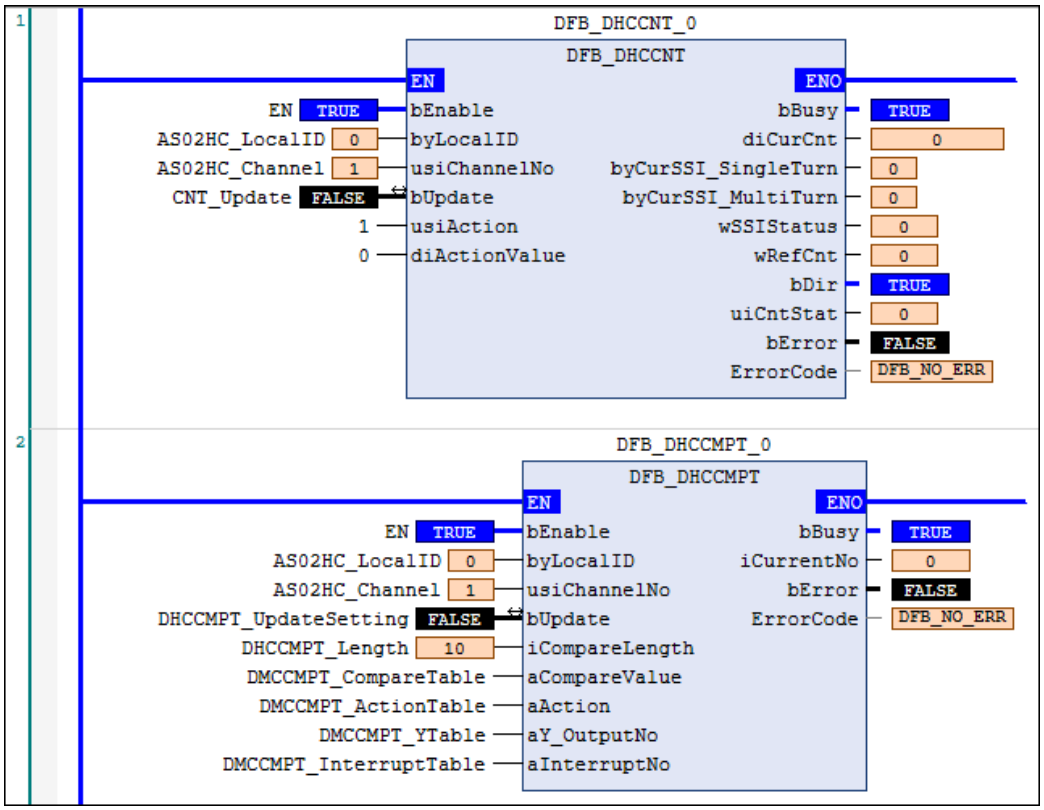
1. Set the parameters of the relevant input pins as shown in the following figure (The settings in the following figure are equivalent to those shown in the following table).

Scope	Name	Address	Data type	Initialization
VAR	**DHCCMPT_Length**		INT	10
VAR	**DMCCMPT_CompareTable**		ARRAY [0..9] OF DINT	[1000, 2000, 3000, 4000, 5000, 6000, 7000, 8000, 9000, 10000]
VAR	**DMCCMPT_ActionTable**		ARRAY [0..9] OF INT	[10(3)]
VAR	**DMCCMPT_YTable**		ARRAY [0..9] OF INT	[0, 1, 2, 3, 0, 1, 2, 3, 0, 1]
VAR	**DMCCMPT_InterruptTable**		ARRAY [0..9] OF INT	[400, 401, 402, 403, 404, 405, 406, 407, 408, 409]

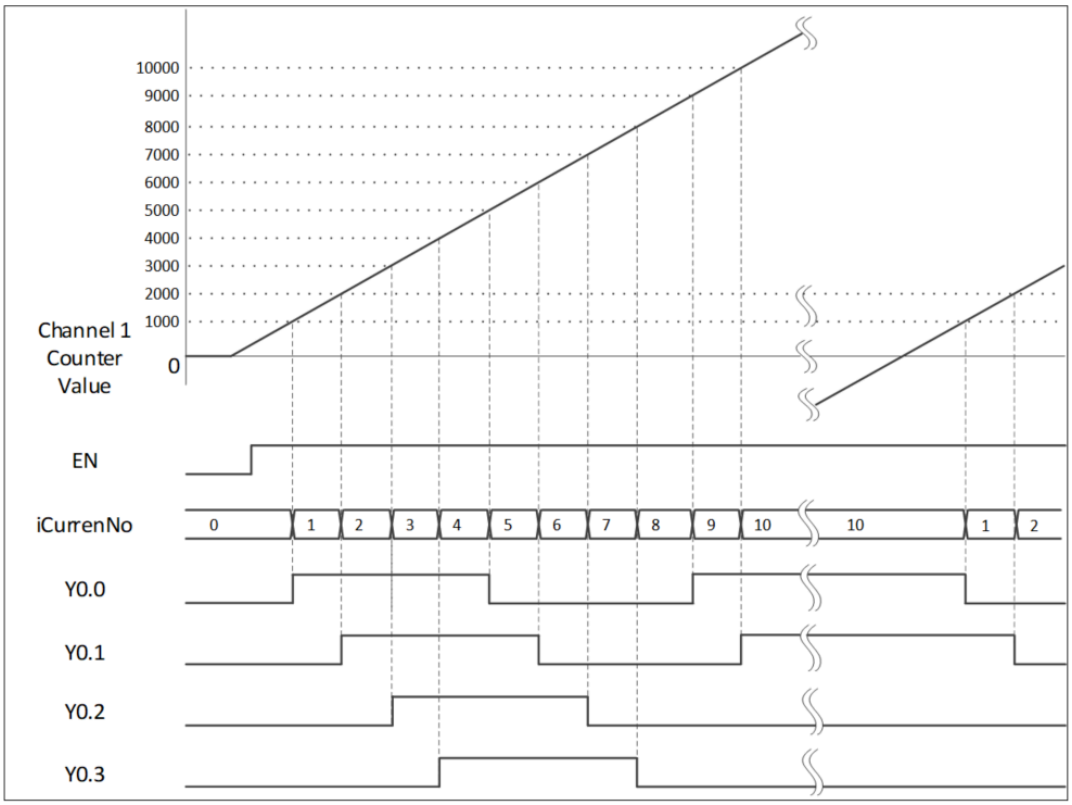
iCurrentNo Item Number	aCompareValue Source of Value to be compared	aActionSpecified Action after comparison and condition is reached	aY_OutputNoOutput Number	aInterruptNoInterruption Number
1	1000	3 (Toggle)	0 (Y0.0)	400
2	2000	3 (Toggle)	1 (Y0.1)	401
3	3000	3 (Toggle)	2 (Y0.2)	402
4	4000	3 (Toggle)	3 (Y0.3)	403
5	5000	3 (Toggle)	0 (Y0.0)	404
6	6000	3 (Toggle)	1 (Y0.1)	405
7	7000	3 (Toggle)	2 (Y0.2)	406

iCurrentNo Item Number	aCompareValue Source of Value to be compared	aActionSpecified Action after comparison and condition is reached	aY_OutputNoOutput Number	aInterruptNoInterruption Number
8	8000	3 (Toggle)	3 (Y0.3)	407
9	9000	3 (Toggle)	0 (Y0.0)	408
10	10000	3 (Toggle)	1 (Y0.1)	409

2. When EN is OFF→ON, the instruction will compare all the compare values in the compare table set by users. The group values that are smaller than the current counter value will Run the compare arrival action. Because the counter value is still smaller than the first compare value 1000 when startup. No compare arrival specified action will be Run, and *iCurrentNo* is 0 currently.
3. When the counter value reaches 1000, compare arrives the first compare value (If it is pulse input, compare arrival specified action will be Run when the counter value is 1000→1001).
4. When the counter value reaches 2000, compare arrives the second compare value, Y0.1 is OFF➔ON, 401 interruption program is running, and *iCurrentNo*=2.
5. When the counter value reaches 3000, compare arrives the third compare value, Y0.2 is OFF➔ON, 402 interruption program is running, and *iCurrentNo*=3.
6. When the counter value reaches 4000, compare arrives the fourth compare value, Y0.3 is OFF➔ON, 403 interruption program is running, and *iCurrentNo*=4.
7. Follow this rule to continue compare arrival from the fifth to tenth compare value. At this time *iCurrentNo*=10. Because the last comparison has done, next compare vale will be set as the first compare value 1000.
8. When the ring counter counts to 1000 again, compare arrives the first counter value, Y1.0 is ON➔OFF, 400 interruption program is running, and *iCurrentNo*=1. Follow this rule to Run the subsequent comparison.



- **Timing Diagram**



- **Supported Devices**
  - AX-3 series
- **Library**
  - DL_ASModuleAPI_AX3.library

6.19. DFB_DHCMEAS

DFB_DHCMEAS measures the HC module frequency and rotation speed.

FB/FC	Instruction	Graphic Expression	ST Language
FB	DFB_DHCMEAS		DFB_DHCMEAS(bEnable:= , byLocalID:= , usiChannelNo:= , bUpdate:= , udiPulsePerRev:= , iSamplingTime:= , iMovingAvgWindow := , bBusy=> , diFrequency=> , diRPM=> , bError=> , ErrorCode=>);

- **Inputs**

Name	Function	Data Type	Setting Value (Default value)
bEnable	Run function block	BOOL	TRUE/FALSE (FALSE)
byLocalID	Expand module number	BYTE	0–31 (0)
usiChannelNo	Channel number	USINT	1–2 (1)
bUpdate	Update parameter flag	BOOL	TRUE/FALSE (FALSE)
udiPulsePerRev	Counting quantity per lap	UDINT	1–4294967295 (1)
iSamplingTime	Sampling period (unit: ms)	INT	1–1000 (1)
iMovingAvgWindow	Average number of moves	INT	1–10 (1)

- **Outputs**

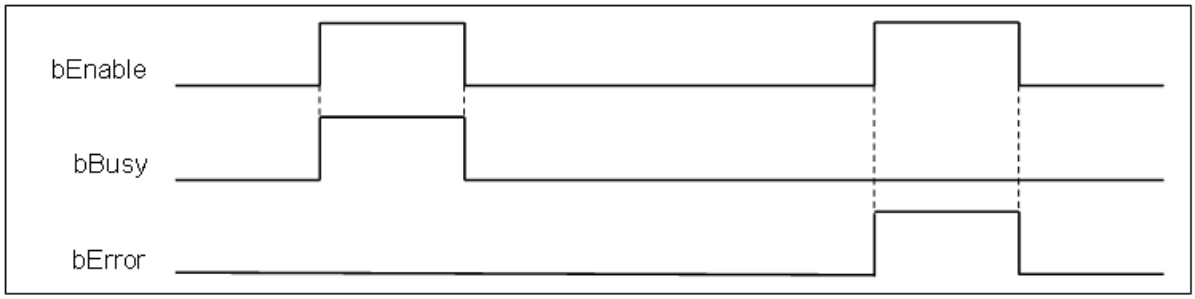
Name	Function	Data Type	Setting Value (Default value)
bBusy	The FB is running.	BOOL	TRUE/FALSE (FALSE)
diFrequency	Average measurement frequency results	DINT	–2, 147, 483, 648–2, 147, 483, 648 (0)
diRPM	Average rotation speed	DINT	–2, 147, 483, 648–2, 147, 483, 648 (0)

Name	Function	Data Type	Setting Value (Default value)
	measurement results		
bError	FB instruction error flag	BOOL	TRUE/FALSE (FALSE)
ErrorCode	Error code	DFB_AS_MODULE_API_ERROR	DFB_AS_MODULE_API_ERROR (DFB_NO_ERROR)

- **Output Updating Timing**

Name	Timing for shifting to TRUE	Timing for shifting to FALSE
bBusy	When *bEnable* shifts to true	When *bEnable* shifts to false
diFrequency	Continuously update after *bEnable*	When *bEnable* shifts to false
diRPM	Continuously update after *bEnable*	When *bEnable* shifts to false
bError	When an error occurs during the running or there are incorrect input values	When *bEnable* shifts to false
ErrorCode		

- **Timing Diagram**



- **Function**
  - It is suggested that this instruction be placed under Main Task.
  - This function is supported with AX-3 firmware V1.0.2 and later.
  - The output point Y0.0–Y0.3 in this instruction is the output point on the 02HC module.
  - The DHCMEAS instruction is AS02HC-A dedicated instruction. Its functions are frequency and rotation speed measurements.
  - DHCMEAS needs to be used with the DHCCNT instruction. Only when DHCCNT is started, the counter value counts according to input signals, and the measurement results are calculated by the change of the counter values.
  - Complete setting *udiPulsePerRev*, *iSamplingTime*, and *iMovingAvgWindow* parameters before running the instruction. *bEnable* will write the parameter once when it is first started. If users want to change parameters during running, the change method is to set new values first, and then set the *bUpdate* flag to On. When this instruction completes the change, the instruction will clear the *bUpdate* flag to Off.
  - *byLocalID* specifies module numbers. The number of the first module on the right of CPU is 0, the number of the second module on the right of CPU is 1, and so on. Regardless of any type of modules, all modules must be counted. The maximum number of modules is 32.

- *usiChannelNo* specifies the channel numbers. The number of channel one is 1, and the number of channel two is 2.
- *udiPulsePerRev* is the counter value of one rotation of the encoder, and its setting range is 1–4294967295 (H'00000001–H'FFFFFFFF).
- *iSamplingTime* is sampling period, and its setting range is 1–1000 (unit: ms). According to the setting of *iSamplingTime*, *diFrequency* frequency measurement result output and *diRPM* rotation speed measurement result output will have different resolution.

  $$\textbf{diFrequency}\text{resolution} = 1000 \div \textbf{iSamplingTime}(\text{unit}:\text{Hz})$$

  $$\textbf{diRPM}\text{resolution} = 60000 \div (\textbf{iSamplingTime} \times \textbf{udiPulsePerRev})(\text{unit}:\text{rev}/\text{min})$$

  Because the rotation speed calculation method is based on the CurCnt counter value of the beginning and end of the sampling period, the following situations need to be excluded when designing sampling periods.

Encoder Type	Counter Type	Factors to Effect Measurement Accuracy
Incremental encoder	Ring Counter	Displacement exceeds 2^{31} in the sampling period.
	Linear Counter	Displacement exceeds 2^{31} in the sampling period, or the counter value exceeds the upper/lower limit.
Absolute SSI encoder	Shows absolute position	Displacement exceeds $2^{(MT+ST^{length})-1}$ in the sampling period.
	Ring Counter	Displacement exceeds 2^{31} in the sampling period.

- *iMovingAvgWindow* is the average number of times, which performs moving average to measurement results. Its setting range is 1–10.
- *diFrequency* is the result of average measurement frequency (unit: Hz). The calculation method of the frequency is as below:

  $$\textbf{diFrequency}\left(\text{Hz}\right) = \frac{\text{CurCnt}(t+\textbf{iSamplingTime}) - \text{CurCnt}(t)}{\textbf{iSamplingTime}(\text{ms}) \times 10^{-3}}$$

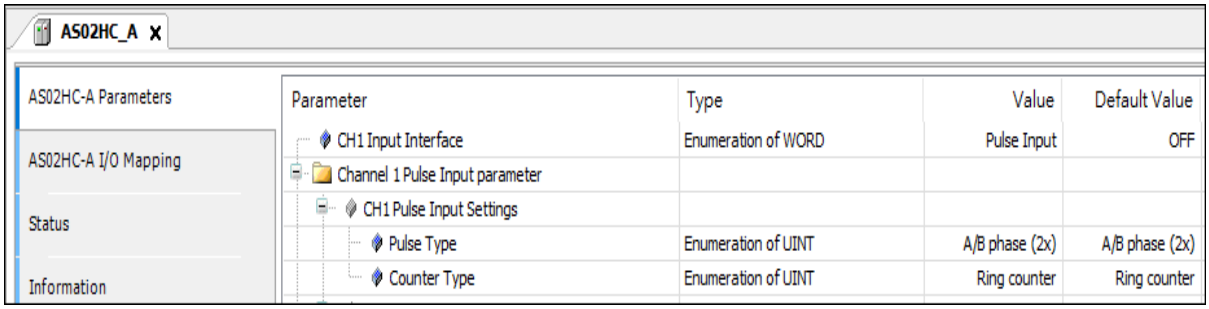
- *diRPM*is the result of the average rotation speed measurement (unit: rev/min). The calculation method of the rotation speed is as below:

  $$\textbf{diRPM}\left(\text{rev}\middle/\text{min}\right) = \frac{(\text{CurCnt}(t+\textbf{iSamplingTime}) - \text{CurCnt}(t)) \times 60}{\textbf{udiPulsePerRev} \times \textbf{iSamplingTime}(\text{ms}) \times 10^{-3}}$$

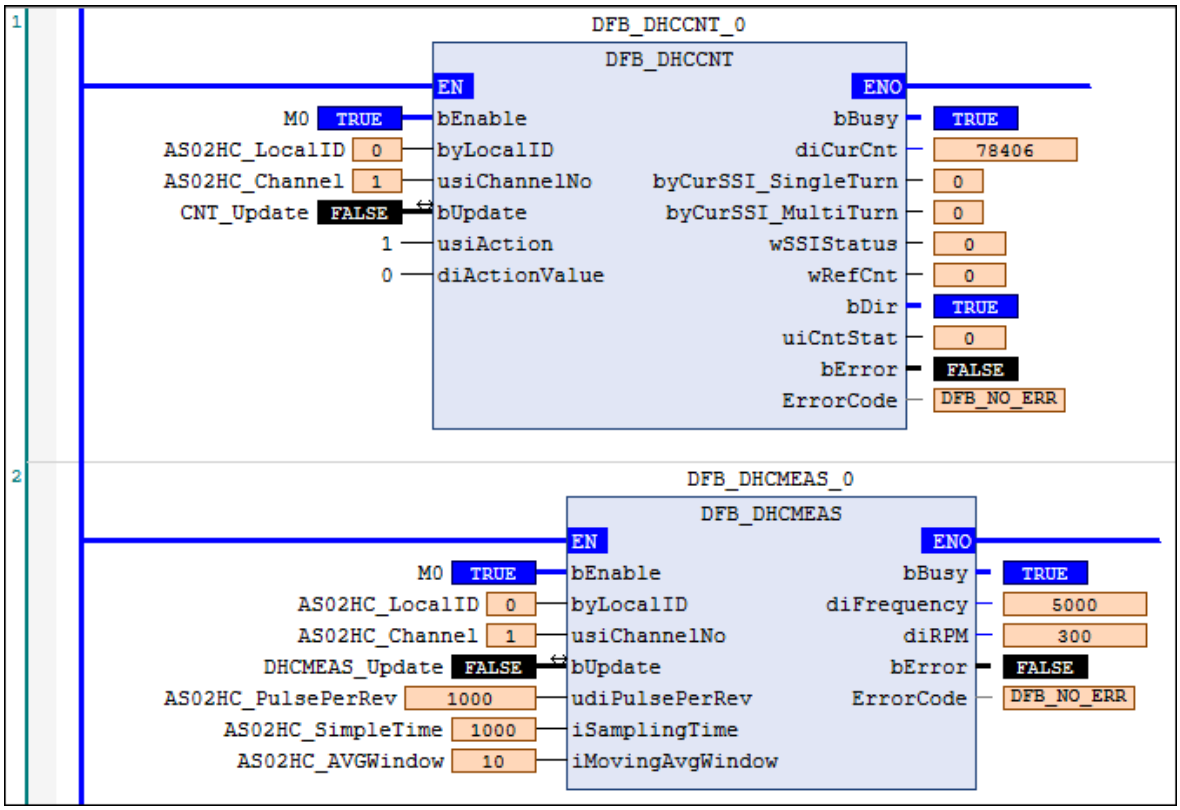
- When the instruction stops, it means that the measurement function is disabled, and *diFrequency*and *diRPM*will remain the same and no more update.
- Note that when the counter value is cleared or users change the counter value, the calculation result of *diFrequency* and *diRPM* of the sampling period will be affected.
- If any error situation occurs during startup, the *bError* error flag will be set to ON. You can refer to ErrorCode for troubleshooting.

- **Programming Example: Incremental encoder**

  1. Set **CH1 Input Interface** in **AS02HC-A Parameters** to **Pulse Input**, set **Pulse Type** to **A/B phase (2x)**, and set **Counter Type** to **Ring counter** as shown in the following figure:



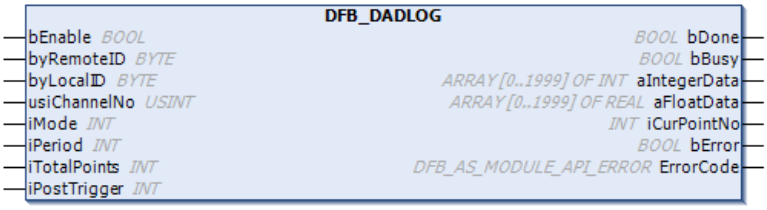
  2. Set *udiPulsePerRev* to 1000, set *iSamplingTime* to 1000, and set *iMovingAvgWindow* to 10.
  3. When setting M0=ON, the DHCCNT counter starts to count. At the same time, set the DHCMEAS parameters to the module, and start to measure frequency and rotation speed. The pulse number counted every 1000ms is displayed in *diFrequency* and *diRPM* **.**
  4. When the motor operation frequency is 5kHz, *diFrequency* shows 5kHz, and *diRPM* shows 300rev/min.



- **Supported products**
  - AX-3 series
- **Library**
  - DL_ASModuleAPI_AX3.library

6.20. DFB_DADLOG

DFB_DADLOG records the analog input module data.

FB/FC	Instruction	Graphic Expression	ST Language
FB	DFB_DADLOG		DFB_DADLOG(bEnable:= , byRemoteID:= , byLocalID:= , usiChannelNo:= , iMode:= , iPeriod:= , iTotalPoints:= , iPostTrigger:= , bDone=> , bBusy=> , aIntegerData=> , aFloatData=> , iCurPointNo=> , bError=> , ErrorCode=>);

- **Inputs**

Name	Function	Data Type	Setting Value (Default value)
bEnable	Run function block	BOOL	TRUE/FALSE (FALSE)
byRemoteID*	CPU or remote module	BYTE	0: CPU 1–15: remote module (0)
byLocalID	Expansion module ID	BYTE	0–31 (0)
usiChannelNo	Specified channel number	USINT	1–2 (1)
iMode	Output mode setting	INT	0–3 (0)
iPeriod	Speed fetch cycle time	INT	10–1000 (1000)
iTotalPoints	total number of records	INT	Refer to the description in the Function description (1)
iPostTrigger	Number of records after triggering	INT	Refer to the description in the Function description (0)

***Note**: Only support mode 0.

- **Outputs**

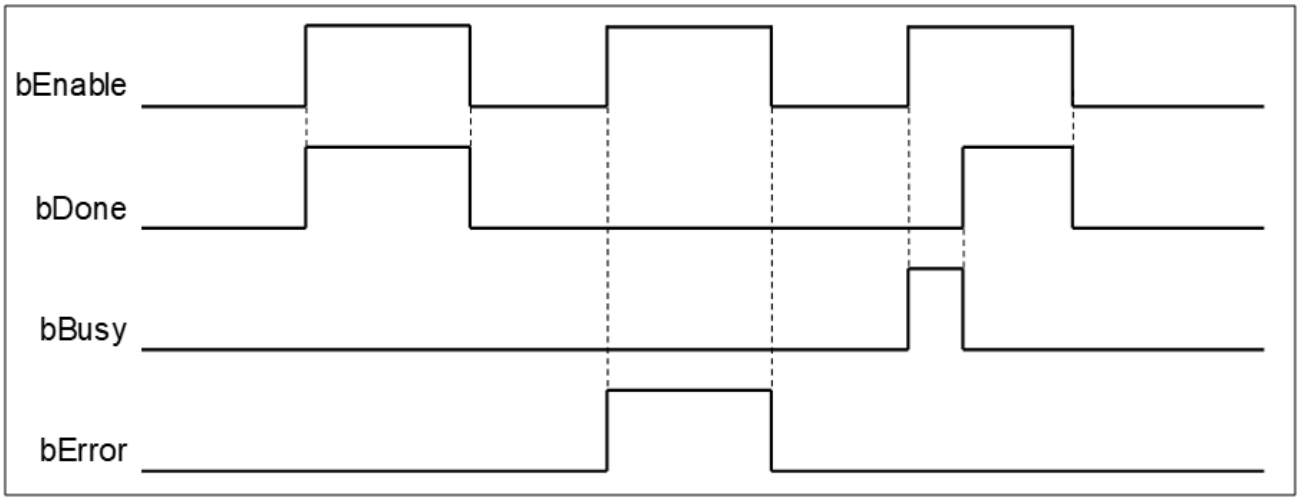
Name	Function	Data Type	Output Range (Default value)
bDone	When the pulse output is done	BOOL	TRUE/FALSE (FALSE)
bBusy	The FB is running	BOOL	TRUE/FALSE (FALSE)

Name	Function	Data Type	Output Range (Default value)
aIntegerData	The device that stores the record value when Format is Integer	ARRAY [0–1999] OF INT	Refer to chapter 15.2.1 of AS Module Manual (0).
aFloatData	The device that stores the recorded value when Format is Floating	ARRAY [0–1999] OF REAL	Refer to chapter 15.2.1 of AS Module Manual (0).
iCurPointNo	Accumulated record points	INT	0–2000 (0)
bError	FB instruction error flag	BOOL	TRUE/FALSE (FALSE)
ErrorCode	Error Code	DFB_AS_MODULE_API_ERROR	DFB_AS_MODULE_API_ERROR (DFB_NO_ERROR)

- **Output Updating Timing**

Name	Timing for shifting to TRUE	Timing for shifting to FALSE
bDone	When the running of FB is complete	When *bExecute* shifts to FALSE
bBusy	When the running of FB starts	When *bExecute* shifts to FALSE
aIntegerData	Keep updating until the record is complete.	When *bExecute* shifts to FALSE
aFloatData	Keep updating until the record is complete.	When *bExecute* shifts to FALSE
iCurPointNo	Keep updating until the record is complete.	When *bExecute* shifts to FALSE
bError	When an error occurs during the running or there are incorrect input values	When *bExecute* shifts to FALSE
ErrorCode		

- **Timing Diagram**



- **Funtion**
  - This function is only supported by the AX-3 series firmware version V1.0.3 or above.
  - This instruction is used for analog input modules (AS04AD–A, AS06XA–A, AS02ADH–A). It is used to enable/disable the recording function, and send the recorded data from the module to the specified aIntegerData or aFloatData (depends on the Format setting in the module Parameters page).
  - The specifications of the Record Period and the number of points corresponding to each Model are described in the following table. For detailed usage, refer to the iMode parameter description.

Model	Record Period	Number of Records
AS04AD–A AS06XA–A	Fixed period mode: The setting range is 1–100, and the unit time is fixed at 10ms.	Fixed to 500 records
AS02ADH–A	Fixed period (Fixed period) mode, trigger start types fixed period (Fixed period + Trigger Start) mode, trigger position designation (Fixed period + Trigger position Assign) recording mode: Setting range 1–32000, selectable time unit 20us/40us/80us (**Note**: The time unit is the sampling period set by HWCONFIG) Point Logging mode: One point is recorded every time the external input point is triggered, and there is no fixed period; the trigger timing of the external input point is set by HWCONFIG. Channel 1 triggers when X0.0 shift to TRUE or FALSE Channel 2 triggers when X0,1 shift to TRUE or FALSE	Can 1–2000 records

- *byRemoteID* specifies the group number of the analog input module connected to the right side of the host or the right side of the remote module, the host number is 0, the number of the first remote module is 1, and so on, the maximum number of groups is 15.
- *byLocalID* specifies the module number. The sequence number of the modules connected to the right side of the host starts from 0, second module is 1, and so on. Regardless of any type of modules, they must be counted, and the maximum number of modules is limited to 32 units.
- *usiChannelNo* specifies the channel number. Channel 1 is numbered 1, channel 2 is numbered 2.
- *iMode* is recording mode. Modes supported by each Model are shown in the table below.

Applicable Model	iMode Value	Name	Recording Mode Description
AS04AD–A AS06XA–A	0	Fixed Period mode	Run fixed period records
AS02ADH–A	0	Fixed Period mode	Run fixed period records
	1	Fixed period + Trigger Start mode	Waiting for the trigger signal of the external input point. When the trigger is received, the fixed period recording is running immediately.
	2	Point Logging mode	No fixed period. One record is recorded every time the external input point is triggered.
	3	Fixed period + Trigger Start mode	The external trigger timing point can be specified to

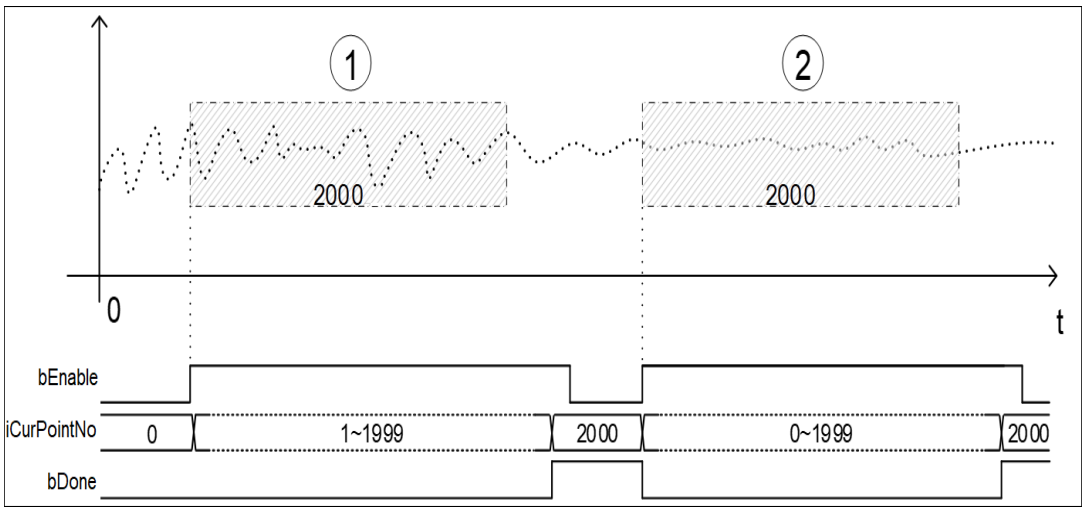
Applicable Model	iMode Value	Name	Recording Mode Descroption
			record the data recorded in the fixed period before / after.

1. Fixed period (Fixed period) mode:

   Set *iMode*=0, the command *bEnable* is turned on to Run the recording with the set Record Period. When the set total number of records is completed, the bDone flag will be automatically set to TRUE.

   Example:

   Set *iTotalPoints* = 2000



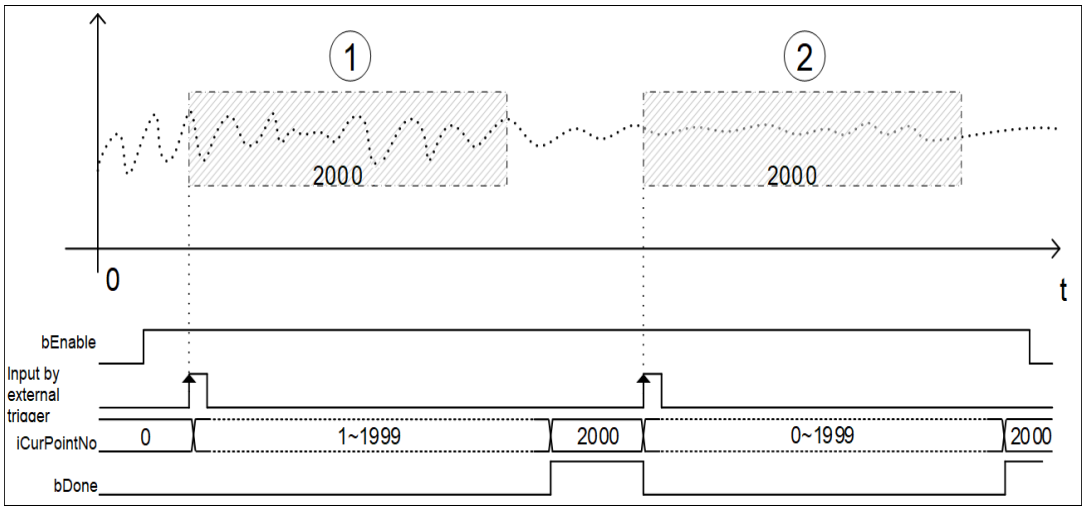
2. Fixed period + Trigger Start mode:

   Set *iMode*=1, turn on the command *bEnable* before starting the recording. When the external trigger input point is triggered, the recording will be Run with the set Record Period immediately, and the bDone flag will be automatically set to TRUE when the recording is completed. Any operation on the external trigger input point will not affect the recording until the set total number of records is completed; however, when the record number has been completed and the bDone flag is TRUE, the external input point can be re–triggered to start a new record. A round of records does not need to be closed and restarted by *bEnable*.

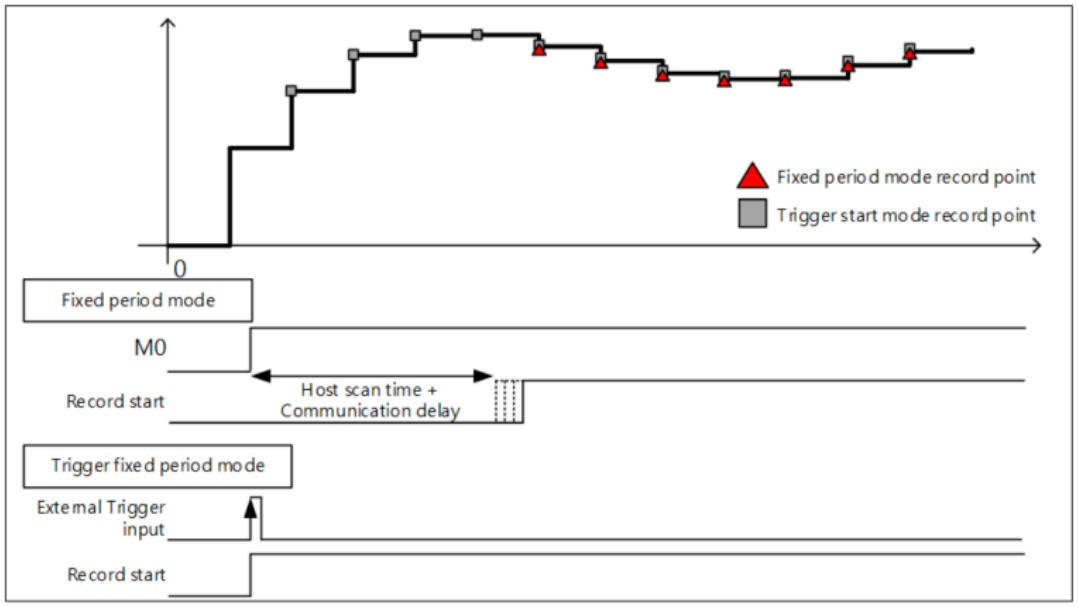
Record Channel	Corresponding to external trigger input signal source (Set the trigger timing of the external input point in the External Trigger Input of the module Parameters)
Channel 1	Triggers when X0.0 shift to TRUE or FALSE
Channel 2	Triggers when X0,1 shift to TRUE or FALSE

   Example:

   Set *iTotalPoints* = 2000. The trigger timing of the external input point is set as the rising edge trigger.



The fixed period + trigger start mode is similar to the fixed period mode, but the recording start timing of the fixed period mode will be affected by the host scan time and module communication time, causing delay, refer to below picture. In the Fixed period mode, it is assumed that M0 is the device that controls the DADLOG command *bEnable*. When M0 is turned OFF–>ON, the module does not start recording immediately, but with a small delay.



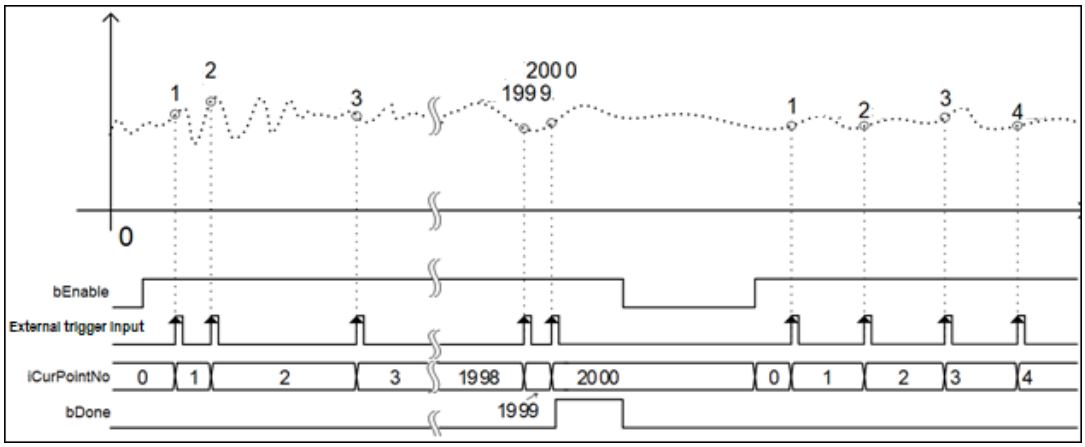
3. Point Logging Mode:

   Set *iMode*=2. Turn on the command *bEnable* before starting to record. Each time the external trigger input point is triggered, one record will be recorded. When the set total number of records is reached, the bDone flag will be automatically set to TRUE; if wish to continue recording after the bDone flag is TRUE, you must restart the command.

Record Channel	Corresponding to external trigger input signal source (Set the trigger timing of the external input point in the External Trigger Input of the module Parameters)
Channel 1	Triggers when X0.0 shift to TRUE or FALSE
Channel 2	Triggers when X0,1 shift to TRUE or FALSE

Example:

Set *iTotalPoints* = 2000, The trigger timing of the external input point is set as the rising edge.



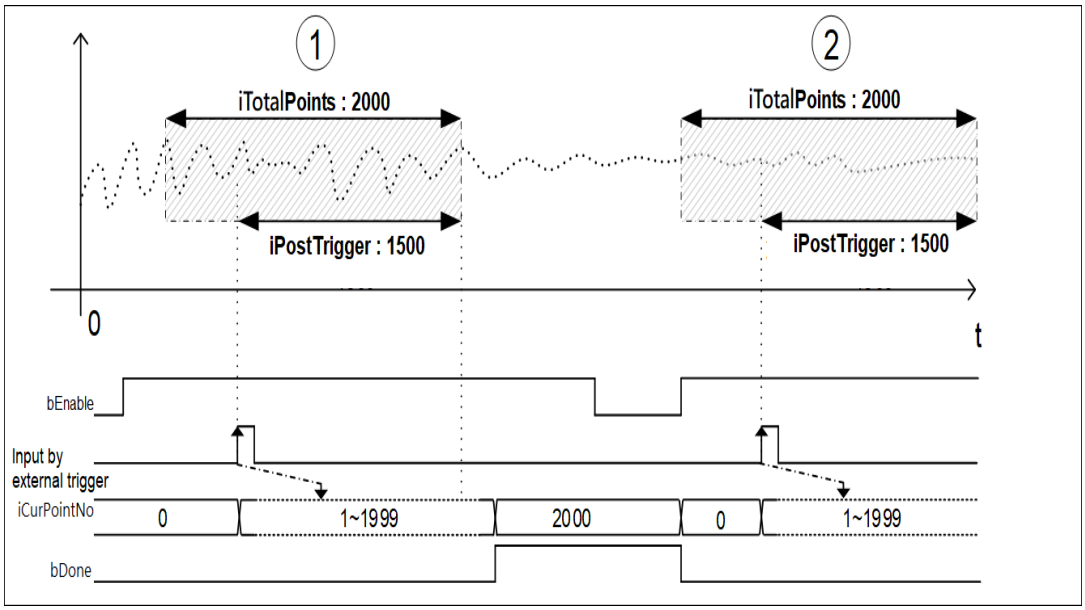
4. Fixed period + Trigger position Assign Mode:

   Set *iMode*=3, and set the *iTotalPoints* and *iPostTrigger* parameters. This mode is triggered by an external input point and records a specific number of strokes before/after the trigger according to the setting. After using the command *bEnable* to enable this recording mode, AS02ADH–A will start to wait for the external trigger input signal, and start sampling immediately after the signal is triggered. When the Number of Records reaches the set number of records, the bDone flag will be automatically set to TRUE. The iCurPointNo value is 0 before the trigger. After the trigger, the module starts to send the record data before the trigger to the host, so the iCurPointNo value will gradually catch up with the accumulated record number.

Record Channel	Corresponding to external trigger input signal source (Set the trigger timing of the external input point in the External Trigger Input of the module Parameters)
Channel 1	Triggers when X0.0 shift to TRUE or FALSE
Channel 2	Triggers when X0,1 shift to TRUE or FALSE

Example:

Set *iMode*=3, *iTotalPoints* = 2000**,** *iPostTrigger* = 1500, means that the position of the 501st point (*iTotalPoints* – *iPostTrigger*) will be the first data recorded by the current external trigger.



   a. *iPeriod* is set for Record Period, the description is as follows:

Model	iPeriod Range	iPeriod Unit
AS04AD–A	1–100	Fixed as 10ms, cannot be changed.

Model	iPeriod Range	iPeriod Unit
AS06XA–A		
AS02ADH–A	1–32000	The sampling cycle time must be set by AS02ADH–A Parameters, 20us, 40us and 80us can be selected. If "iMode=2: Point Logging Mode" is used, this *iPeriod* setting is invalid.

b. *iTotalPoints* is the total number of records. This parameter is only valid for AS02ADH–A, and can be set up to 2000 points; other models are fixed at 500 points regardless of the setting of this parameter.

Model	Total Recordm Setting Range
AS04AD–A AS06XA–A	The setting is invalid, fixed at 500.
AS02ADH–A	Can be set, 1–2000 records.

c. *iPostTrigger* is the points recorded after the trigger occurs. This parameter is used in combination with the total number of records *iTotalPoints* to record the data before and after the trigger. This parameter is only used in Fixed period + Trigger position Assign mode (iMode=3), and is invalid in other modes. Be cautious that this value should not be greater than the total number of iTotalPoints. If it exceeds, it will automatically use the *iTotalPoints* value as Number of Records.

Example: If set *iTotalPoints* = 100**,** *iPostTrigger* = 200. Since *iPostTrigger* is greater than the total number of records, *iPostTrigger* will be automatically regarded as 100, so only 100 records after the trigger will be recorded.

Model	iPostTrigger Setting Range
AS02ADH–A	0 –2000 records (Must not be greater than the total number of *iTotalPoints* records)

Example:

Set *iTotalPoints*= 100, *iPostTrigger*= 700, the 1000 records include the first 300 records and the last 700 records triggered by the external trigger input point.

- *aIntegerData* and *aFloatData* are specified arrays to store record values. (According to the Format setting in the parameters page of the module to determine which array to store in, the value in the other array will be cleared to 0).
- *iCurPointNo* is to display the number of records that the module has sent back to the host. When the record is in progress, the *iCurPointNo* value may not be displayed in a consistent value due to the influence of the scanning cycle.
- *bDone* is the flag of record completion. When *bDone* is OFF–>ON, it means that all record values have been transferred to the specified *aIntegerData* or *aFloatData* array. The *bDone* flag will be automatically initialized to OFF when *bEnable* changes from OFF–>ON.
- If the command is closed, the specified channel will stop updating the values in the right half of the command.
- When any error occurs during startup, the *bError* error flag will be set to ON. Refer to the error code of ErrorCode for troubleshooting.

- **Supported products**
  - AX-3 series
- **Library**
  - DL_ASModuleAPI_AX3.library

6.21. DFB_DADPEAK

DFB_DADPEAK records the Analog input module peak.

FB/FC	Instruction	Graphic Expression	ST LANGUAGE
FB	DFB_DADPEAK		DFB_DADPEAK(bEnable:= , byRemoteID:= , byLocalID:= , usiChannelNo := , bBusy=> , iMaxValue=> , iMinValue=> , rMaxValue=> , rMinValue=> , bError=> , ErrorCode=>);

- **Inputs**

Name	Function	Data Type	Setting Value (Default value)
bEnable	Run function block	BOOL	TRUE/FALSE (FALSE)
byRemoteID*	The CPU or remote module ID	BYTE	0: CPU 1–15: Remote module (0)
byLocalID	Expansion module ID	BYTE	0–31 (0)
usiChannelNo	Specify channel number	USINT	1–2 (1)

Note: Only support mode 0.

- **Outputs**

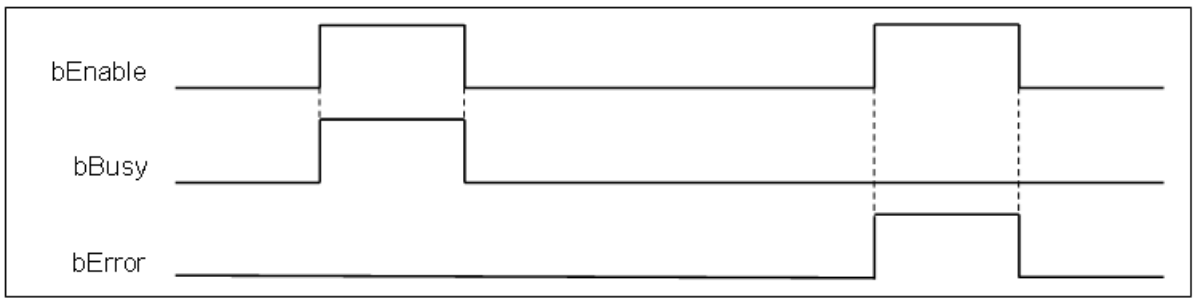
Name	Function	Data Type	Output Range (Default value)
bBusy	Indicates that the function block is running	BOOL	TRUE/FALSE (FALSE)
iMaxValue	Maximum value when Format is Integer	INT	Refer to AS Module Manual chapter 15.2.1 of (0)
iMinValue	Minimum value when Format is Integer	INT	Refer to AS Module Manual chapter 15.2.1 of (0)
rMaxValue	Maximum value when Format is Floating	REAL	Refer to AS Module Manual chapter 15.2.1 of (0)
rMinValue	Minimum value when Format is Floating	REAL	Refer to AS Module Manual chapter 15.2.1 of (0)

Name	Function	Data Type	Output Range (Default value)
bError	FB instruction error flag	BOOL	TRUE/FALSE (FALSE)
ErrorCode	Error Code	DFB_AS_MODULE_API_ERROR	DFB_AS_MODULE_API_ERROR (DFB_NO_ERROR)

- **Output Updating Timing**

Name	Timing for shifting to TRUE	Timing for shifting to FALSE
bBusy	When the running of FB starts	When *bExecute* shifts to FALSE
iMaxValue	Continuous update after *bEnable*	When *bExecute* shifts to FALSE
iMinValue	Continuous update after *bEnable*	When *bExecute* shifts to FALSE
rMaxValue	Continuous update after *bEnable*	When *bExecute* shifts to FALSE
rMinValue	Continuous update after *bEnable*	When *bExecute* shifts to FALSE
bError	When an error occurs during the running or there are incorrect input values	When *bExecute* shifts to FALSE
ErrorCode		

- **Timing Diagram**



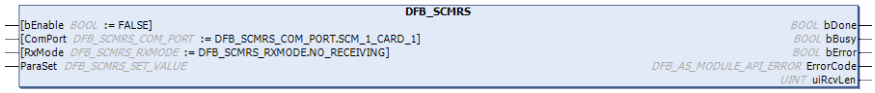
- **Funtion**
  - This Function is only supported with AX-3 series firmware version 1.0.3 or later.
  - AX-3 series firmware version 1.0.7 or later supports AS08AD_B and AS08AD_C modules (v1.0.2 or later).
  - The DADPEAK command is a dedicated command for analog input modules (AS04AD–A, AS08AD–B, AS08AD–C, AS06XA–A, AS02ADH–A), and its Function is to enable/disable the module to record the peak value.
  - *byRemoteID* specifies the group number of the analog input module connected to the right side of the host or the right side of the remote module, the host number is 0, the number of the first remote module is 1, and so on, the maximum number of groups is 15.
  - *byLocalID* specifies the module number, the sequence number of the modules connected to the right side of the host, the number of the first module is 0, the number of the second module is 1, and so on, regardless of any type of modules, they must be counted, and the maximum number of modules is limited. for 32 units.
  - *usiChannelNo* specifies the number of the counter channel to be controlled, the number of Channel 1 is 1, and the number of Channel 2 is 2.
  - According to the Format setting in the module Parameters page, determine whether the maximum and minimum values are placed in *iMaxValue* and *iMinValue* or in *rMaxValue* and *rMinValue*. The maximum and minimum values of another Data Type will be cleared to 0.

- The following describes the maximum and minimum values of INT or REAL type represented by *MaxValue* and *MinValue*.
- *MaxValue* and *MinValue* are the maximum and minimum values respectively. When *bEnable* is changed from OFF–>ON, *MaxValue* and *MinValue* will be initialized to the latest measurement value, and then the peak value recording Function will start, and the channel will continue to detect the maximum and minimum values. value and update to *MaxValue* and *MinValue*.
- If *bEnable* changes from ON to OFF, it means that the peak value recording function is turned off. At this time, the content values of *MaxValue* and *MinValue* will remain unchanged and will not be updated.
- When any error occurs during the startup process, the *bError* flag will be set to ON. Refer to the error code of *ErrorCode* for troubleshooting.

- **Supported Devices**
  - AX-3 series
- **Library**
  - DL_ASModuleAPI_AX3.library

6.22. DFB_ SCMRS

DFB_ SCMRS sends and receives communication data via the COM port.

FB/FC	Instruction	Graphic Expression	ST Language
FB	DFB_ SCMRS		DFB_SCMRS(bEnable:= , ComPort:= , RxMode:= , ParaSet:= , bDone=> , bBusy=> , bError=> , ErrorCode=> , uiRcvLen=>);

- **Inputs**

Name	Function	Data Type	Setting Value (Default value)
bEnable	Run the function block.	BOOL	TRUE/FALSE (FALSE)
ComPort	COM port number	DFB_SCMRS_COM_PORT	(SCM_1_CARD_1)
RxMode*1	Receiving mode	DFB_SCMRS_RXMODE	(NO_RECEIVING)
ParaSet*2	COM port parameters	DFB_SCMRS_SET_VALUE	(COMRS_SET_VALUE)

Note:

1. DFB_SCMRS_RXMODE: STRUCT

Name	Desicription
NO_RECEIVING(0)	No receiving data mode: After the packet is sent, the receiving task is complete. Then the completion flag is set to TRUE.
DISCONTINUOUS_TIME(1)	Discontinuous time mode: When the interval between receiving packets is longer than the specified time, the receiving task is complete. Then the completion flag is set to TRUE. The discontinuous time for receiving the packet can be set via *ParaSet.uiDiscontinuousTime*.
SPECIFIC_END_CHAR_1(2)	Specific end character (1) mode: When a specific character is received, the data receiving is complete. Then the completion flag is set to TRUE.

Name	Desicription
	The end character can be set via *ParaSet.pSpecificEndChar*.
SPECIFIC_END_CHAR_2(3)	Specific end character (2) mode: When two consecutive specific characters are received, the data receiving is complete. Then the completion flag is set to TRUE. The end character can be set via *ParaSet.pSpecificEndChar*.
SPECIFIC_START_CHAR_AND_ DISCONTINUOUS_TIME(4)	Specific start character and discontinuous time mode: When a specific character is received, starts to receive the packet, and when the packet interval is longer than the specified time, the receiving task is complete. The start character can be configured via *ParaSet.pSpecificStartChar* and the discontinuous time can be set via *ParaSet.uiDiscontinuousTime*.
SPECIFIC_START_CHAR_AND_ SPECIFIC_END_CHAR(5)	Specific start character and end character mode: When a specific character is received, starts to receive the packet. The receiving task is not complete until receive an end character. The start character can be set via *ParaSet.pSpecificStartChar*. The end character can be set via *ParaSet.pSpecificEndChar*.
SPECIFIC_LENGTH(6)	Specific length mode: A specific quantity of data is received, and the receiving task is complete. The packet length can be set via *ParaSet.uiSpecificRxLen*.
SPECIFIC_END_CHAR_AND_INTERRUPT (7)	Specific end character and interrupt mode: The receiving task is not complete until it receives a specific character. At the same time, an external interrupt is generated. The end character can be set via *ParaSet.pSpecificEndChar*.
SPECIFIC_LENGTH_AND_INTERRUPT(8)	Specific length and interrupt mode: The receiving task is not complete until it receives a specific data or a specific length of package. At the same time, an external interrupt is generated. The packet length can be set via *ParaSet.uiSpecificRxLen*.
SPECIFIC_ END_CHAR _OR_ SPECIFIC_LENGTH(9)	Specific end character or specific length mode.

Name	Desicription
	The receiving task is not complete until it receives a specific quantity of data. At the same time, an external interrupt is generated. The end character can be set via *ParaSet.pSpecificEndChar,* and the packet length can be set via *ParaSet.uiSpecificRxLen*.

2. DFB_SCMRS_SET_VALUE: STRUCT

Name	Function	Data Type	Setting Value (Default value)
uiWriteLen	The length of packet to be sent (Unit: Byte)	UINT	0 –200 (0)
uiDiscontinuousTime	Packet interval (Unit: ms)	UINT	5–3,000 (5)
pSpecificStartChar	Memory address of the start character	POINTER TO BYTE	Memory address (0)
pSpecificEndChar	Memory address of the end character	POINTER TO BYTE	Memory address (0)
uiSpecificRxLen	Receiving length	UINT (Unit: Byte)	1–198 (1)
tTimeout	Communication time–out	TIME	T#0ms–T#3000ms (T#100ms) T#0ms: No time–out

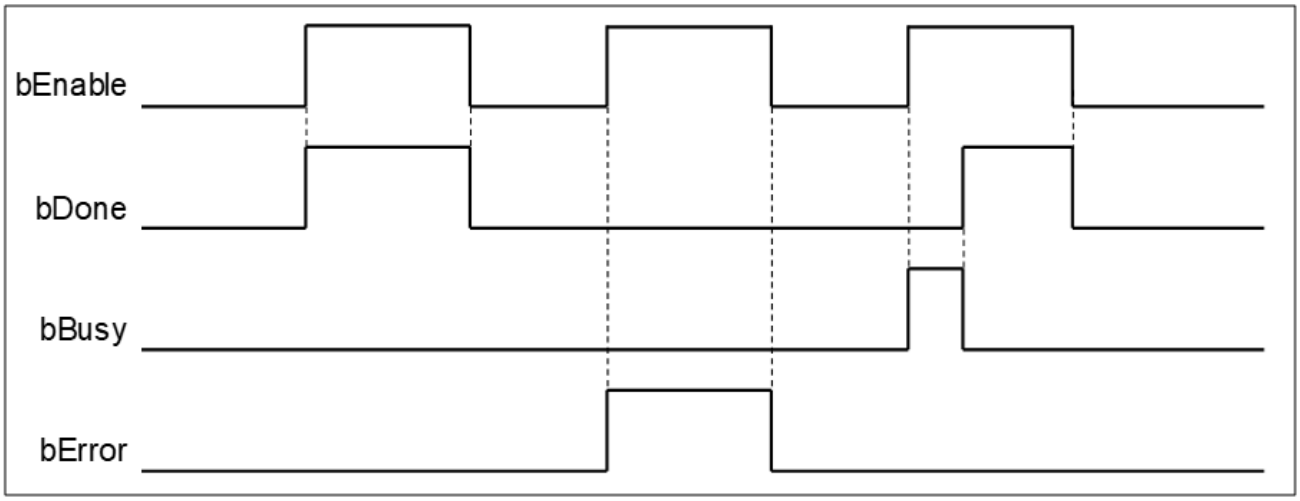
- **Outputs**

Name	Function	Data Type	Output Range(Default value)
bDone	FB is completed.	BOOL	TRUE/FALSE (FALSE)
bBusy	TRUE when the instruction is running	BOOL	TRUE/FALSE (FALSE)
bError	TRUE if an error occurs	BOOL	TRUE/FALSE (FALSE)
ErrorID	Indicates the error code if an error occurs.	DFB_AS_MODULE_API_ERROR	DFB_AS_MODULE_API_ERROR (DFB_NO_ERROR)
uiRcvLen	The length of the data received	UINT (Unit: Byte)	(0)

- **Output Updating Timing**

Name	Timing for shifting to TRUE	Timing for shifting to FALSE
bBusy	When *bEnable* is TRUE	When *bEnable* is FALSE
bError	When an error occurs during the running or there are incorrect input values	When *bEnable* is FALSE
ErrorID		
bDone	When *bEnable* is TRUE	When *bEnable* is FALSE

- **Timing Diagram**



- **Function**

  - This function requires AX-3 series firmware version 1.0.3 or later.
  - This function requires AS00SCM firmware version 2.08.00 or later.
  - Before using this function, the communication protocol corresponding to the SCM function card must be set to SCM RS.
  - *ComPort* is the communication port number. Up to four AS00SCM can be connected to the right side of the PLC. If the communication port range is exceeded, the FB will not receive any communication data.

	Card1 No.	Card2 No.
1st AS00SCM	1	2
2nd AS00SCM	11	12
3rd AS00SCM	21	22
4th AS00SCM	31	32

  - When the condition of ending receiving has the specific character, it is recommended to use ASCII–encoded communication data. If not, it is recommended to use the communication time–out as a mark to end.
  - Take the first AS00SCM CARD1 on the right side of AX-3 as an example. FunctionCardOutput(Slave)[0] is the length of the received data, and FunctionCardOutput(Slave)[0]+1–FunctionCardOutput(Slave)[0]+n is the stored received data. Suppose *RxMode* = SPECIFIC_END_CHAR_2 and *ParaSet.PSpecificEndChar* = 16#0D0A, the FB will store the data after FunctionCardOutput(Slave)[1] according to the order of receiving. The data will not be stopped to receive until the two consecutive ending characters are 16#0D and 16#0A. The length will be entered in FunctionCardOutput(Slave)[0], and then the completion flag will be set to Done.
  - When the receiving mode is 7, 8, and 9, the corresponding communication interrupt numbers generated by each communication port are as follows:

    32 modules can be connected to the right side of the AX-3 controller, and there are different interrupt numbers according to the location of the SCM, as shown in the following table:

The Nth position on the right	1st	2nd	…	31st	32nd
Card1 interrupt No.	I400	I401	…	I430	I431

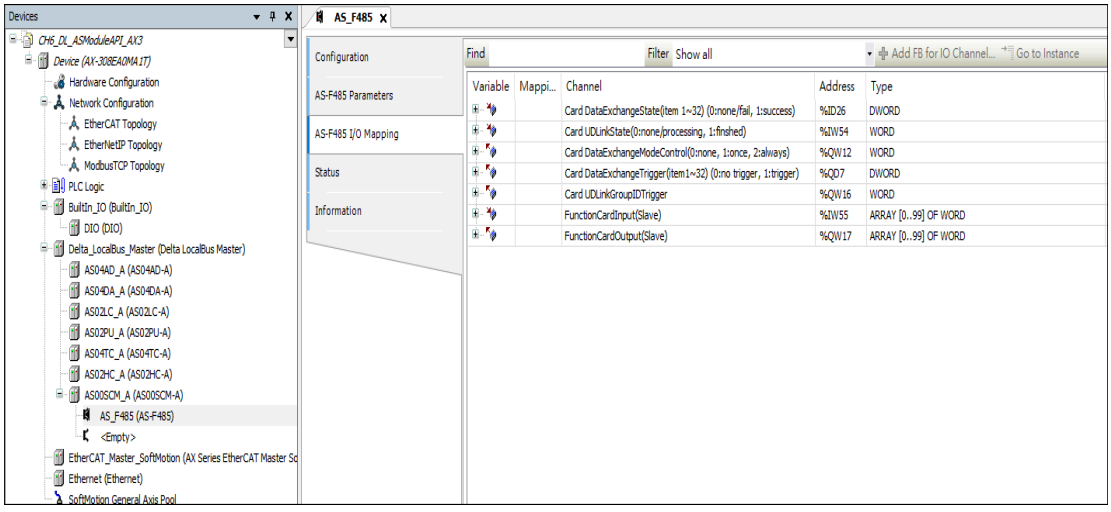
The Nth position on the right	1st	2nd	…	31st	32nd
Card2 interrupt No.	I432	I433	…	I462	I463

Note: Up to four SCM modules can be connected to the right side of the AX-3.

- **Programming Example**
  - **Example 1**
    - This example uses DFB_SCMRS and the communication receiving mode is NO_RECEIVING to send serial communication packets.

```
PROGRAM DFB_SCMRS_Example_1
VAR
    Output AT %QW17: ARRAY[0..199] OF BYTE;
    DFB_SCMRS_0 : DL_ASModuleAPI_AX3.DFB_SCMRS;
END_VAR

DFB_SCMRS_0.ParaSet.uiWriteLen :=8;
DFB_SCMRS_0.ParaSet.tTimeout := T#3000MS;
Output[0] := 0;
Output[1] := 1;
Output[2] := 2;
Output[3] := 3;
Output[4] := 4;
Output[5] := 5;
Output[6] := 6;
Output[7] := 7;

DFB_SCMRS_0 (
    bEnable:= ,
    ComPort:= DFB_SCMRS_COM_PORT. SCM_1_CARD_1,
    RxMode:= DFB_SCMRS_RXMODE. NO_RECEIVING,
    ParaSet:= ,
    bDone=> ,
    bBusy=> ,
    bError=> ,
    ErrorCode=> ,
    uiRcvLen=> );
```

    - Output variable address is based on the output address of the module. %QW7 is the start address in this example.

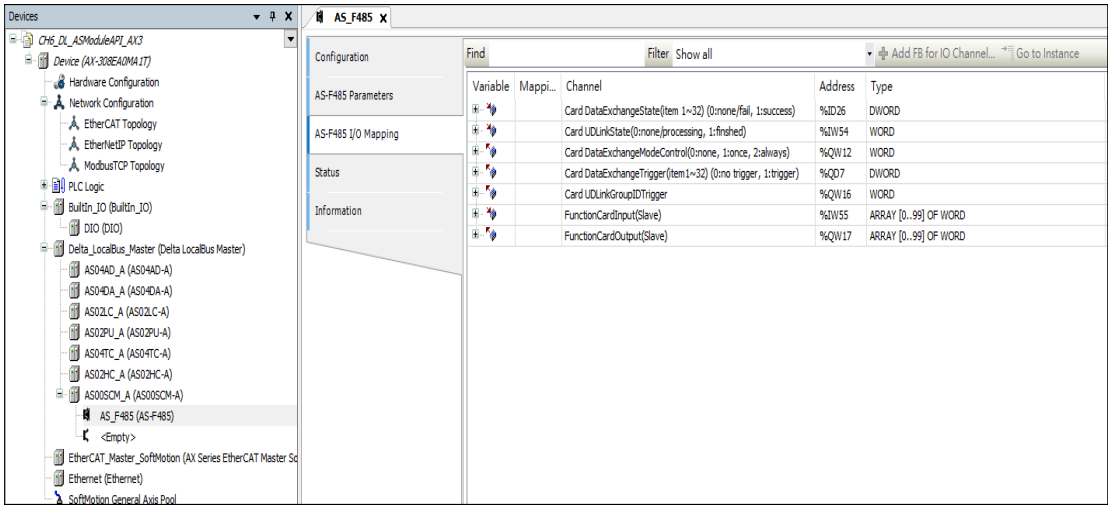


  - **Example 2**

- This example uses DFB_SCMRS and the communication receiving mode is SPECIFIC_START_CHAR_AND_SPECIFIC_END_CHAR to receive serial communication packets.

```
PROGRAM DFB_SCMRS_Example_3
VAR
    Output AT %QW17: ARRAY[0..199] OF BYTE;
    Input AT %IW55: ARRAY[0..199] OF BYTE;
    DFB_SCMRS_0 : DL_ASModuleAPI_AX3.DFB_SCMRS;
    EndChar: ARRAY[0..1] OF BYTE;

END_VAR

DFB_SCMRS_0.ParaSet.uiSpecificRxLen:=8;
DFB_SCMRS_0.ParaSet.tTimeout := T#3000MS;
EndChar[0] := 16#0D;
EndChar[1] := 16#0A;
DFB_SCMRS_0.ParaSet.pSpecificEndChar:= ADR(EndChar);

DFB_SCMRS_0 (
    bEnable:= ,
    ComPort:= DFB_SCMRS_COM_PORT. SCM_1_CARD_1,
    RxMode:= DFB_SCMRS_RXMODE.SPECIFIC_END_CHAR_2,
    ParaSet:= ,
    bDone=> ,
    bBusy=> ,
    bError=> ,
    ErrorCode=> ,
    uiRcvLen=> );
```

- Input and Output variable address is based on the input/output address of the module. %IW55 and %QW17 is the start address in this example.



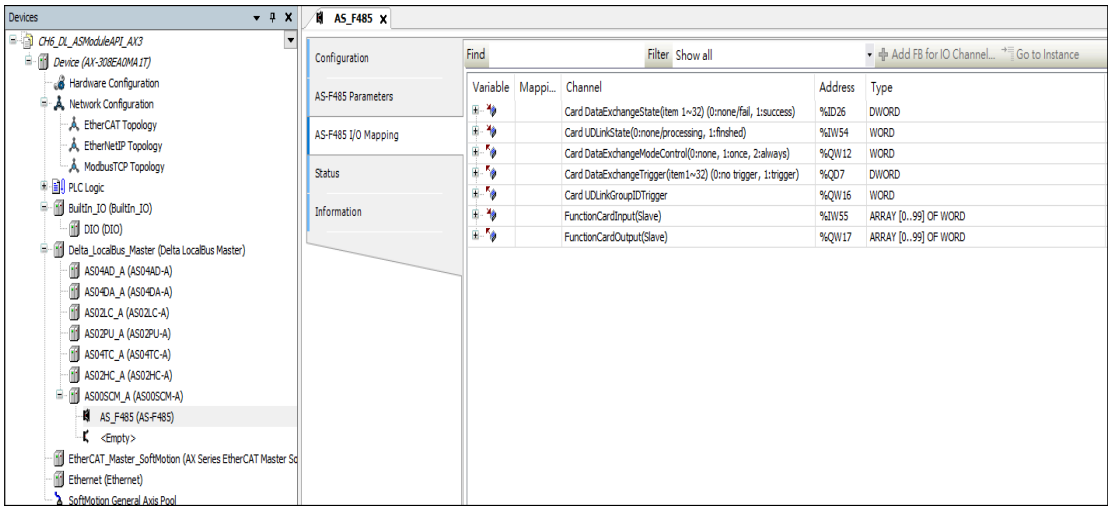
- **Example 3**

- This example uses DFB_SCMRS and the communication receiving mode is SPECIFIC_END_CHAR_2 to stop receiving packets when the two consecutive ending characters are 16#0D and 16#0A.

```
PROGRAM DFB_SCMRS_Example_3
VAR
    Output AT %QW17: ARRAY[0..199] OF BYTE;
    Input AT %IW55: ARRAY[0..199] OF BYTE;
    DFB_SCMRS_0 : DL_ASModuleAPI_AX3.DFB_SCMRS;
    EndChar: ARRAY[0..1] OF BYTE;
END_VAR

DFB_SCMRS_0.ParaSet.uiSpecificRxLen:=8;
```

```
DFB_SCMRS_0.ParaSet.tTimeout := T#3000MS;
EndChar[0] := 16#0D;
EndChar[1] := 16#0A;
DFB_SCMRS_0.ParaSet.pSpecificEndChar:= ADR(EndChar);

DFB_SCMRS_0 (
    bEnable:= ,
    ComPort:= DFB_SCMRS_COM_PORT. SCM_1_CARD_1,
    RxMode:= DFB_SCMRS_RXMODE.SPECIFIC_END_CHAR_2,
    ParaSet:= ,
    bDone=> ,
    bBusy=> ,
    bError=> ,
    ErrorCode=> ,
    uiRcvLen=> );
```

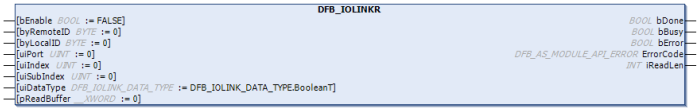
- Input and Output variable address is based on the utput address of the module. In this example, %IW55 and %QW17 is the start address.



- **Supported Devices**
  - AX-3 series
- **Library**
  - DL_ASModuleAPI_AX3.library

6.23. DFB_IOLINKR

DFB_IOLINKR reads parameter data of IO-Link devices.

FB/FC	Instruction	Graphic Expression	ST Language
FB	DFB_IOLINKR		DFB_IOLINKR(bEnable:= , byRemoteID := , byLocalID:= , uiPort:= , uiIndex:= , uiSubIndex:= , uiDataType:= , pReadBuffer := , bDone=> , bBusy=> , bError=> , ErrorCode=> , iReadLen=>);

- **Inputs**

Name	Function	Data Type	Setting Value (Default value)
bEnable	Run the function block.	BOOL	TRUE/FALSE (FALSE)
byRemoteID	Host or remote module number	BYTE	0: host 1–15: remote module (0)
byLocalID	Extended module number	BYTE	0–31
uiPort	Communication port number	UINT	1–4 (0)
uiIndex	Primary index of the parameter	UINT	0–65535 (0)
uiSubIndex	Subindex of the parameter	UINT	0–255 (0)
uiDataType	Parameter data type	DFB_IOLINK_DATA_TYPE*	(BooleanT)
pReadBuffer	Read data.		-

***Note**: DFB_IOLINK_DATA_TYPE: STRUCT

Name	Desicription
BooleanT(0)	Boolean, data length of 1 Byte
UIntegerT(1)	Unsigned integer, data length of 1, 2, 4, or 8 Bytes
IntegerT(2)	Signed integer, data lengths of 1, 2, 4, or 8 Bytes
Float32T(3)	Floating number, data length of 4 Bytes
StringT(4)	String of non-fixed length (1 to 232 Bytes), NULL (0x00) is considered as the end of the string.
OctetStringT(5)	Fixed-length string, data length (1 to 232 Bytes) is defined by the IODD attribute *fixedLength*.
TimeT(6)	Time, data length of 4 Bytes (Resolution: 1 second) + 4 Bytes (Resolution: 2^{-32} seconds)
TimeSpanT(7)	Time, data length of 8 Bytes (Resolution: 2^{-32} seconds)
ArrayT(8)	Read the entire array from subindex 0 of the parameter.
RecordT(9)	Read the entire record from subindex 0 of the parameter.

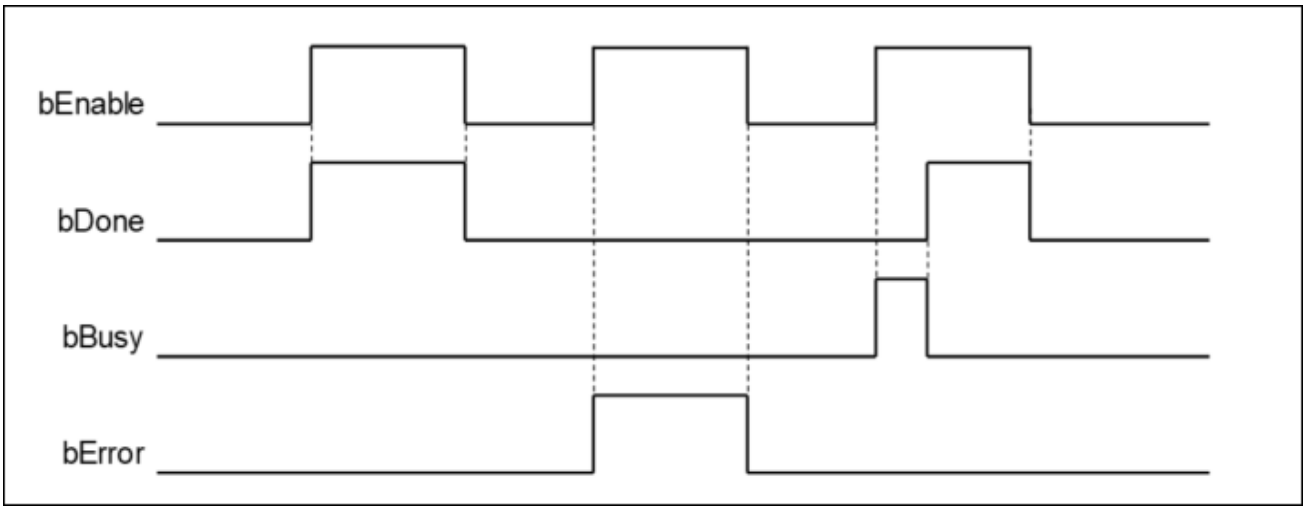
- **Outputs**

Name	Function	Data Type	Output Range(Default value)
bDone	FB is completed.	BOOL	TRUE/FALSE (FALSE)
bBusy	TRUE when the instruction is running	BOOL	TRUE/FALSE (FALSE)
bError	TRUE if an error occurs	BOOL	TRUE/FALSE (FALSE)
ErrorCode	Indicates the error code if an error occurs.	DFB_AS_MODULE_API_ERROR	DFB_AS_MODULE_API_ERROR (DFB_NO_ERROR)
iReadLen	Read the data length.	INT (Unit：Byte)	(0)

- **Output Updating Timing**

Name	Timing for shifting to TRUE	Timing for shifting to FALSE
bDone	When *bEnable* is TRUE	When *bEnable* is FALSE
bBusy	When *bEnable* is TRUE	When *bEnable* is FALSE
bError	When an error occurs during the running or there are incorrect input values	When *bEnable* is FALSE
ErrorCode		

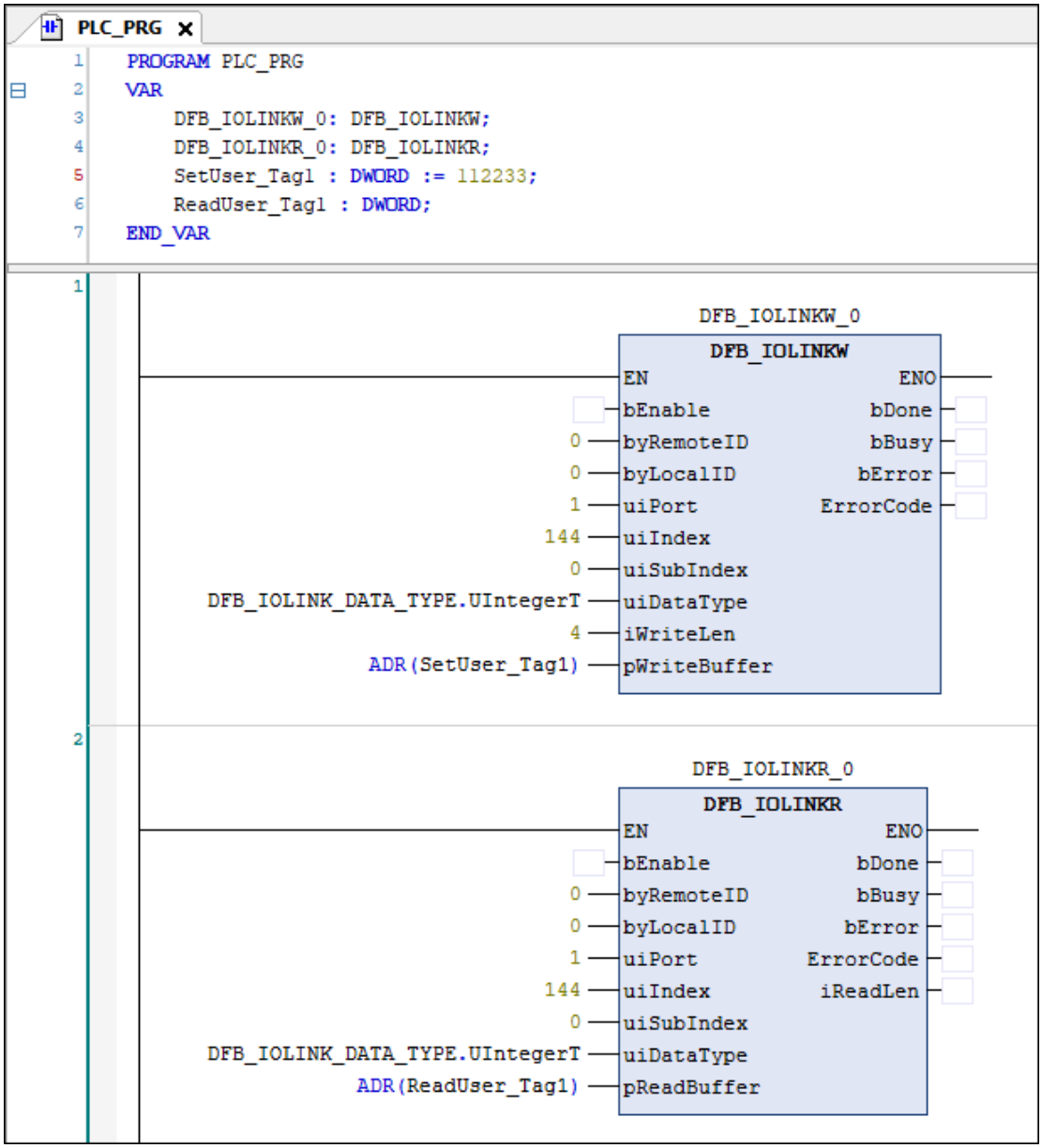
- **Timing Diagram**



- **Function**
  - This function requires AX-3 series firmware version 1.0.5 or later.
  - This function requires AS04SIL firmware version 1.00.00 or later.
  - You can refer to the IODD (IO-Link Device Description) file of the IO-Link device and use this instruction to read its parameters.
  - This instruction has no usage times restrictions. Each AS04SIL module can run only one communication instruction (IOLINKR, IOLINKW) at a time. Subsequently initiated instruction will not be run.
  - Communication with the IO-Link device might take up to 5 seconds. Do not stop or re-run the instruction during communication to avoid errors. To re-run the instruction, first stop the current instruction before restart. It is recommended to use the *bDone* or *bError* flag as logical conditions for stopping and restarting the instruction.
  - *uiPort* specifies the communication port number of the AS04SIL module. The valid range is 1–4, corresponding to Port #1–#4. If an invalid port number is entered, the Error flag will be set to ON.
  - *uiIndex*is the primary index number of the IO-Link device parameter to be read.
  - *uiSubIndex* is the subindex number of the IO-Link device parameter to be read. The valid range is 0–255. If the value exceeds this range, the Error flag will be set to ON.
  - *uiDateType* is the data type of the IO-Link device parameter. The correct data type must be configured to ensure proper data processing of the numerical data type. If the value exceeds the valid range, the Error flag will be set to ON.
- **Programming Example**
  - **Example**

    This example uses DFB_IOLINKW to write parameters and DFB_IOLINKR to read them.

- In this example, the AS04SIL module is mounted in Slot 1 (rightmost position) of the AX-3 controller, with its communication port set to 1. Therefore, *byLocalID* is set to 0 (default for the first module), and *uiPort* is set to 1.
- Based on the IO-Link device manual or the IODD file, the primary index (*uiIndex*) is 144, subindex (*uiSubIndex*) is 0, and data type (*uiDataType*) is UIntegerT(1).



- **Supported Devices**
  - AX-3 series
- **Library**
  - DL_ASModuleAPI_AX3.library

6.24. DFB_IOLINKW

DFB_IOLINKW writes host commands to the IO-Link device.

FB/FC	Instruction	Graphic Expression	ST Language
FB	DFB_IOLINKW		DFB_IOLINKW(bEnable:= , byRemoteID := , byLocalID:= , uiPort:= , uiIndex:= , uiSubIndex:= , uiDataType:= , iWriteLen:= , pWriteBuffer := , bDone=> , bBusy=> , bError=> , ErrorCode=>);

- **Inputs**

Name	Function	Data Type	Setting Value (Default value)
bEnable	Run the function block.	BOOL	TRUE/FALSE (FALSE)
byRemoteID	Host or remote module number	BYTE	0: host 1–15: remote module (0)
byLocalID	Extended module number	BYTE	0–31
uiPort	Communication port number	UINT	1–4 (0)
uiIndex	Primary index of the parameter	UINT	0–65535 (0)
uiSubIndex	Subindex of the parameter	UINT	0–255 (0)
uiDataType	Parameter data type	DFB_IOLINK_DATA_TYPE*	(BooleanT)
iWriteLen	Length of the data to be written	INT	1–232 (0)

Name	Function	Data Type	Setting Value (Default value)
pWriteBuffer	Write data.	__XWORD	-

***Note**: DFB_IOLINK_DATA_TYPE: STRUCT

Name	Desicription
BooleanT(0)	Boolean, data length of 1 Byte
UIntegerT(1)	Unsigned integer, data length of 1, 2, 4, or 8 Bytes
IntegerT(2)	Signed integer, data lengths of 1, 2, 4, or 8 Bytes
Float32T(3)	Floating number, data length of 4 Bytes
StringT(4)	String of non-fixed length (1 to 232 Bytes), NULL (0x00) is considered as the end of the string.
OctetStringT(5)	Fixed-length string, data length (1 to 232 Bytes) is defined by the IODD attribute *fixedLength*.
TimeT(6)	Time, data length of 4 Bytes (Resolution: 1 second) + 4 Bytes (Resolution: 2^{-32} seconds)
TimeSpanT(7)	Time, data length of 8 Bytes (Resolution: 2^{-32} seconds)
ArrayT(8)	Read the entire array from subindex 0 of the parameter.
RecordT(9)	Read the entire record from subindex 0 of the parameter.

