BAUMÜLLER

be in motion be in motion

BM4-O-CAN-06

CANsync Option Module for b maXX PLC
Application Manual

E	5.02066.01a

Title	Application Manual
Product	CANsync Option Module for b maXX PLC BM4-O-CAN-06
Last Revision	May 28, 2003

Binding nature

This Application Manual is a part of the unit/machine. This Application Manual must always be available to operators and be legible. If the unit/machine is sold, the owner must pass on this Application Manual together with the unit/machine.
After selling the unit/machine you must pass on this original and all the copies that you made to the purchaser. After disposing of the machine in any way, you must destroy this original and all the copies that you made.

When you pass on this Application Manual, all earlier revisions of the corresponding Application Manuals are invalidated.
Note that all the data/numbers/information that are quoted are **current values at the time of printing.** This information is **not legally binding** for dimensioning, calculation and costing.
Within the scope of further-development of our products, Baumüller Nürnberg Electronic GmbH & Co. KG reserve the right to change the technical data and handling.

We cannot guarantee this Application Manual is completely error-free unless this is expressly indicated in our General Conditions of Business and Delivery.

Manufacturer

Baumüller Nürnberg Electronic GmbH & Co. KG
Ostendstr. 80 - 90
90482 Nuremberg
Germany
Tel. +49 9 11 54 32 - 0 Fax: +49 9 11 54 32 - 1 30
www.baumueller.de

Hiermit zeigen wir an, dass die Baumüller Nürnberg Electronic GmbH & Co. KG im Wege der Verschmelzung mit Wirkung zum 01.12.2004 in der Baumüller Nürnberg GmbH aufgegangen ist. Ihr zukünftiger Ansprechpartner ist damit die

Baumüller Nürnberg GmbH, Ostendstrasse 80-90, 90482 Nürnberg.

Please notify that with effect from 01-12-2004 Baumüller Nürnberg Electronic GmbH & Co. KG merged with Baumüller Nürnberg GmbH. Your future business partner will be

Baumüller Nürnberg GmbH, Ostendstrasse 80-90, 90482 Nürnberg.

Par la présente, nous vous signalons qu'en voie de la fusion, la Baumüller Nürnberg Electronic GmbH & Co. KG a été intégrée à la Baumüller Nürnberg GmbH avec effet au 1 décembre 2004. Votre interlocuteur sera par conséquent la

Baumüller Nürnberg GmbH, Ostendstrasse 80 – 90, 90482 Nürnberg.

1

INTRODUCTION

This Application Manual is an important component of your b maXX 4400; this means that you must thoroughly read this document, not least to ensure your own safety.

In this chapter, we will describe the first steps.

1.1 First Steps

1. To program the CANsync-Master you need the following hardware:
   a b maXX 4400 basic unit,
   a b maXX PLC option module and
   a CANsync-Master option module.
   You must have installed the hardware in accordance with the respective Operating Instructions and it must be ready for operation.
2. To program the CANsync-Slave you need the following hardware:
   a b maXX 4400 basic unit,
   a b maXX PLC option module and
   a CANsync-Slave option module.
   You must have installed the hardware in accordance with the respective Operating Instructions and it must be ready for operation.
3. Apart from this, you need the following software:
   PROPROG wt II for programming the b maXX PLC, the CANsync-Master option module and the CANsync-Slave option module for PLC.
   WinBASS II for parameterizing the b maXX 4400 basic unit and the CANsync-Slave option module for controller.

1.2 Terms Used

In this documentation, we will also refer to Baumüller's "BM4-O-CAN-06" product as "option module", "plug-in module" or "CANsync-Master option module".

In this documentation, we will also refer to Baumüller's "BM4-O-CAN-05" product as "option module", "plug-in module" or "CANsync-Slave option module".

We will also refer to the "BM4-O-PLC-01" Baumüller product as "b maXX PLC" or "BM4-O-PLC" and we will use the term "b maXX" for the "b maXX 4400 basic unit".

The controller in the basic unit is also referred to as the "b maXX controller".

For a list of the abbreviations that are used, refer to ▹Appendix A - Abbreviations◃ from page 197 onward.

1.3 Conditions

This manual builds on the "b maXX PLC Application Manual" and assumes that you have knowledge of the PROPROG wt II programming tools and have read its manual.

2

BASIC SAFETY INSTRUCTIONS

We have designed and manufactured each Baumüller plug-in module in accordance with the strictest safety regulations. Despite this, working with the plug-in module can be dangerous for you.

In this chapter, we will describe the risks that can occur when working with a Baumüller plug-in module. Risks are illustrated by icons. All the symbols that are used in this documentation are listed and explained.

In this chapter, we cannot explain how you can protect yourself from specific risks in individual cases. This chapter contains only general protective measures. We will go into concrete protective measures in subsequent chapters directly after information about the individual risk.

2.1 Hazard information and instructions

WARNING

The following **may occur**, if you do not observe this warning information:

- serious personal injury
- death

The hazard information is showing you the hazards which can lead to injury or even to death.

Always observe the hazard information given in this documentation.

Hazards are always divided into three danger classifications. Each danger classification is identified by one of the following words:

DANGER

- Considerable damage to property
- Serious personal injury
- Death **will** occur

WARNING

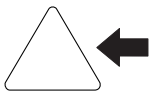
- Considerable damage to property
- Serious personal injury
- Death **can** occur

CAUTION

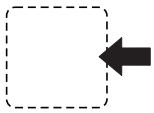
- Damage to property • Slight to medium personal injury **can** occur

2.1.1 Structure of hazard information

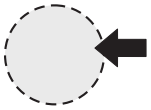
The following two examples show how hazard information is structured in principle. A triangle is used to warn you about danger to living things. If there is no triangle, the hazard information refers exclusively to damage to property.



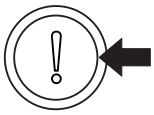
A triangle indicates that there is danger to living things.
The color of the border shows how severe the hazard is: the darker the color, the more severe the hazard is.



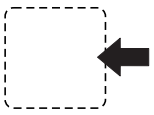
The icon in the rectangle represents the hazard.
The color of the border shows how severe the hazard is: the darker the color, the more severe the hazard is.



The icon in the circle represents an instruction. Users must follow this instruction.
(The circle is shown dashed, since an instruction is not available as an icon for each hazard advisory).



The circle shows that there is a risk of damage to property.



The icon in the rectangle represents the hazard.
The color of the border shows how severe the hazard is: the darker the color, the more severe the hazard is. (The rectangle is shown dashed, since the danger is not represented as an icon with every hazard advisory)

The text next to the icons is structured as follows:

THE SIGNAL WORD IS HERE THAT SHOWS THE DEGREE OF RISK

Here we indicate whether one or more of the results below occurs if you do not observe this warning.

- Here, we describe the possible results. The worst result is always at the extreme right.

Here, we describe the hazard.

Here, we describe what you can do to avoid the hazard.

2.1.2 Hazard advisories that are used

If a signal word is preceded by one of the following danger signs: ⚠ or ⚠ or ⚠, the safety information refers to injury to people.

If a signal word is preceded by a round danger sign: Ⓘ, the safety information refers to damage to property.

2.1.2.1 Hazard advisories about injuries to people

To be able to differentiate visually, we use a separate border for each class of hazard information with the triangular and rectangular pictograms.

For danger classification **DANGER,** we use the ⚠ danger sign. The following hazard information of this danger classification is used in this documentation.

DANGER

The following **will occur**, if you do not observe this danger information:

- serious personal injury
- death

*Danger from: **electricity.** The hazard may be described in more detail here.*

Here, we describe what you can do to avoid the hazard.

DANGER

The following **will occur**, if you do not observe this danger information:

- serious personal injury
- death

*Danger from: **mechanical effects.** The hazard may be described in more detail here.*

Here, we describe what you can do to avoid the hazard.

For danger classification **WARNING,** we use the ⚠ danger sign. The following hazard information of this danger classification is used in this documentation.

WARNING

The following **may occur**, if you do not observe this warning information:

- serious personal injury
- death

*Danger from: **electricity.** The hazard may be described in more detail here.*

Here, we describe what you can do to avoid the hazard.

For danger classification **CAUTION,** we use the ⚠ danger sign. The following hazard information of this danger classification is used in this documentation.

CAUTION

The following **may occur**, if you do not observe this caution information:

- minor to medium personal injury.

*Danger from: **sharp edges.** The hazard may be described in more detail here.*

Here, we describe what you can do to avoid the hazard.

CAUTION

The following **may occur**, if you do not observe this danger information:

- environmental pollution.

*Danger from: **incorrect disposal.** The hazard may be described in more detail here.*

Here, we describe what you can do to avoid the hazard.

2.1.2.2 Hazard advisories about damage to property

If a signal word is preceded by a round danger sign: ⓘ, the safety information refers to damage to property.

CAUTION

The following **may occur**, if you do not observe this caution information:

- property damage.

*Danger from: **electrostatic discharge.** The hazard may be described in more detail here.*

Here, we describe what you can do to avoid the hazard.

2.1.2.3 Instruction signs that are used

carry safety gloves

carry safety shoes

2.2 Information signs

NOTE

This indicates particularly important information.

2.3 Legal information

This documentation is intended for technically qualified personnel that has been specially trained and is completely familiar with all warnings and maintenance measures.

The equipment is manufactured to the state of the art and is safe in operation. It can be put into operation and function without problems if you ensure that the information in the documentation is complied with.

Operators are responsible for carrying out servicing and commissioning in accordance with the safety regulations, applicable standards and any and all other relevant national or local regulations with regard to cable rating and protection, grounding, isolators, overcurrent protection, etc.

Operators are legally responsible for any damage that occurs during assembly or connection.

2.4 Appropriate Use

You must always use the plug-in module appropriately. Some important information is listed below. The information below should give you an idea of what is meant by appropriate use of the plug-in module. The information below has no claim to being complete; always observe all the information that is given in these operating instructions.

- You must only install the plug-in module in series b maXX 4400 units.
- Configure the application such that the plug-in module is always operating within its specifications.
- Ensure that only qualified personnel works with this plug-in module.
- Mount the plug-in module only in the specified slot/slots.
- Install the plug-in module as specified in this documentation.
- Ensure that connections always comply with the stipulated specifications.
- Operate the plug-in module only when it is in technically perfect condition.
- Always operate the plug-in module in an environment that is specified in the technical data.
- Always operate the plug-in module in a standard condition.
  For safety reasons, you must not make any changes to the plug-in module.
- Observe all the information on this topic if you intend to store the plug-in module.

You will be using the plug-in module in an appropriate way if you observe all the comments and information in these operating instructions.

2.5 Inappropriate Use

Below, we will list some examples of inappropriate use. The information below should give you an idea of what is meant by inappropriate use of the plug-in module. We cannon, however, list all possible cases of inappropriate use here. Any and all applications in which you ignore the information in this documentation are inappropriate; particularly, in the following cases:

- You installed the plug-in module in units that are not Series b maXX 4400.
- You ignored information in these operating instructions.
- You did not use the plug-in module as intended.
- You handled the plug-in module as follows
  - you mounted it incorrectly,
  - you connected it incorrectly,
  - you commissioned it incorrectly,
  - you operated it incorrectly,
  - you allowed non-qualified or insufficiently qualified personnel to mount the module, commission it and operate it,
  - you overloaded it,
  - You operated the module
    - with defective safety devices,
    - with incorrectly mounted guards or without guards at all,
    - with non-functional safety devices and guards
    - outside the specified environmental operating conditions
- You modified the plug-in module without written permission from Baumüller Nürnberg Electronic GmbH & Co. KG.
- You ignored the maintenance instructions in the component descriptions.
- You incorrectly combined the plug-in module with third-party products.
- You combined the drive system with faulty and/or incorrectly documented third-party products.
- Your self-written PLC software contains programming errors that lead to a malfunction.

Version 1.1 of Baumüller Nürnberg Electronic GmbH & Co. KG's General Conditions of Sale and Conditions of Delivery dated 2/15/02 or the respective latest version applies in all cases. These will have been available to you since the conclusion of the contract at the latest.

2.6 Protective equipment

In transit, the plug-in modules are protected by their packaging. Do not remove the plug-in module from its packaging until just before you intend to mount it.

The cover on the b maXX units' controller sections provides IP20 protection to the plug-in modules from dirt and damage due to static discharges from contact. This means that you must replace the cover after successfully mounting the plug-in module.

2.7 Personnel training

WARNING

The following **may occur**, if you do not observe this warning information:

- serious personal injury
- death

Only qualified personnel are allowed to mount, install, operate and maintain equipment made by Baumüller Nürnberg Electronic GmbH & Co. KG.

Qualified personnel (specialists) are defined as follows:

Qualified Personnel

Electrical engineers and electricians of the customer or of third parties who are authorized by Baumüller Nürnberg Electronic GmbH & Co. KG and who have been trained in installing and commissioning Baumüller drive systems and who are authorized to commission, ground and mark circuits and equipment in accordance with recognized safety standards.

Qualified personnel has been trained or instructed in accordance with recognized safety standards in the care and use of appropriate safety equipment.

Requirements of the operating staff

The drive system may only be operated by persons who have been trained and are authorized.

Only trained personnel are allowed to eliminate disturbances, carry out preventive maintenance, cleaning, maintenance and to replace parts. These persons must be familiar with the Operating Instructions and act in accordance with them.

Commissioning and instruction must only be carried out by qualified personnel.

2.8 Safety measures in normal operation

- At the unit's place of installation, observe the applicable safety regulations for the plant in which this unit is installed.
- Provide the unit with additional monitoring and protective equipment if the safety regulations demand this.
- Observe the safety measures for the unit in which the plug-in module is installed.

2.9 Responsibility and liability

To be able to work with these option modules in accordance with the safety requirements, you must be familiar with and observe the hazard information and safety instructions in this documentation.

2.9.1 Observing the hazard information and safety instructions

In these operating instructions, we use visually consistent safety instructions that are intended to prevent injury to people or damage to property.

WARNING

The following **may occur**, if you do not observe this warning information:

- serious personal injury
- death

Any and all persons who work on and with Series b maXX units must always have available these Operating Instructions and must observe the instructions and information they contain – this applies in particular to the safety instructions.

