Advantages of Using Data Type DB_ANY with Motion Control Applications

TIA Portal V15

https://support.industry.siemens.com/cs/ww/en/view/109750880

Siemens Industry Online Support

This entry is from the Siemens Industry Online Support. The general terms of use (http://www.siemens.com/terms_of_use) apply.

Security Information

Siemens provides products and solutions with industrial security functions that support the secure operation of plants, systems, machines and networks.
In order to protect plants, systems, machines and networks against cyber threats, it is necessary to implement – and continuously maintain – a holistic, state-of-the-art industrial security concept. Siemens' products and solutions only form one element of such a concept.
The customer is responsible to prevent unauthorized access to its plants, systems, machines and networks. Systems, machines and components should only be connected to the enterprise network or the internet if and to the extent necessary and with appropriate security measures (e.g. use of firewalls and network segmentation) in place.
Additionally, Siemens' guidance on appropriate security measures should be taken into account. For more information about industrial security, please visit http://www.siemens.com/industrialsecurity.

Siemens' products and solutions undergo continuous development to make them more secure. Siemens strongly recommends to apply product updates as soon as available and to always use the latest product versions. Use of product versions that are no longer supported, and failure to apply latest updates may increase the customer's exposure to cyber threats.
To stay informed about product updates, subscribe to the Siemens Industrial Security RSS Feed under http://www.siemens.com/industrialsecurity.

Contents

1 Using the Data Type DB_ANY

1.1 Arrangement of the Technology Objects in an Array

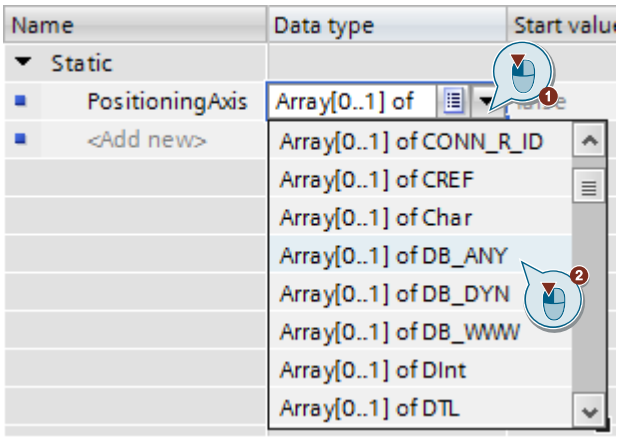
With Technology version V3.0 and higher you can specify the reference to a technology object also via the data type DB_ANY. An "ARRAY of DB_ANY" can represent a list of axes, for example. In this way, technology objects can be processed more flexibly in the program.

The unique assignment of the technology objects to elements of the array must be done once in the user program during startup of the CPU in OB 100, for example. For this you need the following program parts in the user program:

1. Create a data block.

2. In this data block you create an array of the data type DB_ANY in which the technology objects of the project are to be stored. You can define the limits of the array to suit your requirements.

Figure 1-1: Creating the array



3. Create an array element for each technology object. Here you can sort according to the type or use of the technology objects.

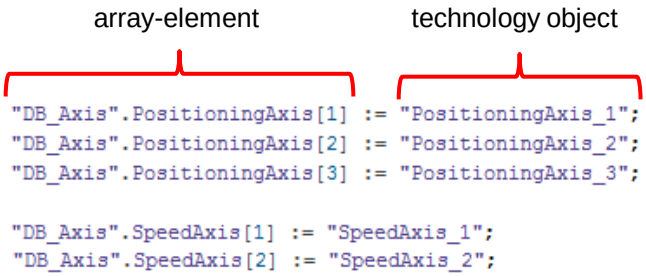
Figure 1-2: Structuring the array

DB_Axis

	Name	Data type	Start value
1	Static		
2	PositioningAxis	Array[1..3] of DB_ANY	
3	PositioningAxis[1]	DB_ANY	0
4	PositioningAxis[2]	DB_ANY	0
5	PositioningAxis[3]	DB_ANY	0
6	SpeedAxis	Array[1..2] of DB_ANY	
7	SpeedAxis[1]	DB_ANY	0
8	SpeedAxis[2]	DB_ANY	0
9	SynchronousAxis	Array[1..5] of DB_ANY	
10	SynchronousAxis[1]	DB_ANY	0
11	SynchronousAxis[2]	DB_ANY	0
12	SynchronousAxis[3]	DB_ANY	0