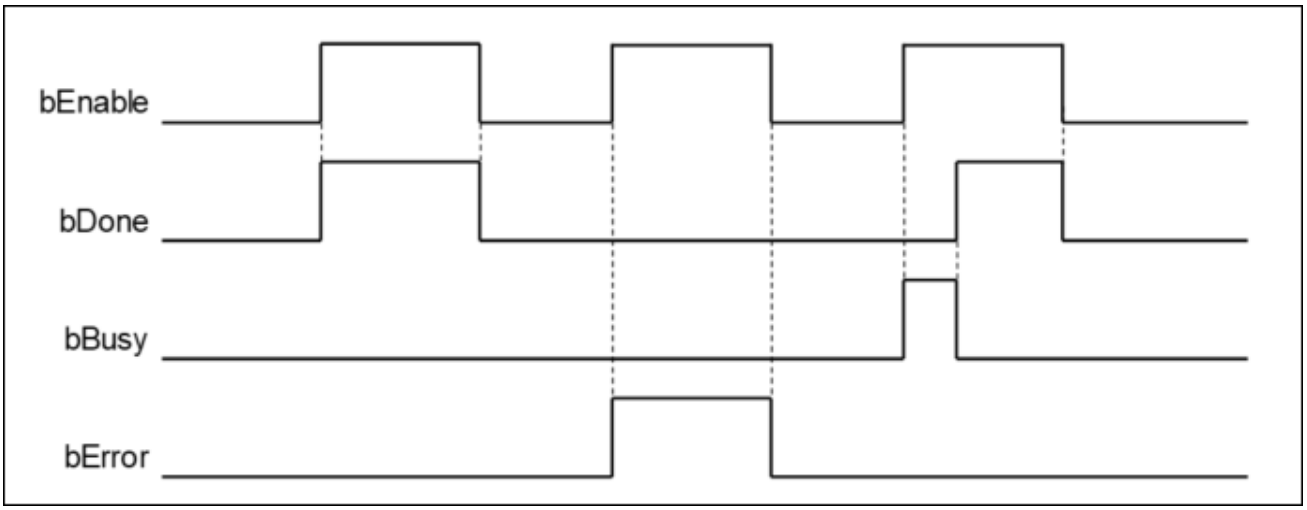
- **Outputs**

Name	Function	Data Type	Output Range(Default value)
bDone	FB is completed.	BOOL	TRUE/FALSE (FALSE)
bBusy	TRUE when the instruction is running	BOOL	TRUE/FALSE (FALSE)
bError	TRUE if an error occurs	BOOL	TRUE/FALSE (FALSE)
ErrorCode	Indicates the error code if an error occurs.	DFB_AS_MODULE_API_ERROR	DFB_AS_MODULE_API_ERROR (DFB_NO_ERROR)

- **Output Updating Timing**

Name	Timing for shifting to TRUE	Timing for shifting to FALSE
bDone	When *bEnable* is TRUE	When *bEnable* is FALSE
bBusy	When *bEnable* is TRUE	When *bEnable* is FALSE
bError	When an error occurs during the running or there are incorrect input values	When *bEnable* is FALSE
ErrorCode		

- **Timing Diagram**



- **Function**
  - This function requires AX-3 series firmware version 1.0.5 or later.
  - This function requires AS04SIL firmware version 1.00.00 or later.
  - You can refer to the IODD (IO-Link Device Description) file of the IO-Link device and use this instruction to write parameters.
  - This instruction has no usage times restrictions. Each AS04SIL module can run only one communication instruction (IOLINKR, IOLINKW) at a time. Subsequently initiated instruction will not be run.
  - Communication with the IO-Link device might take up to 5 seconds. Do not stop or re-run the instruction during communication to avoid errors. To re-run the instruction, first stop the current instruction before restart. It is recommended to use the *bDone* or *bError* flag as logical conditions for stopping and restarting the instruction.
  - *uiPort* specifies the communication port number of the AS04SIL module. The valid range is 1–4, corresponding to Port #1–#4. If an invalid port number is entered, the Error flag will be set to ON.
  - *uiIndex*is the primary index number of the IO-Link device parameter to be written.
  - *uiSubIndex* is the subindex number of the IO-Link device parameter to be written. The valid range is 0–255. If the value exceeds this range, the Error flag will be set to ON.
  - *uiDateType* is the data type of the IO-Link device parameter. The correct data type must be configured to ensure proper data processing of the numerical data type. If the value exceeds the valid range, the Error flag will be set to ON.
  - *iWriteLen* specifies the data length (in bytes) to be written of the IO-Link device parameter. The valid range is 1–232. If the input value exceeds this range, the Error flag will be set to ON.
- **Programming Example**

  Refer to the example of DFB_IOLINKR.
- **Supported Devices**
  - AX-3 series

- **Library**
  - DL_ASModuleAPI_AX3.library

6.25. Error Codes and Troubleshooting

Description	Cause of Error	Corrective Action
DFB_FROM_ERR_PARAMETER	Enter parameter error	Confirm if the input parameters are correct.
DFB_FROM_ERR_COMMUNICATION	CAN bus communication error	Confirm the error record.
DFB_FROM_ERR_CRADDR	CR address error	Check if the CR address is correct.
DFB_TO_ERR_PARAMETER	Enter parameter error	Confirm if the input parameters are correct.
DFB_TO_ERR_COMMUNICATION	CAN bus communication error	Confirm the error record.
DFB_TO_ERR_CRADDR	CR address error	Check if the CR address is correct.
DFB_DLCCAL_ERR_NOT_SUPPORT_MODULE	*byRemoteID* and *byLocalID* correspond to the module that is not the AS02LC module.	Confirm *byRemoteID*, *byLocalID*, and the corresponding module.
DFB_DLCCAL_ERR_INVALID_GROUP	*byRemoteID* input error	Confirm if *byRemoteID* input value is correct.
DFB_DLCCAL_ERR_INVALID_MODULE	*byLocalID* input error	Confirm if *byLocalID* input value is correct.
DFB_DLCCAL_ERR_INVALID_CHNO	*usiChannelNo* input error	Confirm if *usiChannelNo* input value is correct.
DFB_DLCCAL_ERR_INVALID_TWEIGHT	*aTWeight* input error	Confirm if *aTWeight* input value is correct.
DFB_DLCCAL_ERR_INVALID_TPOINT	*iTPoint* input error	Confirm if *iTPoint* input value is correct.
DFB_DLCCAL_ERR_MODULE_REPORTS_AN_ERROR	Module on the right reports error.	Check the error code reported in the Diagnosis Message in the Status of the module page, and check this error code in the AS Series module manual.
DFB_DLCWEI_ERR_NOT_SUPPORT_MODULE	*byRemoteID* and *byLocalID* correspond to the module that is not the AS02LC module.	Confirm *byRemoteID*, *byLocalID*, and the corresponding module.
DFB_DLCWEI_ERR_INVALID_GROUP	*byRemoteID* input error	Confirm if *byRemoteID* input value is correct.
DFB_DLCWEI_ERR_INVALID_MODULE	*byLocalID* input error	Confirm if *byLocalID* input value is correct.
DFB_DLCWEI_ERR_INVALID_CHNO	*usiChannelNo* input error	Confirm if *usiChannelNo* input value is correct.
DFB_DLCWEI_ERR_INVALID_STABLE	*rStable* input error	Confirm if *rStable* input value is correct.
DFB_DLCWEI_ERR_MODULE_REPORTS_AN_ERROR	Module on the right reports error.	Check the error code reported in the Diagnosis Message in the Status of the module page, and check this error code in the AS Series module manual.

Description	Cause of Error	Corrective Action
DFB_DPUCONF_ERR_INVALID_MODULE	*byLocalID* input error	Confirm if *byLocalID* input value is correct.
DFB_DPUCONF_ERR_NOT_SUPPORT_MODULE	*byLocalID* and *iAxis* correspond to the module that is not the PU module.	Confirm *byLocalID*, *iAxis*, and the corresponding module.
DFB_DPUCONF_ERR_INVALID_AXIS	*iAxis* input error	Confirm if *iAxis* input value is correct.
DFB_DPUCONF_ERR_INVALID_MODE	*iMode* input error	Confirm if *iMode* input value is correct.
DFB_DPUCONF_ERR_INVALID_SSPEED	*iStartSpeed* input error	Confirm if *iStartSpeed* input value is correct.
DFB_DPUCONF_ERR_INVALID_ATIME	*iAccTime* input error	Confirm if *iAccTime* input value is correct.
DFB_DPUCONF_ERR_INVALID_DTIME	*iDecTime* input error	Confirm if *iDecTime* input value is correct.
DFB_DPUCONF_ERR_INVALID_MSPEED	*diMaxSpeed* input error	Confirm if *diMaxSpeed* input value is correct.
DFB_DPUCONF_ERR_INVALID_ZNO	*iZ_no* input error	Confirm if *iZ_no* input value is correct.
DFB_DPUCONF_ERR_INVALID_OFFSET	*iOffset* input error	Confirm if *iOffset* input value is correct.
DFB_PUSTAT_ERR_INVALID_MODULE	*byLocalID* input error	Confirm if *byLocalID* input value is correct.
DFB_PUSTAT_ERR_NOT_SUPPORT_MODULE	*byLocalID* and *iAxis* correspond to the module that is not the PU module.	Confirm *byLocalID*, *iAxis*, and the corresponding module.
DFB_PUSTAT_ERR_INVALID_AXIS	*iAxis* input error	Confirm if *iAxis* input value is correct.
DFB_DPUPLS_DPUDRI_DPUDRA_ERR_INVALID_MODULE	*byLocalID* input error	Confirm if *byLocalID* input value is correct.
DFB_DPUPLS_DPUDRI_DPUDRA_ERR_NOT_SUPPORT_MODULE	*byLocalID* and *iAxis* correspond to the module that is not the PU module.	Confirm *byLocalID*, *iAxis*, and the corresponding module.
DFB_DPUPLS_DPUDRI_DPUDRA_ERR_INVALID_AXIS	*iAxis* input error	Confirm if *iAxis* input value is correct.
DFB_DPUPLS_DPUDRI_DPUDRA_ERR_INVALID_TARSPEED	*diTarSpeed* input error	Confirm if *diTarSpeed* input value is correct.
DFB_DPUPLS_DPUDRI_DPUDRA_ERR_POSITIVELIMIT_EXCEEDED	Exceed the positive limit position setting	Close the function block, set to run to the opposite direction and restart.
DFB_DPUPLS_DPUDRI_DPUDRA_ERR_NEGATIVELIMIT_EXCEEDED	Exceed the negative limit position setting	Close the function block, set to run to the opposite direction and restart.
DFB_DPUZRN_ERR_INVALID_MODULE	*byLocalID* input error	Confirm if *byLocalID* input value is correct.
DFB_DPUZRN_ERR_NOT_SUPPORT_MODULE	*byLocalID* and *iAxis* correspond to the module that is not the PU module.	Confirm *byLocalID*, *iAxis*, and the corresponding module.

Description	Cause of Error	Corrective Action
DFB_DPUZRN_ERR_INVALID_AXIS	*iAxis* input error	Confirm if *iAxis* input value is correct.
DFB_DPUZRN_ERR_INVALID_MODE	*iMode* input error	Confirm if *iMode* input value is correct.
DFB_DPUZRN_ERR_INVALID_TARSPEED	*diTarSpeed* input error	Confirm if *diTarSpeed* input value is correct.
DFB_DPUZRN_ERR_INVALID_JOGSPEED	*iJogSpeed* input error	Confirm if *iJogSpeed* input value is correct.
DFB_DPUJOG_ERR_INVALID_MODULE	*byLocalID* input error	Confirm if *byLocalID* input value is correct.
DFB_DPUJOG_ERR_NOT_SUPPORT_MODULE	*byLocalID* and *iAxis* correspond to the module that is not the PU module.	Confirm *byLocalID*, *iAxis*, and the corresponding module.
DFB_DPUJOG_ERR_INVALID_AXIS	*iAxis* input error	Confirm if *iAxis* input value is correct.
DFB_DPUJOG_ERR_INVALID_JOGSPEED	*diJogSpeed* input error	Confirm if *diJogSpeed* input value is correct.
DFB_DPUJOG_ERR_POSITIVELIMIT_EXCEEDED	Exceed the positive limit position setting	Close the function block, set to run to the opposite direction and restart.
DFB_DPUJOG_ERR_NEGATIVELIMIT_EXCEEDED	Exceed the negative limit position setting	Close the function block, set to run to the opposite direction and restart.
DFB_DPUMPG_ERR_INVALID_MODULE	*byLocalID* input error	Confirm if *byLocalID* input value is correct.
DFB_DPUMPG_ERR_NOT_SUPPORT_MODULE	*byLocalID* and *iAxis* correspond to the module that is not the PU module.	Confirm *byLocalID*, *iAxis*, and the corresponding module.
DFB_DPUMPG_ERR_BEUSED_BY_DPUCNT	The pin is currently being used by the DPUCNT function block.	After the DPUCNT function block is closed, re–trigger DPUMPG.
DFB_DPUMPG_ERR_INVALID_AXIS	*iAxis* input error	Confirm if *iAxis* input value is correct.
DFB_DPUCNT_ERR_INVALID_MODULE	*byLocalID* input error	Confirm if *byLocalID* input value is correct.
DFB_DPUCNT_ERR_NOT_SUPPORT_MODULE	The module corresponding to *byLocalID* is not the PU module.	Confirm *byLocalID* and the corresponding module.
DFB_DPUCNT_ERR_BEUSED_BY_OTHER	The pin is currently being used by other function blocks.	After other function blocks are closed, re–trigger DPUCNT.
DFB_DMPID_ERR_NOT_SUPPORT_MODULE	Module does not support this instruction.	Confirm if the module is the TC module.
DFB_DMPID_ERR_INVALID_GROUP_OR_MODULE_ID	Group number or module number setting error	Confirm if the input pin is in the correct range.

Description	Cause of Error	Corrective Action
DFB_DMPID_ERR_COMMUNICAION	No response from module, communication time–out	Check if the module in the device area is properly connected.
DFB_DMPID_ERR_NO_CHANNEL	Channel setting error	Confirm if the input parameter is correct.
DFB_DMPID_ERR_CHANNEL_IS_RUNNING_PID	The channel is running the PID function, and repeating designation is not allowed.	Confirm if the PID function is open and is been Run.
DFB_DHCCNT_ERR_NOT_SUPPORT_MODULE	Module does not support this instruction.	Confirm if the module is the HC module.
DFB_DHCCNT_ERR_INVALID_INPUT_VALUE_TO_HC_MODULE	When updating parameter to the module, module response error.	Confirm if the input pin is in the correct range.
DFB_DHCCNT_ERR_COMMUNICAION	No response from module, communication time–out	Check if the module in the device area is properly connected.
DFB_DHCCNT_ERR_HC_MODULE_CONFIG_ERROR	Module configuration setting error	Confirm if the setting parameter of the input device is correct.
DFB_DHCCNT_ERR_NO_CHANNEL	HC module does not have this counter channel.	Confirm if the used channel number is the module support number.
DFB_DHCCNT_ERR_INTERFACE_OF_ CHANNEL_IS_DISABLE	Input interface is not selected, instruction operation is not allowed.	Confirm if the channel input interface enable the corresponding functions.
DFB_DHCCNT_ERR_INVALID_ACTION _VALUE	*usiAction* value is invalid.	Confirm if the input pin *usiAction* is correct.
DFB_DHCCNT_ERR_CHANNEL_IS _RUNNING_CNT	The module channel is running the counting function, and repeating designation is not allowed.	Confirm if the HC counting function is enabled and is running.
DFB_DHCCAP_ERR_NOT_SUPPORT _MODULE	Module does not support this instruction.	Confirm if the module is the HC module.
DFB_DHCCAP_ERR_INVALID_INPUT _VALUE_TO_HC_MODULE	When updating parameters to the module, module response error.	Confirm if the input pin is in the correct range.
DFB_DHCCAP_ERR_COMMUNICAION	No response from module, communication time–out	Check if the module in the device area is properly connected.
DFB_DHCCAP_ERR_HC_MODULE _CONFIG_ERROR	Module configuration setting error	Confirm if the setting parameter of the input device is correct.
DFB_DHCCAP_ERR_NO_CHANNEL	HC module does not have this counter channel.	Confirm if the used channel number is the module support number.

Description	Cause of Error	Corrective Action
DFB_DHCCAP_ERR_INTERFACE_OF_CHANNEL_IS_DISABLE	Input interface is not selected, instruction operation is not allowed.	Confirm if the channel input interface enable the corresponding functions.
DFB_DHCCAP_ERR_INVALID_TRGSEL_VALUE	*byTrgSel* value is invalid.	Confirm if input pin *byTrgSel* is correct.
DFB_DHCCAP_ERR_CHANNEL_IS_RUNNING_CAP	The module channel is running the capture function, and repeating designation is not allowed.	Confirm if the HC capture function is enabled and is running.
DFB_HCDO_ERR_INVALID_MODULE	*byLocalID* input error	Confirm the input value is correct.
DFB_HCDO_ERR_NOT_SUPPORT_MODULE	*byLocalID* corresponds to a module that is not the HC module.	Confirm the module to which the function block corresponds.
DFB_HCDO_ERR_BEUSED_BY_OTHER_DFB	HC module is current being used by other function blocks.	After other function blocks are closed, re–trigger HCDO.
DFB_DHCCMP_ERR_INVALID_MODULE	*byLocalID* input error	Confirm the input value is correct.
DFB_DHCCMP_ERR_NOT_SUPPORT_MODULE	*byLocalID* corresponds to a module that is not the HC module.	Confirm the module to which the function block corresponds.
DFB_DHCCMP_ERR_INVALID_CHNO	*usiChannelNo* input error	Confirm the input value is correct.
DFB_DHCCMP_ERR_INVALID_ACTION1	*iActionValue1* input error	Confirm the input value is correct.
DFB_DHCCMP_ERR_INVALID_YNO1	*iY_OutputNo1* input error	Confirm the input value is correct.
DFB_DHCCMP_ERR_INVALID_ACTION2	*iActionValue2* input error	Confirm the input value is correct.
DFB_DHCCMP_ERR_INVALID_YNO2	*iY_OutputNo2* input error	Confirm the input value is correct.
DFB_DHCCMP_ERR_INVALID_INTERRUPTNO1	*iInterruptNo1* input error	Confirm the input value is correct.
DFB_DHCCMP_ERR_INVALID_INTERRUPTNO2	*iInterruptNo2* input error	Confirm the input value is correct.
DFB_DHCCMPT_ERR_INVALID_MODULE	*byLocalID* input error	Confirm the input value is correct.
DFB_DHCCMPT_ERR_NOT_SUPPORT_MODULE	*byLocalID* corresponds to a module that is not the HC module.	Confirm the module to which the function block corresponds.
DFB_DHCCMPT_ERR_INVALID_CHNO	*usiChannelNo* input error	Confirm the input value is correct.
DFB_DHCCMPT_ERR_INVALID_COMPARELENGTH	*iCompareLength* input error	Confirm the input value is correct.
DFB_DHCCMPT_ERR_INVALID_COMPARETABLE	*aCompareValue* input error	Confirm the input value is correct.

Description	Cause of Error	Corrective Action
DFB_DHCCMPT_ERR_INVALID_ACTIONTABLE	*aAction* input error	Confirm the input value is correct.
DFB_DHCCMPT_ERR_INVALID_YNOTABLE	*aY_OutputNo* input error	Confirm the input value is correct.
DFB_DHCCMPT_ERR_INVALID_INTERRUPTNOTABLE	*aInterruptNo* input error	Confirm the input value is correct.
DFB_DHCMEAS_ERR_INVALID_MODULE	*byLocalID* input error	Confirm the input value is correct.
DFB_DHCMEAS_ERR_NOT_SUPPORT_MODULE	*byLocalID* corresponds to a module that is not the HC module.	Confirm the module to which the function block corresponds.
DFB_DHCMEAS_ERR_INVALID_CHNO	*usiChannelNo* input error	Confirm the input value is correct.
DFB_DHCMEAS_ERR_INVALID_PULSEPERREV	*udiPulsePerRev* input error	Confirm the input value is correct.
DFB_DHCMEAS_ERR_INVALID_SAMPLINGTIME	*iSamplingTime* input error	Confirm the input value is correct.
DFB_DHCMEAS_ERR_INVALID_AVERAGETIMES	*iMovingAvgWindow* input error	Confirm the input value is correct.
DFB_DADLOG_ERR_INVALID_GROUP	*byRemoteID* input error	Confirm whether the input value of *byRemoteID* is correct.
DFB_DADLOG_ERR_NOT_SUPPORT_MODULE	The module corresponding to *byLocalID* is not an AD module	Confirm the module corresponding to the function block.
DFB_DADLOG_ERR_INVALID_MODULE	*byLocalID* input error	Confirm whether the input value is correct.
DFB_DADLOG_ERR_INVALID_POINTS	*iTotalPoints* typo	Confirm whether the input value is correct.
DFB_DADLOG_ERR_INVALID_CHNO	*usiChannelNo* input error	Confirm whether the input value is correct.
DFB_DADLOG_ERR_INVALID_MODE	*iMode* input error	Confirm whether the input value is correct.
DFB_DADLOG_ERR_INVALID_PERIOD	*iPeriod* input error	Confirm whether the input value is correct.
DFB_DADPEAK_ERR_INVALID_GROUP	*byRemoteID* input error	Confirm whether the input value of *byRemoteID* is correct.
DFB_DADPEAK_ERR_NOT_SUPPORT_MODULE	The module corresponding to *byLocalID* is not an AD module	Confirm the module corresponding to the function block.
DFB_DADPEAK_ERR_INVALID_MODULE	*byLocalID* input error	Confirm whether the input value is correct.
DFB_DADPEAK_ERR_INVALID_CHNO	*usiChannelNo* input error	Confirm whether the input value is correct.

Description	Cause of Error	Corrective Action
DFB_SCMRS_ERR_INVALID_COMPORT	The COM port No. exceeds range (1, 2, 11, 12, 21, 22, 31, 32).	Check whether the entered value is correct.
DFB_SCMRS_ERR_SCM_WITH_SPECIFIED _COMPORT_NOT_EXIST	The COM port No. is specified. The SCM module does not exist.	Check the target module of the FB.
DFB_SCMRS_ERR_INVALID_FUNCTION_CARD _TYPE_WITH_SPECIFIED_COMPORT	The COM port No. is specified. This SCM module function card is not AS–F232 / AS–F485 / AS–F422.	Check the target module of the FB.
DFB_SCMRS_ERR_SPECIFIED_COMPORT _ALREADY_EXECUTING_ANOTHER_SCMRS	The specified COM port No. has been used by other SCMRS instructions.	Check whether the entered value is correct.
DFB_SCMRS_ERR_WRITE_LENGTH_EXCEED_SETTING_RANGE_0_200	*uiWriteLen* is set to be less than 0 or greater than 200.	Check whether the entered value is correct.
DFB_SCMRS_ERR_RESERVED	–	–
DFB_SCMRS_ERR_INVALID_RXMODE	*RxMode* setting out of range (0–9)	Check whether the entered value is correct.
DFB_SCMRS_ERR_SPECEFIC_RXLENGTH_EXCEED_SETTING_RANGE_1_198	The receiving mode of *RxMode* is set to 6 or 8 or 9, and the length specified by uiSpecificRxLen is less than 1 or greater than 198.	Check whether the entered value is correct.
DFB_SCMRS_ERR_RESERVED2	–	–
DFB_SCMRS_ERR_COMMUNICATION _WITH_MODULE_TIMEOUT	Communication with the AS00SCM module timed out.	Check whether the module is functioning properly.
DFB_SCMRS_ERR_MODULE_REPORTS_AN_ERROR	AS00SCM module response error	Check whether the module is functioning properly.
DFB_SCMRS_ERR_RXMODE_CONDITION _NOT_FINISHED_TIMEOUT	Exceed the communication time–out period but still unable to reach the end condition.	Check whether the entered value is correct.
DFB_SCMRS_ERR_RECEIVE_DATA_LENGTH_ EXCEED_198_BYTE	The received data exceeds the maximum length of 198 characters.	Check whether the received data is correct.
DFB_SCMRS_ERR_PROTOCOL_NOT_SCMRS	The communication protocol is not COMRS.	Check the protocol settings.
DFB_IOLINKRW_ERR_IOLINK_DEVICE_APPLICATION_ERROR	IO-Link device application error	Check whether the module is functioning properly.

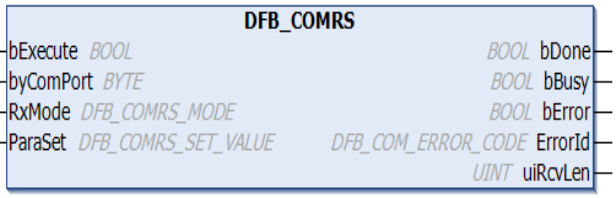
Description	Cause of Error	Corrective Action
DFB_IOLINKRW_ERR_INDEX_NOT_EXIST	Primary index does not exist.	Check whether the entered value is correct.
DFB_IOLINKRW_ERR_SUBINDEX_NOT_EXIST	Subindex does not exist.	Check whether the entered value is correct.
DFB_IOLINKRW_ERR_SERVICE_CURRENTLY_NOT_AVAILABLE	Service temporarily unavailable	-
DFB_IOLINKRW_ERR_SERVICE_CURRENTLY_NOT_AVAILABLE_LOCAL_CONTROL	Service temporarily unavailable-Local control	Check whether the module is functioning properly.
DFB_IOLINKRW_ERR_SERVICE_CURRENTLY_NOT_AVAILABLE_DEVICE_CONTROL	Service temporarily unavailable-Device control	Check whether the module is functioning properly.
DFB_IOLINKRW_ERR_ACCESS_DENIED	Access denied	-
DFB_IOLINKRW_ERR_PARAMETER_EXCEEDS_SETTING_RANGE	Parameter value exceeds valid range.	Check whether the entered value is correct.
DFB_IOLINKRW_ERR_PARAMETER_EXCEEDS_UPPER_LIMIT	Parameter value exceeds the upper limit.	Check whether the entered value is correct.
DFB_IOLINKRW_ERR_PARAMETER_EXCEEDS_LOWER_LIMIT	Parameter value is below the lower limit.	Check whether the entered value is correct.
DFB_IOLINKRW_ERR_PARAMETER_LENGTH_OVERRUN	Parameter length exceeds the upper limit.	Check whether the entered value is correct.
DFB_IOLINKRW_ERR_PARAMETER_LENGTH_UNDERRUN	Parameter length is below the lower limit.	Check whether the entered value is correct.
DFB_IOLINKRW_ERR_FUNCTION_NOT_AVAILABLE	Function not available	-
DFB_IOLINKRW_ERR_FUNCTION_CURRENTLY_NOT_AVAILABLE	Function temporarily unavailable	-
DFB_IOLINKRW_ERR_INVALID_PARAMETER_SET	Invalid parameter set	-
DFB_IOLINKRW_ERR_INCONSISTENT_PARAMETER_SET	Inconsistent parameter set	-
DFB_IOLINKRW_ERR_APPLICATION_NOT_READY_FOR_USE	Application not ready	-
DFB_IOLINKRW_ERR_IOLINK_DEVICE_NOT_IN_IOLINK_MODE	IO-Link device is not in the operation mode.	Check whether the module is functioning properly.
DFB_IOLINKRW_ERR_COMMUNICATION_PORT_NOT_CONNECTED_TO_ANY_IOLINK_DEVICE	Communication port is not connected to the IO-Link device.	Check whether the module is functioning properly.
DFB_IOLINKRW_ERR_CONNECTION_TO_IOLINK_DEVICE_IS_ESTABLISHING	IO-Link device connection initializing	Check whether the module is functioning properly.

Description	Cause of Error	Corrective Action
DFB_IOLINKRW_ERR_COMMUNICATION_PORT_EXCEEDS_SETTING_RANGE_1_4	Communication port number exceeds the valid range (1–4).	Check whether the entered value is correct.
DFB_IOLINKRW_ERR_SUBINDEX_EXCEEDS_SETTING_RANGE_0_255	Subindex number exceeds the valid range (0–255).	Check whether the entered value is correct.
DFB_IOLINKRW_ERR_READ_OR_WRITE_LENGTH_EXCEEDS_SETTING_RANGE_1_232	Read/Write data length exceeds the valid range (1–232 bytes).	Check whether the entered value is correct.
DFB_IOLINKRW_ERR_RESERVED	-	-
DFB_IOLINKRW_ERR_COMMUNICATION_TIMEOUT_NO_RESPONSE_FROM_IOLINK_DEVICE_5000MS	IO-Link device communication timeout (5000 ms)	Check whether the module is functioning properly.
DFB_IOLINKRW_ERR_INVALID_REMOTE_ID_OR_LOCAL_ID	AS04SIL group or module ID is incorrect.	Check whether the entered value is correct.
DFB_IOLINKRW_ERR_INTERNAL_COMMUNCATION_ERROR	Internal communication error	-
DFB_IOLINKRW_ERR_EXECUTE_MORE_THAN_2_IOLINKRW_ON_SAME_MODULE	Concurrent execution of multiple IOLINK Read/Write instructions on the same AS04SIL module	Check for duplicate instruction usage.
DFB_IOLINKRW_ERR_EXCUTE_MORE_THAN_2_IOLINKRW_ON_SAME_MODULE	Read/Write start address + data length exceeds device memory range.	Check whether the entered value is correct.
DFB_IOLINKRW_ERR_INVALID_DATA_TYPE	Parameter value of the corresponding data type exceeds the valid range.	Check whether the entered value is correct.
DFB_IOLINKRW_ERR_DATA_TYPE_AND_READ_WRITE_LENGTH_INCONSISTENT	Parameter data type mismatches with read/write data length.	Check whether the entered value is correct.

Chapter 7: Modbus Communication Instructions

7.1. DFB_COMRS

DFB_COMRS sends and receives communication data via the COM port.

FB/FC	Instruction	Graphic Expression	ST Language
FB	DFB_COMRS		DFB_COMRS (bExecute:= , byComPort:= , RxMode:= , ParaSet:= , bDone=> , bBusy=> , bError=> , ErrorId=> , uiRcvLen=>);

- **Inputs**

Name	Function	Data Type	Setting Value (Default value)
bExecute	Run the function block. (Triggered by the rising–edge)	BOOL	TRUE/FALSE (FALSE)
byComPort*	COM port number	BYTE	(0xFF)
RxMode	Receiving mode	DFB_COMRS_MODE	(NO_RECEIVING)
ParaSet	COM port parameters	DFB_COMRS_SET_VALUE	(COMRS_SET_VALUE)

Note: You need to configure the settings based on the definitions of COM port numbers varied from model to model.

- **DFB_COMRS_MODE**

Name	Description
NO_RECEIVING	No receiving data mode: After the packet is sent, the receiving task is complete. Then the completion flag is set to TRUE.
DISCONTINUOUS_TIME	Discontinuous time mode: When the interval between receiving packets is longer than the specified time, the receiving task is complete. Then the completion flag is set to TRUE. The discontinuous time for receiving the packet can be set via *ParaSet.uiDiscontinuousTime*.(*1)
SPECIFIC_END_CHAR	Specific end character mode: When a specific character is received, the data receiving is complete. Then the completion flag is set to TRUE. The end character and the length can be set via *ParaSet.pSpecificEndChar* and *ParaSet.byEndCharAmt*.

Name	Description
	(*1, *2)
SPECIFIC_START_CHAR_AND_DISCONTINUOUS_TIME	Specific start character and discontinuous time mode: When a specific character is received, starts to receive the packet, and when the packet interval is longer than the specified time, the receiving task is complete. The start character and the length can be configured via ParaSet.pSpecificStartChar and *ParaSet.byStartCharAmt* while the discontinuous time can be set via *ParaSet.uiDiscontinuousTime*. (*1, *2)
SPECIFIC_START_CHAR_AND_SPECIFIC_END_CHAR	Specific start character and end character mode: When a specific character is received, starts to receive the packet. The receiving task is not complete until receive an end character. The start character and the length can be set via *ParaSet.pSpecificStartChar* and *ParaSet.byStartCharAmt*. The end character and the length can be set via *ParaSet.pSpecificEndChar* and *ParaSet.byEndCharAmt*. (*1, *2)
SPECIFIC_LENGTH	Specific length mode: A specific quantity of data is received and the receiving task is complete. The packet length can be set via *ParaSet.uiSetVarue*.

***Note**:

1. When the received data length reaches the size defined in *uiReadBufSize*, the receiving of data is completed.
2. The data length includes both start and end characters.

- **COMRS_SET_VALUE**

Name	Function	Data Type	Setting Value (Default value)
uiWriteLen	The length of packet to be sent (Unit: Byte)	UINT	0–1000 (0)
pWriteBuf	The memory address of packet to be sent	POINTER TO BYTE	—
pReadBuf	The memory address of packet to be stored	POINTER TO BYTE	—
uiReadBufSize	The memory size of packet to be stored(Unit: Byte)	UINT	1–1,000 (100)
uiDiscontinuousTime	Packet interval (Unit: ms)	UINT	2–3,000 (2)
byStartCharAmt	Size of the start character	BYTE (Unit: Byte)	1–255 (1)
pSpecificStartChar	Memory address of the start character	POINTER TO BYTE	Memory address (0)
byEndCharAmt	Size of the end character	BYTE (Unit: Byte)	1–255 (1)

Name	Function	Data Type	Setting Value (Default value)
pSpecificEndChar	Memory address of the end character	POINTER TO BYTE	Memory address (0)
uiSpecificRxLen	Specified receiving length	UINT (Unit: Byte)	1–1000 (1)
tTimeout	Communication time–out	TIME	T#0ms–T#49d17h2m47s295ms(T#100ms) T#0ms: No time–out

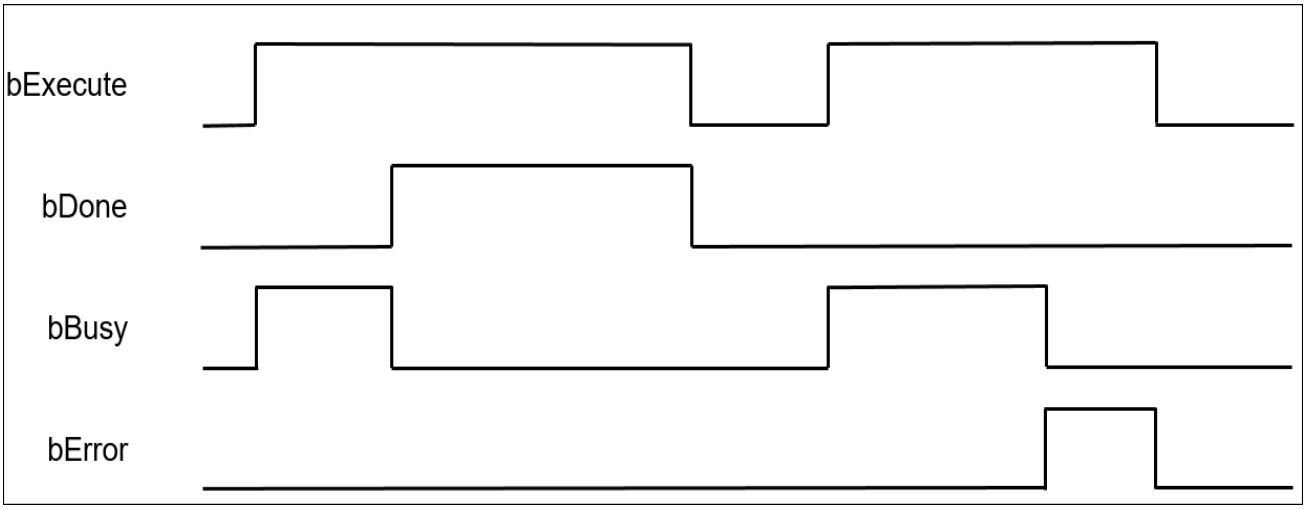
- **Outputs**

Name	Function	Data Type	Output Range (Default value)
bDone	FB is completed.	BOOL	TRUE/FALSE (FALSE)
bBusy	TRUE when the instruction is running	BOOL	TRUE/FALSE (FALSE)
bError	TRUE if an error occurs	BOOL	TRUE/FALSE (FALSE)
ErrorID	Indicates the error code if an error occurs.	DFB_COM_ERROR_CODE	(DFB_UNDEFINED)
uiRcvLen	The length of the data received	UINT (Unit: Byte)	(0)

- **Output Updating Timing**

Name	Timing for shifting to TRUE	Timing for shifting to FALSE
bDone	When FB is completed	When *bExecute* shifts to FALSE
bBusy	When FB starts	• When the running of FB is completed • *bExecute* shifts to FALSE and the running of FB is completed.
bError	When an error occurs during the running or there are incorrect input values	When *bExecute* shifts to FALSE
ErrorID		

- **Timing Diagram**



- **Function**

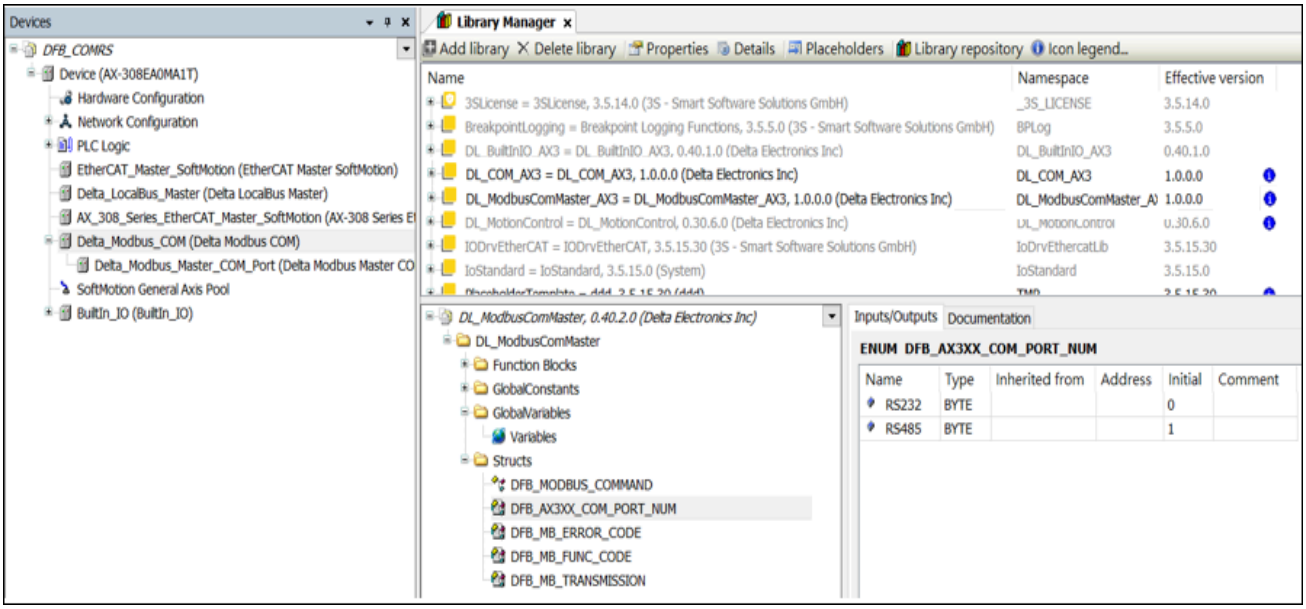
  The FB instruction (DFB_COMRS) is used for sending communication data. You must finish the configuration of COM port of CPU and add Delta_Modbus_Master_COM_Port device before using this instruction (For more details, refer to chapter 9.2 Serial Port Communication in *AX-3 Series Operation Manual*).

- **Programming Example**

  This example used DFB_COMRS to send COM communication data.

```
PROGRAM PLC_PRG
VAR
    bVar0: BOOL:=TRUE;
    bExecute_Var, bDone_Var,bBusy_Var,bError_Var: BOOL;
    FB0: DFB_COMRS;
    RxMode_Var: DL_COM_AX3.DFB_COMRS_MODE;
    ParaSet_Var: DL_COM_AX3.DFB_COMRS_SET_VALUE;
    ErrorID_Var: DL_COM_AX3.DFB_COM_ERROR_CODE;
    uiRcvLen_Var: UINT;
    pSpecificStartChar_Var: BYTE :=16#3A;
    byStartCharAmt_Var: BYTE :=1;
    pSpecificEndChar_Var: ARRAY[0..1] OF BYTE:=[16#0D,16*0A];
    byEndCharAmt_Var: BYTE :=2;
    ar_byVar0: ARRAY [0..13] OF BYTE;
    ar_byVarl: ARRAY [0..16] OF BYTE;=[16#3A,16#30,16#32,16#30,16#31,16#30,
    16#35,16#30,16#30,16#30,16#30,16#30,16#31,16#46,16#37,16#0D,16#0A];
    tTimeout_Var: TIME :=T*500ms;
END_VAR

IF bVar0 THEN
    bVarO:=FALSE;
    bExecute_Var:=TRUE;
    RxMode_Var:=DL_COM_AX3.DFB_COMRS_MODE.SPECIFIC_START_CHAB_AND_SPECIFIC_END_CHAR;
 ParaSet_Var.pSpecificStartChar:=ADR(pSpecificStartChar_Var);
    ParaSet_Var.byStartCharAmt:=byStartCharArat_Var;
    ParaSet_Var.pSpecificEndChar:=ADR(pSpecificEndChar_Var);
    ParaSet_Var.byEndCharAmt:=byEndCharAmt_Var;
    ParaSet_Var.pReadBuf:=ADR(ar_byVar0);
    ParaSet_Var.pWriteBuf:=ADR(ar_byVarl);
    ParaSet_Var.uiWriteLen:=SIZEOF(ar_byVarl);
    ParaSet_Var.tTimeout:=tTimeout Var;
END_IF
IF bDone_Var THEN
bExecute_Var:=FALSE;
END_IF
FB0(
bExecute:=bExecute_Var ,
byComPort:=DL_ModbusComMaster.DFB_AX3XX_COM_PORT_NUM.RS485 ,
RxMode:=RxMode_Var ,
ParaSet:=ParaSet_Var ,
bDone=>bDone Var ,
bBusy=>bBu3y_Var ,
bError=>bError_Var ,
ErrorId=>ErrorID_Var ,
uiRcvLen=>uiRcvLen Var );
```

For AX-3 series controller, the definition of COM port name can be found in Library Manager as shown below.



- **Supported Devices**
  - AX series

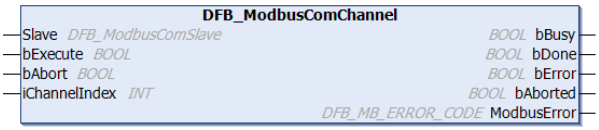
    Note 1: In the AX-8 series, only AX-864E30xE2T is supported.

    Note 2: AX-332E (Library firmware version: 1.0.6.0 and later).

- **Library**
  - DL_COM.library
    Note: From version 1.0.6.0, library DL_COM_AX3 is changed to DL_COM.

7.2. DFB_ModbusComChannel

DFB_ModbusComChannel controls the Modbus Slave COM Port Channel.

FB/FC	Instruction	Graphic Expression	ST Language
FB	DFB_ ModbusComChannel		DFB_ModbusComChannel(Slave:= , bExecute:= , bAbort:= , iChannelIndex:= , bBusy=> , bDone=> , bError=> , bAborted=> , ModbusError=>);

- **Inputs/Outputs**

Name	Function	Data Type	Setting Value (Default value)
Slave	Delta Modbus slave device	DFB_ModbusComSlave	—

- **Inputs**

Name	Function	Data Type	Setting Value (Default value)
bExecute	Run the function block. (Triggered by the rising-dege)	BOOL	TRUE/FALSE (FALSE)
bAbort	No function	BOOL	—
iChannelIndex	Channel index	INT	0–9 (0)

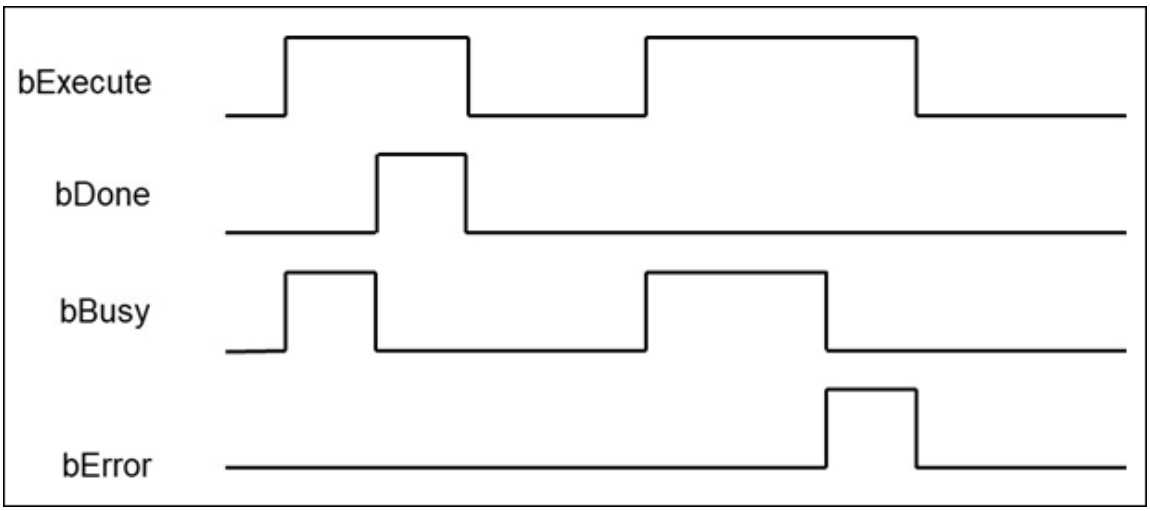
- **Outputs**

Name	Function	Data Type	Output Range (Default value)
bBusy	TRUE when the instruction is running	BOOL	TRUE/FALSE (FALSE)
bDone	The running of FB is complete.	BOOL	TRUE/FALSE (FALSE)
bError	TRUE if an error occurs	BOOL	TRUE/FALSE (FALSE)
bAborted	No function	BOOL	—
ModbusError	Error code	DFB_MB_ERROR_CODE	DL_MB_ERROR_CODE (UNDEFINED)

- **Output Updating Timing**

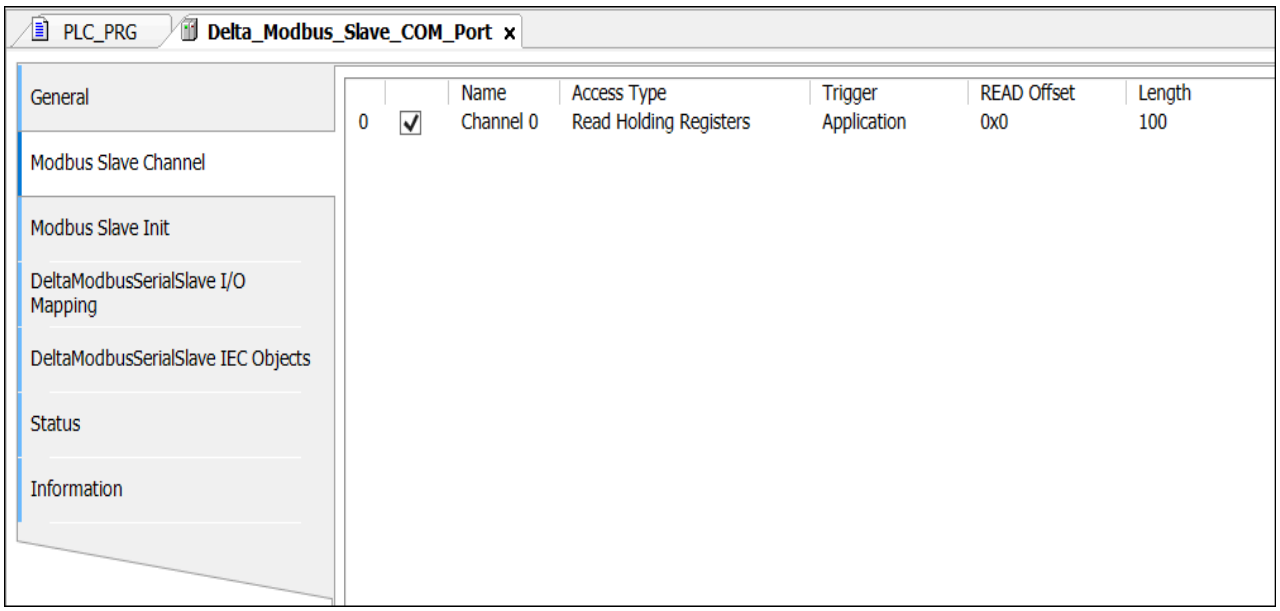
Name	Timing for shifting to TRUE	Timing for shifting to FALSE
bBusy	When the running of FB starts	• When the running of FB is complete • *bExecute* shifts to FALSE and the running of FB is completed.
bError	When an error occurs during the running or there are incorrect input values	When *bExecute* shifts to FALSE
ModbusError		
bDone	When the running of FB is complete	When *bExecute* shifts to FALSE

- **Timing Diagram**



- **Function**

When the trigger mode of the Modbus slave channel is set to Application, the Modbus request action will be triggered by DFB_ModbusComChannel.

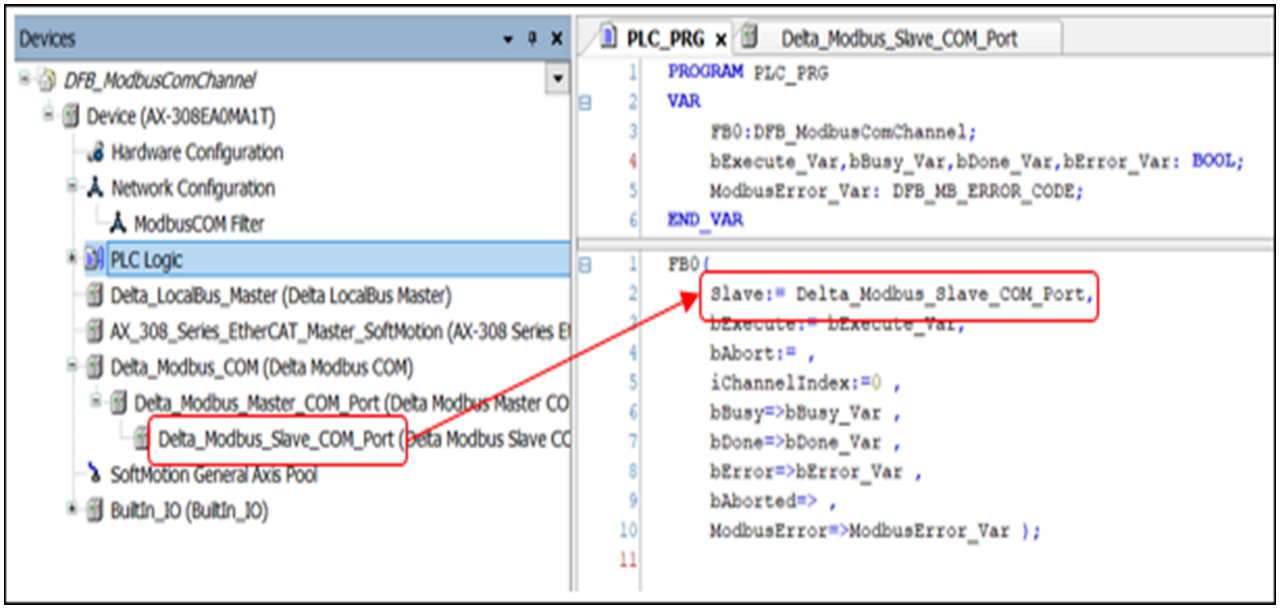


Note:

1. For more details of Modbus slave COM port configuration, you can refer to chapter 9.2 Serial Communication in *AX-3 Series Operation Manual*.
2. While using, the channel must be set to **Enable**.

- **Programming Example**

  This example uses DFB_ModbusComChannel to trigger data exchange with Modbus COM port communication.



  ***Note**: The input of Slave will be the name of Modbus slave device.

- **Supported Devices**

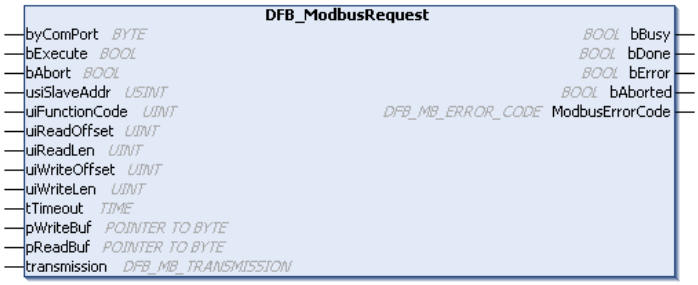
  - AX-3, AX-864E30xE2T, AX-C
    Note: AX-332E, AX-864E30xE2T, AX-C (Library firmware version: 1.0.6.0 and later)

- **Library**

  - DL_ModbusComMaster.library
    Note: From version 1.0.6.0, library DL_ModbusComMaster_AX3 is changed to DL_ModbusComMaster.

7.3. DFB_ModbusRequest

DFB_ModbusRequest sends Modbus communication data.

FB/FC	Instruction	Graphic Expression	ST Language
FB	DFB_ ModbusRequest		DFB_ModbusReque st(byComPort:= , bExecute:= , bAbort:= , usiSlaveAddr:= , uiFunctionCode:= , uiReadOffset:= , uiReadLen:= , uiWriteOffset:= , uiWriteLen:= , tTimeout:= , pWriteBuf:= , pReadBuf:= , transmission:= , bBusy=> , bDone=> , bError=> , bAborted=> , ModbusErrorCode= >);

- **Inputs**

Name	Function	Data Type	Setting Value (Default value)
byComPort*[1]	COM port number	BYTE	(0xFF)
bExecute	Run the function block. (Triggered by the rising-dege)	BOOL	TRUE/FALSE (FALSE)
bAbort	No function	BOOL	—
usiSlaveAddr	Slave station number	USINT	1–247
uiFunctionCode	Modbus function code	DFB_MB_FUNC_CODE	Supported function codes: 0x01: Read Coils 0x02: Read Discrete Inputs 0x03: Read Holding Registers 0x04: Read Input Registers 0x05: Write Single Coil 0x06: Write Single Register

Name	Function	Data Type	Setting Value (Default value)
			0x0F: Write Multiple Coils 0x10: Write Multiple Registers 0x17: Read/Write Multiple Registers (0x03)
uiReadOffset	The start address of memory to be read.	UINT	0–65535 (0)
uiReadLen	The data length of the memory to be read.	UINT	Coil: 1–1920 Register: 1–120 (1)
uiWriteOffset	The start address of memory to be written.	UINT	0–65535 (0)
uiWriteLen*5	The data length of the memory to be written	UINT	Coil: 1–1920 Register: 1–120 (1)
tTimeout*2	Communication time–out	TIME	T#0ms–T#49d17h2m47s295ms 0: No time–out (T#100ms)
pWriteBuf	The memory address of data to be sent.	POINTER TO BYTE	—
pReadBuf	The memory address of data to be stored.	POINTER TO BYTE	—
Transmission*3	Transmission mode	DFB_MB_TRANSMISSION	0: ASCII 1: RTU (ASCII)

***Note**:

1. You need to configure the settings based on the definitions of COM port numbers varied from model to model.
2. The time–out should be greater than the Cycle time set in mdbus Task.
3. When the transmission mode is set to RTU, the data bit of Modbus COM port must be set to 8.
4. If *uiReadLen* is set to 0, the Modbus read command will not be sent.
5. If *uiWriteLen* is set to 0, the Modbus write command will not be sent.

- **Outputs**

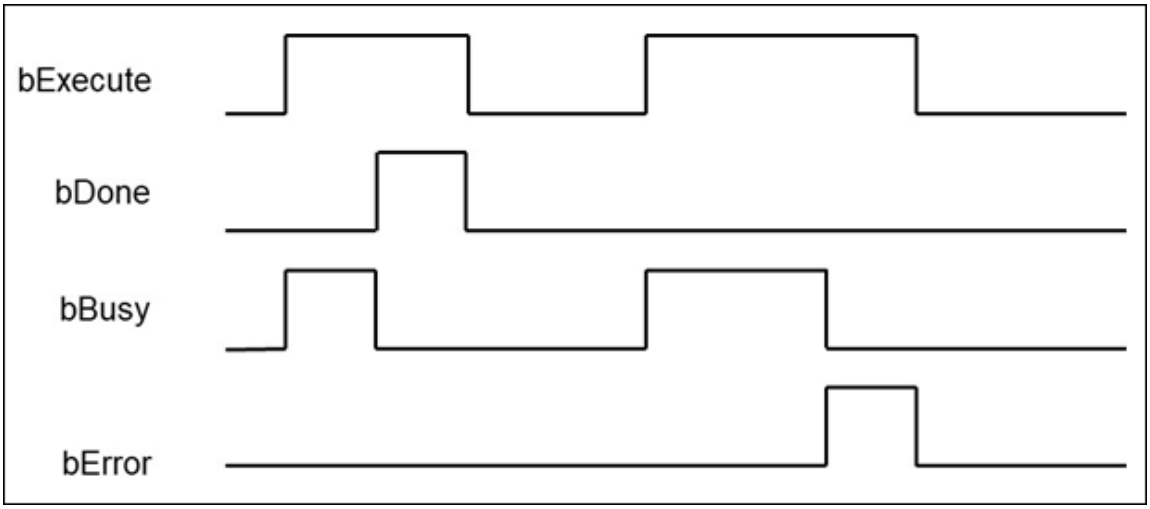
Name	Function	Data Type	Output Range (Default value)
bBusy	TRUE when the instruction is running	BOOL	TRUE/FALSE (FALSE)
bDone	The running of FB is completed.	BOOL	TRUE/FALSE (FALSE)
bError	TRUE if an error occurs	BOOL	TRUE/FALSE (FALSE)
bAborted	No function	BOOL	—
ModbusError	Error code	DFB_MB_ERROR_CODE	DL_MB_ERROR_CODE

Name	Function	Data Type	Output Range (Default value)
			(DFB_UNDEFINED)

- **Output Updating Timing**

Name	Timing for shifting to TRUE	Timing for shifting to FALSE
bDone	When the running of FB is complete	When *bExecute* shifts to FALSE
bBusy	When the running of FB starts	• When the running of FB is complete • *bExecute* shifts to FALSE and the running of FB is completed.
bError	When an error occurs during the running or there are incorrect input values	When *bExecute* shifts to FALSE
ModbusError		

- **Timing Diagram**



- **Function**

  The FB instruction (DFB_ModbusRequest) is used for sending Modbus communication data. You must finish the configuration of COM port of CPU and add Delta_Modbus_Master_COM_Port device before using this instruction. (For more details, refer to chapter 9.2 Serial Port Communication in *AX-3 Series Operation Manual*.)

- **Programming Example**

  This example uses DFB_ModbusRequest to send Modbus commands for reading a 10–word long data (Holding Registers) in the slave station (Slave address = 2), which the start address is 0x0000.

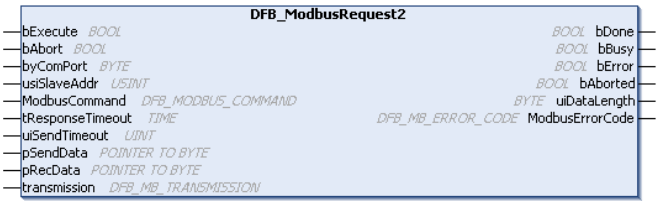
```
PROGRAM PLC_PRG
VAR
    bExecute_Var,bBusy_Var, bDone_Var,bError_Var: BOOL;
    U3iSlaveAddr_Var: USINT :=2 ;
    ar_wVar0: ARRAY[0..200]OF WORD;
    FBO: DFB_ModbusRequest;
    ModbusErrorCode_Var: DFB_MB_ERROR_CODE;
END_VAR

FB0(
    byComPort:=DL_ModbusComMaster_AX3.DFB_AX3_COM_PORT_NUM.RS485,
    bExecute:=bExecute_Var ,
    bAbort:= ,
    usiSlaveAddr:=usiS1aveAddr_Var ,
```

```
    uiPunctionCode:=DFB_MB_FUNC_CODE.READ_HOLDING_REGISTERS ,
    uiReadOffset:=0 ,
    uiReadLen:=100 ,
    uiWriteOffset := ,
    uiWriteLen:= ,
    tTimeout :=T#500MS ,
    pWriteBuf:= ,
    pReadBuf:=ADR(ar_wVar0) ,
    transmission:= ,
    bBusy=>bBusy_Var ,
    bDone=>bDone_Var ,
    bError=>bError_Var ,
    bAborted=> ,
    ModbusErrorCode=>ModbusErrorCode Var );
```

- **Supported Devices**
  - AX-3, AX-864E30xE2T, AX-C
    Note: AX-332E, AX-864E30xE2T, AX-C (Library firmware version: 1.0.6.0 or later)
- **Library**
  - DL_ModbusComMaster.library
    Note: From version 1.0.6.0, library DL_ModbusComMaster_AX3 is changed to DL_ModbusComMaster.

7.4. DFB_ModbusRequest2

DFB_ModbusRequest2 sends Modbus communication data.

FB/FC	Instruction	Graphic Expression	ST Language
FB	DFB_ ModbusRequest2		DFB_ModbusRequest2(bExecute:= , bAbort:= , byComPort:= , usiSlaveAddr:= , ModbusCommand:= , tResponseTimeout:= , uiSendTimeout:= , pSendData:= , pRecvData:= , transmission:= , bDone=> , bBusy=> , bError=> , bAborted=> , uiDataLength=> , ModbusErrorCode=>);

- **Inputs**

Name	Function	Data Type	Setting Value (Default value)
bExecute	Run the function block. (Triggered by the rising-dege)	BOOL	TRUE/FALSE (FALSE)
bAbort	No function	BOOL	—
byComPort*1	COM port number	BYTE	(0xFF)
usiSlaveAddr	Slave station number	USINT	1–247
ModbusCommand	Modbus parameter setting	DFB_MODBUS_COMMAND	—
tResponseTimeout*2	Communication time–out	TIME	T#0ms–T#49d17h2m47s295ms 0: No time–out (T#100ms)
uiSendTimeout	No function	UINT	(0)
pSendData	The memory address of data to be sent	POINTER TO BYTE	—
pRecvData	The memory address of received data to be stored	POINTER TO BYTE	—

Name	Function	Data Type	Setting Value (Default value)
Transmission*3	Transmission mode	DFB_MB_TRANSMISSION	0: ASCII 1: RTU (ASCII)

***Note**:

1. You need to configure the settings based on the definitions of COM port numbers varied from model to model.
2. The time–out should be greater than the Cycle time set in mdbus Task.
3. When the transmission mode is set to RTU, the data bit of Modbus COM port must be set to 8.

- DFB_MODBUS_COMMAND

Name	Function	Data Type	Output Range (Default value)
uiFunctionCode	Modbus function code	DFB_MB_FUNC_CODE	Supported function code: 0x01: Read Coils 0x02: Read Discrete Inputs 0x03: Read Holding Registers 0x04: Read Input Registers 0x05: Write Single Coil 0x06: Write Single Register 0x0F: Write Multiple Coils 0x10: Write Multiple Registers 0x17: Read/Write Multiple Registers (0x03)
uiReadOffset	The start address of memory to be read.	UINT	0–65535 (0)
uiReadLen	The data length of the memory to be read.	UINT	Coil: 1–1920 Register: 1–120 (1)
uiWriteOffset	The start address of memory to be written.	UINT	0–65535 (0)
uiWriteLen	The data length of the memory to be written	UINT	Coil: 1–1920 Register: 1–120 (1)

***Note:**

1. If *uiReadLen* is set to 0, the Modbus read command will not be sent.
2. If *uiWriteLen* is set to 0, the Modbus write command will not be sent.

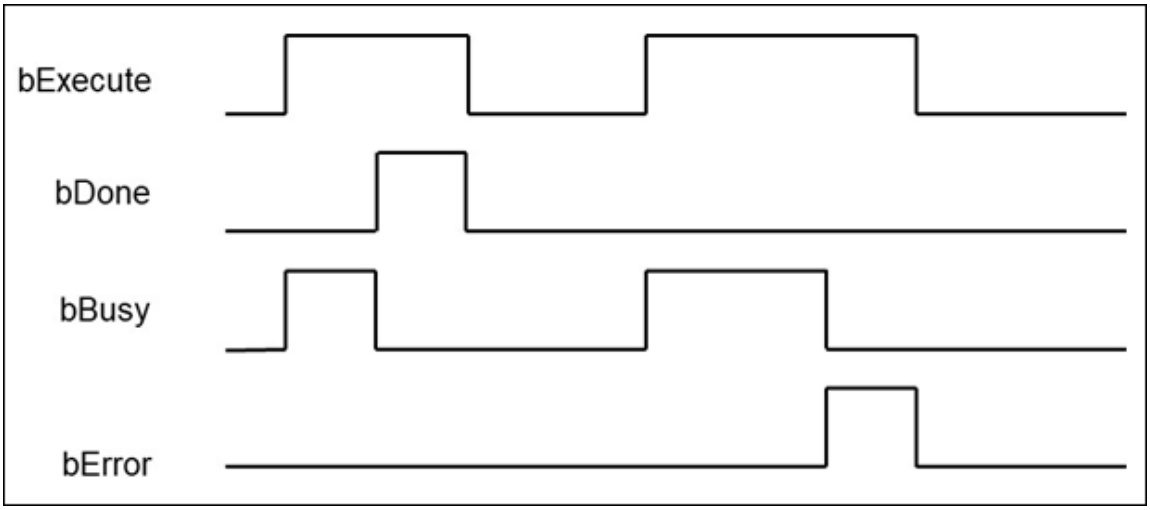
- **Outputs**

Name	Function	Data Type	Output Range (Default value)
bBusy	TRUE when the instruction is running	BOOL	TRUE/FALSE (FALSE)
bDone	The running of FB is complete.	BOOL	TRUE/FALSE (FALSE)
bError	TRUE if an error occurs.	BOOL	TRUE/FALSE (FALSE)
bAborted	No function	BOOL	—
uiDataLength	The received data length	BYTE (Unit: BYTE)	(0)
ModbusError	Error code	DFB_MB_ERROR_CODE	DL_MB_ERROR_CODE (DFB_UNDEFINED)

- **Output Updating Timing**

Name	Timing for shifting to TRUE	Timing for shifting to FALSE
bDone	When the running of FB is complete	When *bExecute* shifts to FALSE
bBusy	When the running of FB starts	• When the running of FB is completed • *bExecute* shifts to FALSE.
bError	When an error occurs during the running or there are incorrect input values	When *bExecute* shifts to FALSE
ModbusError		

- **Timing Diagram**



- **Function**

  The FB instruction (DFB_ModbusRequest2) is used for sending Modbus communication data. You must finish the configuration of COM port of CPU and add Delta_Modbus_Master_COM_Port device before using this instruction. (For more details, Refer to chapter 9.2 “Serial Port Communication” in *AX-3 Series Operation Manual*.)

- **Programming Example**

  This example uses DFB_ModbusRequest2 to send Modbus commands for reading a 100–WORD long data (Holding Registers) in the slave station (Slave address = 2), which the start address is 0x0000.

```
PROGRAM PLC_PRG
VAR
```

```
    bVar: BOOL:=TRUE;
    bExecute_Var,bBusy_Var,bDone_Var,bError_Var:Bool;
    U3iSlaveAddr_Var: USINT :=2 ;
    ar_wVar0: ARRAY[0..200]OF WORD;
    FB0: DFB_ModbusRequest2;
    ModbusErrorCode_Var: DFB_MB_ERROR_CODE;
    ModbusCommand_Var: DFB_MODBUS_COMMAND;
    uiDataLength_Var: UINT;
END_VAR

IF bVar0 THEN
    bVar0:=FALSE;
    ModbusCommand_Var.uiFunctionCode:=3;
    ModbusCommand_Var.uiReadLen:=100;
    ModbusCommand_Var.uiReadOffset:=0;
END_IF

FB0(
    bExecute:=bExecute_Var ,
    bAbort:= ,
    byComPort:=DL_ModbusComMaster.DFB_AX3XX_COM_PORT_NUM.RS485,
    usiSlaveAddr:=usiS1aveAddr_Var ,
    ModbusCommand:=ModbusCommand_Var ,
    tResponseTimeout:=T#500MS ,
    uiSendTimeout:= ,
    pSendData:= ,
    pRecvData:=ADR(ar_wVar0) ,
    transmission:=DFB_MB_TRANSMISSION.ASCII ,
    bDone=>bDone_Var ,
    bBusy=>bBusy_Var ,
    bError=>bError_Var ,
    bAborted=> ,
    uiDataLength=>uiDataLength_Var ,
    ModbusErrorCode=>ModbusErrorCode Var );
```

- **Supported Devices**
  - AX-3, AX-864E30xE2T, AX-C
    Note: AX-332E, AX-864E30xE2T, AX-C (Library firmware version: 1.0.6.0 or later)
- **Library**
  - DL_ModbusComMaster.library

    Note: From version 1.0.6.0, library DL_ModbusComMaster_AX3 is changed to DL_ModbusComMaster

7.5. Error codes and Troubleshooting

- DFB_COM_ERROR_CODE

Description	Cause of Error	Corrective Action
DFB_NO_ERROR	No errors	—
DFB_RESPONSE_TIMEOUT	Slave response time–out	• Check whether the setting for time–out is appropriate or not. • Check on the correctness of communication wiring.
DFB_REQUEST_FAILED_TO_SEND	COM Port errors	Contact us directly.
DFB_INVALID_COMPORT	COM port setting errors	Check on the correctness of the COM port settings.
DFB_INVALID_BUFFER	Invalid memory address for sending and receiving data.	Check if the below parameter settings are correct. • *ParaSet.pWriteBuf* • *ParaSet.pReadBuf*
DFB_INVALID_LENGTH	Invalid data length setting	Check if the below parameter settings are correct. • *ParaSet.uiReadLen* • *ParaSet.uiReadBufSize* • *ParaSet.uiWriteLen*
DFB_NO_MASTER_CONFIG	Delta_Modbus_Master_COM_Port device does not exist.	Check if Delta_Modbus_Master_COM_Port device have been added to the device tree.
DFB_MEMORY_NOT_ENOUGH	Not enough system memory	Check if the program size exceeds the allowable limit.
DFB_INVALID_MODE	Invalid receiving mode set in DFB_COMRS.	Check if the RxMode setting is correct.
DFB_INVALID_SETTING	Invalid parameter setting	Check if the below parameter settings are correct. • *ParaSet.tTimeout* • *ParaSet.uiDiscontinuousTime* • *ParaSet.byEndCharAmt*

Description	Cause of Error	Corrective Action
		• *ParaSet.byStartCharAmt* • *ParaSet.uiSpecificRxLen*
DFB_INVALID_CHAR_BUFFER	Invalid memory address of characters.	Check if the below parameter settings are correct. • *ParaSet.pSpecificStartChar* • *ParaSet.pSpecificEndChar*
DFB_UNDEFINED	Undefined or has not yet been Run.	Wait for the running of FB instruction being completed.

- DFB_MB_ERROR_CODE

Description	Cause of Error	Corrective Action
DFB_NO_ERR	No errors	—
DFB_ILLEGAL_FUNCTION	Unsupported function code	Check on the correctness of the function code you're using.
DFB_ILLEGAL_DATA_ADDRESS	Illegal memory address to write and read.	Check on the correctness of memory address you intend to write and read.
DFB_ILLEGAL_DATA_VALUE	Illegal data values responded by slave.	Check if the slave wires function normally as well as the proper wiring.
DFB_SLAVE_DEVICE_FAILURE	Slave failure	Check slave settings and statuses.
DFB_ACKNOWLEDGE	Slave has received request, but it takes longer to handle.	—
DFB_SLAVE_DEVICE_BUSY	Slave is busy.	—
DFB_GATEWAY_PATH_UNAVAILABLE	Wrong Gateway path	Check Gateway configuration, or Gateway is *busy*.
DFB_GATEWAY_DEVICE_FAILED_TO_RESPOND	Slave device in Gateway fails to respond.	Check if the slave wires function normally as well as the proper wiring.
DFB_RESPONSE_TIMEOUT	No response from slave in time	• Check if the duration set for the time–out is less than the responded time of slave. • Check on the correctness of the wiring.

Description	Cause of Error	Corrective Action
DFB_RESPONSE_CRC_ERROR	Invalid data values responded by slave (Invalid check code)	Check on the correctness of data format responded by slave.
DFB_RESPONSE_WRONG_SLAVE	Invalid data values responded by slave (Invalid station number)	Check on the correctness of data format responded by slave.
DFB_RESPONSE_WRONG_ FUNCTIONCODE	Invalid data values responded by slave (Invalid function code)	Check on the correctness of data format responded by slave.
DFB_REQUEST_FAILED_TO_SEND	Failed to send data.	Contact the vendor directly.
DFB_RESPONSE_INVALID_PROTOCOL	Invalid data values responded by slave (Non–standard Modbus format)	Check on the correctness of data format responded by slave.
DFB_RESPONSE_INVALID_HEAD	Invalid data values responded by slave (Invalid data length)	Check on the correctness of data format responded by slave.
DFB_INVALID_CHANNEL_INDEX	Invalid index of the slave channel	Check if the index of slave channel is correct.
DFB_CHANNEL_SETTING_NOT_SUPPORT	The trigger mode of slave channel is not set to “Application”.	Make sure the trigger mode has been set to “Application”.
DFB_INVALID_COMPORT	Invalid COM port number of the controller	Check if the COM port number is correct.
DFB_INVALID_BUFFER	Invalid memory address setting to send and receive data	Check if the below parameter settings are correct. ModbusRequest: • *pWriteBuf* • *pReadBuf* ModbusRequest2: • *ModbusCommand.pWriteBuf* *ModbusCommand.pReadBuf*
DFB_INVALID_LENGTH	Invalid data length setting	Check if the below parameter settings are correct ModbusRequest: • *uiWriteLen* • *uiReadLen* ModbusRequest2: • *ModbusCommand.uiWriteLen*

Description	Cause of Error	Corrective Action
		ModbusCommand.uiReadLen
DFB_INVALID_SLAVE_ADDRESS	Invalid slave station number	Make sure the station number is set to be within 1–247.
DFB_INVALID_FUNCTION_CODE	Invalid setting for *uiFunctionCode*	Check if the setting value of *uiFunctionCode* is correct.
DFB_NO_MASTER_CONFIG	Delta_Modbus_Master_COM_Port device does not exist.	Make sure that Delta_Modbus_Master_COM_Port device has been added to the device tree.
DFB_MB_ERROR_CODE_MEMORY_NOT_ENOUGH	Not enough system memory	Check if the program size exceeds the allowable limit.
DFB_UNDEFINED	Undefined or has not yet run.	Wait for the running of FB instruction being completed.

Chapter 8: Network Communication Instructions

8.1. DFB_TCP_Client

DFB_TCP_Client sends or receives TCP data packets.

FB/FC	Instruction	Graphic Expression	ST Language
FB	DFB_ TCP_Client		DFB_TCP_Client(bEnable:= , SocketInfo:= , bSend:= , bRecvRestart:= , bBusy=> , bConnected=> , bSent=> , bRcvd=> , bError=> , ErrorID=> , Status=> , uiRcvdLen=>);

- **Inputs**

Name	Function	Data Type	Setting Value (Default value)
bEnable	Run the function block. *1*2	BOOL	TRUE/FALSE (FALSE)
SocketInfo	Connection information on Server	tcpClientSocketInfo	—
bSend	Send data packets. (Triggered by the rising-dege)	BOOL	TRUE/FALSE (FALSE)
bRecvRestart	Restart to receive data packets. *2 (Triggered by the rising-dege)	BOOL	TRUE/FALSE (FALSE)

***Note**:

1. As soon as this function block is running, TCP connection will start to be created. Once connected, the output *bConnected* will be ON.
2. After the function block is running, it starts receiving data packets. When the data receiving is completed and stopped, *bRcvd* will be ON. If you shift *bRecvRestart* to ON, the FB will restart to receive data.

    - **tcpClientsocketInfo**

Name	Function	Data Type	Setting Value (Default value)
byIPAddr	Server's IP address	ARRAY [0..3] OF BYTE	—
uiLPort	Communication ports on local device	UINT	0: Use a random port number 0–65535 (0)

Name	Function	Data Type	Setting Value (Default value)
uiRPort	Communication ports on remote device	UINT	0: Illegal 1–65535 (0)
uiTimeout	Response time–out (Unit: ms)	UINT	0: No time–out 1–65535 (0)
uiKeepAliveTimeout	The time that the socket keeps alive. (Unit: sec)	UINT	0: No time–out 1–65535 (0)
bReconnect	Auto–reconnect function	BOOL	TRUE/FALSE (FALSE) TRUE: When connection time–out or failed, it will try to rebuilt the connection automatically. FALSE: When connection time–out or failed, the output *bError* will be ON.
uiSetValue	The setting value of recvCondition	UINT	(0)
pSendBuf	The memory address of data to be sent	POINT TO BYTE	—
uiSendLen	The length of data to be sent (Unit: Byte)	UINT	0–8192 (0)
pRecvBuf	The memory address where the received data to be stored.	POINT TO BYTE	—
uiRecvBufSize	The memory size of received data (Unit: Byte)	UINT	0–8192 (0)
recvCondition	Conditions for data receiving completion	DFB_SOCK_RECV_MODE	(DFB_SOCKET_NO_RECEVING)

***Note**: *uiTimeout* is the timeout period for communication response, which starts to calculate the timeout period after receiving the first packet. If no packets are received, the timeout period will not be calculated. Otherwise, *uiKeepAlive*Timeout starts to calculate the timeout time when no packets are received.

- **DFB_SOCK_RECV_MODE**

Name	Description
DFB_SOCK_MODE_NO_RECEVING	No receiving data mode

Name	Description
DFB_SOCK_MODE_SPECIFIC_LENGTH	Specific data length mode: A specific quantity of data is received and the receiving task is completed. The data length can be specified via *uiSetValue*. (Unit: Byte)
DFB_SOCK_MODE_SPECIFIC_SINGLE_CHAR	Specific end character mode: The data received ends with a specific character (1 Byte). The end character can be configured via *uiSetValue*. e.g.: If *uiSetValue* is set to 16#00000D0A, the end character will be 16#0A.(*1*2)
DFB_SOCK_RECV_MODE_DFB_SOCK_MODE_SPECIFIC_TWO_CHARS	Specific two end characters mode: The data received ends with the two specific characters (2 Bytes) The end character can be configured via *uiSetValue* e.g.: If *uiSetValue* is set to 16#00000D0A, the end characters will be 16#0D0A. (*1*2)
DFB_SOCK_RECV_MODE_DFB_SOCK_MODE_SPECIFIC_START_CHAR_AND_SPECIFIC_END_CHAR	Specific start character and end character mode: The data received starts with a specific character, and ends with a specific character. Both the start and the end character can be configured via *uiSetValue*. e.g.: If *uiSetValue* is set to 16#00003A0A, the start character will be 16#3A and the end character is 16#0A. (*1*2)
DFB_SOCK_RECV_MODE_DFB_SOCK_MODE_ANY_LENGTH	Any length mode: The receiving ends with a complete data of any length. (*1)

***Note**:

1. When the received data length reaches the pRecvLen limit, the receiving task will be completed.
2. The data length includes the start and end characters.
3. TCP sockets can receive data in a single TCP packet in the *Any length* mode (generally about 1500 bytes, depending on the specifications of each device).

- **Outputs**

Name	Function	Data Type	Output Range (Default value)
bBusy	TRUE when the instruction is running.	BOOL	TRUE/FALSE (FALSE)
bConnected	TCP is connected.	BOOL	TRUE/FALSE (FALSE)
bSent	Sending complete	BOOL	TRUE/FALSE (FALSE)
bRcvd	Receiving complete	BOOL	TRUE/FALSE (FALSE)
bError	TRUE if an error occurs	BOOL	TRUE/FALSE (FALSE)

Name	Function	Data Type	Output Range (Default value)
ErrorID	Indicates the error code if an error occurs.	DFB_SOCKET_ERROR	(DFB_SOCK_NO_ERROR)
Status	The running status of socket	DFB_SOCKET_STATUS	(SOCKET_CLOSED)
uiRcvLen	The length of received data	UINT (Unit: Byte)	(0)

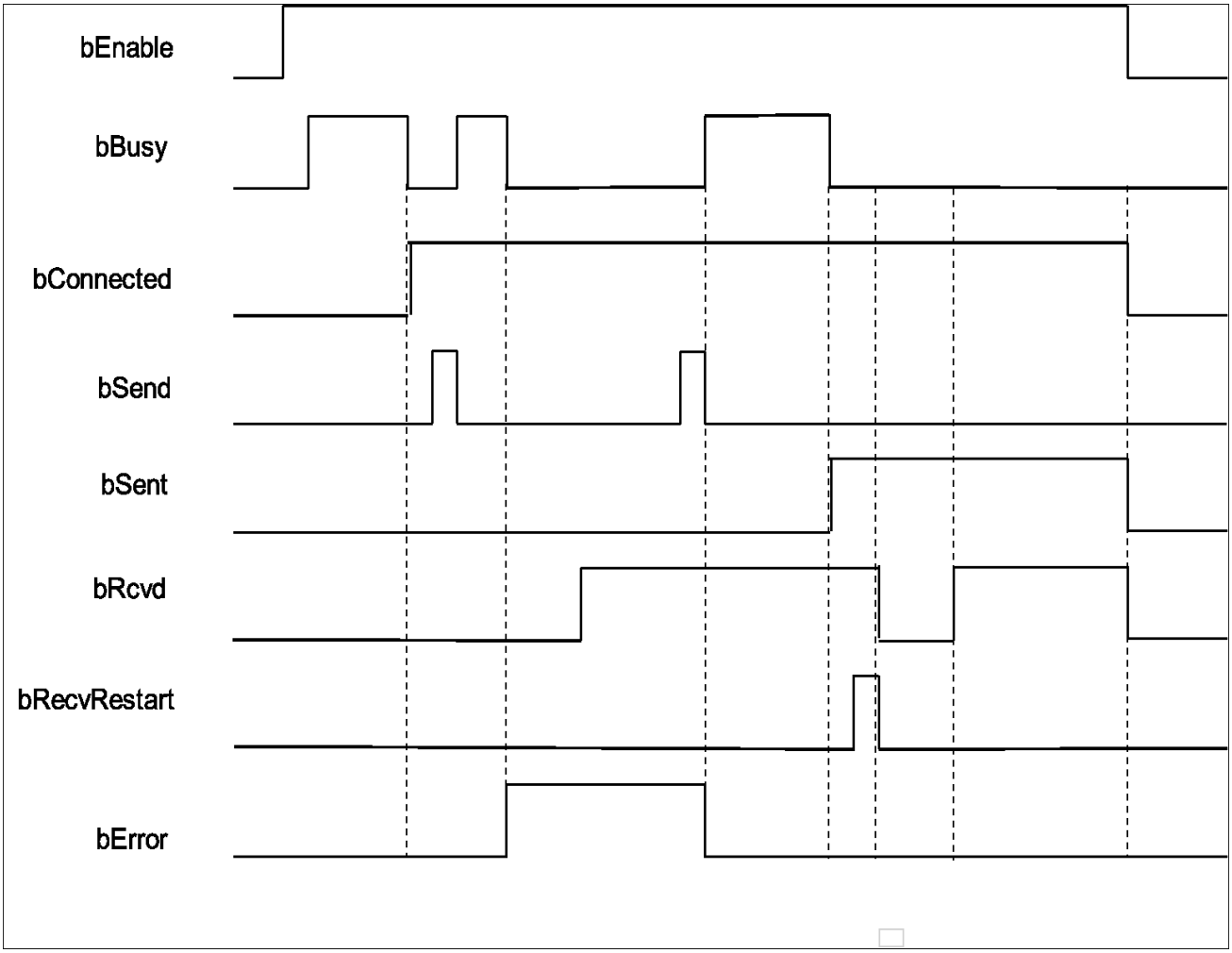
- **DFB_SOCKET_STATUS**

Name	Description	Applicable Protocol
SOCKET_CLOSED	SOCKET connection is closed.	TCP / UDP
SOCKET_CONNECTING	SOCKET is connecting.	TCP
SOCKET_CONNECTED	SOCKET is connected.	TCP
SOCKET_SENDING	SOCKET is sending the data packet.	TCP / UDP
SOCKET_SENT	SOCKET has sent the data packet.	TCP / UDP
SOCKET_RECEIVED	SOCKET has received the data packet.	TCP / UDP
SOCKET_ERROR	SOCKET has errors.	TCP / UDP
SOCKET_ABORTED	SOCKET connection is interrupted.	TCP
SOCKET_READY	SOCKET connection is ready.	UDP

- **Output Updating Timing**

Name	Timing for shifting to TRUE	Timing for shifting to FALSE
bBusy	• When *bEnable* shifts to TRUE • When *bSend* shifts to TRUE	• When *bEnable* shifts to FALSE • When the task is completed
bConnected	When the TCP connection is created	• When *bEnable* shifts to FALSE • When the TCP connection is interrupted on the server side
bSent	When a data packet is sent over TCP	• When *bEnable* shifts to FALSE • When *bSend* shifts to TRUE
bRcvd	When a data packet is received over TCP	• When *bEnable* shifts to FALSE • When *bRecvRestart* shifts to TRUE
bError	When an error occurs during running or the input value of the instruction is incorrect	When *bEnable* shifts to FALSE

- **Timing Diagram**



- **Function**

  Use the FB instruction (DFB_TCP_Client) to create TCP connection so as to send or receive TCP data packets.