Apart from this, any and all persons who work on this unit must be familiar with and observe all the rules and regulations that apply at the place of use.

2.9.2 Danger arising from using this module

The CANsync-Master and CANsync-Slave option modules have been developed and manufactured to the state of the art and comply with applicable guidelines and standards. It is still possible that hazards can arise during use. For an overview of possible hazards, refer to the chapter entitled ▹Basic Safety Instructions ◃ from page 7 onward.
We will also warn you of acute hazards at the appropriate locations in this documentation.

2.9.3 Warranty and Liability

All the information in this documentation is non-binding customer information; it is subject to ongoing further development and is updated on a continuous basis by our permanent change management system.

Warranty and liability claims against Baumüller Nürnberg Electronic GmbH & Co. KG are excluded; this applies in particular if one or more of the causes listed in ▹Inappropriate Use ◃ from page 12 onward or below caused the fault:

- Disaster due to the influence of foreign bodies or force majeure.

3

CANsync

NOTE

The function blocks that are mentioned in this chapter are located in libraries SYSTEM1_PLC01_20bd00 (or above), SYSTEM2_PLC01_20bd00 (or above) and CANsync_PLC01_20bd00 (or above).
The data types that are mentioned in this chapter are defined in library BM_TYPES_20bd03 (or above).
To program the CANsync under PROPROG wt II, you integrate these libraries into a project.

3.1 General

3.1.1 Overview

The CANsync field bus has been developed by Baumüller Nürnberg Electronic GmbH & Co. KG. The aim of replacing mechanical line shafts by an electronic leading axle was achieved by making available to all the connected drives (⇒ CANsync slaves) the leading axle value at the same instant (time-synchronous transfer).

The CAN bus represents the physical basis. The bus was enhanced by adding a synchronization signal (SYNC signal). The SYNC signal is transferred on two additional wires in the CAN cable. The SYNC signal is for hardware synchronizing the CANsync master with all the CANsync slaves that are located on the CANsync bus. By contrast with the CAN bus, this makes it possible to send and receive message frames at defined instants. The system achieves a guaranteed, high data throughput rate, which, in addition, has a fixed time reference on the CANsync bus.

The CANsync bus is a master-slave bus with one CANsync master and up to 32 CANsync slaves. To differentiate the CANsync slaves each one is assigned a slave number. You specify the slave number by means of a DIP switch setting (see the Chapter entitled "DIP switches" in ▹the CANsync slave for b maXX Operating Instructions ◃).

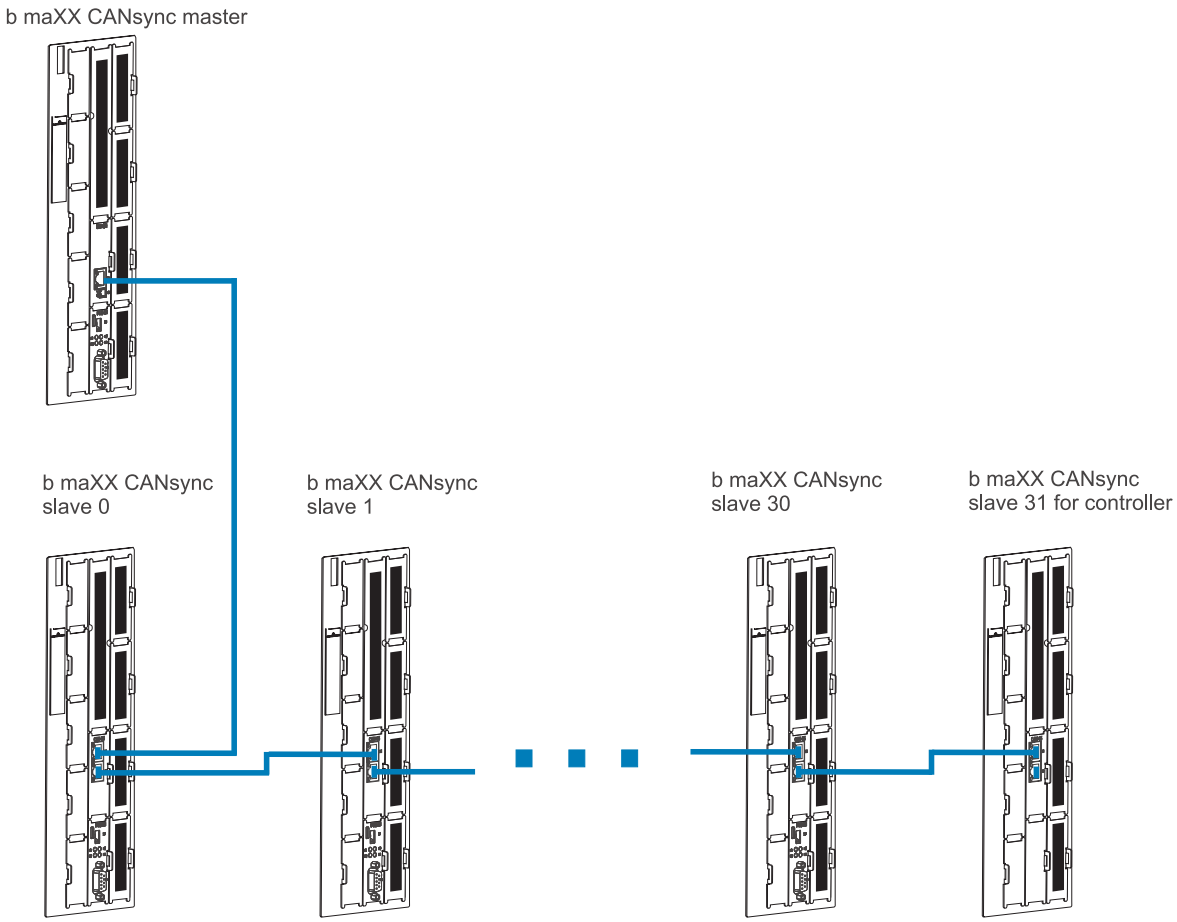


Figure 1: CANsync bus, CANsync master with CANsync slaves 0, 1, .., 30,31

The CANsync synchronization signal (SYNC signal)

- is a specified hardware signal,
- is generated on the CANsync master's CANsync interface module,
- is transferred via two additional lines on the CANsync bus,

Depending on the operating mode and the transmission speed (baud rate) on the CANsync bus, the system generates the SYNC signal in a specific raster of the CANsync cycle time (the time between two falling edges of the SYNC signal).

Baud rate	CANsync cycle time
1 Mbps	1 ms
500 kbps	2 ms
250 kbps	4 ms
125 kbps	8 ms

The message frame traffic of the CANsync bus is carried out in a specified sequence such that individual message frames are always in defined time windows, which are also known as channels. The following channels are defined in CANsync and are sent by the CANsync master:

- Specified value message frames; WRC1 and WRC2 (**Wr**ite**C**hannel)
- Broadcast message frames; CC (**C**ommand**C**hannel)
- Parameter message frames, CC
- Upload/download message frames, CC

The CANsync slaves must adapt their responses to the CANsync master's time scheme; the following channels are available to them:

- Actual value message frames, RDC1 and RDC2 (**R**ea**dC**hannel)
- Parameter response message frames, RC (**R**esponse**C**hannel)
- Upload/download response message frames, RC

NOTE

We will also refer from now on to WRC1 and WRC2 as reference value channel 1 and reference value channel 2.
We will also refer from now on to RDC1 and RDC2 as actual value channel 1 and actual value channel 2.
We will also refer to CC from now on as command channel.
We will also refer to RC from now on as response channel.

The following example shows the time scheme of the respective message frames with a transmission speed of 1 Mbps (CANsync interval with a CANsync cycle time of 1 ms):

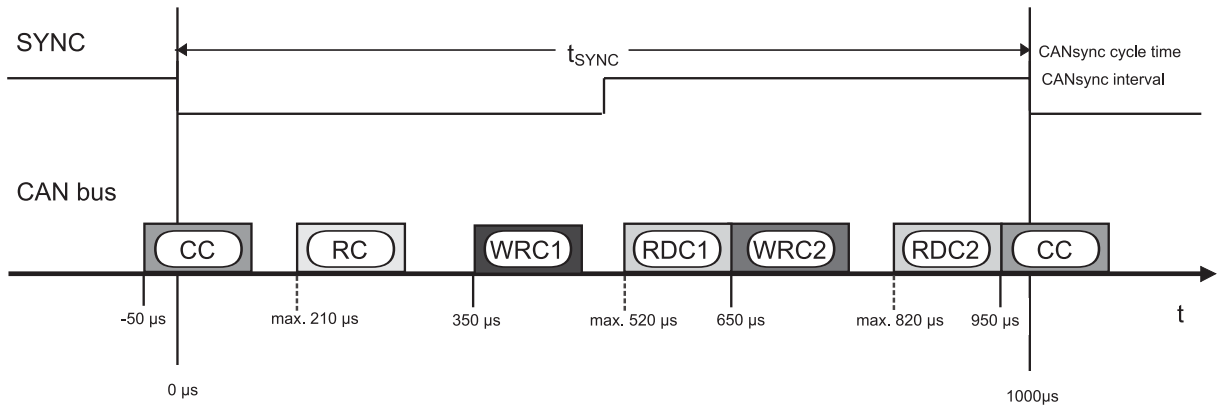


Figure 2: Time scheme of the CANsync interval with a transmission speed of 1 Mbps

The instants for the channels are listed in the table below in dependence on the baud rate:

Table: Assignment of transmission speed (baud rate), CANsync cycle time t_{SYNC} in ms and maximum bus length in m. The quoted times are maximum values.

Baud Rate	CANsync cycle time in µs t_{SYNC}	t_{WRC1} in µs	t_{RDC1} in µs	t_{WRC2} in µs	t_{RDC2} in µs	t_{CC} in µs	t_{RC} in µs	Max. CANsync bus length
1 Mbps	1000	350	520	650	820	950	210	34 m
500 kbps	2000	730	1060	1320	1630	1890	460	134 m
250 kbps	4000	1430	2100	2620	3290	3810	900	300 m
125 kbps	8000	3020	4310	5350	6640	7680	1920	600 m

3.1.1.1 Sequence in principle of communication

Process data communication

Reference value channels WRC1 and WRC2 (WriteChannel), actual value channels RDC1 and RDC2 (ReadChannel) and command channel CC (CommandChannel) are used for process data communication. In every CANsync interval, the system sends them in the specified sequence.

The master sends the reference value message frames on channels WRC1 and WRC2. In each case, they have a defined length of useful data of 64 bits ≙ 8 bytes ≙ 4 words!

All the slaves/nodes receive them with each slave deciding on its own initiative and on the basis of its set configuration (→Æ Mapping) the data that represents reference values for it. You make this setting for

- The CANsync slave option module for b maXX **PLC** (BM4-O-CAN-05; DIP switch 9 = ON) in the mapping to be programmed in the application program on the b maXX PLC option module with FBs CANsync_PD_CFG_SL and CANsync_PD_CFG_READ_SL
- The CANsync slave option module for b maXX **controller** (BM4-O-CAN-05; DIP switch 9 = OFF) in WinBASS II on pages "BACI" and "option module G, H – configuration".

The system adds to the reference value message frame an additional piece of information indicating which slave in the same CANsync interval is to send its prepared actual value message frame back to the master. This means that it is possible to receive the actual value message frames of a maximum of two slaves per CANsync interval, since only two actual value message frames are available (in channels RDC1 and RDC2). These message frames also each have useful data amounting to 64 bits ≙ 8 bytes ≙ 4 words.

You can also program automatic polling of all the slaves in sequence.

Usually, the system only polls one CANsync slave per CANsync interval for its actual values, which means that the actual value message frames on channels RDC1 and RDC2 come from one slave.

NOTE

With a maximum bus configuration of 32 slaves, you need at least 16 CANsync intervals to have available all the current actual values in the master if two slaves per CANsync interval are to report their actual values back to the master!

In the case of the CANsync slave option module for b maXX controller, the control word and the status word belong to the process data. Whereas the system enters the status word like any other ordinary actual value in actual value message frames 1 or 2 in dependence on the set mapping, it must treat the control word separately. It is defined as a broadcast command that is sent in the CC.

Mapping

In the following section, we will describe the principle of mapping using the CANsync master interface module as an example. You make the mapping setting with the CANsync slaves for:

- The CANsync slave option module for b maXX **PLC** (BM4-O-CAN-05; DIP switch 9 = ON) in the application program on the b maXX PLC option module with FBs CANsync_PD_CFG_SL and CANsync_PD_CFG_READ_SL
- The CANsync slave option module for b maXX **controller** (BM4-O-CAN-05; DIP switch 9 = OFF) in WinBASS II on page "option module G, H – configuration.

64 bits are available in each case as the useful data for reference value message frames 1 and 2. This yields the following options for reference value setting by the CANsync master:

4 word reference values (16 bits each)

2 doubleword reference values (32 bits each)

1 doubleword reference value (32 bits) and 2 word reference values (16 bits each)

The system enters the reference values in a field (array) that is connected at FB CANsync_PD_COMM_MA at process data communication. It consists of eight 32 bit entries:

Reference values for reference value message frame 1	Reference value 0	highword	Reference value 0	lowword
	Reference value 1	highword	Reference value 1	lowword
	Reference value 2	highword	Reference value 2	lowword
	Reference value 3	highword	Reference value 3	lowword
Reference values for reference value message frame 2	Reference value 4	highword	Reference value 4	lowword
	Reference value 5	highword	Reference value 5	lowword
	Reference value 6	highword	Reference value 6	lowword
	Reference value 7	highword	Reference value 7	lowword

These reference values are buffered in the CANsync interface module's communication RAM as doublewords. This makes it necessary to tell the CANsync interface module which reference values and which highword or lowword are to be used for the reference value message frame. You must make this setting at function block CANsync_PD_CFG_MA. Since this setting does not generally change any more, users must make it during initialization of the CANsync bus.