4. During the startup of the CPU (in OB 100, for example) you assign the corresponding technology objects of the project to the separate array elements.

Figure 1-3: Assigning the technology objects



5. Now, further on in the program you can access the individual technology objects via the index of the array. In this way in the user program you can also execute a loop operation (FOR instruction) to the technology objects.

Figure 1-4: Loop operation to the technology objects

```
FOR #index := 1 TO 3 DO

    // call MC_Power for all positioning axis
    "CallMcPower"(inputAxis := "DB_Axis".PositioningAxis[#index],
                  instMcPower := #instMcPower[#index]);

    //call MC_Reset for all positioning axis
    "CallMcReset"(inputAxis := "DB_Axis".PositioningAxis[#index],
                  instMcReset := #instMcReset[#index]);

    //call MC_MoveAbsolute for all positioning axis
    #instMcMoveAbsolute[#index](Axis := "DB_Axis".PositioningAxis[#index]);

END_FOR;
```

By using a loop operation you can significantly reduce the size of the user program for addressing the technology objects.

When creating the user program you must take into account that the data type of each technology object is defined by the function of the object. This means that, for example, a technology object "PositioningAxis" retains the data type "TO_PositioningAxis" even when using DB_ANY.

Note The data types of the technology objects are given in the function manual "S7-1500(T) Motion Control V4.0 in the TIA Portal V15".

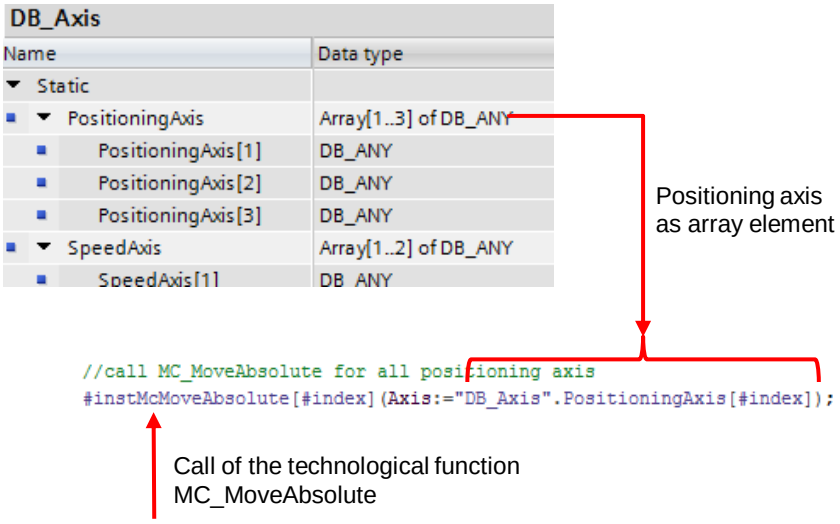
https://support.industry.siemens.com/cs/ww/en/view/109749263

In the following chapters we show how to create the program with identical and different data types taking the example of a positioning axis.