  Note: DFB_TCP_Client, DFB_TCP_Server, and DFB_UDP_Socket instructions can be used in a maximum of 32 groups at the same time.

- **Programming Example**

  This example uses the FB instruction (DFB_TCP_Client) and Server (IP address: 192.168.1.111, Port: 25000) to establish connection and send 200 Bytes packet. It is expected that Server will send back the same 200 Bytes package. After Server has finished sending back the package, 200 Bytes package will be resent.

```
PROGRAM TCP_Client
VAR
      FB0: DFB_TCP_C1ient;
      iIndex: INT;
      blni, FB0_bEnable, FB0_bSend, FB0_bRecvRestart, FB0_bBusy: BOOL;
      FB0_bConnected, FB0_bSent, FB0_bRcvd, FB0_bError: BOOL;
      FB0_SocketInfo: DL_EthernetLib_AX3.tcpClientSocketlnfo;
      FB0_ErrorID: DL_EthernetLib_AX3.DFB_SOCKET_ERROR;
      FB0_Status: DL_EthernetLib_AX3.DFB_SOCKET_STATUS;
      FB0_uiRcvdLen: UINT;
      byar_Send: ARRAY[0..199] OF BYTE;
      byar_Recv: ARRAY[0..499] OF BYTE;
      bError: BOOL;
END_VAR

IF bIni THEN
      FB0_SocketInfo.byIPAddr[0]:= 192;
```

```
      FB0_SocketInfo.bylPAddr[1]:= 168;
      FB0_SocketInfo.byIPAddr[2]:= 1;
      FB0_SocketInfo.byIPAddr[3]:= 111;
      FB0_SocketInfo.uiRPort:= 25000;
      FB0_SocketInfo.uiTimeOut:= 1000;
      FB0_SocketInfo.uiKeepAliveTimeout:=0;
      FB0_SocketInfo.bReconnect:= TRUE;
      FB0_SocketInfo.uiSetValue:= 200;
      FB0_SocketInfo.uiSendBuf:= ADR(byar_Send);
      FB0_SocketInfo.uiSendLen:= 200;
      FB0_SocketInfo.pRecvBuf:= ADR(byar_Recv);
      FB0_SocketInfo.uiRecvBufSize := 500;
      FB0_SocketInfo.recvCondition:= DFB_SOCK_RECV_MODE.DFB_SOCK_MODE_SPECIFIC_LENGTH;
      ilndex:=l;
      blni:= FALSE;
END_IF
CASE iIndex OF
      1:
           FB0_bEnable:= TRUE;
           byar_Send[0]=1;
                IF FB0_bConnected THEN
                ilndex:= 2;
            END_IF
      2:
                IF FB0_bError THEN
                ilndex:= 10;
            ELSE
                FB0_bSend:= TRUE;
                ilndex:= 3;
            END_IF
      3:
            IF FB0_bRcvd THEN
            FB0_bRecvRestart:= FALSE;
            FB0_bSend:= FALSE;
            ilndex:= 4;
            END_IF
                IF FB0_bError THEN ilndex:= 10;
            END_IF
      4:
            IF byar_Send[0] <> byar_Recv[0] THEN
            bError:= TRUE;
            END_IF
            byar_Recv[0]:=0;
            FB0_bRecvRestart:= TRUE;
            iIndex:=2;
      10:
            ilndex:=l;
            FB0_bEnable:= FALSE;
            FB0_bSend:= FALSE;
            FB0_bRecvRestart:= FALSE;
END_CASE

FB0(
      bEnable:=FB0_bEnable ,
      Socketlnfo:= FB0_SocketInfo ,
      bSend:= FB0_bSend ,
      bRecvRestart:= FB0_bRecvRestart,
      bBusy=> FB0_bBusy,
      bConnected=> FB0_bConnected,
      bSent=> FB0_bSent ,
      bRcvd=> FB0_bRcvd ,
      bError=> FB0_bError ,
      ErrorID=> FB0_ErrorID ,
      Status=> FB0_Status ,
      uiRcvdLen=> FB0 uiRcvdLen );
```

- **Supported Devices**
  - AX-3, AX-864E30xE2T
    Note: AX-332E, AX-864E30xE2T (Library firmware version: 1.0.6.0 or later)
- **Library**
  - DL_EthernetLib.library
    Note: From version 1.0.6.0, library DL_ EthernetLib_AX3 is changed to DL_EthernetLib.

8.2. DFB_TCP_Server

DFB_TCP_Server sends or receives TCP data packets.

FB/FC	Instruction	Graphic Expression	ST Language
FB	DFB_ TCP_Server		DFB_TCP_Server(bEnable:= , SocketInfo:= , bSend:= , bRecvRestart:= , bBusy=> , bConnected=> , bSent=> , bRcvd=> , bError=> , ErrorID=> , Status=> , uiRcvdLen=>);

- **Inputs**

Name	Function	Data Type	Setting Value (Default value)
bEnable	Run the function block.*1*2	BOOL	TRUE/FALSE (FALSE)
SocketInfo	Connection information on Server	tcpServerSocketInfo	—
bSend*3	Send data packets. (Triggered by the rising-dege)	BOOL	TRUE/FALSE (FALSE)
bRecvRestart*3	Restart to receive data packets*2 (Triggered by the rising-dege)	BOOL	TRUE/FALSE (FALSE)

***Note**:

1. As soon as this function block is running, a TCP connection will start to be created. Once connected, the output *bConnected* will be ON.
2. After the function block is running, it starts receiving data packets. When the data receiving is completed and stopped, *bRcvd* will be ON. If you shift *bRecvRestart* to ON, the FB will restart to receive data.
3. When running this function block, only after establishing a connection with Client (bConnnected is On) can the bSend and bRecvRestart parameters be triggered as valid.

    - **tcpServersocketInfo**

Name	Function	Data Type	Setting Value (Default value)
byIPAddr	The IP address on Client side allowed to be connected.	ARRAY [0..3] OF BYTE	[0.0.0.0]: No limit.
uiLPort	Communication ports on local device	UINT	0: Illegal values. 1–65535 (0)
uiTimeout	Communication time–out (Unit: ms)	UINT	0: No time–out. 1–65535 (0)
uiKeepAliveTimeout	The time that the connection keeps alive. (Unit: sec)	UINT	0: No time–out. 1–65535 (0)
pSendBuf	The memory address of data to be sent	POINT TO BYTE	—
uiSendLen	The length of data to be sent (Unit: Byte)	UINT	0–8192 (0)
pRecvBuf	The memory address where the received data to be stored	POINT TO BYTE	—
uiRecvBufSize	The memory size of received data (Unit: Byte)	UINT	0–8192 (0)
uiSetValue	The setting value of *recvCondition*	UINT	(0)
recvCondition	Conditions for data receiving completion	DFB_SOCK_RECV_MODE	(DFB_SOCKET_NO_RECEVING)

- **DFB_SOCK_RECV_MODE**

Name	Description
DFB_SOCK_MODE_NO_RECEVING	No receiving data mode.
DFB_SOCK_MODE_SPECIFIC_LENGTH	Specific data length mode: A specific quantity of data is received and the receiving task is completed. The data length can be specified via *uiSetValue*. (Unit: Byte)
DFB_SOCK_MODE_SPECIFIC_SINGLE_CHAR	Specific end character mode: The data received ends with a specific character (1 Byte). The end character can be configured via *uiSetValue*.

Name	Description
	e.g.: If *uiSetValue* is set to 16#00000D0A, the end character will be 16#0A.(*1*2)
DFB_SOCK_RECV_MODE_DFB_SOCK_MODE_SPECIFIC_TWO_CHARS	Specific two end characters mode: The data received ends with the two specific characters (2 Bytes) The end character can be configured via *uiSetValue* e.g.: If *uiSetValue* is set to 16#00000D0A, the end characters will be 16#0D0A.
DFB_SOCK_RECV_MODE_DFB_SOCK_MODE_SPECIFIC_START_CHAR_AND_SPECIFIC_END_CHAR	Specific start character and end character mode: The data received starts with a specific character, and ends with a specific character. Both the start and the end character can be configured via *uiSetValue*. e.g.: If *uiSetValue* is set to 16#00003A0A, the start character will be 16#3A and the end character is 16#0A. (*1*2)
DFB_SOCK_RECV_MODE_DFB_SOCK_MODE_ANY_LENGTH	Any length mode: The receiving ends with a complete data of any length. (*1)

***Note**:

1. When the length of received data reaches the limit set in pRecvLen, the receiving task will be completed.
2. The data length includes both start and end characters.

- **Outputs**

Name	Function	Data Type	Output Range(Default value)
bBusy	TRUE when the instruction is running	BOOL	TRUE/FALSE (FALSE)
bConnected	TCP is connected.	BOOL	TRUE/FALSE (FALSE)
bSent	Sending completed	BOOL	TRUE/FALSE (FALSE)
bRcvd	Receiving completed	BOOL	TRUE/FALSE (FALSE)
bError	TRUE if an error occurs	BOOL	TRUE/FALSE (FALSE)
ErrorID	Indicates the error code if an error occurs.	DFB_SOCKET_ERROR	(DFB_SOCK_NO_ERROR)
Status	The running status of socket	DFB_SOCKET_STATUS	(SOCKET_CLOSED)
uiRcvLen	The length of received data	UINT (Unit: Byte)	(0)

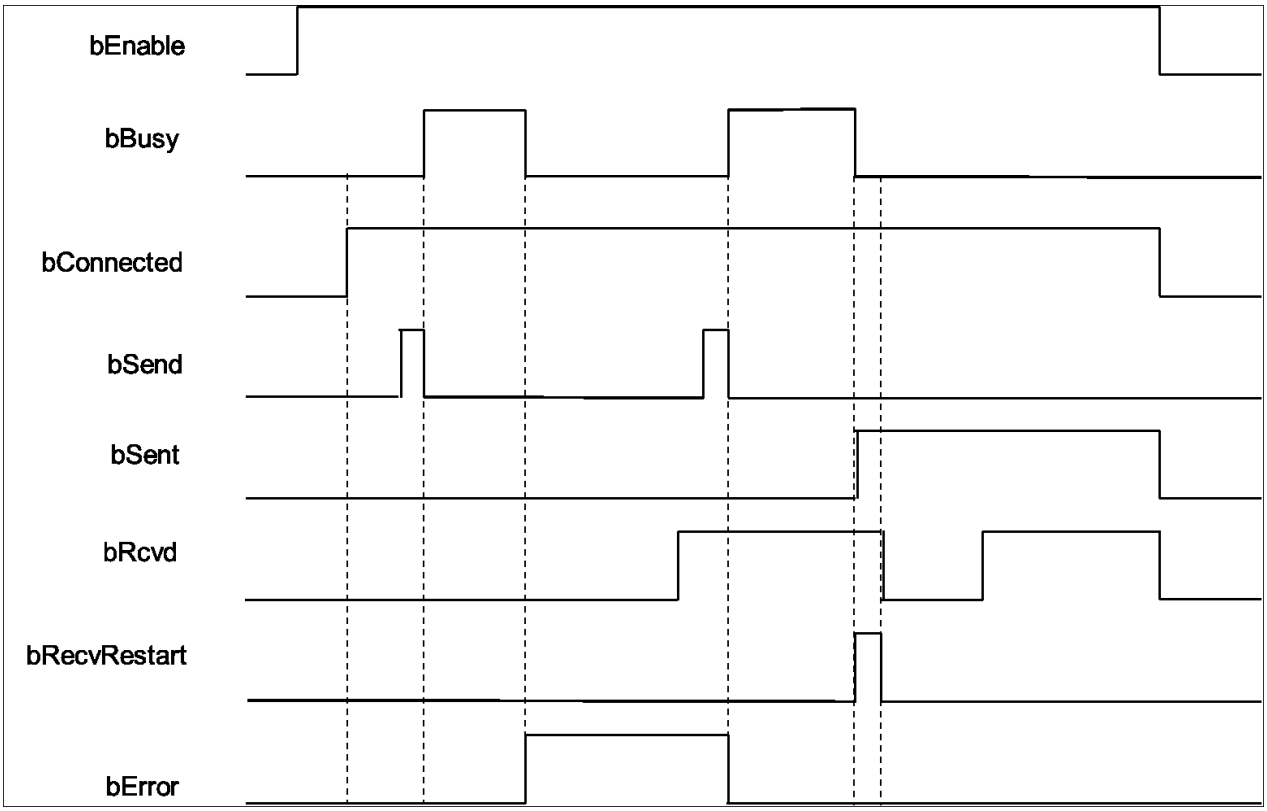
- **DFB_SOCKET_STATUS**

Name	Description	Applicable Protocol
SOCKET_CLOSED	SOCKET connection is closed.	TCP / UDP
SOCKET_CONNECTING	SOCKET is connecting.	TCP
SOCKET_CONNECTED	SOCKET is connected.	TCP
SOCKET_SENDING	SOCKET is sending the data packet.	TCP / UDP
SOCKET_SENT	SOCKET has sent the data packet.	TCP / UDP
SOCKET_RECEIVED	SOCKET has received the data packet.	TCP / UDP
SOCKET_ERROR	SOCKET has errors.	TCP / UDP
SOCKET_ABORTED	SOCKET connection is interrupted.	TCP
SOCKET_READY	SOCKET connection is ready.	UDP

- **Output Updating Timing**

Name	Timing for shifting to TRUE	Timing for shifting to FALSE
bBusy	• When *bEnable* shifts to TRUE • When *bSend* shifts to TRUE	• When *bEnable* shifts to FALSE • When the task is complete
bConnected	When the TCP connection is created	• When *bEnable* shifts to FALSE • When the TCP connection is interrupted on the server side
bSent	When a data packet is sent over TCP	• When *bEnable* shifts to FALSE • When *bSend* shifts to TRUE
bRcvd	When a data packet is received over TCP	• When *bEnable* shifts to FALSE • When *bRecvRestart* shifts to TRUE
bError	When an error occurs during running or the input value of the instruction is incorrect	When *bEnable* shifts to FALSE

- **Timing Diagram**



- **Function**

  Use the FB instruction (DFB_TCP_Server) to create TCP connection so as to send or receive TCP data packets.

  Note: DFB_TCP_Client, DFB_TCP_Server, and DFB_UDP_Socket instructions can be used in a maximum of 32 groups at the same time.

- **Programming Example**

  - This example uses the FB instruction (DFB_TCP_ Server) to open TCP (Port: 502) and restrict the IP address of Client to be 192.168.1.111, and receive the package of 200 Bytes data length. After receiving is finished, the same 200 Bytes package will be sent back.

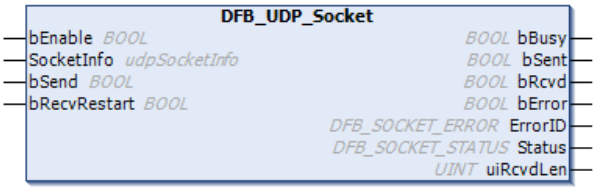
```
PROGRAM TCP_Server
VAR
      bVar0 : BOOL := TRUE;
      FB0 : DFB_TCP_Server;

 bEnable_Var,bSend_Var,bRestart_Var,bBusy_Var,bConnected_Var,bSent_Var,bRcvd_Var,bErro
r_Var:BOOL;
      RemoteInfo_Var: tcpServerSocketlnfo;
      ErrorID_Var: DFB_SOCRET_ERROR;
      Status_Var: DFB_SOCKET_STATUS;
      uiRcvLen_Var: UINT;
      by_arVar0,by_arVarl:ARRAY[0..2000] OF BYTE;
END_VAR

IF bVar0 THEN
      bEnable_Var:=TRUE;
      RemoteInfo_Var.byIPAddr[0]:=192;
      RemoteInfo_Var.byIPAddr[1]:=168;
      RemoteInfo_Var.byIPAddr[2]:=1;
      RemoteInfo_Var.byIPAddr[3]:=111;
      RemoteInfo_Var.uiLPort:=502;
      RemoteInfo_Var.uiTimeOut:=3000;
      RemoteInfo_Var.uiKeepAliveTimeout:=300;
      RemoteInfo_Var.pSendBuf:=ADR(by_arVar0);
```

```
      RemoteInfo_Var.uiSendLen:=1000;
      RemoteInfo_Var.pRecvBuf:=ADR(by_arVar1);
      RemoteInfo_Var.uiRecvBufSize:=2000;
      RemoteInfo_Var.uiSetValue:=1000;
      RemoteInfo_Var.recvCondition:=DFB_SOCR_RECV_MODE.DFB_SOCK_MODE_SPECIFIC_LENGTH;
END_IF
FB0(
      bEnable:=bEnable_Var ,
      SocketInfo:=RemoteInfo_Var ,
      bSend:=bSend_Var ,
      bRecvRestart:=bRecvRestart_Var,
      bBusy=>bBusy_Var,
      bConnected=> bConnected_Var,
      bSent=>bSent_Var,
      bRcvd=>bRcvd_Var,
      bError=>bError_Var,
      ErrorID=>ErrorID_Var,
      Status=>Status_Var,
      uiRcvdLen=> uiRcvdLen_Var);
```

- **Supported Devices**
  - AX-3, AX-864E30xE2T
    Note: AX-332E, AX-864E30xE2T (Library firmware version: 1.0.6.0 or later)
- **Library**
  - DL_EthernetLib.library
    Note: From version 1.0.6.0, library DL_ EthernetLib_AX3 is changed to DL_EthernetLib.

8.3. DFB_UDP_Socket

DFB_UDP_Socket sends or receives UDP data packets.

FB/FC	Instruction	Graphic Expression	ST Language
FB	DFB_ UDP_Socket		DFB_UDP_Socket(bEnable:= , RemoteInfo:= , bSend:= , bRecvRestart:= , bBusy=> , bSent=> , bRcvd=> , bError=> , ErrorID=> , Status=> , uiRcvdLen=>);

- **Inputs**

Name	Function	Data Type	Setting Value (Default Value)
Enable	Run the function block.	BOOL	TRUE/FALSE (FALSE)
SocketInfo	Connection information of socket	udpSocketInfo	—
bSend	Send data packets. (Triggered by the rising-dege)	BOOL	TRUE/FALSE (FALSE)
bRecvRestart	Restart to receive data packets* (Triggered by the rising-dege)	BOOL	TRUE/FALSE (FALSE)

*__Note__: After the function block is running, it starts receiving data packets. When the data receiving is completed and stopped, *bRcvd* will be ON. If you shift *bRecvRestart* to ON, the FB will restart to receive data.

- **udpSocketInfo**

Name	Function	Data Type	Setting Value (Default value)
byIPAddr	The slave IP address allowed to be connected	ARRAY [0..3] OF BYTE	[0.0.0.0]: No limit.
uiLPort(*1)	Communication ports on local device	UINT	0: Use a random port number to send data packets. 0–65535 (0)
uiRPort(*1*2)	Communication ports on remote device	UINT	0: Receive data packets from a random port.

Name	Function	Data Type	Setting Value (Default value)
			1–65535 (0)
pSendBuf	The memory address of data to be sent	POINT TO BYTE	—
uiSendLen	The length of data to be sent (Unit: Byte)	UINT	0–8192
pRecvBuf	The memory address where the received data to be stored	POINT TO BYTE	—
uiRecvBufSize	The memory size of received data (Unit: Byte)	UINT	0–8192 (0)
uiSetValue	The setting value of *recvCondition*	UINT	(0)
recvCondition	Conditions for data receiving completion	DFB_SOCK_RECV_MODE	(DFB_SOCKET_NO_RECEVING)

***Note**:

1. The values of *uiLPort* and *uiRPort* cannot be 0 at the same time.
2. UDP data packets are not allowed to be sent when *uiRPort* is set to 0.

- **DFB_SOCK_RECV_MODE**

Name	Description
DFB_SOCK_MODE_NO_RECEVING	No receiving data mode
DFB_SOCK_MODE_SPECIFIC_LENGTH	Specific data length mode: A specific quantity of data is received and the receiving task is completed. The data length can be specified via *uiSetValue*. (Unit: Byte)
DFB_SOCK_MODE_SPECIFIC_SINGLE_CHAR	Specific end character mode: The data received ends with a specific character (1 Byte). The end character can be configured via *uiSetValue*. e.g.: If *uiSetValue* is set to 16#00000D0A, the end character will be 16#0A.(*1*2)
DFB_SOCK_RECV_MODE_DFB_SOCK_MODE_SPECIFIC_TWO_CHARS	Specific two end characters mode: The data received ends with the two specific characters (2 Bytes) The end character can be configured via *uiSetValue* e.g.: If *uiSetValue* is set to 16#00000D0A, the end characters will be 16#0D0A. (*1*2)

Name	Description
DFB_SOCK_RECV_MODE_DFB_SOCK_MODE _SPECIFIC_START_CHAR_AND_SPECIFIC_END_CHAR	Specific start character and end character mode: The data received starts with a specific character, and ends with a specific character. Both the start and the end character can be configured via *uiSetValue*. e.g.: If *uiSetValue* is set to 16#00003A0A, the start character will be 16#3A and the end character is 16#0A. (*1*2)
DFB_SOCK_RECV_MODE_DFB_SOCK_MODE_ANY_LENGTH	Any length mode: The receiving ends with a complete data of any length.(*1)

***Note**:

1. When the length of received data reaches the limit set in pRecvLen, the receiving task will be completed.
2. The data length includes both start and end characters.

- **Outputs**

Name	Function	Data Type	Output Range (Default value)
bBusy	TRUE when the instruction is running	BOOL	TRUE/FALSE (FALSE)
bSent	TCP is connected.	BOOL	TRUE/FALSE (FALSE)
bRcvd	Sending complete	BOOL	TRUE/FALSE (FALSE)
bError	Receiving complete	BOOL	TRUE/FALSE (FALSE)
ErrorID	TRUE if an error occurs	DFB_SOCKET_ERROR	(DFB_SOCK_NO_ERROR)
Status	Indicates the error code if an error occurs.	DFB_SOCKET_STATUS	(SOCKET_CLOSED)
uiRcvLen	The running status of socket	UINT (Unit: Byte)	(0)

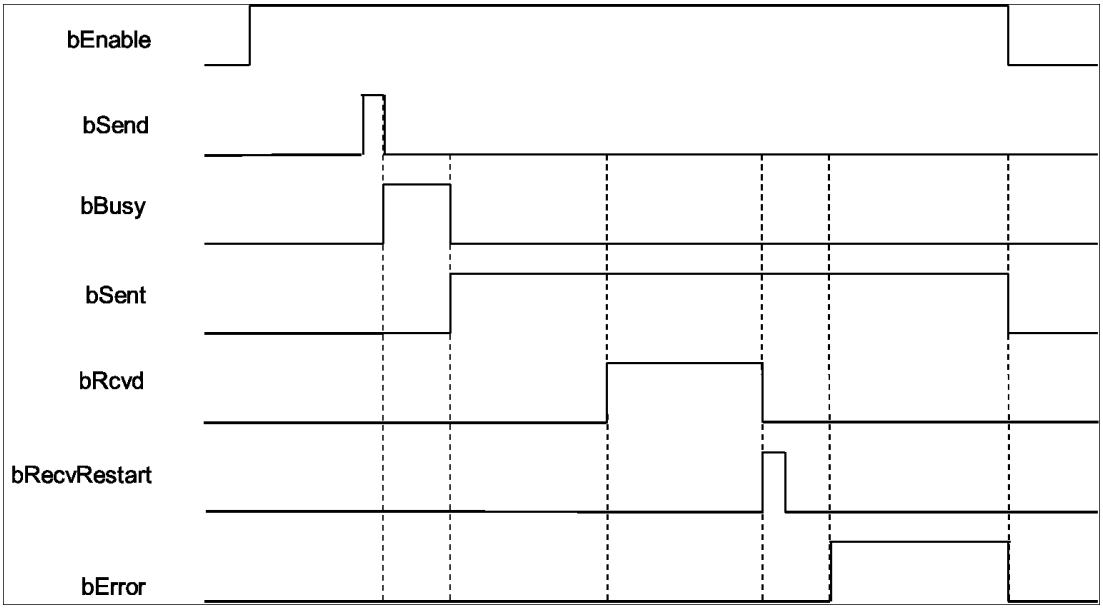
- **DFB_SOCKET_STATUS**

Name	Description	Applicable Protocol
SOCKET_CLOSED	SOCKET connection is closed.	TCP / UDP
SOCKET_CONNECTING	SOCKET is connecting.	TCP
SOCKET_CONNECTED	SOCKET is connected.	TCP
SOCKET_SENDING	SOCKET is sending the data packet.	TCP / UDP
SOCKET_SENT	SOCKET has sent the data packet.	TCP / UDP
SOCKET_RECEIVED	SOCKET has received the data packet.	TCP / UDP
SOCKET_ERROR	SOCKET has errors.	TCP / UDP
SOCKET_ABORTED	SOCKET connection is interrupted.	TCP
SOCKET_READY	SOCKET connection is ready.	UDP

- **Output Updating Timing**

Name	Timing for shifting to TRUE	Timing for shifting to FALSE
bBusy	When *bSend* shifts to TRUE	• When *bEnable* shifts to FALSE • When the task is completed
bSent	When the UDP data packet has been sent successfully	• When *bEnable* shifts to FALSE • When *bSend* shifts to TRUE
bRcvd	When the UDP data packet has been received	• When *bEnable* shifts to FALSE • When *bRecvRestart* shifts to TRUE
bError	When an error occurs during running or the input value of the instruction is incorrect	When *bEnable* shifts to FALSE

- **Timing Diagram**



- **Function**

  Use the FB instruction (DFB_UDP_Socket) to send or receive UDP data packets.

  Note: DFB_TCP_Client, DFB_TCP_Server, and DFB_UDP_Socket instructions can be used in a maximum of 32 groups at the same time.

- **Programming Example**

  This example uses the FB instruction (DFB_UDP_Socket) to send a UDP package of 1000 Bytes to the IP address 192.168.1.111 (port: 3000) and receive a package of 1000 Bytes.

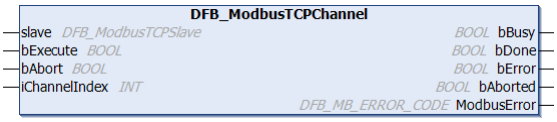
```
PROGRAM UDP
VAR
      bVar0 : BOOL
      FB0: DL_EthernetLib_AX3.DFB_UDP_Socket;
      bEnable_Var,bSend_Var,bRestart_Var,bBusy_Var,bSent_Var,bRcvd_Var,bError_Var:BOOL;
      RemoteInfo_Var: DL_EthernetLib_AX3.udpSocketInfo;
      ErrorID_Var: DL_EthernetLib_AX3.DFB_SOCEKT_Error;
      Status_Var: DL_EthernetLib_AX3.DFB_SOCEKT_Error_STATUS;
```

```
      uiRcvLen_Var: UINT;
      by_arVar0,by_arVar1:ARRAY[0..2000] OF BYTE;
END_VAR

IF bVar0 THEN
      bEnable_Var:=TRUE;
      RemoteInfo_Var.byIPAddr[0]:=192;
      RemoteInfo_Var.byIPAddr[1]:=168;
      RemoteInfo_Var.byIPAddr[2]:= 1;
      RemoteInfo_Var.byIPAddr[3]:=111;
      RemoteInfo_Var.uiLPort:=2000;
      RemoteInfo_Var.uiRPort:=3000;
      RemoteInfo_Var.pRecvBuf:=ADR(by_arVar1);
      RemoteInfo_Var.uiRecvBufSize:=2000;
      RemoteInfo_Var.uiSetValue:=1000;
      RemoteInfo_Var.recvCondition:=DFB_SOCR_RECV_MODE.DFB_SOCK_MODE_SPECIFIC_LENGTH;
END_IF
FB0(
      bEnable:=bEnable_Var ,
      SocketInfo:=RemoteInfo_Var ,
      bSend:=bSend_Var ,
      bRecvRestart:=bRecvRestart_Var,
      bBusy=>bBusy_Var,
      bSent=>bSent_Var,
      bRcvd=>bRcvd_Var,
      bError=>bError_Var,
      ErrorID=>ErrorID_Var,
      Status=>Status_Var,
      uiRcvdLen=> uiRcvdLen_Var);
```

- **Supported Devices**
  - AX-3, AX-864E30xE2T
    Note: AX-332E, AX-864E30xE2T (Library firmware version: 1.0.6.0 or later)
- **Library**
  - DL_EthernetLib.library
    Note: From version 1.0.6.0, library DL_ EthernetLib_AX3 is changed to DL_EthernetLib.

8.4. DFB_ModbusTCPChannel

DFB_ModbusTCPChannel activates the channel of Modus TCP communication.

FB/FC	Instruction	Graphic Expression	ST Language
FB	DFB_ ModbusTCPChan nel		DFB_ModbusTCPChann el(slave:= , bExecute:= , bAbort:= , iChannelIndes:= , bBusy=> , bDone=> , bError=> , bAborted=> , ModbusError=>);

- **Inputs/Outputs**

Name	Function	Data type	Setting value (Default value)
Slave	Delta Modbus TCP slave device	DFB_ModbusTCPSlave	—

- **Inputs**

Name	Function	Data type	Setting value (Default value)
bExcute	Run the function block. (Triggered by the rising-dege)	BOOL	TRUE/FALSE (FALSE)
bAbort	No function	BOOL	—
iChannelIndex	Channel number	INT	0–99 (0)

- **Outputs**

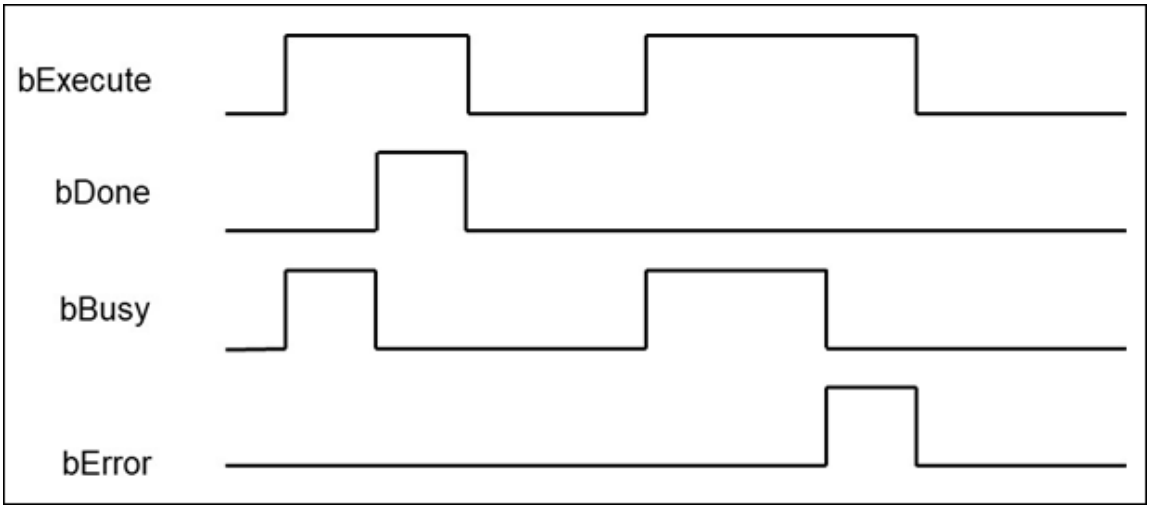
Name	Function	Data type	Output range (Default value)
bBusy	The FB instruction is running.	BOOL	TRUE/FALSE (FALSE)
bDone	The FB instruction running is complete.	BOOL	TRUE/FALSE (FALSE)
bError	FB instruction error flag	BOOL	TRUE/FALSE (FALSE)
bAborted	No function	BOOL	—
ModbusError	Error codes	DFB_MB_ERROR_C ODE	DL_MB_ERROR_CODE (UNDEFINED)

- **Output Updating Timing**

Name	Timing for shifting to TRUE	Timing for shifting to FALSE
bDone	When FB instruction running is completed	When *bExecute* shifts to FALSE

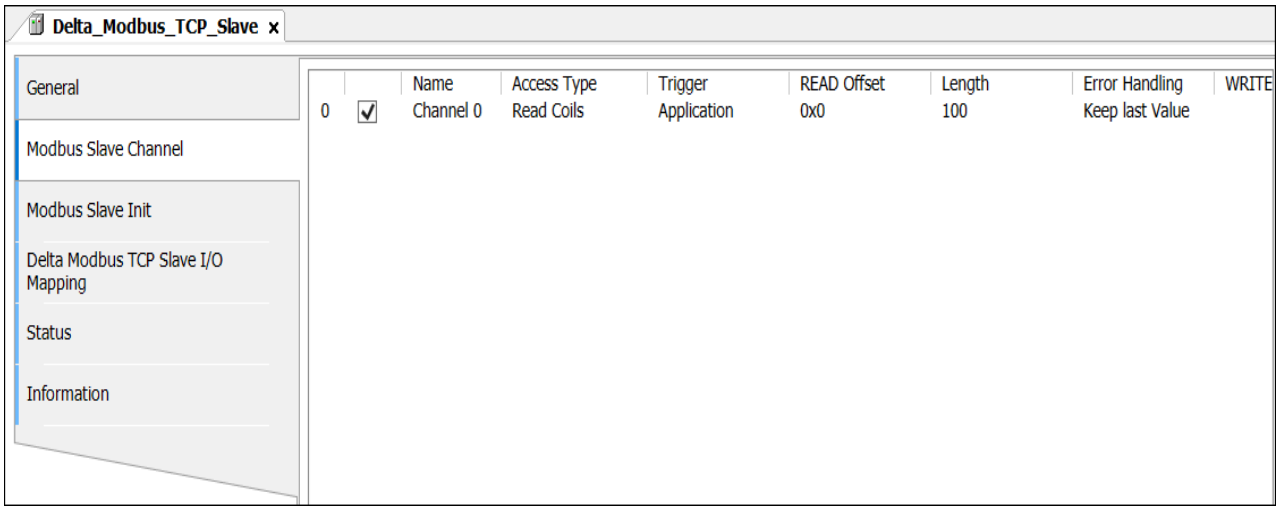
Name	Timing for shifting to TRUE	Timing for shifting to FALSE
bBusy	When FB instruction running starts	• When FB instruction running is complete • When *bExecute* shifts to FALSE
bError	When an error occurs during running or the input value of the instruction is incorrect	When *bExecute* shifts to FALSE
ModbusError		

- **Timing Diagram**



- **Function**

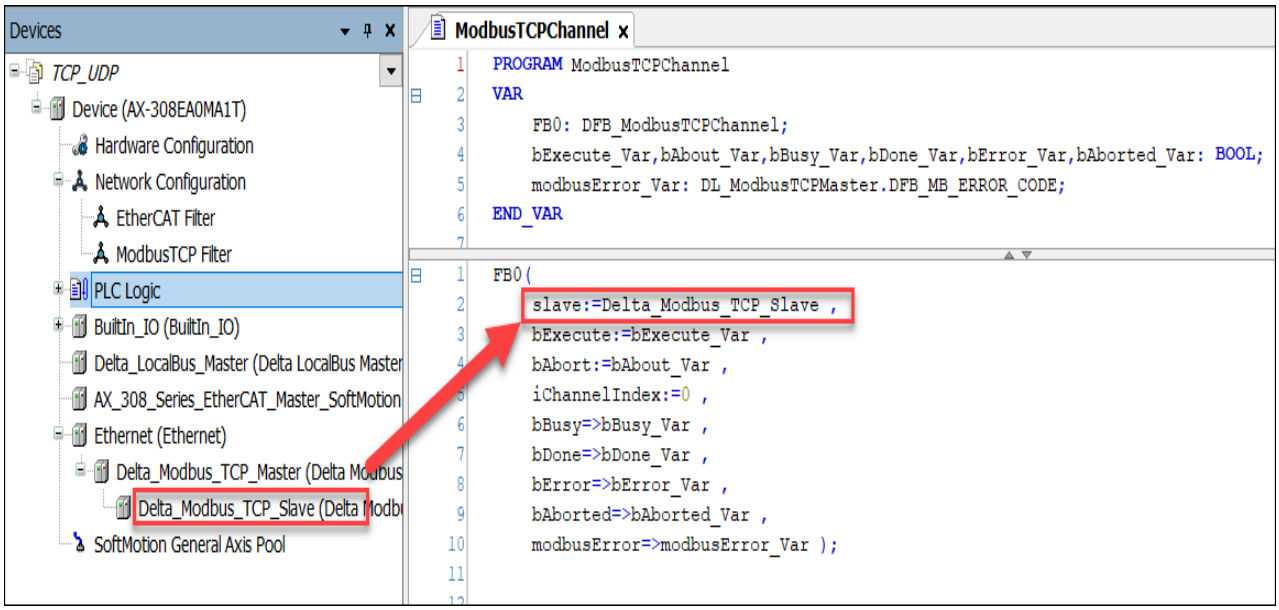
When the trigger mode of the Modbus slave channel is set to Application, the Modbus TCP request action can be triggered by DFB_ModbusTCPChannel.



Note1: For more details of Modbus TCP slave configuration, Refer to chapter 9.3 “Ethernet Communication” in *AX-3 Series Operation Manual*.
Note 2: While using, the channel must be set to Enable.
Note 3: DFB_ModbusTCPChannel and DFB_ModbusTCPRequest instructions can be used in a maximum of 32 groups at the same time.

- **Programming Example**

This example uses DFB_ModbusTCPChannel to trigger channel 0 of Modbus TCP Slave.



***Note**: The input of Slave will be the name of Delta_Modbus_TCP_Slave device.

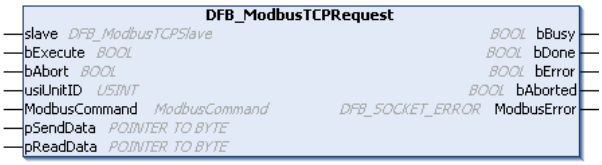
- **Supported Devices**
  - AX-3, AX-864E30xE2T, AX-C

    Note: AX-332E, AX-864E30xE2T, AX-C (Library firmware version: 1.0.6.0 or later)
- **Library**
  - DL_ModbusTCPMaster.library

    Note: From version 1.0.6.0, library DL_ModbusTCPMaster _AX3 is changed to DL_ ModbusTCPMaster.

8.5. DFB_ModbusTCPRequest

DFB_ModbusTCPRequest sends the Mobus TCP communication data.

FB/FC	Instruction	Graphic Expression	ST Language
FB	DFB_ ModbusTCPRequest		DFB_ModbusTCPRequest(slave:= bExecute:= , bAbort:= , usiUnitID:= , ModbusCommand:= , pSendData:= , pRecvData:= , bBusy=> , bDone=> , bError=> , bAborted=> , ModbusError=>);

- **Inputs/Outputs**

Name	Function	Data type	Setting value (Default value)
Slave	Delta Modbus TCP slave device	DFB_ModbusTCPSlave	—

- **Inputs**

Name	Function	Data type	Setting value (Default value)
bExecute	Run the function block. (Triggered by the rising-dege)	BOOL	TRUE/FALSE (FALSE)
bAbort	No function	BOOL	—
usiSlaveAddr	Slave station number	USINT	1–247
ModbusCommand	Modbus parameter setting	ModbusCommand	—
pSendData	The memory address of data to be sent	POINTER TO BYTE	—
pRecvData	The memory address of received data to be stored	POINTER TO BYTE	—

- **Modbus Command**

Name	Function	Data Type	Output Range (Default value)
FunctionCode	Modbus function codes	DFB_MB_FUNC_CODE	Supported function codes: 0x01: Read Coils 0x02: Read Discrete Inputs 0x03: Read Holding Registers

Name	Function	Data Type	Output Range (Default value)
			0x04: Read Input Registers 0x05: Write Single Coil 0x06: Write Single Register 0x0F: Write Multiple Coils 0x10: Write Multiple Registers 0x17: Read/Write Multiple Registers (0x03)
uiReadOffset	The start address of memory to be read.	UINT	0–65535 (0)
uiReadLen	The data length of the memory to be read.	UINT	Coil: 1–1920 Register: 1–120 (1)
uiWriteOffset	The start address of memory to be written.	UINT	0–65535 (0)
uiWriteLen	The data length of the memory to be written	UINT	Coil: 1–1920 Register: 1–120. (1)

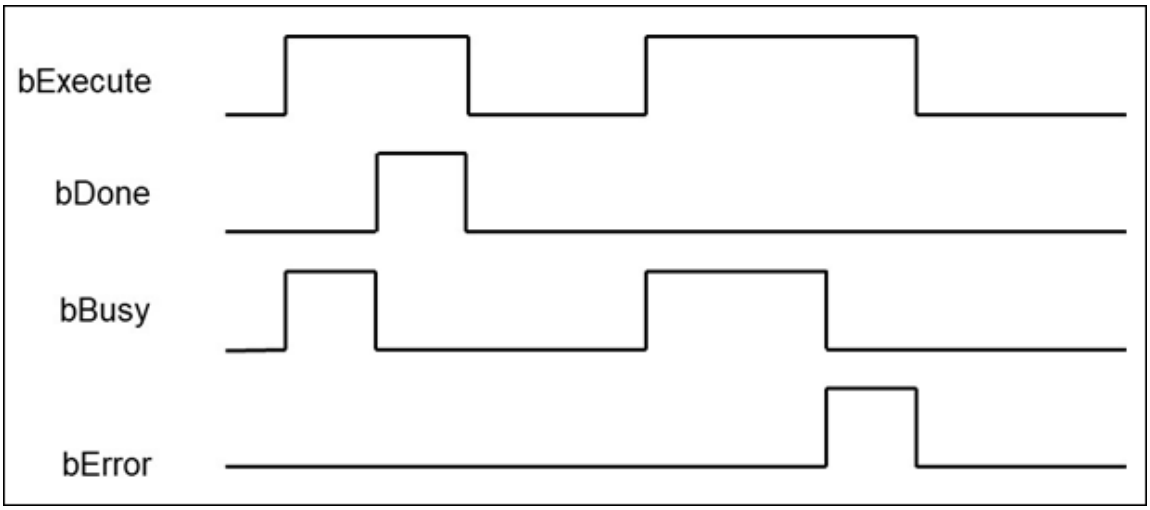
- **Outputs**

Name	Function	Data Type	Output Range (Default value)
bBusy	TRUE when the instruction is running	BOOL	TRUE/FALSE (FALSE)
bDone	The running of FB is completed.	BOOL	TRUE/FALSE (FALSE)
bError	TRUE if an error occurs	BOOL	TRUE/FALSE (FALSE)
bAborted	No function	BOOL	—
ModbusError	Error code	DFB_MB_ERROR_CODE	DL_MB_ERROR_CODE (DFB_UNDEFINED)

- **Output Updating Timing**

Name	Timing for shifting to TRUE	Timing for shifting to FALSE
bDone	When the running of FB is completed	When *bExecute* shifts to FALSE
bBusy	When FB instruction running starts	• When the running of FB is completed • When *bExecute* shifts to FALSE
bError	When an error occurs during the running conditions or there are invalid input values	When *bExecute* shifts to FALSE
ModbusError		

- **Timing Diagram**



- **Function**

  The FB instruction (DFB_ModbusTCPRequest) is used for sending Modbus communication data. You must finish the configuration of Delta_Modbus_TCP_Master and Delta_Modus_TCP_Slave before using this instruction (For more details, refer to Section 9.3 Ethernet Communication in *AX-3 Series Operation Manual*).

  Note: DFB_ModbusTCPChannel and DFB_ModbusTCPRequest instructions can be used in a maximum of 32 groups at the same time.

- **Programming Example**

  This example uses DFB_ModbusTCPRequest to send standard Modbus command (0x17) for reading a 100–word long data in the slave station (Delta_Modbus_TCP_Slave), which the start address is 0x0000, and writing a data of 100–word long to the start address 0x0100 in the memory.

```
VAR
      FB0: DFB_ModbusTCPRequest;
      bVar0: BOOL:=TRUE;
      bExecute_Var,bAbort_Var,bBusy_Var,bDone_Var,bError_Var,bAborted_Var:BOOL;
      usiUnitID_Var: USINT;
      ModbusCommand_Var: DL_ModbusTCPMaster_AX3.ModbusCommand;
      ar_byVar0,ar_byVar1:ARRAY[0..2000] OF BYTE;
      ModbusError_Var: DL_ModbusTCPMaster_AX3.DFB_MB_ERROR_CODE;
END_VAR

IF bVar0 THEN
      bVar0:=FALSE;

 ModbusCommand_Var.FunctionCode:=DL_ModebysTCPMaster_AX3.DFB_MODBUS_FUNC.ReadWrite_Multiple_
Registers;
      ModbusCommand_Var.uiReadLen:=100;
      ModbusCommand_Var.uiReadOffset:=16#0000;
      ModbusCommand_Var.uiWriteLen:=100;
      ModbusCommand_Var.uiWriteOffset:=16#0000;
      usiUnitID_Var:=12;
      bExecute_Var:=TRUE;
END_IF
FB0(
      slave:=Delta_Modbus_TCP_Slave,
      bExecute:=bExecute_Var,
      bAbort:=bAbort_Var,
      usiUnitID:=usiUnitID_Var,
      ModbusCommand:=ModbusCommand_Var,
      pSendData:=ADR(ar_byVar0),
      pReadData:=ADR(ar_byVar1),
      bBusy=>bBusy_Var,
```

```
    bDone=>bDone_Var,
    bError=>bError_Var,
    bAborted=>bAborted_Var,
    ModbusError=>Modbus_Var );
```

- **Supported Devices**
  - AX-3, AX-864E30xE2T, AX-C
    Note: AX-332E, AX-864E30xE2T, AX-C (Library firmware version: 1.0.6.0 or later)
- **Library**
  - DL_ModbusTCPMaster.library
    Note: From version 1.0.6.0, library DL_ModbusTCPMaster _AX3 is changed to DL_ ModbusTCPMaster.

8.6. DFB_SetIPConfig

DFB_SetIPConfig sets the controller network configuration.

FB/FC	Instruction	Graphic Expression	ST Language
FB	DFB_SetIPConfig		DFB_SetIPConfig (bExecute:= , byLanPort:= , strIPAddress:= , strSubMask:= , strGateway:= , bDone=> , bBusy=> , bError=> , eErrorID=>);

- **Inputs**

Name	Function	Data Type	Setting Value (Default Value)
bExecute	Run the function block. (Triggered by the rising-dege)	BOOL	TRUE/FALSE (FALSE)
byLanPort	Lan port	Byte	(1)
strIPAddress	IP address	String(15)	(‘192.168.1.5’)
strMask	Network mask	String(15)	(‘255.255.255.0’)
strGateway	Gateway address	String(15)	(‘192.168.1.1’)

- **Outputs**

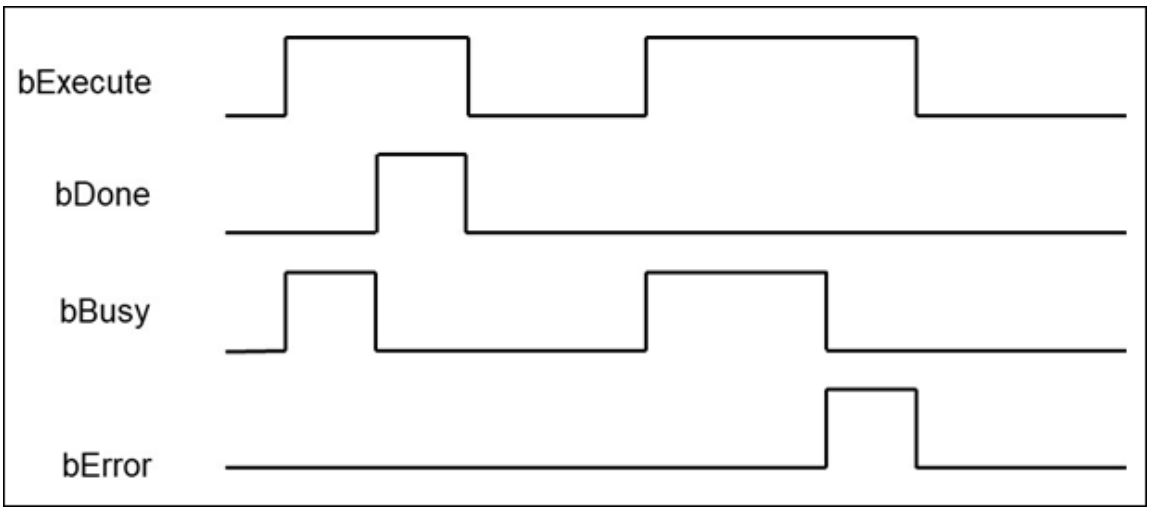
Name	Function	Data Type	Setting Value (Default Value)
bDone	The running of FB is complete.	BOOL	TRUE/FALSE (FALSE)
bBusy	The FB instruction is running.	BOOL	TRUE/FALSE (FALSE)
bError	FB instruction error	BOOL	TRUE/FALSE (FALSE)
eErrorID	Error code	DFB_SOCKET_ERROR	DFB_SOCKET_ERROR (DFB_SOCK_NO_ERROR)

- **Output Updating Timing**

Name	Timing for shifting to TRUE	Timing for shifting to FALSE
bDone	When the running of FB is complete	When *bExecute* shifts to FALSE
bBusy	When FB instruction running starts	• When the running of FB is complete • When *bExecute* shifts to FALSE

Name	Timing for shifting to TRUE	Timing for shifting to FALSE
bError	When an error occurs during the running or there are incorrect input values	When *bExecute* shifts to FALSE
ModbusError		

- **Timing Diagram**



- **Function**

  You can set the network configuration of the AX series controller by DFB_SetIPConfig.

  Note 1: If the controller has multiple independent network ports, the ports must be set to different domains, such as 192.168.1.x and 192.168.2.y.

  Note 2: If the AX controller network configuration is set by DFB_SetIPConfig, and the input parameter *strGateway* is not set or set to 0.0.0.0, the GateWay setting of the AX controller will be cleared.

  Note 3: If **Enable Gateway and DNS Setting** of the network port does not work, you cannot change the gateway setting by DFB_SetIPConfig.

- **Programming Example**

  In this example, the FB instruction (DFB_SetIPConfig) sets the IP address of the X1 network port of the controller to 192.168.1.15.

```
VAR
    FB : DFB_SetIPConfig;
    bExecute_Var: BOOL;
    bDone_Var: BOOL;
    bBusy_Var: BOOL;
    bError_Var: BOOL;
    eErrorID_Var: DL_EthernetLib.DFB_SOCKET_ERROR;
END_VAR
FB(
    bExecute:= bExecute_Var,
    byLanPort:= 1,
    strlPAddress:= '192.168.1.15',
    strSubnetMask:= '255.255.255.0',
    scrOateway:= '192.168.1.1',
    bDone=> bDone_Var,
    bBusy=> bBuay_Var,
    bError=> bError_Var,
    eErrorID=> eErrorlD Var);
```

- **Supported Devices**
  - AX-3 series (firmare v1.0.6.0 and later), AX-864E30xE2T
    Note: AX-332E (firmware 1.0.5.8 and later)
    AX-332E, AX-864E30xE2T (Library firmware version: 1.0.6.0 or later)
- **Library**
  - DL_EthernetLib.library (v1.0.5.1 and later).

8.7. Error Codes and Troubleshooting

- DFB_SOCKET_ERROR

Description	Cause of Error	Corrective Action
DFB_SOCK_ERR_NO_ERROR	No errors	—
DFB_SOCK_ERR _INITIALIZE_FAILED	Socket connection failed.	• Check if the server exists. • Make sure the server configuration is correct.
DFB_SOCK_ERR _CONNREFUSED	Socket connection refused.	Make sure the server configuration is correct.
DFB_SOCK_ERR TIMEDOUT	Server time–out error	• Make sure the internet connection is normal. • Make sure the server configuration is correct.
DFB_SOCK_ERR NOTCONNECTED	Socket has not been connected.	• Wait for Socket being connected. • Make sure the server configuration is correct.
DFB_SOCK_ERR CLOSED	FB instruction has not yet effective.	Make sure the input *bEnable* is ON.
DFB_SOCK_ERR INVALID_SETTING	Invalid setting values for FB instruction	Check if the setting value of *uiSetValue* is correct.
DFB_INVALID_BUFFER	Invalid memory address	Make sure the memory addresses given to *pSenbuf* and *pRecvbuf* are correct.
DFB_INVALID_LENGTH	Invalid setting value for data length	Make sure the input values of *uiSendLen* and *uiRecvLen* are correct.
DFB_IP_ERR	Invalid IP setting	Make sure the IP setting is correct.
DFB_IP_SET_ERR	Setting IP fail	—
DFB_MASK_ERR	Invalid Mask setting	Make sure the Mask setting is correct.
DFB_MASK_SET_ERR	Setting Mask fail	—
DFB_GATEWAY_ERR	Invalid Gateway setting	Make sure the Gateway setting is correct.
DFB_GATEWAY_SET_ERR	Gateway setting failed	—
DFB_LANPORT_ERR	Invalid Lan Port setting	Make sure the LanPort is correct（Only supports NetworkInterface enumeration scope）

- DFB_MB_ERROR_CODE

Description	Cause of Error	Corrective Action
DFB_NO_ERROR	No errors	—

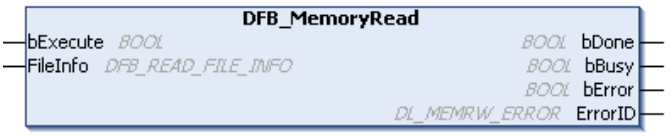
Description	Cause of Error	Corrective Action
DFB_ILLEGAL_FUNCTION	Unsupported function code.	Check on the correctness of the function code you're using.
DFB_ILLEGAL_DATA_ADDRESS	Illegal memory address to write and read.	Check on the correctness of memory address you intend to write and read.
DFB_ILLEGAL_DATA_VALUE	Illegal data values responded by slave.	Check if the slave wires function normally as well as the proper wiring.
DFB_SLAVE_DEVICE_FAILURE	Slave failure	Check slave settings and statuses.
DFB_ACKNOWLEDGE	Slave has received request, but it takes longer to handle.	N/A
DFB_SLAVE_DEVICE_*BUSY*	Slave is *busy*.	N/A
DFB_GATEWAY_PATH_UNAVAILABLE	Wrong Gateway path	Check Gateway configuration, or Gateway is *busy*.
DFB_GATEWAY_DEVICE_FAILED_TO_RESPOND	Slave device in Gateway fails to respond.	Check if the slave wires function normally as well as the proper wiring.
DFB_RESPONSE_TIMEOUT	No response from slave in time	• Check if the duration set for the time–out is less than the responded time of slave. • Check on the correctness of the wiring.
DFB_RESPONSE_CRC_ERROR	Invalid data values responded by slave (Invalid check code)	Check on the correctness of data format responded by slave.
DFB_RESPONSE_WRONG_SLAVE	Invalid data values responded by slave (Invalid station number)	Check on the correctness of data format responded by slave.
DFB_RESPONSE_WRONG_FUNCTIONCODE	Invalid data values responded by slave (Invalid function code)	Check on the correctness of data format responded by slave.
DFB_REQUEST_FAILED_TO_SEND	Failed to send data.	Contact the vendor directly.
DFB_RESPONSE_INVALID_PROTOCOL	Invalid data values responded by slave (Non–standard Modbus format)	Check on the correctness of data format responded by slave.

Description	Cause of Error	Corrective Action
DFB_RESPONSE_INVALID_HEAD	Invalid data values responded by slave (Invalid data length)	Check on the correctness of data format responded by slave.
DFB_INVALID_CHANNEL_INDEX	Invalid index of the slave channel	Check if the index of slave channel is correct.
DFB_CHANNEL_SETTING_NOT_SUPPORT	The trigger mode of slave channel is not set to “Application”.	Make sure the trigger mode has been set to “Application”.
DFB_INVALID_SLAVE	Slave configuration error	Check the correctness of DFB_ModbusTCPRequest Slave configuration.
DFB_INVALID_BUFFER	Invalid memory address setting to send and receive data	Check if the settings of pWriteBuf and pReadBuf of DFB_ModbusTCPRequest are correct.
DFB_INVALID_LENGTH	Invalid data length setting	Check if the settings of *uiReadLen* and *uiWriteLen* of DFB_ModbusTCPRequest are correct.
DFB_INVALID_SLAVE_ADDRESS	Invalid slave station number	Check if the settings of *usiSlaveAddr* of DFB_ModbusTCPRequest are correct.
DFB_INVALID_FUNCTION_CODE	Function block does not support this function code.	Check function block settings.
DFB_NO_ETHERNET_CONFIG	Ethernet Adapter device does not exist.	Check if Ethernet Adapter device has been added to the device tree.
DFB_NO_MASTER_CONFIG	Delta_Modbus_TCP_Master device does not exist.	Check if Delta_Modbus_TCP_Master device has been added to the device tree.
DFB_MEMORY_NOT_ENOUGH	Not enough system memory	Check if the program size exceeds the limit
DFB_CONNECTION_TIMEOUT	TCP connection time–out	• Check if the setting of Modbus TCP Slave is correct. • Check on the correctness of the wiring
DFB_CONNECTION_FAILED	TCP connection failed.	Check if the setting of Modbus TCP Slave is correct.
DFB_UNDEFINED	Undefined or has not yet run.	Wait for the running of the FB instruction completed.

Chapter 9: Instructions for Reading and Writing a Memory Card

9.1. DFB_MemoryRead

DFB_MemoryRead reads a memory card.

FB/FC	Instruction	Graphic Expression	ST Language
FB	DFB_MemoryRead		DFB_MemoryRead(bExecute:= , FileInfo:= , bDone=> , bBusy=> , bError=> , ErrorID=>);

- **Inputs**

Name	Function	Data Type	Setting Value (Default value)
bExecute	Run the function block. (Triggered when TRUE)	BOOL	TRUE/FALSE (FALSE)
FileInfo	Parameter setting for reading a file.	DFB_READ_FILE_INFO	

- **DFB_READ_FILE_INFO**

Name	Function	Data Type	Setting Value (Default value)
sFilePath	The name of the file to read.	STRING	(‘’)
wDataMode	ASCII CODE / BINARY mode	DFB_DATA_MODE	DFB_DATA_MODE. ASCII_MODE DFB_DATA_MODE.BINARY_MODE (DFB_DATA_MODE.ASCII_MODE)
wAsciiShowMode	The display mode of data to be read. (Deximal/ Hexadecimal)	DFB_ASCII SHOW_MODE	DFB_ASCIISHOW_MODE.DECIMAL DFB_ASCIISHOW_MODE.HEX (DFB_ASCIISHOW_MODE.DECIMAL)
wAsciiDec DataType	Data type of the variables to be read.	DFB_DEC_ DATATYPE	DFB_DEC_DATATYPE.BYTE_SIZE DFB_DEC_DATATYPE.WORD_SIZE DFB_DEC_DATATYPE.DWORD_SIZE DFB_DEC_DATATYPE.LWORD_SIZE DFB_DEC_DATATYPE.SINT_SIZE DFB_DEC_DATATYPE.USINT_SIZE DFB_DEC_DATATYPE.INT_SIZE DFB_DEC_DATATYPE.UINT_SIZE DFB_DEC_DATATYPE.DINT_SIZE

Name	Function	Data Type	Setting Value (Default value)
			DFB_DEC_DATATYPE.UDINT_SIZE DFB_DEC_DATATYPE.LINT_SIZE DFB_DEC_DATATYPE.ULINT_SIZE DFB_DEC_DATATYPE.REAL_SIZE DFB_DEC_DATATYPE.LREAL_SIZE (DFB_DEC_DATATYPE.BYTE_SIZE)
dwReadStartPos	The address of the start position to read the memory card's data.*	DWORD	(0)
dwElementLength	The length of the data in the controller's memory card.*	DWORD	1–25,000 (0)
pDestination	The address of the destination to store the controller's memory data.	POINTER TO BYTE	NULL

***Note**: The unit is defined as the data type of the *DFB_READ_FILE_INFO.wAsciiDecDataType*.

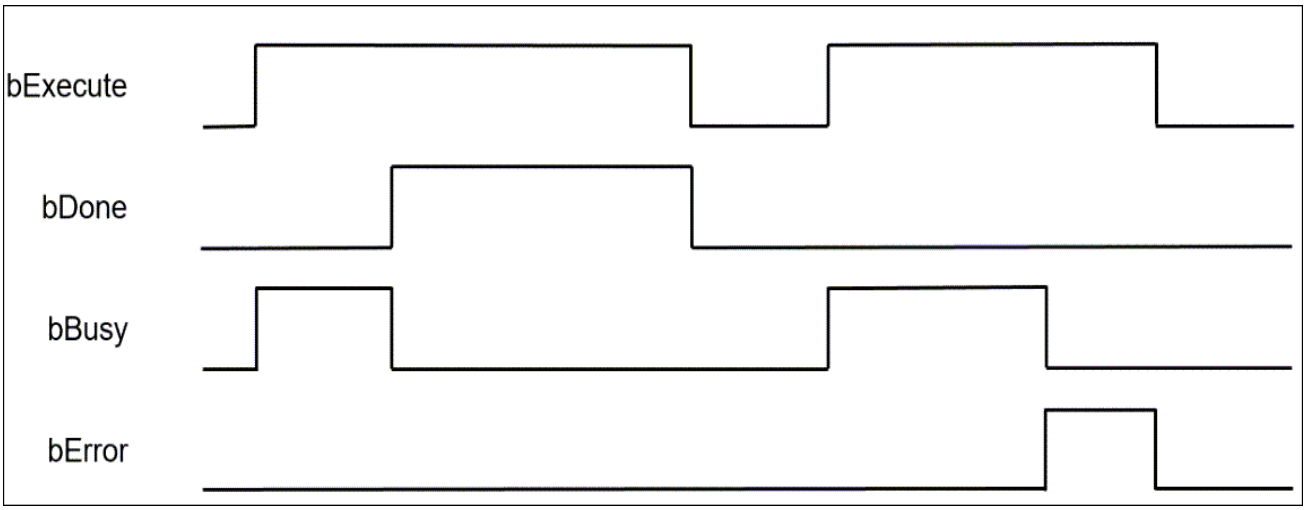
- **Outputs**

Name	Function	Data type	Output range (Default value)
bDone	The FB instruction running is completed.	BOOL	TRUE/FALSE (FALSE)
bBusy	The FB instruction is running.	BOOL	TRUE/FALSE (FALSE)
bError	FB instruction error flags.	BOOL	TRUE/FALSE (FALSE)
ErrorID	Error codes	DL_MEMRW_ERROR	DL_MEMRW_ERROR (DFB_NO_ERR)

- **Outputs Updating Timing**

Name	Timing for shifting to TRUE	Timing for shifting to FALSE
bDone	When FB instruction running is completed	When *bExecute* shifts to FALSE
bBusy	When FB instruction running starts	• When FB instruction running is completed • When *bExecute* shifts to FALSE
bError	When an error occurs during running or the input value of the instruction is incorrect	When *bExecute* shifts to FALSE
ErrorID		

- **Timing Diagram**



- **Function**

  Use the FB instruction (DFB_MemoryRead) to store the retrieved memory card data in the controller's memory.

- **Programming Example**

  This example uses the FB instruction (DFB_MemoryRead) to read the content of Test.csv file in the memory card and store the data in the controller's WORD–type array variable (ar_wVar0).

```
PROGRAM PLC_PRG
VAR
      bVar0: BOOL := TRUE;
      bExecute_Var,bDone_Var,bBusy_Var,bError_Var: BOOL;
      FB0: DFB_MemoryRead;
      FILE_INFO_Var: DFB_READ_FILE_INFO;
      ar_wVar0: ARRAY[0..3] OF WORD;
      ErrorID_Var: DL_MEMRW_ERROR;
END_VAR

IF bVar0 THEN
      FILE_INFO_Var.sFilePath:='Test.csv';
      FILE_INFO_Var.wDataMode:=DFB_DATA_MODE.ASCII_MODE;
      FILE_INFO_Var.wAsciiShowMode:=DFB_ASCIISHOW_MODE.HEX;
      FILE_INFO_Var.wAsciiDecDataType:=DFB_DEC_DATATYPE.WORD_SIZE;
      FILE_INFO_Var.dwReadStartPos:=0;
      FILE_INFO_Var.dwElementLength:=4;
      FILE_INFO_Var.pDestination:=ADR(ar_wVar0);
      bExecute_Var:=TRUE;
      bVar0:=FALSE;
END_IF;
IF bDone_Var THEN
bExecute_Var:=FALSE;
END_IF
FB0(
bExecute:=bExecute_Var ,
Filelnfo:=FILE_INFO_Var ,
bDone=>bDone_Var ,
bBusy=>bBusy_Var ,
bError=>bError_Var ,
ErrorID=>ErrorID Var );
```

  The content of Test.csv file in the memory card is shown as follows.

Values displayed in the Test.csv file					
0	1	2	3	4	5

Read the four consecutive data starting from data 0 in the Test.csv file via the FB instruction(DFB_MemoryRead), then store the retrieved data in the variable array(ar_wVar0), which the result will be ar_wVar0 := [0,1,2,3].

- **Supported Devices**
  - AX-3 series
- **Library**
  - DL_MemRW_AX3.library

9.2. DFB_MemoryWrite

DFB_MemoryWrite writes a memory card.

FB/FC	Instruction	Graphic Expression	ST Language
FB	DFB_MemoryWrite		DFB_MemoryWrite (bExecute:= , FileInfo:= , bDone=> , bBusy=> , bError=> , ErrorID=>);

- **Inputs**

Name	Function	Data Type	Setting Value (Default value)
bExecute	Run the function block. (Triggered by the rising-dege)	BOOL	TRUE/FALSE (FALSE)
FileInfo	Parameter setting for reading a file	DFB_WRITE_FILE_INFO	

- **DFB_WRITE_FILE_INFO**

Name	Function	Data Type	Setting Value (Default value)
sFilePath	The name of the file to create	STRING	(‘’)
wDataMode	ASCII CODE / BINARY mode	DFB_DATA_MODE	DFB_DATA_MODE.ASCII_MODE DFB_DATA_MODE.BINARY_MODE (DFB_DATA_MODE.ASCII_MODE)
wAsciiShowMode	The display mode of data to be written. (Deximal/ Hexadecimal)	DFB_ASCIISHOW_MODE	DFB_ASCIISHOW_MODE.DECIMAL DFB_ASCIISHOW_MODE.HEX (DFB_ASCIISHOW_MODE.DECIMAL)
wAsciiDecDataType	Data type of the variables to be written	DFB_DEC_DATATYPE	DFB_DEC_DATATYPE.BYTE_SIZE DFB_DEC_DATATYPE.WORD_SIZE DFB_DEC_DATATYPE.DWORD_SIZE DFB_DEC_DATATYPE.LWORD_SIZE DFB_DEC_DATATYPE.SINT_SIZE DFB_DEC_DATATYPE.USINT_SIZE DFB_DEC_DATATYPE.INT_SIZE DFB_DEC_DATATYPE.UINT_SIZE DFB_DEC_DATATYPE.DINT_SIZE DFB_DEC_DATATYPE.UDINT_SIZE

Name	Function	Data Type	Setting Value (Default value)
			DFB_DEC_DATATYPE.LINT_SIZE DFB_DEC_DATATYPE.ULINT_SIZE DFB_DEC_DATATYPE.REAL_SIZE DFB_DEC_DATATYPE.LREAL_SIZE (DFB_DEC_DATATYPE.BYTE_SIZE)
wAccessMode	The access mode of the file to be created	DFB_ACCESS_MODE	DFB_ACCESS_MODE.NEW DFB_ACCESS_MODE.APPEND DFB_ACCESS_MODE.OVERWRITE DFB_ACCESS_MODE.INSERT (DFB_ACCESS_MODE.NEW)
wCarriageReturn	CRLF character*	WORD	(0)
dwWriteStartPos	The address of the start position to write the memory card's data*	DWORD	(0)
dwElementLength	The length of the data to write to the controller's memory card*	DWORD	1–25,000 (0)
pSource	The memory address for the controller to store the data	POINTER TO BYTE	NULL

***Note**: The unit is defined as the data type of *DFB_WRITE_FILE_INFO.wAsciiDecDataType*.

- **Outputs**

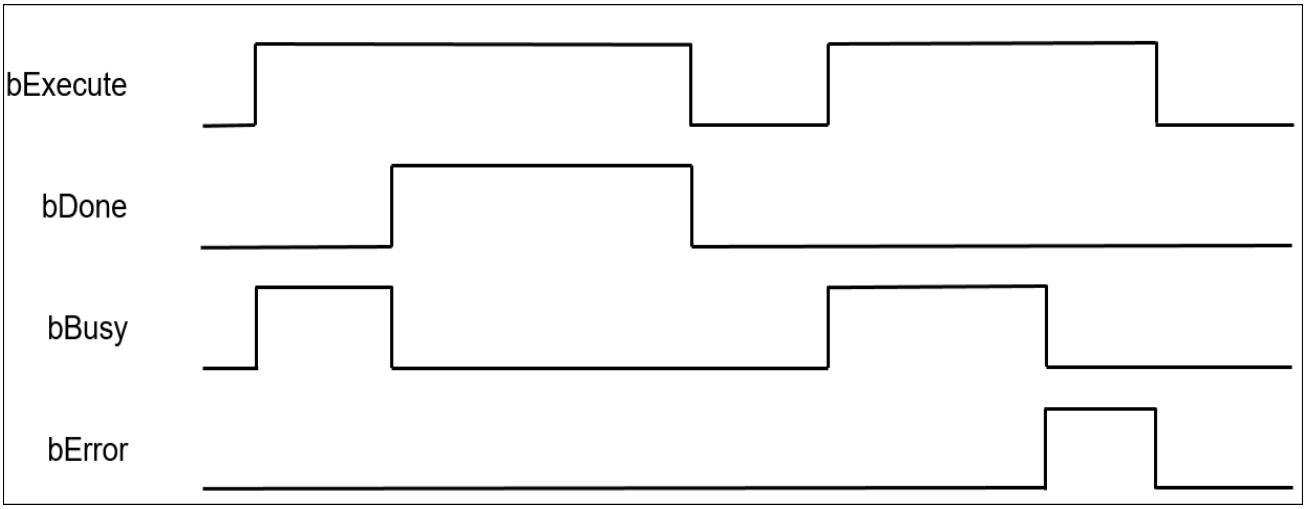
Name	Function	Data type	Output range (Default value)
bDone	The FB instruction running is completed.	BOOL	TRUE/FALSE (FALSE)
bBusy	The FB instruction is running.	BOOL	TRUE/FALSE (FALSE)
bError	FB instruction error flags	BOOL	TRUE/FALSE (FALSE)
ErrorID	Error codes	DL_MEMRW_ERROR	DL_MEMRW_ERROR (DFB_NO_ERR)

- **Output Updating Timing**

Name	Timing for shifting to TRUE	Timing for shifting to FALSE
bDone	When FB instruction running is completed	When *bExecute* shifts to FALSE

Name	Timing for shifting to TRUE	Timing for shifting to FALSE
bBusy	When FB instruction running starts	• When FB instruction running is completed • When *bExecute* shifts to FALSE
bError	When an error occurs during running or the input value of the instruction is incorrect	When *bExecute* shifts to FALSE
ErrorID		

- **Timing Diagram**



- **Function**

  Write the internal data of the controller to the memory card via the FB instruction (DFB_MemoryWrite).

- **Programming Example**

  This example uses the FB instruction (DFB_MemoryWrite) to write the WORD–type array variable to the memory card.

```
PROGRAM PLC_PRG
VAR
      bVar0: BOOL := TRUE;
      bExecute_Var,bDone_Var,bBusy_Var,bError_Var: BOOL;
      ar_wVar0: ARRAY[0..3] OF WORD:=[0,1,2,100];
      FB0: DFB_MemoryWrite;
      FILE_INFO_Var: DFB_WRITE_FILE_INFO;
      ErrorID_Var0: DL_MEMRW_ERROR;
END_VAR

IF bVar0 THEN
      FILE_INFO_Var.sFilePath:='Test.csv';
      FILE_INFO_Var.wDataMode:=DFB_DATA_MODE.ASCII_MODE;
      FILE_INFO_Var.wAsciiShowMode:=DFB_ASCIISHOW_MODE.DECIMAL;
      FILE_INFO_Var.wAsciiDecDataType:=DFB_DEC_DATATYPE.WORD_SIZE;
      FILE_INFO_Var.wAccessMode:=DFB_ACCESS_MODE.NEW;
      FILE_INFO_Var.wCarriageReturn:=0;
      FILE_INFO_Var.dwWriteStartPos:=0;
      FILE_INFO_Var.dwElementLength:=4;
      FILE_INFO_Var.pSource:=ADR(ar_wVar0);
      bExecute_Var:=TRUE;
      bVar0:=FALSE;
END_IF;
IF bDone_Var THEN
bExecute_Var:=FALSE;
END_IF
FB0(
```

```
bExecute:=bExecute_Var ,
Filelnfo:=FILE_INFO_Var ,
bDone=>bDone_Var ,
bBusy=>bBusy_Var ,
bError=>bError_Var ,
ErrorID=>ErrorID Var );
```

Suppose that the written data is ar_wVar0: ARRAY [0..3] OF WORD := [0,1,2,10]. After open the CSV file in the memory card, the content will be displayed as follows.

Values displayed in the Test.csv file			
0	1	2	10

***Note**:

1. In case of *wDataMode* = DFB_DATA_MODE.ASCII_MODE, the controller will write the content of array ar_wVar0 to the memory card in ASCII CODE format.
2. If *wAsciiDecDataType*: = DFB_DEC_DATATYPE.WORD_SIZE, the data length will be word in the CSV file.

- **Supported Devices**
  - AX-3 series
- **Library**
  - DL_MemRW_AX3.library

9.3. Error Codes and Troubleshooting

Description	Cause of Error	Corrective Action
DFB_NO_ERR	No errors.	–
DFB_MEMREAD_ERR_FAILED	Internal errors.	Check external SD card Contact us directly
DFB_MEMREAD_ERR_PARAMETER	Invalid parameter inputs.	Check if the input parameters are correct.
DFB_MEMREAD_ERR_NOTINITIALIZED	The instruction cannot be Run owning to the component has not been initialized.	Reboot the controller.
DFB_MEMREAD_ERR_VERSION	Wrong version.	Check if the firmware and the library version are supported.
DFB_MEMREAD_ERR_TIMEOUT	Operation time–out.	Reset the controller to default (Reset Origin). If the problem remains, Contact us directly.
DFB_MEMREAD_ERR_NOBUFFER	Insufficient memory.	Reset the controller to default (Reset Origin). Then download the project again after compressing the program. If the problem remains, Contact us directly.
DFB_MEMREAD_ERR_PENDING	The program is pending for running.	Reboot the controller.
DFB_MEMREAD_ERR_NUMPENDING	Too many pending programs.	Reset the controller to default (Reset Origin). If the problem remains, Contact us directly.
DFB_MEMREAD_ERR_NOTIMPLEMENTED	The function does not exist.	Check if the firmware and the library version are supported.
DFB_MEMREAD_ERR_INVALIDID	Incorrect ID.	Check if the firmware and the library version are supported.
DFB_MEMREAD_ERR_OVERFLOW	Integer overflow.	Check the data type of inputs and reset the controller to default (Reset Origin). If the problem remains, Contact us directly.
DFB_MEMREAD_ERR_BUFFERSIZE	The buffer size is too small.	Reset the controller to default (Reset Origin). Then download the project again after compressing the program. If the problem remains, Contact us directly.
DFB_MEMREAD_ERR_NO_OBJECT	The object does not exist.	1. Confirm if the controller and library versions support the object. 2. Confirm whether the file exists.

Description	Cause of Error	Corrective Action
		3. Confirm whether the value meets the specification. 4. Confirm whether the starting position is greater than the length of the data.
DFB_MEMREAD_ERR_NOMEMORY	Insufficient memory.	Reset the controller to default (Reset Origin). Then download the project again after compressing the program. If the problem remains, Contact us directly.
DFB_MEMREAD_ERR_DUPLICATE	Duplicate object name.	Check if the firmware and the library version are supported.
DFB_MEMREAD_ERR_MEMORY_OVERWRITE	Memory overwrite error.	Reset the controller to default (Reset Origin). Then download the project again after compressing the program. If the problem remains, Contact us directly.
DFB_MEMREAD_ERR_INVALID_HANDLE	Invalid handle for the object.	Check if the firmware and the library version are supported.
DFB_MEMREAD_ERR_END_OF_OBJECT	The end of the object has been reached.	Check if the firmware and the library version are supported.
DFB_MEMREAD_ERR_NO_CHANGE	No changes happened.	Check if the firmware and the library version are supported.
DFB_MEMREAD_ERR_INVALID_INTERFACE	Invalid or unknown interface	Check if the firmware and the library version are supported.
DFB_MEMREAD_ERR_NOT_SUPPORTED	The function is not supported.	Check if the firmware and the library version are supported.
DFB_MEMREAD_ERR_NO_ACCESS_RIGHTS	No rights to access the operation.	Check if the firmware and the library version are supported.
DFB_MEMREAD_ERR_OUT_OF_LIMITS	Exceeds the limited sources.	Reset the controller to default (Reset Origin). Then download the project again after compressing the program. If the problem remains, Contact us directly.
DFB_MEMREAD_ERR_ENTRIES_REMAINING	Remaining entries that could not be transmitted because of the buffer limitation.	Reset the controller to default (Reset Origin). Then download the project again after compressing the program. If the problem remains, Contact us directly.
DFB_MEMREAD_ERR_INVALID_SESSION_ID	Invalid online session ID.	Log in again or reboot the controller.
DFB_MEMREAD_ERR_EXCEPTION	Exception occurs.	Check the error log.
DFB_MEMWRITE_ERR_FAILED	Internal error	1. Check the external SD card. 2. Contact the original manufacturer.

Description	Cause of Error	Corrective Action
DFB_MEMWRITE_ERR_PARAMETER	Input parameter error	Confirm whether the input parameters are correct.
DFB_MEMWRITE_ERR_NOTINITIALIZED	The instruction cannot be Run because the element initialization has not been completed	Restart the controller.
DFB_MEMWRITE_ERR_VERSION	Wrong version	Check if the firmware and the library version are supported.
DFB_MEMWRITE_ERR_TIMEOUT	Execution time out	Restore the controller to factory settings (Reset Origin). If the problem persists, Contact us directly.
DFB_MEMWRITE_ERR_NOBUFFER	Insufficient memory	Reset the controller to default (Reset Origin). Then download the project again after compressing the program. If the problem remains, Contact us directly.
DFB_MEMWRITE_ERR_PENDING	Program to be Run	Restart the controller.
DFB_MEMWRITE_ERR_NUMPENDING	Too many programs in Run query	Restore the controller to factory settings (Reset Origin). If the problem persists, Contact us directly.
DFB_MEMWRITE_ERR_NOTIMPLEMENTED	No such function	Check if the firmware and the library version are supported.
DFB_MEMWRITE_ERR_INVALIDID	Incorrect ID	Check if the firmware and the library version are supported.
DFB_MEMWRITE_ERR_OVERFLOW	Value out of range	Restore the controller to factory settings (Reset Origin). If the problem persists, Contact us directly.
DFB_MEMWRITE_ERR_BUFFERSIZE	Memory size too small	Reset the controller to default (Reset Origin). Then download the project again after compressing the program. If the problem remains, Contact us directly.
DFB_MEMWRITE_ERR_NO_OBJECT	No such object	1. Check the external SD card. 2. Contact the original manufacturer.
DFB_MEMWRITE_ERR_NOMEMORY	Insufficient memory	Reset the controller to default (Reset Origin). Then download the project again after compressing the program. If the problem remains, Contact us directly.
DFB_MEMWRITE_ERR_DUPLICATE	Duplicate object name	Check if the firmware and the library version are supported.

Description	Cause of Error	Corrective Action
DFB_MEMWRITE_ERR_MEMORY_OVERWRITE	The memory exceeds the writable range	Reset the controller to default (Reset Origin). Then download the project again after compressing the program. If the problem remains, Contact us directly.
DFB_MEMWRITE_ERR_INVALID_HANDLE	Invalid object	Check if the firmware and the library version are supported.
DFB_MEMWRITE_ERR_END_OF_OBJECT	The maximum range of the object has been reached	Check if the firmware and the library version are supported.
DFB_MEMWRITE_ERR_NO_CHANGE	No change	Check if the firmware and the library version are supported.
DFB_MEMWRITE_ERR_INVALID_INTERFACE	Invalid or unknown interface	Check if the firmware and the library version are supported.
DFB_MEMWRITE_ERR_NOT_SUPPORTED	This feature is not supported	Check if the firmware and the library version are supported.
DFB_MEMWRITE_ERR_NO_ACCESS_RIGHTS	Not authorized to Run this command	Check if the firmware and the library version are supported.
DFB_MEMWRITE_ERR_OUT_OF_LIMITS	Exceeded limited resources	Reset the controller to default (Reset Origin). Then download the project again after compressing the program. If the problem remains, Contact us directly.
DFB_MEMWRITE_ERR_ENTRIES_REMAINING	Unable to transmit due to limited resources	Reset the controller to default (Reset Origin). Then download the project again after compressing the program. If the problem remains, Contact us directly.
DFB_MEMWRITE_ERR_INVALID_SESSION_ID	Invalid online ID	Restart the controller.
DFB_MEMWRITE_ERR_EXCEPTION	An exception error occurred	Confirm the error log.

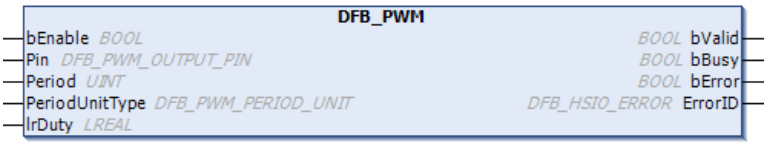
Chapter 10: High Speed Output Instructions

The instructions in this chapter apply to the built–in DO output of the host, and your can generate PWM signals through the instructions.