NOTE

Doubleword parameters in the CANsync slave option module for b maXX controller (BM4-O-CAN-05; DIP switch 9 = OFF) must be transferred in the following sequence:
first the lowword and then the highword (as stated in the following examples).

Example:

You want to write a doubleword reference value as reference value 0 in reference value message frame 1 (in WRC1) and, in addition, two word reference values as reference value 2 and reference value 3:

Setting to make: a_WRC1 = [0, 0, 2, 3]
a_HL_WRC1 = [FALSE, TRUE, FALSE, FALSE]

Reference values for reference value message frame 1	Reference value 0 highword	Reference value 0 lowword
	----	----
	----	Reference value 2 lowword
	----	Reference value 3 lowword

The arrays that are used are connected to the inputs of FB CANsync_PD_CFG_MA. Array a_WRC1 defines the positions of the reference values to be transferred, doubleword to reference value 0, word reference values to reference values 2 and 3. Array a_HL_WRC1 tells the system whether it is to take the highword (TRUE) or the lowword (FALSE) from communication RAM.

This yields the following assignment for reference value message frame 1 in SWK 1:

	Word 3	Word 2	Word 1	Word 0
SWK 1 / WRC 1	Reference value 3 lowword	Reference value 2 lowword	Reference value 0 highword	Reference value 0 lowword

64 bits of useful data are also available for actual value message frames 1 and 2. This means that it is possible to transfer

4 word actual values (16 bits each)

2 doubleword actual values (32 bits each)

1 doubleword actual value (32 bits) and 2 word actual values (16 bits each)

The system also saves these actual values in communication RAM as doublewords, which means that, here too, you must make an assignment.

Actual values from actual value message frame 1	Actual value 0	highword	Actual value 0	lowword
	Actual value 1	highword	Actual value 1	lowword
	Actual value 2	highword	Actual value 2	lowword
	Actual value 3	highword	Actual value 3	lowword
Actual values from actual value message frame 2	Actual value 4	highword	Actual value 4	lowword
	Actual value 5	highword	Actual value 5	lowword
	Actual value 6	highword	Actual value 6	lowword
	Actual value 7	highword	Actual value 7	lowword

Example:

You want to read from actual value message frame 1 (in RDC1) a doubleword actual value as actual value 0 (position in the message frame words 0 and 1) and to read a second doubleword actual value (words 2 and 3) as actual value 2:

Setting to make: a_RDC1 = [0, 0, 2, 2]
a_HL_RDC1 = [FALSE, TRUE, FALSE, TRUE]

Actual values from actual value message frame 1	Actual value 0	highword	Actual value 0	lowword
	----		----	
	Actual value 2	highword	Actual value 2	lowword
	----		----	

The arrays that are used are connected to the inputs of FB CANsync_PD_CFG_READ_MA. Array a_RDC1 defines the positions of the actual values to be transferred, doubleword to actual value 0, doubleword to actual value 2. Array a_HL_RDC1 tells the system whether it is to write the highword (TRUE) or the lowword (FALSE) to communication RAM.

This yields the following assignment for reference value message frame 1 in SWK 1:

	Word 3	Word 2	Word 1	Word 0
IWK 1 / RDC 1	Actual value 2 highword	Actual value 2 lowword	Actual value 0 highword	Actual value 0 lowword

In the CANsync master, the system writes the actual values of the individual CANsync slaves to different areas of communication RAM, with each CANsync slave being assigned its own area. This field comprises 32 (since this is the maximum number of nodes without clusters) times 8 (since there are 8 actual values) entries that each contain 32 bits (since they are doubleword actual values) of data.

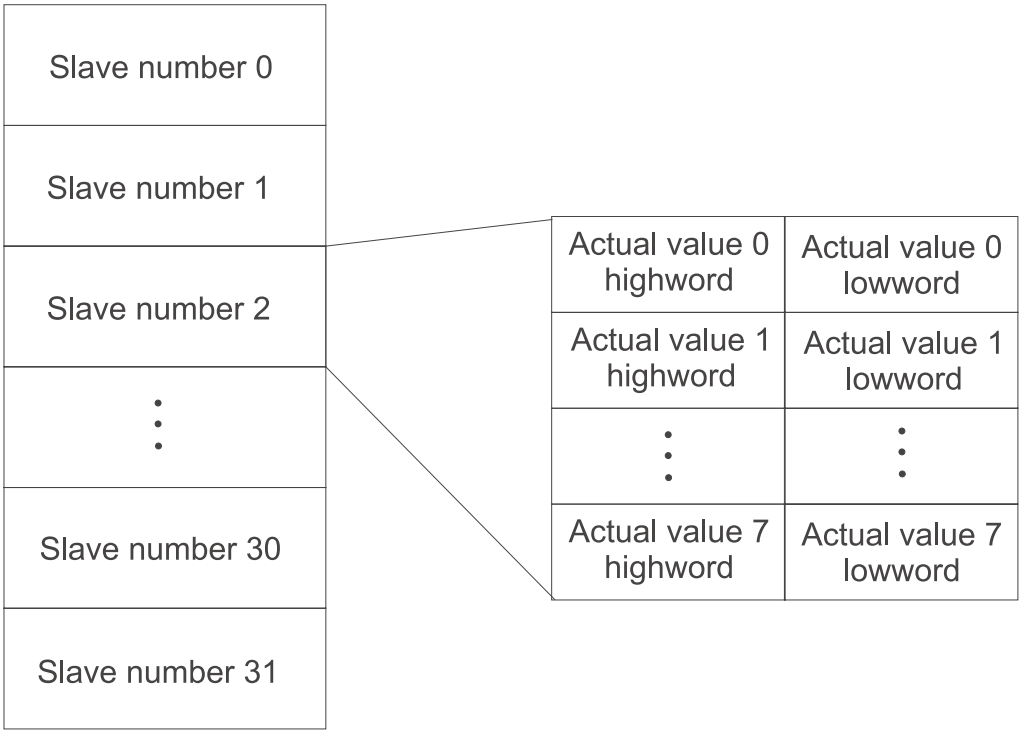


Figure 3: Entering actual values in the CANsync master of the CANsync slaves communication RAM, example for Slave 2

Since only one CANsync slave sends its actual values and possibly its status word in every CANsync actual value message frame, all the other CANsync slaves can monitor these message frames. As a result, it is possible for these actual values to be used as reference values for one or more other CANsync slaves. If this is what you want, you only need to make the setting in the mapping.

Summary of process data communication

In every CANsync interval, the CANsync master sends reference value message frames 1 and 2 and (generally) uses the reference value message frames to request actual value message frames 1 and 2 from a CANsync slave.

All the CANsync slaves receive the CANsync master's reference value message frames and, after a corresponding request, send their actual value message frames to the CANsync master. You make the setting in mapping of which reference values in the reference value message frames are relevant to the CANsync slave. You also define there the combination of actual values that is to be entered in the actual value message frame.

All the CANsync slaves can evaluate the actual value message frames of the other CANsync slaves that the master requests.

You configure mappings for:

- the CANsync master option module for b maXX PLC (BM4-O-CAN-06):
  in the application program on the b maXX PLC with FBs

CANsync_PD_CFG_MA (FB is used once)	Mapping the reference value to be sent message frames 1 and 2 in the b maXX CANsync master
CANsync_PD_CFG_READ_MA (FB is used per CANsync slave a maximum of 32 times)	Mapping received actual value message frames 1 and 2 of a CANsync slaves

- The CANsync slave option module for b maXX **PLC** (BM4-O-CAN-05; DIP switch 9 = ON):
  In the application program on the b maXX PLC with FBs

CANsync_PD_CFG_SL (FB is used once)	Mapping the received reference value message frames 1 and 2 and actual value message frames 1 and 2 to be sent in the CANsync slave for b maXX PLC
CANsync_PD_CFG_READ_SL (FB is used for each further CANsync Slave; max. of 31 times)	Mapping received actual value message frames 1 and 2 of a CANsync slave in the CANsync slave for b maXX PLC

- The CANsync slave option module for b maXX **controller** (BM4-O-CAN-05; DIP switch 9 = OFF)
  in WinBASS II on page "option module G, H – configuration" (see "CANsync slave option module" Operating Instructions).

Requirements Data

In the CANsync, requirements data communication is carried out via the CommandCannel (CC) and the ResponseChannel (RC). In this connection, the CANsync master sends message frames on the command channel (CC) that trigger actions in one or more CANsync slaves.

Several message frames are available:

- broadcast message frames
- control word message frames (special case of a broadcast message frame)
- Parameter message frames
- Upload/download message frames

The CANsync slave sends its response on the response channel (RC); it can be

- parameter response message frames
- Upload/download response message frames.

NOTE

The CANsync master can send command message frames in every CANsync interval. The CANsync slaves only respond following a request by the CANsync master.

The system processes the various message frames on the command channel in a priority-based sequence:

Table: Priority specification of the message frames on the command channel (CC)

Message frame type	Priority
Broadcast message frame 0	Highest
Broadcast message frame 1	↑
Broadcast message frame 2	|
Control word message frames	|
Parameter message frames	↓
Upload/download message frames	Lowest

As a result of these priorities, you cannot send another message frame if a higher-priority one is being transmitted. If you send the control word message frame in every CANsync interval, for example, you can never transmit a parameter message frame or an upload/download message frame!

In each CANsync interval, it is only possible to send one of the following broadcast message frames:

One broadcast message frame to **all the** CANsync slaves or
one control word message frame to **one** CANsync slave or
one parameter message frame to **one** CANsync slave or
one upload/download message frame to **one** CANsync slave.

Assuming that there is no broadcast message frame to send, the system sends control word, parameter or upload/download message frames to one CANsync slave in every CANsync interval.

In this connection, you can set in the CANsync master whether a message frame is to be sent to one specific CANsync slave or if it is to be sent automatically to all the CANsync slaves in succession.

Since the maximum number of CANsync slaves (32 without a cluster) do not necessarily need to be present, it is possible to tell the CANsync master the maximum slave number for sending the control word, parameter, upload/download message frame and for requesting the actual value message frame. It is made at function block CANsync_COMM_CONTROL_MA or CANsync_PD_COMM_MA.

3.1.2 Information on Programming

Initialization

If you use the b maXX 4400 with a b maXX PLC (BM4-O-PLC-01) and the CANsync master option module (BM4-O-CAN-06), you must initialize the CANsync master interface module on the CANsync master option module.

After this, active operation on the CANsync master interface module is enabled.

Necessary FBs and their sequence when initializing the CANsync master interface module on the CANsync master option module:

CANsync_SL_TYP_INIT

CANsync_INIT

CANsync_COMM_CONTROL_MA

CANsync_PD_CFG_MA

CANsync_PD_CFG_READ_MA (one per CANsync slave)

INTR_SET (with i_EVENT = 11) *)

CANsync_MODE_MA

*) FB is not needed if input i_EVENT on FB BACI_INIT becomes 11.

If you use the b maXX 4400 with a b maXX PLC (BM4-O-PLC-01) and the CANsync slave option module (BM4-O-CAN-05, DIP switch 9 = ON), you must initialize the CANsync slave interface module on the CANsync slave option module.

After this, active operation on the CANsync slave interface module is enabled.

Necessary FBs and their sequence when initializing the CANsync slave interface module on the CANsync slave option module:

CANsync_INIT

CANsync_PD_CFG_SL

CANsync_PD_CFG_READ_SL (one per further CANsync slave)

INTR_SET (with i_EVENT = 11) *)

CANsync_MODE_SL

*) FB is not needed if input i_EVENT on FB BACI_INIT becomes 11.

Information on process data communication

The FBs for CANsync process data communication are placed in a POE that is assigned to the CANsync event task (Event 11 - SYNC signal 1 option module). The FBs of process data communication must be called immediately after calling of the event task so that reference values and/or actual values can be sent and/or received in the same call of the event task.

If you use the b maXX 4400 with a b maXX PLC (BM4-O-PLC-01) and the CANsync master option module (BM4-O-CAN-06) as the CANsync master, the system must call FB

CANsync_PD_COMM_MA

If you use the b maXX with a b maXX PLC (BM4-O-PLC-01) and the CANsync slave option module (BM4-O-CAN-05, DIP switch 9 = ON) as a CANsync slave for b maXX PLC, you must call FB

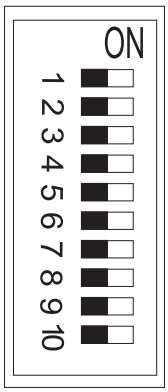
CANsync_PD_COMM_SL

3.1.3 Settings and displays of the option modules

3.1.3.1 Description of the DIP switches on the CANsync slave option module for b maXX

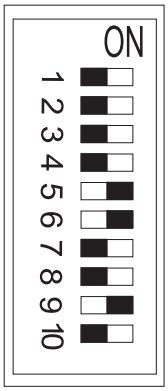
You configure the CANsync slave option module for b maXX PLC using DIP switches. When doing this, you make the following settings:

- CANsync slave number (DIP switches 4 to 8)
- Switching between CANsync slave for b maXX **PLC** and CANsync slave for b maXX **controller** (DIP switch 9)



DIP switches **3 2 1**	reserved or baud rate if DIP switch 9 = OFF
DIP switches **8 7 6 5 4** 0 0 0 0 0 0 0 0 0 1 0 0 0 1 0 .... 1 1 1 1 1	 CANsync slave number 0 CANsync slave number 1 CANsync slave number 2 CANsync slave number 31
DIP switch **9** OFF ON	 CANsync slave for b maXX **controller** CANsync slave for b maXX **PLC**
DIP switch **10**	 Reserved, must be set to OFF

Example 1: CANsync slave for b maXX **PLC**



DIP switches **3 2 1**	Reserved
DIP switches **8 7 6 5 4** 0 0 1 1 0	CANsync slave number 6
DIP switch **9** ON	CANsync slave for b maXX **PLC**
DIP switch **10**	Reserved, must be set to OFF

Example 2: CANsync slave for b maXX **controller**
In this option module application, you must use DIP switches 1 to 3 to set the baud rate (in the illustration, 500 kbps):

DIP switches **3 2 1**	Baud rate
0 0 0	Reserved
0 0 1	Reserved
0 1 0	Reserved
0 1 1	125 kbps
1 0 0	250 kbps
1 0 1	500 kbps
1 1 0	1 Mbps
1 1 1	Reserved
DIP switches **8 7 6 5 4** 0 0 1 0 0	CANsync slave number 4
DIP switch **9** OFF	CANsync slave for b maXX **controller**
DIP switch **10**	Reserved, must be set to OFF

NOTE

DIP switch 9 = ON:
This module is a CANsync slave option module for b maXX **PLC**

DIP switch 9 = OFF:
This module is a CANsync slave option module for b maXX **controller**
see also the ▷Operating Instructions for CANsync slave option module for b maXX,◁ in the section entitled "Operating the CANsync slave Option Module for b maXX Controller".

