Unity Pro
Drive control
Block Library

04/2015

33002547.13

www.schneider-electric.com

Schneider Electric

The information provided in this documentation contains general descriptions and/or technical characteristics of the performance of the products contained herein. This documentation is not intended as a substitute for and is not to be used for determining suitability or reliability of these products for specific user applications. It is the duty of any such user or integrator to perform the appropriate and complete risk analysis, evaluation and testing of the products with respect to the relevant specific application or use thereof. Neither Schneider Electric nor any of its affiliates or subsidiaries shall be responsible or liable for misuse of the information contained herein. If you have any suggestions for improvements or amendments or have found errors in this publication, please notify us.

All pertinent state, regional, and local safety regulations must be observed when installing and using this product. For reasons of safety and to help ensure compliance with documented system data, only the manufacturer should perform repairs to components.

When devices are used for applications with technical safety requirements, the relevant instructions must be followed.

Failure to use Schneider Electric software or approved software with our hardware products may result in injury, harm, or improper operating results.

Failure to observe this information can result in injury or equipment damage.

Table of Contents

Safety Information

Important Information

NOTICE

Read these instructions carefully, and look at the equipment to become familiar with the device before trying to install, operate, or maintain it. The following special messages may appear throughout this documentation or on the equipment to warn of potential hazards or to call attention to information that clarifies or simplifies a procedure.

The addition of this symbol to a “Danger” or “Warning” safety label indicates that an electrical hazard exists which will result in personal injury if the instructions are not followed.

This is the safety alert symbol. It is used to alert you to potential personal injury hazards. Obey all safety messages that follow this symbol to avoid possible injury or death.

⚠ DANGER

DANGER indicates a hazardous situation which, if not avoided, **will result in** death or serious injury.

⚠ WARNING

WARNING indicates a hazardous situation which, if not avoided, **could result in** death or serious injury.

⚠ CAUTION

CAUTION indicates a hazardous situation which, if not avoided, **could result** in minor or moderate injury.

NOTICE

NOTICE is used to address practices not related to physical injury.

PLEASE NOTE

Electrical equipment should be installed, operated, serviced, and maintained only by qualified personnel. No responsibility is assumed by Schneider Electric for any consequences arising out of the use of this material.

A qualified person is one who has skills and knowledge related to the construction and operation of electrical equipment and its installation, and has received safety training to recognize and avoid the hazards involved.

About the Book

At a Glance

Document Scope

This documentation describes the functions and functions blocks in the Motion library.

Validity Note

This document is valid for Unity Pro 10.0 or later.

Related Documents

Title of Documentation	Reference Number
Unity Pro System Bits and Words, Reference Manual	EIO0000002135 (English), EIO0000002136 (French), EIO0000002317 (German), EIO0000002138 (Italian), EIO0000002139 (Spanish), EIO0000002140 (Chinese)

You can download these technical publications and other technical information from our website at www.schneider-electric.com.

Part I
General

At a glance

This section contains general information about the Motion library.

What Is in This Part?

This part contains the following chapters:

Chapter 1
Block Types and their Applications

Overview

This chapter describes the different block types and their applications.

What Is in This Chapter?

This chapter contains the following topics:

Block Types

Block Types

Block Types

Different block types are used in Unity Pro. The general term for the block types is FFB.

There are the following types of block:

- Elementary Function (EF)
- Elementary Function Block (EFB)
- Derived Function Block (DFB)
- Procedure

NOTE: Motion Function Blocks are not available on the Quantum platform.

Elementary Function

Elementary functions (EF) have no internal status and one output only. If the input values are the same, the output value is the same for the executions of the function, for example the addition of two values gives the same result at every execution.

An elementary function is represented in the graphical languages (FBD and LD) as a block frame with inputs and an output. The inputs are represented on the left and the outputs on the right of the block frame. The name of the function, that is the function type, is shown in the center of the block frame.

The number of inputs can be increased with some elementary functions.

⚠ CAUTION
UNEXPECTED BEHAVIOR OF EQUIPMENT For Unity Pro V4.0 and earlier versions, do not use links to connect function blocks outputs, when your application relies on persistent output data of an EF. **Failure to follow these instructions can result in injury or equipment damage.**

NOTE: With Unity Pro V4.0 and earlier versions the deactivation of an EF (EN=0) causes links connected to its Input/Output to be reset. To transfer the state of the signal do not use a link. A variable must be connected to the EF's output and must be used to connect the input of the element. With Unity Pro V4.1 and later versions you can maintain the output links even if an EF is deactivated by activating the option **Maintain output links on disabled EF (EN=0)** via the menu **Tools →Program →Languages →Common**.

Elementary Function Block

Elementary function blocks (EFB) have an internal status. If the inputs have the same values, the value on the outputs can have another value during the individual executions. For example, with a counter, the value on the output is incremented.

An elementary function block is represented in the graphical languages (FBD and LD) as a block frame with inputs and outputs. The inputs are represented on the left and the outputs on the right of the block frame. The name of the function block, that is the function block type, is shown in the center of the block frame. The instance name is displayed above the block frame.

Derived Function Block

Derived function blocks (DFBs) have the same properties as elementary function blocks. They are created by the user in the programming languages FBD, LD, IL and/or ST.

Procedure

Procedures are functions with several outputs. They have no internal state.

The only difference from elementary functions is that procedures can have more than one output and they support variables of the `VAR_IN_OUT` data type.

Procedures do not return a value.

Procedures are a supplement to IEC 61131-3 and must be enabled explicitly.

There is no visual difference between procedures and elementary functions.

⚠ CAUTION

UNEXPECTED BEHAVIOR OF EQUIPMENT

For Unity Pro V4.0 and earlier versions, do not use links to connect function blocks outputs, when your application relies on persistent output data of an EF.

Failure to follow these instructions can result in injury or equipment damage.

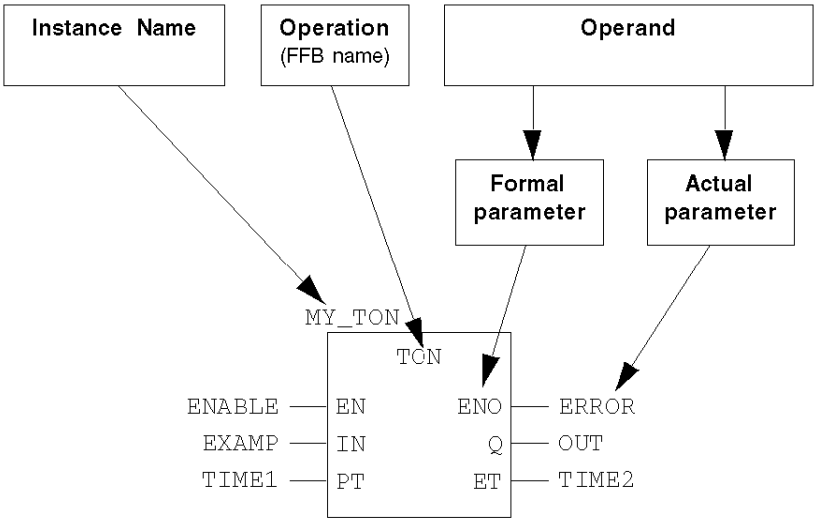
NOTE: With Unity Pro V4.0 and earlier versions the deactivation of an EF (EN=0) causes links connected to its Input/Output to be reset. To transfer the state of the signal do not use a link. A variable must be connected to the EF's output and must be used to connect the input of the element. With Unity Pro V4.1 and later versions you can maintain the output links even if an EF is deactivated by activating the option **Maintain output links on disabled EF (EN=0)** via the menu **Tools →Program →Languages →Common**.

FFB Structure

Structure

Each FFB is made up of an operation (name of the FFB), the operands are required for the operation (formal and actual parameters) and an instance name for elementary/derived function blocks.

Call of a function block in the FBD programming language:



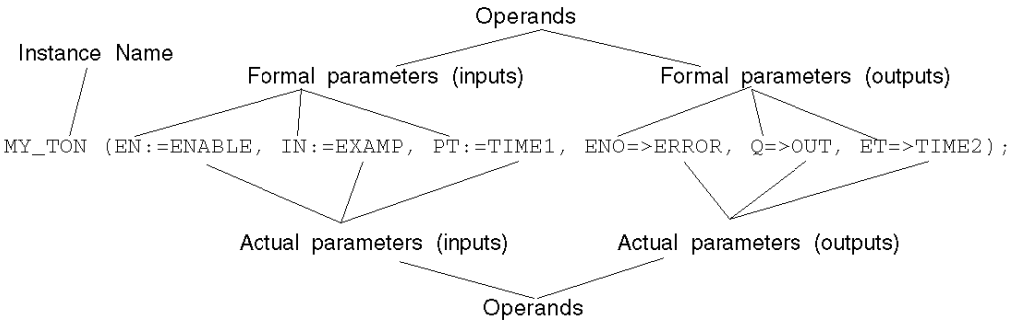
⚠ CAUTION

UNEXPECTED APPLICATION BEHAVIOR

Do not call several times the same block instance within a PLC cycle

Failure to follow these instructions can result in injury or equipment damage.

Formal call of a function block in the ST programming language:



Operation

The operation determines which function is to be executed with the FFB, e.g. shift register, conversion operations.

Operand

The operand specifies what the operation is to be executed with. With FFBs, this consists of formal and actual parameters.

Formal/actual parameters

Inputs and outputs are required to transfer values to or from an FFB. These are called formal parameters.

Objects are linked to formal parameters; these objects contain the current process states. They are called actual parameters.

At program runtime, the values from the process are transferred to the FFB via the actual parameters and then output again after processing.

The data type of the actual parameters must match the data type of the input/output (formal parameters). The only exceptions are generic inputs/outputs whose data type is determined by the actual parameter. If the actual parameters consist of literals, a suitable data type is selected for the function block.

FFB Call in IL/ST

In text languages IL and ST, FFBs can be called in formal and in informal form. Details can be found in the *Reference manual*.

Example of a formal function call:

```
out:=LIMIT (MN:=0, IN:=var1, MX:=5);
```

Example of an informal function call:

```
out:=LIMIT (0, var1, 5);
```

NOTE: The use of EN and ENO is only possible for formal calls.

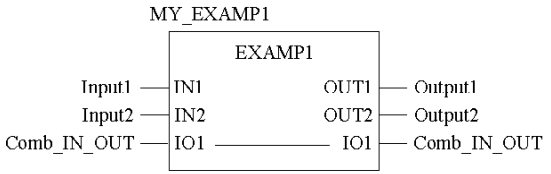
VAR_IN_OUT variable

FFBs are often used to read a variable at an input (input variables), to process it and to output the altered values of the **same** variable (output variables).

This special type of input/output variable is also called a VAR_IN_OUT variable.

The input and output variable are linked in the graphic languages (FBD and LD) using a line showing that they belong together.

Function block with VAR_IN_OUT variable in FBD:



Function block with VAR_IN_OUT variable in ST:

```
MY_EXAMP1 (IN1:=Input1, IN2:=Input2, IO1:=Comb_IN_OUT,
           OUT1=>Output1, OUT2=>Output2);
```

The following points must be considered when using FFBs with VAR_IN_OUT variables:

- The VAR_IN_OUT inputs must be assigned a variable.
- Literals or constants cannot be assigned to VAR_IN_OUT inputs/outputs.

The following additional limitations apply to the graphic languages (FBD and LD):

- When using graphic connections, VAR_IN_OUT outputs can only be connected with VAR_IN_OUT inputs.
- Only one graphical link can be connected to a VAR_IN_OUT input/output.
- Different variables/variable components can be connected to the VAR_IN_OUT input and the VAR_IN_OUT output. In this case the value of the variables/variable component on the input is copied to the output variables/variable component.
- No negations can be used on VAR_IN_OUT inputs/outputs.
- A combination of variable/address and graphic connections is not possible for VAR_IN_OUT outputs.

EN and ENO

Description

An EN input and an ENO output can be configured for the FFBs.

If the value of EN is equal to "0" when the FFB is invoked, the algorithms defined by the FFB are not executed and ENO is set to "0".

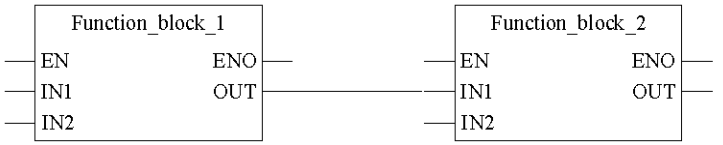
If the value of EN is equal to "1" when the FFB is invoked, the algorithms defined by the FFB will be executed. After the algorithms have been executed successfully, the value of ENO is set to "1". If certain error conditions are detected when executing these algorithms, ENO is set to "0".

If the EN pin is not assigned a value, when the FFB is invoked, the algorithm defined by the FFB is executed (same as if EN equals to "1"), Please refer to *Maintain output links on disabled EF (see Unity Pro, Operating Modes)*.

If the algorithms are executed successfully, then value of ENO is set to "1", else ENO is set to "0".

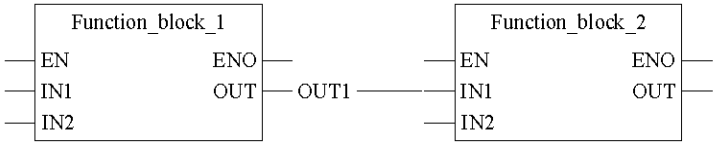
If ENO is set to "0" (caused by EN=0 or a detected error condition during execution or unsuccessful algorithm execution):

- Function blocks
  - EN/ENO handling with function blocks that (only) have one link as an output parameter:



If EN from FunctionBlock_1 is set to "0", the output connection OUT from FunctionBlock_1 retains the status it had in the last correctly executed cycle.

  - EN/ENO handling with function blocks that have one variable and one link as output parameters:



If EN from FunctionBlock_1 is set to "0", the output connection OUT from FunctionBlock_1 retains the status it had in the last correctly executed cycle. The variable OUT1 on the same pin, either retains its previous status or can be changed externally without influencing the connection. The variable and the link are saved independently of each other.

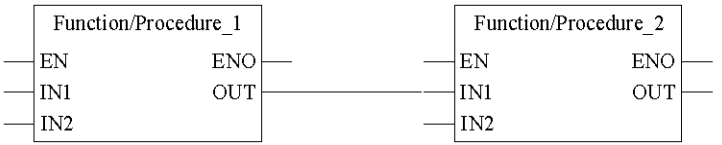
- Functions/Procedures

⚠ CAUTION
UNEXPECTED BEHAVIOR OF EQUIPMENT For Unity Pro V4.0 and earlier versions, do not use links to connect function blocks outputs, when your application relies on persistent output data of an EF. **Failure to follow these instructions can result in injury or equipment damage.**

NOTE: With Unity Pro V4.0 and earlier versions the deactivation of an EF (EN=0) causes links connected to its Input/Output to be reset. To transfer the state of the signal do not use a link. A variable must be connected to the EF's output and must be used to connect the input of the element. With Unity Pro V4.1 and later versions you can maintain the output links even if an EF is deactivated by activating the option **Maintain output links on disabled EF (EN=0)** via the menu **Tools →Program →Languages →Common**.

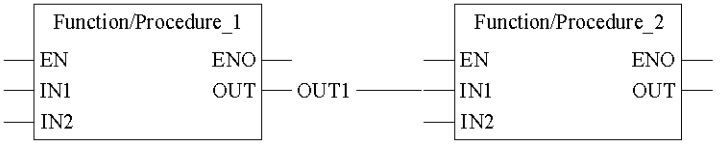
As defined in IEC61131-3, the outputs from deactivated functions (`EN`-input set to "0") is undefined. (The same applies to procedures.)
Here is an explanation of the output status in this case:

- EN/ENO handling with functions/procedures that (only) have one link as an output parameter:



  If `EN` from `Function/Procedure_1` is set to "0", the output connection `OUT` from `Function/Procedure_1` is also set to "0".

- EN/ENO handling with function blocks that have one variable and one link as output parameters:



  If `EN` from `Function/Procedure_1` is set to "0", the output connection `OUT` from `Function/Procedure_1` is also set to "0". The variable `OUT1` on the same pin, either retains its previous status or can be changed externally without influencing the connection. The variable and the link are saved independently of each other.

The output behavior of the FFBs does not depend on whether the FFBs are called up without `EN`/`ENO` or with `EN`=1.

Conditional/Unconditional FFB Call

"Unconditional" or "conditional" calls are possible with each FFB. The condition is realized by pre-linking the input EN.

- EN connected
  conditional calls (the FFB is only processed if EN = 1)
- EN shown, hidden, and marked TRUE, or shown and not occupied
  unconditional calls (FFB is processed independent from EN)

NOTE: For disabled function blocks (EN = 0) with an internal time function (e.g. DELAY), time seems to keep running, since it is calculated with the help of a system clock and is therefore independent of the program cycle and the release of the block.

⚠ CAUTION
UNEXPECTED APPLICATION EQUIPMENT
Do not disable function blocks with internal time function during their operation.
Failure to follow these instructions can result in injury or equipment damage.

Note for IL and ST

The use of EN and ENO is only possible in the text languages for a formal FFB call, e.g.

```
MY_BLOCK (EN:=enable, IN1:=var1, IN2:=var2,
ENO=>error, OUT1=>result1, OUT2=>result2);
```

Assigning the variables to ENO must be done with the operator =>.

With an informal call, EN and ENO cannot be used.

Chapter 2
Block Availability on the Various Hardware Platforms

Block Availability on the Various Hardware Platforms

Introduction

Not all blocks are available on all hardware platforms. The blocks available on your hardware platform can be found in the following tables.

NOTE: The functions and function blocks in this library are not defined in IEC 61131-3.

Axis Control

Availability of the blocks:

Block name	Block type	M340	M580	Quantum	Momentum	Premium
SET_CMD	Procedure	-	-	-	-	+
SMOVE	Procedure	-	-	-	-	+
XMOVE	Procedure	-	-	-	-	+
Legend:						
+	Yes					
-	No					

CAM control

Availability of the blocks:

Block name	Block type	M340	M580	Quantum	Momentum	Premium
DETAIL_OBJECT	Procedure	-	-	-	-	+
MOD_CAM	Procedure	-	-	-	-	+
MOD_PARAM	Procedure	-	-	-	-	+
MOD_TRACK	Procedure	-	-	-	-	+
Legend:						
+	Yes					
-	No					

MMF Start

Availability of the blocks:

Block name	Block type	M340	M580	Quantum	Momentum	Premium
CFG_CP_F	EFB	-	-	+	-	-
CFG_CP_V	EFB	-	-	+	-	-
CFG_CS	EFB	-	-	+	-	-
CFG_FS	EFB	-	-	+	-	-
CFG_IA	EFB	-	-	+	-	-
CFG_RA	EFB	-	-	+	-	-
CFG_SA	EFB	-	-	+	-	-
DRV_DNLD	EFB	-	-	+	-	-
DRV_UPLD	EFB	-	-	+	-	-
IDN_CHK	EFB	-	-	+	-	-
IDN_XFER	EFB	-	-	+	-	-
MMF_BITS	EFB	-	-	+	-	-
MMF_ESUB	EFB	-	-	+	-	-
MMF_INDX	EFB	-	-	+	-	-
MMF_JOG	EFB	-	-	+	-	-
MMF_MOVE	EFB	-	-	+	-	-
MMF_RST	EFB	-	-	+	-	-
MMF_SUB	EFB	-	-	+	-	-
MMF_USUB	EFB	-	-	+	-	-
Legend:						
+	Yes					
-	No					

Lexium

Availability of the blocks:

Block name	Block type	M340	M580	Quantum	Momentum	Premium
LXM_RESTORE	Procedure	+	-	-	-	+
LXM_SAVE	Procedure	+	-	-	-	+

Part II
Axis Control

At a glance

This section describes the elementary functions and elementary function blocks of the `Axis Control` family.

What Is in This Part?

This part contains the following chapters:

Chapter 3
SMOVE: Automatic movement control

Description

Function description

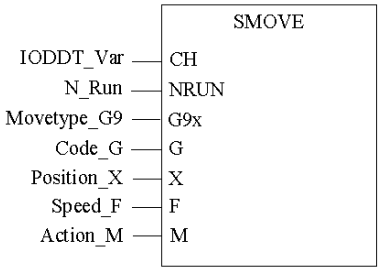
The SMOVE function enables axis control to be programmed in order to carry out a movement automatically.

For a more detailed description of how this function operates, refer to the application specific axis control manual *(see Premium and Atrium Using Unity Pro, Axis Control Modules for Servomotors, User Manual)*.

The additional parameters EN and ENO can be configured.

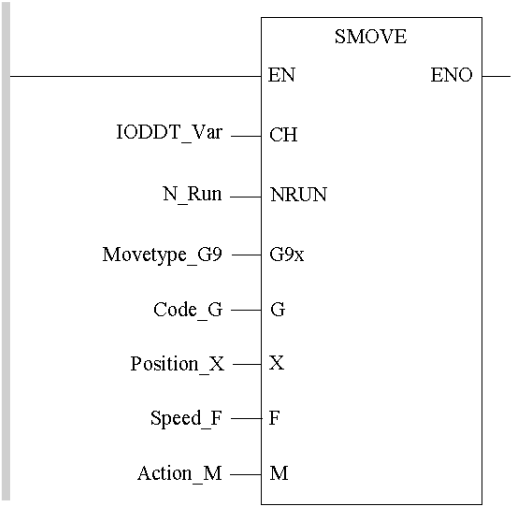
Representation in FBD

Representation:



Representation in LD

Representation:



Representation in IL

Representation:

```
LD IODDT_Var
SMOVE N_Run, Movetype_G9, Code_G, Position_X, Speed_F, Action_M
```

Representation in ST

Representation:

```
SMOVE(IODDT_Var, N_Run, Movetype_G9, Code_G, Position_X, Speed_F,
Action_M);
```

Description of the parameters

The following table describes the input parameters:

Parameter	Type	Comment
IODDT_Var	IODDT*	IODDT variable corresponding to the channel on which the function is to be carried out.* **Example**: IODDT_Var of type T_AXIS_STD
N_Run	INT	Number of movement carried out by the function.
Movetype_G9	INT	Type of movement carried out by the function.
Code_G	INT	Function instruction code.
Position_X	DINT	Coordinates of the position to be reached or towards which the part is to move. In the case of some instructions this may be the original reference point value.
Speed_F	DINT	Movement speed of moving part This depends on the chosen unit of measurement.
Action_M	INT	Manages the following actions: ● Activation or non-activation of the event processing trigger. ● Setting the discrete output associated with the channel to 0 or 1. ● Determines the type of event awaited by the instruction G05.
* IODDT represents the set of types associated with a module or a channel on which the function is to be carried out.		