FB/FC	Description	Library
DFB_PWM	Generates the Pulse–width modulation output signals with adjustable frequency.	DL_BuiltInIO_AX3 / DL_BuiltInIO

10.1. DFB_PWM

DFB_PWM generates the Pulse–width modulation output signals with adjustable frequency.

FB/ FC	Instruction	Graphic Expression	ST Language
FB	DFB_PWM		DFB_PWM_instance (bEnable:=, Pin:=, Period:= PeriodUnitType:=, leDuty:=, bValid =>, bBusy =>, bError =>, ErrorID =>);

- **Inputs**

Name	Function	Data Type	Setting Value (Default value)	Effective time
bEnable	Run the instruction when *bEnable* changes from FALSE to TRUE.	BOOL	TRUE/FALSE (TRUE)	–
Pin	PWM output pins number	DFB_PWM_OUTPUT_PIN	0–7 (OUT00)	When *bEnable* is TRUE.
PeriodUnitType	PWM period unit	DFB_PWM_PERIOD_UNIT	MicroSecond or MilliSecond (MicroSecond)	When *bEnable* is TRUE.
Period	PWM period	UINT	MicroSecond :1–65535 MilliSecond :1–42949 (1)	When *bEnable* is TRUE.
lrDuty	PWM duty cycle	LREAL	0–100 (0)	Continuously effective when *bEnable* is TRUE.

Note: AX-332E only support scope: 0–3.

AX-332E only support scope: MicroSecond: 1000–2000, MilliSecond: 1–20.

- **Outputs**

Name	Function	Data Type	Output Value Range (Default value)
bValid	Function block output is valid.	BOOL	TRUE/FALSE (FALSE)

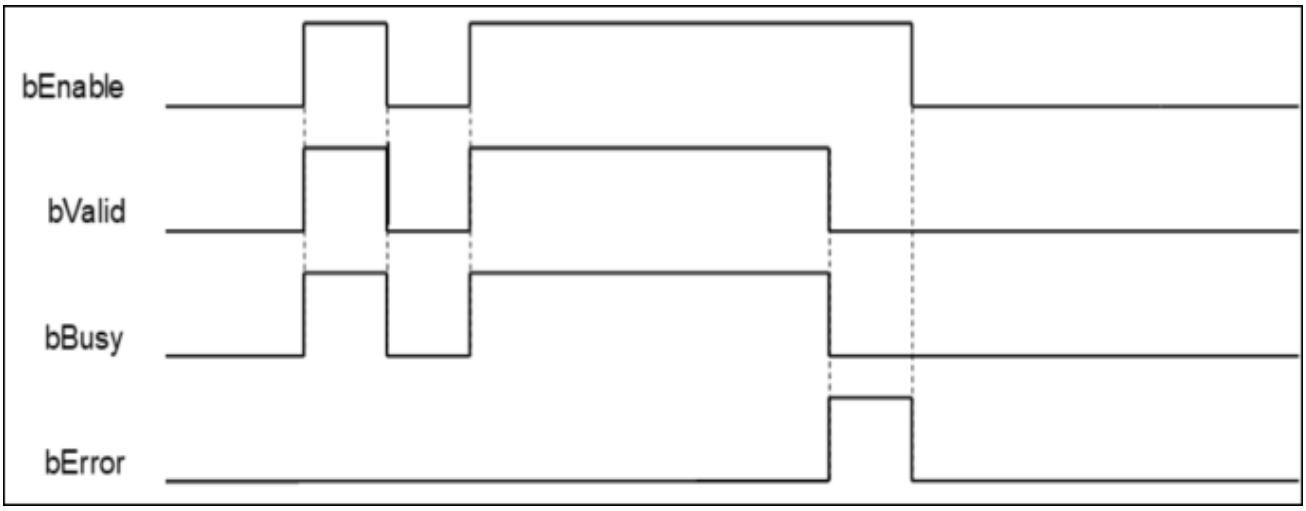
Name	Function	Data Type	Output Value Range (Default value)
bBusy	Function block is being busy.	BOOL	TRUE/FALSE (FALSE)
bError	Function block error	BOOL	TRUE/FALSE (FALSE)
ErrorID	Error ID	DFB_HSIO_ERRO*	DFB_HSIO_ERROR (DFB_HSIO_NO_ERR)

***Note**: DMC_ERROR：Enumeration (Enum)

- **Output Updating Timing**

Name	Timing for shifting to TRUE	Timing for shifting to FALSE
bValid	When *bExecute* shifts to TRUE.	• When *bError* shifts to TRUE. • When *bEnable* shifts to FALSE.
bBusy	When *bExecute* shifts to TRUE.	• When *bError* shifts to TRUE. • When *bEnable* shifts to FALSE.
bError	When instruction input parameter is illegal, or an error occurs during instruction running.	When *bExecute* shifts to FALSE.
ErrorID		

- **Timing Diagram**



- **Function**
  - Will generate a square wave signals of specified frequency and duty cycle which can be used to control plant that needs to receive continuously changing physical quantities.
  - This function only supports output contacts.
  - Firmware version needs to be V1.01.0 or later.

- **Troubleshooting**

  If an error occurs during the running of the instruction, *Error* will change to TRUE, and the axis motion will stop. Refer to *ErrorID* (Error Code) to address the problem.

- **Programming Example**

  - **Example 1**
    This programming example uses 8 sets of PWM output sine wave PWM, and the phase of each set is 45 degrees, which will make OUT LED appear water lamp behavior on the machine. The speed and direction are determined by the size and sign of "F_SIN_Hz" respectively.

```
PROGRAM DFB_PWM_1
VAR
      DFB_PWM : DL_BuiltInIO.DFB_PWM;
      bEnable: BOOL;
      F_SIN_Hz : LREAL := 0.3;
      T_SIN_ms : LREAL;
      tms : UDINT := 0;
      i : UINT;
      PWM : ARRAY[0..7] OF DL_BuiltInIO.DFB_PWM;
END_VAR
VAR CONSTANT
      M_PI : LREAL := 3.14159265358979323846;
END_VAR

FOR i := 0 TO 7 BY 1 DO
      PWM[i].bEnable := TRUE;
      PWM[i].Pin := i;
      PWM[i].PeriodUnitType := DFB_PWM_PERIOD_UNIT.MicroSecond; //usec
      PWM[i].Period := 100;
      PWM[i].lrDuty := 100.0*(0.5*SIN(2*M_PI*tms/T_SIN_ms + i*2*M_PI/8)+0.5);
      PWM[i]();
END_FOR

IF 0.0 = F_SIN_Hz THEN
      F_SIN_Hz := 1.0;
END_IF

T_SIN_ms := TO_DINT(1000/F_SIN_Hz);
      tms := tms + 1;
IF tms >= ABS(T_SIN_ms) THEN
      tms := 0;
END_IF
```

  - **Example 2**

    This progromming example uses the FB instruction (DFB_PWM) to output a signal for the OUT0 pin, and the On/OFF duration is 500ms respectively.

```
PROGRAM DFB_PWM
VAR
      DFB_PWM : DL_BuiltInIO.DFB_PWM;
      bEnable: BOOL;
END_VAR
DFB_PWM(
bEnable:= bEnable,
Pin:= DFB_PWM_OUTPUT_PIN.OUT00,
PeriodUnitType:= DFB_PWM_PERIOD_UNIT.MilliSecond,
Period:= 1000,
lrDuty:= 50,
bValid=> ,
bBusy=> ,
bError=> ,
ErrorID=> );
```

- **Supported Devices**

  - AX-3 series (except for AX-300)

- **Library**
  - DL_BuiltInIO.library

    Note: From version 1.0.5.0, library DL_BuiltInIO_AX3 is changed to DL_BuiltInIO.

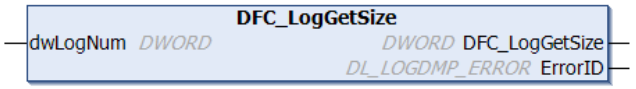
10.2. Error Codes and Troubleshooting

Description	Cause of Error	Corrective Action
DFB_PWM_UNSUPPORT_FW_VERSION	Firmware does not match.	Update firmware
DFB_PWM_PIN_ID_OVER_RANGE	Pin number out of range	Enter an appropriate pin ID
DFB_PWM_THIS_PIN_ID_IS_USED_ON_OTHER_PWM_FB	Reused pin	Do not enter the pin ID that has been used.
DFB_PWM_PERIOD_UNIT_IS_NO_DEFINITION	Wrong unit	Enter an appropriate period unit
DFB_PWM_PERIOD_OVER_RANGE	Period out of range	Enter an appropriate period value

Chapter 11: Additional Instructions

11.1. DFC_LogGetSize

DFC_LogGetSize reads the size of controller's log files.

FB/FC	Instruction	Graphic Expression	ST Language
FC	DFC_LogGetSize		DFC_LogGetSize(dwLogNum:= , ErrorID=>);

- **Inputs**

Name	Function	Data Type	Setting Value (Default value)
dwLogNum*	The number of the target log files.	DWORD	0: Calculate the current data size of all log files in the controller. (0)

***Note**: Data types such as BYTE, WORD and DWORD can be used for *dwLogNum* input. Currently only supports mode0.

- **Outputs**

Name	Function	Data type	Output range (Default value)
DFC_LOG_GETSIZE	Data size of the log files. (The parameter is returned.)	DWORD (Unit: BYTE)	0–65536 (0)
ErrorID	Error codes	DL_LOGDMP_ERROR	DL_LOGDMP_ERR (DFC_NO_ERROR)

- **Function**

  After running the function (DFC_LogGetSize), the data size of the current log files will be calculated.

- **Programming Example**

  This example uses the FC instruction (DFC_LogGetSize) to read the data size of the current log files of the controller.

```
PROGRAM PLC_PRG
VAR
    dwVar0: DWORD;
     ErrorID_Var: DL_LOGDMP_ERROR;
END_VAR
dwVAr0:=DFC_LogGetSize(dwLogNum:=0 , ErrorID=>ErrorID_Var );
```

- **Supported Devices**

  - AX-3 series, AX-864E30xE2T
    Note: AX-332E, AX-864E30xE2T (Library firmware version: 1.0.7.0 or later)

- **Library**

  - DL_LogDmp.library

11.2. DFB_LogDump

DFB_LogDump reads the log files of the controller.

FB/FC	Instruction	Graphic Expression	ST Language
FB	DFB_ LogDump		DFB_LogDump(bExecute:= , pDmpPos:= , dwLogNum:= , dwDmpLength:= , bDone=> , bBusy=> , bError=> , ErrorID=>);

- **Inputs**

Name	Function	Data Type	Setting Value (Default value)
bExecute	Run the function block. (Triggered by the rising–edge)	BOOL	TRUE/FALSE (FALSE)
pDmpPos	The memory address for controller's storage	POINTER TO BYTE	(0)
dwLogNum*[1]	The number of the target log files to read	DWORD	0: Read all the current log files of the controller (0)
dwDmpLength*[2]	The size of the target log files to read	DWORD	(0)

***Note**:

1. Data types such as BYTE, WORD, and DWORD can be used for *dwLogNum* input. Currently only supports mode0.
2. When the input length of *dwDmpLength* is greater than the length of the *pDmpPos* data type, an Exception error will occur.

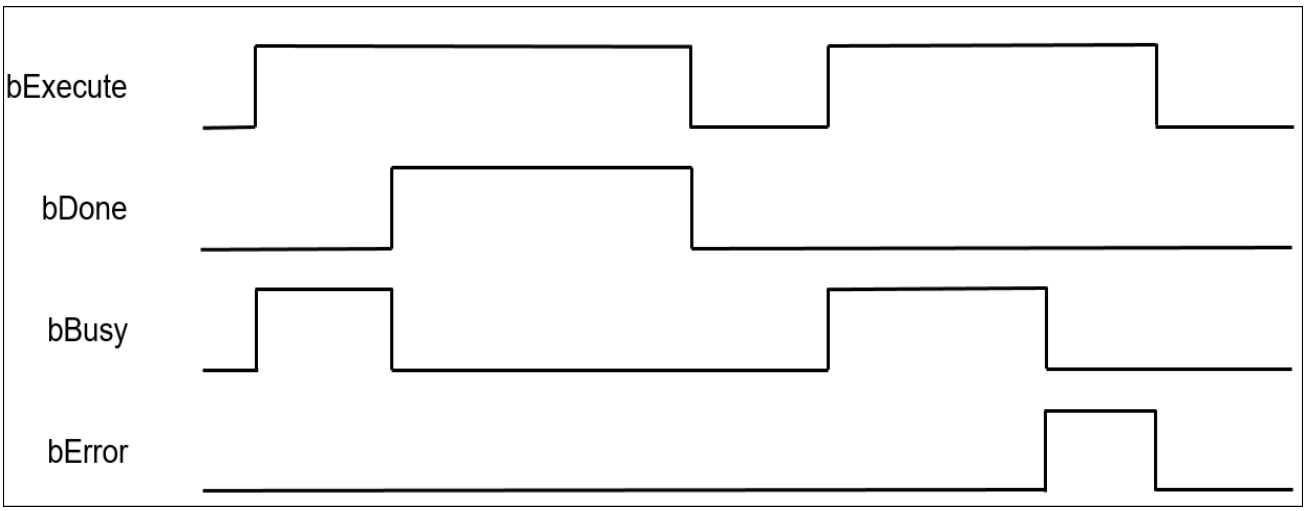
- **Outputs**

Name	Function	Data type	Output range (Default value)
bDone	The FB instruction running is completed.	BOOL	TRUE/FALSE (FALSE)
bBusy	The FB instruction is running.	BOOL	TRUE/FALSE (FALSE)
bError	FB instruction error flags	BOOL	TRUE/FALSE (FALSE)
ErrorID	Error codes	DL_LOGDMP_ERROR	DL_LOGDMP_ERROR (DFB_NO_ERR)

- **Output Updating Timing**

Name	Timing for shifting to TRUE	Timing for shifting to FALSE
bDone	When FB instruction running is completed	When *bExecute* shifts to FALSE
bBusy	When FB instruction running starts	• When FB instruction running is completed • When *bExecute* shifts to FALSE
bError	When an error occurs during running or the input value of the instruction is incorrect	When *bExecute* shifts to FALSE
ErrorID		

- **Timing Diagram**



- **Function**

  Use the FB instruction (DFB_LogDump) to read the log files of the controller.

- **Programming Example**

  This example uses the FB instruction (DFB_LogDump) to read the log files in the PLC and store the files in Byte array variable (ar_byVar) in the ASCII format.

```
PROGRAM PLC_PRG
VAR
        bExecute_Var,bDone_Var,bBusy_Var,bError_Var : BOOL;
        ar_byVar : ARRAY[0..999] OF BYTE;
        dwVar0: DWORD;
        FB0 : DFB_LogDump;
        ErrorID_Var :DL_LOGDMP_ERROR;
END_VAR
IF bDone_Var THEN
        bExecute_Var := FALSE;
END_IF
FB0(
        bExecute:= bExecute_Var,
        pDmpPos:= ADR(ar_byVar),
        dwLogNum:= 0,
        dwDmpLength:= SIZEOF(ar_byVar),
        bDone=> bDone_Var,
        bBu3y=> bBusy_Var,
        bError=> bError_Var,
        ErrorID=> ErrorlD Var);
```

- **Supported Devices**
  - AX-3 series, AX-864E30xE2T
    Note: AX-332E, AX-864E30xE2T (Library firmware version: 1.0.7.0 or later)
- **Library**
  - DL_LogDmp.library

11.3. Error Codes and Troubleshooting

- DL_LOGDMP_ERROR

Description	Cause of Error	Corrective Action
DFB_NO_ERR	No errors	–
DFB_DMP_ERR_FAILED	Internal errors	Contact us directly
DFB_DMP_ERR_PARAMETER	Invalid parameter inputs	Check if the input parameters are correct.
DFB_DMP_ERR_NOTINITIALIZED	The instruction cannot be Run owning to the component has not been initialized.	Reboot the controller.
DFB_DMP_ERR_VERSION	Wrong version	Check if the firmware and the library version are supported.
DFB_DMP_ERR_TIMEOUT	Operation time–out	Reset the controller to default (Reset Origin). If the problem remains, Contact us directly.
DFB_DMP_ERR_NOBUFFER	Insufficient memory	Reset the controller to default (Reset Origin). Then download the project again after compressing the program. If the problem remains, Contact us directly.
DFB_DMP_ERR_PENDING	The program is pending for running.	Reboot the controller.
DFB_DMP_ERR_NUMPENDING	Too many pending programs	Reset the controller to default (Reset Origin). If the problem remains, Contact us directly.
DFB_DMP_ERR_NOTIMPLEMENTED	The function does not exist.	Check if the firmware and the library version are supported.
DFB_DMP_ERR_INVALIDID	Incorrect ID	Check if the firmware and the library version are supported.
DFB_DMP_ERR_OVERFLOW	Integer overflow	Check the data type of inputs and reset the controller to default (Reset Origin). If the problem remains, Contact us directly.
DFB_DMP_ERR_BUFFERSIZE	The buffer size is too small.	Reset the controller to default (Reset Origin). Then download the project again after compressing the program. If the problem remains, Contact us directly.。
DFB_DMP_ERR_NO_OBJECT	The object does not exist.	Check if the firmware and the library version are supported.
DFB_DMP_ERR_NOMEMORY	Insufficient memory	Reset the controller to default (Reset Origin). Then download the project again after compressing the

Description	Cause of Error	Corrective Action
		program. If the problem remains, Contact us directly.。
DFB_DMP_ERR_DUPLICATE	Duplicate object name	Check if the firmware and the library version are supported.
DFB_DMP_ERR_MEMORY_OVERWRITE	Memory overwrite error.	Reset the controller to default (Reset Origin). Then download the project again after compressing the program. If the problem remains, Contact us directly.。
DFB_DMP_ERR_INVALID_HANDLE	Invalid handle for the object	Check if the firmware and the library version are supported.
DFB_DMP_ERR_END_OF_OBJECT	The end of the object has been reached.	Check if the firmware and the library version are supported.
DFB_DMP_ERR_NO_CHANGE	No changes happened.	Check if the firmware and the library version are supported.
DFB_DMP_ERR_INVALID_INTERFACE	Invalid or unknown interface	Check if the firmware and the library version are supported.
DFB_DMP_ERR_NOT_ SUPPORTED	The function is not supported.	Check if the firmware and the library version are supported.
DFB_DMP_ERR_NO_ACCESS_RIGHTS	No rights to access the operation	Check if the firmware and the library version are supported.
DFB_DMP_ERR_OUT_OF_LIMITS	Exceeds the limited sources.	Reset the controller to default (Reset Origin). Then download the project again after compressing the program. If the problem remains, Contact us directly.。
DFB_DMP_ERR_ENTRIES_REMAINING	Remaining entries that could not be transmitted because of the buffer limitation.	Reset the controller to default (Reset Origin). Then download the project again after compressing the program. If the problem remains, Contact us directly.。
DFB_DMP_ERR_INVALID_SESSION_ID	Invalid online session ID	Log in again or reboot the controller.
DFB_DMP_ERR_EXCEPTION	Exception occurs	Check the error log.
DFC_GETSIZE_ERR_FAILED	Internal errors	Contact us directly
DFC_ GETSIZE _ERR_PARAMETER	Invalid parameter inputs	Check if the input parameters are correct.
DFC_ GETSIZE _ERR_NOTINITIALIZED	The instruction cannot be Run owning to the component has not been initialized.	Reboot the controller
DFC_ GETSIZE _ERR_VERSION	Incorrect version	Check if the firmware and the library version are supported.

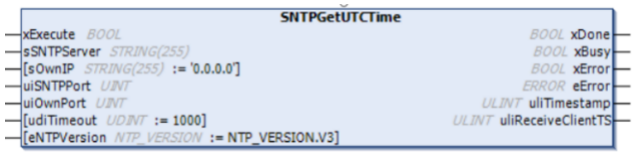
Description	Cause of Error	Corrective Action
DFC_ GETSIZE _ERR_TIMEOUT	Operation time–out	Reset the controller to default (Reset Origin). If the problem remains, Contact us directly.
DFC_ GETSIZE _ERR_NOBUFFER	Insufficient memory	Reset the controller to default (Reset Origin). Then download the project again after compressing the program. If the problem remains, Contact us directly.
DFC_ GETSIZE _ERR_PENDING	The program is pending for running.	Reboot the controller
DFC_ GETSIZE _ERR_NUMPENDING	Too many pending programs	Reset the controller to default (Reset Origin). If the problem remains, Contact us directly.
DFC_ GETSIZE _ERR_NOTIMPLEMENTED	The function does not exist.	Check if the firmware and the library version are supported.
DFC_ GETSIZE _ERR_INVALIDID	Incorrect ID	Check if the firmware and the library version are supported.
DFC_ GETSIZE _ERR_OVERFLOW	Integer overflow	Check the data type of inputs and reset the controller to default (Reset Origin). If the problem remains, Contact us directly.
DFC_ GETSIZE _ERR_BUFFERSIZE	The buffer size is too small.	Reset the controller to default (Reset Origin). Then download the project again after compressing the program. If the problem remains, Contact us directly.
DFC_ GETSIZE _ERR_NO_OBJECT	The object does not exist	Check if the firmware and the library version are supported.
DFC_ GETSIZE _ERR_NOMEMORY	Insufficient memory	Reset the controller to default (Reset Origin). Then download the project again after compressing the program. If the problem remains, Contact us directly.
DFC_ GETSIZE _ERR_DUPLICATE	Duplicate object name	Check if the firmware and the library version are supported.
DFC_GETSIZE_ERR_MEMORY_OVERWRITE	Memory overwrite error	Reset the controller to default (Reset Origin). Then download the project again after compressing the program. If the problem remains, Contact us directly.
DFC_ GETSIZE _ERR_INVALID_HANDLE	Invalid handle for the object	Check if the firmware and the library version are supported.
DFC_ GETSIZE _ERR_END_OF_OBJECT	The end of the object has been reached.	Check if the firmware and the library version are supported.
DFC_ GETSIZE _ERR_NO_	No changes happened.	Check if the firmware and the library version are supported.

Description	Cause of Error	Corrective Action
CHANGE		
DFC_ GETSIZE _ERR_INVALID_INTERFACE	Invalid or unknown interface	Check if the firmware and the library version are supported.
DFC_ GETSIZE _ERR_NOT_SUPPORTED	The function is not supported.	Check if the firmware and the library version are supported.
DFC_ GETSIZE _ERR_NO_ACCESS_RIGHTS	No rights to access the operation	Check if the firmware and the library version are supported.
DFC_ GETSIZE _ERR_OUT_OF_LIMITS	Exceeds the limited sources.	Reset the controller to default (Reset Origin). Then download the project again after compressing the program. If the problem remains, Contact us directly.
DFC_ GETSIZE_ERR_ENTRIES_REMAINING	Remaining entries that could not be transmitted because of the buffer limitation.	Reset the controller to default (Reset Origin). Then download the project again after compressing the program. If the problem remains, Contact us directly.
DFC_ GETSIZE _ERR_INVALID_SESSION_ID	Invalid online session ID	Log in again or reboot the controller.
DFC_ GETSIZE _ERR_EXCEPTION	Exception occurs	Check the error log.

Chapter 12: SNTP Instructions

12.1. SNTPGetUTCTime

SNTPGetUTCTime reads the current time of the NTP/SNTP server.

FB/FC	Instruction	Graphic Expression	ST Language
FB	SNTPGetUTCTime		SNTPGetUTCTime(xRun:= , sSNTPServer:= , sOwnIP:= , uiSNTPPort:= , uiOwnPort:= , udiTimeout:= , eNTPVersion:= , xDone=> , xBusy=> , xError=> , eError=> , uliTimestamp=> , uliReceiveClientTS=>);

- **Inputs**

Name	Function	Data Type	Setting Value (Default value)
xRun	Run the function block. (Triggered by the rising-dege)	BOOL	TRUE/FALSE (FALSE)
sSNTPServer	The IP address of the NTP/SNTP server	STRING (255)	(‘’)
sOwnIP	The IP address of the controller	STRING (255)	(0.0.0.0’)
uiSNTPPort	The UDP port number of the NTP/SNTP server	UINT	(0)
uiOwnPort	PLC UDP Port number	UINT	(0)
udiTimeout	The communication time–out value	UDINT (ms)	(1000)
eNTPVersion	The NTP version	NTP_VERSION	(V3)

- **NTP_VERSION**

Name	Description
V3	NTPv3
V4	NTPv4

- **Outputs**

Name	Function	Data Type	Setting Value (Default value)
xDone	The FB instruction running is completed.	BOOL	TRUE/FALSE (FALSE)
xBusy	The FB instruction is running.	BOOL	TRUE/FALSE (FALSE)
xError	FB instruction error	BOOL	TRUE/FALSE (FALSE)
eError	Error codes	ERROR	ERROR (NO_ERR)
uiTimestamp	The UTC time of the NTP/SNTP server	ULINT (ms)	–
uiReceiveClientTS	The UTC time of the controller	ULINT (ms)	–

- **Function**

  The FB instruction (SNTPGetUTCTime) is used for reading the UTC time of the NTP/SNTP server.

- **Programming Example**

  In this example, the FB instruction reads the the SNTP controller (192.168.1.95) UTC time sets this time to the controller (when *bVar* = TRUE) via the instruction (SystimeRTCSet).

  Note: You can use the FC instruction (SeparateDateTime) of Util.Library to convert the timestamp value of the NTP/SNTP server to IEC time format to confirm the time of its return.

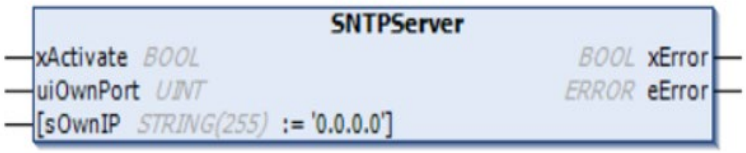
```
PROGRAM SNTP_Get_UTC_Time
VAR
SNTP_SNTPGetUTCTime : SNTP.SNTPGetUTCTime;
bExecute_Var, bDone_Var, bBusy_Var, bError_Var : BOOL;
eError_Var : SNTP.ERROR;
uliTimestamp_Var, uliReceiveClientTS_Var : ULINT;
bVar : BOOL;
eWeelcDay_Var : util. WEEK;
datDate_Var : DATE;
todTime_Var : TIME_OF_DAY;
END_VAR
SNTP_SNTPGetUTCTime(
      xExecute:= bExecute_Var,
      sSNTPServer:= '192.168.1.95',
      sOwnlP:= '192.168.1.5',
      uiSNTPPort:= 123,
      uiOwnPort:= 14567,
      udiTimeout:= 2000,
      eNTPVersion:= ,
      xDone=> bDone_Var,
      xBusy=> bBusy_Var,
      xError=> bError_Var,
      eError=> eError_Var,
      uliTimestamp=> uliTimestamp_Var,
      uliReceiveClientTS=> uliReceiveClientTS_Var);
IF bVar THEN
      SysTimeRtcSet(ulTimestamp:=ULINT_TO_DWORD(uliTime3tainp_Var/1000) ) ;
      bVar := FALSE;
END_IF
      SeparateDateTime(uliDateTime:= uliTimestainp_Var , eWeekDay=> eWeekDay_Var, datDate=>
 datDate_Var, todTime=> todTime_Var);
```

Note: When using this function block, you need to add the library name of the function block (such as SNTP.SNTPGetUTCTime).

- **Supported Devices**
  - AX-3 series (firmware v1.0.5.8 and later)
- **Library**
  - SNTP Service SL.library v1.1.0.0

12.2. SNTPServer

SNTPServer turns on the SNTP service.

FB/FC	Instruction	Graphic Expression	ST Language
FB	SNTPServer		SNTPServer(xActivate:= , uiOwnPort:= , sOwnIP:= , xError=> , eError=>);

- **Inputs**

Name	Function	Data Type	Setting Value (Default value)
xActive	Run the function block. (Triggered by the rising-dege)	BOOL	TRUE/FALSE (FALSE)
uiOwnPort	The UDP Port number of the controller	UINT	–
sOwnIP	The IP address of the controller	STRING (255)	(0.0.0.0’)

- **Outputs**

Name	Function	Data Type	Setting Value (Default value)
xError	FB instruction error	BOOL	TRUE/FALSE (FALSE)
eError	Error Code	ERROR	ERROR (NO_ERR)

- **Function**

  The FB instruction(NTPServer) is used to start the SNTP server of the controller.

- **Programming Example**

  In this example, the FB instruction (NTPServer) starts the SNTP server function of the controller.

```
PROGRAM SNTP_Server
VAR
      SNTP_Server : SNTP.SNTPServer;
      bActivate, bError_Var : BOOL;
      eError_Var : SNTP.ERROR;
END_VAR
SNTP_Server(
      xActivate:= bActivate,
      uiOwnPort:= 123,
      sOwnIP:= '192.168.1.5',
      xError=> bError_Var,
      eError=> eError Var);
```

Note: When using this function block, you need to add the library name of the function block (such as SNTP.SNTPServer).

- **Supported Devices**
  - AX-3 series (firmware v1.0.5.8 and later)
- **Library**
  - SNTP Service SL.library v1.1.0.0

12.3. Error Codes

Name	Description
NO_ERROR	No errors
TIME_OUT	NTP/SNTP server response time–out
INIT_ERROR	Initialization failed
SEND_ERROR	Failed to send
RECEIVE_ERROR	Failed to receive
CAN_NOT_GET_SYSTIME	The current controller UTC time cannot be read.
SERVER_REFUSED_REQUEST	NTP/SNTP sends KoD messages
INVALID_LICENSE	Invalid license

Chapter 13: Util Library Instructions

13.1. SeparateDateTime

SeparateDateTime converts the timestamp to IEC time format.

FB/FC	Instruction	Graphic Expression	ST Language
FC	SeparateDateTime		SeparateDateTime(uliDateTime:= , eWeekDay=> , datDate=> , todTime=>)

- **Inputs**

Name	Function	Data Type	Setting Value (Default value)
uiDateTime	The timestamp data that will be converted	ULINT (sec)	—

- **Outputs**

Name	Function	Data Type	Setting Value (Default value)
eWeekDay	The weekday for that date	WEEKDAY	–
datDate	Date (Year:Month:Date)	DATE	–
todTime	Time (Hour:Minute:Second)	TOD	–

- **WEEKDAY**

Name	Description
UNKNOWN	–
MONDAY	Monday
TUESDAY	Tuesday
WEDNESDAY	Wednesday
THURSDAY	Thursday
FRIDAY	Friday
SATURDAY	Saturday
SUNDAY	Sunday

- **Function**

  The FC instruction (SeparateDateTime) is used to convert the timestamp to the IEC time format.

- **Programming Example**

  In this example, the FC command (SeparateDateTime) converts 86,400,000 ms (1 day) to IEC time, and the converted date is "1970–1–2" and the time is "00:00:00".

```
PROGRAM Separate_DateTime
VAR
eWeekDay_Var : util.WEEK;
datDate_Var : DATE;
```

```
todTime_Var : TIME_OF_DAY;
END_VAR
```

SeparateDateTime(uliDateTime:= 86400000 , eWeekDay=>WeekDay_Var , datDate=>datDate_Var , todTime=>todTime_Var);

- **Supported Devices**
  - AX-3 series
- **Library**
  - Util.library

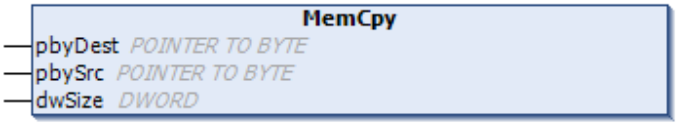
13.2. Error Codes

Name	Description
NO_ERROR	No errors
WRONG_CONFIGUTATION	NTP/SNTP server response time–out
RTC_HIGHRES_NOT_SUPPORTED	—
SYS_TIMEZONE_NOT_SUPPORTED	The timezone is not supported.

Chapter 14: MEMUtils Library Instructions

14.1. MemCpy

MemCpy copys the source address to the destination address.

FB/FC	Instruction	Graphic Expression	ST Language
FC	MemCpy		MemCpy(pbyDest:= , pbySrc:= , dwSize:=);

- **Inputs**

Name	Function	Data Type	Setting Value (Default value)
pbyDest	The destination address	POINTER TO BYTE	—
pbySrc	The source address	POINTER TO BYTE	—
dwSize	The length of the data to be copied	DWORD	1–4294967295 Positive integer (0)

- **Function**

  This FC instruction is used to copy the source address (*pSrc*) value to the specified destination address (pDest), and the length of the copied data is determined by the input parameter *dwSize* (in Byte).

- **Programming Example**

  In this example, MemCpy copies the value of Var_1(*pbySrc*) to the Var_2 (*pbyDest*) address, and the copy length is set to 80 in *dwSize*.

```
PROGRAM MemCpy
VAR
    M1 : BOOL;
    Var_1 : ARRAY[0..9] OF LREAL := [1,2,3,4,5,6,7,8,9,10];
    Var_2 : ARRAY[0..9] OF LREAL;
END_VAR
IF M1 THEN
    MEMUtils.MemCpy(pbyDest:= ADR(Var_2), pbySrc:= ADR(Var_1), dwSize:= 80);
END_IF
```

- **Supported Devices**
  - AX series
- **Libray**
  - MEMUtils.library

14.2. MemSet

MemSet sets the source address range.

FB/FC	Instruction	Graphic Expression	ST Language
FC	MemSet		MemSet(pbyBuffer:= , byValue:= , dwSize:=)

- **Inputs**

Name	Function	Data Type	Setting Value (Default value)
pbyBuffer	The specified address	POINTER TO BYTE	–
byValue	Value	POINTER TO BYTE	–
dwSize	The length of the data to be set	DWORD	1–4294967295 Positive integer (0)

- **Function**

  After running this instruction, *byValue* is written to the value set by the destination address (*pbyBuffer*), and the length of the address is determined by the input parameter *dwSize* (in bytes).

- **Programming Example**

  Use MemSet to write 0(*byValue*) to the value set by %MW1000 (*pbyBuffer*). If the length of the address is 20, the *pbyBuffer* range is %MW1000–%MW1009.

```
PROGRAM MemSet
VAR
    M1 : BOOL;
END_VAR
IF M1 THEN
     MEMUtils.MemSet(pbyBuffer:= ADR(%MW1000), byValue:= 0, dwSize:= 20);
END_IF
```

- **Supported Devices**
  - AX series
- **Libray**
  - MEMUtils.library

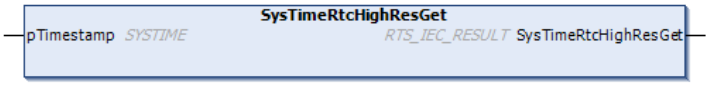
14.3. Error Codes

Name	Description
NO_ERROR	No errors

Chapter 15: SysTimeRtc Library Instructions

15.1. SysTimeRtcHighResGet

SysTimeRtcHighResGet gets the UTC time of the controller.

FB/FC	Instruction	Graphic Expression	ST Language
FC	SysTimeRtcHighResGet		SysTimeRtcHighResGet(pTimestamp:=);

- **Inputs**

Name	Function	Data Type	Setting Value (Default value)
pTimestamp	System time	SYSTIME	—
SysTimeRtcHighResGet	Error code	RTS_IEC_RESULT	—

- **Function**

  This function is supported by AX-3 series (except for AX-332) controllers with device description version 1.0.4.0 and later.

  After running this instruction, the UTC time in the controller is obtained, and then outputs to the *pTimestamp* parameter.

- **Programming Example**

  Use SysTimeRtcHighResGet to output the controller's time to the SysTime variable via *pTimestamp*.

```
PROGRAM SysTimeRtcHighResGet
VAR
    Systime : SysTimeRtc.SYSTIME;
    Errorvar : SysTimeRtc.RTS_IEC_RESULT;
END_VAR
Errorvar:=SysTimeRtc.SysTimeRtcHighResGet(pTimestamp:= SysTime );
```

- **Supported Devices**

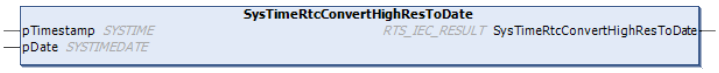
  - AX series

- **Libray**

  - SysTimeRtc.library

15.2. SysTimeRtcConvertHighResToDate

SysTimeRtcConvertHighResToDate converts the time got by SysTimeRtcHighResGet to date information.

FB/FC	Instruction	Graphic Expression	ST Language
FC	SysTimeRtcConvertHighResToDate		SysTimeRtcConvertHighResToDate(pTimestamp:= , pDate:=);

- **Inputs**

Name	Function	Data Type	Setting Value (Default value)
pTimestamp	PLC system time	SYSTIME	—
pDate	Date info	SYSTIMEDATE	—
SysTimeRtcHighResGet	Error code	RTS_IEC_RESULT	—

- **Function**

  When this instruction is running, the controller time obtained by SysTimeRtcHighResGet can be parsed into date info and output to the *pDate* parameter.

- **Programming Example**

  Use SysTimeRtcConvertHighResToDate to parse the controller time obtained by SysTimeRtcHighResGet into date info, and *pDate* output to *Timevar*.

```
PROGRAM SysTimeRtcHighResGet
VAR
      SysTime : SysTimeRtc.SYSTIME;
      Errorvar_1 : SysTimeRtc.RTS_IEC_RESULT;
      Timevar: SysTimeRtc.SYSTIMEDATE;
      Errorvar_2 : SysTimeRtc.RTS_IEC_RESULT;
END_VAR
Errorvar_1 := SysTimeRtc.SysTimeRtcHighResGet(pTimestamp:= SysTime );
Errorvar_2 := SysTimeRtcConvertHighResToDate(pTimestamp:= SysTime, pDate:=Timevar );
```

- **Supported Devices**
  - AX series
- **Libray**
  - SysTimeRtc.library

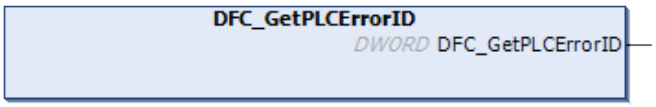
15.3. Error Codes

Name	Description
NO_ERROR	No errors

Chapter 16: DL_SysInfo Library Instructions

16.1. DFC_GetPLCErrorID

DFC_GetPLCErrorID gets the errors of the current controller

FB/FC	Instruction	Graphic Expression	ST Language
FC	DFC_GetPLCErrorID		DFC_GetPLCErrorID ( );

- **Inputs**

Name	Function	Data Type	Setting Value (Default value)
DFB_GetPLCErrorID	Read the current error ID of the PLC.	DWORD	0–65535

- **Function**

  This function reads the 16#140E–16#2203 errors of the controller, refer to *AX-3 Series Operation Manual* A.2 Troubleshooting of CPU Modules for details.

- **Programming Example**

  Use DFC_GetPLCErrorID to read the error ID of the system error.

```
PROGRAM Montion_PRG
VAR
    PLCErrorID : DWORD;
END_VAR
PLCErrorID := DL_SysInfo.DFC_GetPLCErrorID();
```

- **Supported Devices**

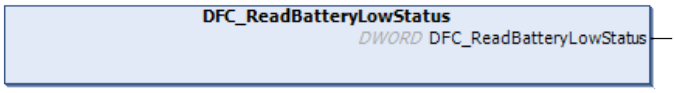
Controller	Firmware version	Library version
AX-3	1.0.7.0 and later	1.0.0.0
AX-332	1.0.6.3 and later	1.0.0.0
AX-C	1.0.3.8 and later	1.0.0.0
AX-8 Linux	1.0.5.11 and later	1.0.0.0
AX-864E30CE2T	1.0.1.12 and later	1.0.0.0

- **Libray**

  - DL_SysInfo.library

16.2. DFC_ReadBatteryLowStatus

DFC_ReadBatteryLowStatus gets the current battery status.

FB/FC	Instruction	Graphic Expression	ST Language
FC	DFC_ReadBatteryLowStatus		DFC_ReadBatteryLowStatus ( );

- **Inputs**

Name	Function	Data Type	Setting Value (Default value)
DFC_ ReadBatteryLowStatus	Read the current battery status of the PLC.	DWORD	0–1 (0)

- **Function**

  This function gets whether the system battery status is in low voltage.

  0 indicates that the battery is normal.

  1 indicates that the battery is abnormal.

- **Programming Example**

  Use DFC_ReadBatteryLowStatus to read whether the system battery status is normal.

```
PROGRAM Montion_PRG
VAR
    PLCErrorID : DWORD;
END_VAR
PLCErrorID := DL_SysInfo.DFC_GetPLCErrorID();
```

- **Supported Devices**

Controller	Firmware version	Library version
AX-3	1.0.7.0 and later	1.0.0.0
AX-332	1.0.6.3 and later	1.0.0.0
AX-C	1.0.3.8 and later	1.0.0.0
AX-8 Linux	1.0.5.11 and later	1.0.0.0
AX-864E30CE2T	1.0.1.12 and later	1.0.0.0

- **Libray**
  - DL_SysInfo.library

16.3. DFB_HWInfo

DFB_HWInfo gets the hardware information of the current controller

FB/FC	Instruction	Graphic Expression	ST Language
FB	DFB_HWInfo		DFB_HWInfo(bEnable:= , bValid=> , bBusy=> , dwErrorID=> , hwinfo=>

- **Inputs**

Name	Function	Data Type	Setting Value (Default value)
bEnable	Start the function block.	BOOL	TRUE/FALSE (FALSE)

- **Outputs**

Name	Function	Data Type	Setting Value (Default value)
bValid	TRUE when the parameter to be read is correct	BOOL	TRUE/FALSE (FALSE)
bBusy	TRUE when the instruction is running	BOOL	TRUE/FALSE (FALSE)
dwErrorID	When an error occurs, the error code is recorded.	DFB_HWINFO_ERROR	DFB_HWINFO_ERROR (DFB_NO_ERROR)
hwinfo	System hardware information	AX_HWINFO	AX_HWINFO*

Note: AX_HWINFO : Struct

Name	Function	Data Type	Unit	AX-3	AX-332 AX-8	AX–5	AX-C
strBootTime	Time of the system that has been running on the current power–up	STRING (31)	s	V	V	V	V
strCPUFrequency	CPU frequency	Array [0..7] OF STRING (11)	MHz	V	V	V	V
iCPUTemperature	CPU temperature	Array [0..7] OF INT	°C		V	V	V
byCPUUsage	CPU usage rate	Array [0..7] OF BYTE	%	V	V	V	V
bIsDHCP_GLAN	Launch DHCP_GLAN	Array [0..7] OF BOOL	–		V		V
wMemSize	CPU memory size	WORD	MB		V	V	V

Name	Function	Data Type	Unit	AX-3	AX-332 AX-8	AX–5	AX-C
byMemUsage	CPU memory usage	BYTE	%		V	V	V
strProductName	Product name	STRING (31)	–	V	V	V	V
strSerialNumber	Product serial number	STRING (31)	–	V	V	V	V
strSDPath	Micor SD path	Array [0..3] OF STRING (31)	–	V	V	V	V
strUDiskPath	USB path	Array [0..7] OF STRING (31)	–		V	V	

- **Output Updating Timing**

Name	Timing for shifting to TRUE	Timing for shifting to FALSE
bValid	When the parameter to be read is correct	When *bExecute* shifts to FALSE
bBusy	When the instruction is running	When *dwErrorID* is not 0
dwErrorID	When an error occurs during the running or there are incorrect input values	When *bEnable* shifts from TRUE to FALSE (Error code is cleared)
ErrorCode		

- **Function**

  This function gets information about the controller.

- **Programming Example**

  Use DFB_WInfo to get information about the controller and the controller information is output to the HWinfo structure variable.

```
PROGRAM DFB_HWInfo
VAR
    DFB_HWInfo: DL_SysInfo.DFB_HWInfo;
    bEnable: BOOL;
    HWinfo: DL_SysInfo.AX_HWINFO;
END_VAR

DFB_HWInfo(
bEnable:= bEnable,
bValid=> ,
bBusy=> ,
dwErrorID=> ,
hwinfo=> HWinfo);
```

- **Supported Devices**

Controller	Firmware version	Library version
AX-3	1.0.7.0 and later	1.0.0.0
AX-332	1.0.6.3 and later	1.0.0.0
AX-C	1.0.3.8 and later	1.0.0.0
AX-8 Linux	1.0.5.11 and later	1.0.0.0
AX-864E30CE2T	1.0.1.12 and later	1.0.0.0

- **Libray**
  - DL_SysInfo.library

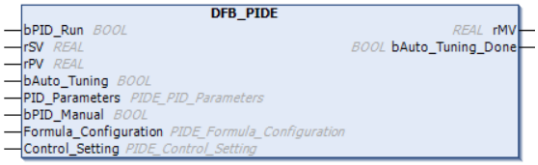
16.4. Error Codes

Name	Description
NO_ERROR	No errors

Chapter 17: DL_Controller_Loop Library Instructions

17.1. DFB_PIDE

DFB_PIDE performs PID calculation.

FB/FC	Instruction	Graphic Expression	ST Language
FB	DFB_PIDE		DFB_PIDE(bPID_Run:= , rSV:= , rPV:= , bAuto_Tuning:= , PID_Parameters:= , bPID_Manual:= , Formula_Configuration:= , Control_Setting:= , rMV=> , bAuto_Tuning_Done=>);

- **Inputs**

Name	Function	Data Type	Setting Value (Default value)
bPID_Run	Start PID calculation.	BOOL	TRUE/ FALSE (FALSE)
rSV	Target value	REAL	Single-precision floating-point value (0.0)
rPV	Current value	REAL	Single-precision floating-point value (0.0)
bAuto_Tuning (Triggered by the rising-edge)(*1) (*2)	PID Parameters self-tuning mode	BOOL	TRUE/FALSE (FALSE)
bPID_Manual	Manual output mode	BOOL	TRUE/FALSE (FALSE) TRUE: Use the value of *rManual_Out* as the output of *rMV*. FALSE: Use the result calculated by the PID formula as the output of *rMV*.
Formula_Configuration	Set a PID formula.	PIDE_Formula_Configuration	-
Control_Setting	PID control settings	PIDE_Control_Setting	-

***Note 1**: After the self-tuning of PID parameters is complete, the output parameter *bAuto_Tuning_Done* is set to TRUE, and the calculated results are output to *rKc_Kp*, *rTi_Ki*, *rTd_Kd*, and *rTf* parameters.

***Note 2**: When using the self-tuning mode, the output value of *rMV* will switch between the maximum output value and the minimum output value (*rMV_MAX* and *rMV_MIN*). If such drastic changes are not acceptable on the site, do not use the self-tuning mode to avoid potential harm to personnel or damage to the system.

- PIDE_ Formula_Configuration

Name	Function	Data Type	Setting Value (Default value)
bPID_EQ	Select a PID formula.	BOOL	TRUE/FALSE (FALSE) TRUE: Independent PID formula FALSE: Dependent PID formula
bPID_DE	Select the PID derivative to calculate.	BOOL	TRUE/FALSE (FALSE) TRUE: Variations in the current value of *rPV* FALSE: Variations in the error value (E)
bPID_DIR	PID feedback type	BOOL	TRUE/FALSE (FALSE) TRUE: Negative feedback control (*rSV–rPV*) FALSE: Positive feedback control (*rPV–rSV*)

The following table lists the TRUE value of the PID formula:

bPID_EQ =FALSE *bPID_DE* = FALSE *bPID_DIR* = FALSE	$rMV = K_pE + K_i \int_0^t Edt + K_d \frac{dE}{dt} + \text{BIAS},\ E = \text{rPV-}rSV$
bPID_EQ =FALSE *bPID_DE* = FALSE *bPID_DIR* = TRUE	$rMV = K_pE + K_i \int_0^t Edt + K_d \frac{dE}{dt} + \text{BIAS},\ E = \text{rSV-}rPV$
bPID_EQ =FALSE *bPID_DE* = TRUE *bPID_DIR* = FALSE	$rMV = K_pE + K_i \int_0^t Edt + K_d \frac{drPV}{dt} + \text{BIAS},\ E = \text{rPV-}rSV$
bPID_EQ =FALSE *bPID_DE* = TRUE *bPID_DIR* = TRUE	$rMV = K_pE + K_i \int_0^t Edt + K_d \frac{drPV}{dt} + \text{BIAS},\ E = \text{rSV-}rPV$
bPID_EQ =TRUE *bPID_DE* = FALSE *bPID_DIR* = FALSE	$rMV = K_c \left[E + \frac{1}{T_i} \int_0^t Edt + T_d \frac{dE}{dt} \right] + BIAS\ ,\ E = \text{rPV-}rSV$
bPID_EQ =TRUE *bPID_DE* = FALSE *bPID_DIR* = TRUE	$rMV = K_c \left[E + \frac{1}{T_i} \int_0^t Edt + T_d \frac{dE}{dt} \right] + BIAS\ ,\ E = \text{rSV-}rPV$

bPID_EQ =TRUE *bPID_DE* = TRUE *bPID_DIR* = FALSE	$rMV = K_c \left[E + \frac{1}{T_i}\int_0^t Edt + T_d \frac{drPV}{dt} \right] + BIAS$ ， E=rPV-*rSV*
bPID_EQ =TRUE *bPID_DE* = TRUE *bPID_DIR* = TRUE	$rMV = K_c \left[E + \frac{1}{T_i}\int_0^t Edt + T_d \frac{drPV}{dt} \right] + BIAS$ ， E=rSV-*rPV*

- PIDE_Control_Setting

Name	Function	Data Type	Setting Value (Default value)
udiPID_Cycle*(1)	Sampling time	UDINT	1–4,000 (0) Unit: ms
rManual_Out	Manual output value of *rMV*	REAL	Single-precision floating-point value (0.0)
rERR_DBW*(2)	Allowable error	REAL	Single-precision floating-point value (0.0)
rMV_MAX	Maximum value of *rMV* output	REAL	Single-precision floating-point value (0.0)
rMV_MIN	Minimum value of *rMV* output	REAL	Single-precision floating-point value (0.0)
rBias	Compensation value of *rMV* output	REAL	Single-precision floating-point value (0.0)

***Note 1**: The sampling time *udiPID_Cycle* is only used for the PID algorithm calculations and is not used for ruuning the PID operation. Therefore, enter the execution cycle of this function block as the sampling time, which is **Interval** of the task to which the POU belongs.

***Note 2**: When the tolerance value (*rErr_DBW*) is set to zero, the system will not monitor whether the deviation is within the tolerance range. If *rErr_DBW* is set to a non-zero value, the deviation (E) will be treated as zero when |E| ≤ |rErr_DBW| and the set value (*rSV*) equals the current value (*rPV*). For more information, see the following Additional remark.

- **Inputs/Outputs**

Name	Function	Data Type	Setting Value (Default value)
PID_Parameters	PID algorithm parameters	PIDE_ PID_Parameters	--

- **Outputs**

Name	Function	Data Type	Setting Value (Default value)
rMV	Output control	REAL	Single-precision floating-point value (0.0)
bAuto_Tuning_Done	Self-tuning is completed.	BOOL	TRUE/FALSE (FALSE)

- PIDE_PID_Parameters

Name	Function	Data Type	Setting Value (Default value)
rKc_Kp	Proportional gain	REAL	Single-precision floating-point value (0.0)
rTi_Ki	Integral gain	REAL	Single-precision floating-point value (0.0)
rTd_Kd	Derivative gain	REAL	Single-precision floating-point value (0.0)
rTf *	Derivative action time constant	REAL	Single-precision floating-point value (0.0)

***Note**: *Tf* (Derivative action time constant): is the time constant for the derivative operation of the PID-D controller on the error change, and its value is inversely proportional to the sensitivity of the PID-D controller to the rate of error change.

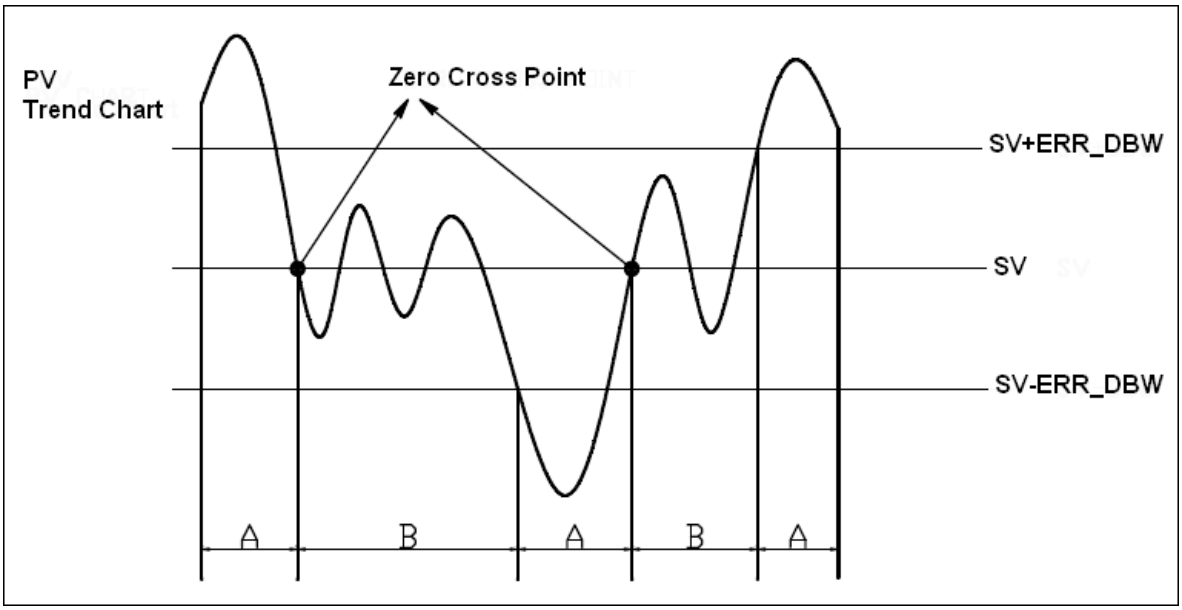
- If *Tf*=0, the derivative action time control is not used (Derivative smoothing). When *Kd* is relatively large, such as *Kd* > 10, it might be preferable not to use *Tf*.
- A large *Tf* value results in smoother output values from the PID-D controller, which helps reduce system sensitivity or oscillations.
- It is recommended to start adjusting the *Tf* value from a small value, such as 0.1 to 0.5, until it meets the requirements.
- If the *Tf* value is set too high, it might lead to sluggish system responses.

- Additional remark:
  - Task type of the DFB_PIDE function block: **Cycle**

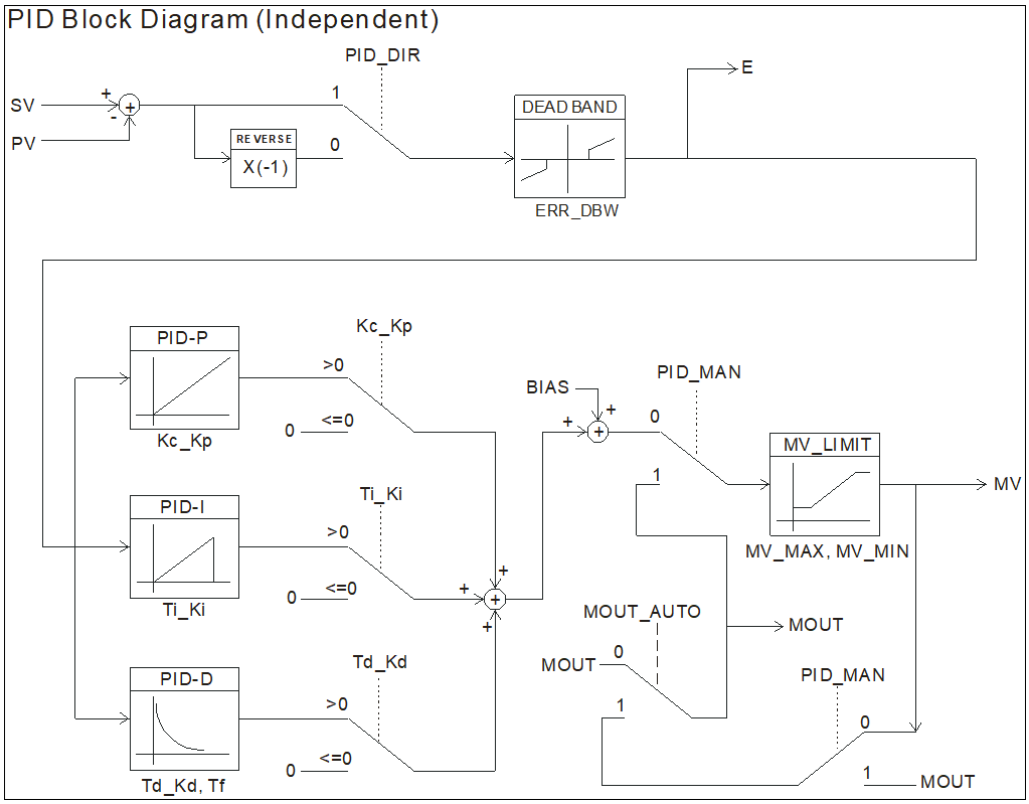
    When the function block is scanned, the PID algorithm is implmented and the output value is refreshed. Whether the scan time reaches the sampling time (*udiPID_Cycle*) is not calculated automatically.
  - PID paramenters adjustments:

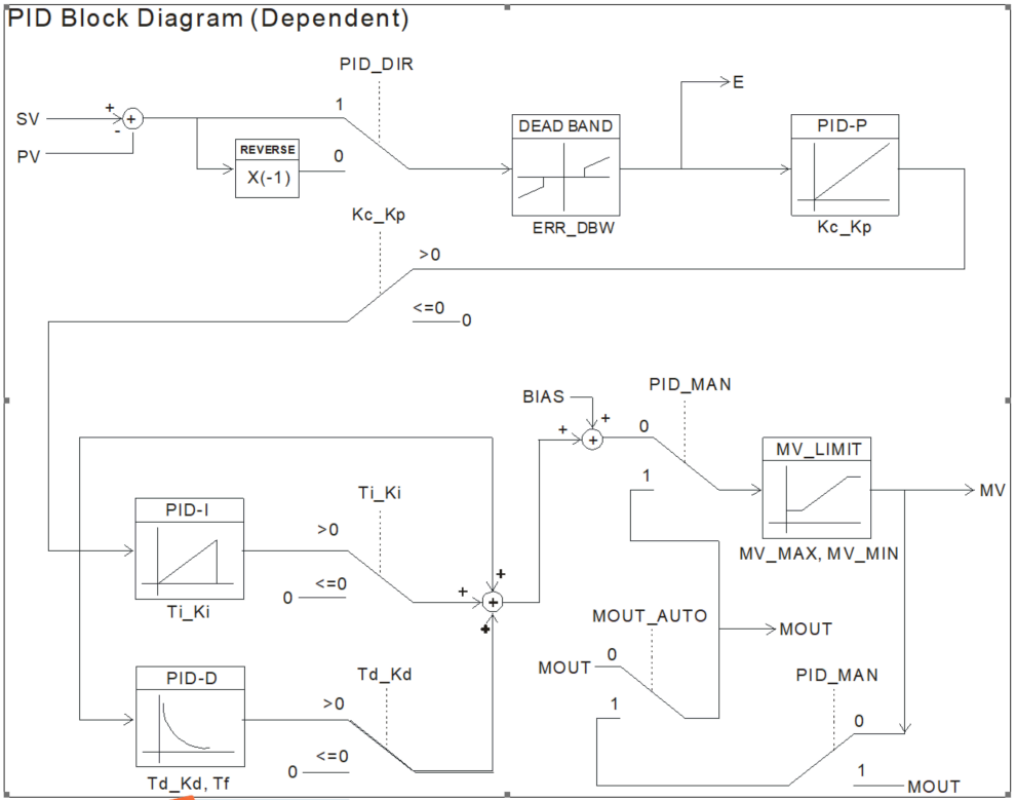
    When you tune the parameters *Kc_Kp*, *Ti_Ki*, and *Td_Kd*, tune *Kc_Kp* first according to experiences, and then set *Ti_Ki* and *Td_Kd* to 0. When you can handle the control, increase the value of *Ti_Ki* and *Td_Kd* (from small to large). When *Kc_Kp* is 1, it means that the proportional gain is 100%. That is, the error is increased by a factor of one. When the proportional gain is less than 100%, the error is decreased. When the proportional gain is greater than 100%, the error is increased .
  - *rERR_DBW*:

    If the *rPV* is in the range indicated by *rERR_DBW*, the CPU module will bring the error into the PID algorithm until the *rPV* reaches the *rSV*. The cross status condition will not be met until the *rPV* crosses the zero crossing point indicated by the *rSV*. If the cross status condition is met, the error will be count as 0 until the *rPV* is out of the range indicated by *rERR_DBW*. If *PID_DE* is set to true, the variations in the *rPV* will be used to calculate the control value of the derivative, and the CPU module will count the Δ *rPV* as 0 after the cross status condition is met (Δ *rPV*=Current *rPV*–Previous *rPV*). In the following *rPV* trend chart, the CPU module implements the PID algorithm normally in the A sections. In the B sections, the CPU module counts the error or the Δ *rPV* as 0 when it implements the PID algorithm.



- PID control diagrams



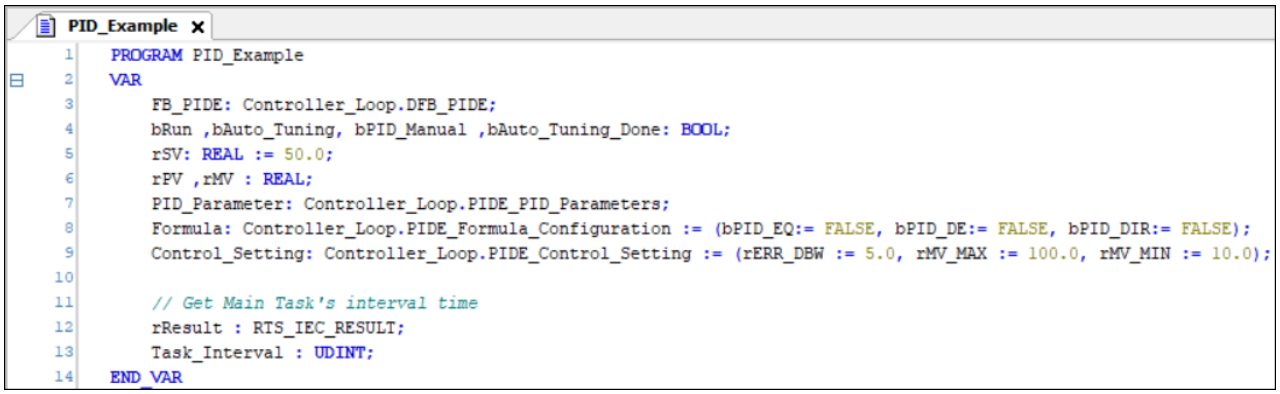


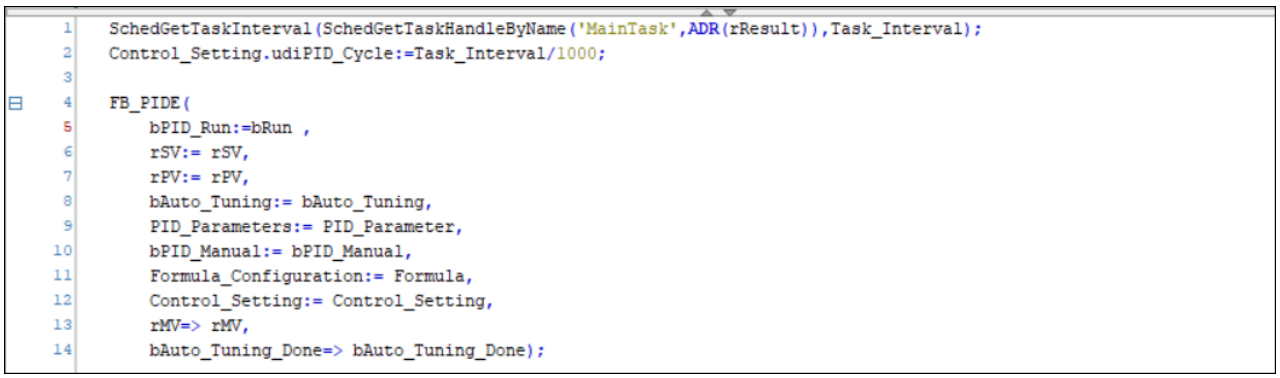
- **Function**

  Perform PID calculations using the FB instruction DFB_PIDE.

- **Programming Example**

  This example performs PID calculations using the FB instruction DFB_PIDE.






```
PROGRAM PID_Example
VAR
 FB_PIDE: Controller_Loop.DFB_PIDE;
 bRun ,bAuto_Tuning, bPID_Manual ,bAuto_Tuning_Done: BOOL;
 rSV: REAL := 50.0;
 rPV ,rMV : REAL;
 PID_Parameter: Controller_Loop.PIDE_PID_Parameters;
 Formula: Controller_Loop.PIDE_Formula_Configuration :=
       (bPID_EQ:= FALSE, bPID_DE:= FALSE, bPID_DIR:= FALSE);
 Control_Setting: Controller_Loop.PIDE_Control_Setting :=
       (rERR_DBW := 5.0, rMV_MAX := 100.0, rMV_MIN := 10.0);

 // Get Main Task's interval time
 rResult : RTS_IEC_RESULT;
 Task_Interval : UDINT;
END_VAR

SchedGetTaskInterval(SchedGetTaskHandleByName('MainTask',ADR(rResult)),Task_Interval);
Control_Setting.udiPID_Cycle:=Task_Interval/1000;

FB_PIDE(
 bPID_Run:=bRun ,
 rSV:= rSV,
 rPV:= rPV,
 bAuto_Tuning:= bAuto_Tuning,
 PID_Parameters:= PID_Parameter,
 bPID_Manual:= bPID_Manual,
 Formula_Configuration:= Formula,
 Control_Setting:= Control_Setting,
 rMV=> rMV,
 bAuto_Tuning_Done=> bAuto_Tuning_Done);
```

Note 1: The SchedGetTaskInterval instruction is used to read the task interval time (μs).

Note 2: To use the FC SchedGetTaskInterval, import the CmpSchedule library. To use the RTS_RESULT data type, import the SysTypes library.

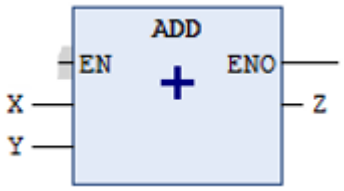
- **Supported Devices**
  - AX series
- **Library**
  - DL_Controller_Loop.library

Chapter 18: CODESYS Basic Instructions

18.1. Arithmetic Instructions

18.1.1. ADD (Addition Instruction)

ADD adds two or more variable values or constants.

FB/FC	Instruction	Graphic Expression	ST Language
FC	ADD		Z := X+Y;

- **Inputs**

Name	Function	Data Type	Setting Value
X, Y	Addend	UXINT, XINT, XWORD, BYTE, DATE, DATE_AND_TIME, DT, LDATE, LDATE_AND_TIME, LDT, LREAL, LTIME, LTOD, TIME_OF_DAY, TOD, WORD, DWORD, LWORD, USINT, UINT, UDINT, ULINT, SINT, INT, DINT, LINT, REAL, TIME	As each data type suggested.

- **Outputs**

Name	Function	Data Type	Setting Value
Z	Sum of the inputs	UXINT, XINT, XWORD, BYTE, DATE, DATE_AND_TIME, DT, LDATE, LDATE_AND_TIME, LDT, LREAL, LTIME, LTOD, TIME_OF_DAY, TOD, WORD, DWORD, LWORD, USINT, UINT, UDINT, ULINT, SINT, INT, DINT, LINT, REAL, TIME	As each data type suggested.

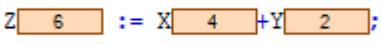
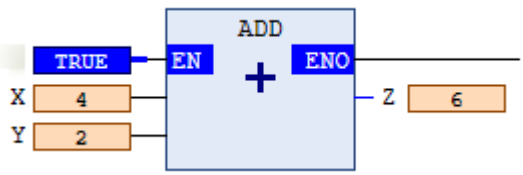
- **Function**

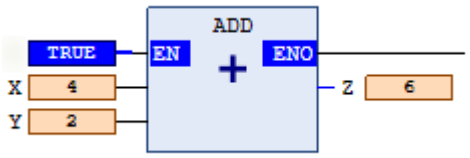
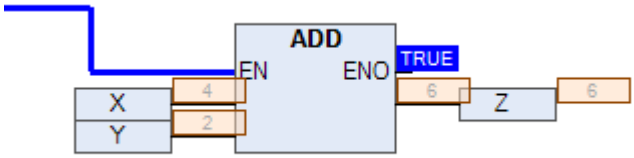
  This FC instruction is used to add the values of variable *X* and *Y*.

- **Programming Example**

  In this example, the FC instruction (ADD) is used to add the values of *X*(4) and *Y*(2) and get the final output as *Z*(6).

  Feature in the ST/LD/FBD & CFC editor:

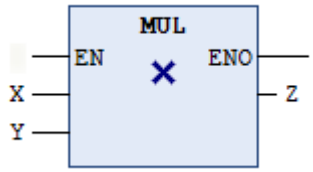
Programming Language	Program
ST	
LD	

Programming Language	Program
FBD	
CFC	

- **Supported Products**
  - AX series
- **Library**
  - None

18.1.2. MUL (Multiplication Instruction)

MUL multiplies two or more variable values or constants.

FB/FC	Instruction	Graphic Expression	ST Language
FC	MUL		Z :=X*Y;

- **Inputs**

Name	Function	Data Type	Setting Value
X, Y	Factor	BYTE, WORD, DWORD, LWORD, USINT, UINT, UDINT, ULINT, SINT, INT, DINT, LINT, TIME, REAL, LREAL	As each data type suggested.

- **Outputs**

Name	Function	Data Type	Setting Value
Z	Product of the inputs	BYTE, WORD, DWORD, LWORD, USINT, UINT, UDINT, ULINT, SINT, INT, DINT, LINT, TIME, REAL, LREAL	As each data type suggested.

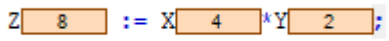
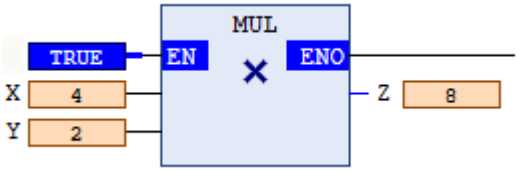
- **Function**

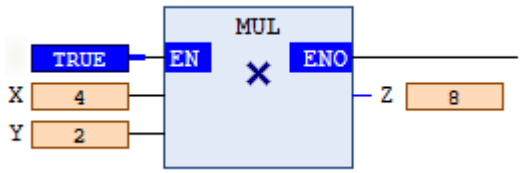
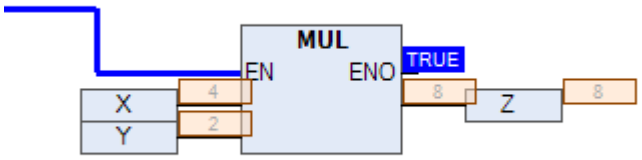
  This FC instruction is used to multiply the values of variable *X* and *Y*.

- **ProgrammingExample**

  In this example, the FC instruction (MUL) is used to multiply the values of *X* (4) and *Y* (2) and get the final output as *Z* (8).

  Feature in the ST/LD/FBD & CFC editor:

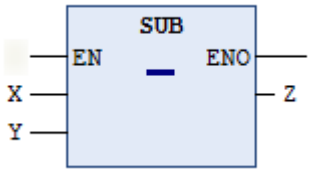
Programming Language	Program
ST	
LD	

Programming Language	Program
FBD	
CFC	

- **Supported Products**
  - AX series
- **Library**
  - None

18.1.3. SUB (Subtraction Instruction)

SUB subtracts one variable value or constant from another.

FB/FC	Instruction	Graphic Expression	ST Language
FC	SUB		Z :=X–Y;

- **Inputs**

Name	Function	Data Type	Setting Value
Y	Subtrahend	BYTE, WORD, DWORD, LWORD, USINT, UINT, UDINT, ULINT, SINT, INT, DINT, LINT, REAL, LREAL, TIME, LTIME, TIME_OF_DAY (TOD), LTIME_OF_DAY (LTOD), DATE, LDATE, DATE_AND_TIME (DT), LDATE_AND_TIME	As each data type suggested.
X	Minuend		

- **Outputs**

Name	Function	Data Type	Setting Value
Z	Difference of the inputs	BYTE, WORD, DWORD, LWORD, USINT, UINT, UDINT, ULINT, SINT, INT, DINT, LINT, REAL, LREAL, TIME, LTIME, TIME_OF_DAY (TOD), LTIME_OF_DAY (LTOD), DATE, LDATE, DATE_AND_TIME (DT), LDATE_AND_TIME	As each data type suggested.

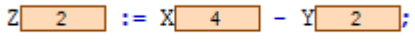
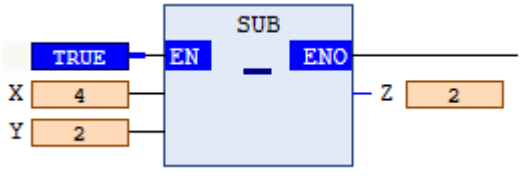
- **Function**

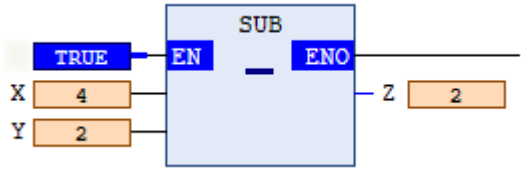
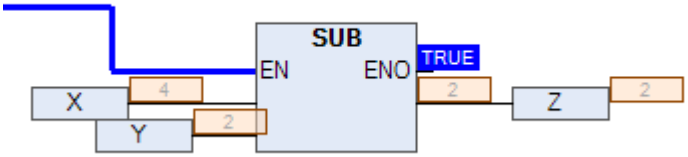
  This FC instruction is used to subtract the value of variable *Y* from variable *X*.

- **Programming Example**

  In this example, the FC instruction (SUB) is used to subtract the value of *Y* (2) from *X* (4) and get the final output as *Z* (2).

  Feature in the ST/LD/FBD & CFC editor:

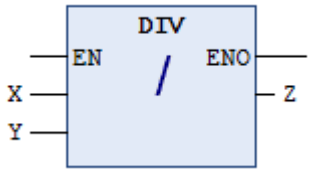
Programming Language	Program
ST	
LD	

Programming Language	Program
FBD	
CFC	

- **Supported Products**
  - AX series
- **Library**
  - None

18.1.4. DIV (Division Instruction)

DIV divides variable values or constants.

FB/FC	Instruction	Graphic Expression	ST Language
FC	DIV		Z :=X/Y;

- **Inputs**

Name	Function	Data Type	Setting Value
Y	Divisor	BYTE, WORD, DWORD, LWORD, USINT, UINT, UDINT, ULINT, SINT, INT, DINT, LINT, TIME, REAL, and LREAL	As each data type suggested.
X	Dividend		

- **Outputs**

Name	Function	Data Type	Setting Value
Z	Quotient of the inputs	BYTE, WORD, DWORD, LWORD, USINT, UINT, UDINT, ULINT, SINT, INT, DINT, LINT, TIME, REAL, and LREAL	As each data type suggested.

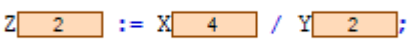
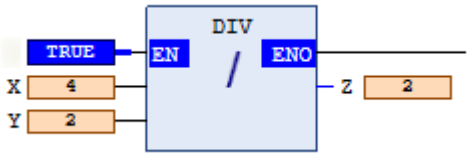
- **Function**

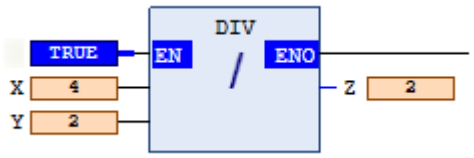
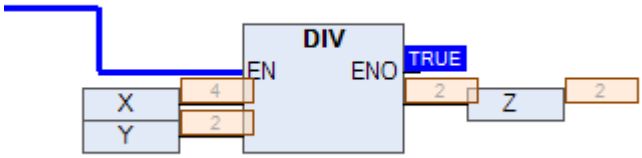
  This FC instruction is used to divide the values of variable X by variable Y.

- **Programming Example**

  In this example, the FC instruction (DIV) is used to divide the value of *X* (4) by *Y* (2) and get the final output as *Z* (2).

  Feature in the ST/LD/FBD & CFC editor:

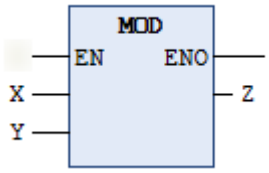
Programming Language	Program
ST	
LD	

Programming Language	Program
FBD	
CFC	

- **Supported Products**
  - AX series
- **Library**
  - None

18.1.5. MOD (Remainder Instruction)

MOD gets the remainder of the division of a variable value or constant, where the remainder is an integer.

FB/FC	Instruction	Graphic Expression	ST Language
FC	MOD		Z :=X MOD Y;

- **Inputs**

Name	Function	Data Type	Setting Value
Y	Divisor	BYTE, WORD, DWORD, LWORD, USINT, UINT, UDINT, ULINT, SINT, INT, DINT, LINT	As each data type suggested.
X	Dividend		

- **Outputs**

Name	Function	Data Type	Setting Value
Z	Remainder after modulo calculation	BYTE, WORD, DWORD, LWORD, USINT, UINT, UDINT, ULINT, SINT, INT, DINT, LINT	As each data type suggested.

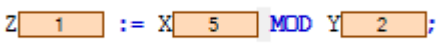
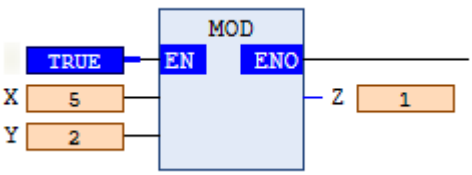
- **Function**

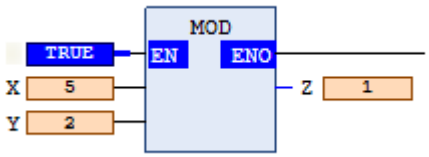
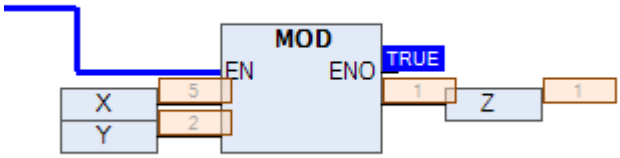
  This FC instruction is used to get remainder values by dividing variable *X* by variable *Y*.

- **Programming Example**

  In this example, the FC instruction (MOD) is used to get the remainder value after dividing *X* (5) by *Y* (2) and get the final output as *Z* (1).

  Feature in the ST/LD/FBD & CFC editor:

Programming Language	Program
ST	
LD	

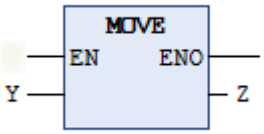
Programming Language	Program
FBD	
CFC	

- **Supported Products**
  - AX series
- **Library**
  - None

18.2. Assignment Instruction

18.2.1. MOVE (Assignment Instruction)

MOVE assigns the value of a constant or variable value to another variable.

FB/FC	Instruction	Graphic Expression	ST Language
FC	MOVE		Z :=Y;

- **Inputs**

Name	Function	Data Type	Setting Value
Y	Variable	BOOL, BYTE, WORD, DWORD, LWORD, USINT, UINT, UDINT, ULINT, SINT, INT, DINT, LINT, REAL, LREAL, TIME, DATE, TOD, DT, STRING	As each data type suggested.

- **Outputs**

Name	Function	Data Type	Setting Value
Z	Execution result	BOOL, BYTE, WORD, DWORD, LWORD, USINT, UINT, UDINT, ULINT, SINT, INT, DINT, LINT, REAL, LREAL, TIME, DATE, TOD, DT, STRING	As each data type suggested.

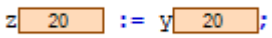
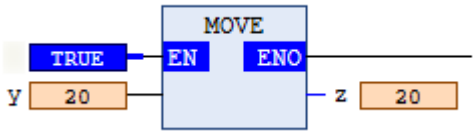
- **Function**

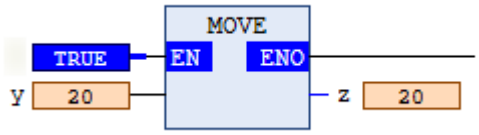
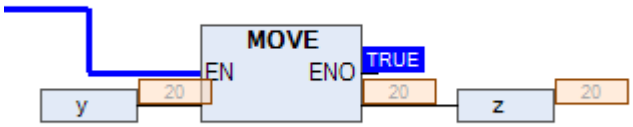
  This FC instruction is used to assign the input *Y* value to the output *Z*.

- **Programming Example**

  In this example, the FC instruction (MOVE) is used to move the value from *Y* (20) to *Z* (20).

  Feature in the ST/LD/FBD & CFC editor:

Programming Language	Program
ST	
LD	

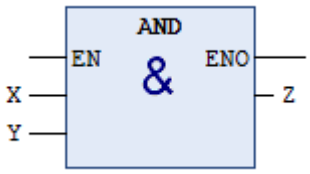
Programming Language	Program
FBD	
CFC	

- **Supported Products**
  - AX series
- **Library**
  - None

18.3. Logical Operation Instructions

18.3.1. AND (AND Instruction)

AND performs an AND operation between variables or constants.

FB/FC	Instruction	Graphic Expression	ST Language
FC	AND		Z :=X AND Y;

- **Inputs**

Name	Function	Data Type	Setting Value
X, Y	Variable	BOOL, BYTE, WORD, DWORD, LWORD	As each data type suggested.

- **Outputs**

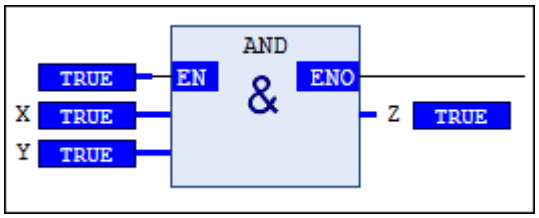
Name	Function	Data Type	Setting Value
Z	Result of the AND operation of the input values	BOOL, BYTE, WORD, DWORD, LWORD	As each data type suggested.

- **Function**

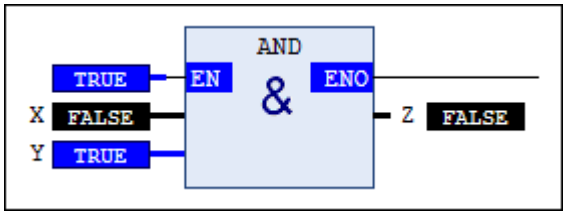
  This FC instruction is used to perform an AND operation between variable *X* and variable *Y*.