3.1.4 Description of the LEDs

3.1.4.1 CANsync master option module

RJ45 socket X1 has two LEDs (one green and one red); from now on, they will be referred to as H1 and H2. The LEDs have different meanings during initialization and during operation.

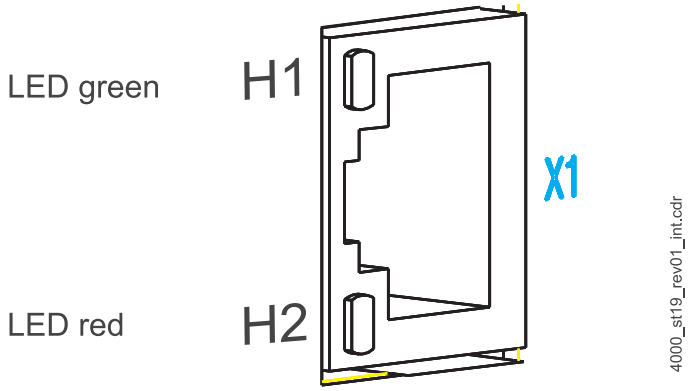


Figure 4: Magnified representation of the LEDs on the front panel

Activating and initializing

After switching on, the LEDs flash briefly one after the other: first the green one and then the red one.

After this, the CANsync master option module is initialized. When doing this, the system displays the following pattern:

Start of initialization	H1 on, H2 off
Initialization running	H1 and H2 on
End of initialization	H1 on, H2 off
Initialization completed	H1 and H2 off

This completes initialization of the CANsync master option module.

If an error occurred at initialization, LED H2 flashes.

For information on eliminating the cause of the error, see the ▷Operating Instructions of the CANsync master Option Module (BM4-O-CAN-06)◁ in the section entitled Finding and Eliminating Disturbances.

Operating the CANsync master option module

After initializing the CANsync master option module, an application program on the b maXX PLC can configure the option module.

Now, the CANsync master option module shows H1 = ON to indicate that is waiting to be configured by the PLC. In the case of the b maXX PLC option module, we also refer to configuration of the CANsync master option module as "initializing the CANsync master adapter on the CANsync master option module for b maXX PLC".

For configuration, see the ▷b maXX PLC Operating Instructions◁ and the ▷b maXX PLC Application Manual ◁.

After an application program on the b maXX PLC has configured the option module, the LEDs have the following meanings:

H1 (green) shows message frames being received and sent to the CANsync bus.

H2 (red) is normally off and only flashes in the case of a fault.

For information on eliminating the cause of the error, see the ▷Operating Instructions of the CANsync master Option Module (BM4-O-CAN-06)◁ in the section entitled Finding and Eliminating Disturbances.

3.1.4.2 CANsync slave option module for b maXX PLC

RJ45 female connectors X1 and X2 have two LEDs each (green and red), which will be referred to from now on as H1 to H4. During initialization and operation of the CANsync slave option module, the LEDs have different meanings.

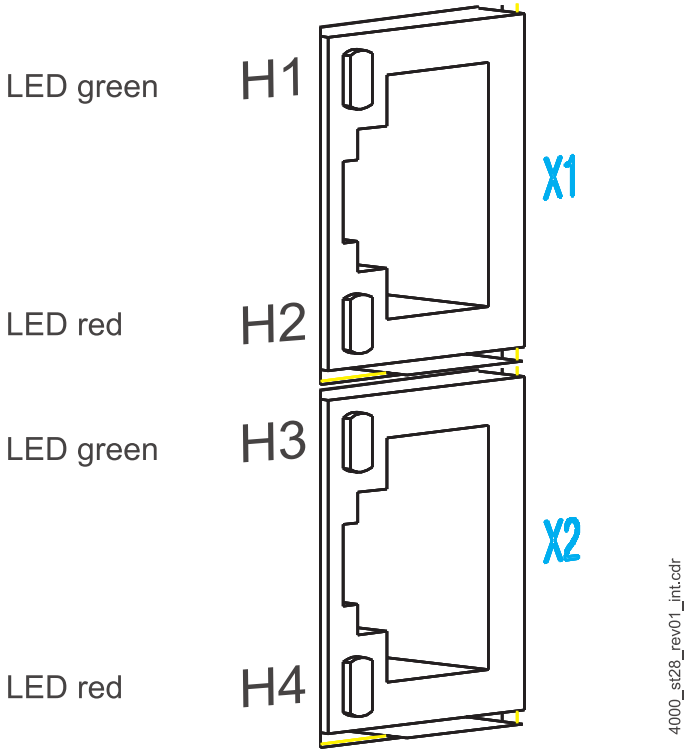


Figure 5: Magnified representation of the LEDs on the front panel

Activating and initializing the CANsync slave option module for b maXX PLC

After switching on, the LEDs briefly light up one after the other in the following order: H1 (green), H2 (red), H3 (green), H4 (red).

After this, the CANsync slave option module is initialized. When doing this, the system displays the following pattern:

Start of initialization	H1 on, H2 to H4 off
End of initialization	H3 on, H1, H2 and H4 off
Initialization completed	H1 to H4 off

This completes basic initialization of the CANsync slave option module. If an error occurred during initialization, LEDs H2 and H4 flash in synch.

For information on eliminating the cause of the error, see the ▷Operating Instructions of the CANsync slave Option Module (BM4-O-CAN-05)◁ in the section entitled Finding and Eliminating Disturbances.

Operating the CANsync slave option module for b maXX PLC

After initializing the CANsync slave option module, an application program on the b maXX PLC can configure the option module.

Now, the CANsync slave option module shows LED H1 = ON to indicate that is waiting to be configured by the PLC.

In the case of the b maXX PLC option module, we also refer to configuration of the CANsync slave option module as "initializing the CANsync slave adapter on the CANsync slave option module for b maXX PLC".

For more information, refer to the ▷b maXX PLC Application Manual◁.

After an application program on the b maXX PLC has configured the option module, the LEDs have the following meanings:

H1 (green) shows message frames being received and sent to the CANsync bus.

H2 (red) shows that the synchronizing signal on the CANsync bus has failed.

H3 (green) flashes to show the duration of a SYNC interval on the CANsync bus in accordance with the following pattern:

Pattern	SYNC Interval (baud rate)
One flash then a pause	1 ms (1 Mbps)
Two flashes then a pause	2 ms (500 kbps)
Four flashes then a pause	4 ms (250 kbps)
Eight flashes then a pause	8 ms (125 kbps)

H4 (red) is normally off and only flashes in the case of a fault.

For information on eliminating the cause of the error, see the ▷Operating Instructions of the CANsync slave Option Module (BM4-O-CAN-05)◁ in the section entitled Finding and Eliminating Disturbances.

3.1.4.3 Activating and initializing the CANsync slave option module for b maXX controller

After switching on, the LEDs briefly light up one after the other in the following order: H1 (green), H2 (red), H3 (green), H4 (red).

After this, the CANsync slave option module is initialized. When doing this, the system displays the following pattern:

Start of initialization H1 on, H2 to H4 off

End of initialization H3 on, H1, H2 and H4 off

Initialization completed H1 to H4 off

This completes basic initialization of the CANsync slave option module. If an error occurred at initialization, LED H2 flashes.

For information on eliminating the cause of the error, see the ▷Operating Instructions of the CANsync slave Option Module (BM4-O-CAN-05)◁ in the section entitled Finding and Eliminating Disturbances.

Operating the CANsync slave option module for b maXX controller

After initializing the CANsync slave option module, you configure the option module according to the parameters that are set in WinBASS II for the option module.

After configuring the option module, the LEDs have the following meanings:

H1 (green) shows message frames being received and sent to the CANsync bus.

H2 (red) shows that the synchronizing signal on the CANsync bus has failed.

H3 (green) flashes to show the duration of a SYNC interval on the CANsync bus in accordance with the following pattern:

Pattern	SYNC Interval (baud rate)
One flash then a pause	1 ms (1 Mbps)
Two flashes then a pause	2 ms (500 kbps)
Four flashes then a pause	4 ms (250 kbps)
Eight flashes then a pause	8 ms (125 kbps)

H4 (red) is normally off and only flashes in the case of a fault.

For information on eliminating the cause of the error, see the ▷Operating Instructions of the CANsync slave Option Module (BM4-O-CAN-05)◁ in the section entitled Finding and Eliminating Disturbances.

H2 (red) and H4 (red) flash alternately:
You used DIP switches 1 to 3 to set a baud rate that the option module does not support.

H2 (red) and H4 (red) ON:
The option module is configured and is waiting for the CANsync master to start the CANsync bus.

3.2 Detailed Information on CANsync

NOTE

This chapter contains detailed information on CANsync. You do not need this information if you use the function blocks of library CANsync_PLC01_20bd00 or above for programming.

t_{RSPTO} - Response Timeout: The time within which the CANsync slave must send a response during initialization.

Baud rate	t_{RSPTO}
1 Mbps	300 µs
500 kbps	600 µs
250 kbps	1100 µs
125 kbps	2100 µs

NOTE

All the following timings are relative to a baud rate of 1 Mbps.

3.2.0.1 Start-up and initialization

Starting characteristics are divided into the following steps:

⇨ **Initializing using FB CANsync_INIT**

After FB CANsync_INIT has been successfully run through, the CANsync master issues the SYNC signal with a 1 ms timing code and the "SYNC mode" action command with data content "activate SYNC operation" to all the CANsync slaves. The CANsync slaves start to synchronize their control task to the SYNC signal.

The CANsync master uses the Parameter Lesen (read parameter) parameter command to request the status word from each CANsync slave that it expects on the CANsync bus.

The CANsync slave must respond with its parameter response within tRSPTO. The CANsync master monitors whether the CANsync slave responds within this time.

The CANsync master does not yet output any reference values during synchronization.

⇨ **Transition to active operation**

Using FB CANsync_Mode (input x_CANsync_RUN = TRUE), you directly start active operation. In this case, the system also transfers reference values and actual values.

In this connection, note that the CANsync slaves take some time (a few seconds) to synchronize to the SYNC signal. The status word of the CANsync slave option module for b maXX controller (BM4-O-CAN-05; DIP switch 9 = OFF) contains the information (bit 15 = TRUE) indicating whether it is synchronized.

If you want to transfer sychronized position reference values, the application program must not start the machine until all the CANsync slaves are synchronized (= status "synchronized"). Other reference values or operating modes are permitted before this, however.

3.2.0.2 Synchronized Status

In the "synchronized" status, all the drives (CANsync master, CANsync slaves) in the same CANsync interval, i.e. all the CANsync slaves process their control tasks at the same time and also take the reference values at the same time.

The CANsync master sends its jobs in a defined sequence with assigned time windows. The CANsync slaves must adapt their responses to this scheme (see ▹Overview ◃ from page 15 onward and ▹Figure2 on page 17).

The CANsync master must transmit the next reference value message frame 1 by 350 µs after the SYNC signal at the latest.

3.2.0.3 Reference value loss at CANsync slave option module for controller

If no new reference value is received in a CANsync interval, the CANsync slave carries out extrapolation using the reference value that it received last.

3.2.1 Structure of message frames

⇨ **CANsync message frames**

Message frame lengths are variable (0 to 8 data bytes). They result from the list above and in some cases are also dependent on the respective operating status.

⇨ **Data format**

The data is stored in the message frames in Intel format (low byte/high byte).

⇨ **Status word**

A CANsync slave's status word indicates its drive status. The SYNC status must be displayed in the topmost bit (15). If the bit is set, then the CANsync slave is synchronized.

3.2.1.1 Reference value channels

Reference value channel 1

On reference value channel 1, the CANsync master sends reference value message frame 1.

With this, the CANsync master transfers to all the CANsync slaves one or more reference values (up to a maximum of four). Number NNNNNNN in the identifier indicates which CANsync slave must send its actual value message frame 1 after reference value message frame 1.

<IDENTIFIER><SOLLWERTE>

<IDENTIFIER>	::=	0010NNNNNNN	Identifier of reference value channel 1
<SOLLWERTE>	::=	W_SOLL | DW_SOLL | W_DW_SOLL	
<W_SOLL>	::=	<W_SOLL_1> | <W_SOLL_1..2> | <W_SOLL_1..3> | <W_SOLL_1..4>	Word reference values only
<DW_SOLL>	::=	<DW_SOLL_1> | <DW_SOLL_1..2>	DWord specified values only
<W_DW_SOLL>	::=	<DW_SOLL_1><W_SOLL_3> | <DW_SOLL_1><W_SOLL_3><W_SOLL_4>	Word and DWord reference values
<W_SOLL_1>	::=	<Word>	Word reference value 1 = CAN-DB 0..1
<W_SOLL_2>	::=	<Word>	Word reference value 2 = CAN-DB 2..3
<W_SOLL_3>	::=	<Word>	Word specified value 3 = CAN-DB 4..5
<W_SOLL_4>	::=	<Word>	Word reference value 4 = CAN-DB 6..7
<DW_SOLL_1>	::=	<DWord>	DWord reference value 1 = CAN-DB 0..3
<DW_SOLL_2>	::=	<DWord>	DWord reference value 2 = CAN-DB 4..7
			CAN-DB: CAN data byte

Reference value channel 2

On reference value channel 2, the CANsync master sends reference value message frame 2.

With this, the CANsync master transfers up to four additional reference values on this reference value channel.