Chapter 4
XMOVE: Interpolation control

Description

Function description

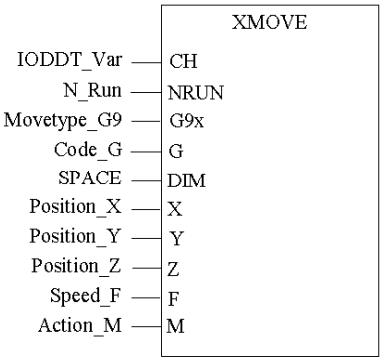
The XMOVE function enables interpolated movement control to be programmed.

For a more detailed description of how this function operates, refer to the application specific axis control manual *(see Premium and Atrium Using Unity Pro, Axis Control Modules for Servomotors, User Manual)*.

The additional parameters EN and ENO can be configured.

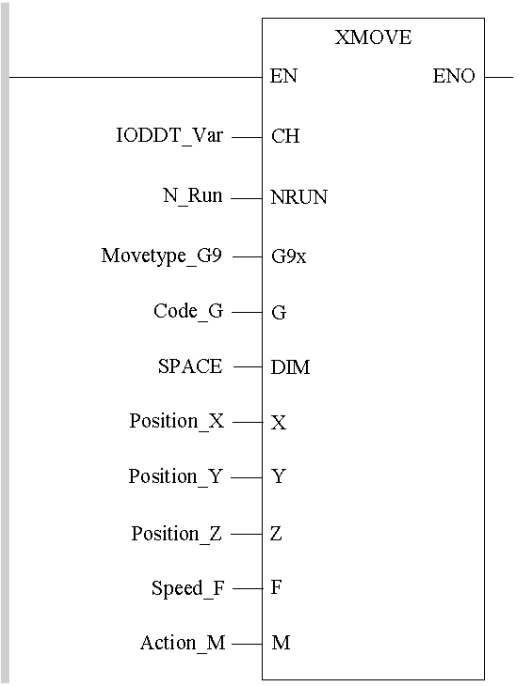
Representation in FBD

Representation:



Representation in LD

Representation:



Representation in IL

Representation:

```
LD IODDT_Var
XMOVE N_Run, Movetype_G9, Code_G, SPACE, Position_X, Position_Y,
Position_Z, Speed_F, Action_M
```

Representation in ST

Representation:

```
XMOVE(IODDT_Var, N_Run, Movetype_G9, Code_G, SPACE, Position_X,
Position_Y, Position_Z, Speed_F, Action_M);
```

Description of the parameters

The following table describes the input parameters:

Parameter	Type	Comment
IODDT_Var	IODDT*	IODDT variable corresponding to the channel on which the function is to be carried out. **Example**: IODDT_Var of type T_INTERPO_STD
N_Run	INT	Number of movement carried out by the function.
Movetype_G9	INT	Type of movement carried out by the function.
Code_G	INT	Function instruction code.
SPACE	INT	Number of the plane or space in which the function is carried out. This parameter specifies the list of axes concerned by the movement: ● 0 : Movement in the XY plane. ● 1 : Movement in the XZ plane. ● 2 : Movement in the YZ plane.
Position_X	DINT	Coordinate on the X axis of the position to be reached or towards which the part is to move.
Position_Y	DINT	Coordinate on the Y axis of the position to be reached or towards which the part is to move.
Position_Z	DINT	Coordinate on the Z axis of the position to be reached or towards which the part is to move.
Speed_F	DINT	Movement speed of moving part This depends on the chosen unit of measurement.
Action_M	INT	Manages the following actions: ● Activation or non-activation of the event processing trigger. ● Determines the type of event awaited by the instruction G05 or G10.
* IODDT represents the set of types associated with a module or a channel on which the function is to be carried out.		

Chapter 5
SET_CMD: Advanced prgramming in "C"

Description

Function description

This function bring the openes of Sercos TSX CSY 164 module for an advanced programming in "C" language. This is not a standard function. For more information, please consult our regional Sales offices.

Part III
CAM Control

Overview

This section describes the elementary functions and elementary function blocks of the `CAM control` family.

What Is in This Part?

This part contains the following chapters:

Chapter 6
DETAIL_OBJECT: Track cam descriptor information

Description

Function description

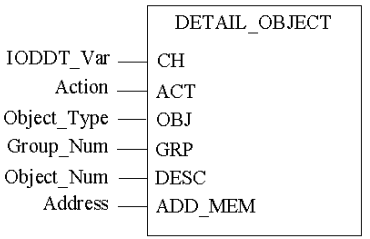
The DETAIL_OBJECT function facilitates the management and creation of a recipe via a Human Machine Interface. This function provides the application program with all information concerning the description of a track or of the cam in a %MW memory zone.

For a more detailed description of how this function operates, refer to the application specific electronic cam manual *(see Premium and Atrium Using Unity Pro, Electronic Cam Module, User Manual)*.

The additional parameters EN and ENO can be configured.

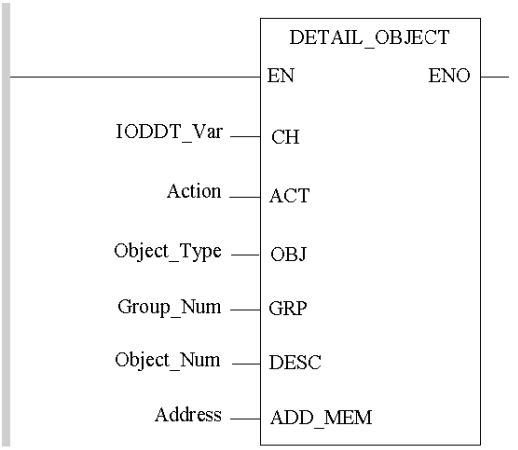
Representation in FBD

Representation:



Representation in LD

Representation:



Representation in IL

Representation:

```
LD IODDT_Var
DETAIL_OBJECT Action, Object_Type, Group_Num, Object_Num, Address
```

Representation in ST

Representation:

```
DETAIL_OBJECT(IODDT_Var, Action, Object_Type, Group_Num, Object_Num,
Address);
```

Description of the parameters

The following table describes the input parameters:

Parameter	Type	Comment
IODDT_Var	IODDT*	IODDT variable corresponding to the channel on which the function is to be carried out. **Example**: IODDT_Var of type T_CCY_GROUP0
Action	INT	Action to be carried out: ● Action = 1: Ext is used to write the cam or track descriptor to a memory zone. ● Action = 0: Inc is used to write the cam or track descriptor with data to the memory zone.
Object_Type	INT	Type of object: ● Object_Type = 0: cam. ● Object_Type = 1: track.
Group_Num	INT	Number of the group to which the cam or track belongs.
Object_Num	INT	Number of the cam or track in the group.
Address	INT	Index of first object of the memory zone of %MWs.
* IODDT represents the set of types associated with a module or a channel on which the function is to be carried out.		

Chapter 7
MOD_CAM: Adjustment of cam parameters

Description

Function description

The MOD_CAM function is used to adjust a cam dynamically. Transferring new data does not require the cam processor to switch to STOP mode.

If the transfer:

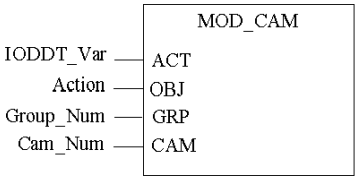
- is correctly carried out, the new parameters are taken into consideration,
- is not successfully carried out, then the cam processor remains in RUN mode with the old values.

For a more detailed description of how this function operates, refer to the application specific electronic cam manual *(see Premium and Atrium Using Unity Pro, Electronic Cam Module, User Manual)*.

The additional parameters EN and ENO can be configured.

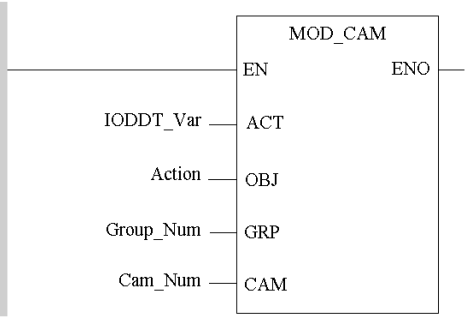
Representation in FBD

Representation:



Representation in LD

Representation:



Representation in IL

Representation:

```
LD IODDT_Var
MOD_CAM Action, Group_Num, Cam_Num
```

Representation in ST

Representation:

```
MOD_CAM(IODDT_Var, Action, Group_Num, Cam_Num);
```

Description of the parameters

The following table describes the input parameters:

Parameter	Type	Comment
`IODDT_Var`	`IODDT`*	IODDT variable corresponding to the channel on which the function is to be carried out. **Example**: `IODDT_Var` of type `T_CCY_GROUP0`
`Action`	`INT`	Type of action to be carried out: ● `Action` = 0: pre-loading of the exchange zone with the initial values. ● `Action` = 1: pre-loading of the exchange zone with the current adjustment values. ● `Action` = 2: sends new values to the module and updates the current parameter zone.
`Group_Num`	`INT`	Number of group to which the cam belongs.
`Cam_Num`	`INT`	Number of cam in the group.
* IODDT represents the set of types associated with a module or a channel on which the function is to be carried out.		

Chapter 8
MOD_PARAM: Adjustment of axis parameters

Description

Function description

The MOD_PARAM function is used to adjust a track dynamically. Transferring new data does not require the cam processor to switch to STOP mode.

If the transfer:

- is correctly carried out, the new parameters are taken into consideration,
- is not successfully carried out, then the cam processor remains in RUN mode with the old values.

The MOD_PARAM function assigns the following parameters to each exchange:

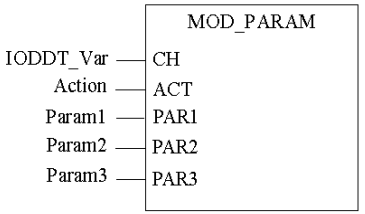
- PRESET_ANG_VALUE: value of angle recalibration.
- PRESET_TURN_VALUE: recalibration value of number of cycles.
- SLACK_VALUE: axis play value.
- MAX_PIECES: parts counter threshold.

For a more detailed description of how this function operates, refer to the application specific electronic cam manual *(see Premium and Atrium Using Unity Pro, Electronic Cam Module, User Manual)*.

The additional parameters EN and ENO can be configured.

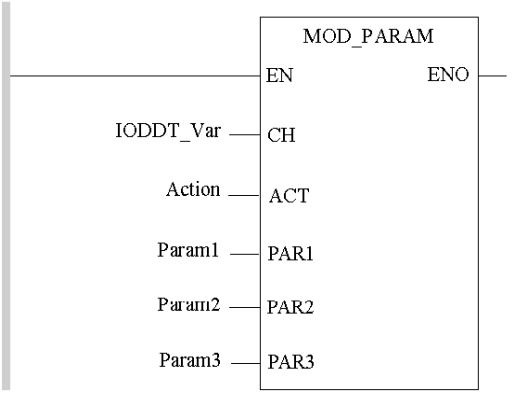
Representation in FBD

Representation:



Representation in LD

Representation:



Representation in IL

Representation:

```
LD IODDT_Var
MOD_PARAM Action, Param1, Param2, Param3
```

Representation in ST

Representation:

```
MOD_PARAM(IODDT_Var, Action, Param1, Param2, Param3);
```

Description of the parameters

The following table describes the input parameters:

Parameter	Type	Comment
`IODDT_Var`	`IODDT*`	IODDT variable corresponding to the channel on which the function is to be carried out. **Example**: `IODDT_Var` of type `T_CCY_GROUP0`
`Action`	`INT`	Type of action to be carried out: ● `Action` = 0: pre-loading of the exchange zone with the initial adjustment values. ● `Action` = 1: pre-loading of the exchange zone with the current adjustment values. ● `Action` = 2: sends new values to the module and updates the current parameter zone.

Parameter	Type	Comment
Param1	INT	Param1 must be equal to 0.
Param2	INT	Param2 must be equal to 0.
Param3	INT	Param3 must be equal to 0.
* IODDT represents the set of types associated with a module or a channel on which the function is to be carried out.		

Chapter 9
MOD_TRACK: Adjustment of track parameters

Description

Function description

The MOD_TRACK function is used to adjust a track dynamically. Transferring new data does not require the cam processor to switch to STOP mode.

If the transfer:

- is correctly carried out, the new parameters are taken into consideration,
- is not successfully carried out, then the cam processor remains in RUN mode with the old values.

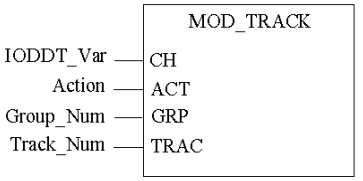
NOTE: the MOD_TRACK function affects only the forward value of a track

For a more detailed description of how this function operates, refer to the application specific electronic cam manual *(see Premium and Atrium Using Unity Pro, Electronic Cam Module, User Manual)*.

The additional parameters EN and ENO can be configured.

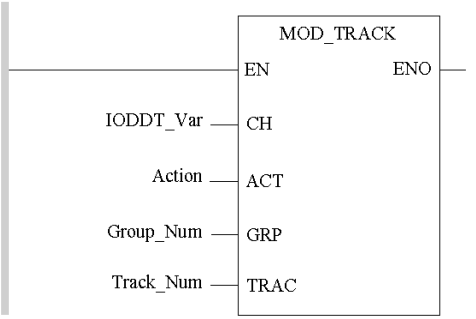
Representation in FBD

Representation:



Representation in LD

Representation:



Representation in IL

Representation:

```
LD IODDT_Var
MOD_TRACK Action, Group_Num, Track_Num
```

Representation in ST

Representation:

```
MOD_TRACK(IODDT_Var, Action, Group_Num, Track_Num);
```

Description of the parameters

The following table describes the input parameters:

Parameter	Type	Comment
`IODDT_Var`	`IODDT*`	IODDT variable corresponding to the channel on which the function is to be carried out. **Example**: `IODDT_Var` of type `T_CCY_GROUP0`
`Action`	`INT`	Type of action to be carried out: ● `Action` = 0: pre-loading of the exchange zone with the initial forward values. ● `Action` = 1: pre-loading of the exchange zone with the current forward values. ● `Action` = 2: updates the forward value of the track in the module and in the current parameters zone.
`Group_Num`	`INT`	Number of group to which the track belongs.
`Track_Num`	`INT`	Number of the track in the group.
* IODDT represents the set of types associated with a module or a channel on which the function is to be carried out.		

Part IV
MMF Start

At a glance

This section describes the elementary functions and elementary function blocks of the `MMF Start` family.

What Is in This Part?

This part contains the following chapters:

Chapter 10
CFG_CP_F: CamProfile

Description

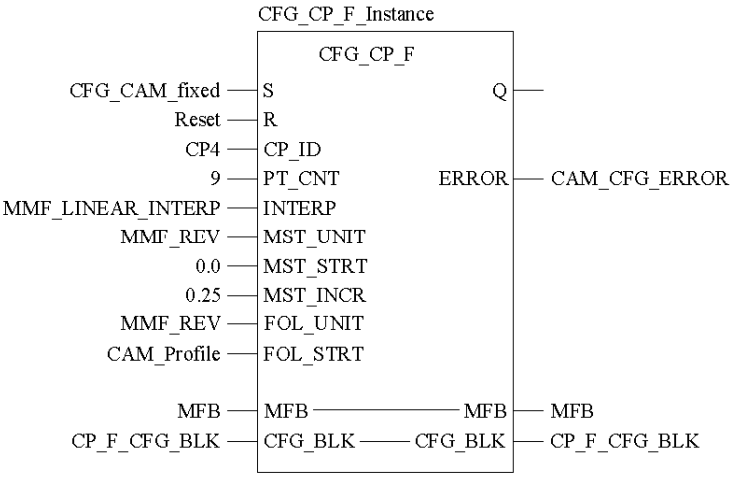
Function Description

This function block configures a CamProfile with fixed master increments.

Additional parameters EN and ENO may be configured.

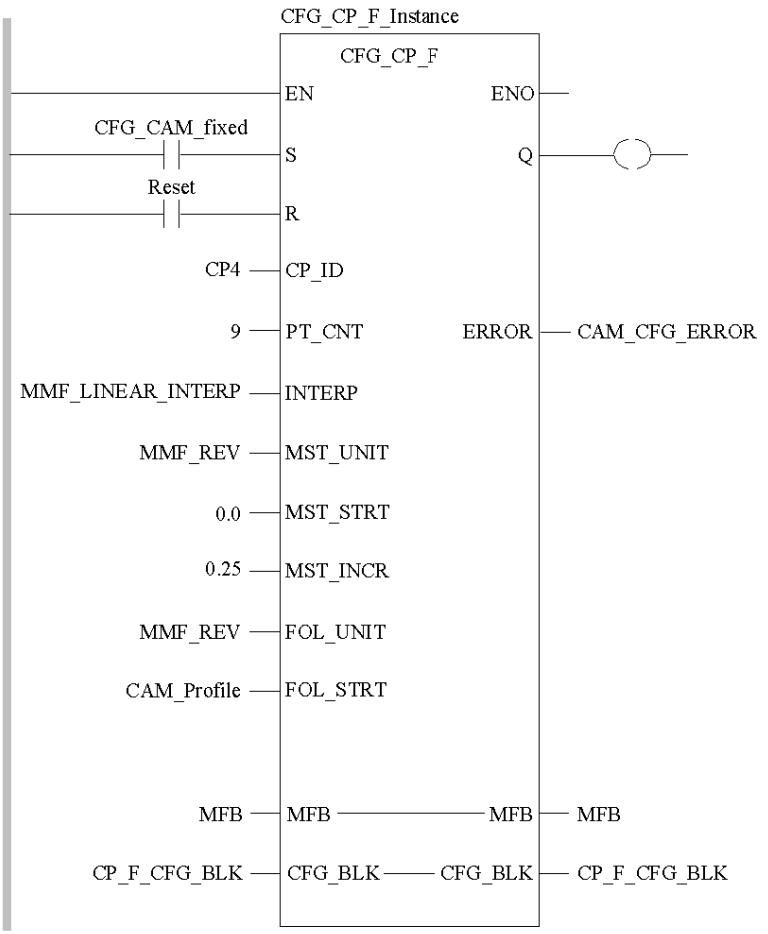
Representation in FBD

Representation:



Representation in LD

Representation:



Representation in IL

Representation:

```
CAL CFG_CP_F_Instance (S:=CFG_CAM_fixed, R:=Reset,
    CP_ID:=CP4, PT_CNT:=9, INTERP:=MMF_LINEAR_INTERP,
    MST_UNIT:=MMF_REV, MST_STRT:=0.0, MST_INCR:=0.25,
    FOL_UNIT:=MMF_REV, FOL_STRT:=CAM_Profile, MFB:=MFB,
    CFG_BLK:=CP_F_CFG_BLK, ERROR=>CAM_CFG_ERROR)
```

Representation in ST

Representation

```
CFG_CP_F_Instance (S:=CFG_CAM_fixed, R:=Reset,
    CP_ID:=CP4, PT_CNT:=9, INTERP:=MMF_LINEAR_INTERP,
    MST_UNIT:=MMF_REV, MST_STRT:=0.0, MST_INCR:=0.25,
    FOL_UNIT:=MMF_REV, FOL_STRT:=CAM_Profile, MFB:=MFB,
    CFG_BLK:=CP_F_CFG_BLK, ERROR=>CAM_CFG_ERROR) ;
```

Parameter Description

Description of input parameters:

Parameter	Data Type	Meaning
S	BOOL	Rising input starts configuration of the CamProfile.
R	BOOL	Resets block and prevents configuration. It halts configuration at the end of the current step.
CP_ID	INT	Axis ID for the CamProfile ID to be configured.
PT_CNT	INT	The number of points in the CamProfile.
INTERP	UDINT	Interpolation type: linear or cubic.
MST_UNIT	UDINT	Position units of the master.
MST_STRT	REAL	Master position data.
MST_INCR	REAL	Fixed master position increment.
FOL_UNIT	UDINT	Position units of the follower.
FOL_STRT	ARRAY [1 .. n] OF REAL	Array where the CamProfile configuration information will be stored.

Description of in-output parameters:

Parameter	Data Type	Meaning
MFB	MMFBlock *(see page 130)*	Must be connected to the MMFStart block of 200 registers.
CFG_BLK	CPCfg *(see page 141)*	Data block where the CamProfile configuration data will be stored.

Description of output parameters:

Parameter	Data Type	Meaning
Q	BOOL	True when the configuration has completed (reset by R).
ERROR	INT	First error code *(see page 143)* that is generated during configuration.

Example

When input CFG_CAM_fixed turns on, a fixed increment CamProfile identified as CamProfile 4 (CP4) is created in the motion controller. The velocity units for both the master and follower are set to revolutions on input pins MST_UNIT and FOL_UNIT. The interpolation type for the profile is LINEAR, as identified on the INTERP input pin. The master moves at a fixed increment starting at 0.0 and incrementing by .25 revolutions. These units are defined on input pins MST_STRT and MST_INCR. The follower path points are located in the array CAM_Profile; the point count for the follower path was identified as eight points at input PT_CNT. The follower path position points are determined and entered into the array CAM_Profile defined on input pin FOL_START. The follower moves from 0 to 1 in a positive direction, then returns to 0.

Follower Path Points

The follower path points entered into the PLC registers are listed in the following table:

Path Point	Follower Position Point
CAM_Profile[0]	0.0
CAM_Profile[1]	0.25
CAM_Profile[2]	0.5
CAM_Profile[3]	0.75
CAM_Profile[4]	1.0
CAM_Profile[5]	0.75
CAM_Profile[6]	0.5
CAM_Profile[7]	0.25
CAM_Profile[8]	0.0

Runtime Errors

If an error occurs during the configuration of this CamProfile, the error number can be examined at CAM_CFG_ERROR. This CamProfile configuration information is located in the register block FS_CFG_BLK after successful completion of the configuration.