- **Programming Example**

  The FC instruction (AND) is used to get the output after using the operation AND. For example, if both *X* and *Y* are TRUE as per AND operation, then *Z* is TRUE as output.

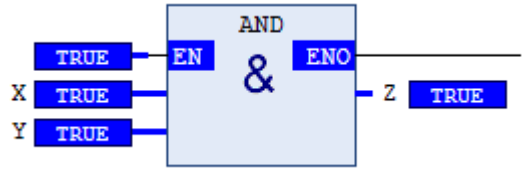
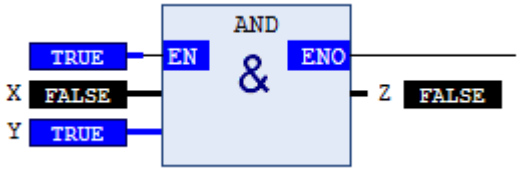
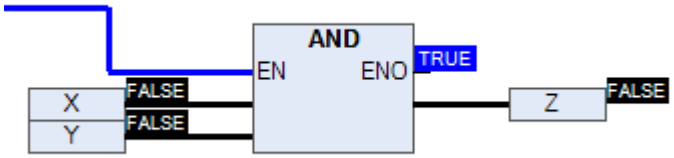


  If *X* is FALSE and *Y* is TRUE as per AND operation, then *Z* is FALSE as output.



  Feature in the ST/LD/FBD & CFC editor:

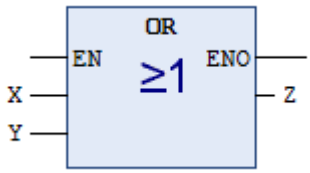
Programming Language	Program
ST	Z FALSE := X TRUE AND Y FALSE ;

Programming Language	Program
LD	
FBD	
CFC	

- **Supported Products**
  - AX series
- **Library**
  - None

18.3.2. OR (OR Instruction)

OR performs an OR operation between variables or constants.

FB/FC	Instruction	Graphic Expression	ST Language
FC	OR		Z :=X OR Y;

- **Inputs**

Name	Function	Data Type	Setting Value
X, Y	Variable	BOOL, BYTE, WORD, DWORD, LWORD	As each data type suggested.

- **Outputs**

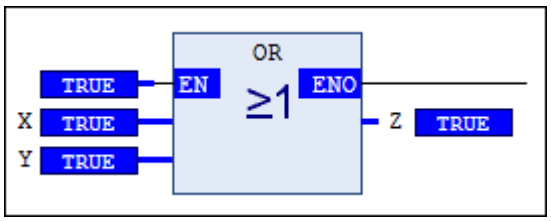
Name	Function	Data Type	Setting Value
Z	Result of the OR operation of the input values	BOOL, BYTE, WORD, DWORD, LWORD	As each data type suggested.

- **Function**

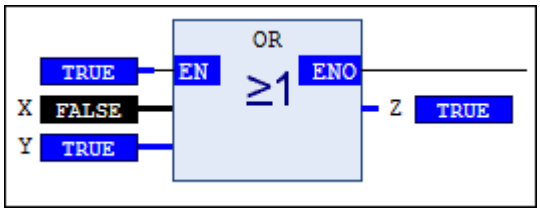
  This FC instruction is used to perform an OR operation between variable *X* and variable *Y*.

- **Programming Example**

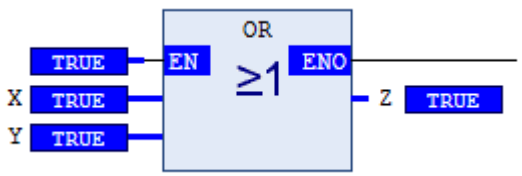
  The FC instruction (OR) is used to get the output after using the operation OR. For example, if both *X* and *Y* are TRUE as per OR operation, then *Z* is TRUE as output.

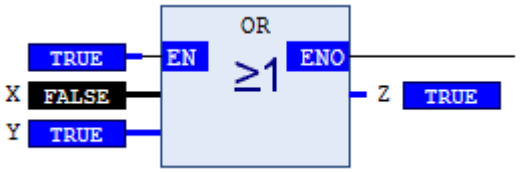
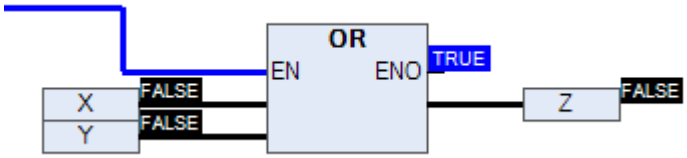


If *X* is FALSE and *Y* is TRUE as per OR operation, then *Z* is TRUE as output.



Feature in the ST/LD/FBD & CFC editor:

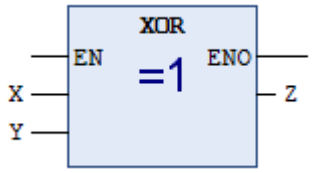
Programming Language	Program
ST	Z TRUE := X TRUE OR Y FALSE ;
LD	

Programming Language	Program
FBD	
CFC	

- **Supported Products**
  - AX series
- **Library**
  - None

18.3.3. XOR (XOR Instruction)

XOR performs an XOR operation between variables or constants.

FB/FC	Instruction	Graphic Expression	ST Language
FC	XOR		Z :=X XOR Y;

- **Inputs**

Name	Function	Data Type	Setting Value
X, Y	Variable	BOOL, BYTE, WORD, DWORD, LWORD	As each data type suggested.

- **Outputs**

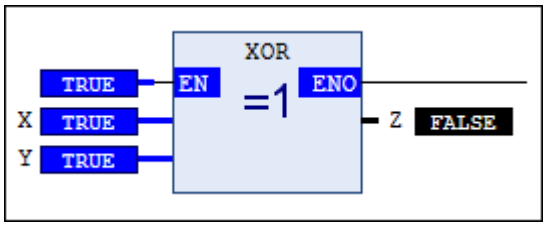
Name	Function	Data Type	Setting Value
Z	Result of the XOR operation of the input values	BOOL, BYTE, WORD, DWORD, LWORD	As each data type suggested.

- **Function**

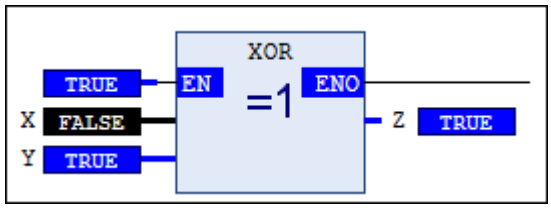
  This FC instruction is used to perform an XOR operation between variable *X* and variable *Y*.

- **Programming Example**

  The FC instruction (XOR) is used to get the output after using the operation XOR. For example, if both *X* and *Y* are TRUE as per XOR operation, then *Z* is FALSE as output.

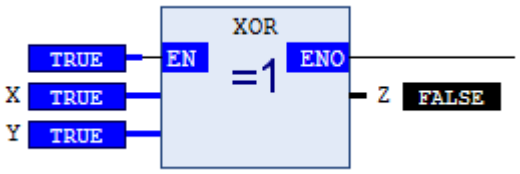
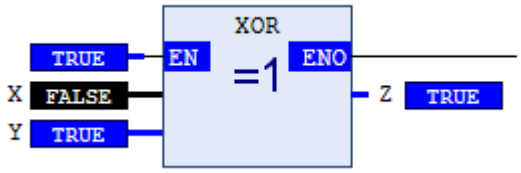
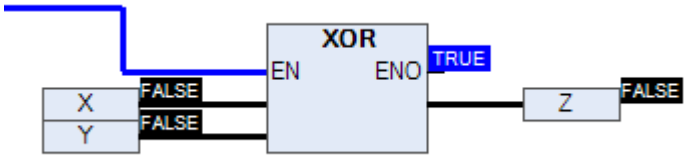


If *X* is FALSE and *Y* is TRUE as per XOR operation, then *Z* is TRUE as output.



Feature in the ST/LD/FBD & CFC editor:

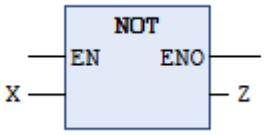
Programming Language	Program
ST	Z TRUE := X TRUE XOR Y FALSE;

Programming Language	Program
LD	
FBD	
CFC	

- **Supported Products**
  - AX series
- **Library**
  - None

18.3.4. NOT (NOT Instruction)

NOT performs a NOT operation between variables or constants.

FB/FC	Instruction	Graphic Expression	ST Language
FC	NOT		Z := NOT X;

- **Inputs**

Name	Function	Data Type	Setting Value
X	Variable	BOOL, BYTE, WORD, DWORD, LWORD	As each data type suggested.

- **Outputs**

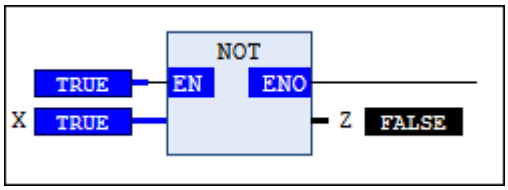
Name	Function	Data Type	Setting Value
Z	Result of the NOT operation of the input values	BOOL, BYTE, WORD, DWORD, LWORD	As each data type suggested.

- **Function**

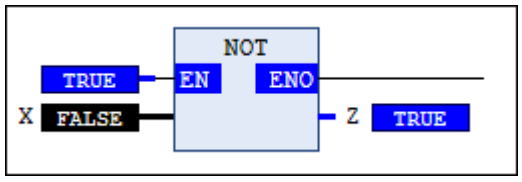
  This FC instruction is used to perform perform a NOT operation of the variable.

- **Programming Example**

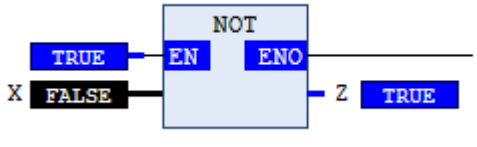
  The FC instruction (NOT) is used to get the output after using the operation NOT. For example, if *X* is TRUE as per NOT operation, then *Z* is FALSE as output.

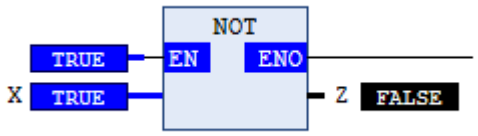
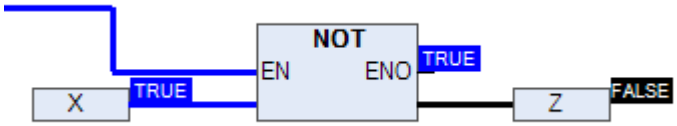


If *X* is FALSE as per NOT operation, then *Z* is TRUE as output.



Feature in the ST/LD/FBD & CFC editor:

Programming Language	Program
ST	Z TRUE := NOT X FALSE;
LD	

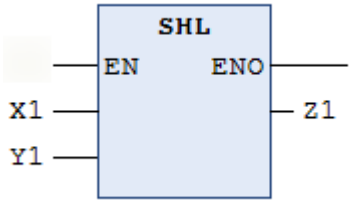
Programming Language	Program
FBD	
CFC	

- **Supported Products**
  - AX series
- **Library**
  - None

18.4. Shift Instructions

18.4.1. SHL (Left Shift Instruction)

SHL performs a bitwise shift of an operand to the left, and 0 is automatically added to the right.

FB/FC	Instruction	Graphic Expression	ST Language
FC	SHL		Z1 := SHL(X1,Y1);

- **Inputs**

Name	Function	Data Type	Setting Value
Y1	Number of bits to shift *X1* to the left	BYTE, WORD, DWORD, LWORD	As each data type suggested.
X1	Operand that is shifted to the left		

- **Outputs**

Name	Function	Data Type	Setting Value
Z1	The final output value after shifting the input to the left	BYTE, WORD, DWORD, LWORD	As each data type suggested.

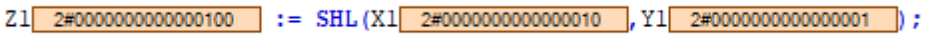
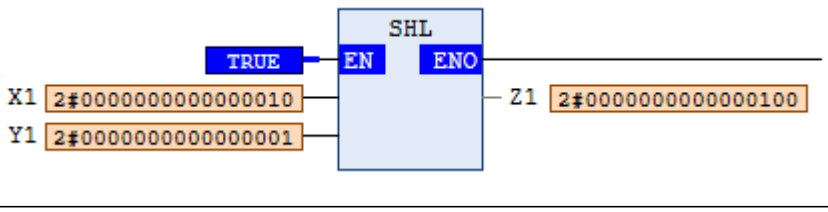
- **Function**

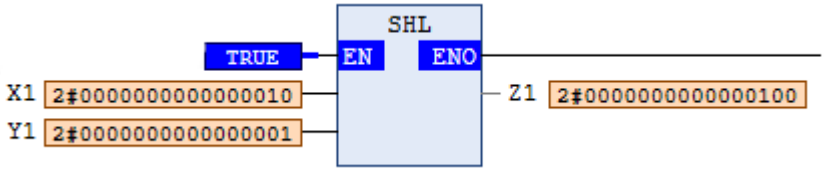
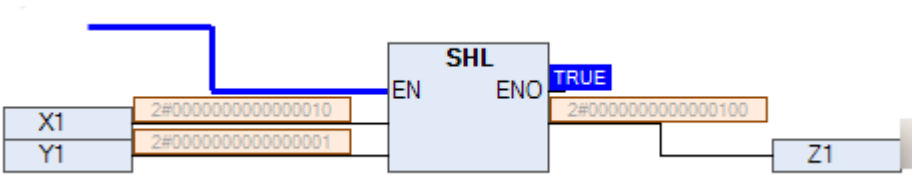
  This FC instruction is used for bitwise shift of the input to the left and add 0 to the right.

- **Programming Example**

  In this example, the FC instruction (SHL) is used for bitwise shift of the value of *X1* to the left by one bit and add one 0 to the right according to the value of *Y1*, so the final output value is as *Z1*.

  Feature in the ST/LD/FBD & CFC editor:

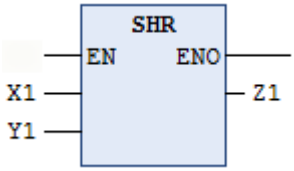
Programming Language	Program
ST	
LD	

Programming Language	Program
FBD	
CFC	

- **Supported Products**
  - AX series
- **Library**
  - None

18.4.2. SHR (Right Shift Instruction)

SHR performs a bitwise shift of an operand to the right, and 0 is automatically added to the left.

FB/FC	Instruction	Graphic Expression	ST Language
FC	SHR		Z1 := SHR(X1,Y1);

- **Inputs**

Name	Function	Data Type	Setting Value
Y1	Number of bits to shift *X1* to the right	BYTE, WORD, DWORD, LWORD	As each data type suggested.
X1	Operand that is shifted to the right		

- **Outputs**

Name	Function	Data Type	Setting Value
Z1	The final output value after shifting the input to the right	BYTE, WORD, DWORD, LWORD	As each data type suggested.

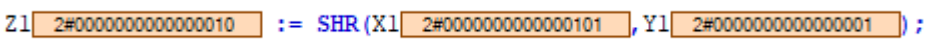
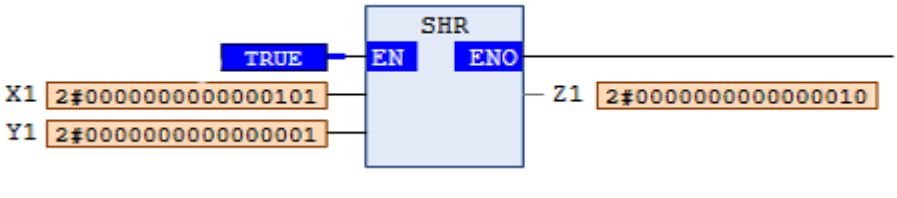
- **Function**

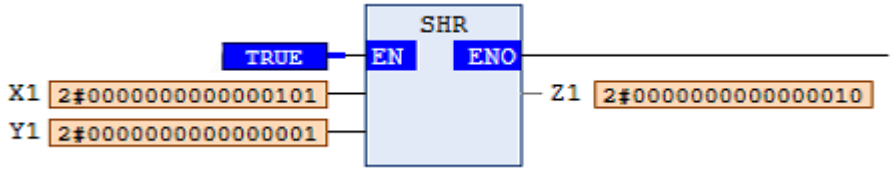
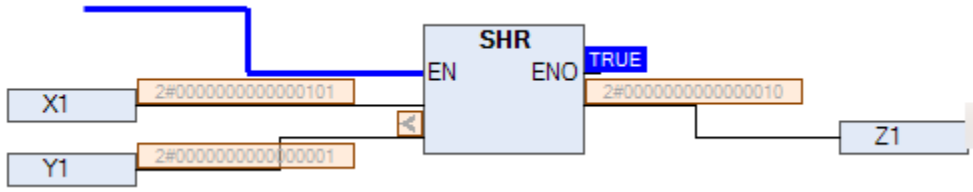
  This FC instruction is used for bitwise shift of the input to the right and add 0 to the left.

- **Programming Example**

  In this example, the FC instruction (SHR) is used for bitwise shift of the value of *X1* to the right by one bit and add one 0 to the left according to the value of *Y1*, so the final value is the output as *Z1*.

  Feature in the ST/LD/FBD & CFC editor:

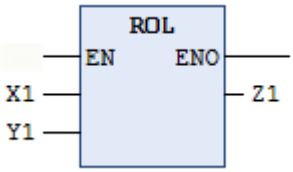
Programming Language	Program
ST	
LD	

Programming Language	Program
FBD	
CFC	

- **Supported Products**
  - AX series
- **Library**
  - None

18.4.3. ROL (Cycle Left Instruction)

ROL performs a bitwise rotation of an operand to the left, and the bits rotated from the left is directly added to the right–side position.

FB/FC	Instruction	Graphic Expression	ST Language
FC	ROL		Z1 := ROL(X1,Y1);

- **Inputs**

Name	Function	Data Type	Setting Value
Y1	Number of bits to rotate *X1* to the left	BYTE, WORD, DWORD, LWORD	As each data type suggested.
X1	Operand that is rotated to the left		

- **Outputs**

Name	Function	Data Type	Setting Value
Z1	The final output value after rotating to the left	BYTE, WORD, DWORD, LWORD	As each data type suggested.

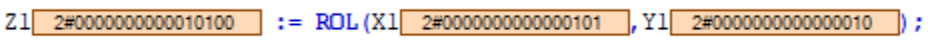
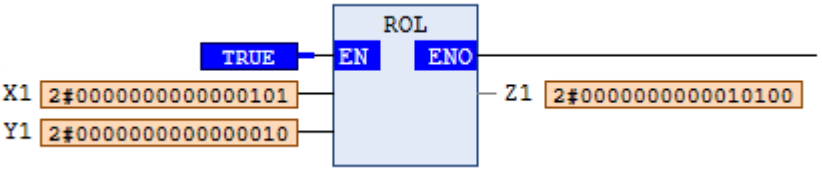
- **Function**

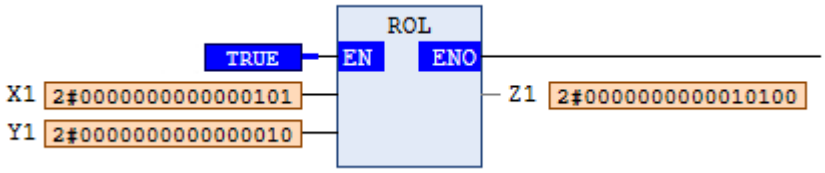
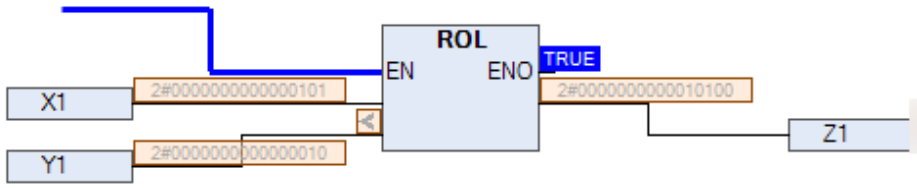
  This FC instruction is used to rotate the input value by bits to the left and add those rotated bits to the right–side.

- **Programming Example**

  In this example, the FC instruction (ROL) is used to rotate the value of *X1* to the left by two bits and add those rotated bits to the right side according to the value of *Y1*, so the final value is the output as *Z1*.

  Feature in the ST/LD/FBD & CFC editor:

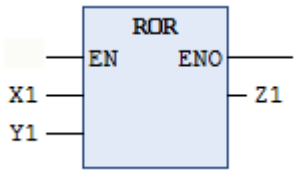
Programming Language	Program
ST	
LD	

Programming Language	Program
FBD	
CFC	

- **Supported Products**
  - AX series
- **Library**
  - None

18.4.4. ROR (Cycle Right Instruction)

ROR performs a bitwise rotation of an operand to the right, and the bits rotated from the right is directly added to the left–side position.

FB/FC	Instruction	Graphic Expression	ST Language
FC	ROR		Z1 := ROR(X1,Y1);

- **Inputs**

Name	Function	Data Type	Setting Value
Y1	Number of bits to rotate *X1* to the right	BYTE, WORD, DWORD, LWORD	As each data type suggested.
X1	Operand that is rotated to the right		

- **Outputs**

Name	Function	Data Type	Setting Value
Z1	The final output value after rotating to the right	BYTE, WORD, DWORD, LWORD	As each data type suggested.

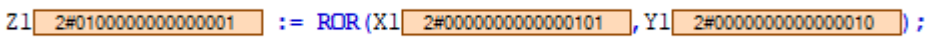
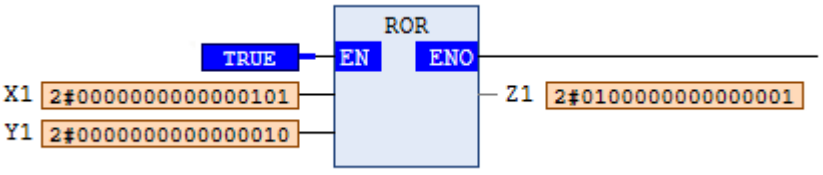
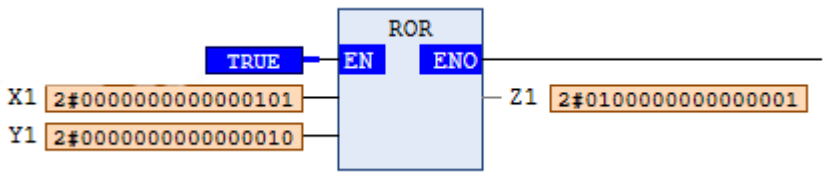
- **Function**

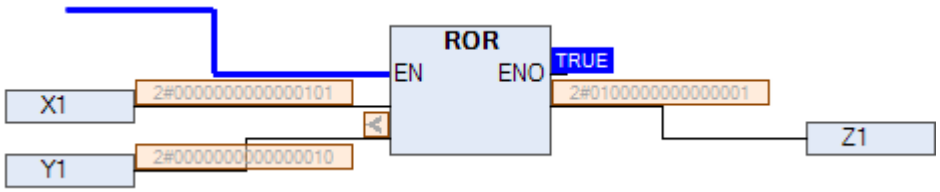
This FC instruction is used to rotate the input value by bits to the right and add those rotated bits to the left–side.

- **Programming Example**

In this example, the FC instruction (ROR) is used to rotate the value of *X1* to the right by two bits and add those rotated bits to the left side according to the value of *Y1*, so the final value is the output as *Z1*.

Feature in the ST/LD/FBD & CFC editor:

Programming Language	Program
ST	
LD	
FBD	

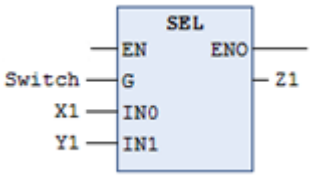
Programming Language	Program
CFC	

- **Supported Products**
  - AX series
- **Library**
  - None

18.5. Selection Instructions

18.5.1. SEL (Select One Instruction)

SEL uses the selector switch to select one of the two inputs as an output. *G* decides whether to assign *IN0* or *IN1* to *Z1*.

FB/FC	Instruction	Graphic Expression	ST Language
FC	SEL		Z1 := SEL (Switch,X1,Y1);

- **Inputs**

Name	Function	Data Type	Setting Value
IN0	Input	Any data type	As each data type suggested.
IN1	Selector switch	BOOL	TRUE, FALSE
G			

- **Outputs**

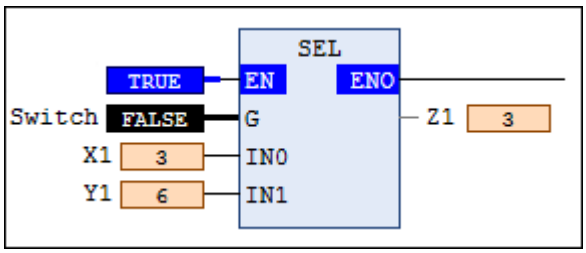
Name	Function	Data Type	Setting Value
Z1	The final output based on the selector switch	Any data type	As each data type suggested.

- **Function**

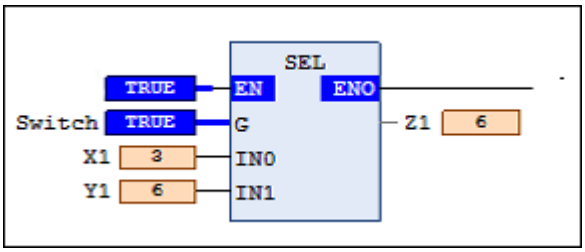
  The FC instruction is used to select one of the variables as the output with the help of selector switch. The output is the *X1* when the selector switch is FALSE, and the output is *Y1* when the selector switch is TRUE.

- **Programming Example**

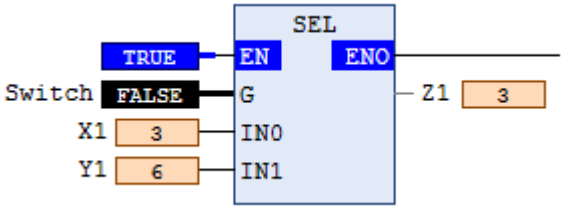
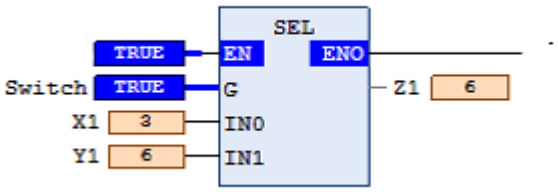
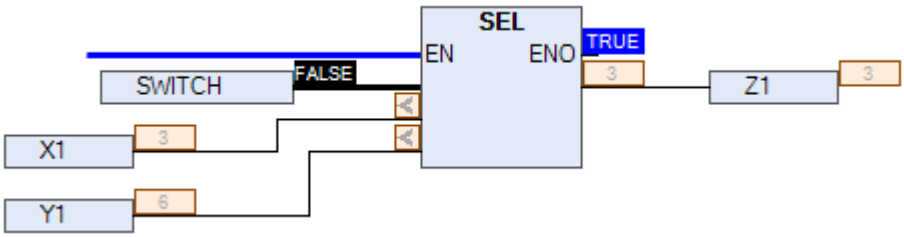
  The FC instruction (SEL) is used to select a value among inputs with the help of selector switch. For example, if *X1* (3) and *Y1* (6) are the inputs and the selector switch is FALSE, then according to the SEL instruction, *Z1* is 3 as output.



If *X1* (3) and *Y1* (6) are the inputs and the selector switch is TRUE, then according to SEL instruction, *Z1* is 6 as output.



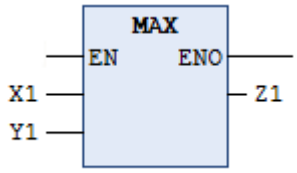
Feature in the ST/LD/FBD & CFC editor:

Programming Language	Program
ST	Z1 6 := SEL(Switch TRUE ,X1 3 ,Y1 6);
LD	
FBD	
CFC	

- **Supported Products**
  - AX series
- **Library**
  - None

18.5.2. MAX (Get Maximum Value Instruction)

MAX selects the maximum value from two input data as an output.

FB/FC	Instruction	Graphic Expression	ST Language
FC	MAX		Z1 := MAX(X1,Y1);

- **Inputs**

Name	Function	Data Type	Setting Value
X1, Y1	Input	Any data type	As each data type suggested.

- **Outputs**

Name	Function	Data Type	Setting Value
Z1	Maximum of both input values	Any data type	As each data type suggested.

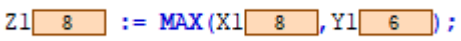
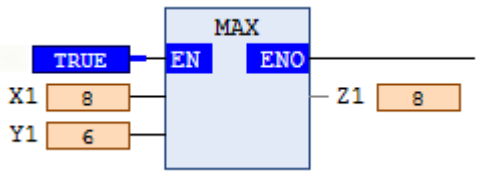
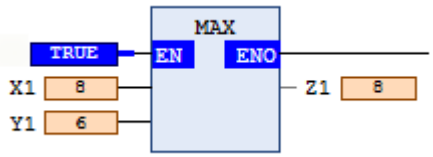
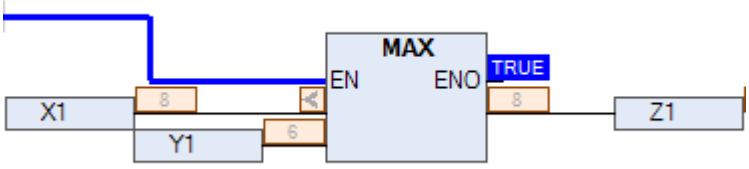
- **Function**

  This FC instruction is used to select the maximum value from variable *X1* and variable *Y1*.

- **Programming Example**

  The FC instruction (MAX) is used to select a maximum value among inputs. For example, if *X1* (8) and *Y1* (6) are inputs, then according to the MAX instruction, the output is *Z1* (8).

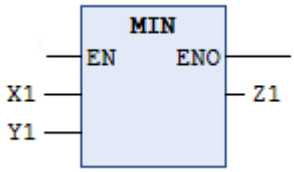
  Feature in the ST/LD/FBD & CFC editor:

Programming Language	Program
ST	
LD	
FBD	
CFC	

- **Supported Products**
  - AX series
- **Library**
  - None

18.5.3. MIN (Get the Minimum Value Instruction)

MIN select the minimum value from two input data as an output.

FB/FC	Instruction	Graphic Expression	ST Language
FC	MIN		Z1 := MIN(X1,Y1);

- **Inputs**

Name	Function	Data Type	Setting Value
X1, Y1	Input	Any data type	As each data type suggested.

- **Outputs**

Name	Function	Data Type	Setting Value
Z1	Minimum of both input values	Any data type	As each data type suggested.

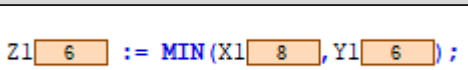
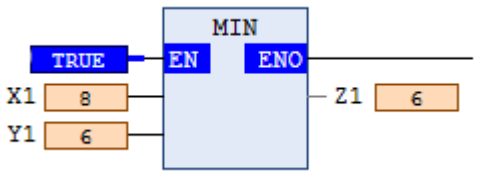
- **Function**

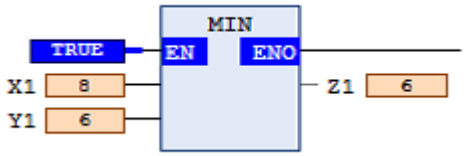
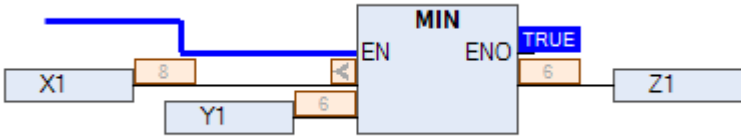
  This FC instruction is used to select the minimum value from variable *X1* and variable *Y1*.

- **Programming Example**

  The FC instruction (MIN) is used to select a minimum value among inputs. For example, if *X1* (8) and *Y1* (6) are inputs, then according to the MIN instruction, output is *Z1* (6).

  Feature in the ST/LD/FBD & CFC editor:

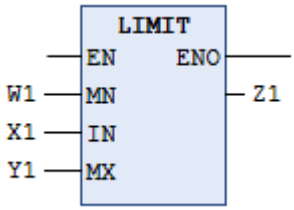
Programming Language	Program
ST	
LD	

Programming Language	Program
FBD	
CFC	

- **Supported Products**
  - AX series
- **Library**
  - None

18.5.4. LIMIT (Limit Value Instruction)

LIMIT limits the ouput data. If the input data is between the minimum value and maximum value, then directly that value is considered as output data. If the input value is lower than the minimum value, then the minimum value is taken as output value, and if the input value is greater than the maximum value, then the maximum value is taken as output value.

FB/FC	Instruction	Graphic Expression	ST Language
FC	LIMIT		Z1 := LIMIT(W1,X1,Y1);

- **Inputs**

Name	Function	Data Type	Setting Value
X1	Input	Any data type	As each data type suggested.
Y1	Maximum value		
W1	Minimum value		

- **Outputs**

Name	Function	Data Type	Setting Value
Z1	Input value within the range	Any data type	As each data type suggested.

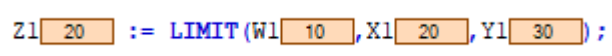
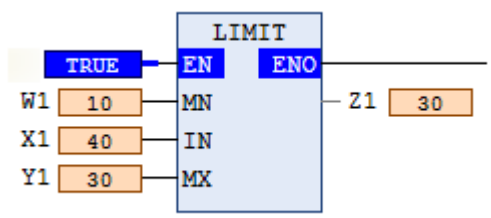
- **Function**

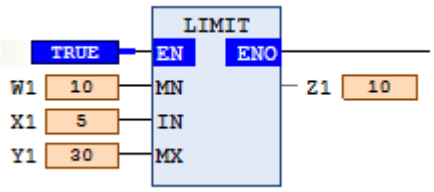
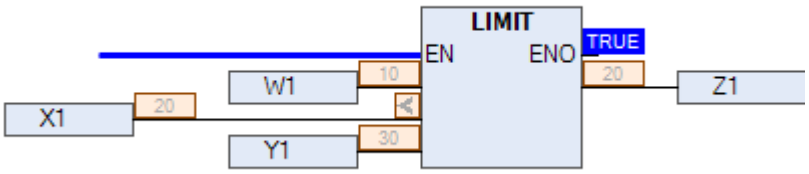
  This FC instruction is used to restrict the input value to a given interval. Input values between the minimum and maximum are unchanged. Input values greater than the maximum are replaced with the maximum value. Input values less than the minimum are replaced with the minimum value.

- **Programming Example**

  The FC instruction (LIMIT) is used to select the in–between value among inputs. For example, if *W1* (10), *X1* (40), and *Y1* (30) are inputs, then according to the LIMIT instruction, output is in between value *Z1* (30).

  Feature in the ST/LD/FBD & CFC editor:

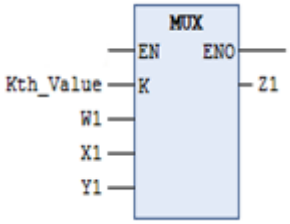
Programming Language	Program
ST	
LD	

Programming Language	Program
FBD	
CFC	

- **Supported Products**
  - AX series
- **Library**
  - None

18.5.5. MUX (Multiple Choice Instruction)

MUX select one value based on the value of control number (*Kth_Value*) as output data for multiple input data. The input value with the sequence *k*+1 is selected as the output. The starting value of *K* is 0. If *K* is greater than other input values, then automatically the last input value is considered as output.

FB/FC	Instruction	Graphic Expression	ST Language
FC	MUX		Z1 := MUX(Kth_Value,W1,X1,Y1);

- **Inputs**

Name	Function	Data Type	Setting Value
W1, X1, Y1	Input	Any data type	As each data type suggested.
Kth_Value	Control number	BYTE, WORD, DWORD, LWORD, USINT, UINT, UDINT, ULINT, SINT, INT, DINT, LINT	

- **Outputs**

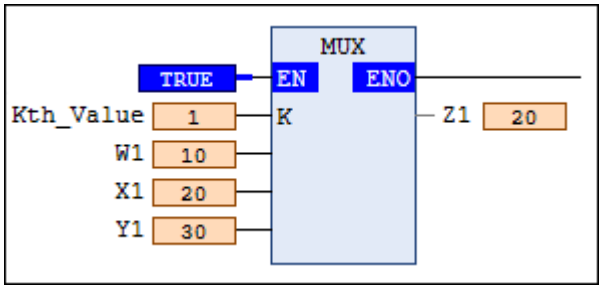
Name	Function	Data Type	Setting Value
Z1	Execution result	Any data type	As each data type suggested.

- **Function**

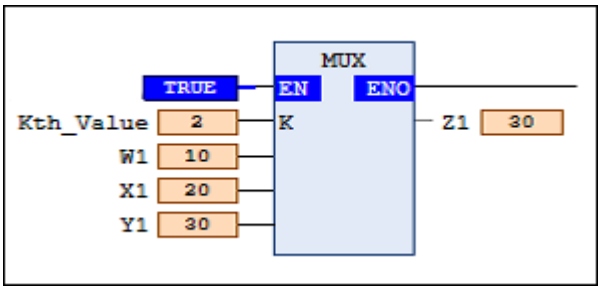
  This FC instruction is used to select the output value based on the value of control number (*Kth_Value*) amongst variable *W1*, variable *X1*, and variable *Y1*.

- **Programming Example**

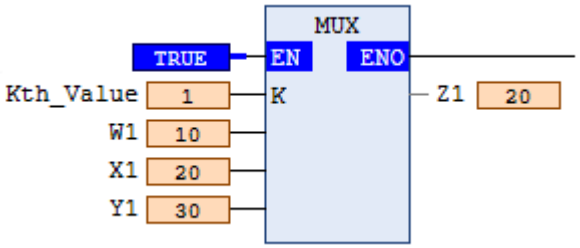
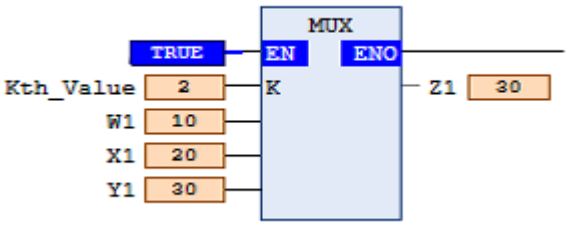
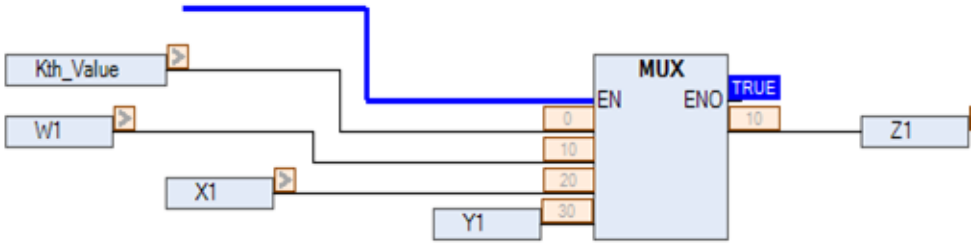
  The FC instruction (MUX) is used to select a value among inputs with the help of *Kth_Value*. For example, if *W1* (10), *X1* (20), and *Y1* (30) are the inputs and *Kth_Value* is 1, then according to the MUX instruction, *Z1* is 20 as output.



  If *W1* (10), *X1* (20), and *Y1* (30) are the inputs and *Kth_Value* is 2, then according to the MUX instruction, *Z1* is 30 as output.



Feature in the ST/LD/FBD & CFC editor:

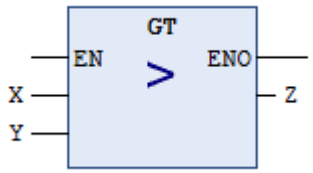
Programming Language	Program
ST	Z1 10 := MUX(Kth_Value 0 ,W1 10 ,X1 20 ,Y1 30);
LD	
FBD	
CFC	

- **Supported Products**
  - AX series
- **Library**
  - None

18.6. Comparison Instructions

18.6.1. GT (Greater than Instruction)

GT checks if the first input is greater than the second input. The output is TRUE when the first number is greater than the second number. Otherwise, it is FALSE.

FB/FC	Instruction	Graphic Expression	ST Language
FC	GT		Z := X > Y;

- **Inputs**

Name	Function	Data Type	Setting Value
X, Y	Input	Any basic data type	As each data type suggested.

- **Outputs**

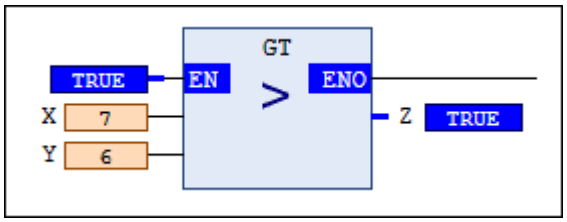
Name	Function	Data Type	Setting Value
Z	Execution result	BOOL	TRUE, FALSE

- **Function**

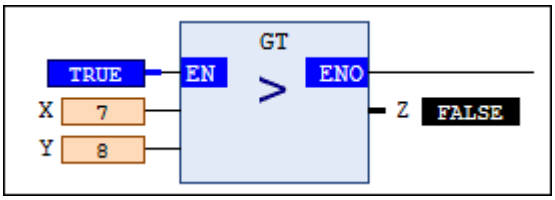
  This FC instruction is used to compare the size of the two inputs, and output TRUE when the first number is greater than the second number, otherwise the output is FALSE.

- **Programming Example**

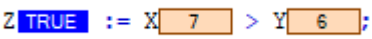
  The FC instruction (GT) is used to compare the value size of inputs. For example, if *X* (7) and *Y* (6) are inputs, then according to the GT instruction, output *Z* is TRUE.

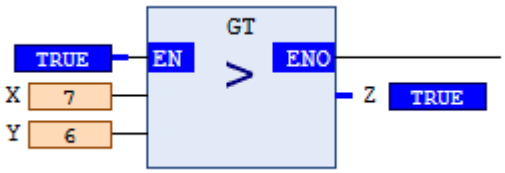
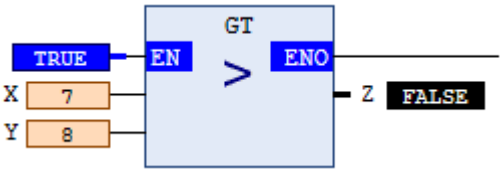
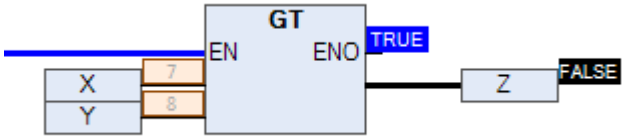


If *X* (7) and *Y* (8) are inputs, then according to the GT instruction, output *Z* is FALSE.



Feature in the ST/LD/FBD & CFC editor:

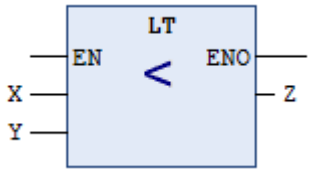
Programming Language	Program
ST	

Programming Language	Program
LD	
FBD	
CFC	

- **Supported Products**
  - AX series
- **Library**
  - None

18.6.2. LT (Less than Instruction)

LT checks if the first input is less than the second input. The output is TRUE when the first number is smaller than the second number. Otherwise, it is FALSE.

FB/FC	Instruction	Graphic Expression	ST Language
FC	LT		Z := X < Y;

- **Inputs**

Name	Function	Data Type	Setting Value
X, Y	Input	Any basic data type	As each data type suggested.

- **Outputs**

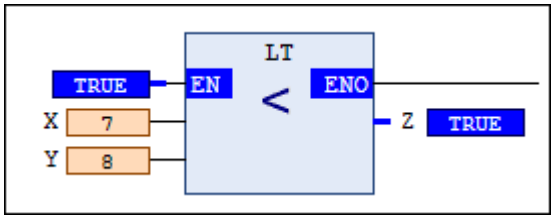
Name	Function	Data Type	Setting Value
Z	Execution result	BOOL	TRUE, FALSE

- **Function**

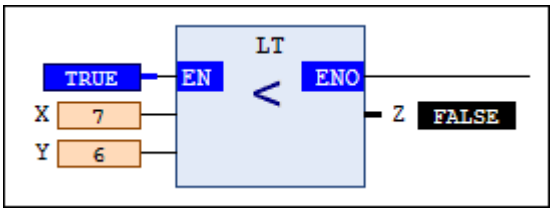
  This FC instruction is used to compare the size of the two inputs, and output TRUE when the first number is less than the second number, otherwise the output is FALSE.

- **Programming Example**

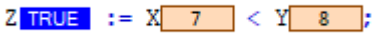
  The FC instruction (LT) is used to compare the value size between inputs. For example, if *X* (7) and *Y* (8) are inputs, then according to the LT instruction, output *Z* is TRUE.

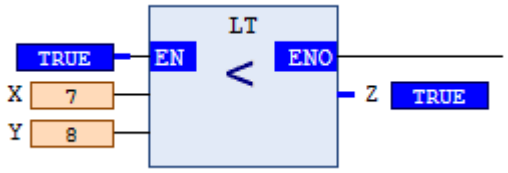
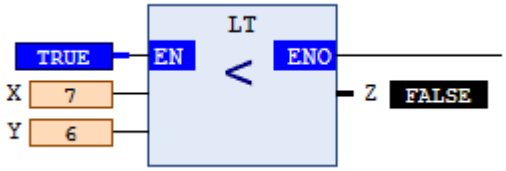
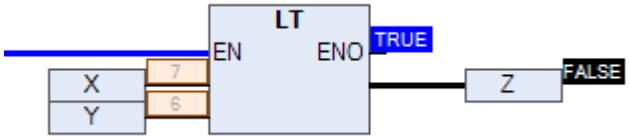


If *X* (7) and *Y* (6) are inputs, then according to the LT instruction, output *Z* is FALSE.



Feature in the ST/LD/FBD & CFC editor:

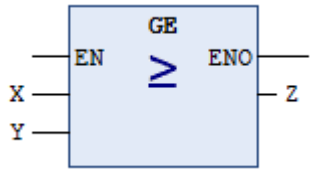
Programming Language	Program
ST	

Programming Language	Program
LD	
FBD	
CFC	

- **Supported Products**
  - AX series
- **Library**
  - None

18.6.3. GE (Greater than or Equal to Instruction)

GE checks if the first input is greater than or equal to the second input. The output is TRUE when the first number is greater than or equal to the second number. Otherwise, it is FALSE.

FB/FC	Instruction	Graphic Expression	ST Language
FC	GE		Z := X >= Y;

- **Inputs**

Name	Function	Data Type	Setting Value
X, Y	Input	Any basic data type	As each data type suggested.

- **Outputs**

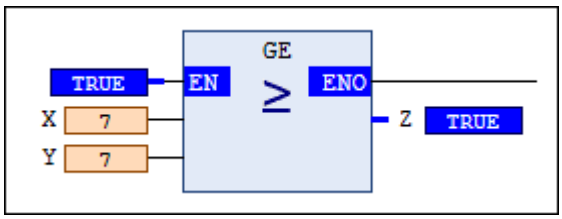
Name	Function	Data Type	Setting Value
Z	Execution result	BOOL	TRUE, FALSE

- **Function**

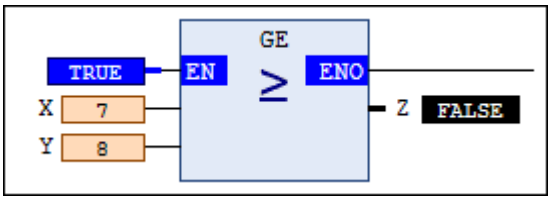
  This FC instruction is used to compare the size of the two inputs, and output TRUE when the first number is greater than or equal to the second number, otherwise the output is FALSE.

- **Programming Example**

  The FC instruction (GE) is used to compare the value size between inputs. For example, if *X* (7) and *Y* (7) are inputs, then according to the GE instruction, output *Z* is TRUE.

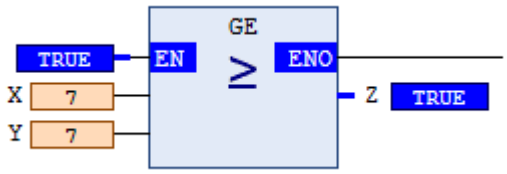
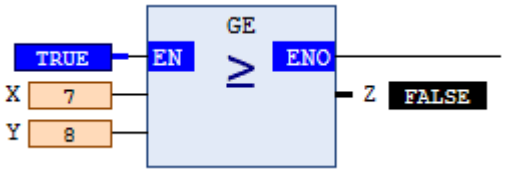
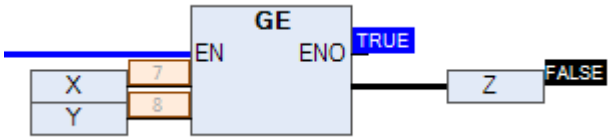


If *X* (7) and *Y* (8) are inputs, then according to the GE instruction, output *Z* is FALSE.



Feature in the ST/LD/FBD & CFC editor:

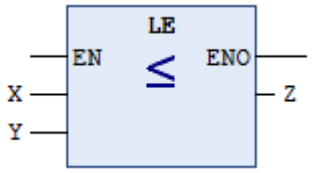
Programming Language	Program
ST	Z TRUE := X 7 >= Y 6;

Programming Language	Program
LD	
FBD	
CFC	

- **Supported Products**
  - AX series
- **Library**
  - None

18.6.4. LE (Less than or Equal to Instruction)

LE checks if the first input is less than or equal to the second input. The output is TRUE when the first number is smaller than or equal to the second number. Otherwise, it is FALSE.

FB/FC	Instruction	Graphic Expression	ST Language
FC	LE		Z := X <= Y;

- **Inputs**

Name	Function	Data Type	Setting Value
X, Y	Input	Any basic data type	As each data type suggested.

- **Outputs**

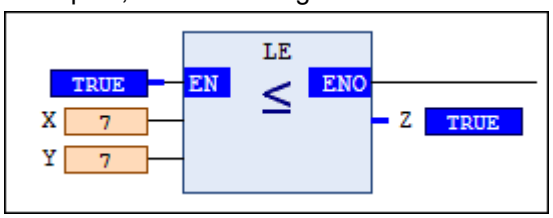
Name	Function	Data Type	Setting Value
Z	Execution result	BOOL	TRUE, FALSE

- **Function**

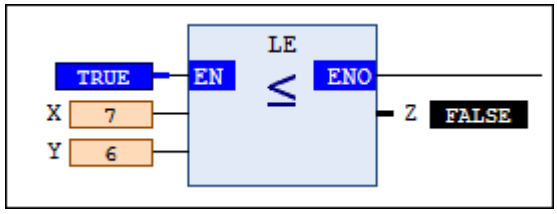
  This FC instruction is used to compare the size of the two inputs, and output TRUE when the first number is smaller than or equal to the second number, otherwise the output is FALSE.

- **Programming Example**

  The FC instruction (LE) is used to compare the value size between inputs. For example, if *X* (7) and *Y* (7) are inputs, then according to the LE instruction, output *Z* is TRUE.

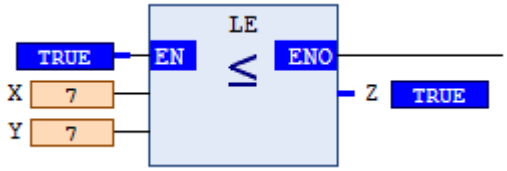
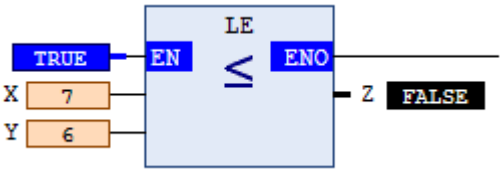
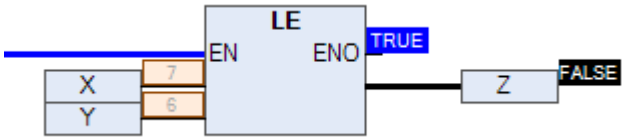


If *X* (7) and *Y* (6) are inputs, then according to the LE instruction, output *Z* is FALSE.



Feature in the ST/LD/FBD & CFC editor:

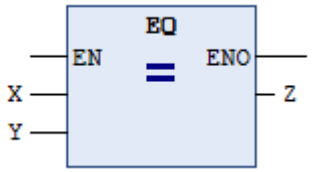
Programming Language	Program
ST	Z TRUE := X 7 <= Y 8;

Programming Language	Program
LD	
FBD	
CFC	

- **Supported Products**
  - AX series
- **Library**
  - None

18.6.5. EQ (Equal to Instruction)

EQ checks if the first input is equal to the second input. The output is TRUE when the first number is equal to the second number. Otherwise, it is FALSE.

FB/FC	Instruction	Graphic Expression	ST Language
FC	EQ		Z := X = Y;

- **Inputs**

Name	Function	Data Type	Setting Value
X, Y	Input	Any basic data type	As each data type suggested.

- **Outputs**

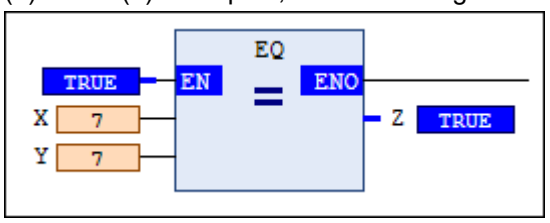
Name	Function	Data Type	Setting Value
Z	Execution result	BOOL	TRUE, FALSE

- **Function**

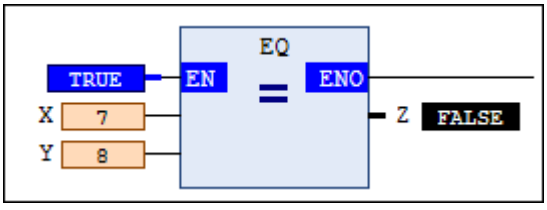
  This FC instruction is used to compare whether the value of the two inputs is equal, and output TRUE when the value is equal, otherwise the output is FALSE.

- **Programming Example**

  The FC instruction (EQ) is used to compare whether the value of the two inputs is equal. For example, if *X* (7) and *Y* (7) are inputs, then according to the EQ instruction, output *Z* is TRUE.

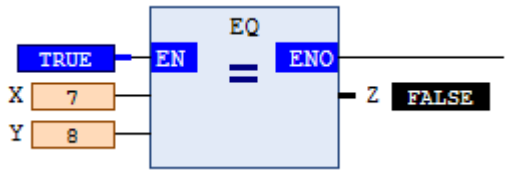
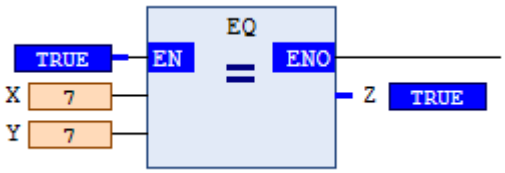
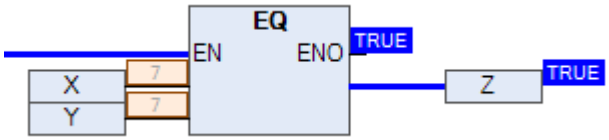


If *X* (7) and *Y* (8) are inputs, then according to the EQ instruction, output *Z* is FALSE.



Feature in the ST/LD/FBD & CFC editor:

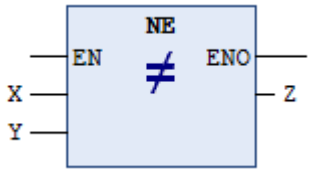
Programming Language	Program
ST	Z TRUE := X 7 = Y 7 ;

Programming Language	Program
LD	
FBD	
CFC	

- **Supported Products**
  - AX series
- **Library**
  - None

18.6.6. NE (Not Equal to Instruction)

NE checks if the first input is enot qual to the second input. The output is TRUE when the first number is not equal to the second number. Otherwise, it is FALSE.

FB/FC	Instruction	Graphic Expression	ST Language
FC	NE		Z := X <> Y;

- **Inputs**

Name	Function	Data Type	Setting Value
X, Y	Input	Any basic data type	As each data type suggested.

- **Outputs**

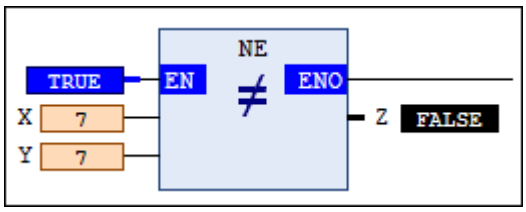
Name	Function	Data Type	Setting Value
Z	Execution result	BOOL	TRUE, FALSE

- **Function**

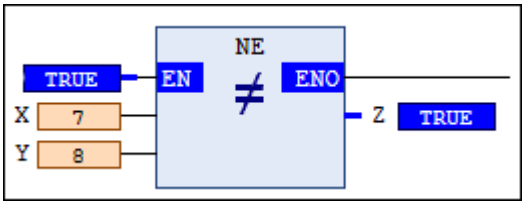
  This FC instruction is used to compare whether the value of the two inputs is not equal, and output TRUE when the value is not equal, otherwise the output is FALSE.

- **Programming Example**

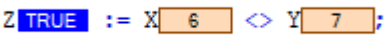
  The FC instruction (NE) is used to compare whether the value of the two inputs is not equal. For example, if *X* (7) and *Y* (7) are inputs, then according to the NE instruction, output *Z* is FALSE.

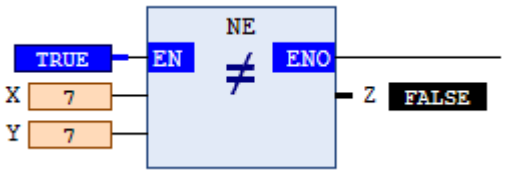
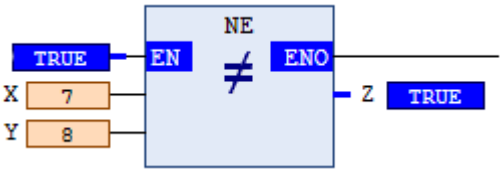
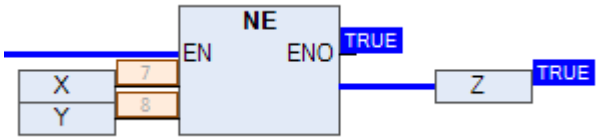


If *X* (7) and *Y* (8) are inputs, then according to the NE instruction, output *Z* is TRUE.



Feature in the ST/LD/FBD & CFC editor:

Programming Language	Program
ST	

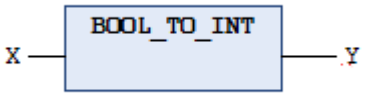
Programming Language	Program
LD	
FBD	
CFC	

- **Supported Products**
  - AX series
- **Library**
  - None

18.7. Data Type Conversion Instructions

18.7.1. BOOL_TO_<TYPE> (Boolean Type Conversion Instruction)

BOOL_TO_<TYPE> converts the Boolean data type to other data types.

FB/FC	Instruction	Graphic Expression	ST Language
FC	BOOL_TO_<TYPE>		Y := BOOL_TO_INT(X);

- **Inputs**

Name	Function	Data Type	Setting Value
X	Input	BOOL	TRUE, FALSE

- **Outputs**

Name	Function	Data Type	Setting Value
Z	Converted value	Any data type	As each data type suggested.

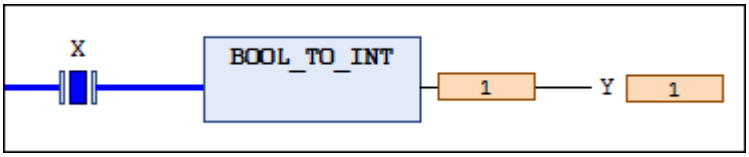
- **Function**

  This FC instruction is used to convert a Boolean value into the specified data type and return a type–converted value. When the output is of the numeric type, the input TRUE becomes 1 and FALSE becomes 0; When the output is a STRING type, the input TRUE becomes "TRUE" and FALSE becomes "FALSE".

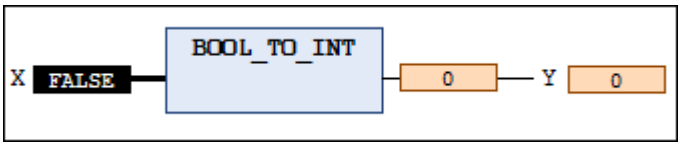
- **Programming Example**

  The FC instruction (BOOL_TO_<TYPE>

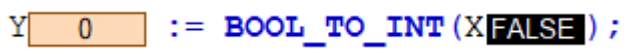
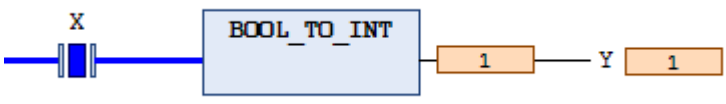
) is used to change the BOOL value of variable *X* to INT type. If the input is TRUE, then output is 1, and if the input is FALSE, then output is 0. For example, if input *X* is TRUE, then output *Y* is 1.

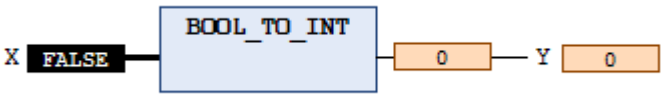
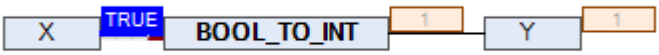


If input *X* is FALSE, then output *Y* is 0.



Feature in the ST/LD/FBD & CFC editor:

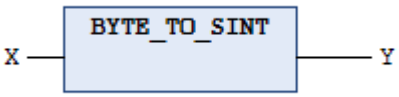
Programming Language	Program
ST	
LD	

Programming Language	Program
FBD	
CFC	

- **Supported Products**
  - AX series
- **Library**
  - None

18.7.2. BYTE_TO_<TYPE> (Byte Type Conversion Instruction)

BYTE_TO_<TYPE> converts the Byte data type to other data types.

FB/FC	Instruction	Graphic Expression	ST Language
FC	BYTE_TO_<TYPE>		Y := BYTE_TO_SINT(X);

- **Inputs**

Name	Function	Data Type	Setting Value
X	Input	BYTE	0 to 255

- **Outputs**

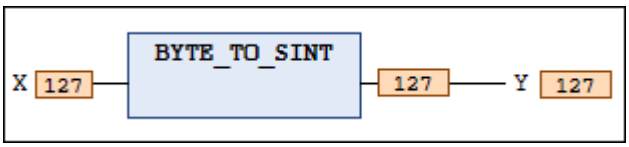
Name	Function	Data Type	Setting Value
Y	Converted value	Any data type	As each data type suggested.

- **Function**

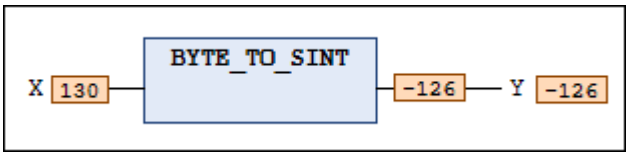
  This FC instruction is used to convert a Byte value to the specified data type and return a type–converted value.

- **Programming Example**

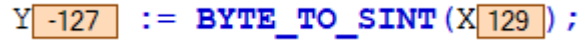
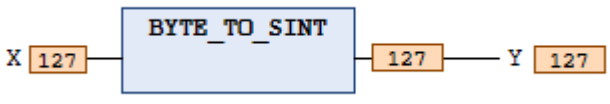
  The FC instruction (BYTE_TO_<TYPE>) is used to change the value of variable *X* to SINT type, where the lower limit and upper limit for data type BYTE is from 0 to 255, and the lower limit and upper limit for data type SINT is from –128 to 127. For example, if input *X* is 127, then output *Y* is 127.

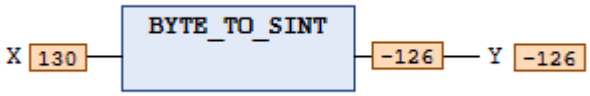
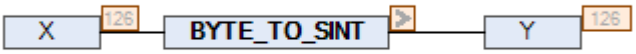


If input *X* is 130, then output *Y* is –126.



Feature in the ST/LD/FBD & CFC editor:

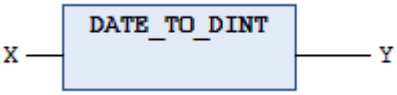
Programming Language	Program
ST	
LD	

Programming Language	Program
FBD	
CFC	

- **Supported Products**
  - AX series
- **Library**
  - None

18.7.3. DATE_TO_<TYPE> (Date Type Conversion Instruction)

DATE_TO_<TYPE> converts the Date data type to other data types.

FB/FC	Instruction	Graphic Expression	ST Language
FC	DATE_TO_<TYPE>		Y := DATE_TO_DINT(X);

- **Inputs**

Name	Function	Data Type	Setting Value
X	Input	DATE	D#1970–01–01 to D#2106–02–07

- **Outputs**

Name	Function	Data Type	Setting Value
Z	Converted value	Any data type	As each data type suggested.

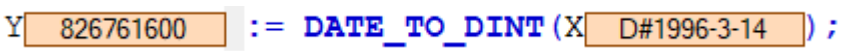
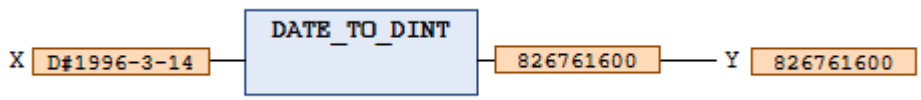
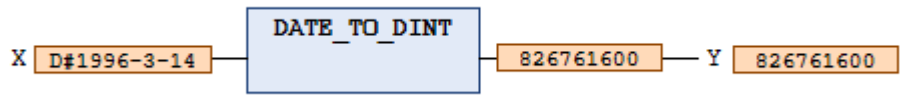
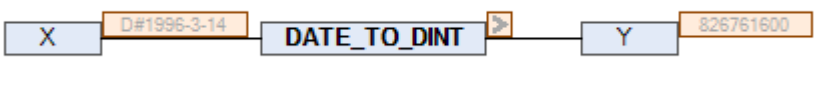
- **Function**

  This FC instruction is used to convert a Date data type value, which starts from 1st January 1970, into the specified data type and return a type–converted value.

- **Programming Example**

  The FC instruction (DATE_TO_<TYPE>) is used to change the value of variable *X* to DINT type, where the lower limit and upper limit for data type DATE is from D#1970–01–01 to D#2106–02–07, and the lower limit and upper limit for data type DINT is from –2147483648 to 2147483647. For example, if input *X* is D#1996–03–14, then output *Y* is 826761600.

  Feature in the ST/LD/FBD & CFC editor:

Programming Language	Program
ST	
LD	
FBD	
CFC	

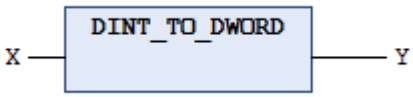
- **Supported Products**

  - AX series

- **Library**
  - None

18.7.4. DINT_TO_<TYPE> (Double Integer Type Conversion Instruction)

DINT_TO_<TYPE> converts the Dint data type to other data types.

FB/FC	Instruction	Graphic Expression	ST Language
FC	DINT_TO_<TYPE>		Y := DINT_TO_DWORD(X);

- **Inputs**

Name	Function	Data Type	Setting Value
X	Input	DINT	–2147483648 to 2147483647

- **Outputs**

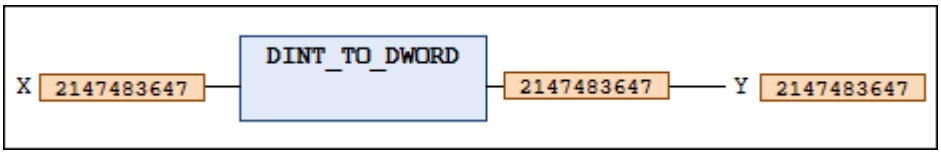
Name	Function	Data Type	Setting Value
Z	Converted value	Any data type	As each data type suggested.

- **Function**

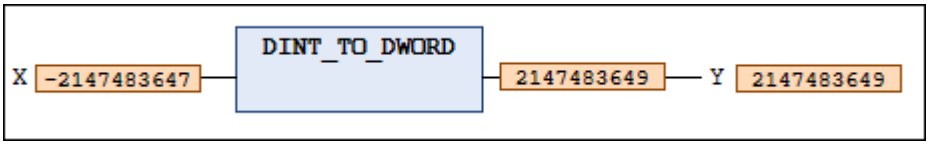
  This FC instruction is used to convert a DINT data type value into the specified data type and return a type–converted value.

- **Programming Example**

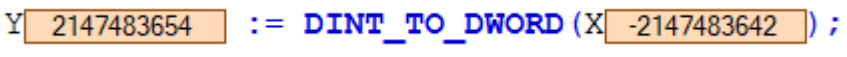
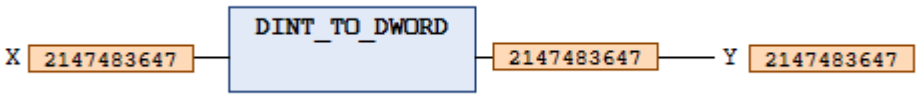
  The FC instruction (DINT_TO_<TYPE>) is used to change the value of variable *X* to DWORD type, where the lower limit and upper limit for data type DINT is from –2147483648 to 2147483647, and the lower limit and upper limit for data type DWORD is from 0 to 4294967295. For example, if input *X* is 2147483647, then output *Y* is 2147483647.

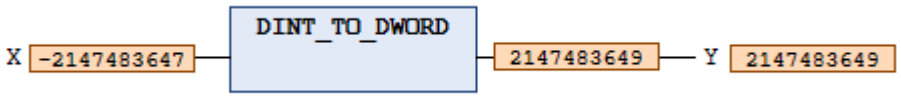
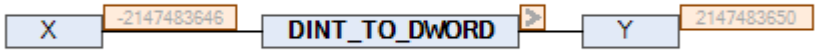


If input *X* is –2147483647, then output *Y* is 2147483649.



Feature in the ST/LD/FBD & CFC editor:

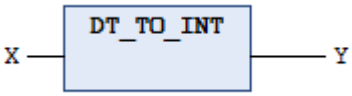
Programming Language	Program
ST	
LD	

Programming Language	Program
FBD	
CFC	

- **Supported Products**
  - AX series
- **Library**
  - None

18.7.5. DT_TO_<TYPE> (Date and Time Type Conversion Instruction)

DT_TO_<TYPE> converts the DATE_AND_TIME data type to other data types.

FB/FC	Instruction	Graphic Expression	ST Language
FC	DT_TO_<TYPE>		Y := DT_TO_INT(X);

- **Inputs**

Name	Function	Data Type	Setting Value
X	Input	DT	DT#1970–01–01–0:0:0 to DT#2106–02–07–06:28:15

- **Outputs**

Name	Function	Data Type	Setting Value
Z	Converted value	Any data type	As each data type suggested.

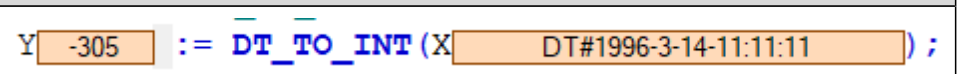
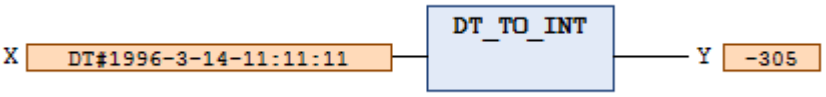
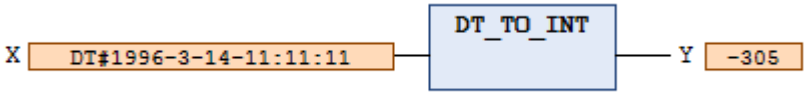
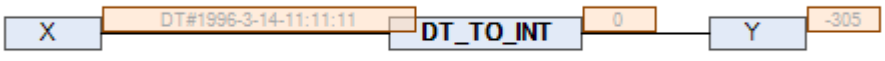
- **Function**

  This FC instruction is used to convert a DATE_AND_TIME data type value, which starts from 1st January 1970, into the specified data type and return a type–converted value.

- **Programming Example**

  The FC instruction (DT_TO_<TYPE>) is used to change the value of variable *X* to INT type, where the lower limit and upper limit for data type DT is from D#1970–01–01–0:0:0 to D#2106–02–07–06:28:15, and the lower limit and upper limit for data type INT is from –32768 to 32767. For example, if input *X* is D#1996–03–14–11:11:11, then output *Y* is –305.

  Feature in the ST/LD/FBD & CFC editor:

Programming Language	Program
ST	
LD	
FBD	
CFC	

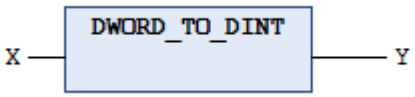
- **Supported Products**

  - AX series

- **Library**
  - None

18.7.6. DWORD_TO_<TYPE> (Double Word Type Conversion Instruction)

DWORD_TO_<TYPE> converts the Double Word data type to other data types.

FB/FC	Instruction	Graphic Expression	ST Language
FC	DWORD_TO_<TYPE>		Y := DT_TO_INT(X);

- **Inputs**

Name	Function	Data Type	Setting Value
X	Input	DWORD	0 to 4294967295

- **Outputs**

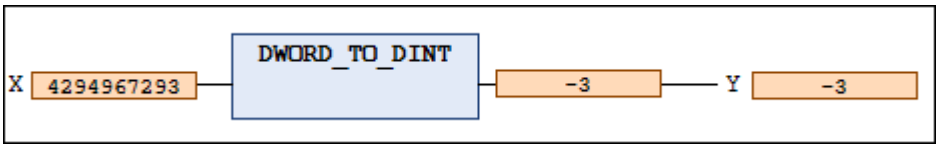
Name	Function	Data Type	Setting Value
Y	Converted value	Any data type	As each data type suggested.

- **Function**

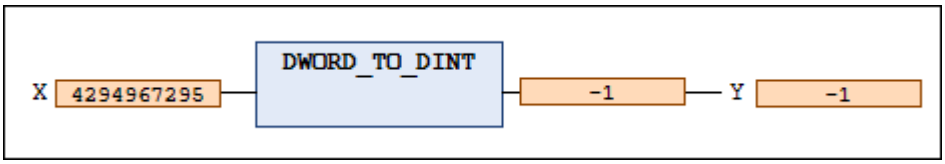
  This FC instruction is used to convert a DWORD data type value into the specified data type and return a type–converted value.

- **Programming Example**

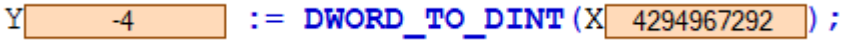
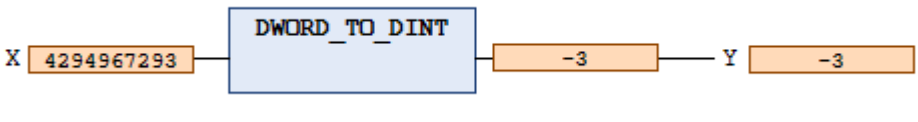
  The FC instruction (DWORD_TO_<TYPE>) is used to change the value of variable *X* to DINT type, where the lower limit and upper limit for data type DWORD is from 0 to 4294967295, and the lower limit and upper limit for data type DINT is from –2147483648 to 2147483647. For example, if input *X* is 42949672, then output *Y* is –3.

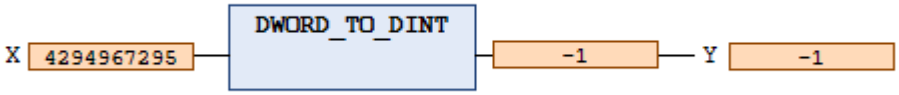
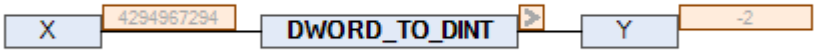


If input *X* is 42949675, then output *Y* is –1.



Feature in the ST/LD/FBD & CFC editor:

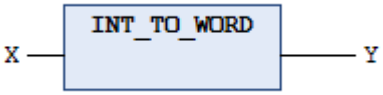
Programming Language	Program
ST	
LD	

Programming Language	Program
FBD	
CFC	

- **Supported Products**
  - AX series
- **Library**
  - None

18.7.7. INT_TO_<TYPE> (Integer Type Conversion Instruction)

INT_TO_<TYPE> converts the Integer data type to other data types.

FB/FC	Instruction	Graphic Expression	ST Language
FC	INT_TO_<TYPE>		Y := INT_TO_WORD(X);

- **Inputs**

Name	Function	Data Type	Setting Value
X	Input	INT	–32768 to 32767

- **Outputs**

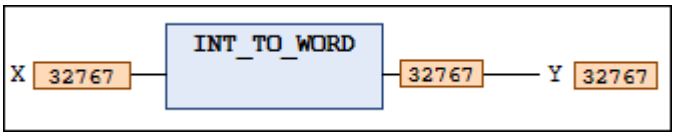
Name	Function	Data Type	Setting Value
Y	Converted value	Any data type	As each data type suggested.

- **Function**

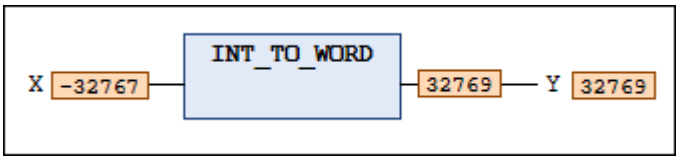
  This FC instruction is used to convert an INT data type value into the specified data type and return a type–converted value.

- **Programming Example**

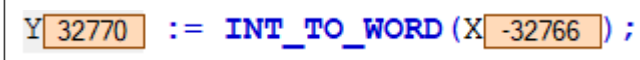
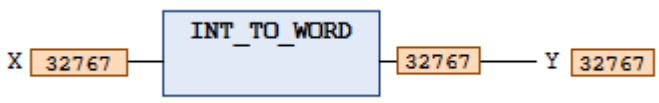
  The FC instruction (INT_TO_<TYPE>) is used to change the value of variable *X* to WORD type, where the lower limit and upper limit for data type INT is from –32768 to 32767, and the lower limit and upper limit for data type WORD is from 0 to 65535. For example, if input *X* is 32767, then output *Y* is 32767.

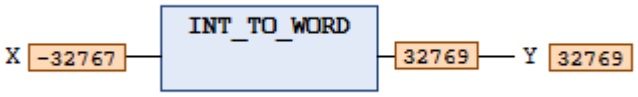
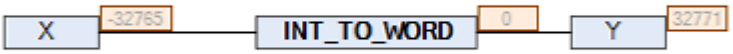


If input *X* is –32767, then output *Y* is 32769.



Feature in the ST/LD/FBD & CFC editor:

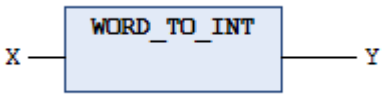
Programming Language	Program
ST	
LD	

Programming Language	Program
FBD	
CFC	

- **Supported Products**
  - AX series
- **Library**
  - None

18.7.8. WORD_TO_<TYPE> (Word Type Conversion Instruction)

WORD_TO_<TYPE> converts the Word data type to other data types.

FB/FC	Instruction	Graphic Expression	ST Language
FC	WORD_TO_<TYPE>		Y := WORD_TO_INT(X);

- **Inputs**

Name	Function	Data Type	Setting Value
X	Input	WORD	0 to 65535

- **Outputs**

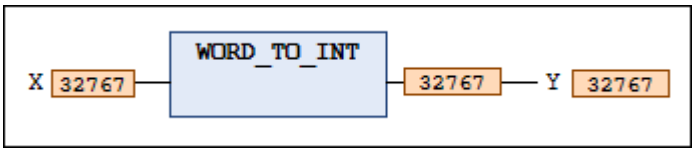
Name	Function	Data Type	Setting Value
Y	Converted value	Any data type	As each data type suggested.

- **Function**

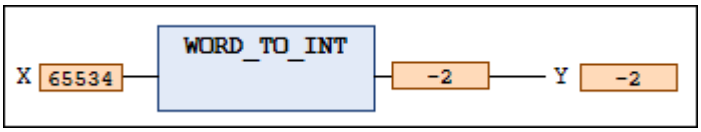
  This FC instruction is used to convert a WORD data type value into the specified data type and return a type–converted value.

- **Programming Example**

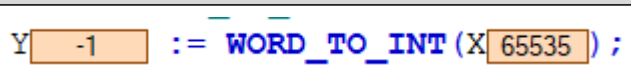
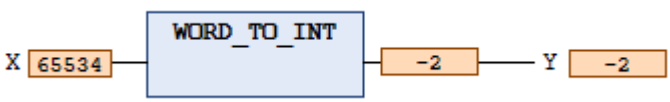
  The FC instruction (WORD_TO_<TYPE>) is used to change the value of variable *X* to INT type, where the lower limit and upper limit for data type WORD is from 0 to 65535, and the lower limit and upper limit for data type INT is from –32768 to 32767. For example, if input *X* is 32767, then output *Y* is 32767.

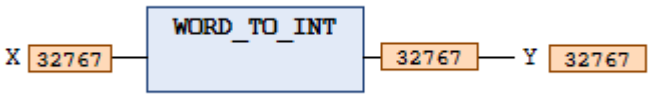
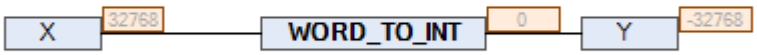


If input *X* is 65534, then output *Y* is –2.



Feature in the ST/LD/FBD & CFC editor:

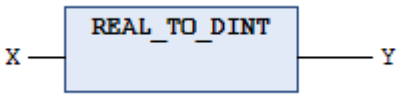
Programming Language	Program
ST	
LD	

Programming Language	Program
FBD	
CFC	

- **Supported Products**
  - AX series
- **Library**
  - None

18.7.9. REAL_TO_<TYPE> (Real Type Conversion Instruction)

REAL_TO_<TYPE> converts the Real data type to other data types.

FB/FC	Instruction	Graphic Expression	ST Language
FC	REAL_TO_<TYPE>		Y := REAL_TO_DINT(X);

- **Inputs**

Name	Function	Data Type	Setting Value
X	Input	REAL	1.0E–44 to 3.402823E+38

- **Outputs**

Name	Function	Data Type	Setting Value
Y	Converted value	Any data type	As each data type suggested.

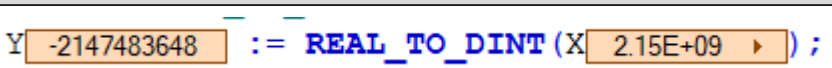
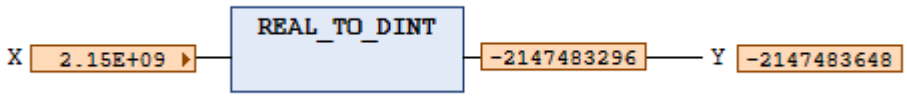
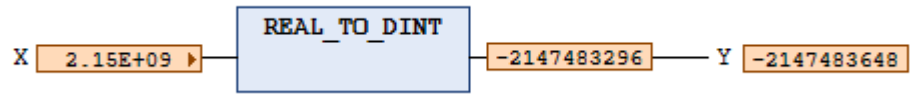
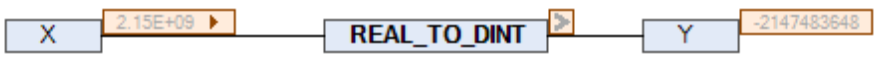
- **Function**

  This FC instruction is used to convert a REAL data type value into the specified data type and return a type–converted value. The value will be rounded up or down to the nearest whole number and converted into the new variable type. Exceptions to this are the variable types STRING, BOOL, REAL, and LREAL.