The structure and function correspond to reference value channel 1. Reference value message frame 2 has a different identifier and assignment of reference values to reference value message frame 1. Word reference values 5...8 and doubleword reference values 3...4 are assigned to reference value channel 2. Number NNNNNNN in the identifier indicates the slave number of the CANsync slaves that responds with actual value message frame 2.

<IDENTIFIER>	::=	0011NNNNNNN	Identifier of specified value channel 2

3.2.1.2 Actual value channels

Actual value channel 1

On actual value channel 1, the CANsync slave sends actual value message frame 1.

The identifier of reference value message frame 1 indicates which CANsync slave may send its actual value message frame 1 as a direct response to the reference value 1 message frame.

<IDENTIFIER><ISTWERTE>			
<IDENTIFIER>	::=	0110NNNNNNN	NNNNNNN = CANsync slave number
<ISTWERTE>	::=	W_IST | DW_IST | W_DW_IST	
<W_IST>	::=	<W_IST_1> | <W_IST_1..2> | <W_IST_1..3> | <W_IST_1..4>	Word actual values only
<DW_IST>	::=	<DW_IST_1> | <DW_IST_1..2>	DWord actual values only
<W_DW_IST>	::=	<DW_IST_1><W_IST_3> | <DW_IST_1><W_IST_3><W_IST_4>	Word and DWord actual values
<W_IST_1>	::=	<Word>	Word actual value 1 = CAN-DB 0..1
<W_IST_2>	::=	<Word>	Word actual value 2 = CAN-DB 2..3
<W_IST_3>	::=	<Word>	Word actual value 3 = CAN-DB 4..5
<W_IST_4>	::=	<Word>	Word actual value 4 = CAN-DB 6..7
<DW_IST_1>	::=	<DWord>	DWord actual value 1 = CAN-DB 0..3
<DW_IST_2>	::=	<DWord>	DWord actual value 2 = CAN-DB 4..7
			CAN-DB: CAN data byte

Actual value channel 2

On actual value channel 2, the CANsync slave sends actual value message frame 2.

The structure and the function of the message frame correspond to actual value message frame 1. A maximum of four further actual values can be transferred. Actual value message frame 2 has a different identifier and assignment of the actual values. Word actual values 5...8 and doubleword actual values 3...4 are assigned to actual value channel 2.

<IDENTIFIER>	::=	0111NNNNNNN	NNNNNNN = CANsync slave number

3.2.1.3 Command Channel & Response Channel

The command channel and the associated response channel consist functionally of three groups of message frames: in this connection, only one command/response can ever occur in any one CANsync interval.

- ⇨ Action commands are for initializing and controlling the CANsync slaves and are sent to one or more CANsync slaves without the CANsync master expecting a response.
- ⇨ Parameter commands are used for reading or writing a parameter and are always directed to one CANsync slave. The CANsync master always expects a response.
- ⇨ Upload/download commands are for transferring large volumes of data (program code, data records) and are always directed to one CANsync slave. The master always expects a response.

Parameter and upload/download commands are sent with the same identifier.

Action command

The CANsync master sends an action command to a single CANsync slave or to a group of CANsync slaves. The system makes the choice by means of a bit strip (SLAVE_GROUP) in which one bit is assigned to each CANsync slave. When this bit is set, the associated CANsync slave must carry out this command. In a broadcast command to all the CANsync slaves, all the bits in the bit strip are set.

The various commands are differentiated in the COMMAND data byte. In dependence on the command, there are different numbers of data bytes that contain the data for the command.

<IDENTIFIER><SLAVE_GROUP><COMAND><DATA>

<IDENTIFIER>	::=	00000010000	Identifier of action command
<SLAVE_GROUP>	::=	<DWord>	Slave bits 0..30 = CAN-DB 0..3
<COMMAND>	::=	<Byte> 1 = Write control word	= CAN-DB 4
<DATA>	::=	<DATA_1>	data that depends on the command
<DATA_1>	::=	<res><Wert>	
<res>	::=	<Byte>	CAN-DB 5
<Wert>	::=	<Word>	Control word = CAN-DB 6..7
			CAN-DB: CAN data byte

Read parameter

The CANsync master uses the Parameter Lesen (read parameter) command to request a parameter of the CANsync slave for reading. The message frame length (of 4 data bytes) tells the CANsync slave that this is a read parameter command. The CANsync slave must not necessarily support the element selection: in this case it always responds with the current data value.

The CANsync slave must respond within the specified response time, $t_{RSPTO.}$ If it cannot finish the job by then, it responds with the parameter response in which the job's param-

eter number is entered and the BUSY bit is set. The next time that the CANsync master repeats the read parameter command to the parameter and the CANsync slave has processed the job in the meantime, it responds with the requested data and the BUSY bit is set to zero.

If the read job cannot be processed or an error occurs, the CANsync slave sets the ERR bit and states an error code in the data bytes.

Parameter numbers can be between 0 and 4095.

Job:
<IDENTIFIER><CONTROL><PARA_NUM_L><SUB-ADRESSE>

<IDENTIFIER>	::=	1010NNNNNNN	NNNNNNN = CANsync slave number
<CONTROL>	::=	<P><ELEMENT><PARA_NUM_H>	CAN-DB 0
<P>	::=	<Bit7>	0 = Identifier: Parameter command
<ELEMENT>	::=	<Bit6..4>	Element selection of the parameter
<PARA_NUM_H>	::=	<Bit3..0>	Bits 11..8 of the parameter number
<PARA_NUM_L>	::=	<Byte>	Bits 7..0 of parameter no. = CAN-DB 1
<SUB-ADRESSE>	::=	<Word>	Sub-slave address = CAN-DB 2..3

Response:
<IDENTIFIER><STATUS><PARA_NUM_L><DATA><SUB-ADRESSE>

<IDENTIFIER>	::=	1011NNNNNNN	NNNNNNN = CANsync slave number
<STATUS>	::=	<P><BUSY><ERR> <FREI><PARA_NUM_H>	CAN-DB 0
<P>	::=	<Bit7>	0 = Identifier: Parameter response
<BUSY>	::=	<Bit6>	0 = Response valid, 1 = Job being processed
<ERR>	::=	<Bit5>	0 = No error, 1 = Error
<FREI>	::=	<Bit4>	free
<PARA_NUM_H>	::=	<Bit3..0>	Bits 11..8 of the parameter number
<PARA_NUM_L>	::=	<Byte>	Bits 7..0 of parameter no. = CAN-DB 1
<DATA>	::=	<2_BYTE> | <4_BYTE> | <ERR_CODE>	
<2_BYTE>	::=	<Word>	Word parameter = CAN-DB 2..3
<4_BYTE>	::=	<Dword>	DWord parameter = CAN-DB 2..5
<ERR_CODE>	::=	<Word>	2-byte error code = CAN-DB 2..3
<SUB-ADRESSE>	::=	<Word>	Sub-slave address = CAN-DB 4..5 / 6..7 CAN-DB: CAN data byte

Write parameter

Using the Write parameter command, the CANsync master writes a parameter to a CANsync slave. The message frame length (6 or 8 data bytes) tells the CANsync slave whether the parameter in question is a word or doubleword parameter. Currently, when writing only element 7, the parameter value, is permissible.

The CANsync slave must respond within the specified response time, $t_{RSPTO.}$ If it cannot finish the job by then, it responds with the parameter response in which the job's parameter number is entered and the BUSY bit is set. The next time that the CANsync master repeats the write parameter command to the parameter and the CANsync slave has processed the job in the meantime, it responds with the parameter number and the BUSY bit is set to zero.

If the write job cannot be processed or an error occurs, the CANsync slave sets the ERR bit and states an error code in the data bytes.

Parameter numbers can be between 0 and 4095.

Job:
<IDENTIFIER><CONTROL><PARA_NUM_L><DATA><SUB-ADRESSE>

<IDENTIFIER>	::=	1010NNNNNNN	NNNNNNN = CANsync slave number
<CONTROL>	::=	<P><ELEMENT><PARA_NUM_H>	CAN-DB 0
<P>	::=	<Bit7>	0 = Identifier: Parameter command
<ELEMENT>	::=	<Bit6..4>	Element selection of the parameter
<PARA_NUM_H>	::=	<Bit3..0>	Bits 11..8 of the parameter number
<PARA_NUM_L>	::=	<Byte>	Bits 7..0 of parameter no. = CAN-DB 1
<DATA>	::=	<2_BYTE> | <4_BYTE>	
<2_BYTE>	::=	<Word>	Word parameter = CAN-DB 2..3
<4_BYTE>	::=	<Dword>	DWord parameter = CAN-DB 2..5
<SUB-ADRESSE>	::=	<Word>	Sub-slave address = CAN-DB 4..5 / 6..7

Response:
<IDENTIFIER><STATUS><PARA_NUM_L><DATA><SUB-ADRESSE>

<IDENTIFIER>	::=	1011NNNNNNN	NNNNNNN = CANsync slave number
<STATUS>	::=	<P><BUSY><ERR> <FREI> <PARA_NUM_H>	CAN-DB 0
<P>	::=	<Bit7>	0 = Identifier: Parameter response
<BUSY>	::=	<Bit6>	0 = Job finished, 1 = Job being processed
<ERR>	::=	<Bit5>	0 = No error, 1 = Error
<FREI>	::=	<Bit4>	Not yet assigned
<PARA_NUM_H>	::=	<Bit3..0>	Bits 11..8 of the parameter number
<PARA_NUM_L>	::=	<Byte>	Bits 7..0 of parameter no. = CAN-DB 1
<DATA>	::=	<0_BYTE> | <ERR_CODE>	
<0_BYTE>	::=	No data bytes if error-free	
<ERR_CODE>	::=	<Word>	2-byte error code = CAN-DB 2..3
<SUB-ADRESSE>	::=	<Word>	Sub-slave address = CAN-DB 2..3/4..5
			CAN-DB: CAN data byte

Start of an upload or download

Using uploading or downloading, you can transfer relatively large contiguous data ranges from the CANsync master to the CANsync slave or vice versa.

You configure the transfer using an initialization message frame.

The CANsync slave must respond within the specified response time, $t_{RSPTO.}$ If it cannot finish the job by then, it responds with the upload/download response in which the BUSY bit is set. The next time that the CANsync master repeats the upload/download message frame and the CANsync slave has processed the job in the meantime, it responds with the response in which the BUSY bit is set to zero.

If the upload or download job cannot be processed or an error occurs, the CANsync slave sets the ERR bit and states an error code in the data bytes.

The start address is a doubleword address. The maximum length of an upload or download is 4096 bytes. You must transfer larger data ranges by means of several upload/ download initializations. As an option, you can also state a sub-slave address. This ad-

dress indicates that the subsequent upload/download does not refer directly to the addressed CANsync slave, but rather that the upload/download message frames are passed on to a sub-slave. This sub-slave address remains valid until the end of the upload/download. The address must be stated again for the next upload/download job. If the sub-address is equal to zero, the system addresses the CANsync slave directly and not a sub-slave.

Job:
<IDENTIFIER><CONTROL><OFFSET_L><ADRESSE><SUB-ADRESSE>

<IDENTIFIER>	::=	1010NNNNNNN	NNNNNNN = CANsync slave number
<CONTROL>	::=	<L><U/D><MODE><OFFSET_H>	CAN-DB 0
<L>	::=	<Bit7>	1 = Identifier: Upload/download job
<U/D>	::=	<Bit6>	0 = Upload, 1 = Download
<MODE>	::=	<Bit5..4>	01 = Initialization
<OFFSET_H>	::=	<Bit3..0>	Block length in bytes bits 11..8[1)]
<OFFSET_L>	::=	<Byte>	Block length in bytes bits 7..0 = CAN-DB 1 [1)]
<ADRESSE>	::=	<Dword>	Absolute start address = CAN-DB 2..5
<SUB-ADRESSE>	::=	<Word>	Sub-slave address = CAN-DB 6..7

Response:
<IDENTIFIER><STATUS><OFFSET_L><DATA>

<IDENTIFIER>	::=	1011NNNNNNN	NNNNNNN = CANsync slave no.
<STATUS>	::=	<L><BUSY><ERR><FREI><OFFSET_H>	CAN-DB 0
<L>	::=	<Bit7>	1 = Identifier: Upload/download response
<BUSY>	::=	<Bit6>	0 = Job finished, 1 = Job being processed
<ERR>	::=	<Bit5>	0 = No error, 1 = Error
<FREI>	::=	<Bit4>	Not yet assigned
<OFFSET_H>	::=	<Bit3..0>	Block length in bytes bits 11..8 [1)]
<OFFSET_L>	::=	<Byte>	Block length in bytes bits 7..0 = CAN-DB 1 [1)]
<DATA>	::=	<0_BYTE> | <ERR_CODE>	
<0_BYTE>	::=	No data bytes if error-free	
<ERR_CODE>	::=	<Word>	Bits 15..0 of error code = CAN-DB 2..3 CAN-DB: CAN data byte

Ongoing Upload and End of an Upload

The upload procedure consists of successive upload message frames in which the CANsync master requests successive data blocks starting from the start address that was set at initialization. The offset address increases consecutively as the byte address. The system transfers 6 bytes of useful data with each message frame. This means that the first message frame starts with offset address 0, the second one requests the data with offset address 6, etc.

---

1) In the case of an upload job, the CANsync slave can determine the length instead of the CANsync master. In this case, the length is stated in the answer in <OFFSET_H> and <OFFSET_L>. Otherwise, these values in the answer are zero.

The CANsync slave must respond within the specified response time, $t_{RSPTO.}$ If it cannot finish the job by then, it responds with the upload response in which the job's offset address is entered and the BUSY bit is set. The next time that the CANsync master repeats the upload message frame and the CANsync slave has processed the job in the meantime, it responds with the requested data block and the BUSY bit is set to zero.

In the last message frame, MODE is set to 11. The CANsync slave checks whether it has reached the end of the set data range and always sends the last data block with six data bytes. If the memory area to be loaded does not contain as much data, the system pads it with irrelevant data. If the CANsync slave has not reached the end, it responds with a set ERR bit and an error code.