NOTE: For a list of all error codes and values of the block, refer to *Tables of Error Codes for the Motion Library*, *page 164*.

Chapter 11
CFG_CP_V: CamProfile with Variable Master Increments

Description

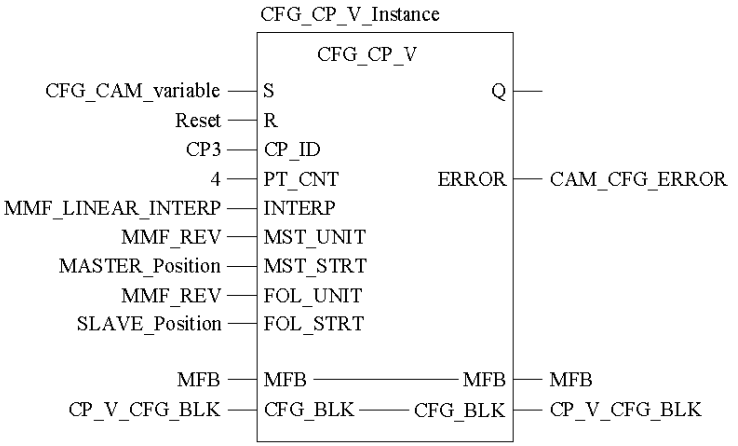
Function Description

This function block configures a CamProfile with variable master increments.

Additional parameters EN and ENO may be configured.

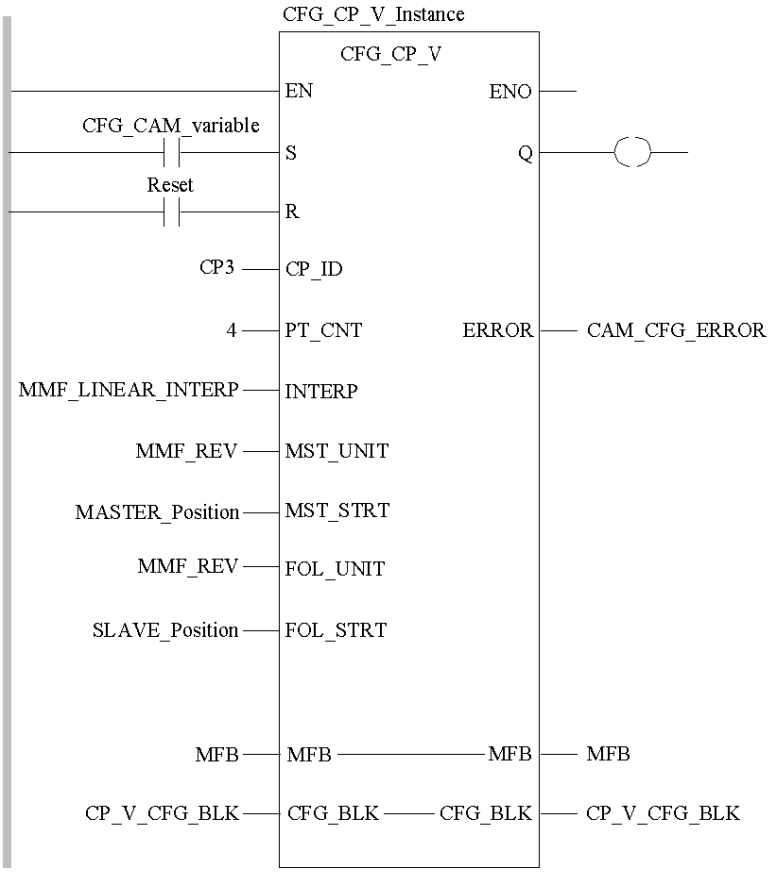
Representation in FBD

Representation:



Representation in LD

Representation:



Representation in IL

Representation:

```
CAL CFG_CP_V_Instance (S:=CFG_CAM_variable, R:=Reset,
    CP_ID:=CP3, PT_CNT:=4, INTERP:=MMF_LINEAR_INTERP,
    MST_UNIT:=MMF_REV, MST_STRT:=MASTER_Position,
    FOL_UNIT:=MMF_REV, FOL_STRT:=SLAVE_Position, MFB:=MFB,
    CFG_BLK:=CP_V_CFG_BLK, ERROR=>CAM_CFG_ERROR)
```

Representation in ST

Representation

```
CFG_CP_V_Instance (S:=CFG_CAM_variable, R:=Reset,
    CP_ID:=CP3, PT_CNT:=4, INTERP:=MMF_LINEAR_INTERP,
    MST_UNIT:=MMF_REV, MST_STRT:=MASTER_Position,
    FOL_UNIT:=MMF_REV, FOL_STRT:=SLAVE_Position, MFB:=MFB,
    CFG_BLK:=CP_V_CFG_BLK, ERROR=>CAM_CFG_ERROR) ;
```

Parameter Description

Description of input parameters:

Parameter	Data Type	Meaning
S	BOOL	Rising input starts configuration of the CamProfile.
R	BOOL	Resets block and prevents configuration. It halts configuration at the end of the current step.
CP_ID	INT	Axis ID for CamProfile to be configured.
PT_CNT	INT	The number of points in the CamProfile.
INTERP	UDINT	Interpolation type: linear or cubic.
MST_UNIT	UDINT	Position units of the master.
MST_STRT	ARRAY [1 .. n] OF REAL	Array of the master position data (n must be ≥ PT_CNT).
FOL_UNIT	UDINT	Position units of the slave.
FOL_STRT	ARRAY [1 .. n] OF REAL	Array of the slave position data (n must be ≥ PT_CNT).

Description of in-output parameters:

Parameter	Data Type	Meaning
MFB	MMFBlock *(see page 130)*	Must be connected to the MMFStart block of 200 registers.
CFG_BLK	CPCfg *(see page 141)*	Data block where the CamProfile configuration data will be stored.

Description of output parameters:

Parameter	Data Type	Meaning
Q	BOOL	True when the configuration has completed (reset by R).
ERROR	INT	First error code *(see page 143)* that is generated during configuration.

Example

When input CFG_CAM_variable turns on, a variable master increment CamProfile, identified as CamProfile 3 (CP3), is created in the motion controller, the velocity units for both the master and follower are set to revolutions on the input pins MST_UNIT and FOL_UNIT. The interpolation type is configured as LINEAR on the INTERP input pin. The master follows the variable point paths that are defined in register block MASTER_Position and specified on input pin MST_STRT. The follower path positions are defined in register block SLAVE_Position and specified on input pin FOL_STRT. Both the master and follower variable positions must be determined and entered into the specified PLC registers.

Master and Follower Positions

The master and follower positions that are entered into the PLC registers are listed in the following table:

Point Number	Master 409500	Follower 409600
1	0.0	0.0
2	0.2	0.5
3	0.6	0.5
4	0.8	0.0

During this profile execution, both the master and follower will start at position 0. When the master reaches +0.2 revolutions, the follower reaches +0.5 revolutions. While the master is moving from +0.2 through +0.6 revolutions, the follower remains at +0.5 revolutions. When the master reaches 0.8 revolutions, the follower moves back to 0 revolutions. While the master is moving from +0.8 revolutions to 0 revolutions, the follower remains at 0 revolutions.

Runtime Errors

If an error occurs during the configuration of this CamProfile, the error number can be examined at CAM_CFG_ERROR. This CamProfile configuration information is located in the register block CP_V_CFG_BLK after successful completion of the configuration.

NOTE: For a list of all error codes and values of the block, refer to *Tables of Error Codes for the Motion Library*, *page 164*.

Chapter 12
CFG_CS: CoordinatedSet

Description

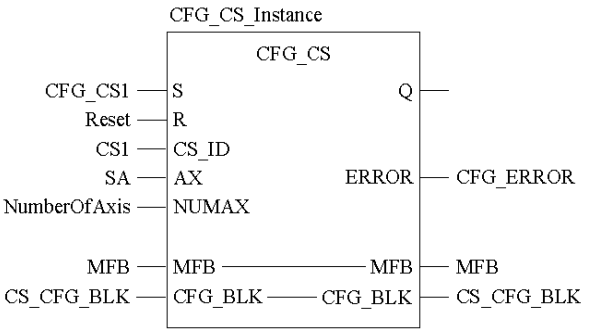
Function Description

This function block configures a CoordinatedSet.

Additional parameters EN and ENO may be configured.

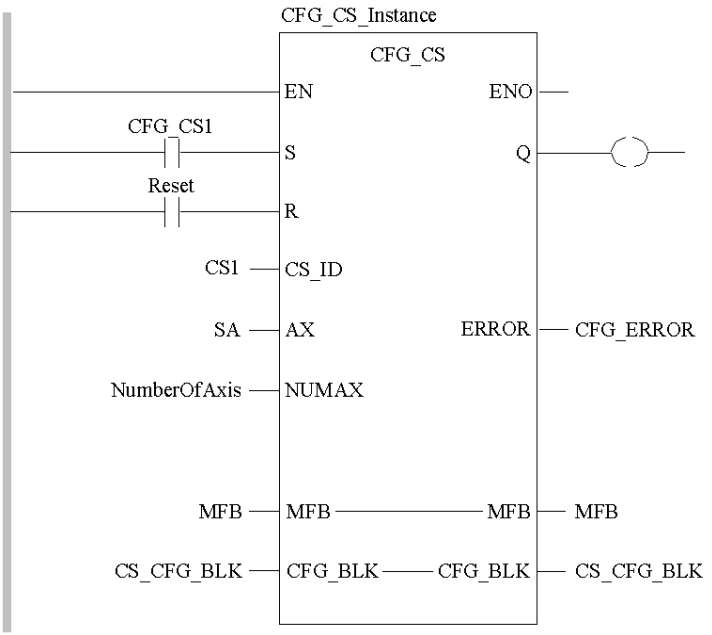
Representation in FBD

Representation:



Representation in LD

Representation:



Representation in IL

Representation:

```
CAL CFG_CS_Instance (S:=CFG_CS1, R:=Reset, CS_ID:=CS1,
    AX:=SA, NUMAX:=NumberOfAxis, MFB:=MFB,
    CFG_BLK:=CS_CFG_BLK, ERROR=>CFG_ERROR)
```

Representation in ST

Representation

```
CFG_CS_Instance (S:=CFG_CS1, R:=Reset, CS_ID:=CS1,
    AX:=SA, NUMAX:=NumberOfAxis, MFB:=MFB,
    CFG_BLK:=CS_CFG_BLK, ERROR=>CFG_ERROR) ;
```

Parameter Description

Description of input parameters:

Parameter	Data Type	Meaning
S	BOOL	Rising input starts configuration.
R	BOOL	Resets block and prevents configuration. It halts configuration at the end of the current step.
CS_ID	INT	Axis ID for the CoordinatedSet to be configured.
AX	ARRAY [1..8] of INT	Axis ID for single axis member to be included in reset.
NUMAX	UINT	Number of Axis. Range from 2 to 8.

Description of in-output parameters:

Parameter	Type	Meaning
MFB	MMFBlock *(see page 130)*	Must be connected to the MMFStart block of 200 registers.
CFG_BLK	CSCfg *(see page 139)*	Data block where the CoordinatedSet data will be stored.

Description of output parameters:

Parameter	Type	Meaning
Q	BOOL	True when the configuration has completed (reset by R).
ERROR	INT	First error code *(see page 143)* that is generated during configuration.

Example

When CFG_CS1 turns on, the configuration of CS1 occurs. During the creation of CS1, axes SA is defined on input pin AX , is assigned as member of this set. The number of pins ranges from 2 to 8 and the user must give the appropriate value on the NUMAX pin. If the user gives a value out of bounds, the input is forced to either the max. or min. value according to which boundary is exceeded. The CoordinatedSet information, including the member axes, is moved into the PLC register block CS_CFG_BLK that is assigned at output pin CFG_BLK. For detailed information on the size and layout of this register data block, refer to the CSCfg Data Structure *(see page 139)*.

Runtime Errors

For a list of all error codes and values of the block, refer to *Tables of Error Codes for the Motion Library, page 164*.

Chapter 13
CFG_FS: FollowerSet

Description

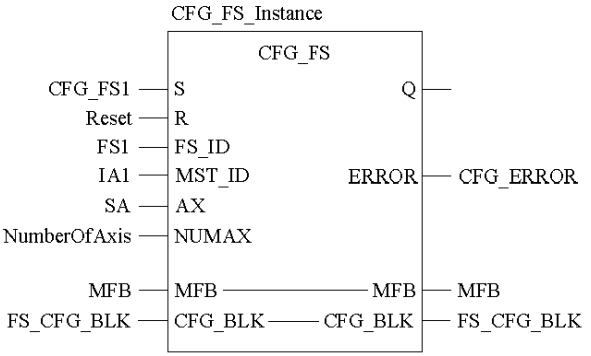
Function Description

This function block configures a FollowerSet.

Additional parameters EN and ENO may be configured.

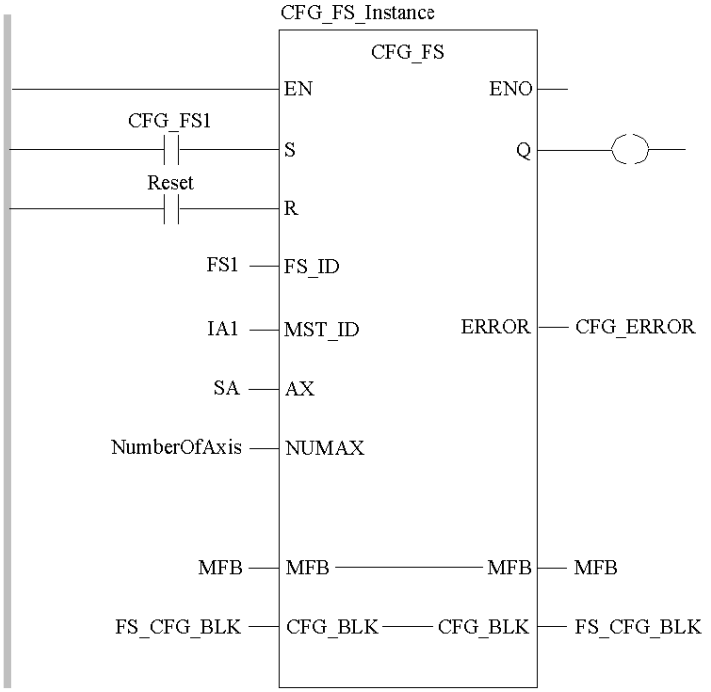
Representation in FBD

Representation:



Representation in LD

Representation:



Representation in IL

Representation:

```
CAL CFG_FS_Instance (S:=CFG_FS1, R:=Reset, FS_ID:=FS1,
    MST_ID:=IA1, AX:=SA, NUMAX:=NumberOfAxis, MFB:=MFB,
    CFG_BLK:=FS_CFG_BLK, ERROR=>CFG_ERROR)
```

Representation in ST

Representation

```
CFG_FS_Instance (S:=CFG_FS1, R:=Reset, FS_ID:=FS1,
    MST_ID:=IA1, AX:=SA, NUMAX:=NumberOfAxis, MFB:=MFB,
    CFG_BLK:=FS_CFG_BLK, ERROR=>CFG_ERROR) ;
```

Parameter Description

Description of input parameters:

Parameter	Data Type	Meaning
S	BOOL	Rising input starts configuration.
R	BOOL	Resets block and prevents configuration. It halts configuration at the end of the current step.
FS_ID	INT	Axis ID for the FollowerSet to be configured.
MST_ID	INT	Axis ID for the master axis.
AX	ARRAY [1..8] OF INT	Axis ID for single axis member to be included in the set.
NUMAX	UINT	Number of Axis. Range from 2 to 8.

Description of in-output parameters:

Parameter	Data Type	Meaning
MFB	MMFBlock *(see page 130)*	Must be connected to the MMFStart block of 200 registers.
CFG_BLK	FSCfg *(see page 140)*	Data block where the FollowerSet data will be stored.

Description of output parameters:

Parameter	Data Type	Meaning
Q	BOOL	True when the configuration has completed (reset by R).
ERROR	INT	First error code *(see page 143)* that is generated during configuration.

Example

When CFG_FS1 turns on, the configuration of FS1 occurs. During the creation of FS1, IA1 is assigned as the master at input pin MST_ID, and axes SA is assigned at input pin AX as member of this set. The number of pins ranges from 2 to 8 and the user must give the appropriate value on the NUMAX pin. If the user gives a value out of bounds, the input is forced to either the max. or min. value according to which boundary is exceeded. The FollowerSet information, including the member axes, are moved into the PLC register block FS_CFG_BLK that is assigned at output pin CFG_BLK. For detailed information on the size and layout of this register data block, refer to the FSCfg Data Structure *(see page 140)*. The default follower configuration for the member axes is MMF_CONTROLLER_GEAR_FEEDBACK with a gear ratio of 1:1. If the feedback configuration requirements are not the default, use the MMF_SUB block to modify the configuration for that member axis.

Runtime Errors

For a list of all error codes and values of the block, refer to *Tables of Error Codes for the Motion Library, page 164*.

Chapter 14
CFG_IA: ImaginaryAxis

Description

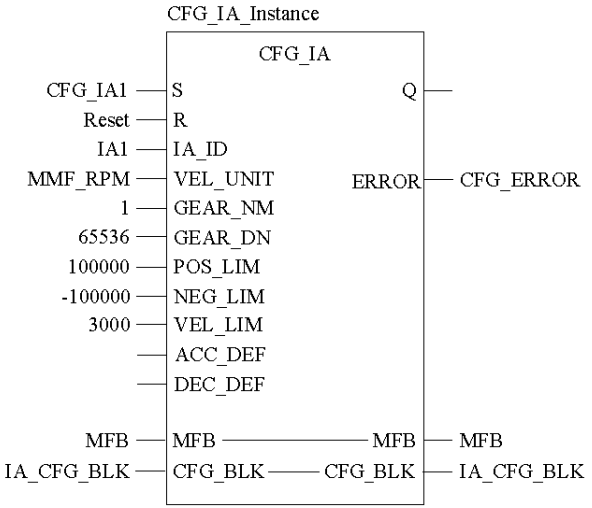
Function Description

This function block configures an ImaginaryAxis.

Additional parameters EN and ENO may be configured.

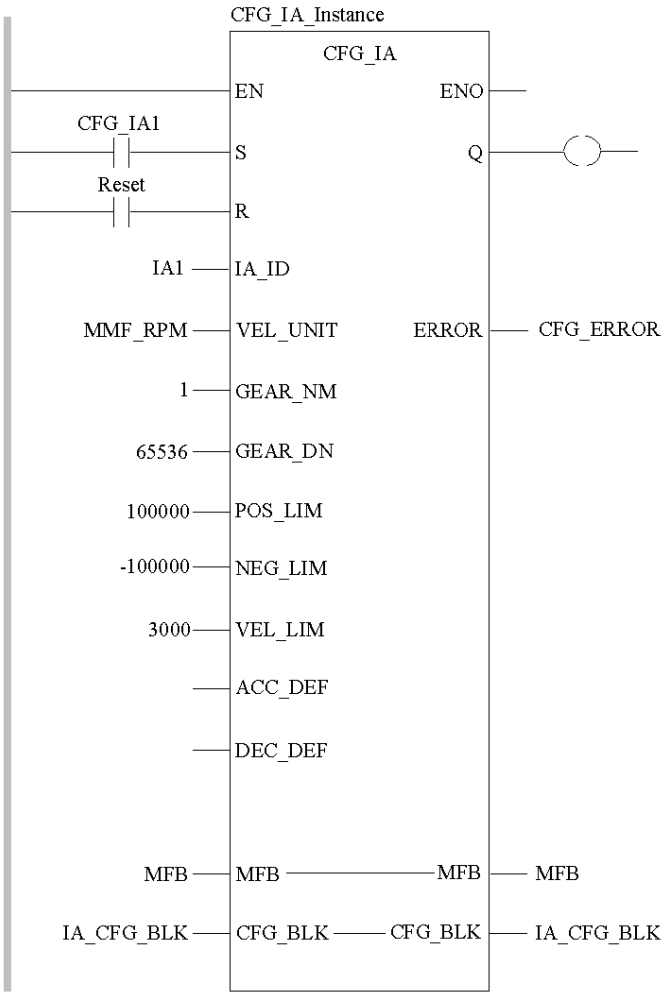
FBD Representation

Representation:



LD Representation

Representation:



IL Representation

Representation

```
CAL CFG_IA_Instance (S:=CFG_IA1, R:=Reset, IA_ID:=IA1,
    VEL_UNIT:=MMF_RPM, GEAR_NM:=1, GEAR_DN:=65536,
    POS_LIM:=100000, NEG_LIM:=-100000, VEL_LIM:=3000,
    MFB:=MFB, CFG_BLK:=IA_CFG_BLK, ERROR=>CFG_ERROR)
```

ST Representation

Representation

```
CFG_IA_Instance (S:=CFG_IA1, R:=Reset, IA_ID:=IA1,
    VEL_UNIT:=MMF_RPM, GEAR_NM:=1, GEAR_DN:=65536,
    POS_LIM:=100000, NEG_LIM:=-100000, VEL_LIM:=3000,
    MFB:=MFB, CFG_BLK:=IA_CFG_BLK, ERROR=>CFG_ERROR) ;
```

Parameter Description

Description of input parameters:

Parameter	Data Type	Meaning
S	BOOL	Rising input starts configuration.
R	BOOL	Resets block and prevents configuration. It halts configuration at the end of the current step.
IA_ID	INT	Axis ID for ImaginaryAxis to be configured.
VEL_UNIT	UDINT	Velocity units for this axis.
GEAR_NM	REAL	Number of axis units.
GEAR_DN	REAL	Number of drive units.
POS_LIM	REAL	Positive position limit (this is optional).
NEG_LIM	REAL	Negative position limit (this is optional).
VEL_LIM	REAL	Velocity limit (this is optional).
ACC_DEF	REAL	Default acceleration (this is optional).
DEC_DEF	REAL	Default deceleration (this is optional).