- **Programming Example**

  The FC instruction (REAL_TO_<TYPE>) is used to change the value of variable *X* to DINT type, where the lower limit and upper limit for data type REAL is from 1.0E–44 to 3.402823E+38, and the lower limit and upper limit for data type DINT is from –2147483648 to 2147483647. For example, if input *X* is 2.15E+09, then output *Y* is –2147483296.

  Feature in the ST/LD/FBD & CFC editor:

Programming Language	Program
ST	
LD	
FBD	
CFC	

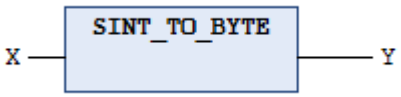
- **Supported Products**

  - AX series

- **Library**
  - None

18.7.10. SINT_TO_<TYPE> (Short Integer Type Conversion Instruction)

SINT_TO_<TYPE> converts the Short Integer data type to other data types.

FB/FC	Instruction	Graphic Expression	ST Language
FC	SINT_TO_<TYPE>		Y := SINT_TO_BYTE(X);

- **Inputs**

Name	Function	Data Type	Setting Value
X	Input	SINT	–128 to 127

- **Outputs**

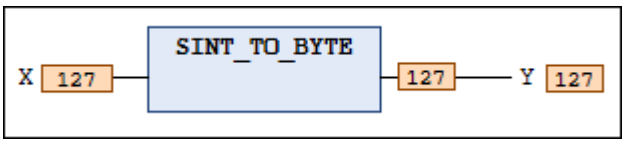
Name	Function	Data Type	Setting Value
Y	Converted value	Any data type	As each data type suggested.

- **Function**

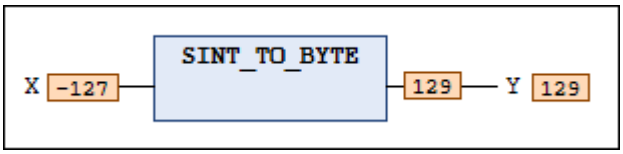
  This FC instruction is used to convert a SINT data type value into the specified data type and return a type–converted value.

- **Programming Example**

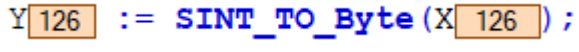
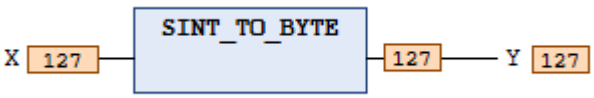
  The FC instruction (SINT_TO_<TYPE>) is used to change the value of variable *X* to BYTE type, where the lower limit and upper limit for data type SINT is from –128 to 127, and the lower limit and upper limit for data type BYTE is from 0 to 255. For example, if input *X* is 127, then output *Y* is 127.

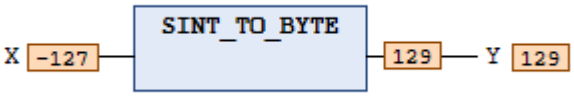
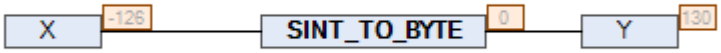


If input *X* is –127, then output *Y* is 129.



Feature in the ST/LD/FBD & CFC editor:

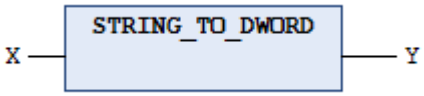
Programming Language	Program
ST	
LD	

Programming Language	Program
FBD	
CFC	

- **Supported Products**
  - AX series
- **Library**
  - None

18.7.11. STRING_TO_<TYPE> (String Type Conversion Instruction)

STRING_TO_<TYPE> converts the Strings data type to other data types.

FB/FC	Instruction	Graphic Expression	ST Language
FC	STRING_TO_<TYPE>		Y := STRING_TO_DWORD(X);

- **Inputs**

Name	Function	Data Type	Setting Value
X	Input	STRING	As each data type suggested.

- **Outputs**

Name	Function	Data Type	Setting Value
Y	Converted value	Any data type	As each data type suggested.

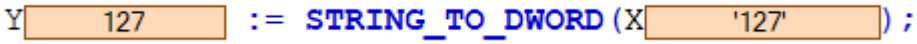
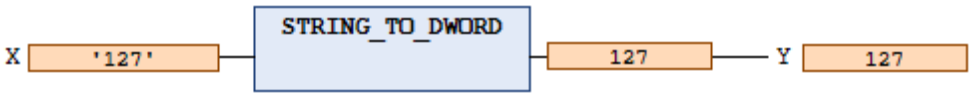
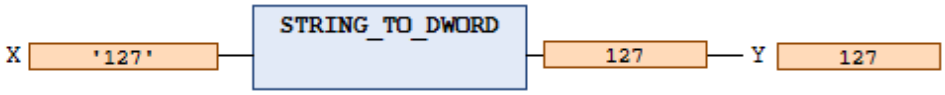
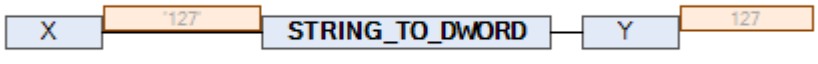
- **Function**

  This FC instruction is used to convert a STRING into the specified data type and return a type–converted value.

- **Programming Example**

  The FC instruction (STRING_TO_<TYPE>) is used to change the value of variable *X* to DINT type, where the limit for data type STRING is one byte per character, and the lower limit and upper limit for data type DWORD is from 0 to 4294967295. For example, if input *X* is 127, then output *Y* is 127.

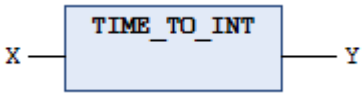
  Feature in the ST/LD/FBD & CFC editor:

Programming Language	Program
ST	
LD	
FBD	
CFC	

- **Supported Products**
  - AX series
- **Library**
  - None

18.7.12. TIME_TO_<TYPE> (Time Type Conversion Instruction)

TIME_TO_<TYPE> converts the time data to other data types. Time is stored internally as DWORD in milliseconds.

FB/FC	Instruction	Graphic Expression	ST Language
FC	TIME_TO_<TYPE>		Y := TIME_TO_INT(X);

- **Inputs**

Name	Function	Data Type	Setting Value
X	Input	TIME	T#0d0h0m0s0ms to T#49d17h2m47s295ms

- **Outputs**

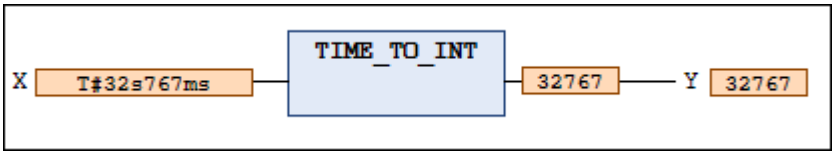
Name	Function	Data Type	Setting Value
Y	Converted value	Any data type	As each data type suggested.

- **Function**

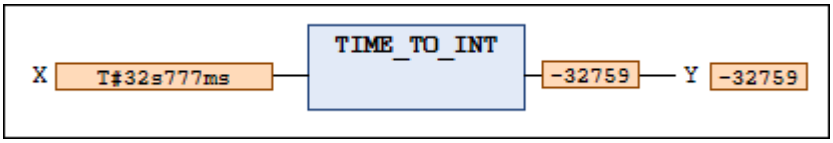
  This FC instruction is used to convert a TIME value into the specified data type and return a type–converted value.

- **Programming Example**

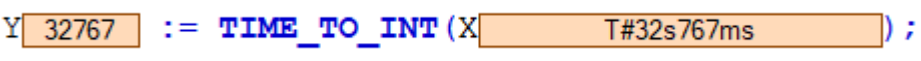
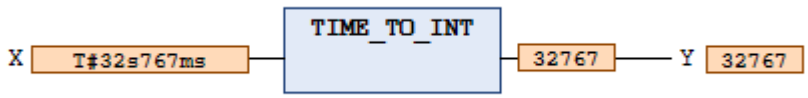
  The FC instruction (TIME_TO_<TYPE>) is used to change the value of variable *X* to INT type, where the lower limit and upper limit for data type TIME is from T#0d0h0m0s0ms to T#49d17h2m47s295ms, and the lower limit and upper limit for data type INT is from –32768 to 32767. For example, if input *X* is T#32s767ms, then output *Y* is 32767.

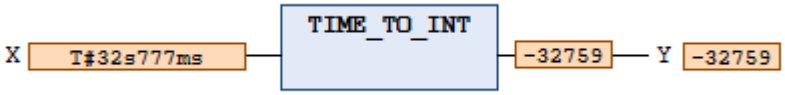
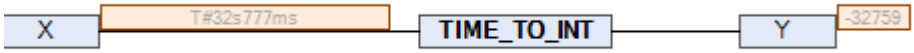


If input *X* is T#32s777ms, then output *Y* is –32759.



Feature in the ST/LD/FBD & CFC editor:

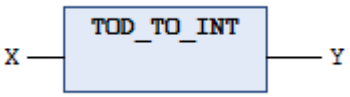
Programming Language	Program
ST	
LD	

Programming Language	Program
FBD	
CFC	

- **Supported Products**
  - AX series
- **Library**
  - None

18.7.13. TOD_TO_<TYPE> (Time Type Conversion Instruction)

TOD_TO_<TYPE> converts the time data type to other data types, and date is converted internally in milliseconds.

FB/FC	Instruction	Graphic Expression	ST Language
FC	TOD_TO_<TYPE>		Y := TOD_TO_INT(X);

- **Inputs**

Name	Function	Data Type	Setting Value
X	Input	TOD	TOD#0:0:0 to TOD#23:59:59:999

- **Outputs**

Name	Function	Data Type	Setting Value
Y	Converted value	Any data type	As each data type suggested.

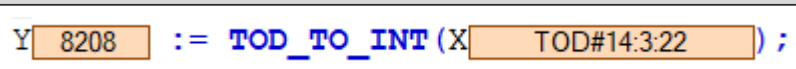
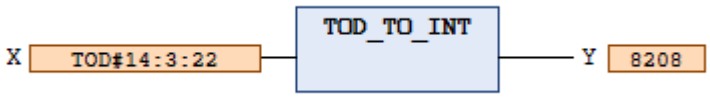
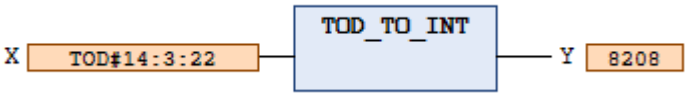
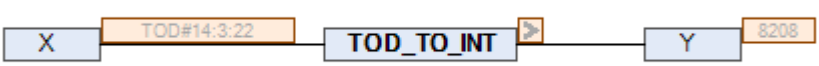
- **Function**

  This FC instruction is used to convert a TIME value into the specified data type and return a type–converted value.

- **Programming Example**

  The FC instruction (TOD_TO_<TYPE>) is used to change the value of variable *X* to INT type, where the lower limit and upper limit for data type TOD is from TOD#0:0:0 to TOD#23:59:59:999, and the lower limit and upper limit for data type INT is from –32768 to 32767. For example, if input *X* is TOD#14:3:22, then output *Y* is 8208.

  Feature in the ST/LD/FBD & CFC editor:

Programming Language	Program
ST	
LD	
FBD	
CFC	

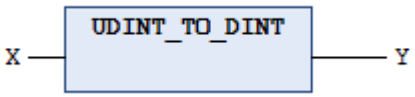
- **Supported Products**

  - AX series

- **Library**
  - None

18.7.14. UDINT_TO_<TYPE> (Unsigned Double Integer Type Conversion Instruction)

UDINT_TO_<TYPE> converts the Unsigned Double Integer data type to other data types.

FB/FC	Instruction	Graphic Expression	ST Language
FC	UDINT_TO_<TYPE>		Y := UDINT_TO_DINT(X);

- **Inputs**

Name	Function	Data Type	Setting Value
X	Input	UDINT	0 to 4294967295

- **Outputs**

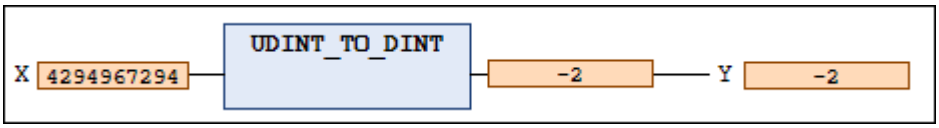
Name	Function	Data Type	Setting Value
Y	Converted value	Any data type	As each data type suggested.

- **Function**

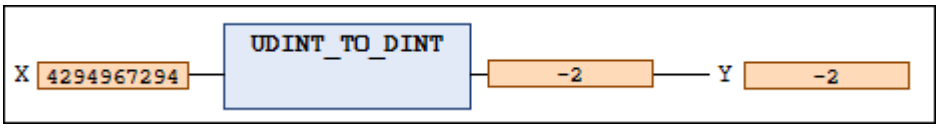
  This FC instruction is used to convert a UDINT value into the specified data type and return a type–converted value.

- **Programming Example**

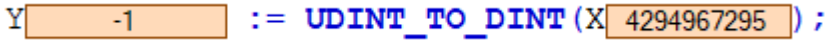
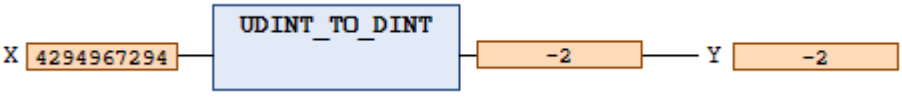
  The FC instruction (UDINT_TO_<TYPE>) is used to change the value of variable *X* to DINT type, where the lower limit and upper limit for data type UDINT is from 0 to 4294967295, and the lower limit and upper limit for data type DINT is from –2147483648 to 2147483647. For example, if input *X* is 4294967294, then output *Y* is –2.

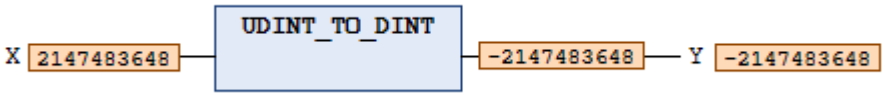
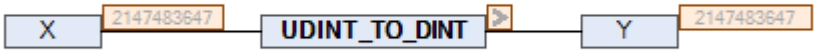


If input *X* is 2147483648, then output *Y* is –2147483648.



Feature in the ST/LD/FBD & CFC editor:

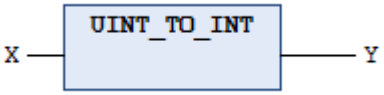
Programming Language	Program
ST	
LD	

Programming Language	Program
FBD	
CFC	

- **Supported Products**
  - AX series
- **Library**
  - None

18.7.15. UINT_TO_<TYPE> (Unsigned Integer Type Conversion Instruction)

UINT_TO_<TYPE> converts the Unsigned Integer data type to other data types.

FB/FC	Instruction	Graphic Expression	ST Language
FC	UINT_TO_<TYPE>		Y := UINT_TO_INT(X);

- **Inputs**

Name	Function	Data Type	Setting Value
X	Input	UDINT	0 to 4294967295

- **Outputs**

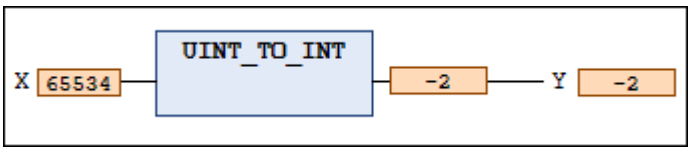
Name	Function	Data Type	Setting Value
Y	Converted value	Any data type	As each data type suggested.

- **Function**

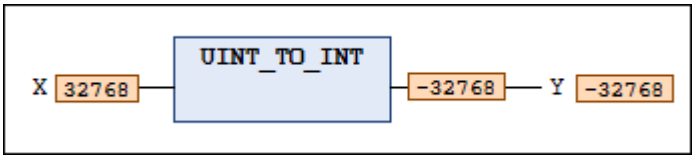
  This FC instruction is used to convert a UINT value into the specified data type and return a type–converted value.

- **Programming Example**

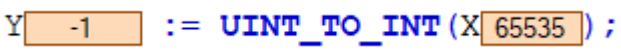
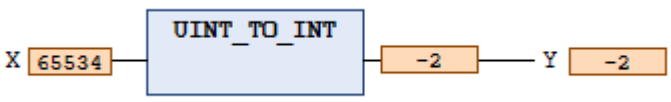
  The FC instruction (UINT_TO_<TYPE>) is used to change the value of variable X to INT type, where the lower limit and upper limit for data type UINT is from 0 to 65535, and the lower limit and upper limit for data type INT is from –32768 to 32767. For example, if input *X* is 65534, then output *Y* is –2.

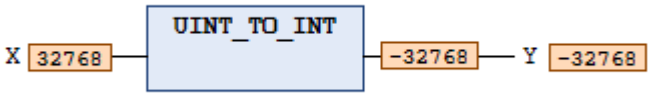
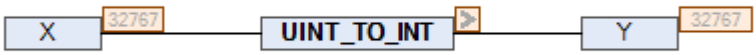


If input *X* is 32768, then output *Y* is –32768.



Feature in the ST/LD/FBD & CFC editor:

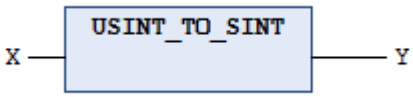
Programming Language	Program
ST	
LD	

Programming Language	Program
FBD	
CFC	

- **Supported Products**
  - AX series
- **Library**
  - None

18.7.16. USINT_TO_<TYPE> (Unsigned Short Integer Type Conversion Instruction)

USINT_TO_<TYPE> converts the Unsigned Short Integer data type to other data types.

FB/FC	Instruction	Graphic Expression	ST Language
FC	USINT_TO_<TYPE>		Y := USINT_TO_SINT(X);

- **Inputs**

Name	Function	Data Type	Setting Value
X	Input	USINT	0 to 255

- **Outputs**

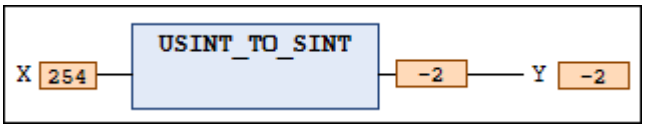
Name	Function	Data Type	Setting Value
Y	Converted value	Any data type	As each data type suggested.

- **Function**

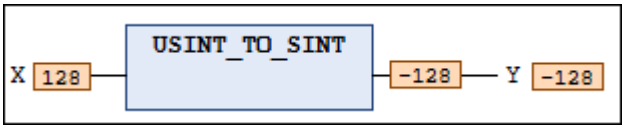
  This FC instruction is used to convert a USINT value into the specified data type and return a type–converted value.

- **Programming Example**

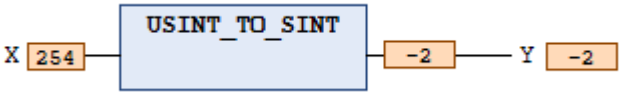
  The FC instruction (USINT_TO_<TYPE>) is used to change the value of variable *X* to SINT type, where the lower limit and upper limit for data type USINT is from 0 to 255, and the lower limit and upper limit for data type SINT is from –128 to 127. For example, if input *X* is 254, then output *Y* is –2.

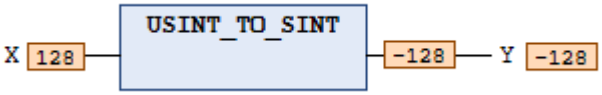
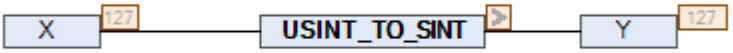


If input *X* is 128, then output *Y* is –128.



Feature in the ST/LD/FBD & CFC editor:

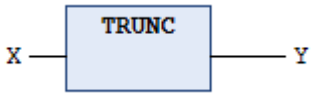
Programming Language	Program
ST	
LD	

Programming Language	Program
FBD	
CFC	

- **Supported Products**
  - AX series
- **Library**
  - None

18.7.17. TRUNC (Truncation Instruction)

TRUNC truncates the input data, leaving only the Integer.

FB/FC	Instruction	Graphic Expression	ST Language
FC	TRUNC		Y := TRUNC(X);

- **Inputs**

Name	Function	Data Type	Setting Value
X	Input	REAL	1.0E–44–3.402823E+38

- **Outputs**

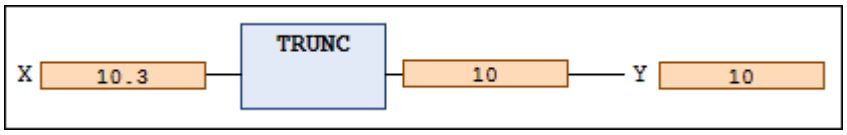
Name	Function	Data Type	Setting Value
Y	Converted value	Any data type	As each data type suggested.

- **Function**

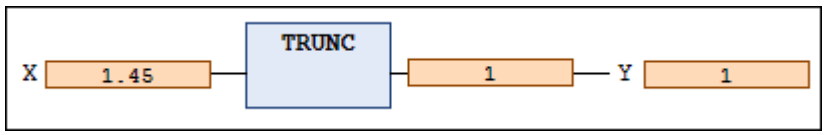
  This FC instruction is used to truncate the value of variable. It truncates the decimal part of the data and retains only the integer part.

- **Programming Example**

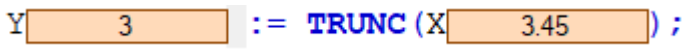
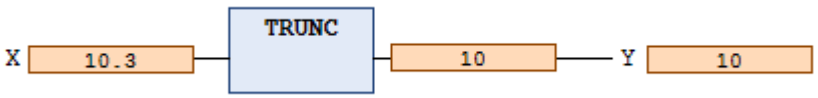
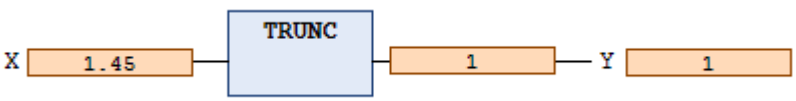
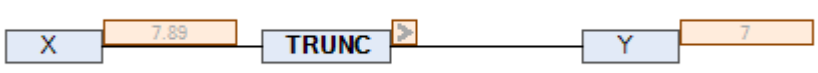
  The FC instruction (TRUNC) is used to truncate the value of variable *X*. For example, if the input value *X* is 10.3, then the output value *Y* is 10.



If the input value *X* is 1.45, then the output value *Y* is 1.



Feature in the ST/LD/FBD & CFC editor:

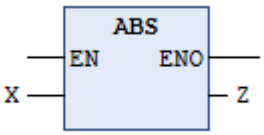
Programming Language	Program
ST	
LD	
FBD	
CFC	

- **Supported Products**
  - AX series
- **Library**
  - None

18.8. Elementary Math Operation Instructions

18.8.1. ABS (Absolute Value Instruction)

ABS allocates the absolute value of the input data to the output variable.

FB/FC	Instruction	Graphic Expression	ST Language
FC	ABS		Z := ABS(X);

- **Inputs**

Name	Function	Data Type	Setting Value
X	Input	BOOL, BYTE, WORD, DWORD, LWORD, USINT, UINT, UDINT, ULINT, SINT, INT, DINT, LINT, REAL, LREAL, DATE, TOD, DT, TIME	As each data type suggested.

- **Outputs**

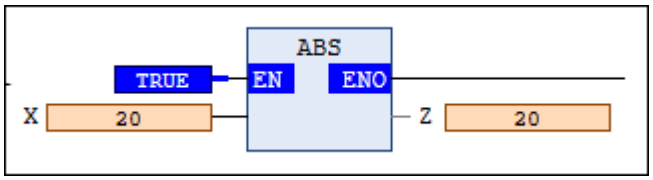
Name	Function	Data Type	Setting Value
Z	Absolute value of the input (always positive)	BOOL, BYTE, WORD, DWORD, LWORD, USINT, UINT, UDINT, ULINT, SINT, INT, DINT, LINT, REAL, LREAL, DATE, TOD, DT, TIME	As each data type suggested.

- **Function**

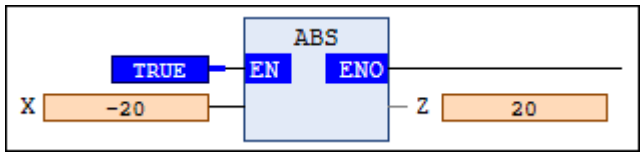
  This FC instruction is used to assign the absolute value of the input data to the output. The input value can be positive or negative, but the output value must be positive.

- **Programming Example**

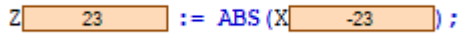
  The FC instruction (ABS) is used to return the absolute (positive) value of the input. For example, if the input value *X* is 20, then the output value *Z* is 20.

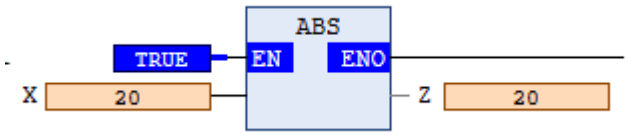
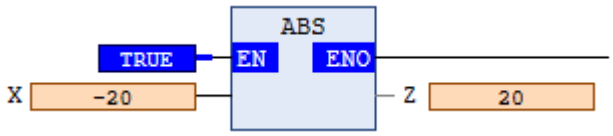
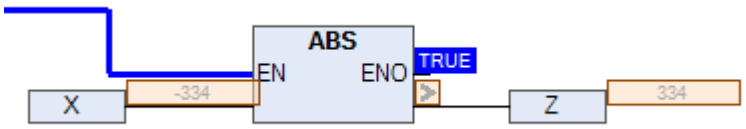


If the input value *X* is –20, then the output value *Z* is 20.



Feature in the ST/LD/FBD & CFC editor:

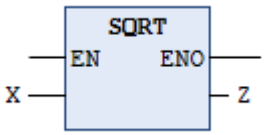
Programming Language	Program
ST	

Programming Language	Program
LD	
FBD	
CFC	

- **Supported Products**
  - AX series
- **Library**
  - None

18.8.2. SQRT (Square Root Instruction)

SQRT returns the square root of the input value as output.

FB/FC	Instruction	Graphic Expression	ST Language
FC	SQRT		Z := SQRT(X);

- **Inputs**

Name	Function	Data Type	Setting Value
X	Input	BYTE, WORD, DWORD, LWORD, USINT, UINT, UDINT, ULINT, SINT, INT, DINT, LINT, LREAL, REAL	As each data type suggested.

- **Outputs**

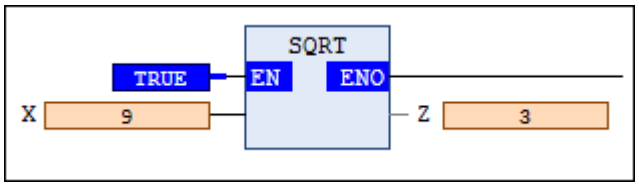
Name	Function	Data Type	Setting Value
Z	Square root of the input value	REAL and LREAL	As each data type suggested.

- **Function**

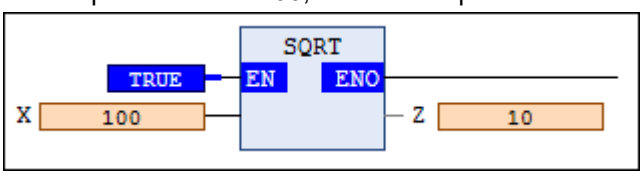
  This FC instruction is used to calculate the square root of the input value as output. Input value must be positive.

- **Programming Example**

  The FC instruction (SQRT) is used to calculate the square root of input value as output. For example, if the input value *X* is 9, then the output value *Z* is 3.

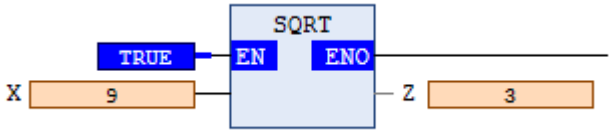
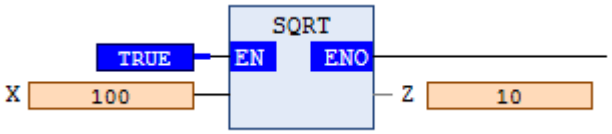
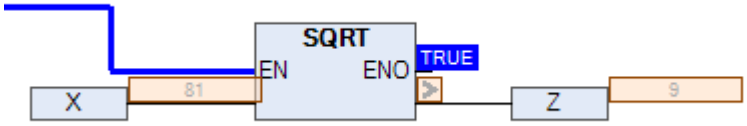


If the input value *X* is 100, then the output value *Z* is 10.



Feature in the ST/LD/FBD & CFC editor:

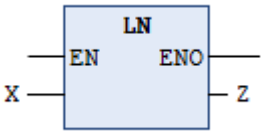
Programming Language	Program
ST	Z 4 := SQRT(X 16);

Programming Language	Program
LD	
FBD	
CFC	

- **Supported Products**
  - AX series
- **Library**
  - None

18.8.3. LN (Natural Logarithm Instruction)

LN returns the natural logarithm of the input value as output.

FB/FC	Instruction	Graphic Expression	ST Language
FC	LN		Z := LN(X);

- **Inputs**

Name	Function	Data Type	Setting Value
X	Input	BYTE, WORD, DWORD, LWORD, USINT, UINT, UDINT, ULINT, SINT, INT, DINT, LINT, LREAL, REAL	As each data type suggested.

- **Outputs**

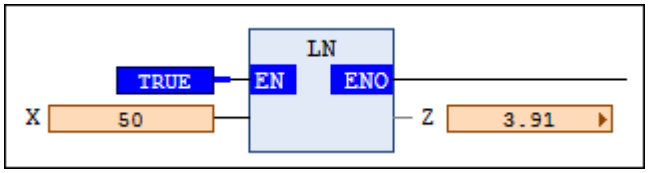
Name	Function	Data Type	Setting Value
Z	Natural logarithm of the input value	REAL and LREAL	As each data type suggested.

- **Function**

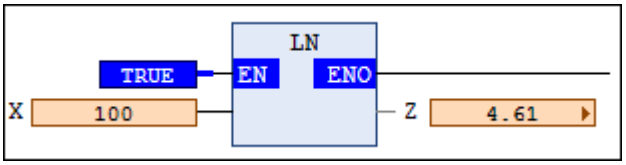
  This FC instruction is used to calculate the natural logarithm of the input value which must be an integer.

- **Programming Example**

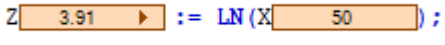
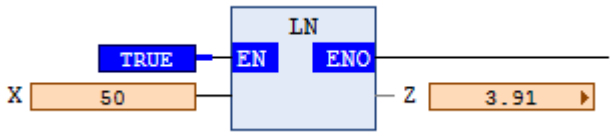
  The FC instruction (LN) is used to calculate the natural logarithm of the input value. For example, if the input value *X* is 50, then the output value *Z* is 3.91.

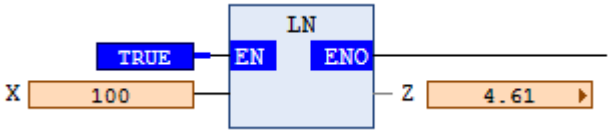
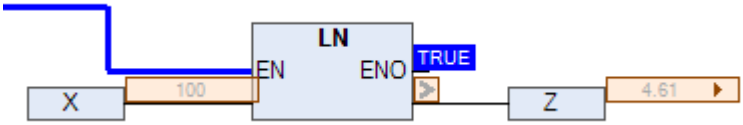


If the input value *X* is 100, then the output value *Z* is 4.61.



Feature in the ST/LD/FBD & CFC editor:

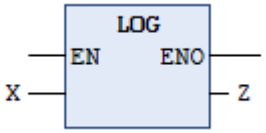
Programming Language	Program
ST	
LD	

Programming Language	Program
FBD	
CFC	

- **Supported Products**
  - AX series
- **Library**
  - None

18.8.4. LOG (Commonly Used Logarithm Instruction)

LOG returns the commonly used logarithm (base 10) of the input value as output.

FB/FC	Instruction	Graphic Expression	ST Language
FC	LOG		Z := LOG(X);

- **Inputs**

Name	Function	Data Type	Setting Value
X	Input	BYTE, WORD, DWORD, LWORD, USINT, UINT, UDINT, ULINT, SINT, INT, DINT, LINT, LREAL, REAL	As each data type suggested.

- **Outputs**

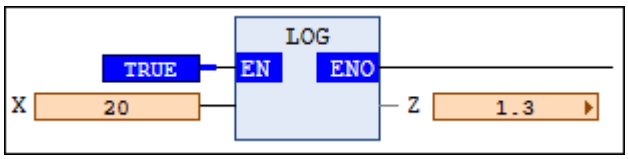
Name	Function	Data Type	Setting Value
Z	Logarithm (base 10) of the input value	REAL and LREAL	As each data type suggested.

- **Function**

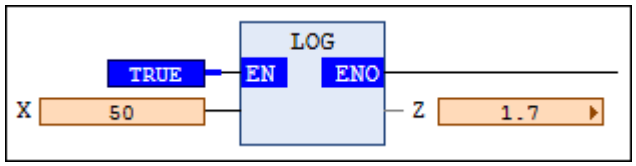
  This FC instruction is used to calculate the commonly used logarithm of the input value with 10 as the base number, and the input value should be a positive.

- **Programming Example**

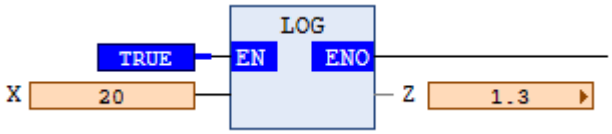
  The FC instruction (LOG) is used to calculate the commonly used logarithm of the input value. For example, If the input value *X* is 20, then the output value *Z* is 1.3.

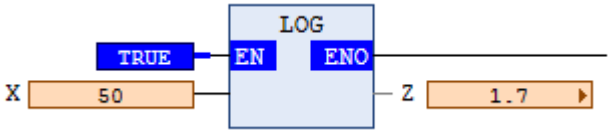
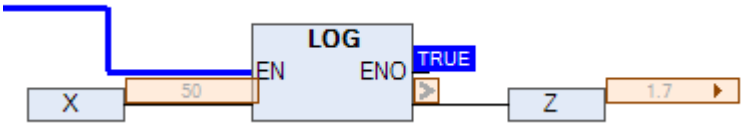


If the input value *X* is 50, then the output value *Z* is 1.7.



Feature in the ST/LD/FBD & CFC editor:

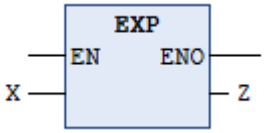
Programming Language	Program
ST	
LD	

Programming Language	Program
FBD	
CFC	

- **Supported Products**
  - AX series
- **Library**
  - None

18.8.5. EXP (Exponent Instruction)

EXP performs a power calculation with the input value as an exponent and e as base.

FB/FC	Instruction	Graphic Expression	ST Language
FC	EXP		Z := EXP(X);

- **Inputs**

Name	Function	Data Type	Setting Value
X	Input	BYTE, WORD, DWORD, LWORD, USINT, UINT, UDINT, ULINT, SINT, INT, DINT, LINT, LREAL, REAL	As each data type suggested.

- **Outputs**

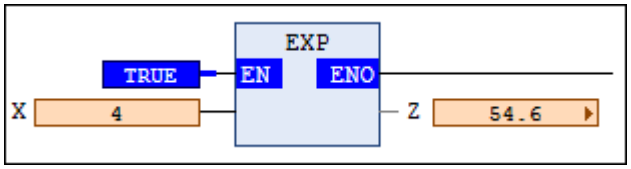
Name	Function	Data Type	Setting Value
Z	Power	REAL and LREAL	As each data type suggested.

- **Function**

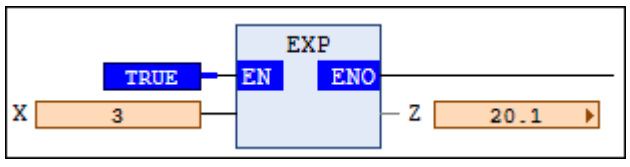
  This FC instruction is used to perform a power calculation with the input value as an exponent and e as base, $Z = e^{X}$.

- **Programming Example**

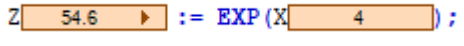
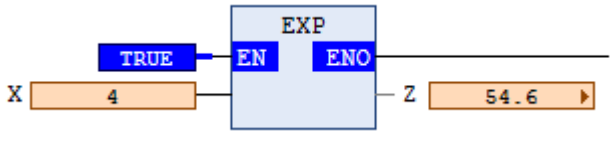
  The FC instruction (EXP) yields the exponential function. For example, if the input value *X* is 4, then the output value *Z* is 54.6.

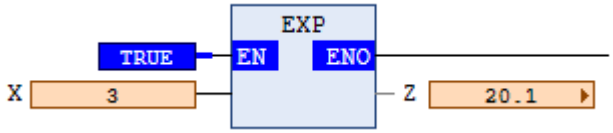
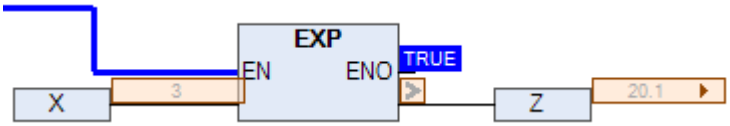


If the input value *X* is 3, then the output value *Z* is 20.1.



Feature in the ST/LD/FBD & CFC editor:

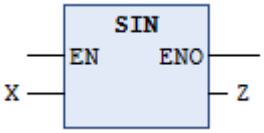
Programming Language	Program
ST	
LD	

Programming Language	Program
FBD	
CFC	

- **Supported Products**
  - AX series
- **Library**
  - None

18.8.6. SIN (Sine Instruction)

SIN returns the sine value of the input value as output.

FB/FC	Instruction	Graphic Expression	ST Language
FC	SIN		Z := SIN(X);

- **Inputs**

Name	Function	Data Type	Setting Value
X	Input (In radians)	BYTE, WORD, DWORD, LWORD, USINT, UINT, UDINT, ULINT, SINT, INT, DINT, LINT, LREAL, REAL	-2^{63} to $+2^{63}$

- **Outputs**

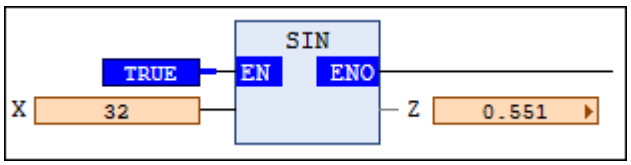
Name	Function	Data Type	Setting Value
Z	Sine of the input value	REAL and LREAL	As each data type suggested.

- **Function**

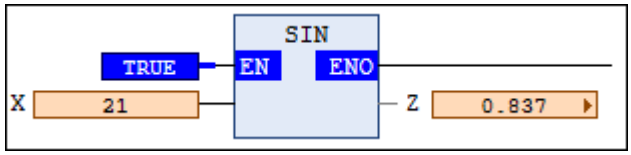
  This FC instruction is used to calculate the sine value of variable *X* as output.

- **Programming Example**

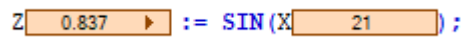
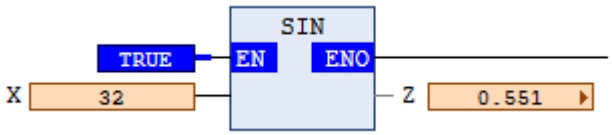
  The FC instruction (SIN) is used to calculate the sine value. For example, if the input value *X* is 32, then the output value *Z* is 0.551.

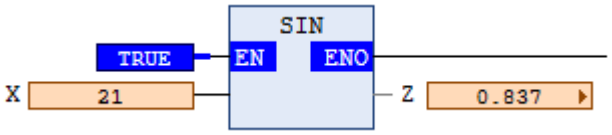
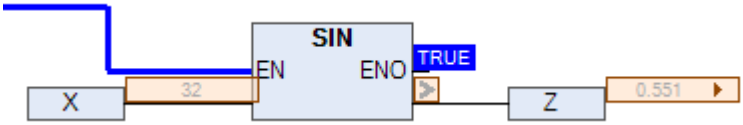


If the input value *X* is 21, then the output value *Z* is 0.837.



Feature in the ST/LD/FBD & CFC editor:

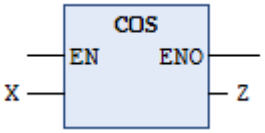
Programming Language	Program
ST	
LD	

Programming Language	Program
FBD	
CFC	

- **Supported Products**
  - AX series
- **Library**
  - None

18.8.7. COS (Cosine Instruction)

COS returns the cosine value of the input value as output.

FB/FC	Instruction	Graphic Expression	ST Language
FC	COS		Z := COS(X);

- **Inputs**

Name	Function	Data Type	Setting Value
X	Input (In radians)	BYTE, WORD, DWORD, LWORD, USINT, UINT, UDINT, ULINT, SINT, INT, DINT, LINT, LREAL, REAL	-2^{63} to $+2^{63}$

- **Outputs**

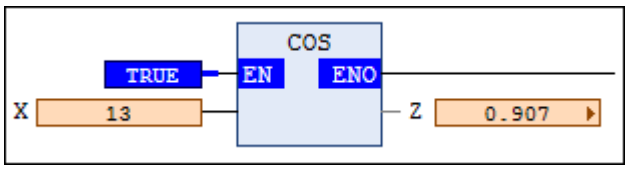
Name	Function	Data Type	Setting Value
Z	Cosine of the input value	REAL and LREAL	As each data type suggested.

- **Function**

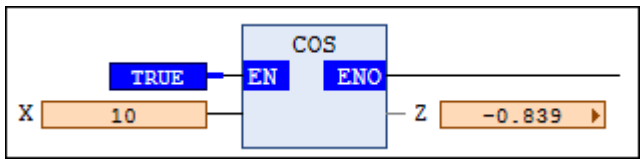
  This FC instruction is used to calculate the cosine value of variable *X* as output.

- **Programming Example**

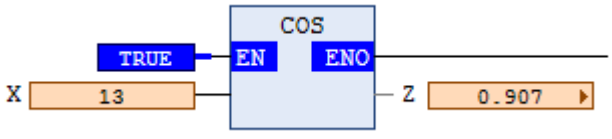
  The FC instruction (COS) is used to calculate the cosine value. For example, if the input value *X* is 13, then the output value *Z* is 0.907.

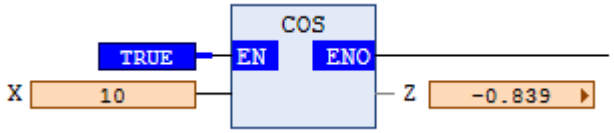
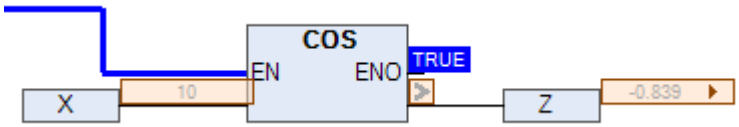


If the input value *X* is 10, then the output value *Z* is –0.839.



Feature in the ST/LD/FBD & CFC editor:

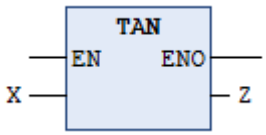
Programming Language	Program
ST	
LD	

Programming Language	Program
FBD	
CFC	

- **Supported Products**
  - AX series
- **Library**
  - None

18.8.8. TAN (Tangent Instruction)

TAN returns the tangent value of the input value as output.

FB/FC	Instruction	Graphic Expression	ST Language
FC	TAN		Z := TAN(X);

- **Inputs**

Name	Function	Data Type	Setting Value
X	Input (In radians)	BYTE, WORD, DWORD, LWORD, USINT, UINT, UDINT, ULINT, SINT, INT, DINT, LINT, LREAL, REAL	-2^{63} to $+2^{63}$

- **Outputs**

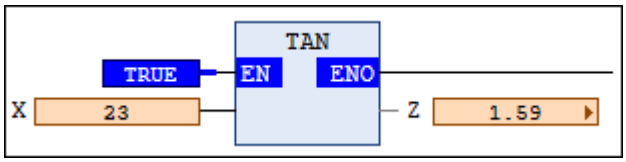
Name	Function	Data Type	Setting Value
Z	Tangent of the input value	REAL and LREAL	As each data type suggested.

- **Function**

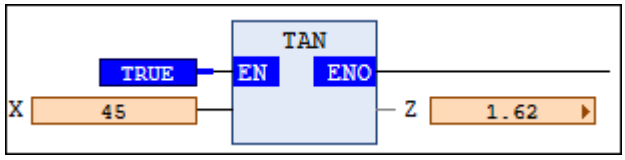
  This FC instruction is used to calculate the tangent value of variable *X* as output.

- **Programming Example**

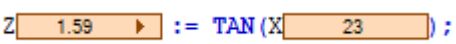
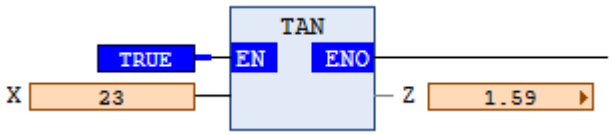
  The FC instruction (TAN) is used to calculate the tangent value. For example, if the input value *X* is 23, then the output value *Z* is 1.59.

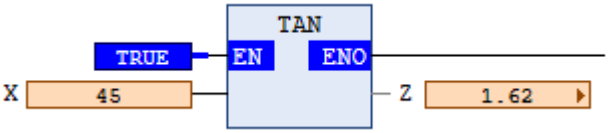
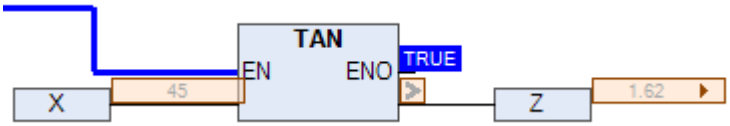


If the input value *X* is 45, then the output value *Z* is 1.62.



Feature in the ST/LD/FBD & CFC editor:

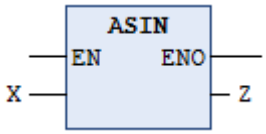
Programming Language	Program
ST	
LD	

Programming Language	Program
FBD	
CFC	

- **Supported Products**
  - AX series
- **Library**
  - None

18.8.9. ASIN (Arcsine Instruction)

ASIN returns the arcsine value of the input value as output.

FB/FC	Instruction	Graphic Expression	ST Language
FC	ASIN		Z := ASIN(X);

- **Inputs**

Name	Function	Data Type	Setting Value
X	Input (In radians)	BYTE, WORD, DWORD, LWORD, USINT, UINT, UDINT, ULINT, SINT, INT, DINT, LINT, LREAL, REAL	As each data type suggested.

- **Outputs**

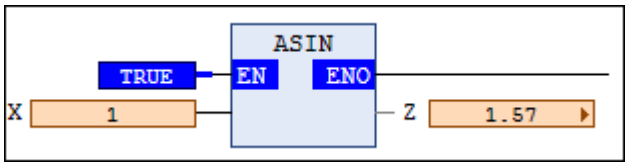
Name	Function	Data Type	Setting Value
Z	Arcsine of the input value (In radians)	REAL and LREAL	As each data type suggested.

- **Function**

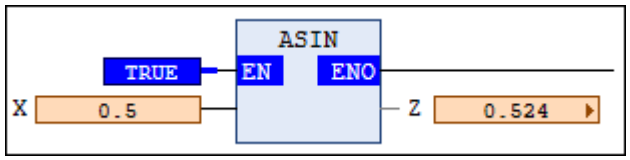
  This FC instruction is used to calculate the arcsine value of variable *X* as output.

- **Programming Example**

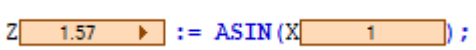
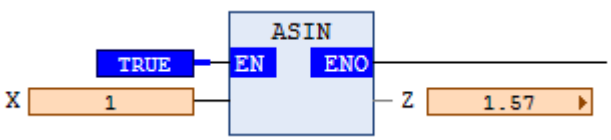
  The FC instruction (ASIN) is used to calculate the arcsine value. For example, if the input value *X* is 1, then the output value *Z* is 1.57.

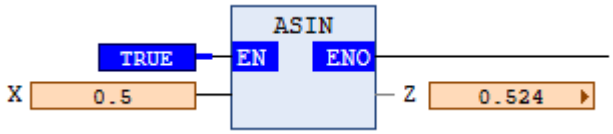
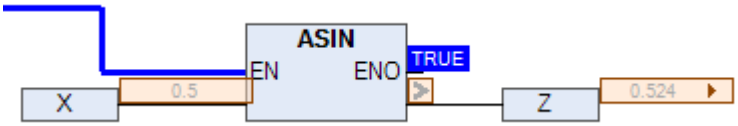


If the input value *X* is 0.5, then the output value *Z* is 0.524.



Feature in the ST/LD/FBD & CFC editor:

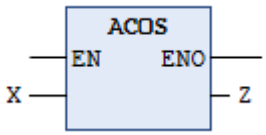
Programming Language	Program
ST	
LD	

Programming Language	Program
FBD	
CFC	

- **Supported Products**
  - AX series
- **Library**
  - None

18.8.10. ACOS (Arccosine Instruction)

ACOS returns the arccosine value of the input value as output.

FB/FC	Instruction	Graphic Expression	ST Language
FC	ACOS		Z := ACOS(X);

- **Inputs**

Name	Function	Data Type	Setting Value
X	Input (In radians)	BYTE, WORD, DWORD, LWORD, USINT, UINT, UDINT, ULINT, SINT, INT, DINT, LINT, LREAL, REAL	As each data type suggested.

- **Outputs**

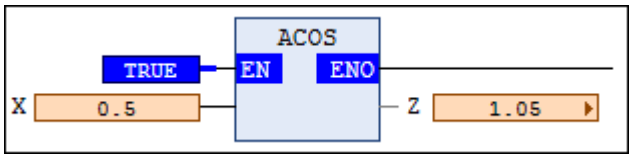
Name	Function	Data Type	Setting Value
Z	Arccosine of the input value (In radians)	REAL and LREAL	As each data type suggested.

- **Function**

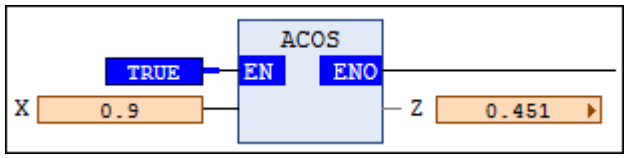
  This FC instruction is used to calculate the arccosine value of variable *X* as output.

- **Programming Example**

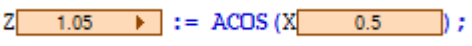
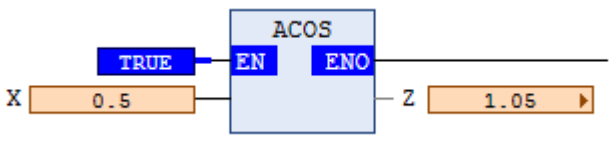
  The FC instruction (ACOS) is used to calculate the arccosine value. For example, if the input value *X* is 0.5, then the output value *Z* is 1.05.

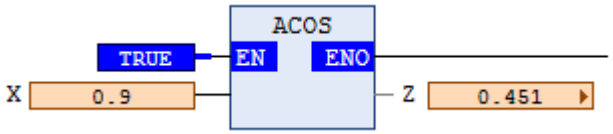
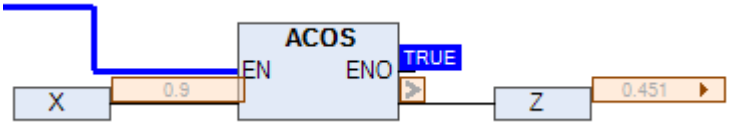


If the input value *X* is 0.9, then the output value *Z* is 0.451.



Feature in the ST/LD/FBD & CFC editor:

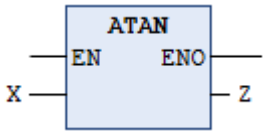
Programming Language	Program
ST	
LD	

Programming Language	Program
FBD	
CFC	

- **Supported Products**
  - AX series
- **Library**
  - None

18.8.11. ATAN (Arctangent Instruction)

ATAN returns the arctangent value of the input value as output.

FB/FC	Instruction	Graphic Expression	ST Language
FC	ATAN		Z := ATAN(X);

- **Inputs**

Name	Function	Data Type	Setting Value
X	Input	BYTE, WORD, DWORD, LWORD, USINT, UINT, UDINT, ULINT, SINT, INT, DINT, LINT, LREAL, REAL	As each data type suggested.

- **Outputs**

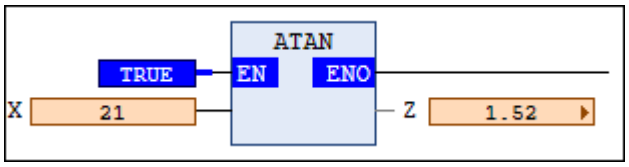
Name	Function	Data Type	Setting Value
Z	Arctangent of the input value (In radians)	REAL and LREAL	As each data type suggested.

- **Function**

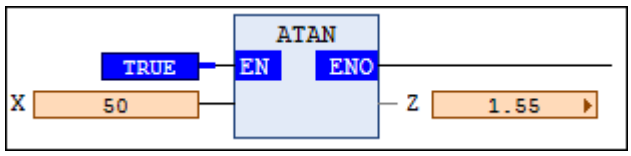
  This FC instruction is used to calculate the arctangent value of variable *X* as output.

- **Programming Example**

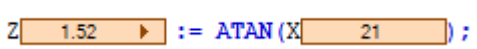
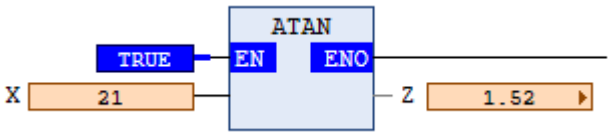
  The FC instruction (ATAN) is used to calculate the arctangent value. For example, if the input value *X* is 21, then the output value *Z* is 1.52.

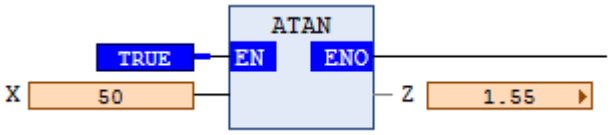
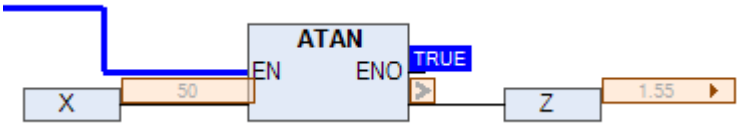


If the input value *X* is 50, then the output value *Z* is 1.55.



Feature in the ST/LD/FBD & CFC editor:

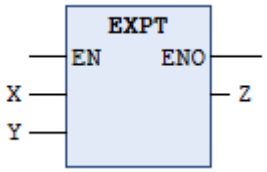
Programming Language	Program
ST	
LD	

Programming Language	Program
FBD	
CFC	

- **Supported Products**
  - AX series
- **Library**
  - None

18.8.12. EXPT (Power Instruction)

EXPT raises a number to a higher power and returns the power of the base raised to the exponent: $Z = X^Y$. The input *X* is the base and *Y* is the exponent.

FB/FC	Instruction	Graphic Expression	ST Language
FC	EXPT		Z := EXPT(X,Y);

- **Inputs**

Name	Function	Data Type	Setting Value
Y	Exponent	BYTE, WORD, DWORD, LWORD, USINT, UINT, UDINT, ULINT, SINT, INT, DINT, LINT, LREAL, REAL	As each data type suggested.
X	Base		

- **Outputs**

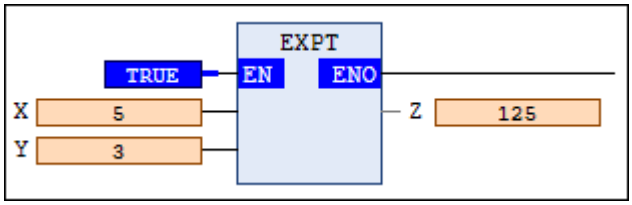
Name	Function	Data Type	Setting Value
Z	Power	REAL and LREAL	As each data type suggested.

- **Function**

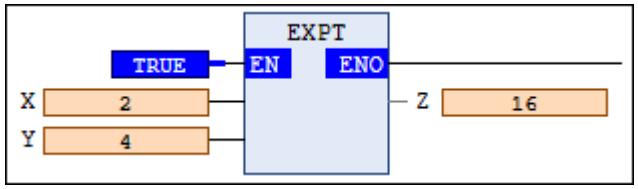
  This FC instruction is used to raise the value of *X* (base) to the power of *Y* (exponent) and outputs the result of the operation.

- **Programming Example**

  The FC instruction (EXPT) is used to exponentiate the input value. For example, if the input value *X* is 5 and *Y* is 3, then the output value *Z* is 125.

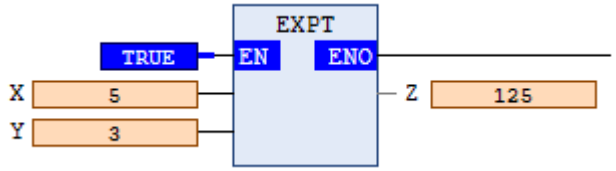
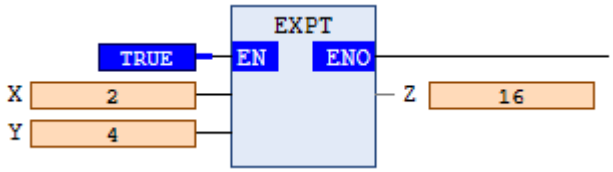
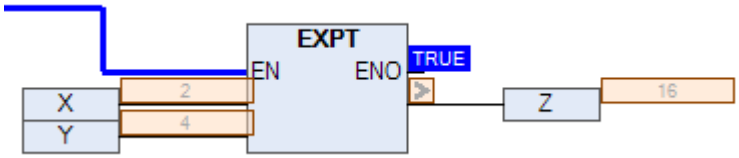


If the input value *X* is 2 and *Y* is 4, then the output value *Z* is 16.



Feature in the ST/LD/FBD & CFC editor:

Programming Language	Program
ST	Z 125 := EXPT(X 5 , Y 3);

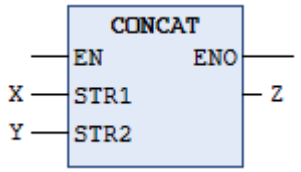
Programming Language	Program
LD	
FBD	
CFC	

- **Supported Products**
  - AX series
- **Library**
  - None

18.9. String Instructions

18.9.1. CONCAT (Merge String Instructions)

CONCAT merges two strings into one new string.

FB/FC	Instruction	Graphic Expression	ST Language
FC	CONCAT		Z := CONCAT(X,Y);

- **Inputs**

Name	Function	Data Type	Setting Value
STR1,STR2	Input	STRING	As each data type suggested.

- **Outputs**

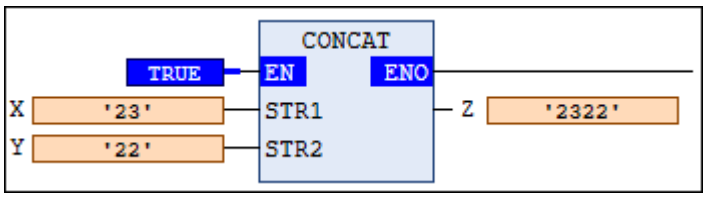
Name	Function	Data Type	Setting Value
Z	Merged string	STRING	As each data type suggested.

- **Function**

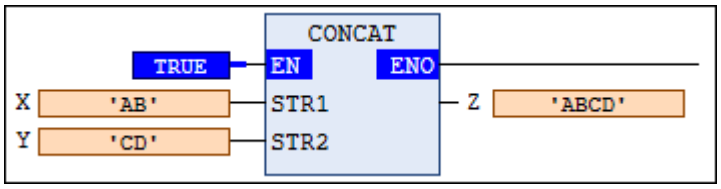
  This FC instruction is used to merge two string inputs sequentially into one output string.

- **Programming Example**

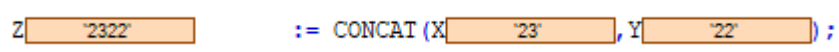
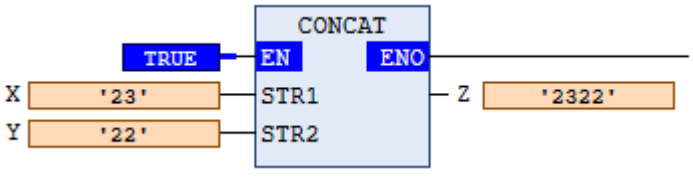
  The FC instruction (CONCAT) is used to merge two string inputs into one string output. For example, if the input value *X* is 23 and *Y* is 22, then the output value *Z* is 2322.

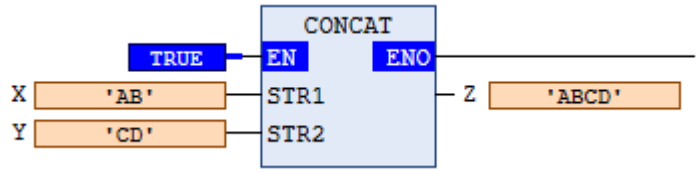
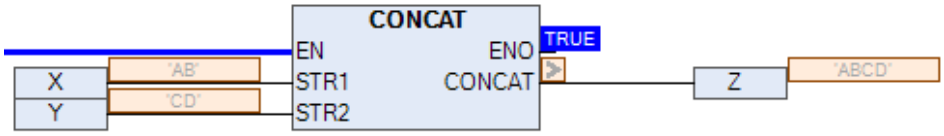


  If the input value *X* is AB and *Y* is CD, then the output value *Z* is ABCD.



  Feature in the ST/LD/FBD & CFC editor:

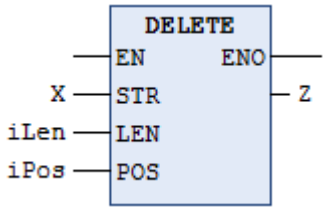
Programming Language	Program
ST	
LD	

Programming Language	Program
FBD	
CFC	

- **Supported Products**
  - AX series
- **Library**
  - Standard.lib

18.9.2. DELETE (Delete Characters Instruction)

DELETE deletes characters from a string.

FB/FC	Instruction	Graphic Expression	ST Language
FC	DELETE		Z := DELETE(X, iLen, iPos);

- **Inputs**

Name	Function	Data Type	Setting Value
STR	Input	STRING	As each data type suggested.
LEN	Number of characters to be deleted	INT	–32768 to 32767
POS	Position of the first deleted character		

- **Outputs**

Name	Function	Data Type	Setting Value
Z	Modified string	STRING	As each data type suggested.

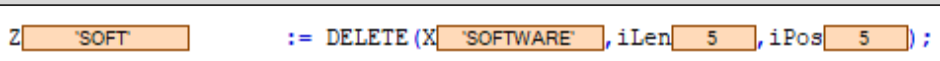
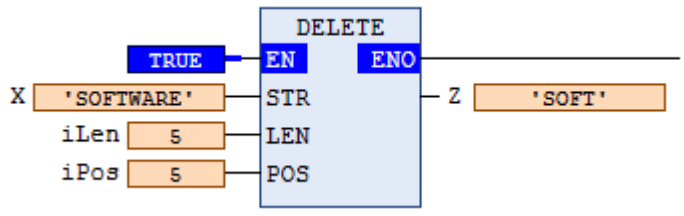
- **Function**

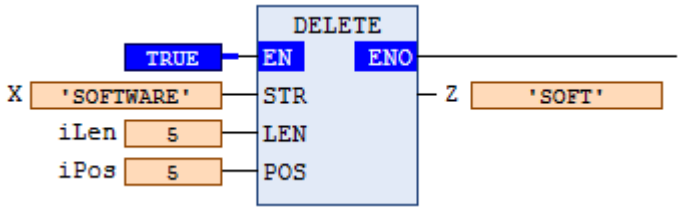
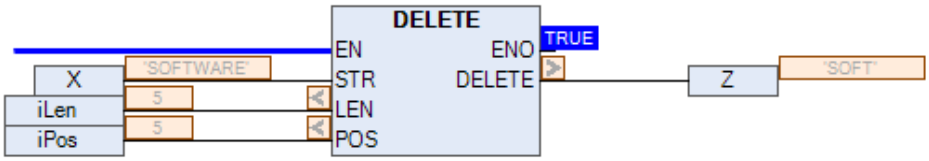
  This FC instruction is used to delete characters from *POS* as the starting position to the right and delete *LEN* lengths.

- **Programming Example**

  The FC instruction (DELETE) is used to delete the character from a string. For example, if the input value *X* is SOFTWARE, *iLen* is 5 and *iPos* is 5, then the output value *Z* is SOFT.

  Feature in the ST/LD/FBD & CFC editor:

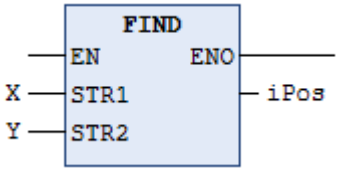
Programming Language	Program
ST	
LD	

Programming Language	Program
FBD	
CFC	

- **Supported Products**
  - AX series
- **Library**
  - Standard.lib

18.9.3. FIND (Find String Instruction)

FIND locates and provides the position of sub-strings within strings.

FB/FC	Instruction	Graphic Expression	ST Language
FC	FIND		iPos := FIND(X, Y);

- **Inputs**

Name	Function	Data Type	Setting Value
STR1,STR2	Input	STRING	As each data type suggested.

- **Outputs**

Name	Function	Data Type	Setting Value
iPos	The position of the first character of the first occurrence of the sub-string *STR2*	INT	–32768 to 32767

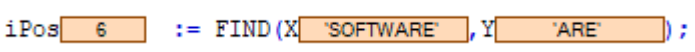
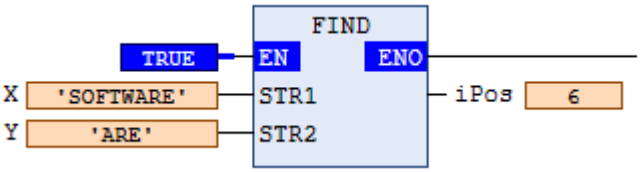
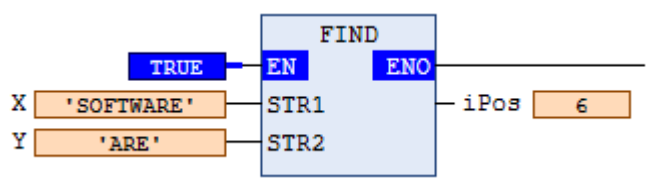
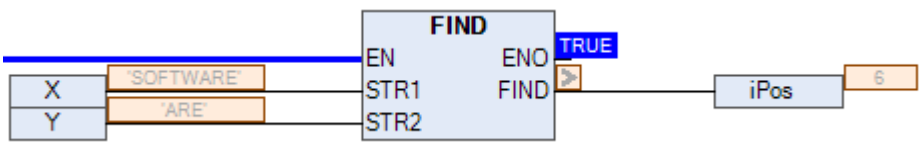
- **Function**

  This FC instruction is used to find the exact same part of the string *STR2* in the string *STR1* and returns the starting position of the same part in string *STR1*. If there are no same parts, the output value is 0.

- **Programming Example**

  The FC instruction (FIND) is used to find the content from input. For example, if the input value *X* is SOFTWARE and *Y* is ARE, then the output value *iPos* is 6.

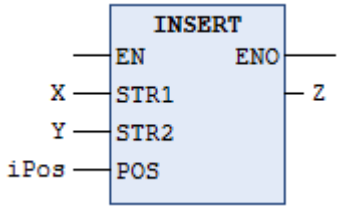
  Feature in the ST/LD/FBD & CFC editor:

Programming Language	Program
ST	
LD	
FBD	
CFC	

- **Supported Products**
  - AX series
- **Library**
  - Standard.lib

18.9.4. INSERT (Insert String Instruction)

INSERT inserts a string at the user–defined position within a string.

FB/FC	Instruction	Graphic Expression	ST Language
FC	INSERT		Z := INSERT(X, Y, iPos);

- **Inputs**

Name	Function	Data Type	Setting Value
STR1	Initial string	STRING	As each data type suggested.
STR2	String to be inserted		
POS	Position of the insertion	INT	–32768 to 32767

- **Outputs**

Name	Function	Data Type	Setting Value
Z	Modified string	STRING	As each data type suggested.

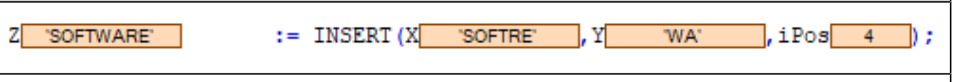
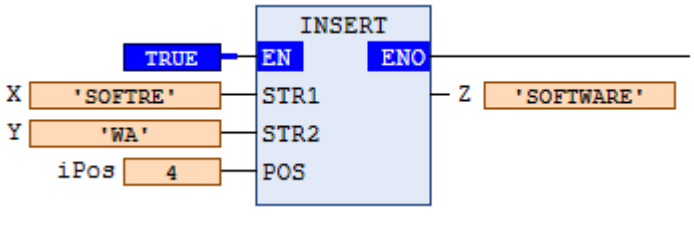
- **Function**

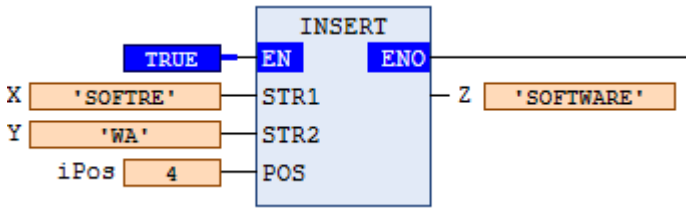
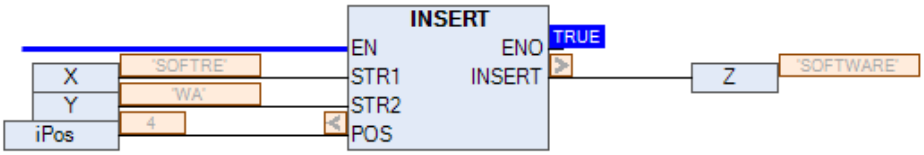
  This FC instruction is used to insert *STR2* to *STR1* after the *POS* position. *POS* is used to find the position where the string needs to be inserted.

- **Programming Example**

  The FC instruction (INSERT) is used to insert one input into another input. For example, if the input value *X* is SOFTRE, *Y* is WA, and *iPos* is 4, then the output value *Z* is SOFTWARE.

  Feature in the ST/LD/FBD & CFC editor:

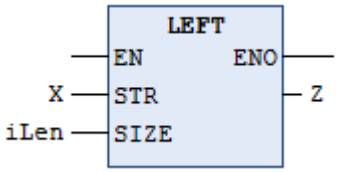
Programming Language	Program
ST	
LD	

Programming Language	Program
FBD	
CFC	

- **Supported Products**
  - AX series
- **Library**
  - Standard.lib

18.9.5. LEFT (Extract String from Left Instruction)

LEFT extracts the content from the left side of a string.

FB/FC	Instruction	Graphic Expression	ST Language
FC	LEFT		Z := LEFT(X, iLen);

- **Inputs**

Name	Function	Data Type	Setting Value
STR	Input	STRING	As each data type suggested.
SIZE	Number of characters to be extracted	INT	–32768 to 32767

- **Outputs**

Name	Function	Data Type	Setting Value
Z	Left part of the string (its length = *SIZE*)	STRING	As each data type suggested.

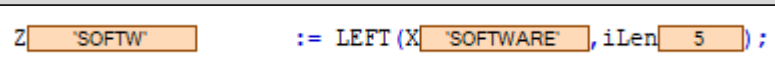
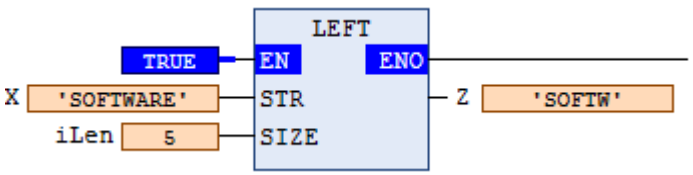
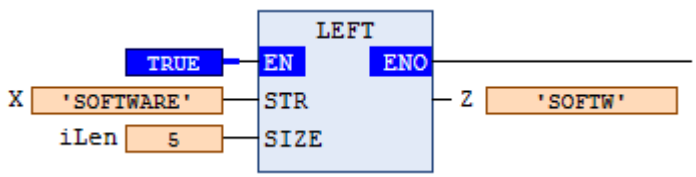
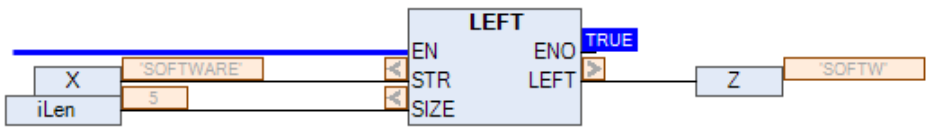
- **Function**

  This FC instruction is used to extract the content from the left side of a string with *SIZE* indicating the length of the content to be extracted.

- **Programming Example**

  The FC instruction (LEFT) is used to extract the content from the left side of a string. For example, if input value *X* is SOFTWARE and *iLen* is 5, then output value *Z* is SOFTW.

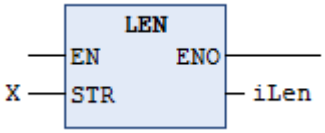
  Feature in the ST/LD/FBD & CFC editor:

Programming Language	Program
ST	
LD	
FBD	
CFC	

- **Supported Products**
  - AX series
- **Library**
  - Standard.lib

18.9.6. LEN (Extract String Length Instruction)

LEN calculates the length of a string.

FB/FC	Instruction	Graphic Expression	ST Language
FC	LEN		iLen := LEN(X);

- **Inputs**

Name	Function	Data Type	Setting Value
STR	Input	STRING	As each data type suggested.

- **Outputs**

Name	Function	Data Type	Setting Value
iLen	Number of characters in the string	INT	–32768 to 32767

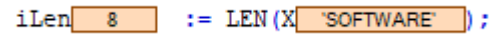
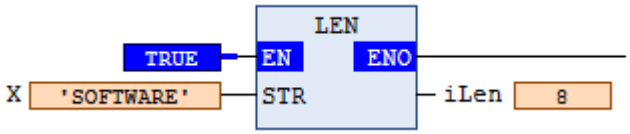
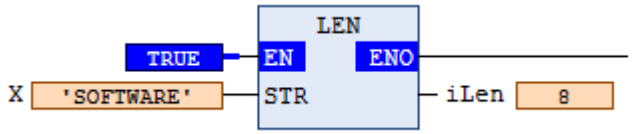
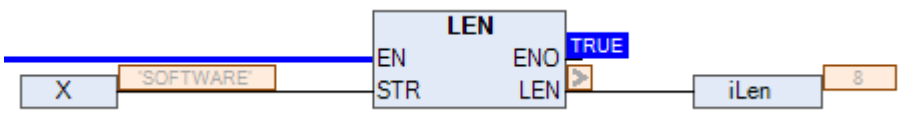
- **Function**

  This FC instruction is used to calculate the length of the input string as output.

- **Programming Example**

  The FC instruction (LEN) is used to calculate the length of the input string. For example, if the input value *X* is SOFTWARE, then the output value *iLen* is 8.

  Feature in the ST/LD/FBD & CFC editor:

Programming Language	Program
ST	
LD	
FBD	
CFC	

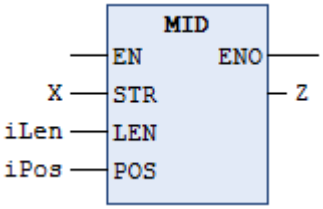
- **Supported Products**

  - AX series

- **Library**
  - Standard.lib

18.9.7. MID (Extract Middle of a String Instruction)

MID extracts characters from a string at the specified position.

FB/FC	Instruction	Graphic Expression	ST Language
FC	MID		Z := MID(X, iLen, iPos);

- **Inputs**

Name	Function	Data Type	Setting Value
STR	Input	STRING	As each data type suggested.
LEN	Number of characters to be extracted	INT	–32768 to 32767
POS	Position of the sub-string		

- **Outputs**

Name	Function	Data Type	Setting Value
Z	Middle part of the string (its length = *LEN*)	STRING	As each data type suggested.

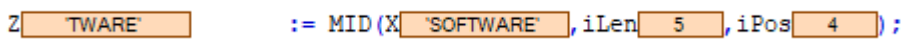
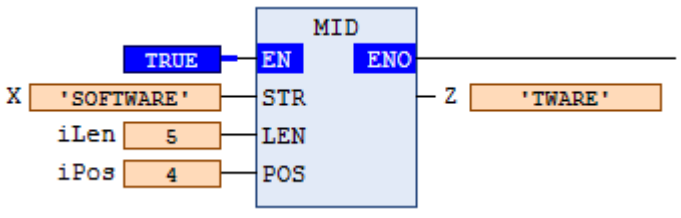
- **Function**

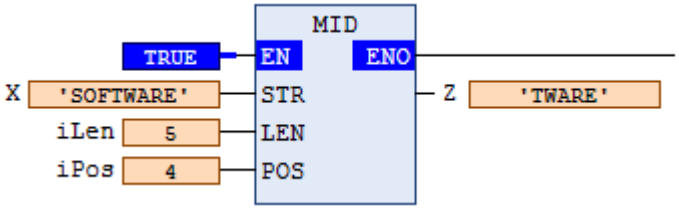
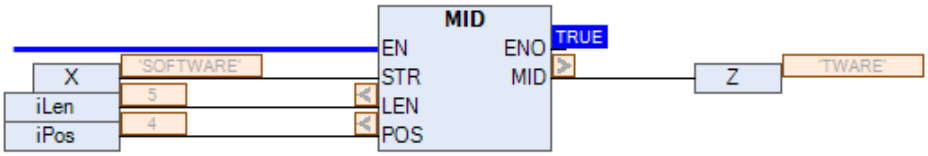
  This FC instruction is used to extract the content from the middle of a string with *LEN* indicating the length of the content and *POS* indicating the position of the content.

- **Programming Example**

  The FC instruction (MID) is used to extract the content from the middle of a string. For example, if the input value *X* is SOFTWARE, *iLen* is 5, and *iPos* is 4, then the output value *Z* is TWARE.

  Feature in the ST/LD/FBD & CFC editor:

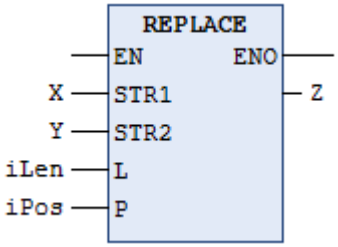
Programming Language	Program
ST	
LD	

Programming Language	Program
FBD	
CFC	

- **Supported Products**
  - AX series
- **Library**
  - Standard.lib

18.9.8. REPLACE (Replace String Instruction)

REPLACE replaces a partial string from a larger string with a third string.

FB/FC	Instruction	Graphic Expression	ST Language
FC	REPLACE		Z := REPLACE(X, Y, iLen, iPos);

- **Inputs**

Name	Function	Data Type	Setting Value
STR1	Input	STRING	As each data type suggested.
STR2	The replacement string		
L	Number of characters to be replaced, counting from left	INT	–32768 to 32767
P	Position of the first modified character		

- **Outputs**

Name	Function	Data Type	Setting Value
Z	Modified string	STRING	As each data type suggested.

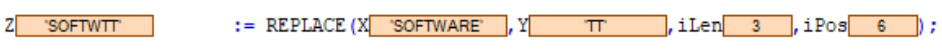
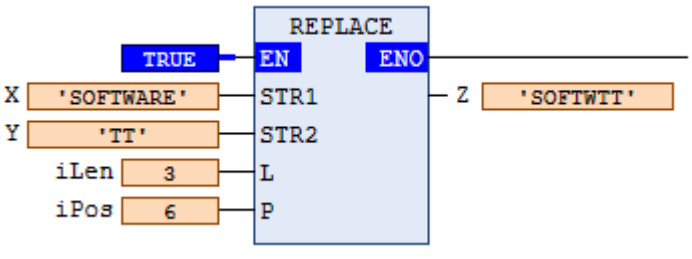
- **Function**

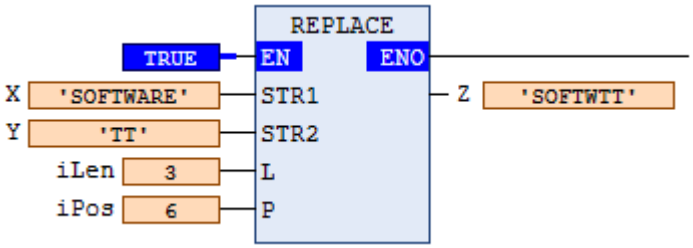
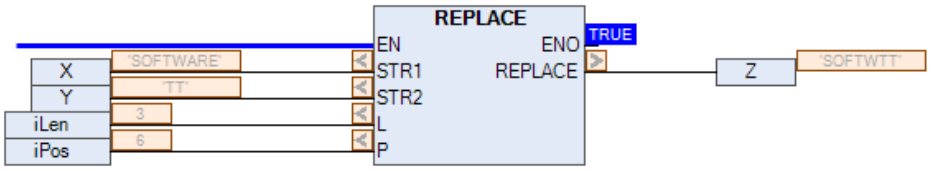
  This FC instruction is used to replace a partial string in *STR1* with *STR2*, where *L* indicates the length of the content and *P* indicates the position of the content.

- **Programming Example**

  The FC instruction (REPLACE) is used to replace a partial string with another string. For example, if input value *X* is SOFTWARE, *Y* is TT, *iLen* is 3, and *iPos* is 6, then output value *Z* is SOFTWTT.

  Feature in the ST/LD/FBD & CFC editor:

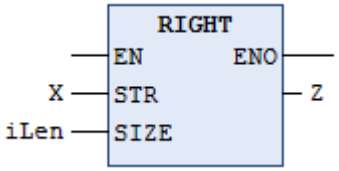
Programming Language	Program
ST	
LD	

Programming Language	Program
FBD	
CFC	

- **Supported Products**
  - AX series
- **Library**
  - Standard.lib

18.9.9. RIGHT (Get String from Right Instruction)

RIGHT extracts the content from the right side of a string.

FB/FC	Instruction	Graphic Expression	ST Language
FC	RIGHT		Z := RIGHT(X, iLen);

- **Inputs**

Name	Function	Data Type	Setting Value
STR	Input	STRING	As each data type suggested.
SIZE	Number of characters to be extracted	INT	–32768 to 32767

- **Outputs**

Name	Function	Data Type	Setting Value
Z	Right part of the string (its length = *SIZE*)	STRING	As each data type suggested.