1.2 Assignment with Identical Data Types

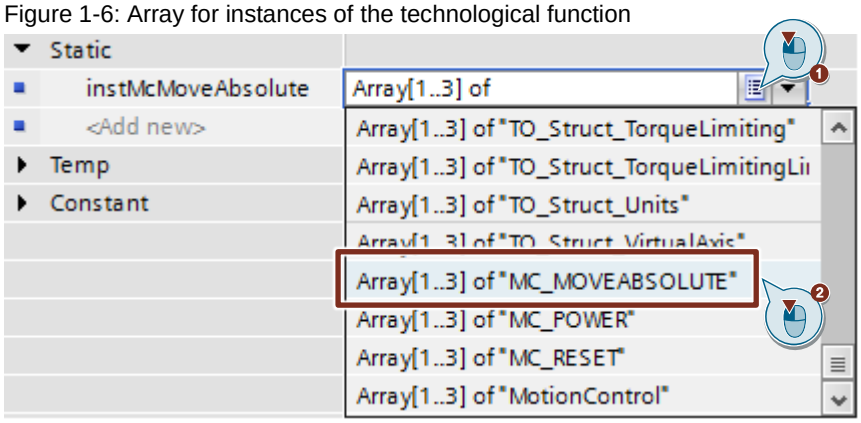
The assignment of the array elements to parameters of a function block or a technology object may only be done with identical data types. For this reason, in the above example of the technological function "MC_MoveAbsolute" you can assign a technology object directly from the "PositioningAxis" array. Only positioning axes are assigned to the "PositioningAxis" array and the "Axis" input of the technological function "MC_MoveAbsolute" is of the data type "TO_PositioningAxis".

Figure 1-5: Using an array element with identical data types



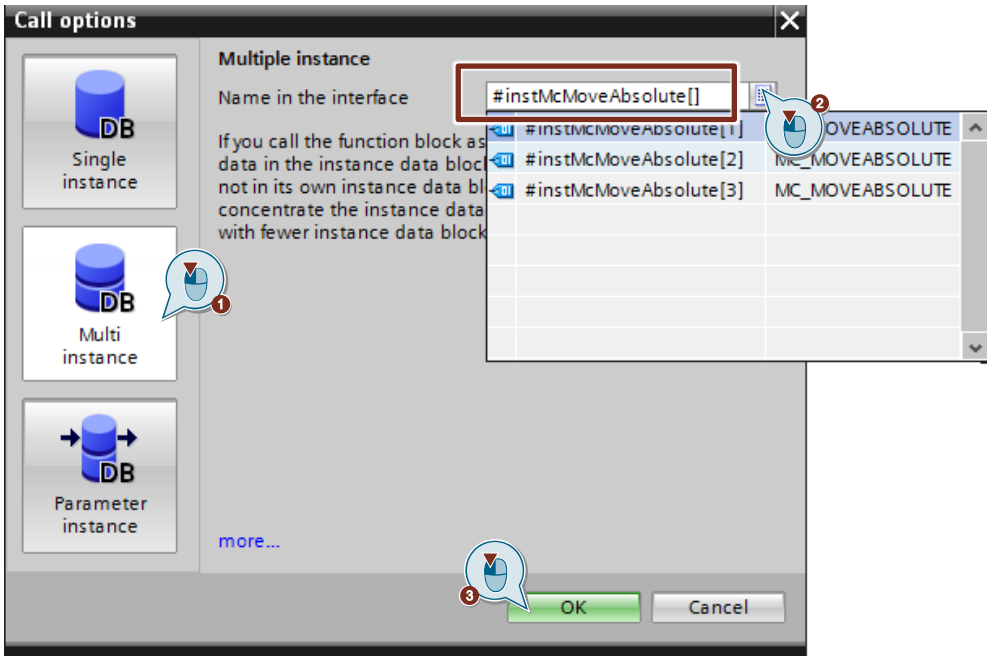
In order to be able to operate the user program with a loop operation, you also have to structure the instances of the technological function in an array in addition to the technology objects. For this you create an array in the static storage area of the function block. The data type of the array corresponds to the technological function to be called.

Figure 1-6: Array for instances of the technological function



Then you use the multi-instance option when calling the technological functions in the user program. You can assign the individual elements of the instance array.