Description of in-/output parameters:

Parameter	Data Type	Meaning
MFB	MMFBlock *(see page 130)*	Must be connected to the MMFStart block of 200 registers.
CFG_BLK	AXCfg *(see page 138)*	Data block where the ImaginaryAxis configuration data will be stored.

Description of output parameters:

Parameter	Data Type	Meaning
Q	BOOL	True when the configuration has completed (reset by R).
ERROR	INT	First error code *(see page 143)* that is generated during configuration.

Example

When CFG_IA1 turns on, the configuration of IA1 occurs. The parameters assigned to the input pins are moved into the PLC register block IA_CFG_BLK, that is assigned at output pin CFG_BLK. For detailed information on the size and layout of this register data block, refer to the AXCfg Data Structure *(see page 138)*. The values placed in GEAR_NM and GEAR_DN represent the axis and drive units. In this example, the VEL_UNIT input pin is configured as RPM so the axis units are revolutions. The ImaginaryAxis drive units are set up as counts, 65536 counts per revolution. This function block configuration sets up the gear ratio to be 65536 counts of motor rotation, which equals one revolution of position change.

Runtime Errors

For a list of all error codes and values of the block, refer to *Tables of Error Codes for the Motion Library, page 164*.

Chapter 15
CFG_RA: RemoteAxis

Description

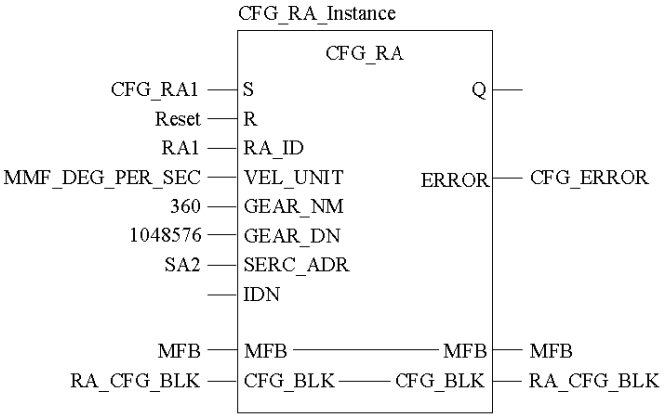
Function Description

This function block configures a RemoteAxis.

Additional parameters EN and ENO may be configured.

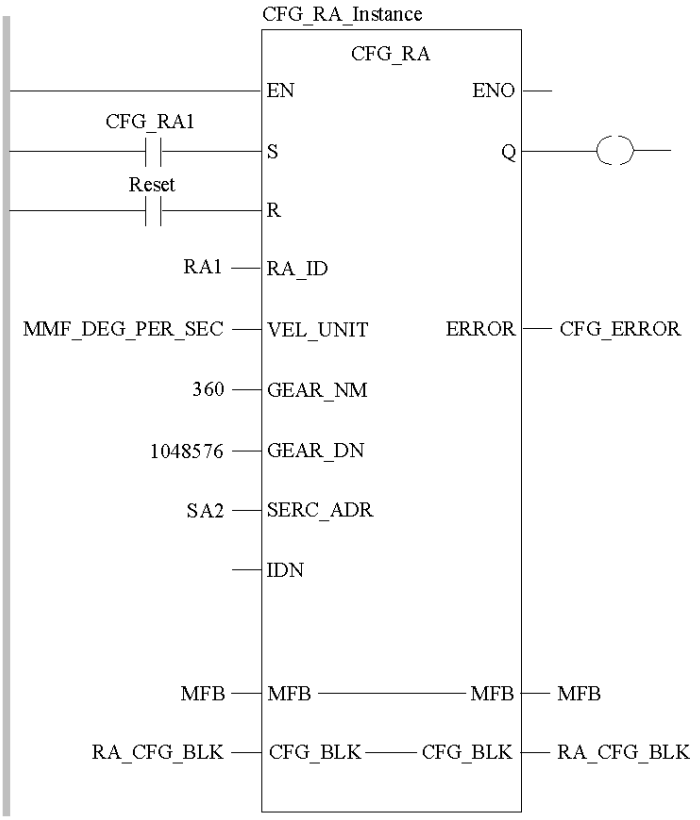
FBD Representation

Representation:



LD Representation

Representation:



IL Representation

Representation:

```
CAL CFG_RA_Instance (S:=CFG_RA1, R:=Reset, RA_ID:=RA1,
    VEL_UNIT:=MMF_DEG_PER_SEC, GEAR_NM:=360,
    GEAR_DN:=1048576, SERC_ADR:=SA2, MFB:=MFB,
    CFG_BLK:=RA_CFG_BLK, ERROR=>CFG_ERROR)
```

ST Representation

Representation:

```
CFG_RA_Instance (S:=CFG_RA1, R:=Reset, RA_ID:=RA1,
    VEL_UNIT:=MMF_DEG_PER_SEC, GEAR_NM:=360,
    GEAR_DN:=1048576, SERC_ADR:=SA2, MFB:=MFB,
    CFG_BLK:=RA_CFG_BLK, ERROR=>CFG_ERROR) ;
```

Parameter Description

Description of input parameters:

Parameter	Data Type	Meaning
S	BOOL	Rising input starts configuration.
R	BOOL	Resets block and prevents configuration. It halts configuration at the end of the current step.
RA_ID	INT	Axis ID for RemoteAxis to be configured.
VEL_UNIT	UDINT	Velocity units for this axis.
GEAR_NM	REAL	Number of position units per unit of measure.
GEAR_DN	REAL	Number of revolutions.
SERC_ADR	INT	Axis ID of the SercosAxis with secondary feedback device.
IDN	UINT	SERCOSIDN of secondary feedback device (optional).

Description of in-output parameters:

Parameter	Data Type	Meaning
MFB	MMFBlock *(see page 130)*	Must be connected to the MMFStart block of 200 registers.
CFG_BLK	AXCfg *(see page 138)*	Data block where the Remote axis configuration data will be stored.

Description of output parameters:

Parameter	Data Type	Meaning
Q	BOOL	True when the configuration has completed (reset by R).
ERROR	INT	First error code *(see page 143)* that is generated during configuration.

Example

When CFG_RA turns on, the configuration of RA1 occurs and the configuration data is placed in the PLC register data block RA_CFG_BLK that is assigned at output pin CFG_BLK. For detailed information on the size and layout of this register data block refer to the AXCfg Data Structure *(see page 138)*. During the creation of RA1, axis SA2 is interrogated for the feedback register number where the RemoteAxis input counts will be located. The values placed in GEAR_NM and GEAR_DN represent the axis and drive units, respectively. In this example, the VEL_UNIT input pin is configured as degrees per second, so the axis units are degrees. The remote feedback axis units are set up as counts -- 1048576 counts per revolution. This is due to SERCOSSpecification IEC 1491 (IDN 179). Therefore, this number is fixed when using a Lexium drive at the value of 1048576. This EFB configuration sets up the gear ratio to be 1048576 counts per revolution of position change.

Runtime Errors

For a list of all error codes and values of the block, refer to *Tables of Error Codes for the Motion Library, page 164*.

Chapter 16
CFG_SA: SercosAxis

Description

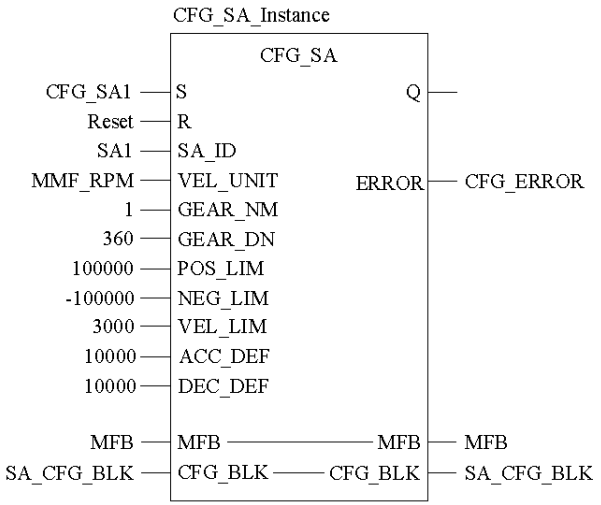
Function Description

This function block configures a SercosAxis.

Additional parameters EN and ENO may be configured.

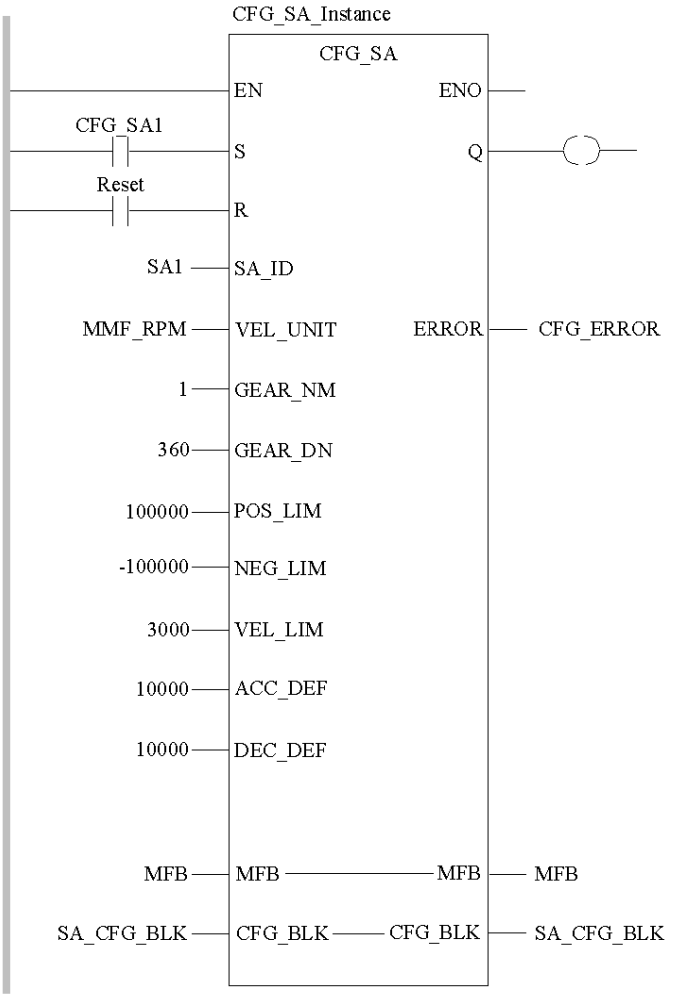
FBD Representation

Representation:



LD Representation

Representation:



IL Representation

Representation:

```
CAL CFG_SA_Instance (S:=CFG_SA1, R:=Reset, SA_ID:=SA1,
    VEL_UNIT:=MMF_RPM, GEAR_NM:=1, GEAR_DN:=360,
    POS_LIM:=100000, NEG_LIM:=-100000, VEL_LIM:=3000,
    ACC_DEF:=10000,  DEC_DEF:=10000, MFB:=MFB,
    CFG_BLK:=SA_CFG_BLK, ERROR=>CFG_ERROR)
```

ST Representation

Representation:

```
CFG_SA_Instance (S:=CFG_SA1, R:=Reset, SA_ID:=SA1,
    VEL_UNIT:=MMF_RPM, GEAR_NM:=1, GEAR_DN:=360,
    POS_LIM:=100000, NEG_LIM:=-100000, VEL_LIM:=3000,
    ACC_DEF:=10000,  DEC_DEF:=10000, MFB:=MFB,
    CFG_BLK:=SA_CFG_BLK, ERROR=>CFG_ERROR) ;
```

Parameter Description

Description of input parameters:

Parameter	Data Type	Meaning
S	BOOL	Rising input starts configuration.
R	BOOL	Resets block and prevents configuration. It halts configuration at the end of the current step.
SA_ID	INT	Axis ID for SercosAxis to be configured.
VEL_UNIT	UDINT	Velocity units for this axis.
GEAR_NM	REAL	Number of axis units.
GEAR_DN	REAL	Number of drive units.
POS_LIM	REAL	Positive position limit (this is optional).
NEG_LIM	REAL	Negative position limit (this is optional).
VEL_LIM	REAL	Velocity limit (this is optional).
ACC_DEF	REAL	Default acceleration (this is optional).
DEC_DEF	REAL	Default deceleration (this is optional).

Description of in-output parameters:

Parameter	Data Type	Meaning
MFB	MMFBlock *(see page 130)*	Must be connected to the MMFStart block of 200 registers.
CFG_BLK	AXCfg *(see page 138)*	Data block where the SERCOSaxis configuration data will be stored.

Description of output parameters:

Parameter	Data Type	Meaning
Q	BOOL	True when the configuration has completed (reset by R).
ERROR	INT	First error code *(see page 143)* that is generated during configuration.

Example

When CFG_SA1 turns on, the configuration of SA1 occurs. The parameters assigned to the input pins are moved into the PLC register block SA_CFG_BLK , that is assigned at output pin CFG_BLK. For detailed information on the size and layout of this register data block refer to the AXCfg Data Structure *(see page 138)*. The values placed in GEAR_NM and GEAR_DN represent the axis and drive units. In this example, the VEL_UNIT input pin is configured as RPM so the axis units are revolutions. During drive configuration, the drive units were set to degrees. This function block configuration will set up the gear ratio to be 360 degrees of motor rotation, which equals one revolution of position change.

Runtime Errors

For a list of all error codes and values of the block, refer to *Tables of Error Codes for the Motion Library, page 164*.

Chapter 17
DRV_DNLD: Download to SERCOSdrive

Description

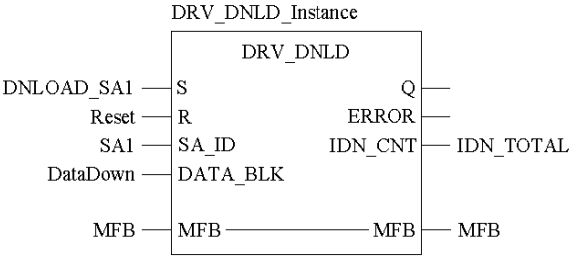
Function Description

This function block downloads information to a SERCOS drive from the PLC.

Additional parameters `EN` and `ENO` may be configured.

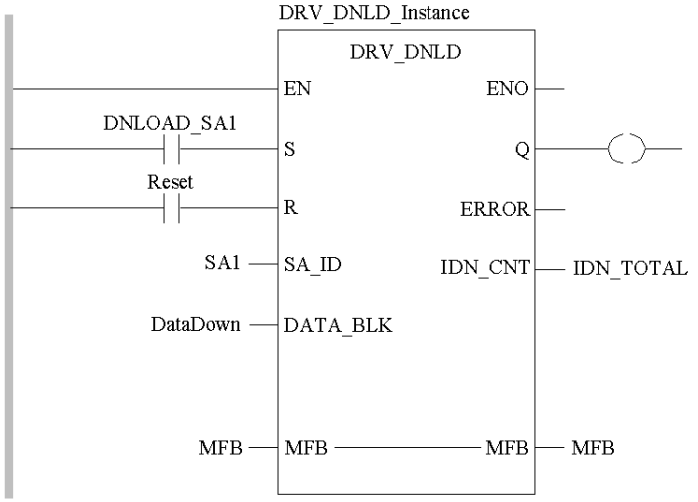
Representation in FBD

Representation:



Representation in LD

Representation:



Representation in IL

Representation:

```
CAL DRV_DNLD_Instance (S:=DNLOAD_SA1, R:=Reset, SA_ID:=SA1,
    DATA_BLK:=DataDown, MFB:=MFB, IDN_CNT=>IDN_TOTAL)
```

Representation in ST

Representation

```
DRV_DNLD_Instance (S:=DNLOAD_SA1, R:=Reset, SA_ID:=SA1,
    DATA_BLK:=DataDown, MFB:=MFB, IDN_CNT=>IDN_TOTAL) ;
```

Parameter Description

Description of input parameters:

Parameter	Data Type	Meaning
S	BOOL	Rising input starts the download to the drive.
R	BOOL	Resets block and prevents download (It halts the upload at the end of the current step).
SA_ID	INT	Axis ID for the SercosAxis to be downloaded.
DATA_BLK	ARRAY [1 .. n] OF UDINT	Array containing data to be downloaded (must be large enough to hold the amount of data to be downloaded).

Description of in-output parameters:

Parameter	Data Type	Meaning
MFB	MMFBlock *(see page 130)*	Must be connected to the MMFStart block of 200 registers.

Description of output parameters:

Parameter	Data Type	Meaning
Q	BOOL	True when the download has completed (reset by R).
ERROR	INT	First error code *(see page 143)* that is generated during download.
IDN_CNT	UDINT	The actual number of IDNs successfully downloaded to the drive (may be different to the number attempted due to normal errors).

Example

When DNLOAD_SA1 turns on, the drive parameters are downloaded from DataDown to SA1. The value placed in IDN_TOTAL is the total numbers of IDNs that were downloaded to the drive.

Runtime Errors

For a list of all error codes and values of the block, refer to *Tables of Error Codes for the Motion Library, page 164*.

Chapter 18
DRV_UPLD: Upload from SERCOSdrive

Description

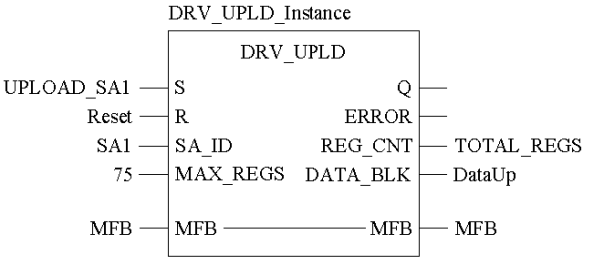
Function Description

This function block uploads information from a SERCOS drive to the PLC.

Additional parameters `EN` and `ENO` may be configured.

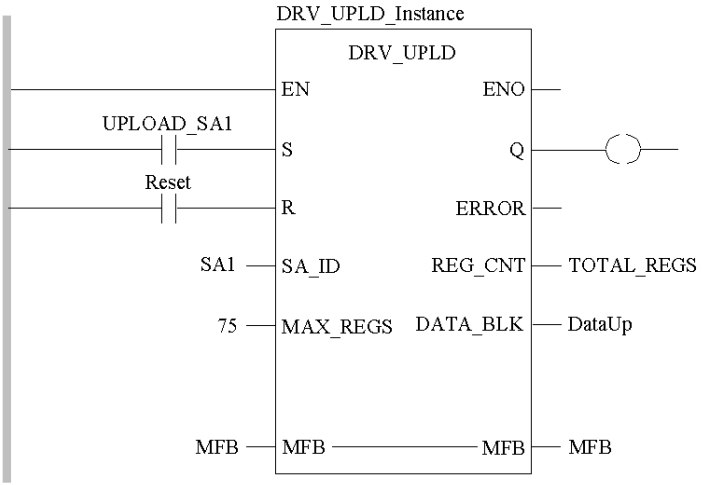
Representation in FBD

Representation:



Representation in LD

Representation:



Representation in IL

Representation:

```
CAL DRV_UPLD_Instance (S:=UPLOAD_SA1, R:=Reset,
    SA_ID:=SA1, MAX_REGS:=75, MFB:=MFB, REG_CNT=>TOTAL_REGS,
    DATA_BLK=>DataUp)
```

Representation in ST

Representation

```
DRV_UPLD_Instance (S:=UPLOAD_SA1, R:=Reset,
    SA_ID:=SA1, MAX_REGS:=75, MFB:=MFB, REG_CNT=>TOTAL_REGS,
    DATA_BLK=>DataUp) ;
```

Parameter Description

Description of input parameters:

Parameter	Data Type	Meaning
S	BOOL	Rising input starts the upload from the drive.
R	BOOL	Resets block and prevents upload (It halts the upload at the end of the current step).
SA_ID	INT	Axis ID for the SercosAxis to be uploaded.
MAX_REGS	UDINT	Maximum number of registers that may be uploaded (Each IDN consumes 3 registers plus one for overhead. If absent or zero, a default maximum of 601 registers is used.).

Description of in-output parameters:

Parameter	Data Type	Meaning
MFB	MMFBlock *(see page 130)*	Must be connected to the MMFStart block of 200 registers.

Description of output parameters:

Parameter	Data Type	Meaning
Q	BOOL	True when upload has completed (reset by R).
ERROR	INT	First error code *(see page 143)* that is generated during configuration.
REG_CNT	UDINT	Number of Registers actually used to hold uploaded data.
DATA_BLK	ARRAY [4 .. n] of UDINT	Array for storage of uploaded data (n must be ≥ number of IDNs uploaded * 3 + 1).

Example

When UPLOAD_SA1 turns on, the drive parameters are uploaded from SA1 and placed in the DataUp Array that is assigned at output pin DATA_BLK. The first register in the data block has the number (n) of IDNs that were uploaded. The next (n) registers have the IDN numbers. Following the IDN numbers is the IDN data. The data section of the register data block is in a REAL format and the data contained in these registers are raw (unconverted) values.

The length of the register data block is: Number of IDNs uploaded * 3 + 1

Runtime Errors

For a list of all error codes and values of the block, refer to *Tables of Error Codes for the Motion Library, page 164*.

Chapter 19
IDN_CHK: Check SERCOSIDN

Description

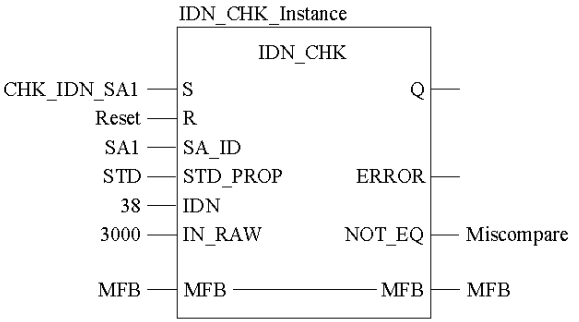
Function Description

This function block compares the data of the specified IDN on a SERCOSdrive to an expected value.

Additional parameters `EN` and `ENO` may be configured.