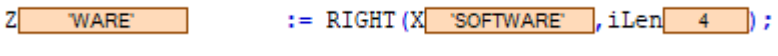
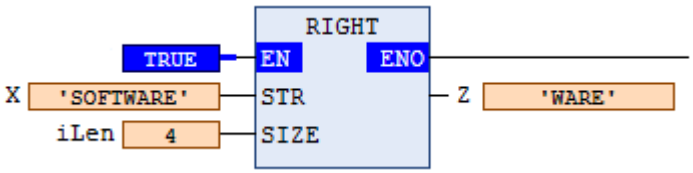
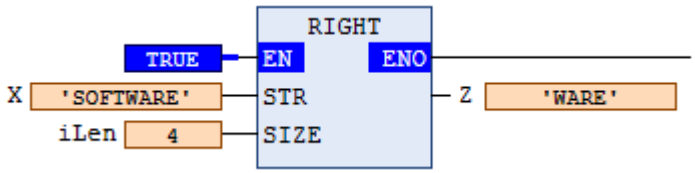
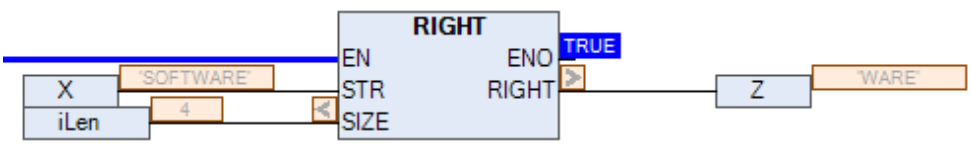
- **Function**

  This FC instruction is used to extract the content from the right side of the string with *SIZE* indicating the length of the content.

- **Programming Example**

  The FC instruction (RIGHT) is used to extract the content from the right side of the string. For example, if the input value *X* is SOFTWARE, and *iLen* is 4, then the output value *Z* is WARE.

  Feature in the ST/LD/FBD & CFC editor:

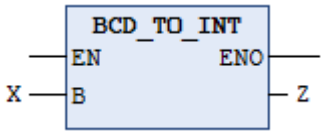
Programming Language	Program
ST	
LD	
FBD	
CFC	

- **Supported Products**
  - AX series
- **Library**
  - Standard.lib

18.10. BCD Conversion Instructions

18.10.1. BCD_TO_INT (BCD Code to INT Conversion Instruction)

BCD_TO_INT converts the BCD code into an INT value.

FB/FC	Instruction	Graphic Expression	ST Language
FC	BCD_TO_INT		Z := BCD_TO_INT (X);

- **Inputs**

Name	Function	Data Type	Setting Value
B	The binary form of the BCD code (or the decimal and hexadecimal corresponding to that binary)	BYTE	0 to 255

- **Outputs**

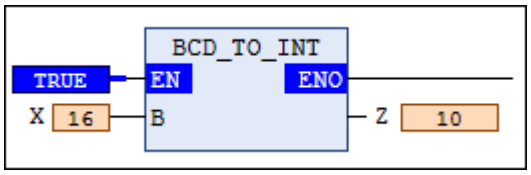
Name	Function	Data Type	Setting Value
Z	The actual value of the BCD code	INT	–32768 to 32767

- **Function**

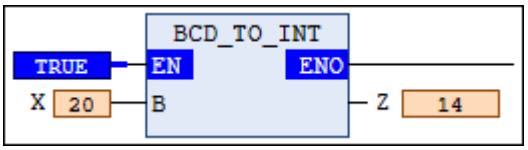
  This FC instruction is used to convert decimal value to hexadecimal as output.

- **Programming Example**

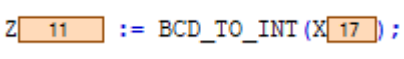
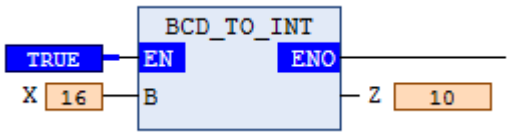
  The FC instruction (BCD_TO_INT) is used to convert BCD code into INT value as output. For example, if the input value *X* is 16, then the output value *Z* is 10.

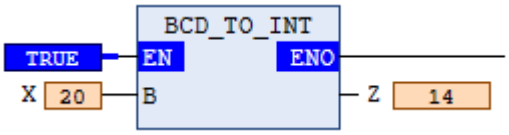
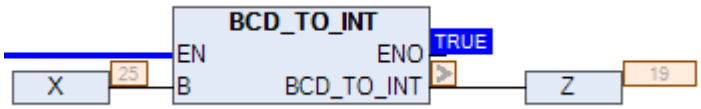


If input value *X* is 20, then output value *Z* is 14.



Feature in the ST/LD/FBD & CFC editor:

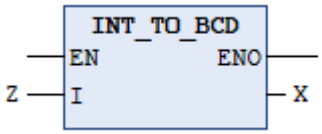
Programming Language	Program
ST	
LD	

Programming Language	Program
FBD	
CFC	

- **Supported Products**
  - AX series
- **Library**
  - None

18.10.2. INT_TO_BCD (Integer to BCD Code Conversion Instruction)

INT_TO_BCD converts an INT value into the BCD code.

FB/FC	Instruction	Graphic Expression	ST Language
FC	INT_TO_BCD		X := INT_TO_BCD (Z);

- **Inputs**

Name	Function	Data Type	Setting Value
I	Input	INT	–32768 to 32767

- **Outputs**

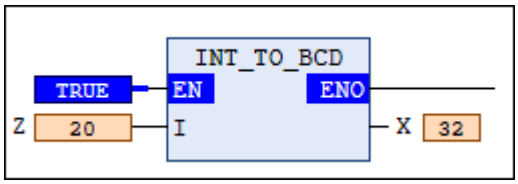
Name	Function	Data Type	Setting Value
X	Converted value	BYTE	0 to 255

- **Function**

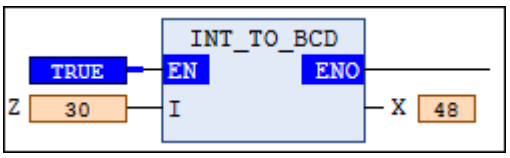
  This FC instruction is used to convert hexadecimal value to decimal as output.

- **Programming Example**

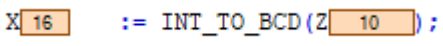
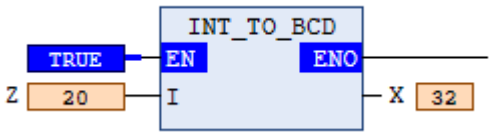
  The FC instruction (INT_TO_BCD) is used to convert an INT value into the BCD code as output. For example, if the input value *Z* is 20, then the output value *X* is 32.

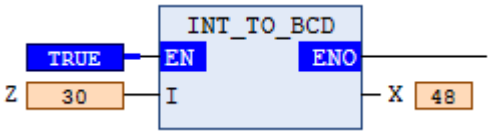
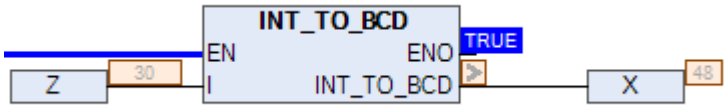


If the input value *Z* is 30, then the output value *X* is 48.



Feature in the ST/LD/FBD & CFC editor:

Programming Language	Program
ST	
LD	

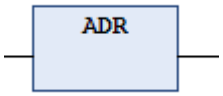
Programming Language	Program
FBD	
CFC	

- **Supported Products**
  - AX series
- **Library**
  - None

18.11. Address Operation Instructions

18.11.1. ADR (Address Function)

ADR gets the memory address of the input variable as output.

FB/FC	Instruction	Graphic Expression	ST Language
FC	ADR		P_T_byAddress := ADR(byADR);

- **Inputs**

Name	Function	Data Type	Setting Value
byADR	Input	POINTER TO <TYPE>	As each data type suggested.

- **Outputs**

Name	Function	Data Type	Setting Value
P_T_byAddress	Memory address of the input variable	POINTER TO <TYPE>	As each data type suggested.

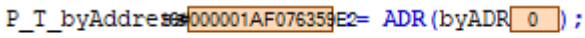
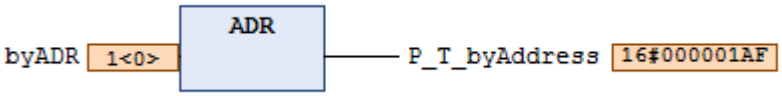
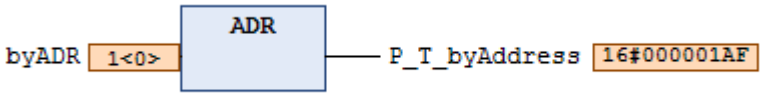
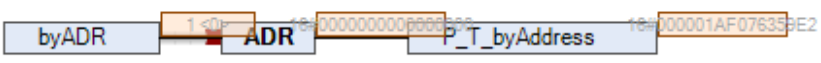
- **Function**

  The ADR function is used to get the memory address of the input variable and output it. It can be passed as a pointer to a function or used as a pointer within the program.

- **Programming Example**

  The memory address of the input variable *byADR* is obtained and can be used as a pointer in the program.

  Feature in the ST/LD/FBD & CFC editor:

Programming Language	Program
ST	
LD	
FBD	
CFC	

- **Supported Products**

  - AX series

- **Library**

  - None

18.11.2. ^ (Fetch Address Content Instruction)

^ obtains the data for the address that the pointer points to by adding the "^" symbol to the end of the pointer variable.

FB/FC	Instruction	Graphic Expression	ST Language
FC	^(FETCH)	^	byADR2 := P_T_byAddress^;

- **Inputs**

Name	Function	Data Type	Setting Value
P_T_byAddress	Input	POINTER TO <TYPE>	As each data type suggested.

- **Outputs**

Name	Function	Data Type	Setting Value
byADR	Address data	BYTE	0 to 255

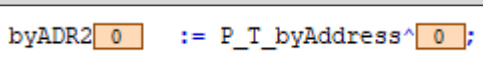
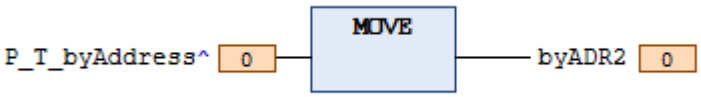
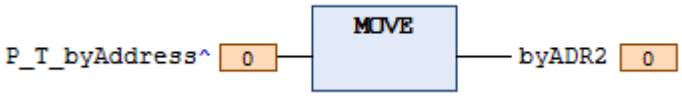
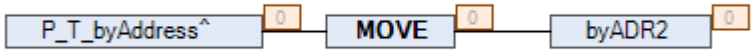
- **Function**

  This FC function is used to obtain the data for the address that the pointer points to by adding the "^" symbol to the end of the pointer variable.

- **Programming Example**

  The memory address pointed by the pointer *P_T_byAddress*^ (adding "^" symbol to the P_T_byAddress variable) helps to obtain the address data and output the result in the *byADR2* variable.

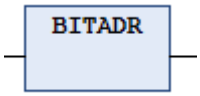
  Feature in the ST/LD/FBD & CFC editor:

Programming Language	Program
ST	
LD	
FBD	
CFC	

- **Supported Products**
  - AX series
- **Library**
  - None

18.11.3. BITADR (Bit Address Initialization Function)

BITADR outputs the bit offset within a segment in DWORD.

FB/FC	Instruction	Graphic Expression	ST Language
FC	BITADR		dwBitAddress := BITADR(xBool);

- **Inputs**

Name	Function	Data Type	Setting Value
xBool	Input	BOOL	TRUE, FALSE

- **Outputs**

Name	Function	Data Type	Setting Value
dwBitAddress	The offset address of the BOOL variable	DWORD	0 to 4294967295

- **Function**

  This function is used to yield the bit offset within a segment in DWORD. The offset depends on whether the **Byte addressing** checkbox is selected or cleared in the target system settings. When using pointers to addresses, note that applying an online change can shift the contents of addresses.

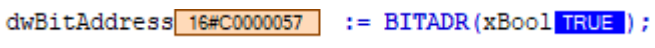
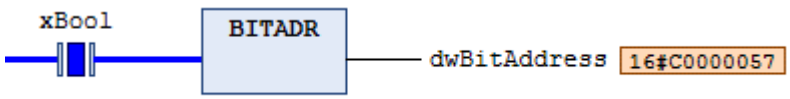
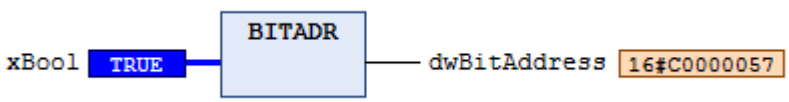
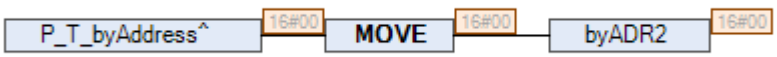
- **Programming Example**

  The mapping of Bit address is defined in the following three ways:

  - %MX for flag, given as 16#4 in hexadecimal
  - %IX for input, given as 16#8 in hexadecimal
  - %QX for output, given as 16#C in hexadecimal

    For example, the output for the *xBool* input type is given as 16#C in hexadecimal format.

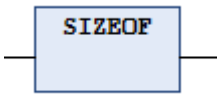
  Feature in the ST/LD/FBD & CFC editor:

Programming Language	Program
ST	
LD	
FBD	
CFC	

- **Supported Products**
  - AX series
- **Library**
  - None

18.11.4. SIZEOF (Data Type Size Function)

SIZEOF defines the number of bytes required by the input variable.

FB/FC	Instruction	Graphic Expression	ST Language
FC	SIZEOF		ui_arSize:=SIZEOF(arrInt);

- **Inputs**

Name	Function	Data Type	Setting Value
arrInt	Input	ARRAY[0..10] OF INT, BOOL	As each data type suggested.

- **Outputs**

Name	Function	Data Type	Setting Value
ui_arSize	Execution result	UINT, DWORD	As each data type suggested.

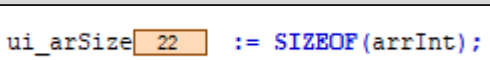
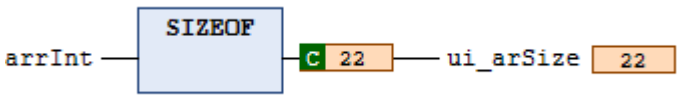
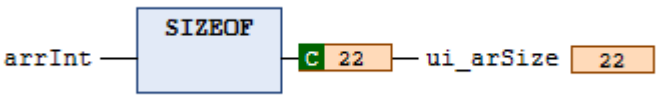
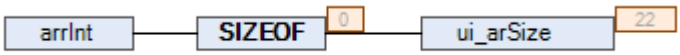
- **Function**

  This function defines the number of bytes required by the input variable. It always yields an unsigned value. The returned value is to detect the size of the input variable.

- **Programming Example**

  The output is 22 bytes for the input Array=[0..10] which contains 11 elements. Each element has a size of 2 bytes.

  Feature in the ST/LD/FBD & CFC editor:

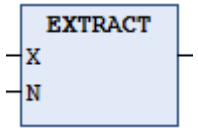
Programming Language	Program
ST	
LD	
FBD	
CFC	

- **Supported Products**
  - AX series
- **Library**
  - None

18.12. Bit/byte Manipulation Instructions

18.12.1. EXTRACT Function (Bit Fetch Instruction)

EXTRACT extracts the *Nth* bit from the input variable and outputs its value as a BOOL, starting the count from the zero bit.

FB/FC	Instruction	Graphic Expression	ST Language
FC	EXTRACT		xEXT_Out:= EXTRACT(X:=dwEXT_IN1, N:= byEXT_IN2);

- **Inputs**

Name	Function	Data Type	Setting Value
N	Control number	BYTE	0 to 255
X	Input	DWORD	0 to 4294967295

- **Outputs**

Name	Function	Data Type	Setting Value
xEXT_Out	Value of the N^{th} bit	BOOL	TRUE, FALSE

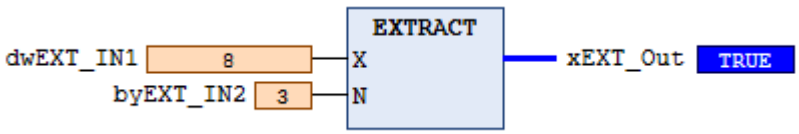
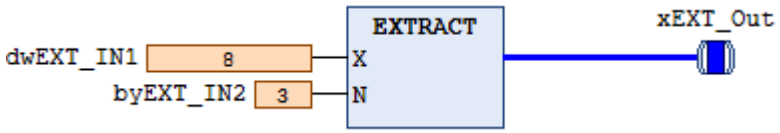
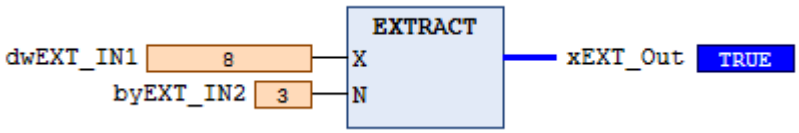
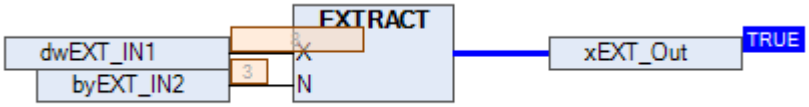
- **Function**

  This function fetches the input variable *X*, converts it to binary form, and outputs the *Nth* bit of the binary form of the input variable *X* in BOOL type.

- **Programming Example**

  The variable *dwEXT_IN1*=8 is the number to be converted to binary by the EXTRACT function. *byEXT_IN2* is the *Nth* (*N*=3) bit value. Output *xEXT_Out* as TRUE.

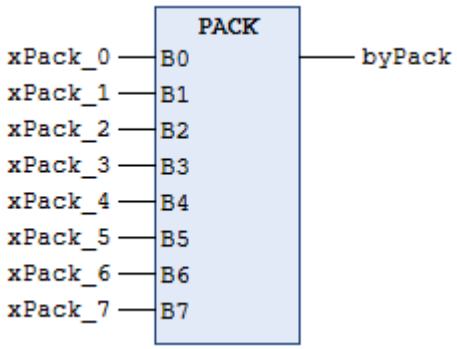
  Feature in the ST/LD/FBD & CFC editor:

Programming Language	Program
ST	
LD	
FBD	
CFC	

- **Supported Products**
  - AX series
- **Library**
  - Util.lib

18.12.2. PACK Function (Bit Integration Instruction)

PACK combines the 8 input bits from BOOL into a single Byte.

FB/FC	Instruction	Graphic Expression	ST Language
FC	PACK		byPack := PACK(B0 := xPack_0, B1:= xPack_1, B2:= xPack_2, B3:= xPack_3, B4:= xPack_4, B5:= xPack_5, B6:= xPack_6, B7:= xPack_7);

- **Inputs**

Name	Function	Data Type	Setting Value
B0, B1, B2, B3, B4, B5, B6, B7	Input	BOOL	TRUE, FALSE

- **Outputs**

Name	Function	Data Type	Setting Value
byPack	Execution result	BYTE	0 to 255

- **Function**

  This function delivers back eight input bits *B0, B1, ..., B7* from type BOOL as a BYTE.

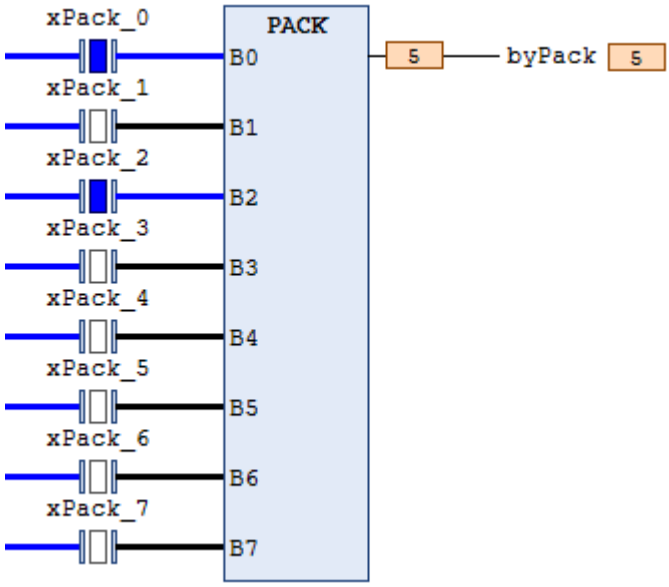
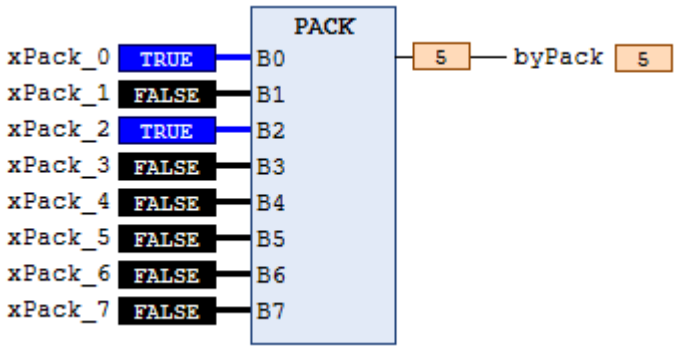
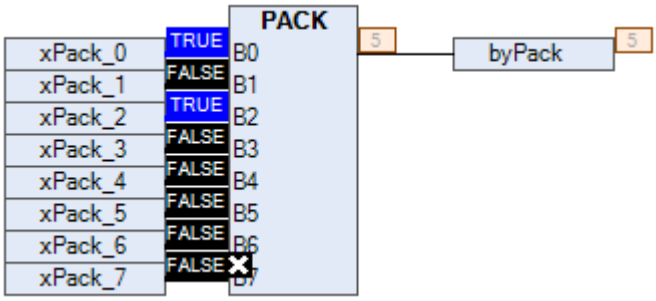
  The opposite instruction is the UNPACK instruction.

- **Programming Example**

  The bits corresponding to TRUE (1) and FALSE (0) is considered from the top and packed to give the output in 1 byte. The value after packing for the input considered as 101 is 5.

  Feature in the ST/LD/FBD & CFC editor:

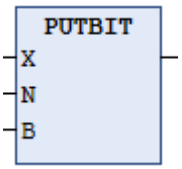
Programming Language	Program
ST	byPack 5 := PACK(B0:= xPack_0 TRUE, B1:= xPack_1 FALSE, B2:= xPack_2 TRUE, B3:= xPack_3 FALSE, B4:= xPack_4 FALSE, B5:= xPack_5 FALSE, B6:= xPack_6 FALSE, B7:= xPack_7 FALSE) 5;

Programming Language	Program
LD	
FBD	
CFC	

- **Supported Products**
  - AX series
- **Library**
  - Util.lib

18.12.3. PUTBIT Function (Bit Assignment Instruction)

PUTBIT puts the value of a bit in a binary sequence into a specific position of a variable.

FB/FC	Instruction	Graphic Expression	ST Language
FC	PUTBIT		dwPutbit_Out := PUTBIT(X := dwPutbit_In, N := byPutbit_Var, B := xPutbit_Var);

- **Inputs**

Name	Function	Data Type	Setting Value
X	Value to be converted	DWORD	0 to 4294967295
N	Position of the bit to be changed, starting with 0	BYTE	0 to 255
B	Value of the specified bit	BOOL	TRUE, FALSE

- **Outputs**

Name	Function	Data Type	Setting Value
dwPutbit_Out	Value with the changed bit	DWORD	0 to 4294967295

- **Function**

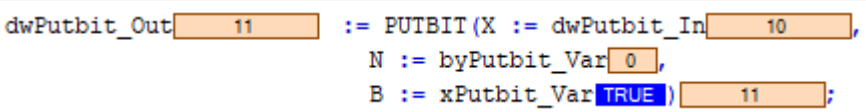
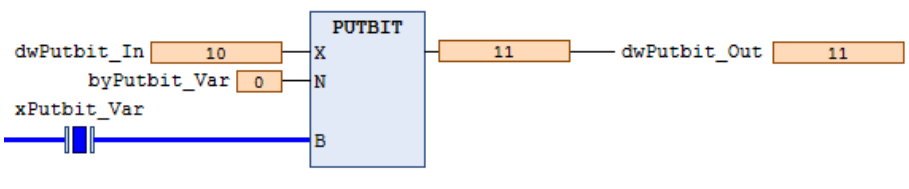
  This function is used to assign the *Nth* bit (*N*=0,1...) of the input variable *X* to variable *B* and output the converted value of *X*.

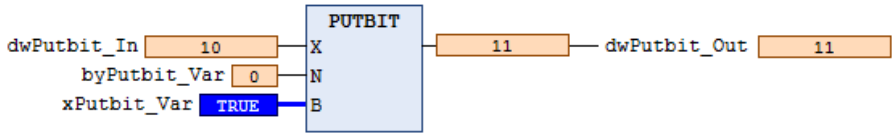
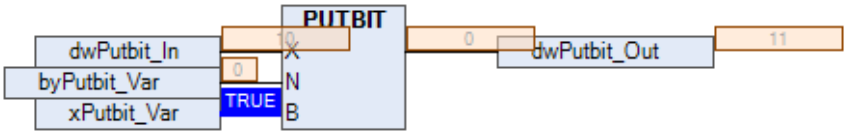
- **Programming Example**

  The variable *dwPutbit_In* takes the binary input of an integer value 10 and the variable *byPutbit_Var*=0 is used to specify the bit place in the binary form of the input value. Then assign the value of variable *xPutbit_Var*=TRUE(1).

The output value is the binary form of input variable value 10 with its 0th bit converted and give the final output as 11.

Feature in the ST/LD/FBD & CFC editor:

Programming Language	Program
ST	
LD	

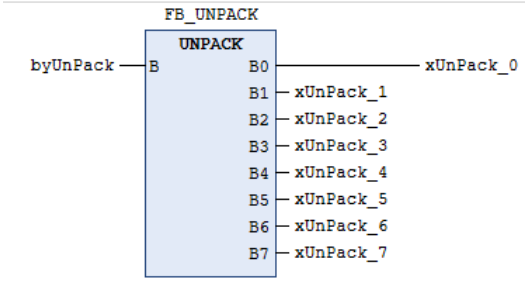
Programming Language	Program
FBD	
CFC	

- **Supported Products**
  - AX series
- **Library**
  - Util.lib

18.12.4. UNPACK Function (Bit Split Instruction)

UNPACK splits the byte input B and convert it into 8 BOOL type output variables *B0,...,B7*.

This instruction is the opposite to the PACK instruction.

FB/FC	Instruction	Graphic Expression	ST Language
FB	UNPACK		FB_UNPACK(B:= byUnPack, B0=> xUnPack_0, B1=> xUnPack_1, B2=> xUnPack_2, B3=> xUnPack_3, B4=> xUnPack_4, B5=> xUnPack_5, B6=> xUnPack_6, B7=> xUnPack_7);

- **Inputs**

Name	Function	Data Type	Setting Value
B	Input	BYTE	0 to 255

- **Outputs**

Name	Function	Data Type	Setting Value
B0, B1, B2, B3, B4, B5, B6, B7	Execution result	BOOL	TRUE, FALSE

- **Function**

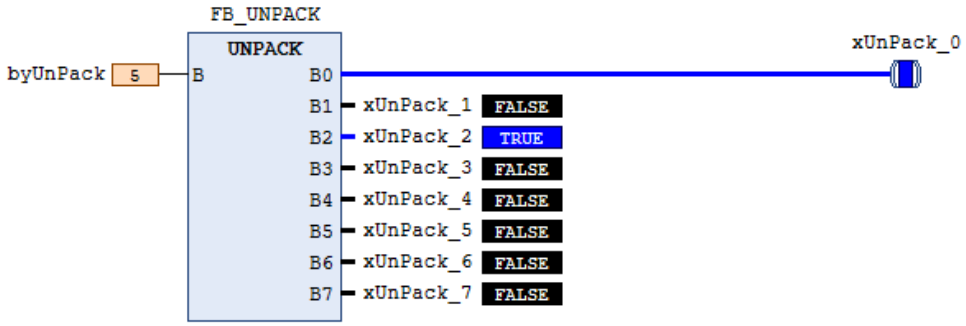
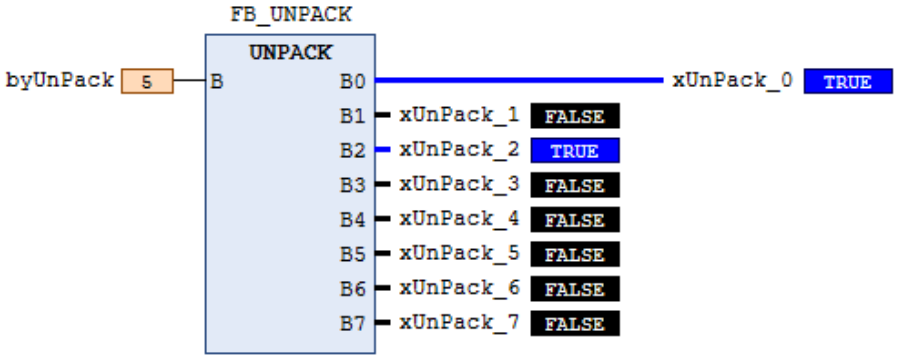
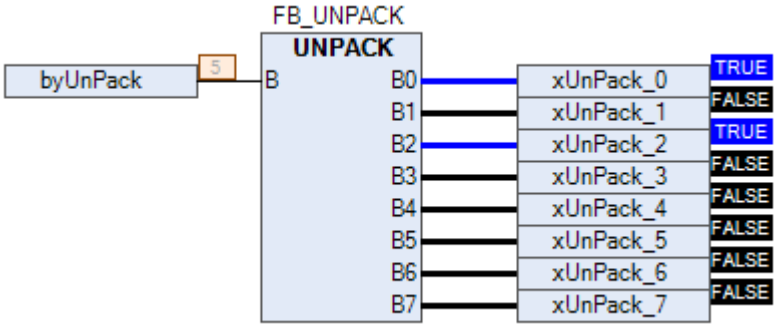
  This function splits the given byte input and converts it into 8 BOOL type output variables *B0...B7*. This instruction is the opposite to the PACK instruction.

- **Programming Example**

  The value of the input variable *byUnPack* is 5. After running the UNPACK instruction, the output provides the binary equivalent of the decimal number 5, which is 101, in BOOL format from top to bottom, where TRUE=1 and FALSE=0.

  Feature in the ST/LD/FBD & CFC editor:

Programming Language	Program
ST	FB_UNPACK(B 5 := byUnPack 5 , B0 TRUE => xUnPack_0 TRUE , B1 FALSE => xUnPack_1 FALSE , B2 TRUE => xUnPack_2 TRUE , B3 FALSE => xUnPack_3 FALSE , B4 FALSE => xUnPack_4 FALSE , B5 FALSE => xUnPack_5 FALSE , B6 FALSE => xUnPack_6 FALSE , B7 FALSE => xUnPack_7 FALSE);

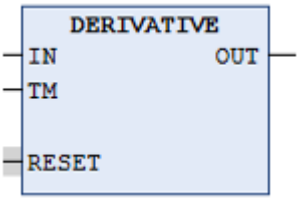
Programming Language	Program
LD	
FBD	
CFC	

- **Supported Products**
  - AX series
- **Library**
  - Util.lib

18.13. Advanced Mathematical Operation Instructions

18.13.1. DERIVATIVE Function (Differential Instruction)

DERIVATIVE performs a differential operation on continuous input variables.

FB/FC	Instruction	Graphic Expression	ST Language
FB	DERIVATIVE		FB_DERIVATIVE(IN := , TM := , RESET := , OUT =>);

- **Inputs**

Name	Function	Data Type	Setting Value
IN	Continuous input	REAL	1.0E–44 to 3.402823E+38
TM	Time since last call in msec	DWORD	0 to 4294967295
RESET	TRUE: *OUT* is set to zero and saved values are set to the current input value *IN*.	BOOL	TRUE, FALSE

- **Outputs**

Name	Function	Data Type	Setting Value
OUT	Derivative result	REAL	1.0E–44 to 3.402823E+38

- **Function**

  This instruction performs a differential operation on continuous input variables. The instruction only differentiates the latest four input values to reduce errors caused by inaccuracies in inputs.

  Differential iteration formula:

$$OUT = \frac{3*[IN(k)-IN(k-3)]+IN(k-1)-IN(k-2)}{3*TM(k-2)+4*TM(k-1)+3*TM(k)}$$

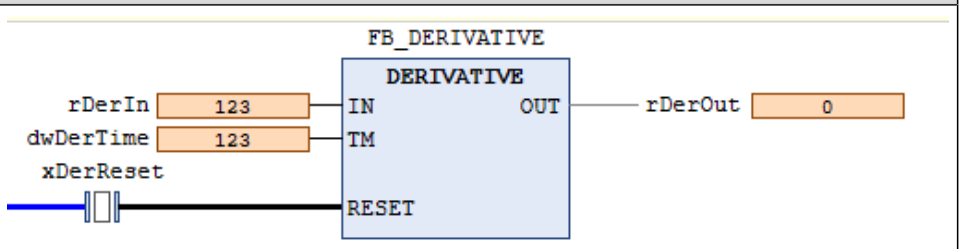
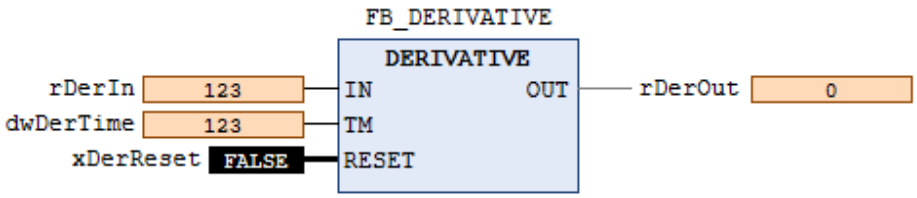
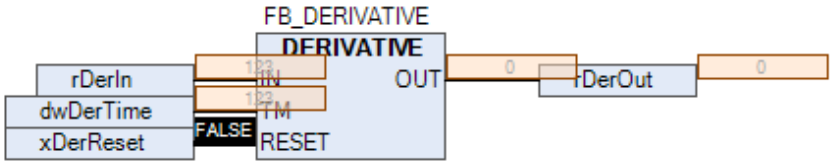
  k–3, k–2, and k–1 are the marks for four consecutive input values over the course of time.

- **Programming Example**

  The variable *rDerIn* represents the input variable *IN* and *dwDerTime* represents the input variable *TM* in the above equation. Once the *xDerReset* value is set to FALSE, the differential value is calculated and given as *rDerOut* which is the *OUT* value in the above equation.

  Feature in the ST/LD/FBD & CFC editor:

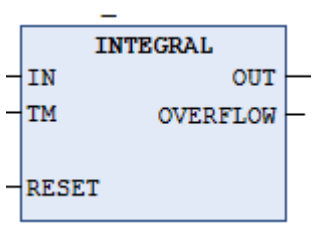
Programming Language	Program
ST	FB_DERIVATIVE(IN 123 := rDerIn 123, TM 123 := dwDerTime 123, RESET FALSE := xDerReset FALSE, OUT 0 => rDerOut 0);

Programming Language	Program
LD	
FBD	
CFC	

- **Supported Products**
  - AX series
- **Library**
  - Util.lib

18.13.2. INTEGRAL Function (Integral Instruction)

INTEGRAL performs an integral operation on continuous input variables.

FB/FC	Instruction	Graphic Expression	ST Language
FB	INTEGRAL		FB_INTEGRAL(IN := , TM := , RESET :=, OUT =>, OVERFLOW =>);

- **Inputs**

Name	Function	Data Type	Setting Value
IN	Continuous input	REAL	1.0E–44 to 3.402823E+38
TM	Time since last call in msec	DWORD	0 to 4294967295
RESET	TRUE: *OUT* is set to zero and *OVERFLOW* is set to FALSE.	BOOL	TRUE, FALSE

- **Outputs**

Name	Function	Data Type	Setting Value
OUT	Value of the integral	REAL	1.0E–44 to 3.402823E+38
OVERFLOW	TRUE: The value of *OUT* is out of the range of REAL variables.	BOOL	TRUE, FALSE

- **Function**

  This instruction performs an integral operation on continuous input variables.

  Integral iteration formula:

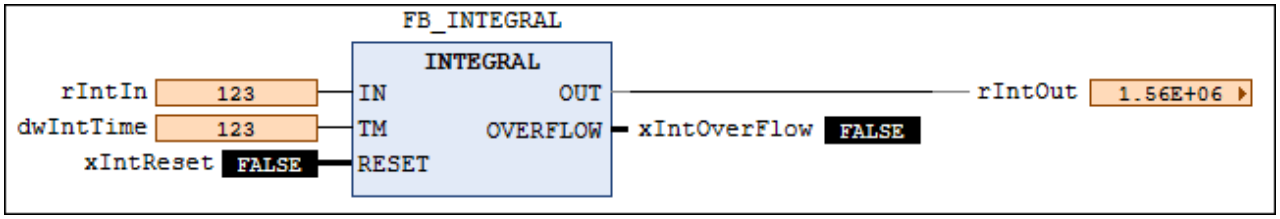
  *A(k) = A(k–1) +TM * IN(k–1), B(k) = B(k–1) +TM * IN(k)*

  $$OUT(k) = \frac{A(k) + B(k)}{2}$$

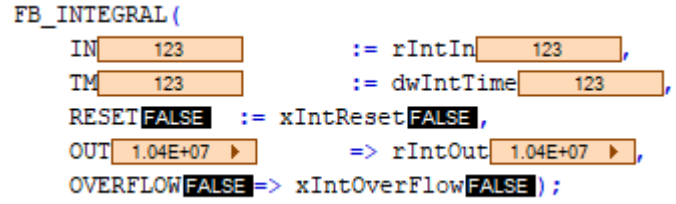
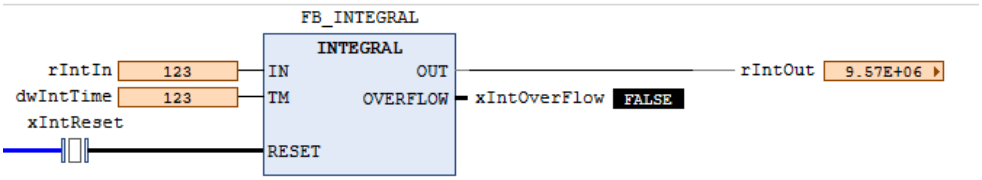
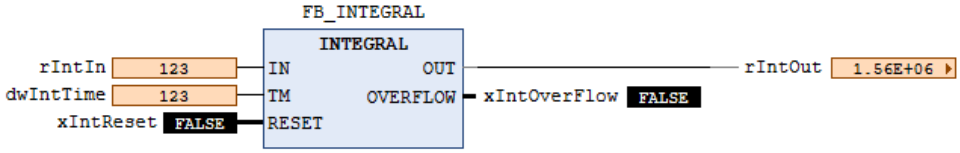
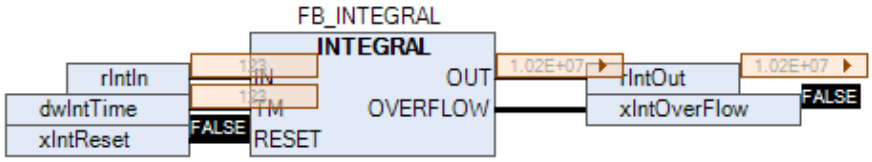
  k–1 and k are marks for two consecutive input values over time.

- **Programming Example**

  The variable *rIntIn* represents the input variable *IN* and *dwIntTime* represents the input variable *TM* in the above equation. Once the *xIntReset* value is set to FALSE, the integral value is calculated over time and given as *rDerOut* which is the *OUT* value in the above equation.



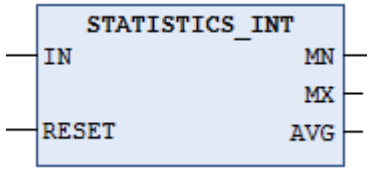
  Feature in the ST/LD/FBD & CFC editor:

Programming Language	Program
ST	
LD	
FBD	
CFC	

- **Supported Products**
  - AX series
- **Library**
  - Util.lib

18.13.3. STATISTICS_INT Function (Integer Statistics Instruction)

STATISTICS_INT is used to calculate the minimum, maximum, and average values of integer inputs. The calculation is done over time and can be reset in order to start a fresh calculation.

FB/FC	Instruction	Graphic Expression	ST Language
FB	STATISTICS_INT		FB_STATISTICS_INT(IN := , RESET :=, MN => , MX => , AVG =>);

- **Inputs**

Name	Function	Data Type	Setting Value
IN	Input	INT	−32768 to 32767
RESET	TRUE: *AVG* is set to 0 and the instruction is reset.	BOOL	TRUE, FALSE

- **Outputs**

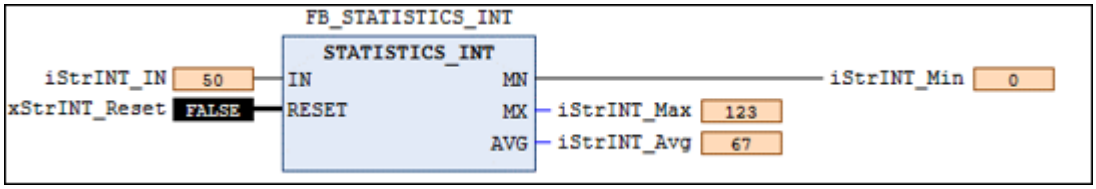
Name	Function	Data Type	Setting Value
MN	Minimum value	INT	−32768 to 32767
MX	Maximum value		
AVG	Average value		

- **Function**

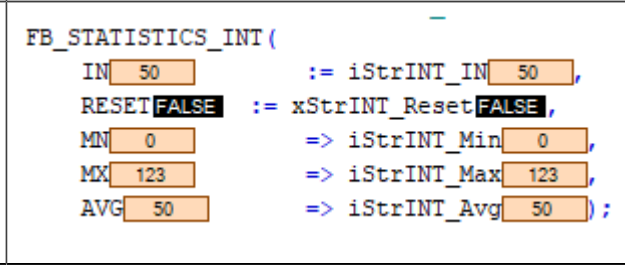
  This function is used to calculate the minimum, maximum, and average values of integer inputs.

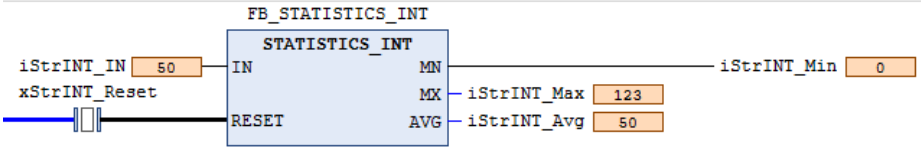
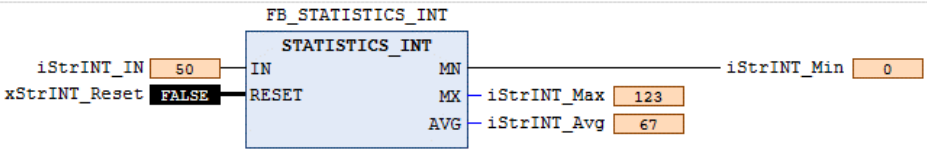
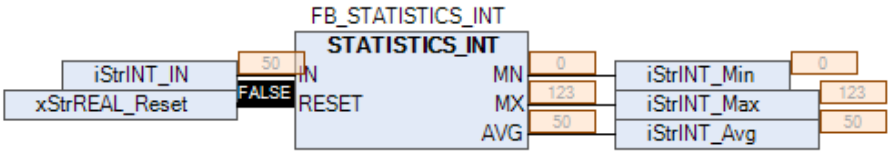
- **Programming Example**

  The variable *iStrINT_IN* is integer input. When the value of *xStrINT_Reset* is FALSE, the minimum, maximum, and average values are calculated and given as output in the *iStrINT_Min*, *iStrINT_Max*, and *iStrINT_Avg* variables respectively.



  Feature in the ST/LD/FBD & CFC editor:

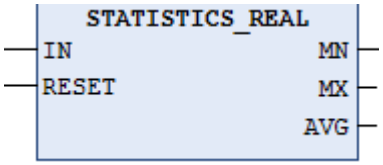
Programming Language	Program
ST	

Programming Language	Program
LD	
FBD	
CFC	

- **Supported Products**
  - AX series
- **Library**
  - Util.lib

18.13.4. STATISTICS_REAL Function (Real Statistics Instruction)

STATISTICAL_REAL is used to calculate the minimum, maximum, and average values of REAL inputs.

FB/FC	Instruction	Graphic Expression	ST Language
FB	STATISTICAL_REAL		FB_STATISTICS_REAL(IN := , RESET :=, MN =>, MX => , AVG =>);

- **Inputs**

Name	Function	Data Type	Setting Value
IN	Input	REAL	1.0E–44 to 3.402823E+38
RESET	TRUE: *AVG* is set to 0 and the instruction is reset.	BOOL	TRUE, FALSE

- **Outputs**

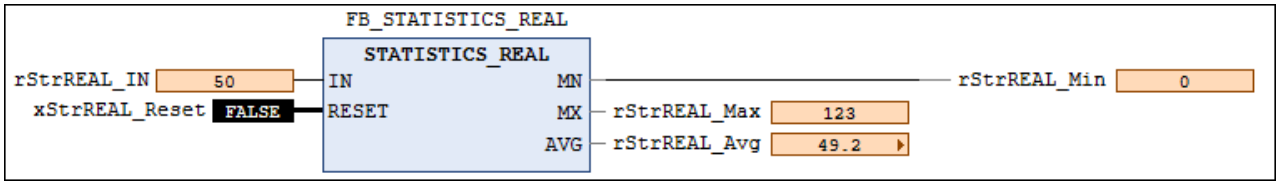
Name	Function	Data Type	Setting Value
MN	Minimum value	REAL	1.0E–44 to 3.402823E+38
MX	Maximum value		
AVG	Average value		

- **Function**

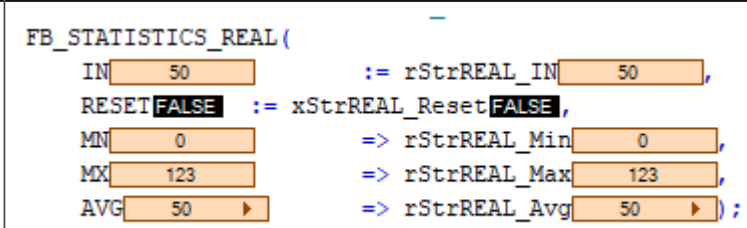
  This function is used to calculate the minimum, maximum, and average values of REAL inputs.

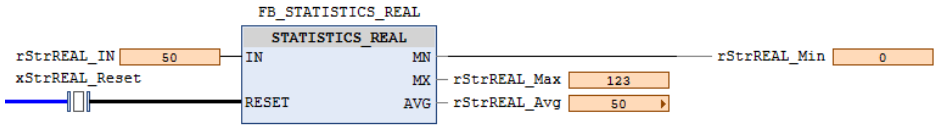
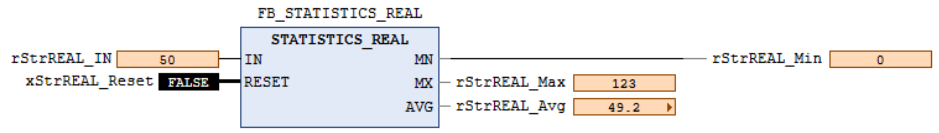
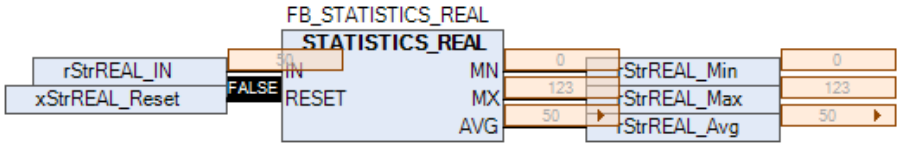
- **Programming Example**

  The variable *rStrREAL_IN* is the REAL input. When the value of *xStrREAL_Reset* is FALSE, the minimum, maximum, and average values are calculated and given as output in the *rStrREAL_Max*, *rStrREAL_Avg*, and *rStrREAL_Min* variables respectively.



  Feature in the ST/LD/FBD & CFC editor:

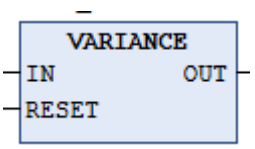
Programming Language	Program
ST	

Programming Language	Program
LD	
FBD	
CFC	

- **Supported Products**
  - AX series
- **Library**
  - Util.lib

18.13.5. VARIANCE Function (Squared Deviation Instruction)

VARIANCE calculates the square deviation of the variable input value.

FB/FC	Instruction	Graphic Expression	ST Language
FB	VARIANCE		FB_VARIANCE(IN := , RESET := , OUT =>);

- **Inputs**

Name	Function	Data Type	Setting Value
IN	Input	REAL	1.0E–44 to 3.402823E+38
RESET	TRUE: reset the instruction.	BOOL	TRUE, FALSE

- **Outputs**

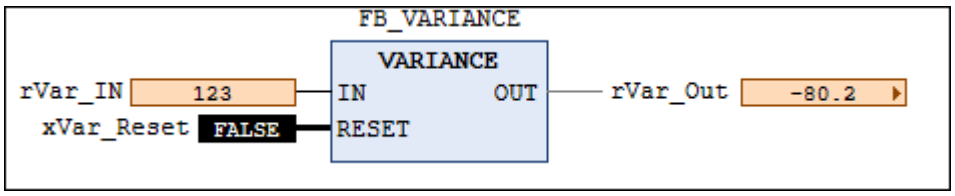
Name	Function	Data Type	Setting Value
OUT	Square deviation	REAL	1.0E–44 to 3.402823E+38

- **Function**

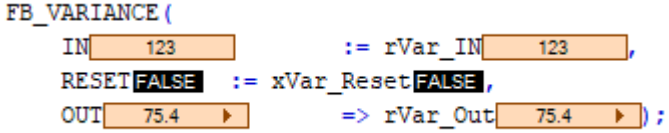
  This instruction calculates the square deviation of the variable input value over time until the variance is extended to each call to the function block and the reset is done. The standard deviation can be obtained by performing the square root calculation of the VARIANCE value. VARIANCE can be reset by setting *RESET*=TRUE.

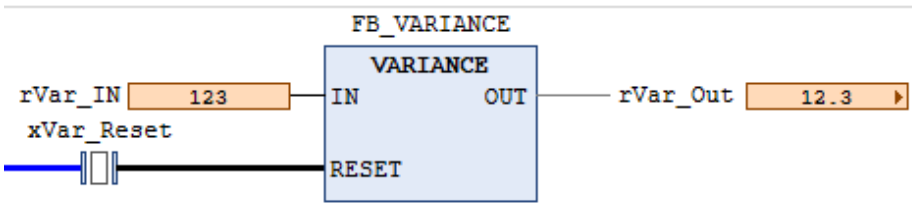
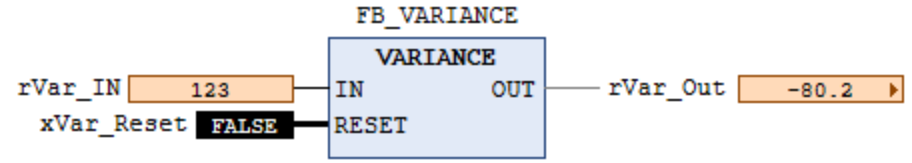
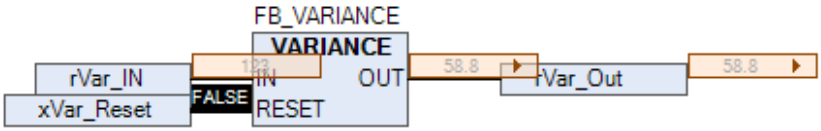
- **Programming Example**

  The variable *rVar_IN* provides the input value from which the square deviation is calculated over time until the variance is extended to each call to the function block and reset is done and output the value in *rVar_Out*.



Feature in the ST/LD/FBD & CFC editor:

Programming Language	Program
ST	

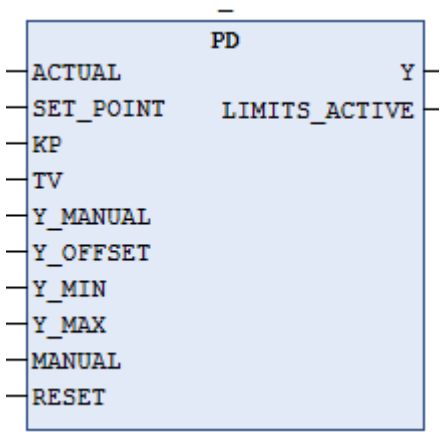
Programming Language	Program
LD	
FBD	
CFC	

- **Supported Products**
  - AX series
- **Library**
  - Util.lib

18.14. Controller Instructions

18.14.1. PD (Proportional Derivative Controller Instruction)

PD works as a proportional differential controller.

FB/FC	Instruction	Graphic Expression	ST Language
FB	PD		FB_PD(ACTUAL = , SET_POINT := , KP := , TV = , Y_MANUA L:= , Y_OFFSET := , Y_MIN := , Y_MAX := , MANUAL := , RESET := , Y => , LIMITS_ACTIVE =>);

- **Inputs**

Name	Function	Data Type	Setting Value
ACTUAL	Actual value, process variable	REAL	1.0E–44 to 3.402823E+38
SET_POINT	Specified value, set point		
KP	Proportionality constant P		
TV	Rate time, derivative time D (sec) If set to 0, it works as a P controller.		
Y_MANUAL	*Y* is set to this value as long as *MANUAL* = TRUE.		
Y_OFFSET	Offset of Y		
Y_MIN	Minimum value of *Y*		
Y_MAX	Maximum value of *Y*		
MANUAL	TRUE: Manual, *Y* is not influenced by controller.	BOOL	TRUE, FALSE
RESET	TRUE: Set *Y* to *Y_OFFSET*; reset the controller.		

- **Outputs**

Name	Function	Data Type	Setting Value
LIMITS_ACTIVE	TRUE: *Y* has exceeded the given limits *Y_MIN* and *Y_MAX*.	BOOL	TRUE, FALSE
Y	Output, manipulated variable	REAL	1.0E–44 to 3.402823E+38

- **Function**

  This instruction is a proportional differential controller. The error value which occurs as a difference between a desired set point and a measured process variable is calculated by the proportional derivative controller.

  The equation which gives the output is:

  $$Y = KP * (\Delta + TV * \frac{\sigma\Delta}{\sigma t}) + Y\_OFFSET$$

  The instruction will automatically calculate the $\frac{\sigma\Delta}{\sigma t}$ value.

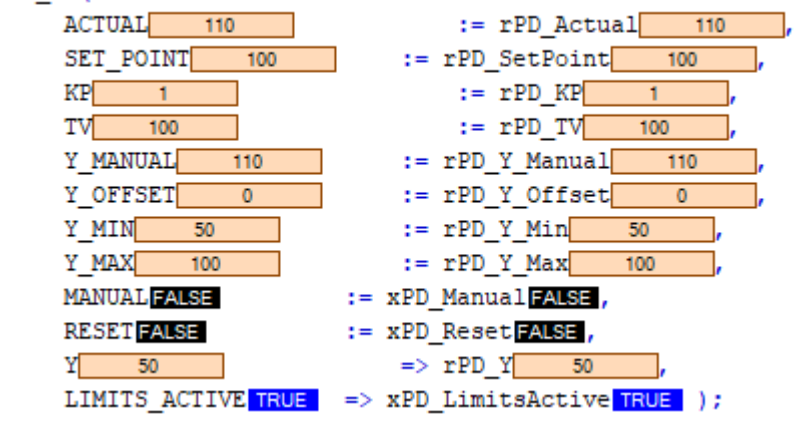
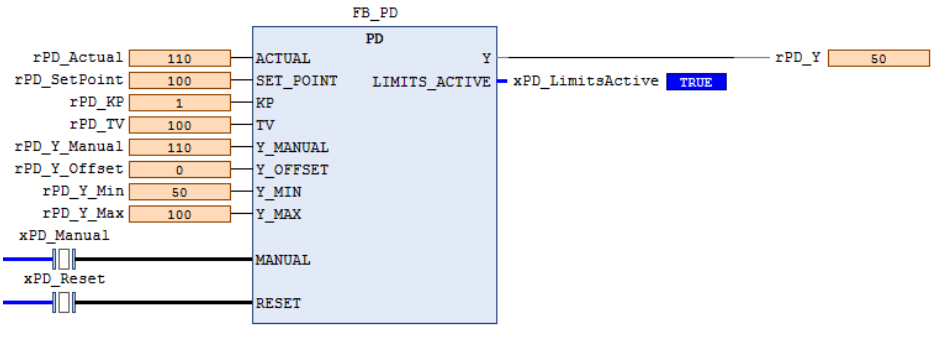
- **Programming Example**

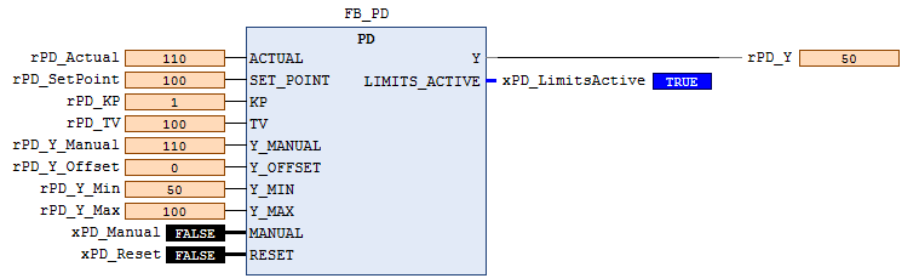
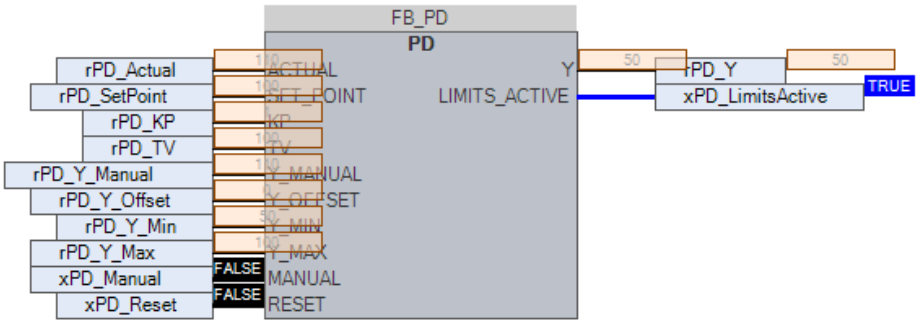
  The variable *rPD_Y_Min*=50 and *rPD_Y_Max*=100 are the minimum and maximum threshold values of the input respectively. If *rPD_Y* reaches the limit value, the output *xPD_LimitsActive* will set to TRUE, and *rPD_Y* will be maintained within the specified range. This function block operates only when *rPD_Y_MIN* < *rPD_Y_MAX*. The variable *rPD_Actual* is the current value of the input. The variable *rPD_Y_Manual* is the manual value given as input. This is deactivated once the values of the variable *xPD_Reset* is set to FALSE.

  The value of the variable *rPD_KP*=1 is the proportionality constant. *rPD_TV* enables the proportional derivative controller to maintain the value within the close range set in the *rPD_TV* variable. The value of r*PD_Y_Offset* is used only in the critical condition to achieve the output within the given *rPD_Y_Min* and *rPD_Y_Max* range respectively.

  The output value is calculated with the above equation and displayed as *rPD_Y*=50 along with *xPD_LimitsActive* = FALSE which means that the PD controller tries to maintain the output value within the range between 50 and 150 which are the values of *rPD_Y_Min* and *rPD_Y_Max* respectively since the anticipated output value of *rPD_Y* is below 50.

  Feature in the ST/LD/FBD & CFC editor:

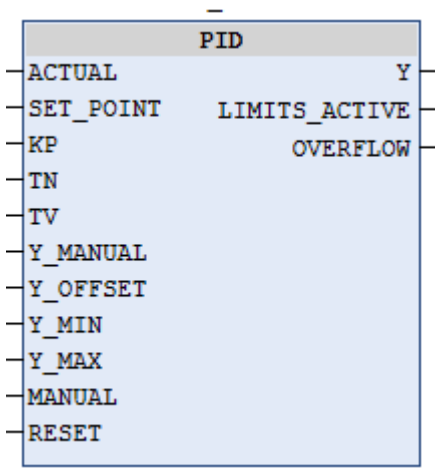
Programming Language	Program
ST	
LD	

Programming Language	Program
FBD	
CFC	

- **Supported Products**
  - AX series
- **Library**
  - Util.lib

18.14.2. PID (Proportional Integral Derivative Controller Instruction)

PID works as a proportional integral differential controller.

FB/FC	Instruction	Graphic Expression	ST Language
FB	PID		FB_PID(ACTUAL := , SET_POINT := , KP := , TN := , TV := , Y_MANUAL := , Y_OFFSET := , Y_MIN := , Y_MAX := , MANUAL := , RESET := , Y=>, LIMITS_ACTIVE => , OVERFLOW=>);

- **Inputs**

Name	Function	Data Type	Setting Value
ACTUAL	Actual value, process variable	REAL	1.0E–44 to 3.402823E+38
SET_POINT	Specified value, set point		
KP	Proportionality constant P		
TN	Integral time (s)		
TV	Rate time, derivative time D (sec) If set to 0, it works as a PI controller.		
Y_MANUAL	*Y* is set to this value as long as *MANUAL* = TRUE.		
Y_OFFSET	Offset of *Y*		
Y_MIN	Minimum value of *Y*		
Y_MAX	Maximum value of *Y*		
MANUAL	TRUE: Manual, *Y* is not influenced by controller.	BOOL	TRUE, FALSE
RESET	TRUE: Set *Y* to *Y_OFFSET*; reset the controller.		

- **Outputs**

Name	Function	Data Type	Setting Value
OVERFLOW	Output overflow flag TRUE: Overflow in integral part, the controller is paused and will only be activated again when it is reinitialized.	BOOL	TRUE, FALSE
LIMITS_ACTIVE	TRUE: *Y* has exceeded the given limits *Y_MIN* and *Y_MAX*.		
Y	Output, manipulated variable	REAL	1.0E–44 to 3.402823E+38

- **Function**

  This instruction is a proportional integral differential controller. When the value of *rPID_TV*=0, PID controller acts as PI controller. The error value, a difference between a desired set point and a measured process variable, is calculated by the proportional integral derivative controller. The PID controller applies the correction based on proportional, integral, and derivative terms which give their name to the controller type. This PID instruction works in a closed loop concept.

  The equation which gives the output is:

$$Y = KP*(\Delta + \frac{1}{TN}*\int \Delta(t)dt + TV*\frac{\sigma\Delta}{\sigma t}) + Y\_OFFSET$$

$$\Delta = SET\_POINT - ACTUAL$$

  The instruction will automatically calculate $\frac{\sigma\Delta}{\sigma t}$ value.

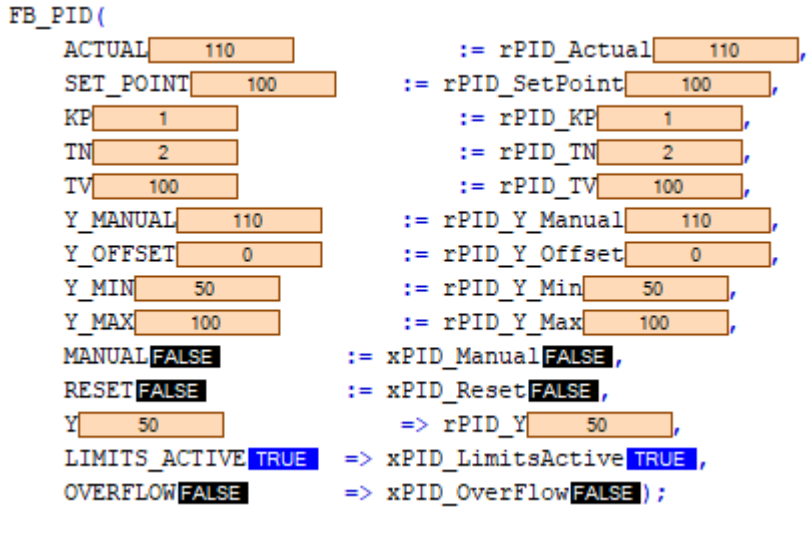
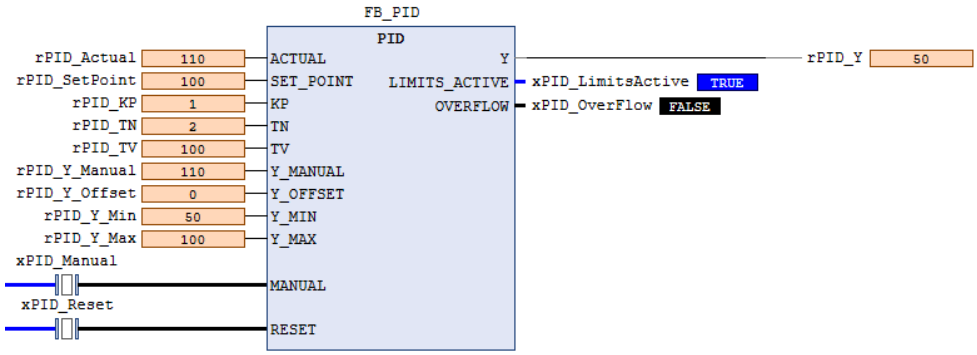
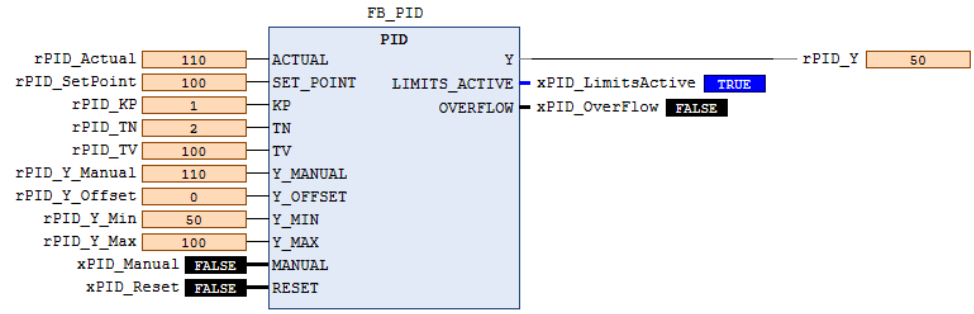
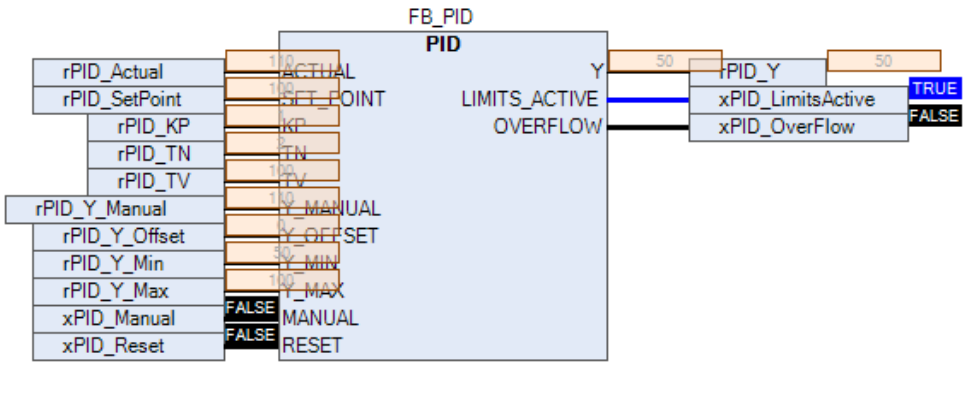
- **Programming Example**

  The variable *rPID_Y_Min*=50 and *rPID_Y_Max*=100 are the minimum and maximum threshold values of the input respectively. If *rPID_Y* reaches the limit value, the output *xPID_LimitsActive* will set to TRUE, and *rPID_Y* will be maintained within the specified range. This function block operates only when *rPID_Y_MIN* < *rPID_Y_MAX*. The variable *rPID_Actual* is the current value of the input. The variable *rPID_Y_Manual* is the manual value given as input. This is deactivated once the values of the variable *xPID_Reset* is set to FALSE.

  The value of the variable *rPID_KP*=1 is the proportionality constant. *rPID_TV* enables the proportional derivative controller to maintain the value within the close range set in the *rPID_TV* variable. The value of r*PID_Y_Offset* is used only in the critical condition to achieve the output within the given *rPID_Y_Min* and *rPID_Y_Max* range respectively. The variable *rPID_TN* indicates the Reset time(t) which is the time taken by the output (*rPID_Y*) to achieve the *rPID_SetPoint* value.

  The output value is calculated with the above equation and displayed as *rPID_Y*=50 along with *xPID_LimitsActive* = FALSE which means that the PID controller tries to maintain the output value within the range between 50 and 150 which are the values of *rPID_Y_Min* and *rPID_Y_Max* respectively since the anticipated output value of *rPID_Y* is gone below 50. The variable *xPID_OverFlow* indicates the overflow in the integral part.

  Feature in the ST/LD/FBD & CFC editor:

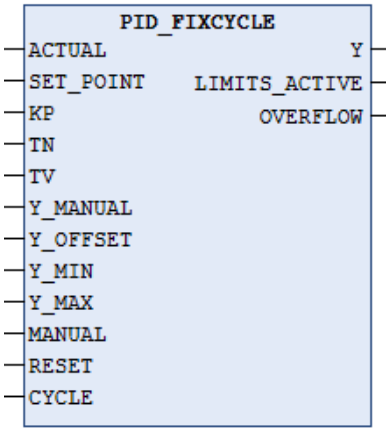
Programming Language	Program
ST	
LD	
FBD	
CFC	

- **Supported Products**
  - AX series

- **Library**
  - Util.lib

18.14.3. PID_FIXCYCLE (Proportional Integral Derivative Controller)

PID_FIXCYCLE works as a proportional integral differential controller. Compared with the PID controller mentioned above, there is an additional sampling period parameter *CYCLE*, which is used to set the time step for integral and differential in seconds.

FB/FC	Instruction	Graphic Expression	ST Language
FB	PID_FIXCYCLE		FB_PID_Fixcycle(ACTUAL := , SET_POINT:= , KP := , TN := , TV := , Y_MANUAL := , Y_OFFSET := , Y_MIN := , Y_MAX := , MANUAL:= , RESET := , CYCLE := , Y=> , LIMITS_ACTIVE => , OVERFLOW =>);

- **Inputs**

Name	Function	Data Type	Setting Value
ACTUAL	Actual value, process variable	REAL	1.0E–44 to 3.402823E+38
SET_POINT	Specified value, set point		
KP	Proportionality constant P		
TN	Integral time (s)		
TV	Rate time, derivative time D (sec) If set to 0, it works as a PI controller.		
Y_MANUAL	*Y* is set to this value as long as *MANUAL* = TRUE.		
Y_OFFSET	Offset of *Y*		
Y_MIN	Minimum value of *Y*		
Y_MAX	Maximum value of *Y*		
CYCLE	Time(s) between two calls		
MANUAL	TRUE: Manual, *Y* is not influenced by controller.	BOOL	TRUE, FALSE

Name	Function	Data Type	Setting Value
RESET	TRUE: Set *Y* to *Y_OFFSET*; reset the controller.		

- **Outputs**

Name	Function	Data Type	Setting Value
LIMITS_ACTIVE	TRUE: *Y* has exceeded the given limits *Y_MIN* and *Y_MAX*.	BOOL	TRUE, FALSE
OVERFLOW	Output overflow flag TRUE: Overflow in integral part, the controller is paused and will only be activated again when it is reinitialized.		
Y	Execution result	REAL	1.0E–44 to 3.402823E+38

- **Function**

  This instruction is a proportional integral differential controller. Compared with the PID controller introduced previously, there is one more *CYCLE* parameter, which means a sampling period. *CYCLE* is a REAL input parameter, which is used to set the time step of integral and differential in seconds.

  The equation which gives the output is:

$$Y = KP*(\Delta + \frac{1}{TN}*\int \Delta(t)dt + TV*\frac{\sigma\Delta}{\sigma t}) + Y\_OFFSET$$

$$\Delta = SET\_POINT - ACTUAL$$

  The instruction will automatically calculate $\frac{\sigma\Delta}{\sigma t}$ value.

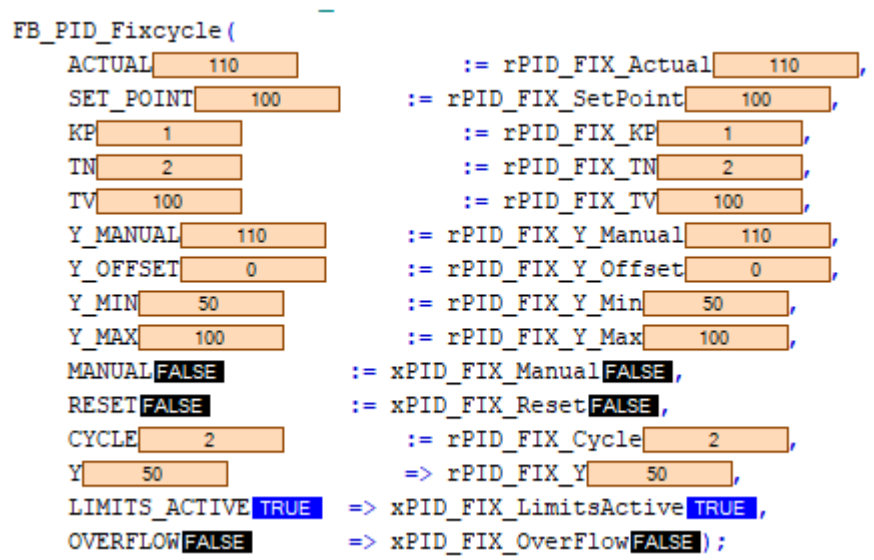
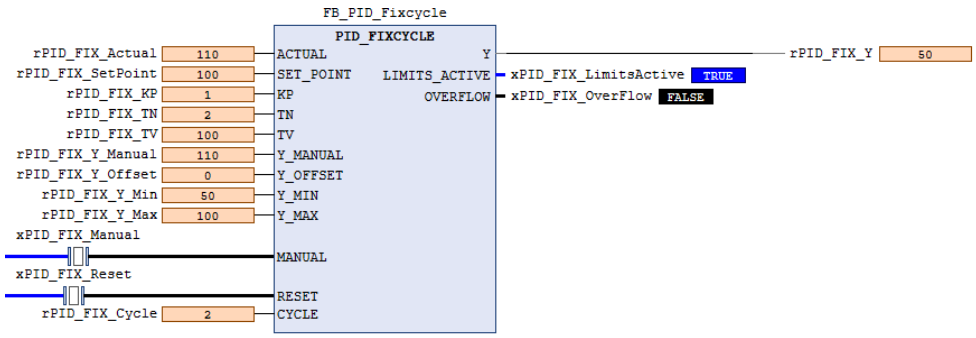
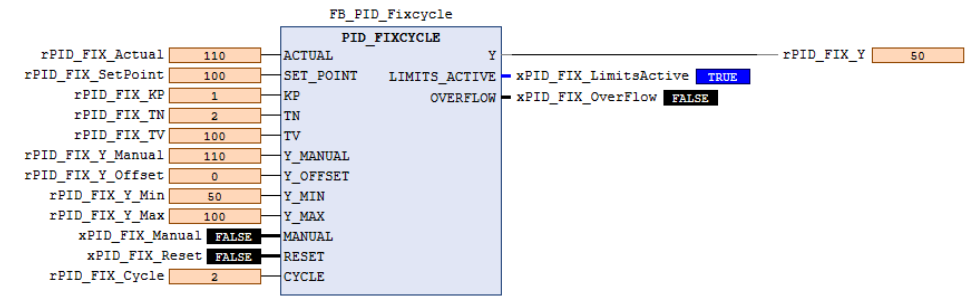
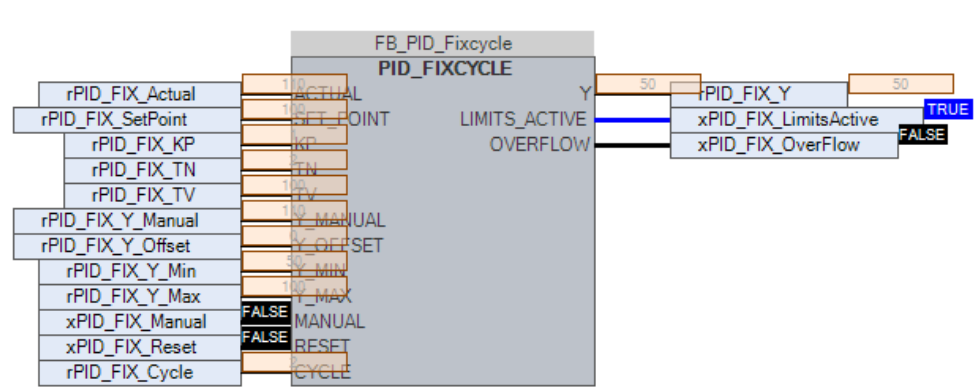
- **Programming Example**

  The variable *rPID_FIX _Y_Min*=50 and *rPID_FIX_Y_Max*=100 are the minimum and maximum threshold values of the input respectively. If *rPID_FIX_Y* reaches the limit value, the output *xPID_FIX_LimitsActive* will set to TRUE, and *rPID_FIX_Y* will be maintained within the specified range. This function block operates only when *rPID_FIX_Y_MIN* <*rPID_FIX_Y_MAX*. The variable *rPID_FIX_Actual* is the current value of the input. The variable *rPID _FIX_Y_Manual* is the manual value given as input. This is deactivated once the value of the variable *xPID_FIX_Reset* is set to FALSE.

  The value of the variable *rPID_FIX_KP*=1 is the proportionality constant. The value *rPID_FIX_TV* enables the proportional derivative controller to maintain the value within the close range of the value in *rPID_FIX_TV* variable. The value *rPID_FIX_Y_Offset* is used only in the critical condition to achieve the output within the given *rPID_Y_Min* and *rPID_FIX_Y_Max* range values respectively. The variable *rPID_FIX_TN* indicates the Reset time (t) which is the time taken by the output value (*rPID_FIX_Y*) to achieve the *rPID_FIX _SetPoint* value. One more variable is *rPID_FIX_Cycle*=2 value which sets the cycle for each sampling period of the instruction.

  The output value is calculated with the above equation and displayed as *rPID _FIX_Y*=50 along with *xPID_FIX_LimitsActive* = FALSE which means that the PID controller is trying to maintain the output value within the range between 50 and 150 which are the values of *rPID_FIX_Y_Min* and *rPID_FIX_Y_Max* respectively since the anticipated output value of *rPID_FIX_Y* is gone below 50. The variable *xPID_FIX_OverFlow* indicates the overflow in the integral part.

  Feature in the ST/LD/FBD & CFC editor:

Programming Language	Program
ST	
LD	
FBD	
CFC	

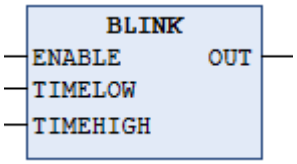
- **Supported Products**
  - AX series

- **Library**
  - Util.lib

18.15. Signal Generator Instructions

18.15.1. BLINK Function (Pulse Signal Generator)

BLINK is used to generate pulse signal value.

FB/FC	Instruction	Graphic Expression	ST Language
FB	BLINK		FB_Blink(ENABLE := , TIMELOW := , TIMEHIGH := , OUT =>);

- **Inputs**

Name	Function	Data Type	Setting Value
ENABLE	TRUE: Starts blinking. FALSE: Stops blinking whereas *OUT* keeps its value.	BOOL	TRUE, FALSE
TIMELOW	Low level time	TIME	As each data type suggested
TIMEHIGH	High level time		

- **Outputs**

Name	Function	Data Type	Setting Value
OUT	Pulse signal value	TIME	As each data type suggested.

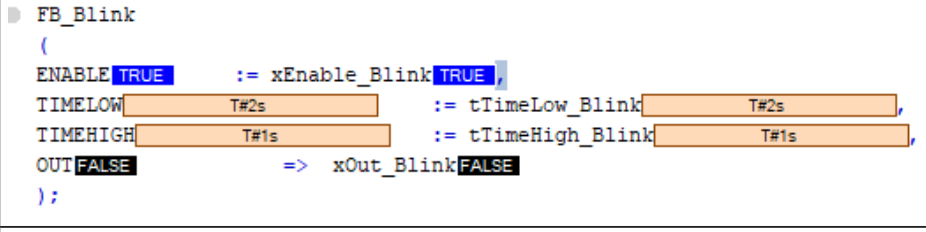
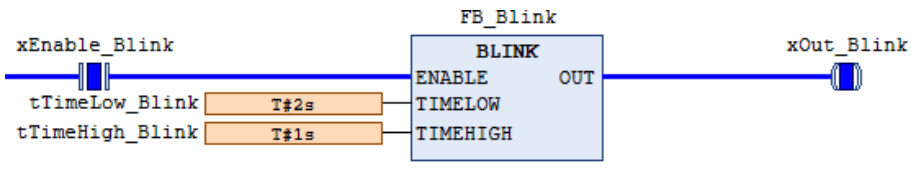
- **Function**

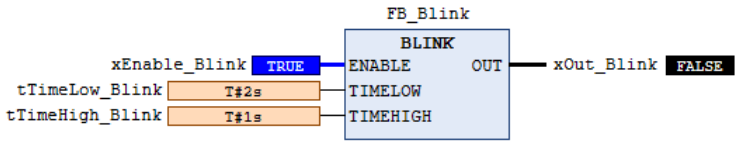
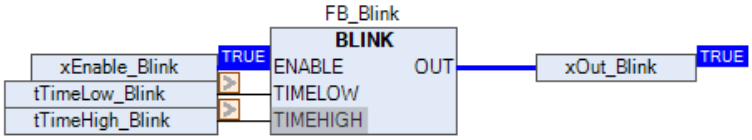
  Pulse signal is generated by this instruction. When the pulse generator starts, the output high level time is *TIMEHIGH*, and the output low level time is *TIMELOW*. The output is periodic.

- **Programming Example**

  When *xEnable_Blink*=TRUE, the *xOut_Blink* value is 2 seconds low and 1 second high, corresponding to the value of *TIMELOW* and *TIMEHIGH* variables respectively. BLINK function is used to generate pulse signal of high level time and low level time by *ENABLE* input.

  Feature in the ST/LD/FBD & CFC editor:

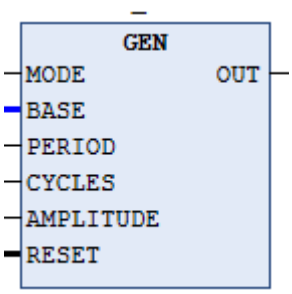
Programming Language	Program
ST	
LD	

Programming Language	Program
FBD	
CFC	

- **Supported Products**
  - AX series
- **Library**
  - Util.lib

18.15.2. GEN Function (Typical Periodic Signal Generator)

GEN generates periodic signals of different types.