The CANsync slave sets the ERR bit and indicates an error code in the data bytes even if the upload job cannot be processed or if the CANsync slave determines a gap in the requested offset addresses. If necessary, the CANsync master can repeat the failed message frame, or it cancels the upload by setting MODE to 01 and entering 0 as the base address and the block length.

Job:
<IDENTIFIER><CONTROL><OFFSET_L>

<IDENTIFIER>	::=	1010NNNNNNN	NNNNNNN = CANsync slave number
<CONTROL>	::=	<UD><U/D><MODE><OFFSET_H>	CAN-DB 0
<UD>	::=	<Bit7>	1 = Identifier: Upload/download job
<U/D>	::=	<Bit6>	0 = Upload
<MODE>	::=	<Bit5..4>	10 = Body
			11 = Last block
<OFFSET_H>	::=	<Bit3..0>	Bits 11..8 of the offset address
<OFFSET_L>	::=	<Byte>	Bits 7..0 of the offset address = CAN-DB 1

Response:
<IDENTIFIER><STATUS><OFFSET_L><DATA>

<IDENTIFIER>	::=	1011NNNNNNN	NNNNNNN = CANsync slave number
<STATUS>	::=	<UD><BUSY><ERR><FREI><OFFSET_H>	CAN-DB 0
<UD>	::=	<Bit7>	1 = Identifier: Upload/download response
<BUSY>	::=	<Bit6>	0 = Job finished,
			1 = Job being processed
<ERR>	::=	<Bit5>	0 = No error, 1 = Error
<FREI>	::=	<Bit4>	not yet assigned
<OFFSET_H>	::=	<Bit3..0>	Bits 11..8 of the offset address
<OFFSET_L>	::=	<Byte>	Bits 7..0 of the offset address = CAN-DB 1
<DATA>	::=	<DATEN> | <ERR_CODE>	
<DATEN>	::=	<Word><Word><Word>	6 bytes of data = CAN-DB 2..7
<ERR_CODE>	::=	<Word>	Error code = CAN-DB 2..3
			CAN-DB: CAN data byte

Ongoing Download and End of a Download

The download procedure consists of successive download message frames in which the CANsync master sends successive data blocks starting from the start address that was set at initialization. The offset address increases consecutively as the byte address. This means that the first message frame starts with offset address 0, the second one sends the data with offset address 6, etc.

The CANsync slave must respond within the specified response time, t_{RSPTO}. If it cannot finish the job by then, it responds with the download response in which the job's offset address is entered and the BUSY bit is set. The next time that the CANsync master repeats the download message frame and the CANsync slave has processed the job in the meantime, it responds with the response message frame in which the BUSY bit is set to zero.

In the last message frame, MODE is set to 11 and it also contains six data bytes. However, the CANsync slave may only take the data bytes that correspond to the previously set download length. If the CANsync slave has not yet reached the end, it responds with a set ERR bit and an error code.

The CANsync slave sets the ERR bit and indicates an error code in the data bytes even if the download job cannot be processed or if the CANsync slave determines a gap in the sent offset addresses. If necessary, the CANsync master can repeat the failed message frame, or it cancels the download by setting MODE to 01 and entering 0 as the base address and the block length.

Job:
<IDENTIFIER><CONTROL><OFFSET_L><DATA>

<IDENTIFIER>	::=	1010NNNNNNN	NNNNNNN = CANsync slave number
<CONTROL>	::=	<UD><U/D><MODE><OFFSET_H>	CAN-DB 0
<UD>	::=	<Bit7>	1 = Identifier: Upload/download job
<U/D>	::=	<Bit6>	1 = Download
<MODE>	::=	<Bit5..4>	10 = Body
			11 = Last block
<OFFSET_H>	::=	<Bit3..0>	Bits 11..8 of the offset address
<OFFSET_L>	::=	<Byte>	Bits 7..0 of the offset address = CAN-DB 1
<DATA>	::=	<Word><Word><Word>	6 bytes of useful data = CAN-DB 2..7

Response:
<IDENTIFIER><STATUS><OFFSET_L><DATA>

<IDENTIFIER>	::=	1011NNNNNNN	NNNNNNN = CANsync slave number
<STATUS>	::=	<UD><BUSY><ERR><FREI><ERR_CODE_H>	CAN-DB 0
<UD>	::=	<Bit7>	1 = Identifier: Upload/download
<BUSY>	::=	<Bit6>	0 = Job finished,
			1 = Job being processed
<ERR>	::=	<Bit5>	0 = No error, 1 = Error
<FREI>	::=	<Bit4>	not yet assigned
<OFFSET_H>	::=	<Bit3..0>	Bits 11..8 of the offset address
<OFFSET_L>	::=	<Byte>	Bits 7..0 of the offset address = CAN-DB 1
<DATA>	::=	<0_BYTE> | <ERR_CODE>	
<0_BYTE>	::=	No data bytes if error-free	
<ERR_CODE>	::=	<Word>	Bits 15..0 of error code = CAN-DB 2..3
			CAN-DB: CAN data byte

3.2.2 Register structure and function of the CANsync master option module for b maXX PLC

Below, we will go into the register structure of communication RAM in the CANsync master option module for b maXX PLC.

To allow you to access registers of the communication RAM in the PROPROG wt II project, data types are defined that map the register structure. The system uses these data types to declare variables that are assigned to the CANsync interface module's base address.

After this, it is possible to access the registers of communication RAM via the structure elements of the declared variables.

At initialization of the CANsync master interface module, the registers in communication RAM have a different meaning than after initialization in cyclical operation.

This means that, for initialization there is the

CANsync_PLC_INIT_BMSTRUCT

and for cyclical operation, the

CANsync_PLC_MA_BMSTRUCT

These structures are defined from library BM_TYPES_20bd03 onwards. After you have integrated library BM_TYPES_20bd03 in the project, the data types are available.

These structures contain

- 8-bit elements,
- 16-bit elements,
- 32-bit elements,
- Structures from the elements mentioned above
- Fields (ARRAY) and structures from the elements and structures mentioned above

Short designations have been prepended to the data types (8-, 16-, 32-bit elements, structures and fields) that are used in a structure. This is for the sake of clarity when using the structures in programming.

Data type	Short designation	Number of bits
BYTE	b	8
WORD	w	16
DWORD (double word)	d	32
SINT (short integer)	si	8
DINT (double integer)	di	32
USINT (unsigned short integer)	us	8
UINT (unsigned integer)	u	16
UDINT (unsigned double integer)	ud	32
STRUCT	_ (underline)	-
ARRAY	a	-

Other data types that are not used in the structures include:

Data type	Short designation	Number of bits
BOOL (bit)	x	1
TIME	t	-

3.2.2.1 Explanation of declaring the global variables

For initialization, global variables of data type CANsync_PLC_INIT_BMSTRUCT are already declared for each option module slot in template "BM4-O-PLC01".

These are the following global variables

```
_CANsyncMaster_Init_Slot_G  AT  %MB3.2001792  :
                                CANsync_PLC_INIT_BMSTRUCT;
_CANsyncMaster_Init_Slot_H  AT  %MB3.3001792  :
                                CANsync_PLC_INIT_BMSTRUCT;
_CANsyncMaster_Init_Slot_J  AT  %MB3.4001792  :
                                CANsync_PLC_INIT_BMSTRUCT;
_CANsyncMaster_Init_Slot_K  AT  %MB3.5001792  :
                                CANsync_PLC_INIT_BMSTRUCT;
_CANsyncMaster_Init_Slot_L  AT  %MB3.6001792  :
                                CANsync_PLC_INIT_BMSTRUCT;
_CANsyncMaster_Init_Slot_M  AT  %MB3.7001792  :
                                CANsync_PLC_INIT_BMSTRUCT;
```

If these variables are no longer in the PROPROG wt II project, create – in dependence on the slot (G to M) – the global variable `_CANsyncMaster_Init_Slot_G` (to `_CANsyncMaster_Init_Slot_M`) of data type `CANsync_PLC_INIT_BMSTRUCT`.

At declaration of global variables, this variable must be connected to the base address of the CANsync interface module on the option module.

In dependence on the slot, the base address of the CANsync interface module on the CANsync master option module (BM4-O-CAN-06) is as follows:

Slot	Base address of CANsync interface module
G	%MB3.2001792
H	%MB3.3001792
J	%MB3.4001792
K	%MB3.5001792
L	%MB3.6001792
M	%MB3.7001792

Example:

CANsync interface module on the CANsync master option module in the b maXX's slot G

```
_CANsyncMaster_Init_Slot_G  AT  %MB3.2001792  :
                                     CANsync_PLC_INIT_BMSTRUCT;
```

Where:

`_CANsyncMaster_Init_Slot_G`	is the variable name with the data type short designation "_" for Struct
`CANsync_PLC_BMSTRUCT`	is the data type
`%MB3.2001792`	is the base address of the CANsync interface module on the CANsync master option module in the b maXX's slot G

For cyclic operation, global variables of data type CANsync_PLC_MA_BMSTRUCT are already declared for each option module slot in template "BM4-O-PLC01".

These are the following global variables

```
_CANsyncMaster_Ctrl_Slot_G  AT  %MB3.2001792  :
                                     CANsync_PLC_MA_BMSTRUCT;
_CANsyncMaster_Ctrl_Slot_H  AT  %MB3.3001792  :
                                     CANsync_PLC_MA_BMSTRUCT;
_CANsyncMaster_Ctrl_Slot_J  AT  %MB3.4001792  :
                                     CANsync_PLC_MA_BMSTRUCT;
_CANsyncMaster_Ctrl_Slot_K  AT  %MB3.5001792  :
                                     CANsync_PLC_MA_BMSTRUCT;
_CANsyncMaster_Ctrl_Slot_L  AT  %MB3.6001792  :
                                     CANsync_PLC_MA_BMSTRUCT;
_CANsyncMaster_Ctrl_Slot_M  AT  %MB3.7001792  :
                                     CANsync_PLC_MA_BMSTRUCT;
```

If these variables are no longer in the PROPROG wt II project, create – in dependence on the slot (G to M) – the global variable `_CANsyncMaster_Ctrl_Slot_G` (to `_CANsyncMaster_Ctrl_Slot_M`) of data type `CANsync_PLC_MA_BMSTRUCT`.

At declaration of global variables, this variable must be connected to the base address of the CANsync interface module on the option module.

In dependence on the slot, the base address of the CANsync interface module on the CANsync master option module (BM4-O-CAN-06) is as follows:

Slot	Base address of CANsync interface module
G	%MB3.2001792
H	%MB3.3001792
J	%MB3.4001792
K	%MB3.5001792
L	%MB3.6001792
M	%MB3.7001792

Example:

CANsync interface module on the CANsync master option module in the b maXX's slot G

```
_CANsyncMaster_Ctrl_Slot_G  AT  %MB3.2001792  :
                                    CANsync_PLC_MA_BMSTRUCT;
```

Where:

`_CANsyncMaster_Ctrl_Slot_G`	is the variable name with the data type short designation "_" for Struct
`CANsync_PLC_MA_BMSTRUCT`	is the data type
`%MB3.2001792`	is the base address of the CANsync interface module on the CANsync master option module in the b maXX's slot G

NOTE

In the following tables, the variable name is replaced by an asterisk (*).
It is also assumed that the CANsync master option module for b maXX PLC is fitted in slot G.

This means that you access register `*.w_CPU_CONTROL` via

`_CANsyncMaster_Init_Slot_G.w_CPU_CONTROL`,

you access `*.w_OPTION_STATUS` via

`_CANsyncMaster_Init_Slot_G.w_OPTION_STATUS.`

Where:

`_CANsyncMaster_Init_Slot_G`	is the variable name with the data type short designation "_" for Struct
`w_CPU_CONTROL`	is the control register of the CANsync interface module with data type short designation "w" for WORD

Registers `*.w_CPU_CONTROL` and `*.w_OPTION_STATUS` can also be triggered via the structure for cyclical operation. This makes possible access via

`_CANsyncMaster_Ctrl_Slot_G.w_CPU_CONTROL` and

`_CANsyncMaster_Ctrl_Slot_G.w_OPTION_STATUS.`

Where

`_CANsyncMaster_Ctrl_Slot_G`	is the variable name with the data type short designation "_" for Struct
`w_CPU_CONTROL`	is the status register of the CANsync interface module with data type short designation "w" for WORD

Example of accessing an element of a field that is used in the structure:

According to the table: `*.a_WR_VALUE[3]`

Access: `_CANsyncMaster_Ctrl_Slot_G.a_WR_VALUE[3]`

Where

`_CANsyncMaster_Ctrl_Slot_G`	is the variable name with the data type short designation "_" for Struct
`a_WR_VALUE[3]`	is the register for reference value 3 with the data type short designation "a" for ARRAY. The data type of the elements of the field (of the reference values) is taken from the corresponding table and the description.

Example of accessing an element of a two-dimensional field that is used in the structure:

According to the table: `*.a_RD_VALUE[5][7]`

Access: `_CANsyncMaster_Ctrl_Slot_G.a_RD_VALUE[5][7]`

Where

`_CANsyncMaster_Ctrl_Slot_G`	is the variable name with the data type short designation "_" for Struct
`a_RD_VALUE[5][7]`	is the register for actual value 7 of CANsync slave 5 with the data type short designation "a" for ARRAY. The data type of the elements of the field (of the reference values) is taken from the corresponding table and the description.