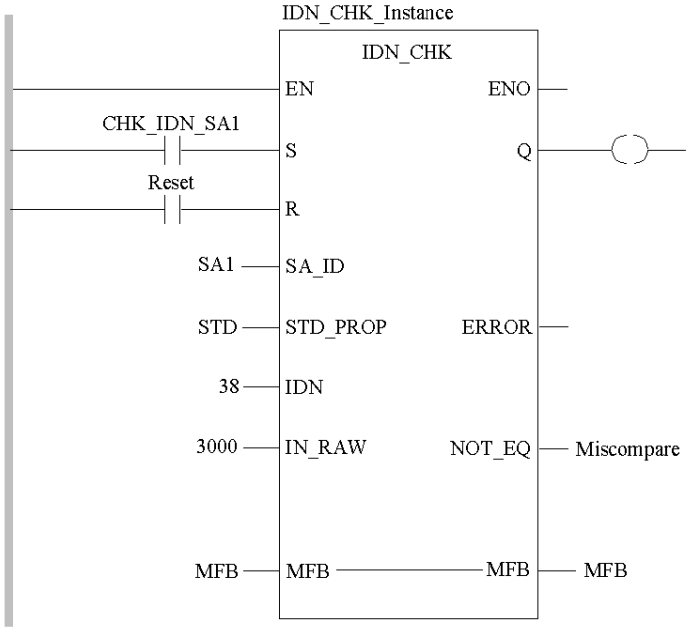
Representation in FBD

Representation:



Representation in LD

Representation:



Representation in IL

Representation:

```
CAL IDN_CHK_Instance (S:=CHK_IDN_SA1, R:=Reset,
    SA_ID:=SA1, STD_PROP:=STD, IDN:=38, IN_RAW:=3000,
    MFB:=MFB, NOT_EQ=>Miscompare)
```

Representation in ST

Representation

```
IDN_CHK_Instance (S:=CHK_IDN_SA1, R:=Reset,
    SA_ID:=SA1, STD_PROP:=STD, IDN:=38, IN_RAW:=3000,
    MFB:=MFB, NOT_EQ=>Miscompare) ;
```

Parameter Description

Description of input parameters:

Parameter	Data Type	Meaning
S	BOOL	Rising input checks SERCOSIDN.
R	BOOL	Resets block and prevents check.
SA_ID	INT	Axis ID for the SercosAxis to be checked.
STD_PROP	BOOL	Data value: Standard (0) IDN or Proprietary (1) IDN.
IDN	UINT	IDN of the data to check.
IN_RAW	UDINT	Expected data value.

Description of in-output parameters:

Parameter	Data Type	Meaning
MFB	MMFBlock *(see page 130)*	Must be connected to the MMFStart block of 200 registers.

Description of output parameters:

Parameter	Data Type	Meaning
Q	BOOL	True when the check has completed (reset by R).
ERROR	INT	First error code *(see page 143)* that is generated during the check.
NOT_EQ	BOOL	True if drive data does not match the IN_RAW.

Example

When CHK_IDN_SA1 turns on, the value of IDN 38 (positive velocity limit) in drive SA1 is compared with the value 3000. If the value in the drive is not 3000, output signal Miscompare will turn on.

Runtime Errors

For a list of all error codes and values of the block, refer to *Tables of Error Codes for the Motion Library, page 164*.

Chapter 20
IDN_XFER: Transfer to and from SERCOSdrive

Description

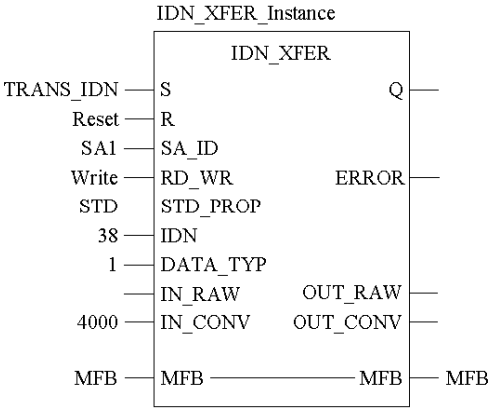
Function Description

This function block transfers 16-bit and 32-bit SERCOSdata via SERCOSIDNs to and from the SERCOSdrive.

Additional parameters `EN` and `ENO` may be configured.

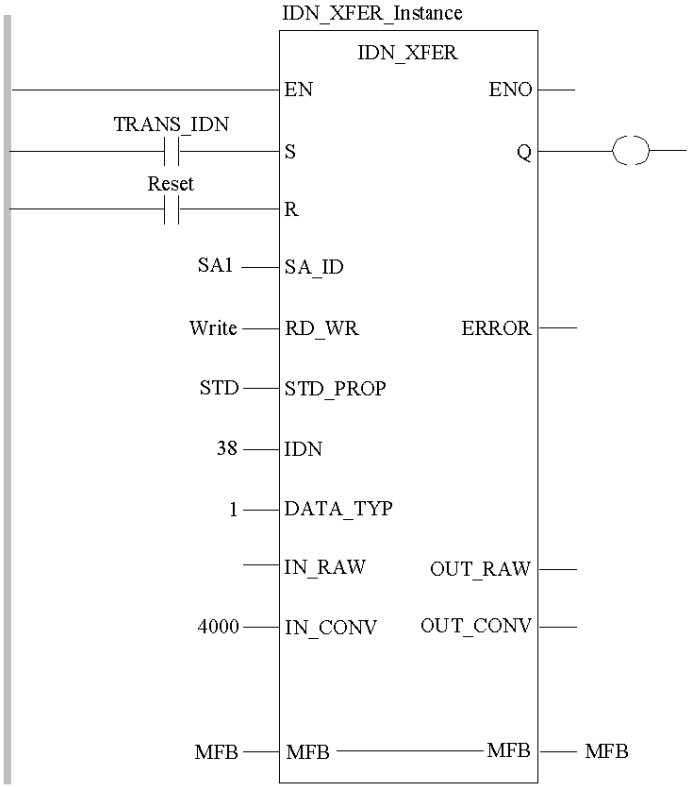
Representation in FBD

Representation:



Representation in LD

Representation:



Representation in IL

Representation:

```
CAL IDN_XFER_Instance (S:=TRANS_IDN, R:=Reset, SA_ID:=SA1,
    RD_WR:=Write, STD_PROP:=STD, IDN:=38, DATA_TYP:=1,
    IN_CONV:=4000, MFB:=MFB)
```

Representation in ST

Representation

```
IDN_XFER_Instance (S:=TRANS_IDN, R:=Reset, SA_ID:=SA1,
    RD_WR:=Write, STD_PROP:=STD, IDN:=38, DATA_TYP:=1,
    IN_CONV:=4000, MFB:=MFB) ;
```

Parameter Description

Description of input parameters:

Parameter	Data Type	Meaning
S	BOOL	Rising input transfers data to and from the SERCOSdrive.
R	BOOL	Resets block and prevents the data transfer.
SA_ID	INT	Axis ID for the SercosAxis.
RD_WR	BOOL	Read (0) or write (1) IDN data.
STD_PROP	BOOL	Standard (0) or Proprietary (1) IDN.
IDN	UINT	IDN of the SERCOSdata to be transfered.
DATA_TYP	BOOL	Type of data to be transfered: 0 = Drive units 1 = Converted by motion control
IN_RAW	UDINT	Data written to SERCOSdrive directly.
IN_CONV	REAL	Data written to SERCOSdrive with conversion.

Description of in-output parameters:

Parameter	Data Type	Meaning
MFB	MMFBlock *(see page 130)*	Must be connected to the MMFStart block of 200 registers.

Description of output parameters:

Parameter	Data Type	Meaning
Q	BOOL	True when the transfer has completed (reset by R).
ERROR	INT	First error code *(see page 143)* that is generated during this read.
OUT_RAW	UDINT	Data read directly from the SERCOSdrive.
OUT_CONV	REAL	Drive data that has been converted by the motion controller.

Example

When TRANS_IDN turns on, the value 4000 for IDN 38 (positive velocity limit) is transferred to drive SA1. With input pin DATA_TYP set to 1, the value 4000 is a motion controller-converted number, so it is placed in input pin IN_CONV. If the data is a raw (unconverted) drive value, then a 0 is entered for DATA_TYP and the known raw (unconverted) value would be at input pin IN_RAW.

Runtime Errors

For a list of all error codes and values of the block, refer to *Tables of Error Codes for the Motion Library, page 164*.

Chapter 21
MMF_BITS: Control and Status Bits

Description

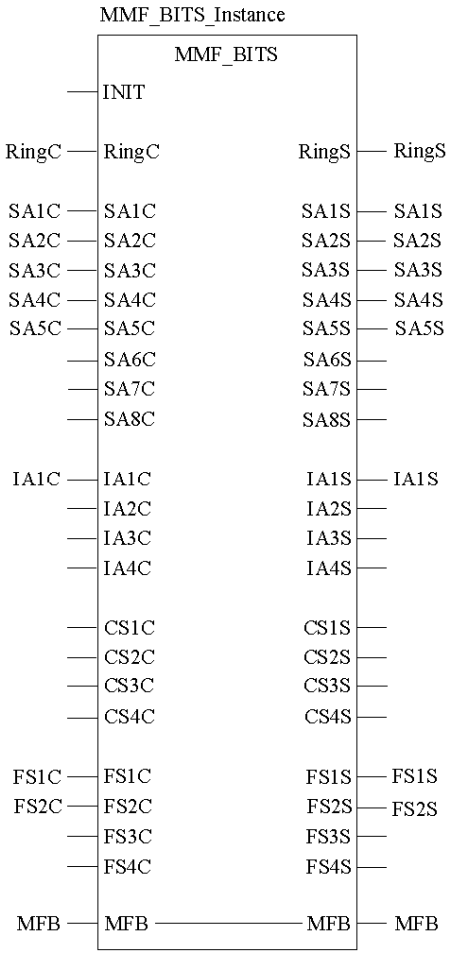
Function Description

This function block makes the MMF MotionControl bits and MMF MotionStatus bits available in Unity Pro.

Additional parameters `EN` and `ENO` may be configured.

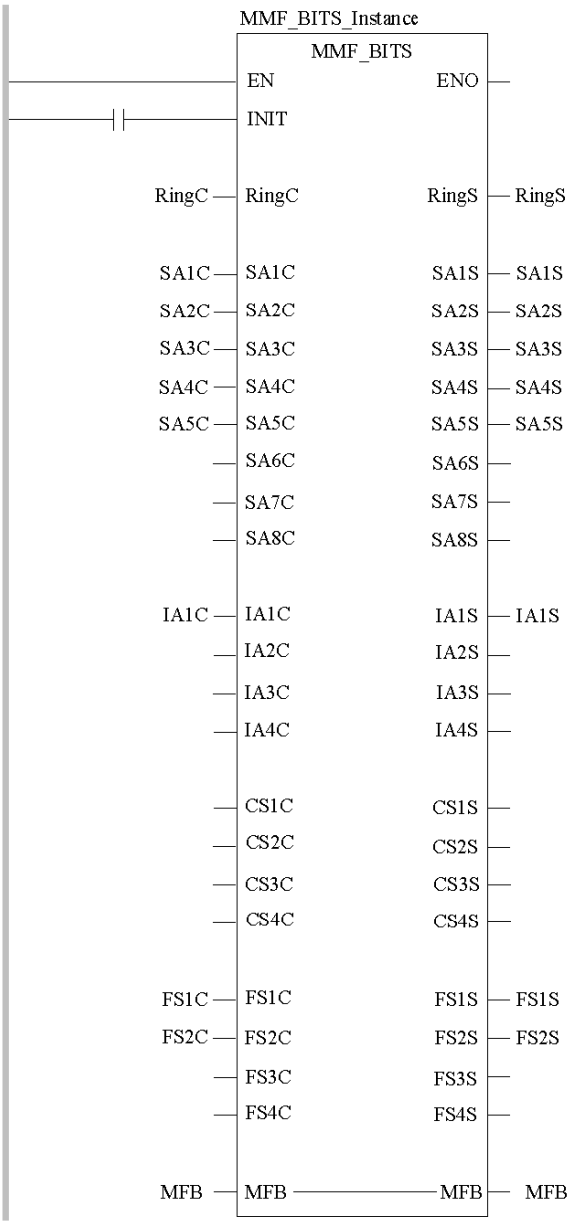
FBD Representation

Representation:



LD Representation

Representation:



IL Representation

Representation:

```
CAL MMF_BITS_Instance (RingC:=RingC, SA1C:=SA1C, SA2C:=SA2C,
    SA3C:=SA3C, SA4C:=SA4C, SA5C:=SA5C, IA1C:=IA1C,
    FS1C:=FS1C, FS2C:=FS2C, MFB:=MFB, RingS=>RingS,
    SA1S=>SA1S, SA2S=>SA2S, SA3S=>SA3S, SA4S=>SA4S,
    SA5S=>SA5S, IA1S=>IA1S, FS1S=>FS1S, FS2S=>FS2S)
```

ST Representation

Representation:

```
MMF_BITS_Instance (RingC:=RingC, SA1C:=SA1C, SA2C:=SA2C,
    SA3C:=SA3C, SA4C:=SA4C, SA5C:=SA5C, IA1C:=IA1C,
    FS1C:=FS1C, FS2C:=FS2C, MFB:=MFB, RingS=>RingS,
    SA1S=>SA1S, SA2S=>SA2S, SA3S=>SA3S, SA4S=>SA4S,
    SA5S=>SA5S, IA1S=>IA1S, FS1S=>FS1S, FS2S=>FS2S) ;
```

Parameter Description

Description of input parameters:

Parameter	Data Type	Meaning
INIT	BOOL	Rising input forces all Control registers to FFFF0000 (all ALLOW bits on; all CONTROL bits off) and initializes the attached MMFControlBits as well (This is unconventional but useful writing of inputs.).
RingC	MMFControlBits *(see page 133)*	Control for the SERCOSRing.
SA1C - SA8C	MMFControlBits	Control for SercosAxes 1...8.
IA1C - IA4C	MMFControlBits	Control for ImaginaryAxes 1...4.
CS1C - CS4C	MMFControlBits	Control for CoordinatedSets 1...4.
FS1C - FS4C	MMFControlBits	Control for Follower Sets 1...4.

Description of in-output parameters:

Parameter	Data Type	Meaning
MFB	MMFBlock *(see page 130)*	Must be connected to the MMFStart block of 200 registers (usually named MFB).

Description of output parameters:

Parameter	Data Type	Meaning
RingS	MMFStatusBits *(see page 134)*	Status from the SERCOSRing.
SA1S - SA8S	MMFStatusBits	Status for SercosAxes 1...8.
IA1S - IA4S	MMFStatusBits	Status for ImaginaryAxes 1...4.
CS1S - CS4S	MMFStatusBits	Status for CoordinatedSets 1...4.
FS1S - FS4S	MMFStatusBits	Status for FollowerSets 1...4.

Example

This function block allows Unity Pro programmers to use Booleans (coils and contacts) to effect Control bits and evaluate Status bits. For instance, the contact for in position for axis 5 would be SA5S.IN_POSITION. To clear faults on the first ImaginaryAxis, energize coil IA1C.CONTROL_CLEAR_FAULT.

Chapter 22
MMF_ESUB: Subroutine Using Extended Parameters

Description

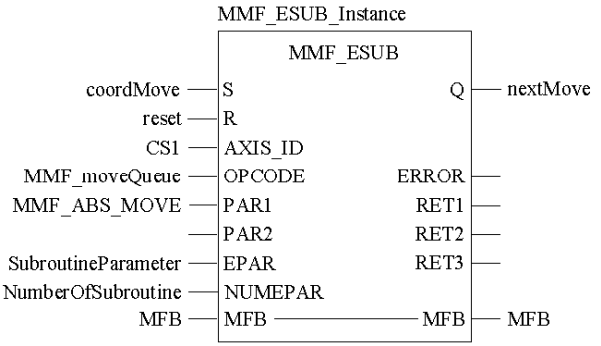
Function Description

Issues an MMFStart subroutine using extended parameters.

Additional parameters `EN` and `ENO` may be configured.

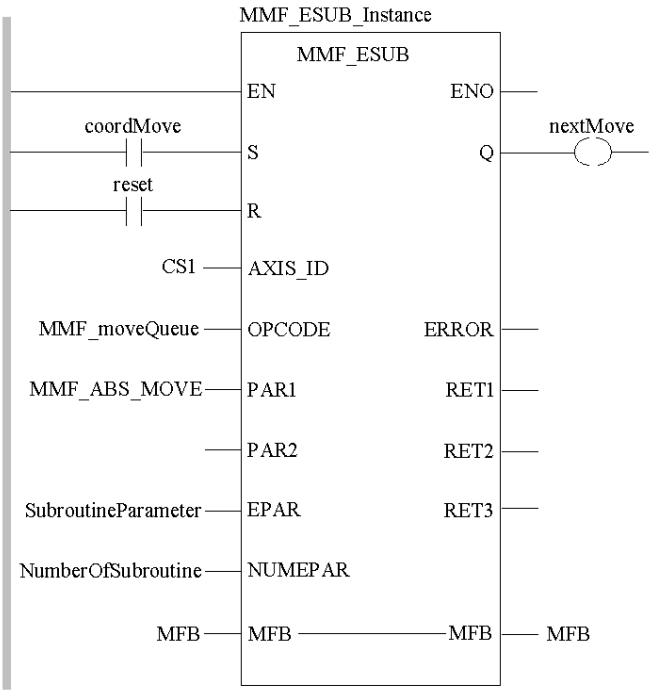
FBD Representation

Representation:



LD Representation

Representation:



IL Representation

Representation:

```
CAL MMF_ESUB_Instance (S:=coordMove, R:=reset,
    AXIS_ID:=CS1, OPCODE:=MMF_moveQueue,
    PAR1:=MMF_ABS_MOVE, EPAR:=SubroutineParameter,
    NUMEPAR:=NumberOfSubroutine, MFB:=MFB, Q=>nextMove)
```

ST Representation

Representation:

```
MMF_ESUB_Instance (S:=coordMove, R:=reset,
    AXIS_ID:=CS1, OPCODE:=MMF_moveQueue,
    PAR1:=MMF_ABS_MOVE, EPAR:=SubroutineParameter,
    NUMEPAR:=NumberOfSubroutine, MFB:=MFB, Q=>nextMove) ;
```

Parameter Description

Description of input parameters:

Parameter	Data Type	Meaning
S	BOOL	Rising input starts subroutine.
R	BOOL	Resets block and prevents subroutine from starting (does not halt a subroutine in progress).
AXIS_ID	INT	Axis ID for this subroutine.
OPCODE	INT	Unique number of the subroutine to be executed.
PAR1	UDINT	First parameter for this subroutine.
PAR2	UDINT	Second parameter for this subroutine.
EPAR	ARRAY [1..16] OF REAL	Extended parameter for this subroutine.
NUMEPAR	UINT	NUMEPAR value is the **total number** of position **and** velocity and must be an even number. The valid range is 2 to 16.The consequence of erroneous NUMEPAR value can lead to an uncontroled speed. **CAUTION:** Not controlling the speed could lead to overshoot the move.

Description of in-output parameters:

Parameter	Data Type	Meaning
MFB	MMFBlock *(see page 130)*	Must be connected to the MMFStart block of 200 registers (usually named MFB).

Description of output parameters:

Parameter	Data Type	Meaning
Q	BOOL	True when this subroutine has been recognized by motion controller (reset by R).
ERROR	INT	First error code *(see page 143)* that is generated upon attempting to start this subroutine (written only when Q rises).
RET1	UDINT	First return value from this subroutine.
RET2	REAL	Second return value from this subroutine.
RET3	REAL	Third return value from this subroutine.

Example

This function block is designed specifically for executing `moveImmed` and `moveQueue` subroutines with Coordinated Sets. `PAR1` specifies the MoveType (Absolute or Incremental) and `EPAR` takes the Position for all the "`n`" axes in the Coordinated Set. Then `EPAR` takes the Velocity of all "`n`" axes in the Coordinated Set, up to eight axes. `NUMEPAR` defines how many of the array elements are to be used. The `NUMEPAR` value is the total number of position and velocity and must be an even number. The valid range is 2 to 16. If the user gives a value out of bounds the input is forced to either the max. or min. value according to which boundary is exceeded. For these "move" subroutines, `PAR2` is not used and there are no Return values, but they are included in the EFB for future subroutines.

Runtime Errors

For a list of all error codes and values of the block, refer to *Tables of Error Codes for the Motion Library, page 164*.

Chapter 23
MMF_INDX: Immediate Incremental Move

Description

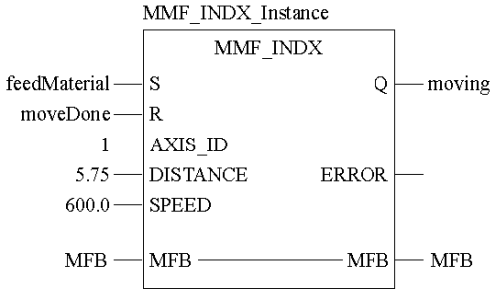
Function Description

This function block issues an MMFStart Immediate Incremental Move.

Additional parameters EN and ENO may be configured.