FB/FC	Instruction	Graphic Expression	ST Language
FB	GEN		FB_GEN(MODE := , BASE := , PERIOD := , CYCLES := , AMPLITUDE := , OUT =>);

- **Inputs**

Name	Function	Data Type	Setting Value
MODE	Signal type	INT	−32768 to 32767
CYCLES	Number of calls per period Only active when *BASE* = TRUE.		
AMPLITUDE	Amplitude of the signal to be generated		
PERIOD	Period time Only active when *BASE* = TRUE.	TIME	Initial: TIME#1s0ms
BASE	FALSE: Period refers to calls (CYCLES). TRUE: Period refers to time (PERIOD).	BOOL	TRUE, FALSE
RESET	TRUE: Set *OUT* to zero.		

- **Outputs**

Name	Function	Data Type	Setting Value
OUT	Generated signal value	INT	−32768 to 32767

- **Function**

  This instruction is used to generate seven types of typical periodic signals: triangle wave, zero starting point triangle wave, rising sawtooth wave, descending sawtooth wave, square wave, sine wave, and cosine wave.

  The generation may be done relative to specific time base, or a given call–count base (*BASE*).

- **Programming Example**

  The *MODE* types available in the GEN function block are:

  TRIANGLE: Triangular from −AMPLITUDE to +AMPLITUDE

  TRIANGULAR_POS: Triangular from 0 to +AMPLITUDE

  SAWTOOTH_RISE: Sawtooth increasing from −AMPLITUDE to +AMPLITUDE

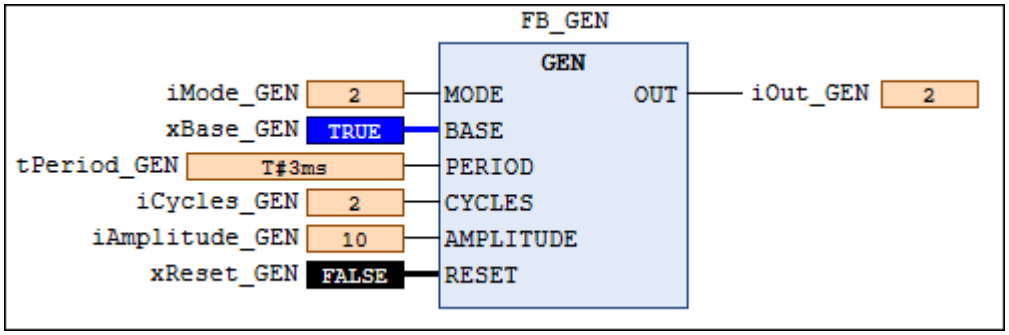
  SAWTOOTH_FALL: Sawtooth decreasing from +AMPLITUDE to–AMPLITUDE

  RECTANGLE: Rectangular switching from −AMPLITUDE to +AMPLITUDE

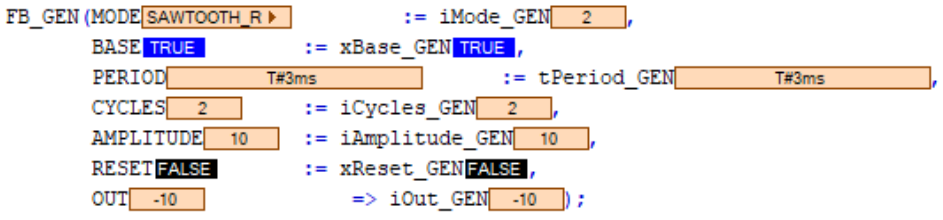
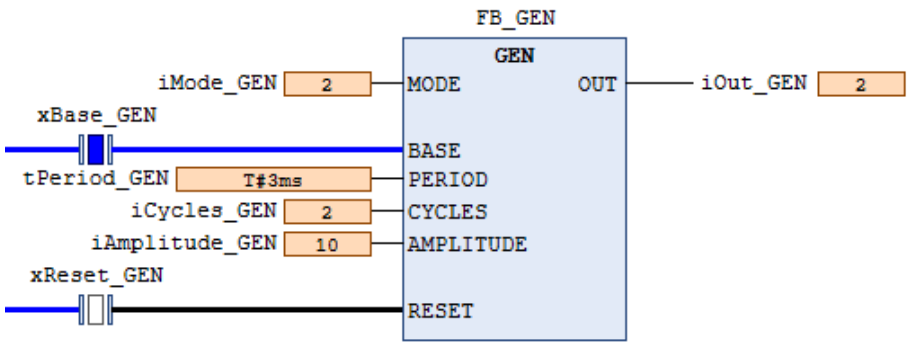
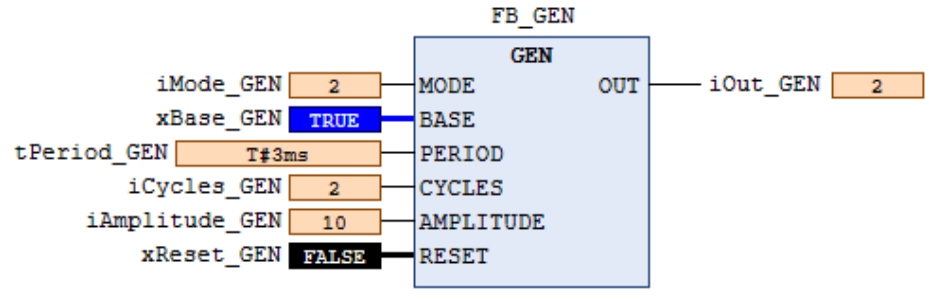
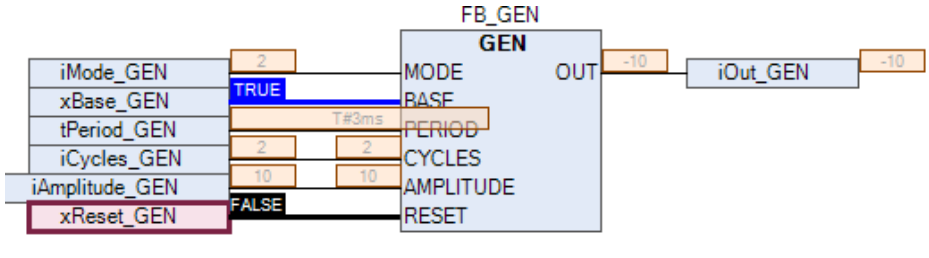
SINE: Sine

COSINE: Cosine

And these are numbered to give the input value of *iMode_GEN*, starting from 0. Here *iMode_GEN*=2 indicates it is a SAWTOOTH_RISE wave.



Feature in the ST/LD/FBD & CFC editor:

Programming Language	Program
ST	
LD	
FBD	
CFC	

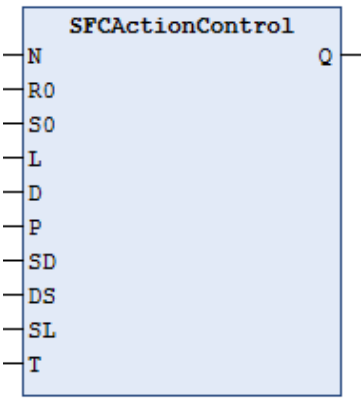
- **Supported Products**
  - AX series

- **Library**
  - Util.lib

18.16. SFC Motion Control Instruction

18.16.1. SFCActionControl Function (SFC Motion Control)

SFCActionControl is automatically integrated into the project by the SFC Plug–In.

FB/FC	Instruction	Graphic Expression	ST Language
FB	SFCActionControl		FB_SFCActionControl(N := , R0 := , S0 := , L := , D := , P := , SD := , DS := , SL := , T := , Q =>);

- **Inputs**

Name	Function	Data Type	Setting Value
N	Non stored action qualifier Only active when the step is activated.	BOOL	TRUE, FALSE
R0	Overriding reset action qualifier Action inactive		
S0	Set (stored) action qualifier When the step is activated, the action is activated. The action is continued even if the step becomes inactive. The action does not stop until it is restarted.		
L	Time limited action qualifier When the step is activated, the action is activated. The action continues until the step becomes inactive or time is up.		
D	Time delayed action qualifier TRUE: Enable time delay.		
P	Pulse action qualifier After the step is activated or deactivated, the action starts and only runs one time.		
SD	Stored and time delayed action qualifier		

Name	Function	Data Type	Setting Value
	The action starts once the time delay is activated and continues until a restart.		
DS	Delayed and stored action qualifier If the set delay time has passed and the step is still activated, the action starts and continues until a restart.		
SL	Stored and time limited action qualifier The action starts after the step is activated and continues for a certain duration or until a restart.		
T	Current time	TIME	As each data type suggested.

- **Outputs**

Name	Function	Data Type	Setting Value
Q	TRUE: run the associated action.	BOOL	TRUE, FALSE

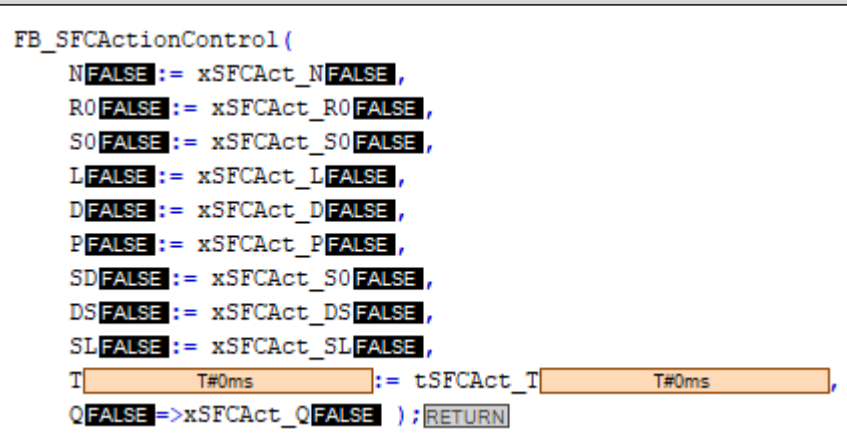
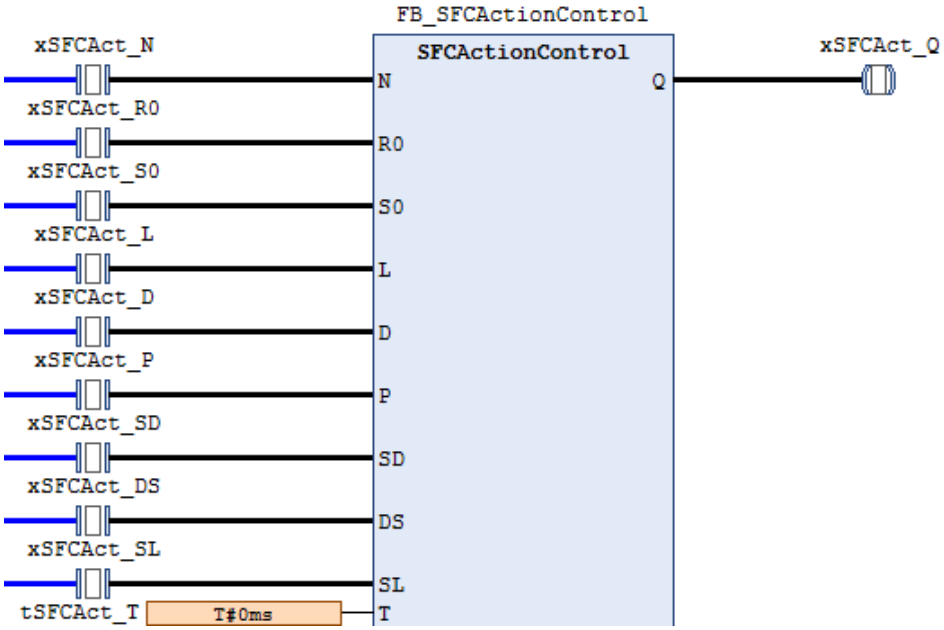
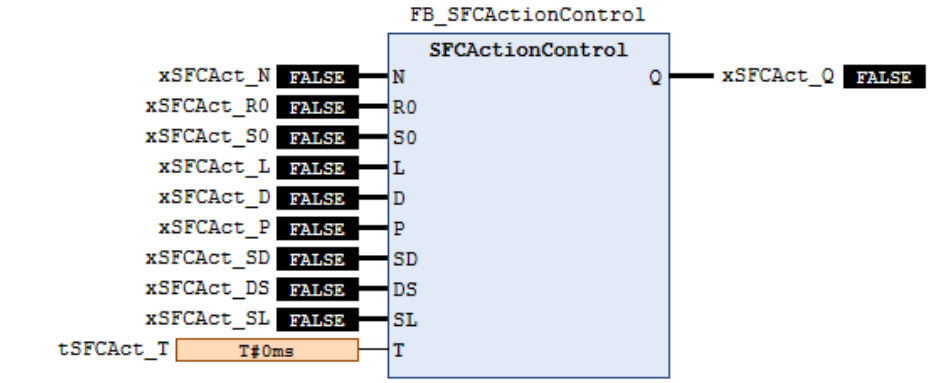
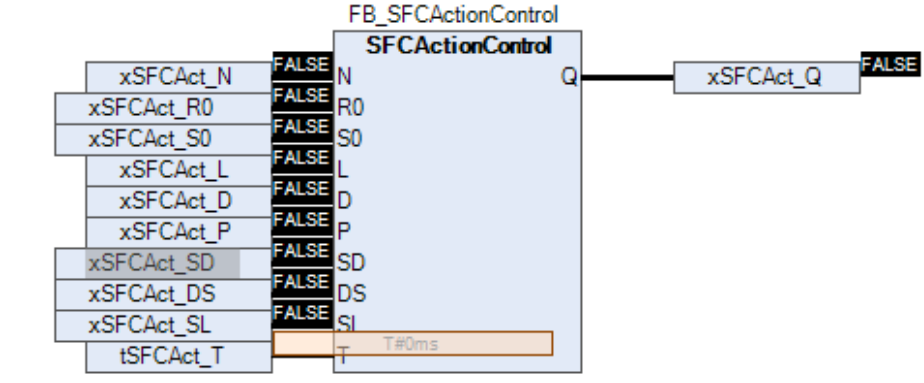
- **Function**

  This function block is automatically integrated into the project by the SFC Plug–In.

- **Programming Example**

  In the below example, the variable *xSFCAct_N* indicates that the action is activated as long as the step is active. The variable *xSFCAct_R0* indicates that the action is deactivated. The variable *xSFCAct_S0* indicates that this action is run as soon as the step is active. The action is continued even if the step becomes inactive. The action does not stop until it is restarted. The variable *xSFCAct_L* indicates that the action is activated as the step is active. The action continues until the step is deactivated or the given time span has elapsed. The variable *xSFCAct_D* indicates that the action is activated only when the given delay time has elapsed and the step is still active. The action does not stop until the step is deactivated. The variable *xSFCAct_P* runs the action for exactly two times: One is when the step is activated, and the other time is when the step is deactivated. The variable *xSFCAct_SD* indicates that the action is activated only after the given delay time has elapsed and the step is active. The action is run until it gets reset. The variable *xSFCAct_DS* indicates that the action is activated only when the given delay time has elapsed and the step is still active. The action is run until it gets reset. The variable *xSFCAct_SL* indicates that the action is activated as soon as the step is activated. The action is run until the specified time has elapsed or it gets reset. Finally, the output of Bool type is displayed in the variable *xSFCAct_Q*.

  Feature in the ST/LD/FBD & CFC editor:

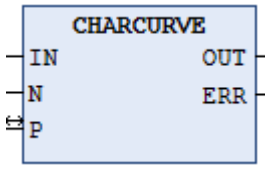
Programming Language	Program
ST	
LD	
FBD	
CFC	

- **Supported Products**
  - AX series
- **Library**
  - Iecsfc.lib

18.17. Manipulator Instructions

18.17.1. CHARCURVE (Characteristic Curve)

CHARCURVE maps an input signal onto a characteristic curve.

FB/FC	Instruction	Graphic Expression	ST Language
FB	CHARCURVE		FB_CHARCURVE(IN := , N := , P := , OUT => , ERR =>)

- **Inputs**

Name	Function	Data Type	Setting Value
IN	The value of the X-axis coordinate	INT	−32768 to 32767
N	The number of points in the array used for the curve (2 to 11)	BYTE	0 to 255
P	Define the characteristic curve on the XY coordinate system.	ARRAY [0..10] OF POINT	As each data type suggested.

- **Outputs**

Name	Function	Data Type	Setting Value
ERR	Error 0: No errors 1: Errors in the X value of *P*, wrong sequence 2: *IN* is out of the range of *P*. 3: *N* is invalid; the number of points is not within the allowed range of 2 to 11.	BYTE	0 to 255
OUT	The value of the Y-axis corresponding to the curve on the coordinate system	INT	−32768 to 32767

- **Function**

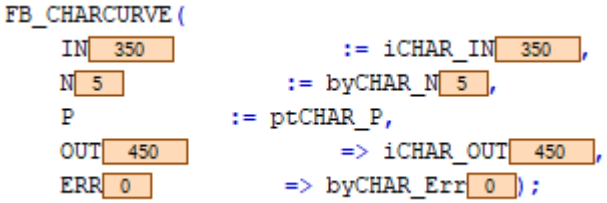
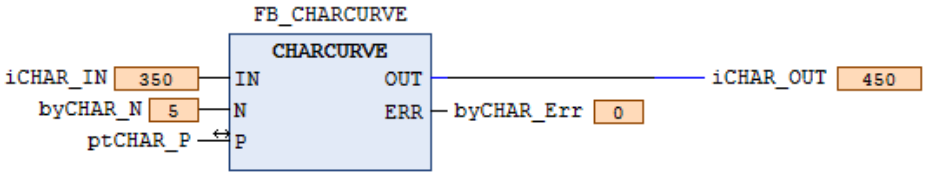
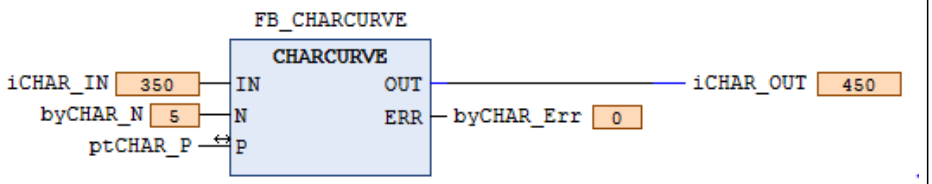
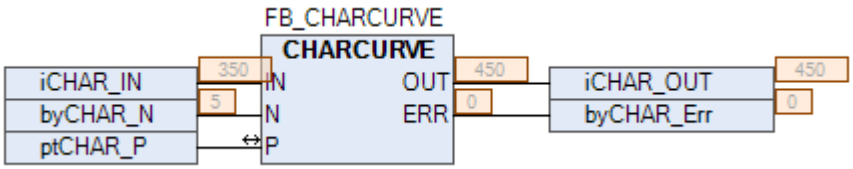
  The input point type array[0..10] defines a curve on the XY coordinate. The characteristic curve is given by an array of points, which include a set of X-values with their corresponding Y-values. Entering the point of the X-axis specified by the input *IN* outputs the corresponding Y-axis value of the curve on the coordinate system (*OUT*).

- **Programming Example**

  There are a maximum of 11 array values, each with (x, y) coordinate values. These values of array p are p[0]=(x=150, y=250), p[1]=(x=200, y=300), p[2]=(x=250, y=350), p[3]=(x=300, y=400), p[4]=(x=350, y=450). Here the variable *iCHAR_IN*=350 is the corresponding value of X–axis and the variable *byCHAR_N*=5

indicates the array with five (x, y) values as given above. The output variable *iCHAR_OUT*=450 gives the corresponding Y value for the input variable *iCHAR_IN* specified. The variable *byCHAR_Err*=0 indicates that there are no errors.

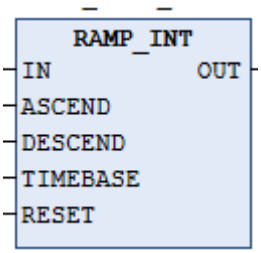
Feature in the ST/LD/FBD & CFC editor:

Programming Language	Program
ST	
LD	
FBD	
CFC	

- **Supported Products**
  - AX series
- **Library**
  - Util.lib

18.17.2. RAMP_INT (Integer Speed Limit)

RAMP_INT limits the slope of an INT value to a certain value.

FB/FC	Instruction	Graphic Expression	ST Language
FB	RAMP_INT		FB_RAMP_INT(IN := , ASCEND := , DESCEND := , TIMEBASE := , RESET := , OUT =>);

- **Inputs**

Name	Function	Data Type	Setting Value
IN	Input value If the current input value is greater than the previous value, addition is performed according to the set *ASCEND*, and the result is output by *OUT*. If the current input value is less than the previous value, subtraction is performed according to the set *DESCEND*, and the result is output by *OUT*.	INT	−32768 to 32767
ASCEND	Limitation of acceleration: Maximum ascent per time base		
DESCEND	Limitation of deceleration: Maximum descent per time base		
TIMEBASE	Time reference for *ASCEND*/*DESCEND* t#0s : *ASCEND*/*DESCEND* is defined per call. else : *ASCEND*/*DESCEND* is defined per specified time. If *TIMEBASE* = t#0s, then the time base is equal to the task cycle time. In this case, the limitation refers to a task cycle.	TIME	As each data type suggested.
RESET	Reset the function block. TRUE: Stops the internal calculation and reinitializes the function block. The last calculated output value in *OUT* is maintained in order to start the internal calculation with it at the next restart of the function block. FALSE: Outputs the smoothed input signal at output *OUT*.	BOOL	TRUE, FALSE

- **Outputs**

Name	Function	Data Type	Setting Value
OUT	Output value	INT	−32768 to 32767

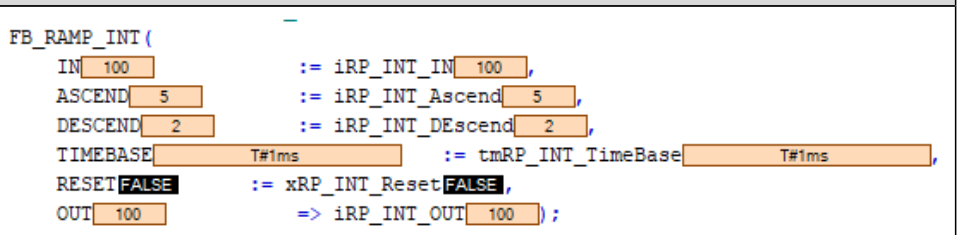
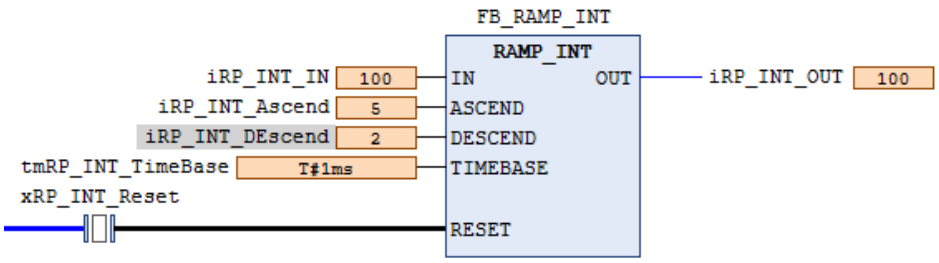
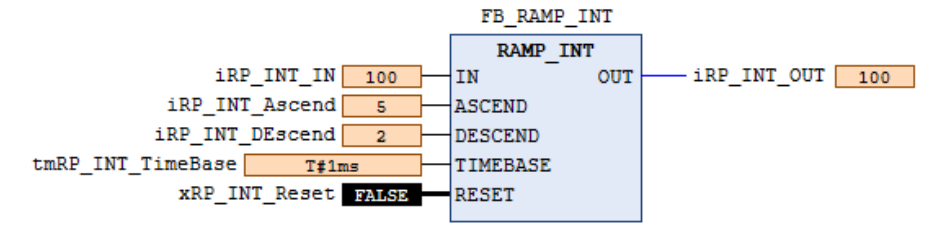
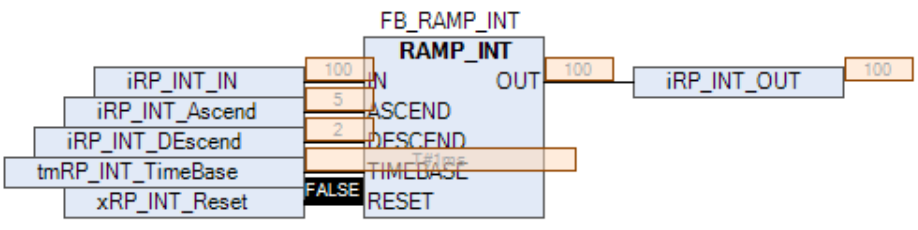
- **Function**

  This function block is used as a limit to the rising and falling speed of the integer input.

- **Programming Example**

  In the below example, the variable *iRP_INT_IN*=100 is the Integer input of the RAMP_INT function block. The variable *iRP_INT_Ascend*=5 indicates the maximum positive slope. The variable *iRP_INT_Descend*=2 indicates the maximum negative slope. The variable *xRP_INT_Reset* is used to reset the function block. The variable *tmRP_INT_TimeBase*=1ms is used as a reference to maximum positive/negative slopes. The variable *iRP_INT_OUT*=100 is the output obtained for the RAMP_INT function.

  Feature in the ST/LD/FBD & CFC editor:

Programming Language	Program
ST	
LD	
FBD	
CFC	

- **Supported Products**

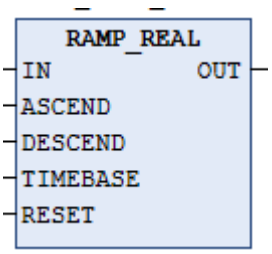
  - AX series

- **Library**

  - Util.lib

18.17.3. RAMP_REAL (Real Speed Limit)

RAMP_REAL limits the slope of a REAL value to a certain value.

FB/FC	Instruction	Graphic Expression	ST Language
FB	RAMP_REAL		FB_RAMP_REAL(IN := , ASCEND := , DESCEND := , TIMEBASE := , RESET := , OUT =>);

- **Inputs**

Name	Function	Data Type	Setting Value
IN	Input value If the current input value is greater than the previous value, addition is performed according to the set *ASCEND*, and the result is output by *OUT*. If the current input value is less than the previous value, subtraction is performed according to the set *DESCEND*, and the result is output by *OUT*.	REAL	1.0E–44 to 3.402823E+38
ASCEND	Limitation of acceleration: Maximum ascent per time base		
DESCEND	Limitation of deceleration: Maximum descent per time base		
TIMEBASE	Time reference for *ASCEND*/*DESCEND* t#0s : *ASCEND*/*DESCEND* is defined per call. else : *ASCEND*/*DESCEND* is defined per specified time. If *TIMEBASE* = t#0s, then the time base is equal to the task cycle time. In this case, the limitation refers to a task cycle.	TIME	As each data type suggested.
RESET	TRUE: Reset the function block.	BOOL	TRUE, FALSE

- **Outputs**

Name	Function	Data Type	Setting Value
OUT	Output value	REAL	1.0E–44 to 3.402823E+38

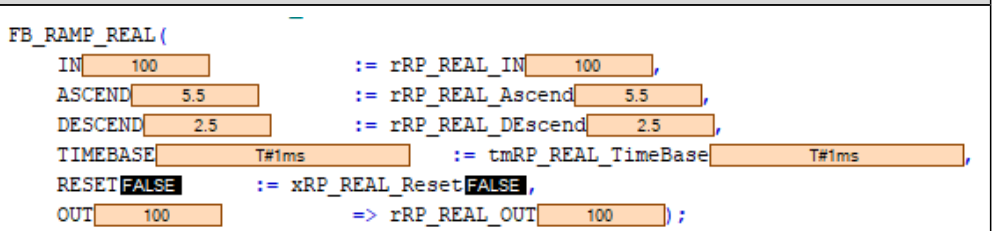
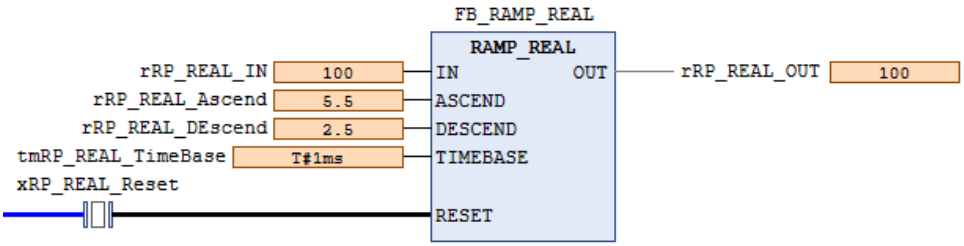
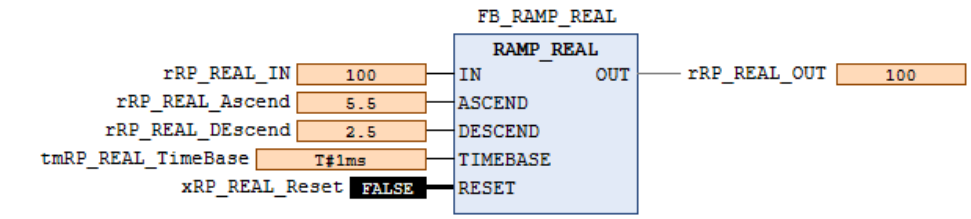
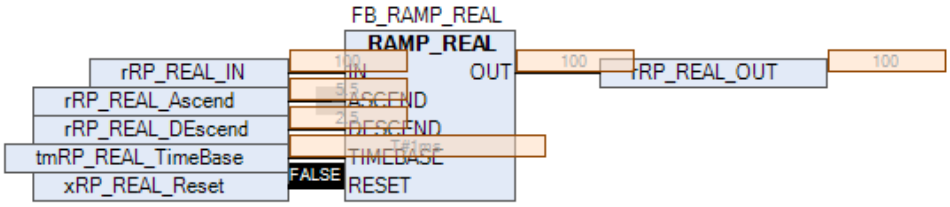
- **Function**

  This function block is used as a limit to the rising and falling speed of the Real input.

- **Programming Example**

  In the below example, the variable *rRP_REAL_IN*=100 is the REAL input of the RAMP_REAL function block. The variable *rRP_REAL_Ascend*=5.5 indicates the maximum positive slope. The variable *rRP_REAL_Descend*=2.5 indicates the maximum negative slope. The variable *xRP_REAL_Reset* is used to reset the function block. The variable *tmRP_REAL_TimeBase*=1ms is used as a reference to maximum positive/negative slopes. The variable *rRP_REAL_OUT*=100 is the output obtained for the RAMP_REAL function.

  Feature in the ST/LD/FBD & CFC editor:

Programming Language	Program
ST	
LD	
FBD	
CFC	

- **Supported Products**

  - AX series

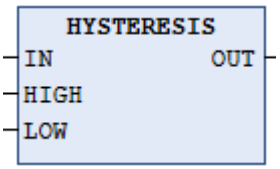
- **Library**

  - Util.lib

18.18. Analog Processing Instructions

18.18.1. HYSTERESIS Function (Lag Function)

HYSTERESIS realizes a Boolean hysteresis on different INT values.

FB/FC	Instruction	Graphic Expression	ST Language
FB	HYSTERESIS		FB_HYSTERESIS(IN:= , HIGH := , LOW := , OUT =>);

- **Inputs**

Name	Function	Data Type	Setting Value
IN	Input value	INT	−32768 to 32767
HIGH	Upper limit		
LOW	Lower limit		

- **Outputs**

Name	Function	Data Type	Setting Value
OUT	TRUE: The input is less than the lower limit. FALSE: The input is greater than the upper limit.	BOOL	TRUE, FALSE

- **Function**

  The input of this instruction includes three INT type values *IN*, *HIGH*, and *LOW*. If *IN* is less than the lower limit *LOW*, *OUT* is TRUE and remains so until *IN* exceeds the upper limit *HIGH*, at which point it becomes FALSE. *OUT* stays FALSE until *IN* is less than the lower limit *LOW* again, at which point it becomes TRUE. This process repeats in a loop.

- **Programming Example**

  In the below example, the output is decided according to the following conditions:

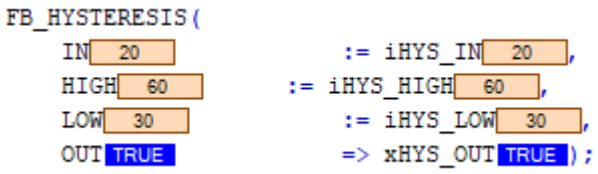
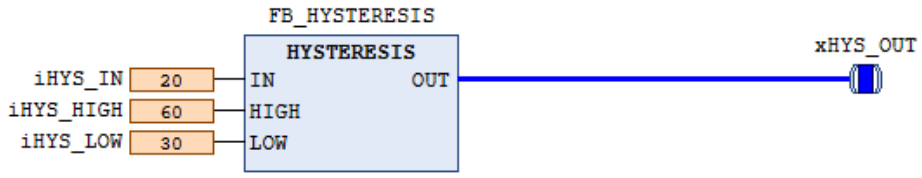
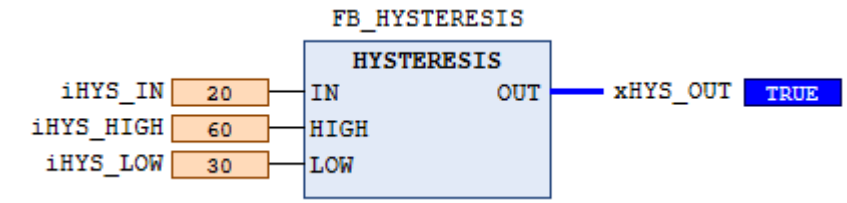
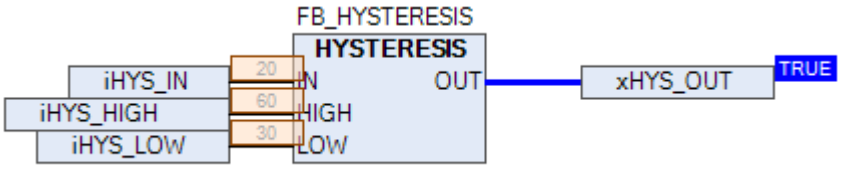
  TRUE: Input *iHYS_IN* is less than input *iHYS_LOW*.

  FALSE: Input *iHYS_IN* is greater than input *iHYS_HIGH*.

  The variable *iHYS_IN*=20 is the input value for the given function block and the output is variable *xHYS_OUT*=TRUE.

  iHYS_IN and *iHYS_HIGH* are used to display the lowest and highest values of the Hysteresis function.

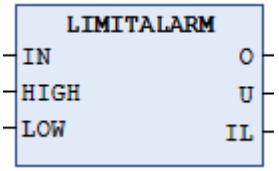
  Feature in the ST/LD/FBD & CFC editor:

Programming Language	Program
ST	
LD	
FBD	
CFC	

- **Supported Products**
  - AX series
- **Library**
  - Util.lib

18.18.2. LIMITALARM Function (Upper and Lower Limit Alarm Function)

LIMITALARM monitors if any input value is between the lower and upper limit and output the value if it is out of range.

FB/FC	Instruction	Graphic Expression	ST Language
FB	LIMITALARM		FB_LIMITALARM(IN := , HIGH := , LOW := , O => , U => , IL =>);

- **Inputs**

Name	Function	Data Type	Setting Value
IN	Input value	INT	−32768 to 32767
HIGH	Upper limit		
LOW	Lower limit		

- **Outputs**

Name	Function	Data Type	Setting Value
O, U, IL	Output values	BOOL	TRUE, FALSE

- **Function**

  This function block monitors if any input value is between lower and upper limit. The input of this instruction includes three INT type values *IN*, *HIGH*, and *LOW*. If *IN* exceeds the upper limit *HIGH*, then *O* is TRUE, and *U* and *IL* are FALSE. If *IN* is less than the lower limit *LOW*, then *U* is TRUE, and *O* and *IL* are FALSE. If *IN* is between the lower limit *LOW* and the upper limit *HIGH*, then *IL* is TRUE, and *O* and *U* are FALSE.

- **Programming Example**

  In the below example, the variable *iLIM_HIGH*=60 specifies the highest limit value and the variable *iLIM_LOW*=30 specifies the lowest limit value. So, the respective output will be:

  xLIM_O: TRUE: *IN* > *HIGH*

  FALSE: Else

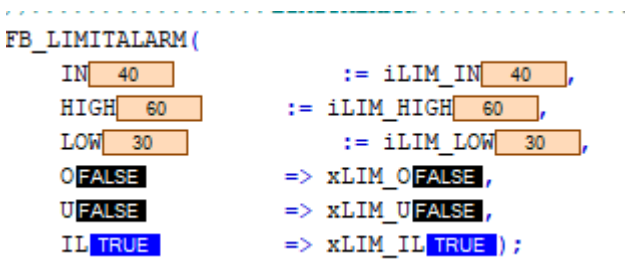
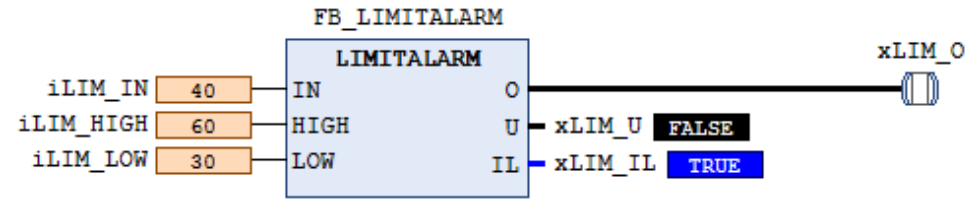
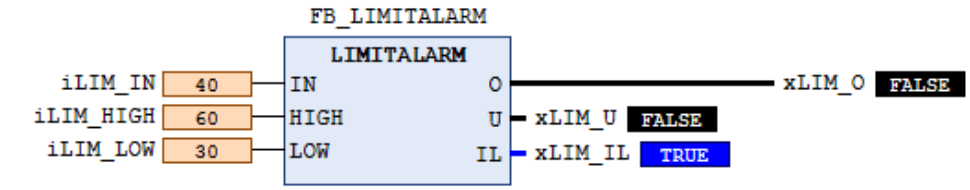
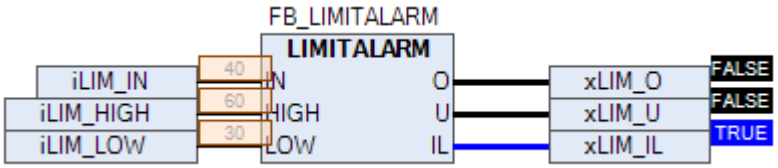
  xLIM_U: TRUE: *IN* < *LOW*

  FALSE: Else

  xLIM_IL: TRUE: *xLIM_O* and *xLIM_U* are FALSE.

  For the given input value *iLIM_IN*=40, the *xLIM_IL* value is TRUE since *xLIM_O* and *xLIM_U* are FALSE according to the above condition.

  Feature in the ST/LD/FBD & CFC editor:

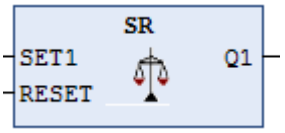
Programming Language	Program
ST	
LD	
FBD	
CFC	

- **Supported Products**
  - AX series
- **Library**
  - Util.lib

18.19. Bistable Instructions

18.19.1. SR Function (Set Dominant Bistable Function)

SR realizes a bistable SET dominant latch.

FB/FC	Instruction	Graphic Expression	ST Language
FB	SR		FB_SR(SET1 := , RESET := , Q1 =>);

- **Inputs**

Name	Function	Data Type	Setting Value
SET1	Set signal Rising edge: Set *Q1* to TRUE (prior).	BOOL	TRUE, FALSE
RESET	Reset signal Rising edge: Reset *Q1* to FALSE.		

- **Outputs**

Name	Function	Data Type	Setting Value
Q1	Q1=(NOT *RESET* AND *Q1*) OR *SET1*	BOOL	TRUE, FALSE

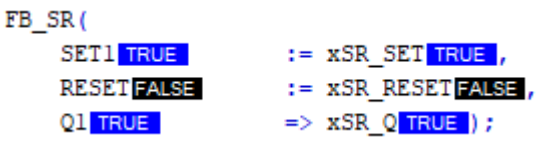
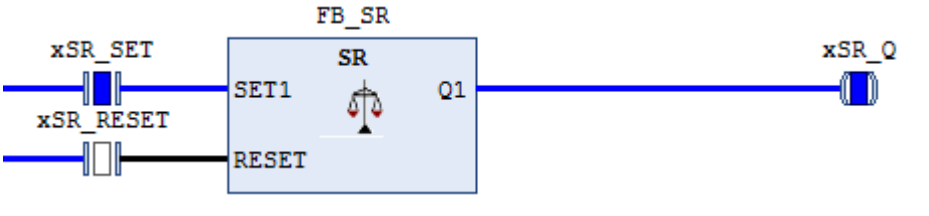
- **Function**

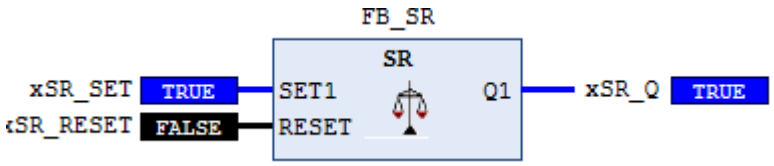
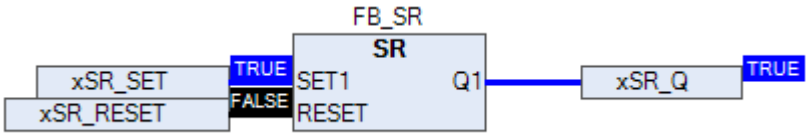
  This instruction realizes a bistable SET dominant latch.

- **Programming Example**

  In the below example, the variable *xSR_SET*=TRUE is the Set input value and *xSR_RESET*=FALSE is the Reset input value given to the function block. Both are BOOL type. Thus, the output is *xSR_Q*=FALSE which is also BOOL type. The output is decided by the value of the input variable *xSR_SET* which has high priority over the *xSR_RESET* value.

  Feature in the ST/LD/FBD & CFC editor:

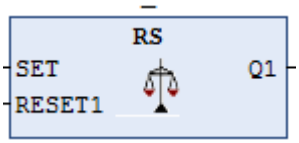
Programming Language	Program
ST	
LD	

Programming Language	Program
FBD	
CFC	

- **Supported Products**
  - AX series
- **Library**
  - Standard.lib

18.19.2. RS Function (Reset Dominant Bistable Function)

RS realizes a bistable RESET dominant latch.

FB/FC	Instruction	Graphic Expression	ST Language
FB	RS		FB_RS(SET := , RESET1 := , Q1 =>);

- **Inputs**

Name	Function	Data Type	Setting Value
SET	Set signal Rising edge: Set *Q1* to TRUE.	BOOL	TRUE, FALSE
RESET1	Reset signal Rising edge: Reset *Q1* to FALSE (prior).		

- **Outputs**

Name	Function	Data Type	Setting Value
Q	Q1=NOT *RESET1* AND (*Q1* OR *SET*)	BOOL	TRUE, FALSE

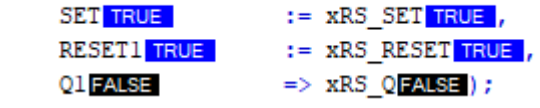
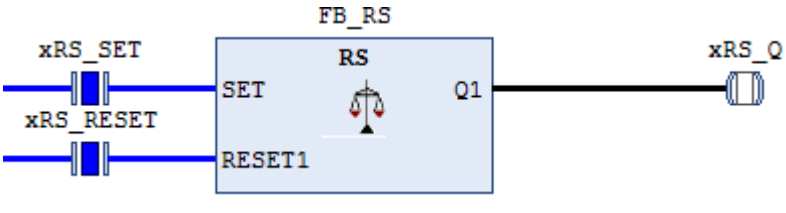
- **Function**

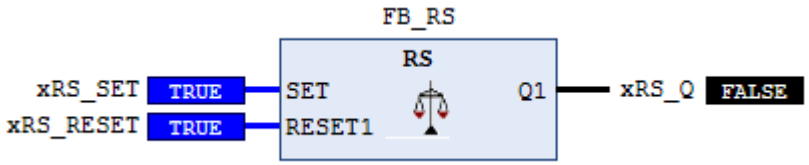
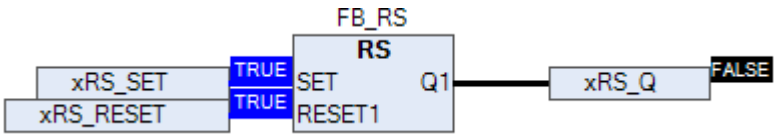
  This instruction realizes a bistable RESET dominant latch.

- **Programming Example**

  In the below example, the variable *x RS_SET*=TRUE is the Set input value and *xRS_RESET*=TRUE is the Reset input value given to the function block. Both of which are BOOL type. Thus, the output is *xRS_Q*=FALSE which is also BOOL type. The output is decided by the value of the input variable *xRS_RESET* which has high priority over the *xRS_SET* input value. When *xRS_RESET*=TRUE, whatever we reset the value using *xRS_SET* =TRUE, the output *xRS_Q* is always FALSE.

  Feature in the ST/LD/FBD & CFC editor:

Programming Language	Program
ST	
LD	

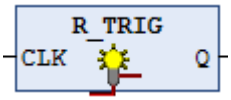
Programming Language	Program
FBD	
CFC	

- **Supported Products**
  - AX series
- **Library**
  - Standard.lib

18.20. Trigger Instructions

18.20.1. R_TRIG Function (Rising Edge Detection Function)

R_TRIG is used to detect a rising edge of a Boolean variable.

FB/FC	Instruction	Graphic Expression	ST Language
FB	R_TRIG		FB_R_Trige(CLK := , Q =>);

- **Inputs**

Name	Function	Data Type	Setting Value
CLK	Detect the rising edge.	BOOL	TRUE, FALSE

- **Outputs**

Name	Function	Data Type	Setting Value
Q	TRUE: When *CLK* is TRUE	BOOL	TRUE, FALSE

- **Function**

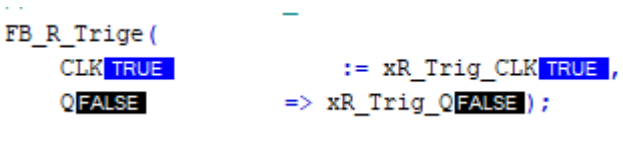
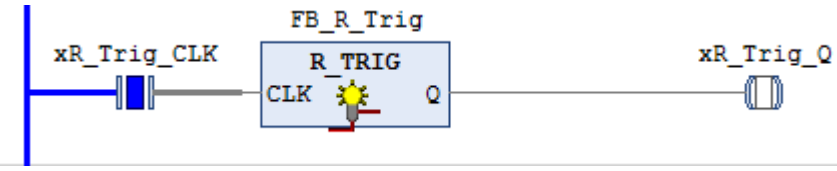
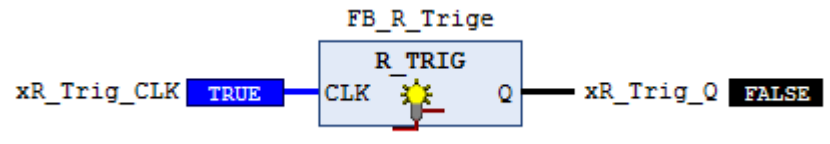
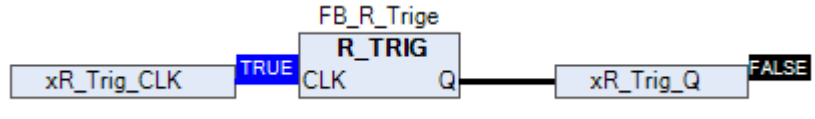
  This instruction is used to detect a rising edge of a Boolean function.

- **Programming Example**

  In the below example, *xR_Trig_CLK* is an intermediate variable with an initial value of TRUE. The output value *xR_Trig_Q* returns FALSE. But *xR_Trig_Q* is TRUE when the CLK variable *xR_Trig_CLK* detects a rising edge (FALSE–>

  TRUE).

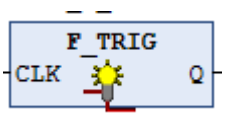
  Feature in the ST/LD/FBD & CFC editor:

Programming Language	Program
ST	
LD	
FBD	
CFC	

- **Supported Products**
  - AX series
- **Library**
  - Standard.lib

18.20.2. F_TRIG Function (Falling Edge Detection Function)

F_TRIG is used to detect a falling edge of a Boolean variable.

FB/FC	Instruction	Graphic Expression	ST Language
FB	F_TRIG		FB_F_Trige(CLK := , Q =>);

- **Inputs**

Name	Function	Data Type	Setting Value
CLK	Detect the falling edge.	BOOL	TRUE, FALSE

- **Outputs**

Name	Function	Data Type	Setting Value
Q	TRUE: When *CLK* is TRUE	BOOL	TRUE, FALSE

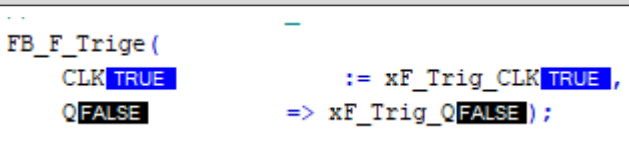
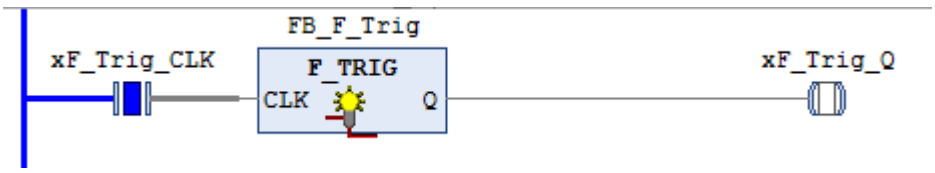
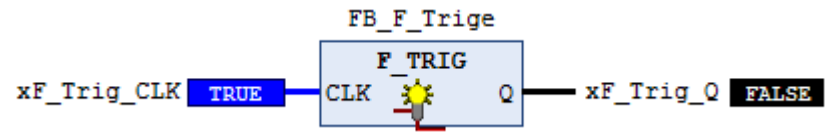
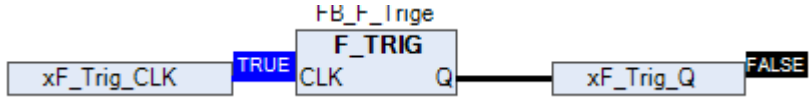
- **Function**

  This instruction is used to detect a falling edge of a Boolean function.

- **Programming Example**

  In the below example, the input variable *x F_Trig_CLK* is an intermediate variable with an initial value of TRUE. The output value of *xF_Trig_Q* returns FALSE. But *xF_Trig_Q* is TRUE when the variable *xF_Trig_CLK* detects a falling edge (TRUE–>FALSE).

  Feature in the ST/LD/FBD & CFC editor:

Programming Language	Program
ST	
LD	
FBD	
CFC	

- **Supported Products**

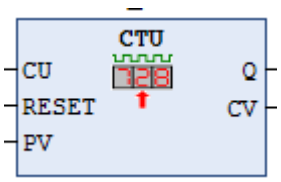
  - AX series

- **Library**
  - Standard.lib

18.21. Counter Instructions

18.21.1. CTU Function (Count Up)

CTU increments from 0 to a given input value.

FB/FC	Instruction	Graphic Expression	ST Language
FB	CTU		FB_CTU(CU := , RESET := , PV := , Q => , CV =>);

- **Inputs**

Name	Function	Data Type	Setting Value
CU	Counter input TRUE: Rising edge detected, increment *CV* by one. FALSE: Falling edge detected, hold the counter value at the same value.	BOOL	TRUE, FALSE
RESET	TRUE: Reset *CV* to 0.		
PV	Maximum value of the counter	WORD	0 to 65535

- **Outputs**

Name	Function	Data Type	Setting Value
Q	Indicates whether the instruction has resulted in a number greater than or equal to the maximum value of the counter. TRUE: *CV* >= *PV* FALSE: *CV* < *PV*	BOOL	TRUE, FALSE
CV	Current counter result	WORD	0 to 65535

- **Function**

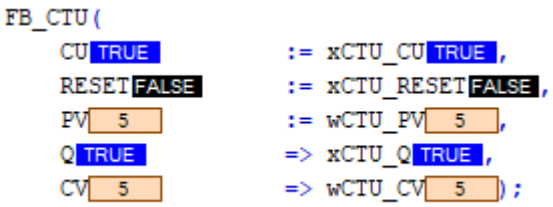
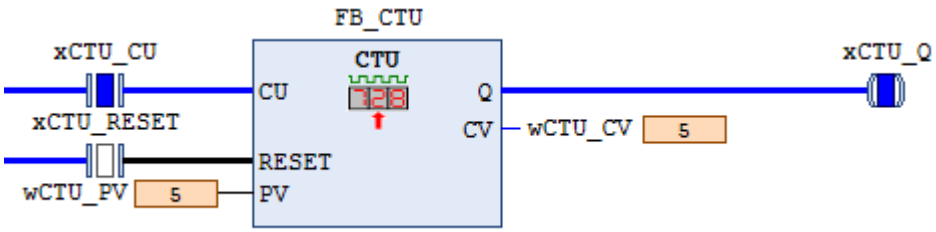
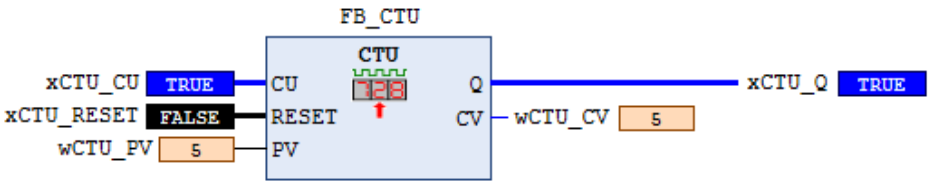
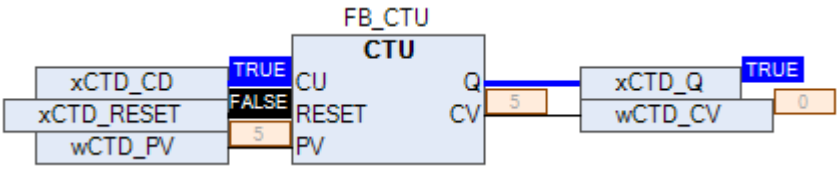
  This function increments a given input value by one.

- **Programming Example**

  In the below example, the input variable *wCTU_PV* is used to specify the target value 5. When the target of 5 is achieved in the output variable *wCTU_CV*, the output variable *xCTU_Q* will be TRUE. The input variable *xCTU_RESET* is used to reset the output variable *wCTU_CV* to its initial value of 0.

The output variable *wCTU_CV* is incremented each time only when the input value of the variable *xCTU_CU* is TRUE considered that it becomes FALSE successively. For example, TRUE–>FALSE–>TRUE, the value of *wCTU_CV* is incremented by 1 each time.

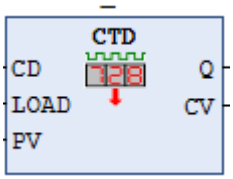
Feature in the ST/LD/FBD & CFC editor:

Programming Language	Program
ST	
LD	
FBD	
CFC	

- **Supported Products**
  - AX series
- **Library**
  - Standard.lib

18.21.2. CTD Function (Count Down)

CTD decrements a given input value to 0.

FB/FC	Instruction	Graphic Expression	ST Language
FB	CTD		FB_CTD(CD := , LOAD := , PV := , Q => , CV =>);

- **Inputs**

Name	Function	Data Type	Setting Value
CD	Counter input TRUE: Rising edge detected, decrement *CV* by one. FALSE: Falling edge detected, hold the counter value at the same value.	BOOL	TRUE, FALSE
Load	TRUE: Set *CV* = *PV*.		
PV	Maximum value of the counter	WORD	0 to 65535

- **Outputs**

Name	Function	Data Type	Setting Value
Q	Indicates whether the instruction has resulted in a number less than or equal to the maximum value of the counter. TRUE: *CV* <= *PV* FALSE: *CV* > *PV*	BOOL	TRUE, FALSE
CV	Current counter result	WORD	0 to 65535

- **Function**

  This function decrements a given input value by one.

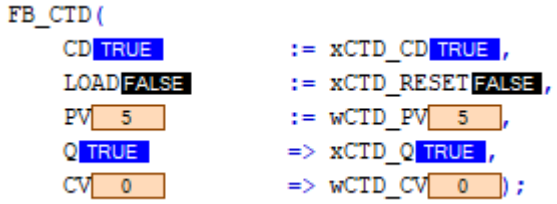
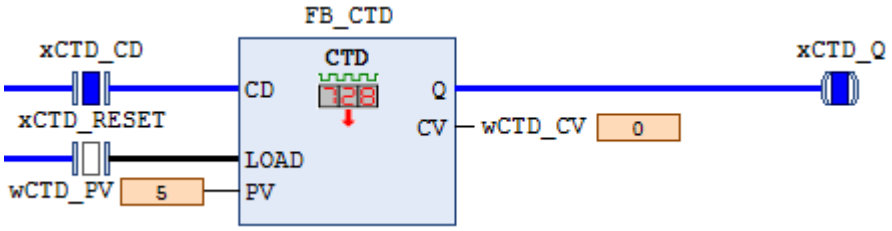
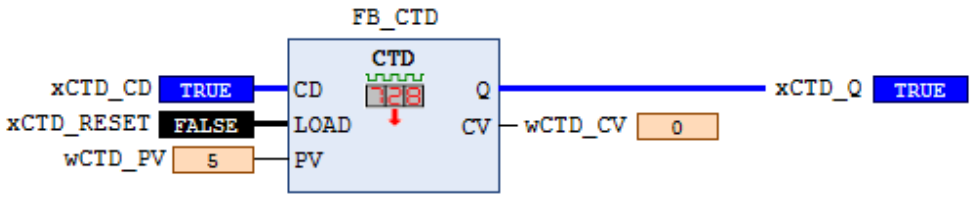
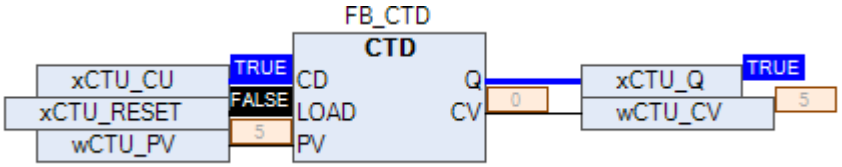
- **Programming Example**

  In the below example, the input variable *wCTD_PV* is used to specify the target value 5. When the target of 0 is achieved in the output variable *wCTD_CV*, the output variable *xCTD_Q* becomes TRUE. The input variable *xCTD_RESET* is used to reset the output variable *wCTD_CV* to its initial value of 5.

  The output variable *wCTD_CV* is decremented each time only when the input value of the variable *xCTD_CD* is TRUE considered that it becomes FALSE successively. For example, TRUE–>

FALSE–>TRUE, the value of *wCTD_CV* is decremented by 1 each time.

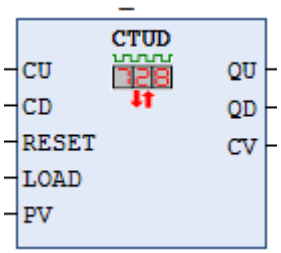
  Feature in the ST/LD/FBD & CFC editor:

Programming Language	Program
ST	
LD	
FBD	
CFC	

- **Supported Products**
  - AX series
- **Library**
  - Standard.lib

18.21.3. CTUD Function (Count Up/Count Down)

CTUD increments and decrement a given input value.

FB/FC	Instruction	Graphic Expression	ST Language
FB	CTUD		FB_CTUD(CU := , CD := , RESET := , LOAD := , PV := , QU => , QD => , CV =>);

- **Inputs**

Name	Function	Data Type	Setting Value
CU	Counter input TRUE: Rising edge detected, increment *CV* by one.	BOOL	TRUE, FALSE
CD	Counter input TRUE: Rising edge detected, decrement *CV* by one.		
RESET	TRUE: Reset *CV* to 0.		
LOAD	TRUE: Set *CV* = *PV*.		
PV	Maximum value of the counter	WORD	0 to 65535

- **Outputs**

Name	Function	Data Type	Setting Value
QU	Overflow TRUE: When *CV* >= *PV*	BOOL	TRUE, FALSE
QD	Underflow TRUE: When *CV* = 0		
CV	Current counter result	WORD	0 to 65535

- **Function**

  This function block increments and decrements a given input value by one.

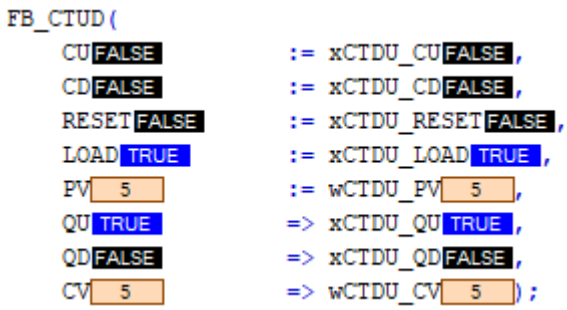
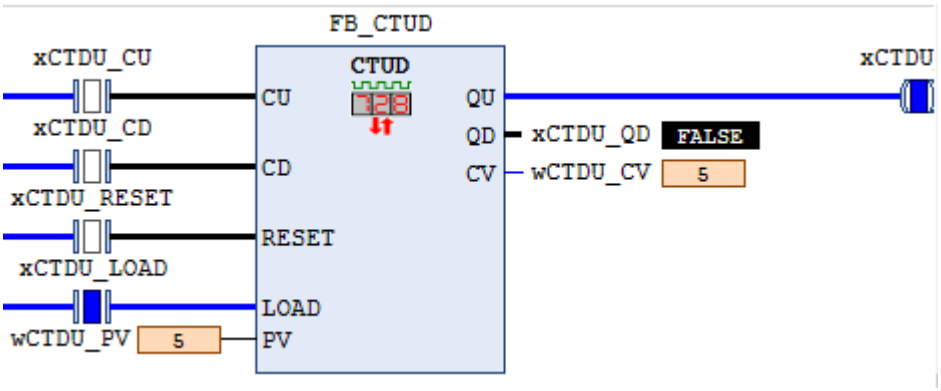
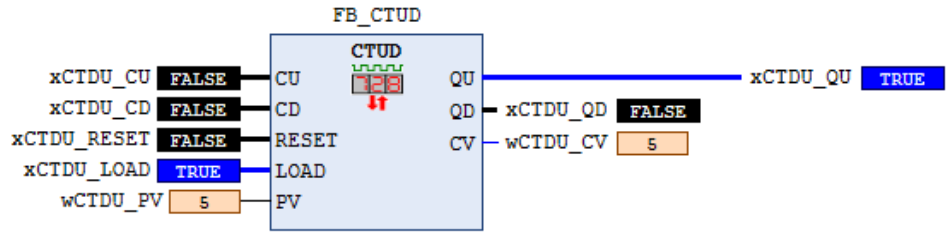
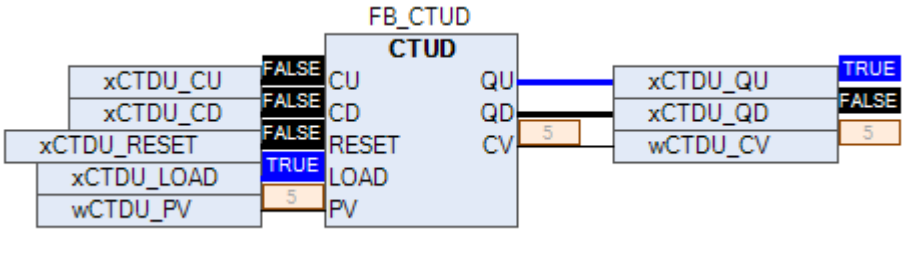
- **Programming Example**

  In the below example, the input variable *wCTDU_PV* is used to specify the target value 5. When the target of 5 is achieved in the output variable *wCTDU_CV*, the output variable *xCTDU_QU* will be TRUE.

If the target of 0 is achieved in the output variable *wCTDU_CV*, the output variable *xCTDU_QD* will be TRUE. The input variable *xCTDU_RESET* or *xCTDU_LOAD* is used to reset the output variable. If RESET is valid, *wCTDU_CV* will be initialized to 0; if *xCTDU_LOAD*=TRUE is valid, *wCTDU_CV* will be initialized to *wCTDU_PV*.

The output variable *wCTDU_CV* is incremented or decremented each time only when the input value of the variable *xCTDU_CU* or *xCTDU_CD* is TRUE considered that it becomes FALSE successively. For example, TRUE–>FALSE –>TRUE, the value of *wCTDU_CV* is incremented or decremented by 1 each time.

Feature in the ST/LD/FBD & CFC editor:

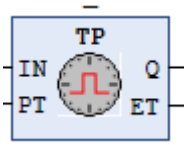
Programming Language	Program
ST	
LD	
FBD	
CFC	

- **Supported Products**
  - AX series
- **Library**
  - Standard.lib

18.22. Timer Instructions

18.22.1. TP Function (Pulse Timer)

TP implements the pulse timer.

FB/FC	Instruction	Graphic Expression	ST Language
FB	TP		FB_TP(IN := , PT := , Q => , ET =>);

- **Inputs**

Name	Function	Data Type	Setting Value
IN	If rising edge, starts increasing the internal timer until *ET=PT*.	BOOL	TRUE, FALSE
PT	Length of the pulse (high–signal)	TIME	As each data type suggested.

- **Outputs**

Name	Function	Data Type	Setting Value
Q	When *IN* is TRUE and *ET* is less than or equal to *PT*, *Q* is TRUE; otherwise, *Q* is FALSE.	BOOL	TRUE, FALSE
ET	The time elapsed since the timer started. It will remain unchanged after reaching *PT*.	TIME	As each data type suggested.

- **Function**

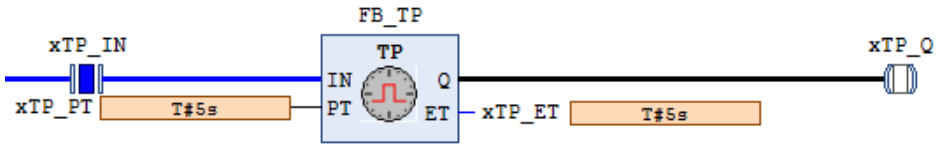
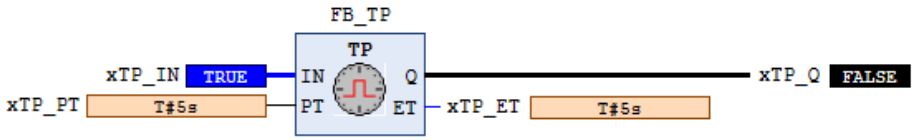
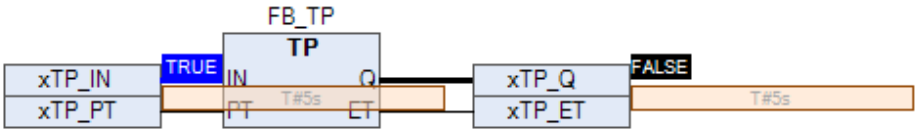
  This function implements the pulse timer.

- **Programming Example**

  In the below example, the variable *xTP_IN* is the Boolean input variable which starts the pulse timer for the rising edge. The variable *xTP_PT*=5s is the time specified for the length of the pulse. The output variable *xTP_Q* is initially TRUE and is set to FALSE only when the specified pulse time of variable *xTP_PT*=5s is elapsed. The elapsed time is shown in the variable *xTP_ET*.

  Feature in the ST/LD/FBD & CFC editor:

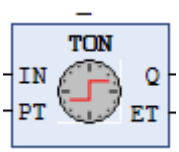
Programming Language	Program
ST	FB_TP(IN TRUE := xTP_IN TRUE, PT T#5s := xTP_PT T#5s, Q FALSE => xTP_Q FALSE, ET T#5s => xTP_ET T#5s);

Programming Language	Program
LD	
FBD	
CFC	

- **Supported Products**
  - AX series
- **Library**
  - Standard.lib

18.22.2. TON Function (Turn on Delay Timer)

TON implements a timer with a turn–on delay.

FB/FC	Instruction	Graphic Expression	ST Language
FB	TON		FB_TON(IN := , PT := , Q => , ET =>);

- **Inputs**

Name	Function	Data Type	Setting Value
IN	If rising edge, starts increasing the internal timer until *ET*=*PT*.	BOOL	TRUE, FALSE
PT	Time for the delay counter (ms)	TIME	As each data type suggested.

- **Outputs**

Name	Function	Data Type	Setting Value
Q	When *IN* is TRUE and *ET* is equal to *PT*, *Q* is TRUE; Otherwise, *Q* is FALSE.	BOOL	TRUE, FALSE
ET	The time elapsed since the timer started. It will remain unchanged after reaching *PT*.	TIME	As each data type suggested.

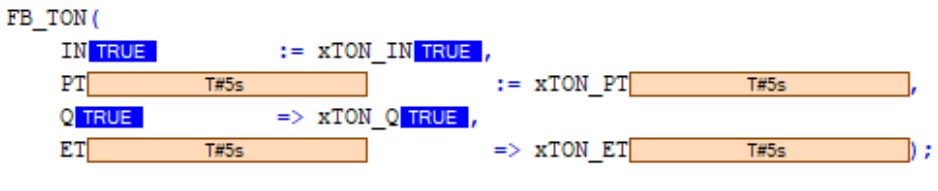
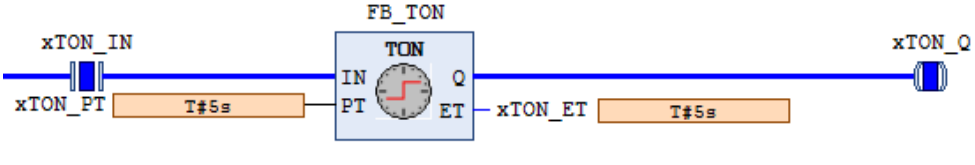
- **Function**

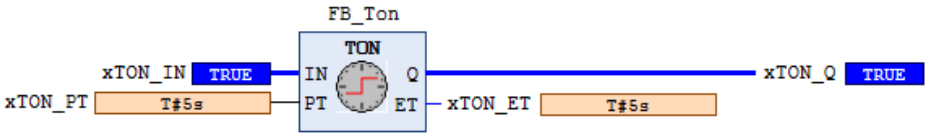
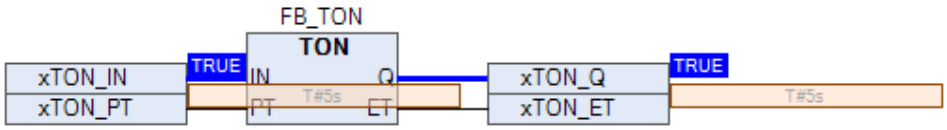
  This function implements a timer with a turn–on delay. It increases an internal timer up to a given value.

- **Programming Example**

  In the below example, the variable *xTON_IN* is the Boolean input variable which starts the delay counter for the rising edge and resets the delay counter for the falling edge. The variable *xTON_PT*=5s is the time specified for the delay counter. The output variable *xTON_Q* is initially FALSE and is set to TRUE only when the delay time of variable *xTON_PT*=5s is elapsed. The elapsed time is shown in the variable *xTON_ET*.

  Feature in the ST/LD/FBD & CFC editor:

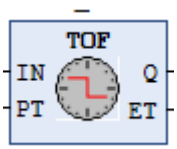
Programming Language	Program
ST	
LD	

Programming Language	Program
FBD	
CFC	

- **Supported Products**
  - AX series
- **Library**
  - Standard.lib

18.22.3. TOF Function (Turn off Delay Timer)

TOF implements a timer with a turn–off delay.

FB/FC	Instruction	Graphic Expression	ST Language
FB	TOF		FB_TOF(IN := , PT := , Q => , ET =>);

- **Inputs**

Name	Function	Data Type	Setting Value
IN	If falling edge, starts increasing the internal timer until *ET=PT*.	BOOL	TRUE, FALSE
PT	Time for the delay counter (ms)	TIME	As each data type suggested.

- **Outputs**

Name	Function	Data Type	Setting Value
Q	When *IN* is FALSE and *ET* is equal to *PT*, *Q* is FALSE; Otherwise, *Q* is TRUE.	BOOL	TRUE, FALSE
ET	The time elapsed since the timer started. It will remain unchanged after reaching *PT*.	TIME	As each data type suggested.

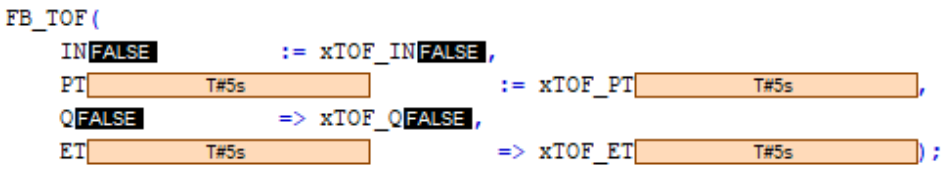
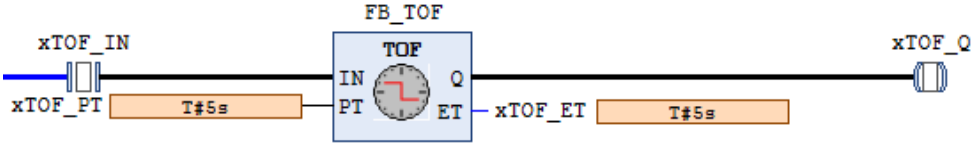
- **Function**

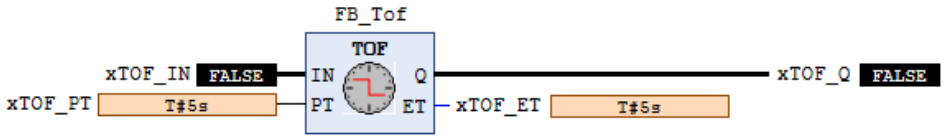
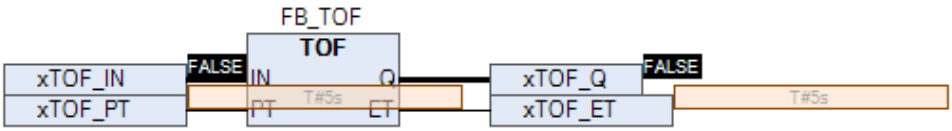
  This function implements a timer with a turn–off delay. It increases an internal timer up to a given value.

- **Programming Example**

  In the below example, variable *xTOF_IN* is used to denote the rising edge of the pulse. Its initial value is TRUE. The variable *xTOF_PT*=5s is used to denote the length of the pulse. But when the input value *xTOF_IN* is set to FALSE, and the variable *xTOF_ET* reaches 5s (equals to *PT*), the output variable *xTOF_Q* becomes FALSE.

  Feature in the ST/LD/FBD & CFC editor:

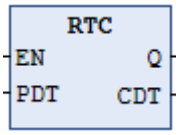
Programming Language	Program
ST	
LD	

Programming Language	Program
FBD	
CFC	

- **Supported Products**
  - AX series
- **Library**
  - Standard.lib

18.22.4. RTC Function (Real Time Clock)

RTC calculates the elapsed time since a given start time.

FB/FC	Instruction	Graphic Expression	ST Language
FB	RTC		FB_RTC(EN := , PDT := , Q => , CDT =>);

- **Inputs**

Name	Function	Data Type	Setting Value
EN	Turn on the function block. TRUE: Rising edge: *CDT* is set to *PDT* and *CDT* starts increasing.	BOOL	TRUE, FALSE
PDT	Preset date and time.	DT	DT#1970–01–01–0:0:0 to DT#2106–02–07–06:28:15

- **Outputs**

Name	Function	Data Type	Setting Value
Q	TRUE as long as *CDT* is counting	BOOL	TRUE, FALSE
CDT	Date and time elapsed since *PDT* When *EN* is FALSE, *CDT* is 1970–01–01 00:00:00. When *EN* is TRUE, *CDT* starts counting from *PDT* in seconds. *PDT* is set on the rising edge of *EN*.	DT	DT#1970–01–01–0:0:0 to DT#2106–02–07–06:28:15

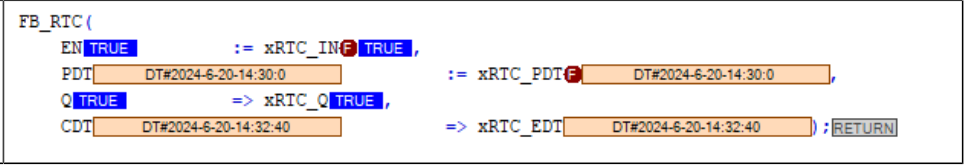
- **Function**

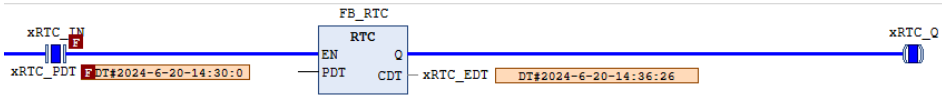
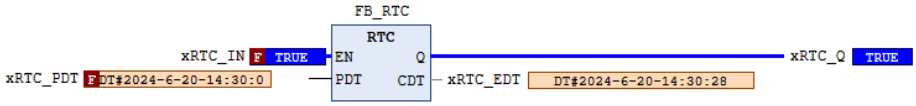
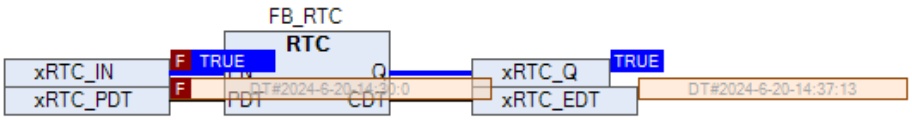
  This function block calculates the elapsed time since a given start time. When *EN* is TRUE (rising edge), the time of *PDT* is set and counted in seconds and output to *CDT*. Once *EN* is reset to FALSE, *CDT* will be reset to the initial value DT#1970–01–01–00:00:00.

- **Programming Example**

  In the below example, variable *xRTC_PDT* is the present Date and Time. *xRTC_EDT* is the output variable to specify the date and time elapsed since.

  Feature in the ST/LD/FBD & CFC editor:

Programming Language	Program
ST	

Programming Language	Program
LD	
FBD	
CFC	

- **Supported Products**
  - AX series
- **Library**
  - Standard.lib

Smarter. Greener. Together.

Industrial Automation Headquarters

Delta Electronics, Inc.
Taoyuan Technology Center
No.18, Xinglong Rd., Taoyuan District,
Taoyuan City 330477, Taiwan
TEL: +886-3-362-6301 / FAX: +886-3-371-6301

Asia

Delta Electronics (Shanghai) Co., Ltd.
No.182 Minyu Rd., Pudong Shanghai, P.R.C.
Post code : 201209
TEL: +86-21-6872-3988 / FAX: +86-21-6872-3996
Customer Service: 400-820-9595

Delta Electronics (Japan), Inc.
Industrial Automation Sales Department
2-1-14 Shibadaimon, Minato-ku
Tokyo, Japan 105-0012
TEL: +81-3-5733-1155 / FAX: +81-3-5733-1255

Delta Electronics (Korea), Inc.
1511, 219, Gasan Digital 1-Ro., Geumcheon-gu,
Seoul, 08501 South Korea
TEL: +82-2-515-5305 / FAX: +82-2-515-5302

Delta Energy Systems (Singapore) Pte Ltd.
4 Kaki Bukit Avenue 1, #05-04, Singapore 417939
TEL: +65-6747-5155 / FAX: +65-6744-9228

Delta Electronics (India) Pvt. Ltd.
Plot No.43, Sector 35, HSIIDC Gurgaon,
PIN 122001, Haryana, India
TEL: +91-124-4874900 / FAX: +91-124-4874945

Delta Electronics (Thailand) PCL.
909 Soi 9, Moo 4, Bangpoo Industrial Estate (E.P.Z),
Pattana 1 Rd., T.Phraksa, A.Muang,
Samutprakarn 10280, Thailand
TEL: +66-2709-2800 / FAX: +66-2709-2827

Delta Electronics (Australia) Pty Ltd.
Unit 2, Building A, 18-24 Ricketts Road,
Mount Waverley, Victoria 3149 Australia
Mail: IA.au@deltaww.com
TEL: +61-1300-335-823 / +61-3-9543-3720

Americas

Delta Electronics (Americas) Ltd.
5101 Davis Drive, Research Triangle Park, NC 27709, U.S.A.
TEL: +1-919-767-3813 / FAX: +1-919-767-3969

Delta Electronics Brazil Ltd.
Estrada Velha Rio-São Paulo, 5300 Eugênio de
Melo - São José dos Campos CEP: 12247-004 - SP - Brazil
TEL: +55-12-3932-2300 / FAX: +55-12-3932-237

Delta Electronics International Mexico S.A. de C.V.
Gustavo Baz No. 309 Edificio E PB 103
Colonia La Loma, CP 54060
Tlalnepantla, Estado de México
TEL: +52-55-3603-9200

EMEA

Delta Electronics (Netherlands) B.V.
Sales: Sales.IA.EMEA@deltaww.com
Marketing: Marketing.IA.EMEA@deltaww.com
Technical Support: iatechnicalsupport@deltaww.com
Customer Support: Customer-Support@deltaww.com
Service: Service.IA.emea@deltaww.com
TEL: +31(0)40 800 3900

Delta Electronics (Netherlands) B.V.
Automotive Campus 260, 5708 JZ Helmond, The Netherlands
Mail: Sales.IA.Benelux@deltaww.com
TEL: +31(0)40 800 3900

Delta Electronics (Netherlands) B.V.
Coesterweg 45, D-59494 Soest, Germany
Mail: Sales.IA.DACH@deltaww.com
TEL: +49 2921 987 238

Delta Electronics (France) S.A.
ZI du bois Challand 2, 15 rue des Pyrénées,
Lisses, 91090 Evry Cedex, France
Mail: Sales.IA.FR@deltaww.com
TEL: +33(0)1 69 77 82 60

Delta Electronics Solutions (Spain) S.L.U
Ctra. De Villaverde a Vallecas, 265 1º Dcha Ed.
Hormigueras – P.I. de Vallecas 28031 Madrid
TEL: +34(0)91 223 74 20

Carrer Llacuna 166, 08018 Barcelona, Spain
Mail: Sales.IA.Iberia@deltaww.com

Delta Electronics (Italy) S.r.l.
Via Meda 2–22060 Novedrate(CO)
Piazza Grazioli 18 00186 Roma Italy
Mail: Sales.IA.Italy@deltaww.com
TEL: +39 039 8900365

Delta Energy System LLC
Vereyskaya Plaza II, office 112 Vereyskaya str.
17 121357 Moscow Russia
Mail: Sales.IA.RU@deltaww.com
TEL: +7 495 644 3240

Delta Greentech Elektronik San. Ltd. Sti. (Turkey)
Şerifali Mah. Hendem Cad. Kule Sok. No:16-A
34775 Ümraniye – İstanbul
Mail: Sales.IA.Turkey@deltaww.com
TEL: + 90 216 499 9910

Eltek Dubai (Eltek MEA DMCC)
OFFICE 2504, 25th Floor, Saba Tower 1,
Jumeirah Lakes Towers, Dubai, UAE
Mail: Sales.IA.MEA@deltaww.com
TEL: +971(0)4 2690148

DIAS-Manual-0025-EN
2023/05/11