Example of accessing an element a (sub) structure, which is itself a field that is used in the structure:

According to the table: `*.a_CFG_RDC_WORD[31].b_CFG_RDC2_WORD3`

Access:
`_CANsyncMaster_Ctrl_Slot_G.a_CFG_RDC_WORD[31].b_CFG_RDC2_WORD3`

Where

`_CANsyncMaster_Ctrl_Slot_G`	is the variable name with the data type short designation "_" for Struct
`a_CFG_RDC_WORD[31]`	is the field containing the configuration data for mapping the words of CANsync slave 31's actual value message frames with the data type short designation "a" for ARRAY
`b_CFG_RDC2_WORD3`	is the register for the configuration data for mapping the third word of actual value message frame 2 (of CANsync slave 31) with the data type short designation "b" for BYTE

Example of accessing an element of a (sub) structure that is used in the structure:

According to the table: `*._CFG_WRC_WORD.b_CFG_WRC1_WORD0`

Access:
`_CANsyncMaster_Ctrl_Slot_G._CFG_WRC_WORD.b_CFG_WRC1_WORD0`

Where

`_CANsyncMaster_Ctrl_Slot_G`	is the variable name with the data type short designation "_" for Struct
`_CFG_WRC_WORD`	is the structure containing the configuration data for mapping the words of the reference value message frames with the data type short designation "_" for STRUCT
`b_CFG_WRC1_WORD0`	is the register for the configuration data for mapping the 0th word of reference value message frame 1 with the data type short designation "b" for BYTE

3.2.2.2 General Registers of the CANsync Interface Module

Register	Contents
*.w_CANsync_STATUS	CANsync status
*.i_SW1_NR	Card software number
*.i_SW1_RELEASE	Software revision incompatible and compatible

CANsync status

Each time the CANsync processor cycle is run through, the system outputs the CANsync status to *.w_CANsync_STATUS.

Meaning:

Bit No.	Meaning (bit = TRUE)
0	Reserved
1	Overrun: A CANsync message could not be received
2	CANsync send buffer is free
3	CANsync send job executed successfully
4	CANsync message currently being received
5	CANsync message currently being sent
6	Error present (warning)
7	CANsync node is deactivated (BUS off)
8-15	Reserved

Software number and software version

In register *.i_SW1_NR, the system displays the number of the CANsync software on the CANsync master option module.

In register *.i_SW1_RELEASE, the system displays the compatible and the incompatible revision of the CANsync software on the CANsync master option module.

3.2.2.3 Initialization

For initialization, global variables of data type CANsync_PLC_INIT_BMSTRUCT are already declared for each option module slot in template "BM4-O-PLC01".

These are the following global variables

```
_CANsyncMaster_Init_Slot_G  AT  %MB3.2001792  :
                                     CANsync_PLC_INIT_BMSTRUCT;
_CANsyncMaster_Init_Slot_H  AT  %MB3.3001792  :
                                     CANsync_PLC_INIT_BMSTRUCT;
_CANsyncMaster_Init_Slot_J  AT  %MB3.4001792  :
                                     CANsync_PLC_INIT_BMSTRUCT;
_CANsyncMaster_Init_Slot_K  AT  %MB3.5001792  :
                                     CANsync_PLC_INIT_BMSTRUCT;
_CANsyncMaster_Init_Slot_L  AT  %MB3.6001792  :
                                     CANsync_PLC_INIT_BMSTRUCT;
_CANsyncMaster_Init_Slot_M  AT  %MB3.7001792  :
                                     CANsync_PLC_INIT_BMSTRUCT;
```

If these variables are no longer in the PROPROG wt II project, create – in dependence on the slot (G to M) – the global variable `_CANsyncMaster_Init_Slot_G` (to `_CANsyncMaster_Init_Slot_M`) of data type `CANsync_PLC_INIT_BMSTRUCT`.

At declaration of global variables, this variable must be connected to the base address of the CANsync interface module on the option module.

In dependence on the slot, the base address of the CANsync interface module on the CANsync master option module (BM4-O-CAN-06) is as follows:

Slot	Base address of CANsync interface module
G	%MB3.2001792
H	%MB3.3001792
J	%MB3.4001792
K	%MB3.5001792
L	%MB3.6001792
M	%MB3.7001792

Example:

CANsync interface module on the CANsync master option module in the b maXX's slot G

```
_CANsyncMaster_Init_Slot_G  AT  %MB3.2001792  :
                                    CANsync_PLC_INIT_BMSTRUCT;
```

Where:

`_CANsyncMaster_Init_Slot_G`	is the variable name with the data type short designation "_" for Struct
`CANsync_PLC_BMSTRUCT`	is the data type
`%MB3.2001792`	is the base address of the CANsync interface module on the CANsync master option module in the b maXX's slot G

In addition, you carry out configuration for synchronous operation:

Meaning		**Register**	**Value**
Baud rate	e.g.: 1 Mbps	*.b_BT_0 *.b_BT_1	16#40 16#34
CANsync interval	e.g. 1 ms	*.b_TIME_PATTERN	16#01
Acceptance Code	All message frames	*.b_AC	16#FF
Acceptance Mask		*.b_AM	16#FF
Output Control	16#FA	*.b_OUTPUT_CONTROL	16#FA
Clock Divider	16#07	*.b_CLOCK_DIVIDER	16#07
Slave/Master	Master	*.b_MA_SL_MODE	16#00
Slave types	CANsync slave with slave number x not available / available	*.a_SL_TYP[x]	16#00 / 16#01

You state the CANsync slave type for each CANsync slave interface module on the CANsync bus (slave number setting using DIP switches). Currently, there is only CANsync slave type 16#01. If the value is set to 16#00, this means that no CANsync slave is present with this slave number or that the CANsync master does not expect one.

You set the operating mode via register *.w_CPU_CONTROL. The system displays the currently active operating mode in register *.w_OPTION_STATUS. You can change the operating mode even after it has been started successfully.

Register	Contents
*.w_CPU_CONTROL	Control register of CANsync interface module
*.w_OPTION_STATUS	Status register of CANsync interface module

(* Corresponds at initialization to
_CANsyncMaster_Init_Slot_G, for example, and after initialization to
_CANsyncMaster_Ctrl_Slot_G), for example

Control register of CANsync interface module	Meaning
16#0000	Cold restart
16#0001	Handshake
16#0002	Take over initialization data
16#0012	Reserved
16#0013	Reserved
16#0020	Start synchronous operating mode
16#0040	Enable active operation
16#0080	(Bit 7 = TRUE) Reset CAN controller

Status register of CANsync inter-face module	Meaning
16#0001	Start up
16#0002	Take over waiting for initialization data
16#0003	Waiting for start
16#0011	Reserved
16#0012	Reserved
16#0013	*Setting up synchronous operation of CANsync slave*
16#0020	*Synchronous operation of CANsync slave is active*
16#0041	Reserved
16#0042	Reserved
16#0043	Setting up synchronous operation of CANsync master
16#0080	Synchronous operation of CANsync master is active

Initialization is carried out using commands 16#0000, 16#0001 and 16#0002 to *.w_CPU_CONTROL. This starts set-up mode. Next, the system reports the slave status of the initialized CANsync slaves (see ▹Command and Response Channel ◃ from page 62 onward). When all the CANsync slaves have reported and have the synchronized status, active operation must be enabled. This is done by setting bit 6 in *.w_CPU_CONTROL (*.w_CPU_CONTROL = 16#0040).

Enabling may be carried out even if not all the CANsync slaves have reported but if the application can administer this.

If you set bit 7 of *.w_CPU_CONTROL (*.w_CPU_CONTROL = 16#0080), the CAN controller is reset and the bit is cleared. This makes it possible to reset the CAN controller's BUS-OFF status and to send and receive CANsync message frames again. The system displays the BUS-OFF status in *.w_CANsync_STATUS (see ▹General Registers of the CANsync Interface Module ◃ from page 47 onward).

The following table lists the registers that can be operated at initialization.

Register	Contents
*.b_MA_SL_MODE	Operating mode: Master/slave (SYNC-OUT/SYNC-IN)
*.b_AC	Acceptance Code of the CANsync controller

Register	Contents
*.b_AM	Acceptance Mask of the CANsync controller
*.b_BT_0	Bit timing register 0 of the CANsync controller
*.b_BT_1	Bit timing register 1 of the CANsync controller
*.b_OUTPUT_CONTROL	Output control register of the CANsync controller
*.b_CLOCK_DIVIDER	Clock divider of the CANsync controller
*.b_TIME_PATTERN	CANsync interval in ms
*.a_SL_TYP[0]	CANsync slave type 0
*.a_SL_TYP[1]	CANsync slave type 1
...	...
*.a_SL_TYP[31]	CANsync slave type 31

(* At initialization, corresponds to structure _CANsync_INIT_Slot_G, for example)

3.2.2.4 Reference values

The system sends the reference values in the CANsync event task.

Register	Contents
*.b_CTRLREG_WRC1	Control register of reference value channel 1
*.b_CTRLREG_WRC2	Control register of reference value channel 2
*.b_CTRLREG_WRC3	Reserved
*.b_CTRLREG_WRC4	Reserved
*.b_CTRLREG_WRC5	Reserved
*.b_CTRLREG_WRC6	Reserved
*.b_CTRLREG_WRC7	Reserved
*.b_CTRLREG_WRC8	Reserved

(* After initialization, corresponds to variable _CANsyncMaster_Ctrl_Slot_G, for example)

In the control register, the system marks with 16#05 the fact that new reference values for the respective reference value message frame (as well as the reference value channel [SWK or WRC]) were entered and that the message frame will be sent. The CANsync interface module acknowledges the command with 16#04.

The reference value message frames are configured in the following registers.

Register	Contents
*._CFG_WRC_WORD.b_CFG_WRC1_WORD0	Configuration of reference value message frame 1 word 0
*._CFG_WRC_WORD.b_CFG_WRC1_WORD1	Configuration of reference value message frame 1 word 1
*._CFG_WRC_WORD.b_CFG_WRC1_WORD2	Configuration of reference value message frame 1 word 2
*._CFG_WRC_WORD.b_CFG_WRC1_WORD3	Configuration of reference value message frame 1 word 3
*._CFG_WRC_WORD.b_CFG_WRC2_WORD0	Configuration of reference value message frame 2 word 0
*._CFG_WRC_WORD.b_CFG_WRC2_WORD1	Configuration of reference value message frame 2 word 1
*._CFG_WRC_WORD.b_CFG_WRC2_WORD2	Configuration of reference value message frame 2 word 2
*._CFG_WRC_WORD.b_CFG_WRC2_WORD3	Configuration of reference value message frame 2 word 3
*._CFG_WRC_WORD.b_CFG_WRC3_WORD0	Reserved

Register	Contents
*._CFG_WRC_WORD.b_CFG_WRC3_WORD1	Reserved
*._CFG_WRC_WORD.b_CFG_WRC3_WORD2	Reserved
*._CFG_WRC_WORD.b_CFG_WRC3_WORD3	Reserved
...	...
*._CFG_WRC_WORD.b_CFG_WRC8_WORD0	Reserved
*._CFG_WRC_WORD.b_CFG_WRC8_WORD1	Reserved
*._CFG_WRC_WORD.b_CFG_WRC8_WORD2	Reserved
*._CFG_WRC_WORD.b_CFG_WRC8_WORD3	Reserved

(* Corresponds, for example, to _CANsyncMaster_Ctrl_Slot_G)

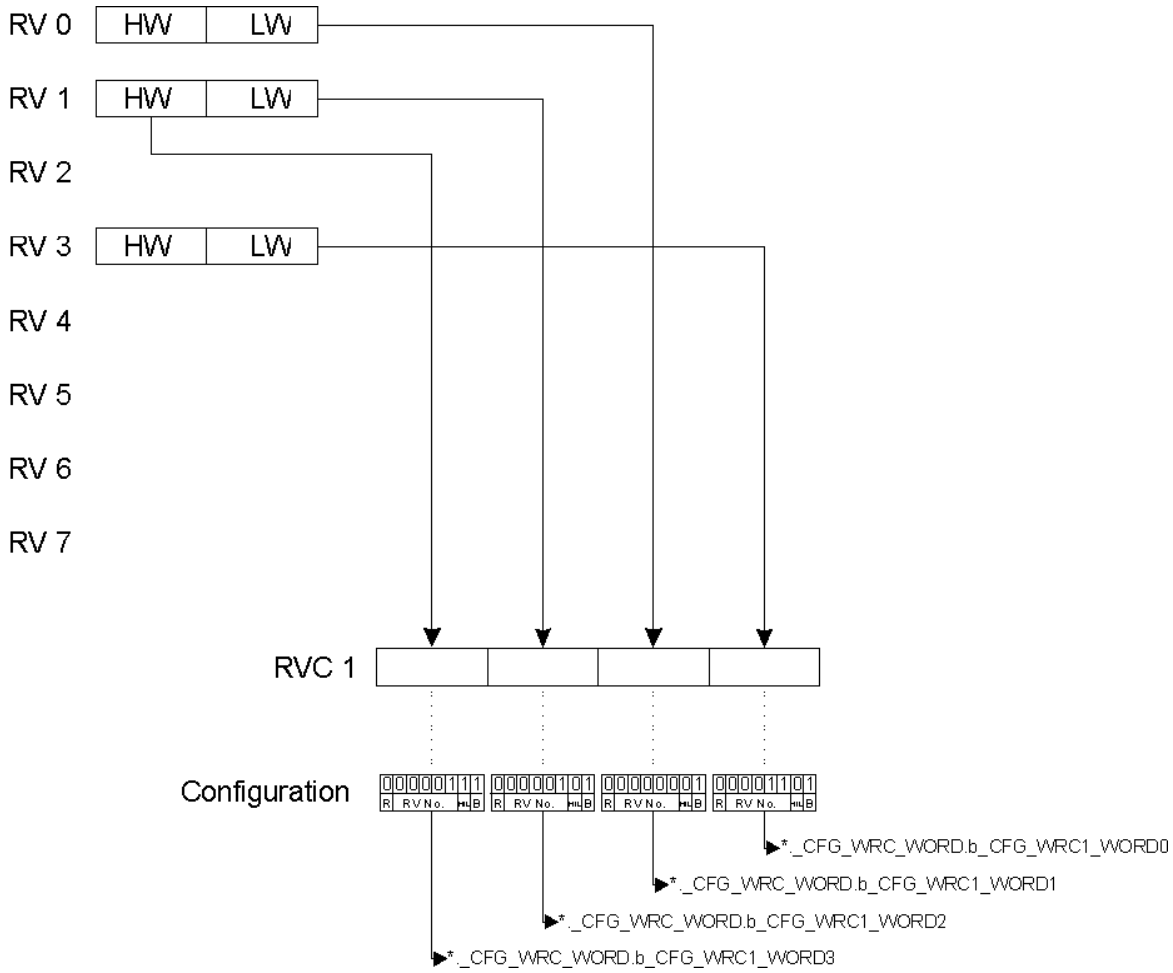
With the configuration, you state which reference value is to be entered at what location (..._WORD0, ..., ..._WORD3) in a reference value message frame (..._WRC1_..., ..._WRC2_...).