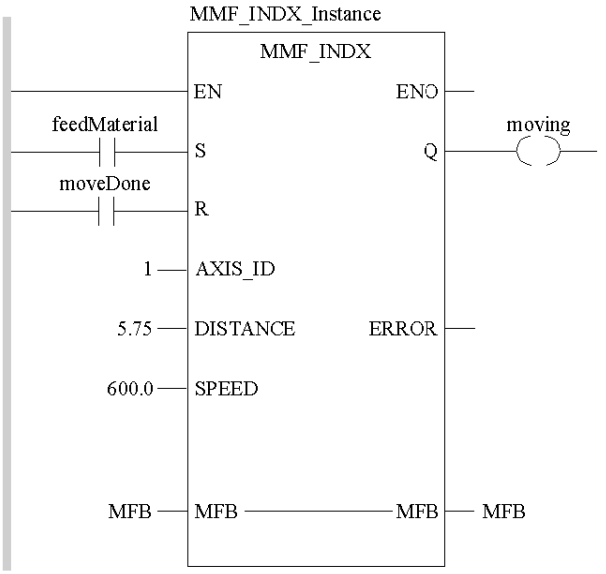
FBD Representation

Representation:



LD Representation

Representation:



IL Representation

```
CAL MMF_INDX_Instance (S:=feedMaterial, R:=moveDone,
    AXIS_ID:=1, DISTANCE:=5.75, SPEED:=600.0, MFB:=MFB,
    Q=>moving)
```

ST Representation

```
MMF_INDX_Instance (S:=feedMaterial, R:=moveDone,
    AXIS_ID:=1, DISTANCE:=5.75, SPEED:=600.0, MFB:=MFB,
    Q=>moving) ;
```

Parameter Description

Description of input parameters:

Parameter	Data Type	Meaning
S	BOOL	Rising input starts move.
R	BOOL	Resets block and prevents move from starting (it does not halt a move in progress).
AXIS_ID	INT	Single Axis ID for this move.
DISTANCE	REAL	Length of this incremental move.
SPEED	REAL	Constant speed of the move.

Description of in-output parameters:

Parameter	Data Type	Meaning
MFB	MMFBlock *(see page 130)*	Must be connected to the MMFStart block of 200 registers (usually named MFB).

Description of output parameters:

Parameter	Data Type	Meaning
Q	BOOL	True when move subroutine has been recognized by motion controller (reset by R).
ERROR	INT	First error code *(see page 143)* that is generated upon attempting to start this move.

Example

feedMaterial turns on and causes axis number 1 to index by 5.75 units at a speed of 600. When the axis is IN_POSITION the moveDone coil turns on to reset the MMF_INDX block.

Runtime Errors

For a list of all error codes and values of the block, refer to *Tables of Error Codes for the Motion Library, page 164*.

Chapter 24
MMF_JOG: JOG

Description

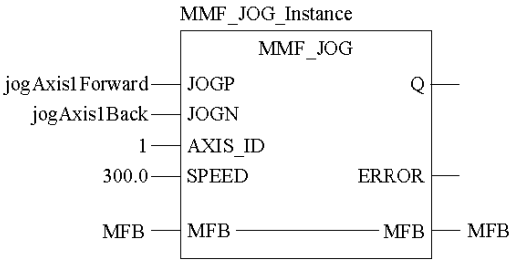
Function Description

Jogs an axis using MMFStart Immediate Continuous Move and Halt.

Additional parameters EN and ENO may be configured.

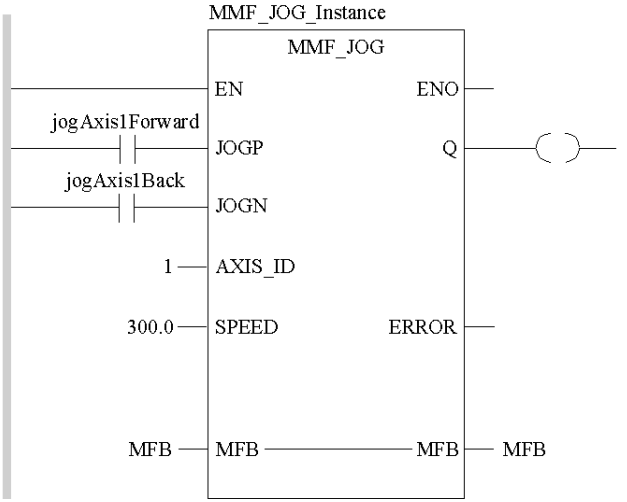
FBD Representation

Representation:



LD Representation

Representation:



IL Representation

```
CAL MMF_JOG_Instance (JOGP:=jogAxis1Forward,
    JOGN:=jogAxis1Back, AXIS_ID:=1, SPEED:=300.0,
    MFB:=MFB)
```

ST Representation

```
MMF_JOG_Instance (JOGP:=jogAxis1Forward,
    JOGN:=jogAxis1Back, AXIS_ID:=1, SPEED:=300.0,
    MFB:=MFB) ;
```

Parameter Description

Description of input parameters.

Parameter	Date Type	Meaning
JOGP	BOOL	Rising input starts positive motion; falling input halts motion.
JOGN	BOOL	Rising input starts negative motion; falling input halts motion.
AXIS_ID	INT	Single Axis ID to jog.
SPEED	REAL	Constant jog speed.

Description of in-output parameters.

Parameter	Data Type	Meaning
MFB	MMFBlock *(see page 130)*	Must be connected to the MMFStart block of 200 registers (usually named MFB).

Description of output parameters.

Parameter	Data Type	Meaning
Q	BOOL	True when jog is issued; reset after issuing a halt.
ERROR	INT	First error code *(see page 143)* that is generated upon attempting to move or halt.

Example

jogAxis1Forward is mapped to a button on the operator's panel. When the input is on, as shown, the motor will move in the positive direction at a velocity of 300. This could be RPMs or inches per minute, and so on. When the button is released the axis will halt.

Runtime Errors

For a list of all error codes and values of the block, refer to *Tables of Error Codes for the Motion Library, page 164*.

Chapter 25
MMF_MOVE: Absolute Move

Description

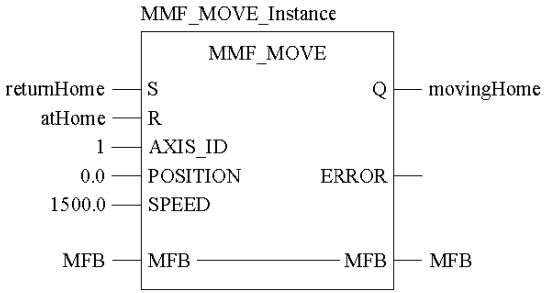
Function Description

This function block issues an MMFStart Immediate Absolute Move.

Additional parameters EN and ENO may be configured.

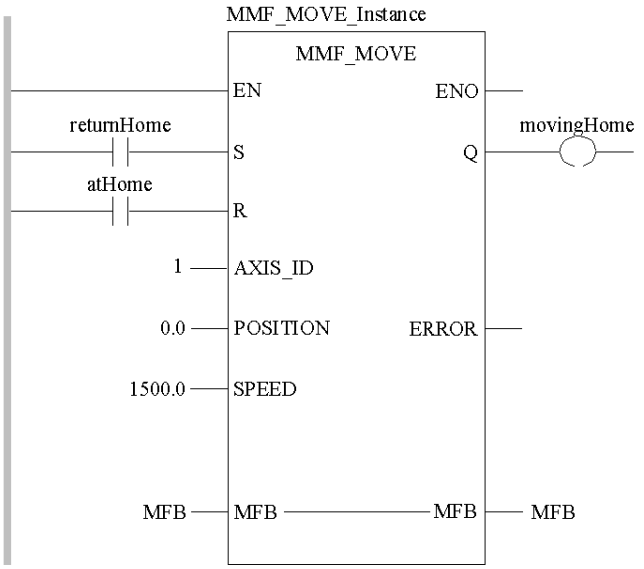
FBD Representation

Representation:



LD Representation

Representation:



IL Representation

```
CAL MMF_MOVE_Instance (S:=returnHome, R:=atHome, AXIS_ID:=1,
    POSITION:=0.0, SPEED:=1500.0, MFB:=MFB, Q=>movingHome)
```

ST Representation

```
MMF_MOVE_Instance (S:=returnHome, R:=atHome, AXIS_ID:=1,
    POSITION:=0.0, SPEED:=1500.0, MFB:=MFB, Q=>movingHome) ;
```

Parameter Description

Description of input parameters.

Parameter	Data Type	Meaning
S	BOOL	Rising input starts move.
R	BOOL	Resets block and prevents move from starting (it does not halt a move in progress).
AXIS_ID	INT	Single Axis ID for this move.
POSITION	REAL	Absolute Position to move to.
SPEED	REAL	Constant speed of the move.

Description of in-output parameters.

Parameter	Data Type	Meaning
MFB	MMFBlock *(see page 130)*	Must be connected to the MMFStart block of 200 registers location (usually named MFB).

Description of output parameters.

Parameter	Data Type	Meaning
Q	BOOL	True when move subroutine has been recognized by motion controller (reset by R).
ERROR	INT	First error code *(see page 143)* that is generated upon attempting to start this move.

Example

returnHome turns on to command an absolute move to zero for axis number 1. atHome turns on when Q turns on and the axis IN_POSITION bit is set. This causes the MMF_MOVE block to be reset.

Runtime Errors

For a list of all error codes and values of the block, refer to *Tables of Error Codes for the Motion Library, page 164*.

Chapter 26
MMF_RST: Reset

Description

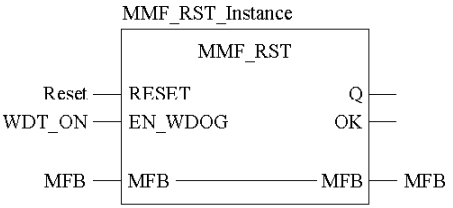
Function Description

This function block resets the subroutine interface to the motion controller and all MMFStart function blocks.

Additional parameters EN and ENO may be configured.

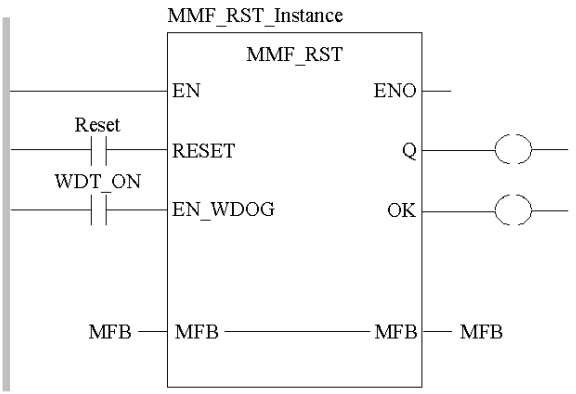
Representation in FBD

Representation:



Representation in LD

Representation:



Representation in IL

Representation:

```
CAL MMF_RST_Instance (RESET:=Reset, EN_WDOG:=WDT_ON,
    MFB:=MFB)
```

Representation in ST

Representation

```
MMF_RST_Instance (RESET:=Reset, EN_WDOG:=WDT_ON,
    MFB:=MFB) ;
```

Parameter Description

Description of input parameters:

Parameter	Data Type	Meaning
RESET	BOOL	Rising input starts reset process. High input inhibits any subroutine from being initiated. Falling input has no effect, except to clear Q.
EN_WDOG	BOOL	Enable watchdog timer. High enables the watchdog timer (WDT). Low disables the WDT. A transition from High to Low stops the WDT. The motion controller will interpret this as a WDT fault and will stop and disable the drive.

Description of in-output parameters:

Parameter	Data Type	Meaning
MFB	MMFBlock *(see page 130)*	Must be connected to the MMFStart block of 200 registers.

Description of output parameters:

Parameter	Data Type	Meaning
Q	BOOL	True when reset process is complete.
OK	BOOL	Low during the reset process. High when the motion controller reports the correct revolution number.

Example

When Reset is turned on, all MMFStart blocks are reset. When WDT_ON is Low, the watchdog function is disabled. When WDT_ON turns on, the watchdog between the PLC and SERCOScontroller interprets this as a watchdog fault, and it stops and disables the drives.

Chapter 27
MMF_SUB: Standard Subroutine

Description

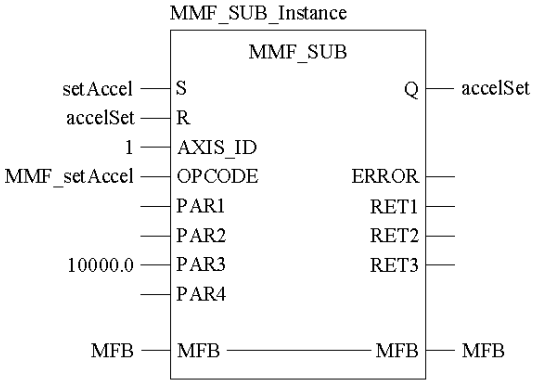
Function Description

This function block can be used to execute any MMF standard subroutine except moves to CoordinatedSets (see `MMF_ESUB` *(see page 103)*).

Additional parameters `EN` and `ENO` may be configured.

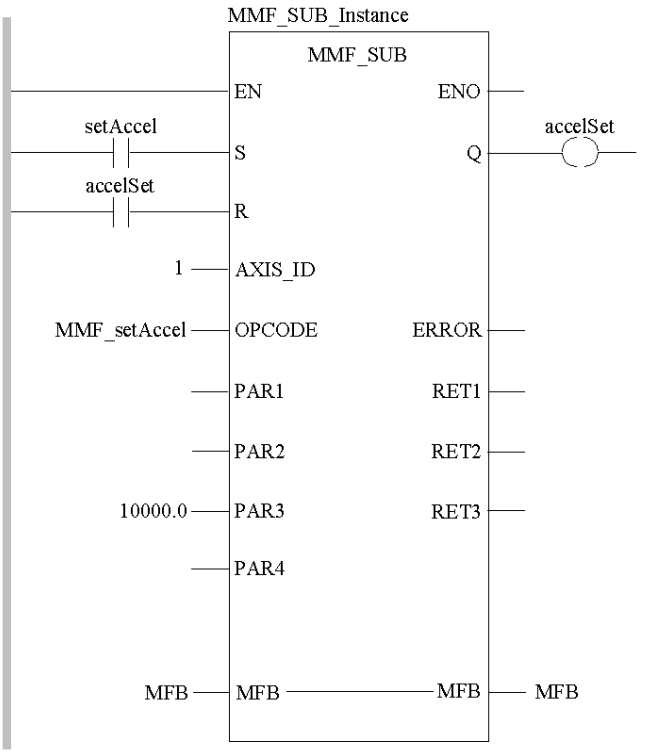
FBD Representation

Representation:



LD Representation

Representation:



IL Representation

Representation:

```
CAL MMF_SUB_Instance (S:=setAccel, R:=accelSet, AXIS_ID:=1,
    OPCODE:=MMF_setAccel, PAR3:=10000.0, MFB:=MFB,
    Q=>accelSet)
```

ST Representation

Representation:

```
MMF_SUB_Instance (S:=setAccel, R:=accelSet, AXIS_ID:=1,
    OPCODE:=MMF_setAccel, PAR3:=10000.0, MFB:=MFB,
    Q=>accelSet) ;
```

Parameter Description

Description of input parameters.

Parameter	Data Type	Meaning
S	BOOL	Rising input starts subroutine.
R	BOOL	Resets block and prevents subroutine from starting (does not halt a subroutine in progress.)
AXIS_ID	INT	Axis ID for this subroutine.
OPCODE	INT	Unique number of the subroutine to be executed.
PAR1	UDINT	First parameter for this subroutine.
PAR2	UDINT	Second parameter for this subroutine.
PAR3	REAL	Third parameter for this subroutine.
PAR4	REAL	Fourth parameter for this subroutine.

Description of in-output parameters.

Parameter	DataType	Meaning
MFB	MMFBlock *(see page 130)*	Must be connected to the MMFStart block of 200 registers (usually named MFB).

Description of output parameters.

Parameter	DataType	Meaning
Q	BOOL	True when this subroutine has been recognized by the motion controller (reset by R).
ERROR	INT	First error code *(see page 143)* that is generated upon attempting to start this subroutine (written only when Q rises).
RET1	UDINT	First return value from this subroutine.
RET2	REAL	Second return value from this subroutine.
RET3	REAL	Third return value from this subroutine.

Example

The Boolean setAccel turns on, and the new value for the acceleration is written to axis number 1. This block can also be used to get the current settings for parameters in the motion controller. For a list of the correct parameters associated with any Motion C function refer to your MMF user manual.

Runtime Errors

For a list of all error codes and values of the block, refer to *Tables of Error Codes for the Motion Library, page 164*.

Chapter 28
MMF_USUB: User Subroutine

Description

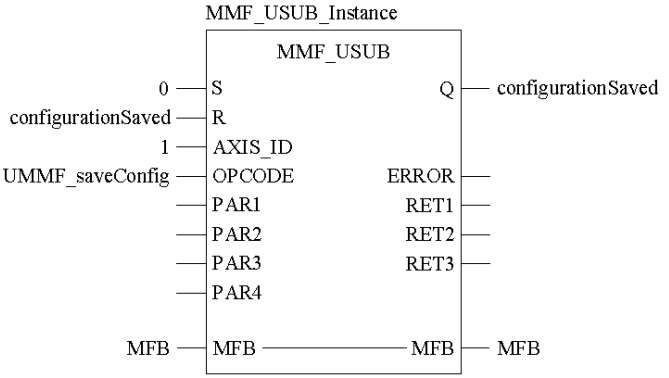
Function Description

This function block can execute user subroutines. User subroutines are special built-in routines constructed to simplify a multi-step process.

Additional parameters `EN` and `ENO` may be configured.

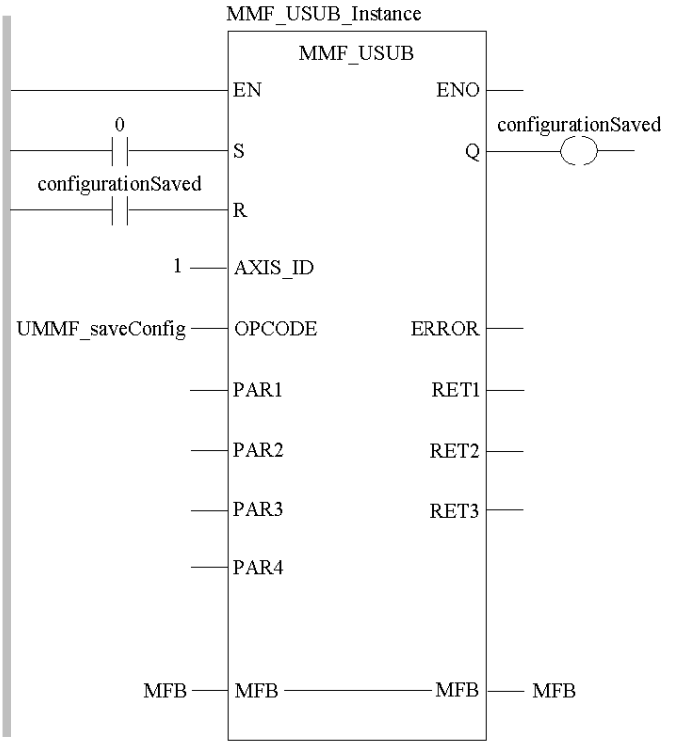
Representation in FBD

Representation:



Representation in LD

Representation:



Representation in IL

Representation:

```
CAL MMF_USUB_Instance (S:=0, R:=configurationSaved,
    AXIS_ID:=1, OPCODE:=UMMF_saveConfig,
    MFB:=MFB, Q=>configurationSaved)
```

Representation in ST

Representation

```
MMF_USUB_Instance (S:=0, R:=configurationSaved,
    AXIS_ID:=1, OPCODE:=UMMF_saveConfig,
    MFB:=MFB, Q=>configurationSaved) ;
```

Parameter Description

Description of input parameters:

Parameter	Data Type	Meaning
S	BOOL	Rising input starts subroutine.
R	BOOL	Resets block and prevents subroutine from starting (does not halt a subroutine in progress).
AXIS_ID	INT	Axis ID for this subroutine.
OPCODE	INT	Unique number of the subroutine to be executed.
PAR1	UDINT	First parameter for this subroutine.
PAR2	UDINT	Second parameter for this subroutine.
PAR3	REAL	Third parameter for this subroutine.
PAR4	REAL	Fourth parameter for this subroutine.

Description of in-output parameters:

Parameter	Data Type	Meaning
MFB	MMFBlock *(see page 130)*	Must be connected to the MMFStart block of 200 registers (usually named MFB).

Description of output parameters:

Parameter	Data Type	Meaning
Q	BOOL	True when this subroutine has been recognized by motion controller (reset by R).
ERROR	INT	First error code *(see page 143)* that is generated upon attempting to start this subroutine (written only when Q rises).
RET1	UDINT	First return value from this subroutine.
RET2	REAL	Second return value from this subroutine.
RET3	REAL	Third return value from this subroutine.

Runtime Errors

For a list of all error codes and values of the block, refer to *Tables of Error Codes for the Motion Library, page 164*.

Chapter 29
MMF Derived Datatypes and Warning/Error Codes

Overview

This chapter describes the derived datatypes and warning/error codes used by the MMF function blocks.

What Is in This Chapter?

This chapter contains the following topics:

MMF Derived Datatypes

Overview

The following part describes the structure of the Derived Datatypes (DDT) used by the function blocks of the MMF family..