Figure 1-7: Creating the multi-instance

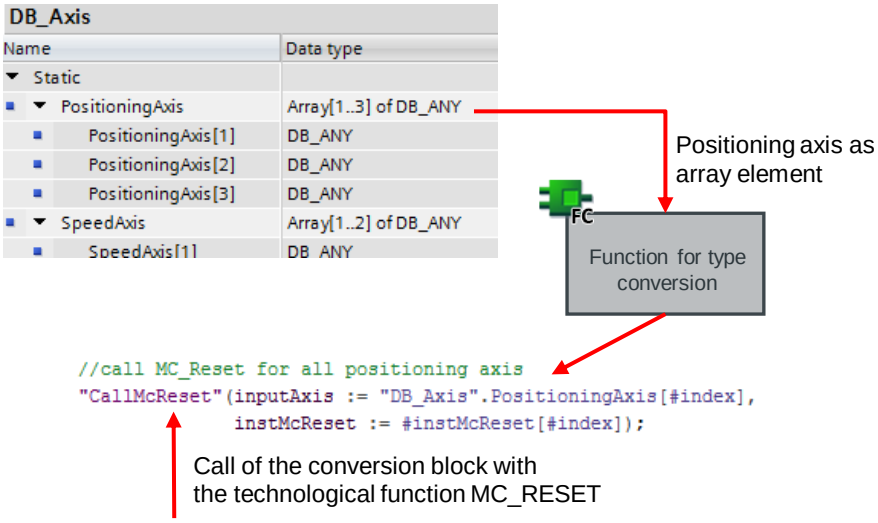


1.3 Assignment with Different Data Types

If you want to assign the technology object "PositioningAxis" as an array element to an input of a technological function that does not correspond to the data type of the technology object ("TO_PositioningAxis"), then you have to convert the data type. For example, this is how a technology object is transferred as data type "TO_Object" to the technological function "MC_RESET".

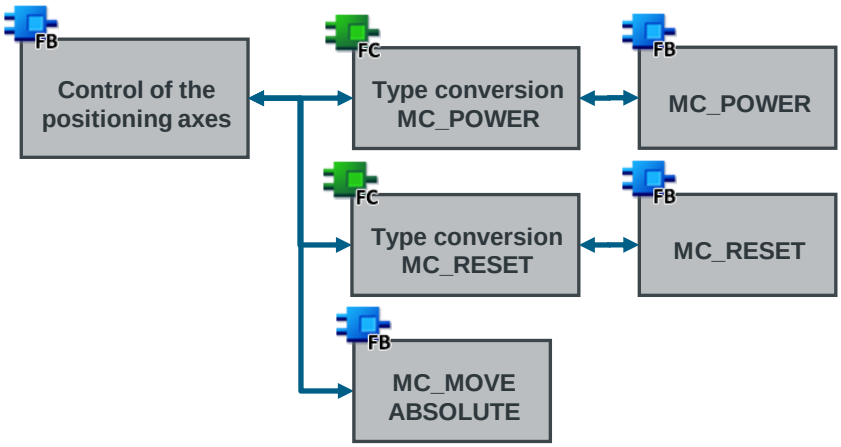
In order to be able to assign a positioning axis as array element of the data type "TO_PositioningAxis" to the technological function "MC_RESET" you have to call the technological function in a separate block for type conversion.

Figure 1-8: Using an array element with different data types



In using the necessary functions for data type conversion the sample program (see Figure 1-4) has the following block structure:

Figure 1-9: Block structure

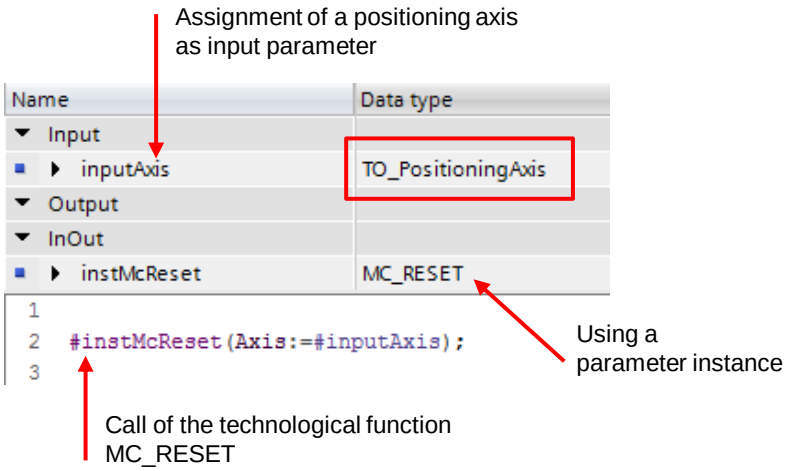


Structure of the conversion function

The blocks for converting data types require an input parameter (Input) for the technology object with the corresponding data type.

If, for instance, you want to assign a positioning axis as array element to the technological function "MC_RESET", the input of the conversion function has to correspond to the data type "TO_PositioningAxis".

Figure 1-10: Content of the conversion function



In the conversion function you can now assign the positioning axis to the technological function. For calling the technological function it is recommended to use a parameter instance.

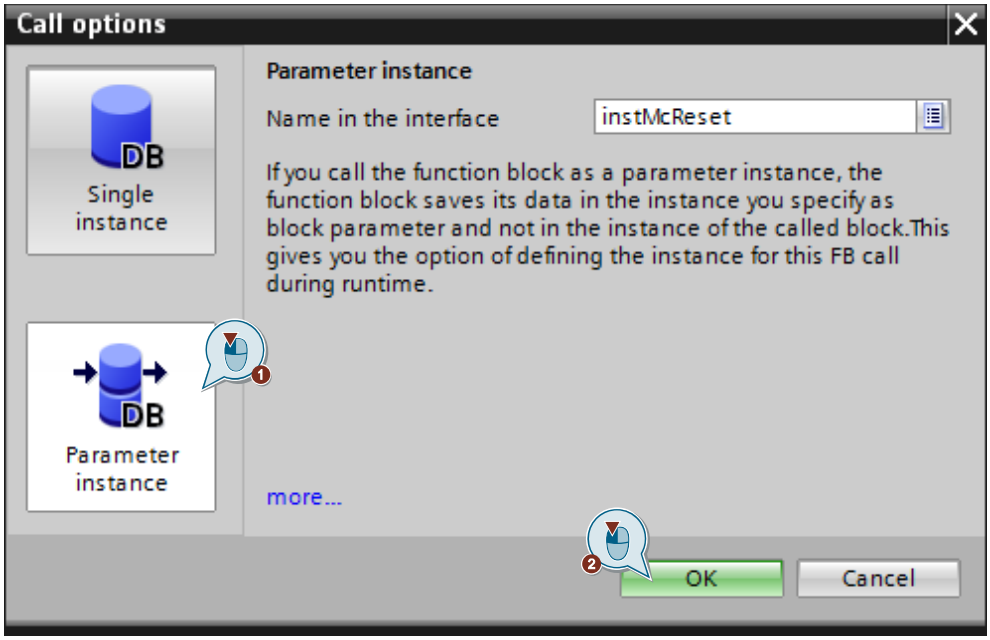
Note For each technology object used (positioning axis, speed axis, etc.) you have to create a separate conversion block with the corresponding data type as input parameter.

Using the parameter instance

When using the type conversion in a separate block it is recommended to configure the instances of the implemented Motion Control blocks as a parameter instance.

In this way you can store the instances of the Motion Control blocks as array in the processing function block.

Figure 1-11: Using the parameter instance



By using the parameter instance you can store the instances of the Motion Control blocks in the form of an array. This is necessary for using a loop operation in the user program.

Figure 1-12: Instances of the Motion Control blocks as array