EFBs using DDTs

Usage of DDTs:

Type	Used by EFB
MMFBlock	MMF_USUB, MMF_ESUB, MMF_SUB, MMF_MOVE, MMF_JOG, MMF_INDX, CFG_SA, CFG_IA, CFG_RA, CFG_CS, CFG_FS, CFG_CP_F, CFG_CP_V, DRV_UPLD, DRV_DNLD, IDN_CHK, IDN_XFER, MMF_RST
MMFControlBits	MMF_BITS
MMFStatusBits	MMF_BITS
AXCfg	CFG_SA, CFG_IA, CFG_RA
CSCfg	CFG_CS
CPCfg	CFG_CP_F, CFG_CP_V
FSCfg	CFG_FS

MMFBlock

Elements of the MMFBlock Data Structure:

Element	Datatype
RingControl	UDINT
WatchDogCont	INT
Debug	INT
SubNumber	INT
AxisID	INT
Parameter1	UDINT
Parameter2	UDINT
Parameter3	REAL
Parameter4	REAL
Parameter	ARRAY [5..18] OF REAL
reserved	ARRAY [1..8] OF INT
SA1Control	UDINT

Element	Datatype
…	
SA8Control	UDINT
IA1Control	UDINT
…	
IA4Control	UDINT
CS1Control	UDINT
…	
CS4Control	UDINT
FS1Control	UDINT
…	
FS4Control	UDINT
USubNumber	INT
UAxisID	INT
UParameter1	UDINT
UParameter2	UDINT
UParameter3	REAL
UParameter4	REAL
RingStatus	UDINT
WatchDogStat	INT
NumberOfAxes	INT
FaultAxis	INT
FaultCode	INT
WarnAxis	INT
WarnCode	INT
SubNumEcho	INT
AxisIDEcho	INT
Error	INT
Return1	UDINT
Return2	REAL
Return3	REAL
RevNumber	INT
SA1Position	REAL
…	
SA8Position	REAL

Element	Datatype
IA1Position	REAL
...	L
IA4Position	REAL
RA1Position	REAL
...	
RA4Position	REAL
SA1Status	UDINT
...	
SA8Status	UDINT
IA1Status	UDINT
...	
IA4Status	UDINT
CS1Status	UDINT
...	
CS4Status	UDINT
FS1Status	UDINT
...	
FS4Status	UDINT
USubNumEcho	INT
UAxisIDEcho	INT
UError	INT
UReturn1	UDINT
UReturn2	REAL
UReturn3	REAL

MMFControlBits

Elements of the MMFControlBits Data Structure:

Element	Datatype	Description
CONTROL_CAPTURE	BOOL	Start the capturing of data into the data capture buffer.
BIT1	BOOL	
CONTROL_ACQUIRE	BOOL	Acquire command of controlled axes and make them be linked to the MotionSet. The AXIS_IN_COMMAND MotionStatus bit is set if successful.
BIT3	BOOL	
BIT4	BOOL	
BIT5	BOOL	
BIT6	BOOL	
BIT7	BOOL	
BIT8	BOOL	
BIT9	BOOL	
CONTROL_ENABLE	BOOL	Enable the controlled axes. The DRIVE_ENABLED MotionStatus bit is set if successful.
CONTROL_FOLLOW	BOOL	Turn following on for the FollowerSet or member of a FollowerSet. The AXIS_IS_LINKED MotionStatus bit is set when following is on.
CONTROL_RESUME	BOOL	Resume from a hold. The AXIS_HOLD MotionStatus bit is clear if the resuming starts.
BIT13	BOOL	
BIT14	BOOL	
CONTROL_CLEAR_FAULT	BOOL	Clear MotionFaults. The AXIS_SUMMARY_FAULT MotionStatus bit is clear if successful.
ALLOW_CAPTURE	BOOL	Stop the capturing of data into the data capture buffer.
BIT17	BOOL	
ALLOW_ACQUIRE	BOOL	Release the controlled axes. The AXIS_IN_COMMAND MotionStatus bit is clear when released.
BIT19	BOOL	
BIT20	BOOL	

Element	Datatype	Description
BIT21	BOOL	
BIT22	BOOL	
BIT23	BOOL	
BIT24	BOOL	
BIT25	BOOL	
ALLOW_ENABLE	BOOL	Disable the controlled axes. The DRIVE_DISABLED MotionStatus bit is set when disabled.
ALLOW_FOLLOW	BOOL	Turn following off for FollowerSet or a member of a FollowerSet. The AXIS_IS_LINKED MotionStatus bit is clear when following is off.
ALLOW_RESUME	BOOL	ssue a hold to the controlled axes. The AXIS_HOLD MotionStatus bit is set when the motion profile is holding with 0 commanded speed.
ALLOW_MOVE	BOOL	Issue a halt. The AXIS_HALT MotionStatus bit is set when the halt process starts.
ALLOW_NOT_FASTSTOP	BOOL	Issue a fastStop command to the controlled axes. The AXIS_FASTSTOP MotionStatus bit is set.
ALLOW_NOT_FAULT	BOOL	Cause a user fault to occur. The AXIS_SUMMARY_FAULT MotionStatus bit is set.

MMFStatusBits

Elements of the MMFStatusBits Data Structure:

Element	Type	Description
RAMPING	BOOL	The commanded motion profile is accelerating or decelerating to a different speed.
STEADY	BOOL	The motion profile is commanding a steady speed.
STOPPING	BOOL	The commanded motion profile is decelerating to a stop. STOPPING is set until PROFILE_END is set. STOPPING remains set until IN_POSITION is set.
PROFILE_END	BOOL	The commanded motion profile has completed. The axis may still be settling until IN_POSITION is set or STOPPING is cleared. PROFILE_END is set when the axis is disabled. PROFILE_END is set after a halt is completed.

Element	Type	Description
IN_POSITION	BOOL	The axis position is within the InPositionBand after the commanded motion profile has started STOPPING. The STOPPING bit is cleared as soon as PROFILE_END and IN_POSITION are both set. IN_POSITION is clear when the axis is disabled. IN_POSITION is valid after a halt is completed.
AXIS_HOMING	BOOL	The axis is performing a home function.
AXIS_HOMED	BOOL	The axis home function completed successfully. The axis position reading is referenced off the home position and is valid.
AXIS_NOT_FOLLOWING	BOOL	The drive is ignoring the controller's motion profile while it performs a special operation (for example, homing, fastStop, or EStop). This is not available on all SERCOS drives.
HOLDING	BOOL	The axis is in the process of holding or is holding the motion profile. The axis is either decelerating into the hold or is being commanded to hold position with 0 speed.
RESUMING	BOOL	The axis is in the process of resuming from holding the motion profile. The axis is accelerating back onto the motion profile. RESUMING is cleared when the original motion profile is being commanded.
DRIVE_ENABLED	BOOL	The drive is enabled with motor power. If the AXIS_READY bit is set, the axis may be commanded to move. If DRIVE_ENABLED is clear, the axis is enabling, disabling, or disabled.
DRIVE_DIAG	BOOL	This bit indicates if any SERCOS class 3 diagnostic bit has changed. The SERCOS Standard IDN 0013 retrieves the actual information. This bit is cleared when the class 3 diagnostic is read, but the SERCOS class 3 diagnostic bits themselves are unaffected by the SERCOS Standard IDN 0013.
DRIVE_WARNING	BOOL	This bit indicates if any SERCOS class 2 diagnostic bit has changed. The SERCOS Standard IDN 0012 retrieves the actual information. This bit and the SERCOS class 2 diagnostic bits themselves are cleared (and re-armed) by the SERCOS Standard IDN 0012.

Element	Type	Description
DRIVE_FAULT	BOOL	This bit indicates if any SERCOS class 1 diagnostic bit has changed. The SERCOS Standard IDN 0011 retrieves the actual information. When SERCOS Standard IDN 0011 is called, this bit is not cleared. This bit and the SERCOS class 1 diagnostic bits themselves are cleared by the clearFault function.
DRIVE_DISABLED	BOOL	The drive is disabled and power has been removed from the motor. If this bit is clear, the drive is disabling, enabling, or enabled.
AXIS_SUMMARY_FAULT	BOOL	A servo drive DRIVE_FAULT, SERCOS communications fault (AXIS_COMM_OK clear), ALLOW_NOT_FAULT is clear, or a controller motion profile fault has occurred. Fault information is retrieved by issuing a getMotionFault function. A clearFault function resets the MotionFault information and clears the fault if possible. If the fault is not cleared, the AXIS_SUMMARY_FAULT bit is set again.
AXIS_COMM_OK	BOOL	The drive has successfully powered up and the communications between the motion controller and the drive are active.
AXIS_IS_LINKED	BOOL	The axis is an active part of an axis collection. The axis is responding to commands from a CoordinatedSet or FollowerSet.
AXIS_IN_COMMAND	BOOL	The axis will respond to motion commands.
AXIS_CAPTURE	BOOL	The axis is capturing data into its data capture buffer.
AXIS_AT_TARGET	BOOL	The axis position is within the InPositionBand of the target position of a move command. AXIS_AT_TARGET is set after the commanded motion profile has reached PROFILE_END and the axis position is within the InPositionBand of the target. AXIS_AT_TARGET is cleared when a new move command is issued. Unlike IN_POSITION, AXIS_AT_TARGET is not changed when the axis is disabled, nor is it set after a halt is completed unless the halted position reaches the original motion profile's target position.

Element	Type	Description
AXIS_POS_LIMIT	BOOL	The commanded axis position is at or beyond the positive position limit. AXIS_POS_LIMIT is valid if the axis is DRIVE_ENABLED and AXIS_HOMED. An AXIS_SUMMARY_FAULT occurs when the commanded position first exceeds or equals the limit. A clearFault function with the axis still on the limit clears the fault but leaves the AXIS_POS_LIMIT set until the position is more negative than the limit. Moving the axis back into the limit will then generate another AXIS_SUMMARY_FAULT.
AXIS_NEG_LIMIT	BOOL	The commanded axis position is at or beyond the negative position limit. AXIS_NEG_LIMIT is valid if the axis is DRIVE_ENABLED and AXIS_HOMED. An AXIS_SUMMARY_FAULT occurs when the commanded position first exceeds or equals the limit. A clearFault function with the axis still on the limit clears the fault but leaves the AXIS_NEG_LIMIT set until the position is more positive than the limit. Moving the axis back into the limit will then generate another AXIS_SUMMARY_FAULT.
AXIS_WARNING	BOOL	The motion controller has detected a MotionWarning condition for the axis. Issue a getMotionWarning function and examine the result to determine the warnings.
BIT24	BOOL	Reserved
BIT25	BOOL	Reserved
DRIVE_REALTIME_BIT1	BOOL	
DRIVE_REALTIME_BIT2	BOOL	
AXIS_HOLD	BOOL	The motion axis is holding with a speed of 0 being commanded to the axis. This bit is similar to the HOLDING bit, but HOLDING is set while the axis is decelerating to a hold. AXIS_HOLD only comes on when the axis has stopped due to a hold function being issued. This bit is clear if any of the AXIS_HALT, AXIS_FAST_STOP, AXIS_HOMING, DRIVE_DISABLED, or AXIS_SUMMARY_FAULT bits is set, or the axis is an inactive axis with AXIS_IN_COMMAND clear.

Element	Type	Description
AXIS_HALT	BOOL	The ALLOW_MOVE bit is clear in the MotionControl register. The axis will not accept move commands while this bit is clear. This bit is clear if any of the AXIS_FAST_STOP, AXIS_HOMING, DRIVE_DISABLED, or AXIS_SUMMARY_FAULT bits is set, or the axis is an inactive axis with AXIS_IN_COMMAND clear.
AXIS_FASTSTOP	BOOL	An axis is in the fastStop state. The bit is cleared when the fastStop state is exited as a result of a disableDrive function, axis fault, or enableDrive function. This bit is clear if any of the DRIVE_DISABLED, AXIS_SUMMARY_FAULT bits is set, or the axis is an inactive axis with AXIS_IN_COMMAND clear.
AXIS_READY	BOOL	The axis is ready to respond to a move command. The axis is active with AXIS_IN_COMMAND and DRIVE_ENABLED set, and AXIS_HOMING, AXIS_HOLD, AXIS_HALT, AXIS_FASTSTOP, AXIS_NOT_FOLLOWING, and AXIS_SUMMARY_FAULT are all cleared.

AXCfg

Elements of the AXCfg Data Structure:

Element	Datatype	Description
AxVersion	INT	This is the interface layout version of this data block for this motion axis type.
AxisId	INT	The axis ID for this motion axis object.
ConfigBit	INT	Bit pattern to configure axis features bit.
SercosAddress	INT	Address of the Axis.
Accel	REAL	Default acceleration (dau)
Decel	REAL	Default deceleration (dau)
AccelType	INT	Default AccelType
InPositionBand	REAL	Default inPositionBand (dpu)
EnablePosBand	REAL	Default enablePositionBand (dpu)
MaxRollover	REAL	Positive rolloverLimit (dpu)
MinRollover	REAL	Negative rolloverLimit (dpu)
MaxAccel	REAL	Maximum acceleration (dau)
MaxDecel	REAL	Maximum deceleration (dau)
MaxSpeed	REAL	Maximum speed (dvu)

Element	Datatype	Description
MaxPos	REAL	Positive positionLimit (dpu)
MinPos	REAL	Negative positionLimit (dpu)
GearNumerator	REAL	Numerator of the gear ratio in axis default position units (dpu)
GearDenominator	REAL	Denominator of the gear ratio in drive units. For ImaginaryAxis and RemoteAxis objects, this must be in fbCount units.
AccelUnits	INT	Default acceleration units (AccelUnits) for the axis.
VelUnits	INT	Default velocity units (VelocityUnits) for this axis.
PosUnits	INT	Default position units (PositionUnits) for this axis.
FeedbackReg	UDINT	For RemoteAxis only. If AX_SERCOS_AXIS_REG is 0, then use this register address for reading the remote axis feedback counter.
FeedbackIDN	INT	

CSCfg

Elements of the CSCfg Data Structure:

Element	Datatype	Description
AxVersion	INT	This is the interface layout version of this data block for this motion axis type.
AxisId	INT	The axis ID for this motion axis object.
ConfigBit	INT	Reserved
NumberinSet	INT	This specifies how many axes are members of this CoordinatedSet.
CAxisId	ARRAY [1..8] OF INT	The axis ID of the member n in the set. (n = 1 ...8)
Accel	REAL	Default acceleration (dau)
Decel	REAL	Default deceleration (dau)
AccelType	INT	Default AccelType
MaxAccel	REAL	Maximum acceleration (dau)
MaxDecel	REAL	Maximum deceleration (dau)
MaxSpeed	REAL	Maximum speed (dvu)
AccelUnits	INT	Default acceleration units (AccelUnits) for the axis.
VelUnits	INT	Default velocity units (VelocityUnits) for this axis.

FSCfg

Elements of the FSCfg Data Structure:

Element	Datatype	Description
AxVersion	INT	This is the interface layout version of this data block for this motion axis type.
AxisId	INT	The axis ID for this motion axis object.
ConfigBit	INT	Reserved
NumberinSet	INT	This specifies how many axes are members of this CoordinatedSet.
MasterId	INT	The axis ID of master axis.
F	ARRAY [1..8] OF FSMember	Data of the member n in the set. (n = 1 ...8)

FSMember

Elements of the FSMember Data Structure:

Element	Datatype	Description
AxisId	INT	Axis ID of the member
FollowerMode	INT	The follower mode of the member
CamOrNumerator	REAL	For camming: The camProfile number For gearing: The numerator of the FollowerMode (dspu)
OffsetOrDenom	REAL	For camming: The masterOffset. For gearing: The denominator of the FollowerMode (dmpu)
Trigger	REAL	The trigger position of the master (dmpu)

CPCfg

Elements of the CPCfg Data Structure:

Element	Datatype	Description
ProfileType	INT	CAM profile
INTerpType	INT	interpolation type
NumberOfCoords	INT	Number of coordinates
NumberOfPoINTs	INT	Number of table points
reserved	ARRAY [1..6] OF INT	
MasterOffset	INT	Register offset number for start of first master position register block.
FollowerOffset	INT	Register offset number for the start of the first follower position register block.

Faults and Error Reporting

Introduction

The motion controller accommodates several methods for debugging. The simplest method reports faults and warnings via the data structure `MMFBlock` *(see page 130)*.

Error Registers

The table shows how the data structure `MMFBlock` could be used to report faults and warnings.

`MMFBlock` Element	**Address**	**Value**	**Format**
`FaultAxis`	%MW1105	1	Dec
`FaultCode`	%MW1106	4002	Dec
`WarnAxis`	%MW1107	1	Dec
`WarnCode`	%MW1108	42	Dec

In the case above, the axis reporting the fault is axis number 1 (`FaultAxis` = 1) and the actual fault is a motor overtemperature condition (`FaultCode`=4002). (See Table Faults and Warnings for a complete list.)

The axis reporting the warning is axis number 1 (`WarnAxis`=1) and the actual warning is that the driver is not enabled (`WarnCode`=42). These codes are also reported in the `ERROR` output of the `MMF_*` and `CFG_*` function blocks. (See Table Motion Error Codes for a complete list.)

If several errors or warnings occur, the error code associated with the first error is reported.

To clear the error register, issue a `faultClear( )` function to the SERCOS ring.

Faults and Warnings

Table of faults and warnings reported by `FaultCode`:

Name	**FaultCode**	**Description**
FAULT_OVERLOAD	4000	RMS current fault
FAULT_AMP_OVERTEMP	4001	Drive overtemperature condition
FAULT_MOTOR_OVERTEMP	4002	Drive overtemperature condition
FAULT_FEEDBACK	4005	Resolver or encoder feedback fault
FAULT_COMMUTATION	4006	General drive fault (phase error)
FAULT_OVERCURRENT	4007	Drive short circuit fault
FAULT_UNDERVOLTAGE	4009	Drive voltage fault
FAULT_POS_DEVIATION	4011	Following error fault
FAULT_DRIVE_COMM	4012	Drive has detected communications fault
FAULT_TRAVEL_LIMIT	4013	Hardware end-of-travel fault

Name	FaultCode	Description
FAULT_GENERAL	4015	Homing, digital output or control conflict fault (enable from 2 sources)
FAULT_MASTER_COMM	4016	SERCOS master has detected communication fault
FAULT_WATCHDOG	5001	The watchdog expired and all axes disabled
WARN_BAD_SUBROUTINE	7001	MMFStart does not recognize this subroutine number
WARN_BAD_AXIS_ID	7002	The Axis ID is not valid for this subroutine
WARN_BAD_DATA	7003	Parameter data is out of range
WARN_SUBROUTINE_ABORT	7004	SubNum/SubNumEcho handshake error
WARN_SUBROUTINE_TIMEOUT	7005	Subroutine does not complete in time
WARN_BAD_USER_SUBROUTINE	7010	MMFStart does not recognize this subroutine number
WARN_SAMPLE_USER_SUBROUTINE	7777	A call was made to the SAMPLE_USER_SUBROUTINE
WARN_BAD_DATA	7003	Parameter data is out of range
MMF_ABORT_USUB	9004	User Subroutine abort error code
MMF_USUB_TIMEOUT	9005	User Subroutine Timeout error code

Motion Error Codes

Table of faults and warnings reported by `WarnCode` and `ERROR` output:

Error Code	Name	Description
1	RANGE_ERROR	Attempt to assign value out of range
2	UNITS_MISMATCH	Attempt to assign incompatible units
3	UNSUPPORTED_UNIT	Unit not supported or unknown
4	DOWNLOAD_ERROR	Bug or drive fault during download
5	UPLOAD_ERROR	Bug or drive fault during upload
6	NULL_OBJECT	Unexpected null pointer to object
7	SET_UNITS_ERROR	Failed to set units in drive
8	UNITS_NOT_SET	Units not set in SASSDriveInterface
9	STRING_TOO_BIG	String too big to fit into MotionString
10	INVALID_INDEX	Invalid index into a collection
11	INVALID_VALUE	Invalid value in command
12	INVALID_ENUM_VALUE	Invalid value for an enum
13	INVALID_TOKEN	Invalid token in input
14	INVALID_FEEDBACK_CHANNEL	Invalid feedback channel for command
15	INVALID_FEEDBACK_DEVICE	Invalid feedback device for command
16	INVALID_FEEDBACK_CLOCKING_RATE	Invalid feedback clocking rate

Error Code	Name	Description
17	INVALID_FEEDBACK_POWER_SRC	Invalid feedback power source
18	INVALID_FEEDBACK_RES	Invalid feedback resolution
19	INVALID_HOLDING_REG_ADDR	Invalid holding register address
20	HOLDING_REGISTERS_NOT_CONFIGURED	Holding register database not configured
21	NO_HOLDING_REGS	Holding register database empty
22	REG_BLOCK_TOO_BIG	Holding register block too big
23	REG_BLOCK_NOT_FIT	Holding register block doesn't fit into database
24	CANNOT_GRANT_ACCESS	Cannot grant access to holding register block
25	CANNOT_RELEASE_ACCESS	Cannot release access to holding register block
26	FILE_OPEN_ERROR	File open failed
27	FILE_WRITE_ERROR	File write failed
28	FILE_READ_ERROR	File read failed
29	FILE_CLOSE_ERROR	File close failed
30	FILE_SEEK_ERROR	File seek failed
31	SYNTAX_ERROR	Malformed input
32	CLEAR_FAULT_ERROR	Clear fault function failed to clear faults
33	MISSING_TAG	Missing tag in tags.cfg
34	NO_AXIS_AVAILABLE	No axis object available
35	TOO_MANY_AXES	Too many axes in rc-/rv-axis
36	DUPLICATE_AXES	Duplicate axes in rc-/rv-axis
37	INVALID_AXIS	Missing or invalid axis
38	NO_SUCH_AXIS	Axis object or config file not found
39	WRONG_NUMBER_OF_COORDS	Value has different number of coordinates from axis
40	DEVICE_NOT_IN_CONTROL	Motion axis not active
41	MOVE_FAULT_ERROR	A move fault has occurred front end)
42	AXIS_NOT_ENABLED	Drive is not enabled.
43	COMMAND_TIMED_OUT	Command timed out
44	INVALID_SERCOS_RING	SERCOS ring not A or B
45	RENAME_ERROR	Axis renamed failed
46	CANNOT_PERFORM_WITH_THIS_CONFIG	Cannot perform command as currently configured
47	TYPE_MISMATCH	Object is wrong type
48	DRIVE_MUST_BE_DISABLED	Drive must be disabled to perform command

Error Code	Name	Description
49	DRIVE_MUST_BE_ENABLED	Drive must be enabled to perform command
50	CMD_NOT_ALLOWED	Command not allowed at this time
51	DRIVE_FAULT_ERROR	Command cannot be completed due to drive fault
1000	TARGET_NOT_RESP	Target not responding
1001	COMM_GARBLED	Garbled communications
1002	SERCOS_EXCEPTION	SERCOS error
1003	SASS_NO_OPCODE_ECHO	No opcode echo from drive
1004	SERCOS_NOT_READY	SERCOS loop not ready
1005	SERCOS_DRIVER_ERROR	SERCOS error
1006	SERCOS_CYCLIC_READ_ERROR	SERCOS read failed (cyclic channel)
2000	UNKNOWN_ERROR	Unclassified error
2001	BOOCH148_ERROR	Error in Booch Components v1.4.8
2002	BOOCH23_ERROR	Error in Booch Components v2.3
2003	SYSTEM_DEBUG_ERROR	System fault used in debugging
2004	SASS_CMD_NOT_ALLOWED_ERROR	Drive got command not allowed fault
2005	SASS_ILLEGAL_CMD_ERROR	Drive got illegal command fault
2006	SASS_PROGRAMMING_ERROR	Drive got programming error fault
2007	UNSUPPORTED_COMMAND	Invalid command ID
2008	SET_LINK_ERROR	Failed to link object to drive interface
2009	SEMAPHORE_CREATE_ERROR	Semaphore create failed
2010	SEMAPHORE_DELETE_ERROR	Semaphore delete failed
2011	SEMAPHORE_LOCK_ERROR	Semaphore lock failed
2012	SEMAPHORE_UNLOCK_ERROR	Semaphore unlock failed
2013	SEMAPHORE_INQ_ERROR	Semaphore inquiry failed
2014	PROXY_NOT_CONNECTED	Proxy not connected
2015	NOT_IMPLEMENTED	Command not implemented yet
2016	SCX_UNIQUE_ERROR	scx_unique failed
2017	QUEUE_CREATE_ERROR	Queue create failed
2018	QUEUE_FULL	Queue is full
2019	INVALID_QUEUE_ID	Invalid queue id
2020	UNKNOWN_QUEUE_STATUS	Unknown queue status
2021	QUEUE_INQ_ERROR	Queue inquiry failed
2022	EVENT_GRP_CREATE_ERROR	Event group create failed
2023	EVENT_GRP_PEND_ERROR	Event group pend error

Error Code	Name	Description
2024	EVENT_GRP_CLEAR_ERROR	Event group clear error
2025	EVENT_FLAGS_CREATE_ERROR	Event flags create failed
2026	OBJECT_NOT_FOUND	Object lookup failed
2027	OBJECT_MANAGER_MISSING	Object manager not found
2028	INVALID_PLANNER_STATE	Invalid motion planner state
2029	OUT_OF_MEMORY	Memory allocation field
2030	GET_TASK_ID_FAILED	Error occurred upon attempting to get task Id from the operating system
2031	TOO_MANY_ERROR_HANDLERS	Attempt to install too many error handlers
2032	THREAD_CREATE_ERROR	Thread create failed
2033	THREAD_DELETE_ERROR	Thread destructor failed
2034	THREAD_CONFIG_ERROR	Problem with configuring a thread has occurred
2035	TASK_SUSPEND_ERROR	Error occurred while attempting to suspend a thread
2036	TASK_RESUME_ERROR	Error occurred while attempting to resume a thread
2037	OBJECT_CREATE_ERROR	Error occurred while creating object

Part V
Lexium Drives

Overview

This part describes the communication functions with Lexium drives.

What Is in This Part?

This part contains the following chapters:

Chapter 30
LXM_RESTORE: Restoring Parameters and Lexium Tasks

Subject of this Chapter

This chapter describes the function used to restore Lexium parameters and motion tasks, `LXM_RESTORE`.

What Is in This Chapter?

This chapter contains the following topics:

Description

Description of the Function

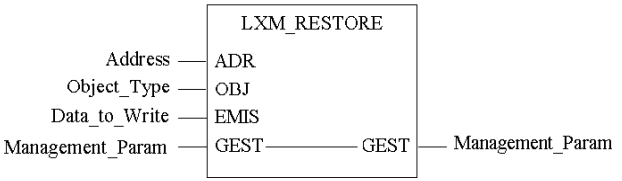
The LXM_RESTORE function enables you to restore the parameters or motion tasks for Lexium drives on a Fipio bus.

For additional details on Lexium drives and faulty device management *(see Lexium 15 using Unity Pro, Communication by Fipio, Setup Manual)*, please refer to the information on drives.

The additional parameters EN and ENO can be configured.

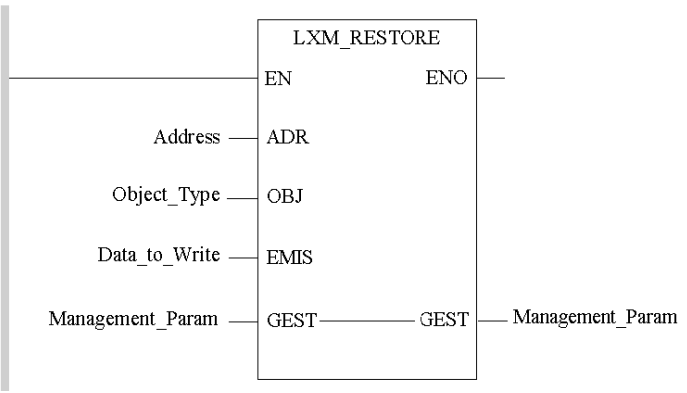
FBD Representation

Representation:



LD Representation

Representation:



IL Representation

Representation:

```
LD Address
LXM_RESTORE Object_Type, Data_to_Write, Management_Param
```

ST Representation

Representation:

```
LXM_RESTORE(Address, Object_Type, Data_to_Write, Management_Param);
```

Description of Parameters

The following table describes the input parameters:

Parameter	Type	Comment
Address	ARRAY [0.. 5] OF INT	Address of the destination entity of the exchange. Generic example: ADDR('\b.e\SYS') b: bus number e: Fipio agent number of the Lexium drive on the bus
Object_Type	STRING	Type of object to be restored: ● 'P' = parameters, ● 'MT' = tasks (Motion tasks).
Data_to_Write	ARRAY [n... m] OF INT	Word table containing the value of the objects to be restored. This table was created previously using the LXM_SAVE function when drive parameters or motion tasks for a Lexium drive were being saved.

The following table describes the input/output parameters:

Parameter	Type	Comment
Management_Param	ARRAY [0.. 13] OF INT	Exchange management table *(see page 152)*

Description of Management Information

At a Glance

Management information is used to control the correct operation of the LXM_RESTORE function, as explained below.

Management Table for the LXM_RESTORE Function

The following table describes the management information contained in the Management_Param I/O parameter.

Table elements	Most significant byte	Least significant byte
Management_Param[0]	Exchange number.	-
Management_Param[1]	Operation report.	Communication report.
Management_Param[2]	Timeout: if the exchange is not completed in a time under the Timeout value, the function is cancelled (the exchange is stopped). The Timeout value is a multiple of 100ms, for example: 20 means 20x100ms, or 2 seconds.	
Management_Param[3]	Length: number of bytes sent to the Lexium controller. This value is automatically written by the system after the execution of a LXM_RESTORE function.	
Management_Param[4]	-	Activity bit.
The words Management_Param[5] to Management_Param[13] are reserved.		

Description of Reports

The following table shows the main report descriptions depending on the values returned.

Description	Operation report value	Communication report value
The address format is incorrect.	16#00	16#03
The type of object is different to ‘P’ or ‘MT’.	16#00	16#06
The management parameters are less than 14 words long.	16#00	16#05
The frame received from the Fipio card does not contain any data.	16#03	16#00
The frame received from the Fipio card is of an incorrect length.		

Description	Operation report value	Communication report value
The frame received from the Fipio card contains the response code FD. (1)	16#01	16#00
The length of the word zone where the data is stored is insufficient. (2)	16#00	16#0A
The checksum of the word zone where the data is stored is incorrect.	16#30	16#00
The type of Lexium on the Fipio bus is different from that whose parameters have been saved.	16#31	16#00
Incorrect response from Lexium.	16#32	16#00
Memory capacity of Fipio card on Lexium exceeded.	16#33	16#00
Incorrect type of memory zone.	16#34	16#00
Key:		
(1)	For example, when another request is being processed.	
(2)	In this case, the minimum number of bytes required to restore the data is specified in the word %MWg+3.	

Chapter 31
LXM_SAVE: Saving Parameters and Lexium Tasks

Subject of this Chapter

This chapter describes the function used to save Lexium parameters and motion tasks, `LXM_SAVE`.

What Is in This Chapter?

This chapter contains the following topics:

Description

Description of the Function

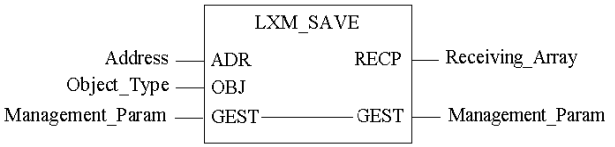
The LXM_SAVE function enables you to save the parameters or motion tasks for Lexium drives on a Fipio bus.

For additional details on Lexium drives and faulty device management *(see Lexium 15 using Unity Pro, Communication by Fipio, Setup Manual)*, please refer to the information on drives.

The additional parameters EN and ENO can be configured.

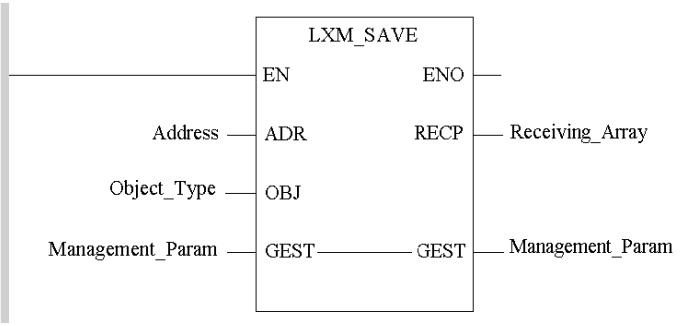
FBD Representation

Representation:



LD Representation

Representation:



IL Representation

Representation:

```
LD Address
LXM_SAVE Object_Type, Management_Param, Receiving_Array
```

ST Representation

Representation:

```
LXM_SAVE(Address, Object_Type, Management_Param, Receiving_Array);
```

Description of Parameters

The following table describes the input parameters:

Parameter	Type	Comment
Address	ARRAY [0.. 5] OF INT	Address of the destination entity of the exchange. Generic example: ADDR('\b.e\SYS') b: bus number e: Fipio agent number of the Lexium drive on the bus
Object_Type	STRING	Type of object to be saved: • 'P' = parameters, • 'MT' = tasks (Motion tasks).

The following table describes the input/output parameters:

Parameter	Type	Comment
Management_Param	ARRAY [0.. 13] OF INT	Exchange management table *(see page 158)*

The following table describes the output parameters:

Parameter	Type	Comment
Receiving_Array	ARRAY [n... m] OF INT	Word table containing the value of the objects saved. When the LXM_RESTORE function is used, this table will serve to reload (restore) the parameters or motion tasks in the destination Lexium drive.

Description of Management Information

At a Glance

Management information is used to control the correct operation of the `LXM_SAVE` function, as explained below.

Management Table for the LXM_SAVE Function

The following table describes the management information contained in the `Management_Param` I/O parameter.

Table elements	Most significant byte	Least significant byte
`Management_Param[0]`	Exchange number.	-
`Management_Param[1]`	Operation report.	Communication report.
`Management_Param[2]`	Timeout: if the exchange is not completed in a time under the Timeout value, the function is cancelled (the exchange is stopped). The Timeout value is a multiple of 100ms, for example: 20 means 20x100ms, or 2 seconds.	
`Management_Param[3]`	Length: number of bytes received in the reception table. This length shows the exact size of the data received from the Lexium controller.	
`Management_Param[4]`	-	Activity bit.
The words `Management_Param[5]` to `Management_Param[13]` are reserved.		

Description of Reports

The following table shows the main report descriptions depending on the values returned.

Description	Operation report value	Communication report value
The address format is incorrect.	16#00	16#03
The type of object is different from `'P'` or `'MT'`.	16#00	16#06
The management parameters are less than 14 words long.	16#00	16#05
The frame received from the Fipio card does not contain any data.	16#03	16#00
The frame received from the Fipio card is of an incorrect length.		

Description	Operation report value	Communication report value
The frame received from the Fipio card contains the response code FD. (1)	16#01	16#00
The length of the word zone is insufficient to save data. (2)	16#00	16#09
Incorrect response from Lexium.	16#32	16#00
Memory capacity of Fipio card on Lexium exceeded.	16#33	16#00
Key:		
(1)	For example, when another request is being processed.	
(2)	In this case, the minimum number of bytes required to save the data is specified in the word %MWg+3.	

Appendices

Appendix A
EFB Error Codes and Values

Introduction

The following tables show the error codes and error values created for the EFBs of the Motion Library.

What Is in This Chapter?

This chapter contains the following topics:

Tables of Error Codes for the Motion Library

Introduction

The following tables show the error codes and error values created for the EFBs of the Motion Library.

MMF Start

Table of error codes and errors values created for EFBs of the `MMF Start` family.

EFB name	Error code	ENO state in case of error	Error value in Dec	Error value in Hex	Error description
CFG_CP_F	BAD_REVISION	F	-30200	16#8A08	defined as E_EFB_USER_ERROR_1
CFG_CP_F	MMF_BAD_4X	T	9010	16#2332	-
CFG_CP_F	MMF_ABORT_SUB	T	7004	16#1B5C	SubNum/SubNumEcho handshake error
CFG_CP_V	BAD_REVISION	F	-30200	16#8A08	defined as E_EFB_USER_ERROR_1
CFG_CP_V	MMF_BAD_4X	T	9010	16#2332	-
CFG_CP_V	MMF_ABORT_SUB	T	7004	16#1B5C	SubNum/SubNumEcho handshake error
CFG_CS	BAD_REVISION	F	-30200	16#8A08	defined as E_EFB_USER_ERROR_1
CFG_CS	MMF_ABORT_SUB	T	7004	16#1B5C	SubNum/SubNumEcho handshake error
CFG_FS	BAD_REVISION	F	-30200	16#8A08	defined as E_EFB_USER_ERROR_1
CFG_FS	MMF_ABORT_SUB	T	7004	16#1B5C	SubNum/SubNumEcho handshake error
CFG_IA	BAD_REVISION	F	-30200	16#8A08	defined as E_EFB_USER_ERROR_1
CFG_IA	MMF_ABORT_SUB	T	7004	16#1B5C	SubNum/SubNumEcho handshake error
CFG_RA	BAD_REVISION	F	-30200	16#8A08	defined as E_EFB_USER_ERROR_1
CFG_RA	MMF_ABORT_SUB	T	7004	16#1B5C	SubNum/SubNumEcho handshake error
CFG_SA	BAD_REVISION	F	-30200	16#8A08	defined as E_EFB_USER_ERROR_1
CFG_SA	MMF_ABORT_SUB	T	7004	16#1B5C	SubNum/SubNumEcho handshake error
DRV_DNLD	BAD_REVISION	F	-30200	16#8A08	defined as E_EFB_USER_ERROR_1
DRV_DNLD	MMF_ABORT_SUB	T	7004	16#1B5C	SubNum/SubNumEcho handshake error
DRV_UPLD	BAD_REVISION	F	-30200	16#8A08	defined as E_EFB_USER_ERROR_1
DRV_UPLD	MMF_ABORT_SUB	T	7004	16#1B5C	SubNum/SubNumEcho handshake error
IDN_CHK	BAD_REVISION	F	-30200	16#8A08	defined as E_EFB_USER_ERROR_1
IDN_CHK	MMF_ABORT_SUB	T	7004	16#1B5C	SubNum/SubNumEcho handshake error
IDN_XFER	BAD_REVISION	F	-30200	16#8A08	defined as E_EFB_USER_ERROR_1

EFB name	Error code	ENO state in case of error	Error value in Dec	Error value in Hex	Error description
IDN_XFER	MMF_ABORT_SUB	T	7004	16#1B5C	SubNum/SubNumEcho handshake error
MMF_BITS	BAD_REVISION	F	-30200	16#8A08	defined as E_EFB_USER_ERROR_1
MMF_ESUB	BAD_REVISION	F	-30200	16#8A08	defined as E_EFB_USER_ERROR_1
MMF_ESUB	MMF_ABORT_SUB	T	7004	16#1B5C	SubNum/SubNumEcho handshake error
MMF_IDNX	BAD_REVISION	F	-30200	16#8A08	defined as E_EFB_USER_ERROR_1
MMF_IDNX	MMF_ABORT_SUB	T	7004	16#1B5C	SubNum/SubNumEcho handshake error
MMF_JOG	BAD_REVISION	F	-30200	16#8A08	defined as E_EFB_USER_ERROR_1
MMF_JOG	MMF_ABORT_SUB	T	7004	16#1B5C	SubNum/SubNumEcho handshake error
MMF_JOG	MMF_SUB_TIMEOUT	T	7005	16#1B5D	Subroutine does not complete in time
MMF_MOVE	BAD_REVISION	F	-30200	16#8A08	defined as E_EFB_USER_ERROR_1
MMF_MOVE	MMF_ABORT_SUB	T	7004	16#1B5C	SubNum/SubNumEcho handshake error
MMF_RST	BAD_REVISION	F	-30200	16#8A08	defined as E_EFB_USER_ERROR_1
MMF_SUB	BAD_REVISION	F	-30200	16#8A08	defined as E_EFB_USER_ERROR_1
MMF_SUB	MMF_ABORT_SUB	T	7004	16#1B5C	SubNum/SubNumEcho handshake error
MMF_USUB	BAD_REVISION	F	-30200	16#8A08	defined as E_EFB_USER_ERROR_1
MMF_USUB	MMF_ABORT_SUB	T	7004	16#1B5C	SubNum/SubNumEcho handshake error

NOTE: For details about MMF error codes and error values, please refer to the Faults and Error Reporting *(see page 142)* description in the Motion Library.

Common Floating Point Errors

Introduction

The following table shows the common error codes and error values created for floating point errors. These error information are displayed in the Diagnostic Viewer and the error code values are written in %SW125 *(see Unity Pro, System Bits and Words, Reference Manual)*.

Common Floating Point Errors

Table of common floating point errors

Error codes	Error value in Dec	Error value in Hex	Error description
FP_ERROR	-30150	16#8A3A	Base value (not apearing as an error value)
E_FP_STATUS_FAILED_IE	-30151	16#8A39	Illegal floating point operation
E_FP_STATUS_FAILED_DE	-30152	16#8A38	Operand is denormalized - not a valid REAL number
E_FP_STATUS_FAILED_ZE	-30154	16#8A36	Illegal divide by zero
E_FP_STATUS_FAILED_ZE_IE	-30155	16#8A35	Illegal floating point operation / Divide by zero
E_FP_STATUS_FAILED_OE	-30158	16#8A32	Floating point overflow
E_FP_STATUS_FAILED_OE_IE	-30159	16#8A31	Illegal floating point operation / Overflow
E_FP_STATUS_FAILED_OE_ZE	-30162	16#8A2E	Floating point overflow / Divide by zero
E_FP_STATUS_FAILED_OE_ZE_IE	-30163	16#8A2D	Illegal floating point operation / Overflow / Divide by zero
E_FP_NOT_COMPARABLE	-30166	16#8A2A	Internal error

Glossary

A

ARRAY

An ARRAY is a table containing elements of a single type.

The syntax is as follows: ARRAY [<limits>] OF <Type>

Example:

ARRAY [1..2] OF BOOL is a one-dimensional table with two elements of type BOOL.

ARRAY [1..10, 1..20] OF INT is a two-dimensional table with 10x20 elements of type INT.

B

BOOL

BOOL is the abbreviation for the Boolean type. This is the basic data type in computing. A BOOL variable can have either of the following two values: 0 (FALSE) or 1 (TRUE).

A bit extracted from a word is of type BOOL, for example: %MW10.4.

D

DINT

DINT is the abbreviation of Double INTeger (encoded in 32 bits).

The upper/lower limits are as follows: -(2 to the power of 31) to (2 to the power of 31) - 1.

Example:

-2147483648, 2147483647, 16#FFFFFFFF.

E

EN

EN stands for **EN**able; it is an optional block input. When the EN input is enabled, an ENO output is set automatically.

If EN = 0, the block is not enabled; its internal program is not executed, and ENO is set to 0.

If EN = 1, the block's internal program is run and ENO is set to 1. If an error occurs, ENO is set to 0.

If the EN input is not connected, it is set automatically to 1.

ENO

ENO stands for **E**rror **NO**tification; this is the output associated with the optional input EN.

If ENO is set to 0 (because EN = 0 or in case of an execution error):

- the status of the function block outputs remains the same as it was during the previous scanning cycle that executed correctly;
- the output(s) of the function, as well as the procedures, are set to "0".

I

INT

INT is the abbreviation of single INTeger (encoded in 16 bits).

The upper/lower limits are as follows: -(2 to the power of 15) to (2 to the power of 15) - 1.

Example:

-32768, 32767, 2#1111110001001001, 16#9FA4.

IODDT

IODDT is the abbreviation of Input/Output Derived Data Type.

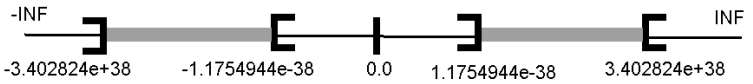
The term IODDT indicates a structured data type representing a module or a channel of a PLC module. Each expert module has its own IODDTs.

R

REAL

The REAL type is encoded in a 32 bit format.

The possible value ranges are shown in the figure below:



When a result is:

- between -1,175494e-38 and 1,175494e-38, it is considered to be a DEN;
- less than -3.402824e+38, the symbol -INF (for - infinity) is displayed;
- greater than +3.402824e+38, the symbol INF (for + infinity) is displayed;
- undefined (square root of a negative number), the symbolNAN is displayed.

NOTE: The IEC 559 standard defines two classes of NAN: the silent NAN (QNAN) and the signaling NAN (SNAN). A QNAN is a NAN with a most significant fraction bit while an SNAN is a NAN without a most significant fraction bit (bit number 22). QNANs can be propagated via most arithmetic operations without throwing an exception. As for SNANs, they generally indicate n invalid operation when they are used as operands in arithmetic operations (see %SW17 and %S18).

NOTE: When a DEN (non-standardized number) is used as an operand, the result is not significant.

S

STRING

A STRING variable is a series of ASCII characters. The maximum length of a string is 65,534 characters.

U

UDINT

UDINT is the abbreviation of Unsigned Double INTeger (encoded in 32 bits). The upper/lower limits are as follows: 0 to (2 to the power of 32) - 1.

Example:

0, 4294967295, 2#11111111111111111111111111111111, 8#37777777777, 16#FFFFFFFF.

UINT

UINT is the abbreviation of the Unsigned INTeger format (encoded in 16 bits). The upper/lower limits are as follows: 0 to (2 to the power of 16) - 1.

Example:

0, 65535, 2#1111111111111111, 8#177777, 16#FFFF.

Index

S

X