	Meaning
Bit 0	Assigned If = 1, the system uses this word of the message frame
Bit 1	Highword/lowword If = 1, the system enters the highword of the reference value
Bit 2	Reference value number Number of reference value 0 to 31
Bit 3	
Bit 4	
Bit 5	
Bit 6	
Bit 7	Reserved, must be set to zero

NOTE

One word of the message is only not used if the following settings are made: reference value number = 0, highword/lowword = 0 and assigned = 0.

Example:



*._CFG_WRC_WORD.b_CFG_WRC1_WORD0 = 16#0D

0	0	0	0	1	1	0	1
R	SV No.					H/L	B

(* Corresponds, for example, to _CANsyncMaster_Ctrl_Slot_G)

This setting enters the lowword of reference value 3 in reference value message frame 1 in word 0.

Register	Contents
*.a_WR_VALUE[0]	Reference value 0
*.a_WR_VALUE[1[	Reference value 1
*.a_WR_VALUE[2]	Specified value 2
*.a_WR_VALUE[3]	Reference value 3
*.a_WR_VALUE[4]	Reference value 4
*.a_WR_VALUE[5[	Reference value 5
*.a_WR_VALUE[6]	Reference value 6
*.a_WR_VALUE[7]	Reference value 7

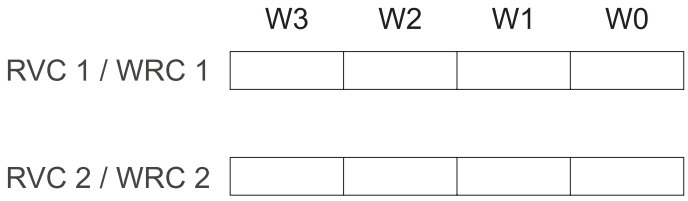
Register	Contents
*.a_WR_VALUE[8]	Reserved
...	...
*.a_WR_VALUE[31]	Reserved

(* Corresponds, for example, to _CANsyncMaster_Ctrl_Slot_G)

The reference values can be word or doubleword reference values.

Explanation of Using the Reference Value Channels

In synchronous operation, two reference value channels (channels 1 and 2; also WRC1 and WRC2) are available. Reference value message frames 1 and 2 are sent on reference value channels 1 and 2.



Both reference value message frames consist of four words each (W0 to W3). After CAN initialization, you must state at least once the assignment of these words. This can be carried out in the initialization program. To do this, enter in areas *._CFG_WRC_WORD.b_CFG_WRC1_WORD0 to *._CFG_WRC_WORD.b_CFG_WRC2_WORD3 WORD3 for each word of the reference value message frames the reference value that is to be transferred at this location. Valid reference value numbers are in the range 0 to 7. For doubleword reference values, you must use two words in the message frame.

You can change the configuration even during active operation. The system applies the change in the next CANsync interval at the latest. At the start of the CANsync interval, the CANsync interface module reads the configuration data.

The system generates reference values in synchronous operation in the CANsync event task. Approximately 70 µs elapse from the falling edge of the SYNC signal to execution of the application program in the CANsync event task. The system must enter the reference values for the reference value message frames by 200 µs after the start of the CANsync event task in communication RAM as the reference value (*.a_WR_VALUE[0] to *.a_WR_VALUE[7]). This is because the CANsync interface module starts generating the reference value message frames at this time. To identify the fact that new reference values have been entered, the system must enter 16#05 in the appropriate control register (*.b_CTRLREG_WRC1 or *.b_CTRLREG_WRC2). This is the enable telling the CANsync interface module that the reference values can be read and that the reference value message frames are being generated. If this enable is not issued by 200 µs after the start of the CANsync event task, reference value message frames are omitted in this CANsync interval. To acknowledge generation of the reference value message frames, the CANsync interface module enters 16#04 in the control registers. This allows users to check whether the system finished generation of the reference value in good time or not. If generation of the reference values takes longer, the system must always generate the reference values for the next start of the CANsync event task. This means that at the beginning of the new CANsync event task the system only needs to copy the precalculated reference values to the corresponding locations in communication RAM.

The reference value message frame is used to request the actual value message frame of a CANsync slave. The number of the CANsync slave interface module is entered in control register actual value request (*.b_CTRLREG_RD_ORDER_RDC1 or *.b_CTRLREG_RD_ORDER_RDC2) (for a further description, see ▷Actual Values ◁ from page 55 onward). Reference value message frame 1 requests actual value message frame 1 and the reference value message frame requests actual value message frame 2.

3.2.2.5 Actual Values

Actual value message frames are requested by a specific piece of information in the identifiers of reference value message frames (see ▷Reference value channel 1 ◁ on page 34).

Register	Contents
*.b_MAX_SL_NR	Maximum slave number (bus address of the CANsync slave

(* Corresponds, for example, to _CANsyncMaster_Ctrl_Slot_G)

This states the highest slave number of the CANsync slave for automatic actual value request (see *.b_CTRLREG_RD_ORDER_RDC1 ff.)

Register	Contents
*.b_CTRLREG_RD_ORDER_RDC1	Control register actual value request of actual value channel 1
*.b_CTRLREG_RD_ORDER_RDC2	Control register actual value request of actual value channel 2
*.b_CTRLREG_RD_ORDER_RDC3	Reserved
*.b_CTRLREG_RD_ORDER_RDC4	Reserved

(* Corresponds, for example, to _CANsyncMaster_CtrlSlot_G)

In the control register, you can state the slave number of the CANsync slave that is to report its actual values in the corresponding actual value message frame. If 16#80 is entered in the control register, the system increments the slave number by 1 in every cycle until the maximum slave number (*.b_MAX_SL_NR) is reached. The system then starts again with slave number 0.

Register	Contents
*.b_STATREG_RDC1	Status register of actual value channel 1
*.b_STATREG_RDC2	Status register of actual value channel 2
*.b_STATREG_RDC3	Reserved
*.b_STATREG_RDC4	Reserved

(* Corresponds, for example, to _CANsyncMaster_Ctrl_Slot_G)

The system enters in the status register the slave number of the CANsync slave from which an actual value message frame was received.

Register	Contents
*.a_STATREG_RDC[0].b_STATREG_RDC1	Actual value acknowledgement of actual value channel 1 of slave 0

Register	Contents
*.a_STATREG_RDC[0].b_STATREG_RDC2	Actual value acknowledgement of actual value channel 2 of slave 0
*.a_STATREG_RDC[0].b_STATREG_RDC3	Reserved
*.a_STATREG_RDC[0].b_STATREG_RDC4	Reserved
*.a_STATREG_RDC[1].b_STATREG_RDC1	Actual value acknowledgement of actual value channel 1 of slave 1
*.a_STATREG_RDC[1].b_STATREG_RDC2	Actual value acknowledgement of actual value channel 2 of slave 1
*.a_STATREG_RDC[1].b_STATREG_RDC3	Reserved
*.a_STATREG_RDC[1].b_STATREG_RDC4	Reserved
*.a_STATREG_RDC[2].b_STATREG_RDC1	Actual value acknowledgement of actual value channel 1 of slave 2
*.a_STATREG_RDC[2].b_STATREG_RDC2	Actual value acknowledgement of actual value channel 2 of slave 2
*.a_STATREG_RDC[2].b_STATREG_RDC3	Reserved
*.a_STATREG_RDC[2].b_STATREG_RDC4	Reserved
...	...
*.a_STATREG_RDC[31].b_STATREG_RDC1	Actual value acknowledgement of actual value channel 1 of slave 31
*.a_STATREG_RDC[31].b_STATREG_RDC2	Actual value acknowledgement of actual value channel 2 of slave 31
*.a_STATREG_RDC[31].b_STATREG_RDC3	Reserved
*.a_STATREG_RDC[31].b_STATREG_RDC4	Reserved

(* Corresponds, for example, to _CANsyncMaster_Ctrl_Slot_G)

When the corresponding actual value message frame for a CANsync slave has arrived, the system enters 16#02 in the actual value acknowledgement register.

Actual value message frames are configured in the following registers.

Register	Contents
*.a_CFG_RDC_WORD[0].b_CFG_RDC1_WORD0	Configuration of actual value message frame 1 word 0 slave 0
*.a_CFG_RDC_WORD[0].b_CFG_RDC1_WORD1	Configuration of actual value message frame 1 word 1 slave 0
*.a_CFG_RDC_WORD[0].b_CFG_RDC1_WORD2	Configuration of actual value message frame 1 word 2 slave 0
*.a_CFG_RDC_WORD[0].b_CFG_RDC1_WORD3	Configuration of actual value message frame 1 word 3 slave 0
*.a_CFG_RDC_WORD[0].b_CFG_RDC2_WORD0	Configuration of actual value message frame 2 word 0 slave 0
*.a_CFG_RDC_WORD[0].b_CFG_RDC2_WORD1	Configuration of actual value message frame 2 word 1 slave 0
*.a_CFG_RDC_WORD[0].b_CFG_RDC2_WORD2	Configuration of actual value message frame 2 word 2 slave 0
*.a_CFG_RDC_WORD[0].b_CFG_RDC2_WORD3	Configuration of actual value message frame 2 word 3 slave 0
*.a_CFG_RDC_WORD[0].b_CFG_RDC3_WORD0	Reserved
*.a_CFG_RDC_WORD[0].b_CFG_RDC3_WORD1	Reserved
*.a_CFG_RDC_WORD[0].b_CFG_RDC3_WORD2	Reserved
*.a_CFG_RDC_WORD[0].b_CFG_RDC3_WORD3	Reserved
*.a_CFG_RDC_WORD[0].b_CFG_RDC4_WORD0	Reserved
*.a_CFG_RDC_WORD[0].b_CFG_RDC4_WORD1	Reserved
*.a_CFG_RDC_WORD[0].b_CFG_RDC4_WORD2	Reserved
*.a_CFG_RDC_WORD[0].b_CFG_RDC4_WORD3	Reserved

Register	Contents
*.a_CFG_RDC_WORD[1].b_CFG_RDC1_WORD0	Configuration of actual value message frame 1 word 0 slave 1
*.a_CFG_RDC_WORD[1].b_CFG_RDC1_WORD1	Configuration of actual value message frame 1 word 1 slave 1
*.a_CFG_RDC_WORD[1].b_CFG_RDC1_WORD2	Configuration of actual value message frame 1 word 2 slave 1
*.a_CFG_RDC_WORD[1].b_CFG_RDC1_WORD3	Configuration of actual value message frame 1 word 3 slave 1
*.a_CFG_RDC_WORD[1].b_CFG_RDC2_WORD0	Configuration of actual value message frame 2 word 0 slave 1
*.a_CFG_RDC_WORD[1].b_CFG_RDC2_WORD1	Configuration of actual value message frame 2 word 1 slave 1
*.a_CFG_RDC_WORD[1].b_CFG_RDC2_WORD2	Configuration of actual value message frame 2 word 2 slave 1
*.a_CFG_RDC_WORD[1].b_CFG_RDC2_WORD3	Configuration of actual value message frame 2 word 3 slave 1
*.a_CFG_RDC_WORD[1].b_CFG_RDC3_WORD0	Reserved
*.a_CFG_RDC_WORD[1].b_CFG_RDC3_WORD1	Reserved
*.a_CFG_RDC_WORD[1].b_CFG_RDC3_WORD2	Reserved
*.a_CFG_RDC_WORD[1].b_CFG_RDC3_WORD3	Reserved
*.a_CFG_RDC_WORD[1].b_CFG_RDC4_WORD0	Reserved
*.a_CFG_RDC_WORD[1].b_CFG_RDC4_WORD1	Reserved
*.a_CFG_RDC_WORD[1].b_CFG_RDC4_WORD2	Reserved
*.a_CFG_RDC_WORD[1].b_CFG_RDC4_WORD3	Reserved
...	...
*.a_CFG_RDC_WORD[31].b_CFG_RDC1_WORD0	Configuration of actual value message frame 1 word 0 slave 31
*.a_CFG_RDC_WORD[31].b_CFG_RDC1_WORD1	Configuration of actual value message frame 1 word 1 slave 31
*.a_CFG_RDC_WORD[31].b_CFG_RDC1_WORD2	Configuration of actual value message frame 1 word 2 slave 31
*.a_CFG_RDC_WORD[31].b_CFG_RDC1_WORD3	Configuration of actual value message frame 1 word 3 slave 31
*.a_CFG_RDC_WORD[31].b_CFG_RDC2_WORD0	Configuration of actual value message frame 2 word 0 slave 31
*.a_CFG_RDC_WORD[31].b_CFG_RDC2_WORD1	Configuration of actual value message frame 2 word 1 slave 31
*.a_CFG_RDC_WORD[31].b_CFG_RDC2_WORD2	Configuration of actual value message frame 2 word 2 slave 31
*.a_CFG_RDC_WORD[31].b_CFG_RDC2_WORD3	Configuration of actual value message frame 2 word 3 slave 31
*.a_CFG_RDC_WORD[31].b_CFG_RDC3_WORD0	Reserved
*.a_CFG_RDC_WORD[31].b_CFG_RDC3_WORD1	Reserved
*.a_CFG_RDC_WORD[31].b_CFG_RDC3_WORD2	Reserved
*.a_CFG_RDC_WORD[31].b_CFG_RDC3_WORD3	Reserved
*.a_CFG_RDC_WORD[31].b_CFG_RDC4_WORD0	Reserved
*.a_CFG_RDC_WORD[31].b_CFG_RDC4_WORD1	Reserved
*.a_CFG_RDC_WORD[31].b_CFG_RDC4_WORD2	Reserved
*.a_CFG_RDC_WORD[31].b_CFG_RDC4_WORD3	Reserved

(* Corresponds, for example, to _CANsyncMaster_Ctrl_Slot_G)

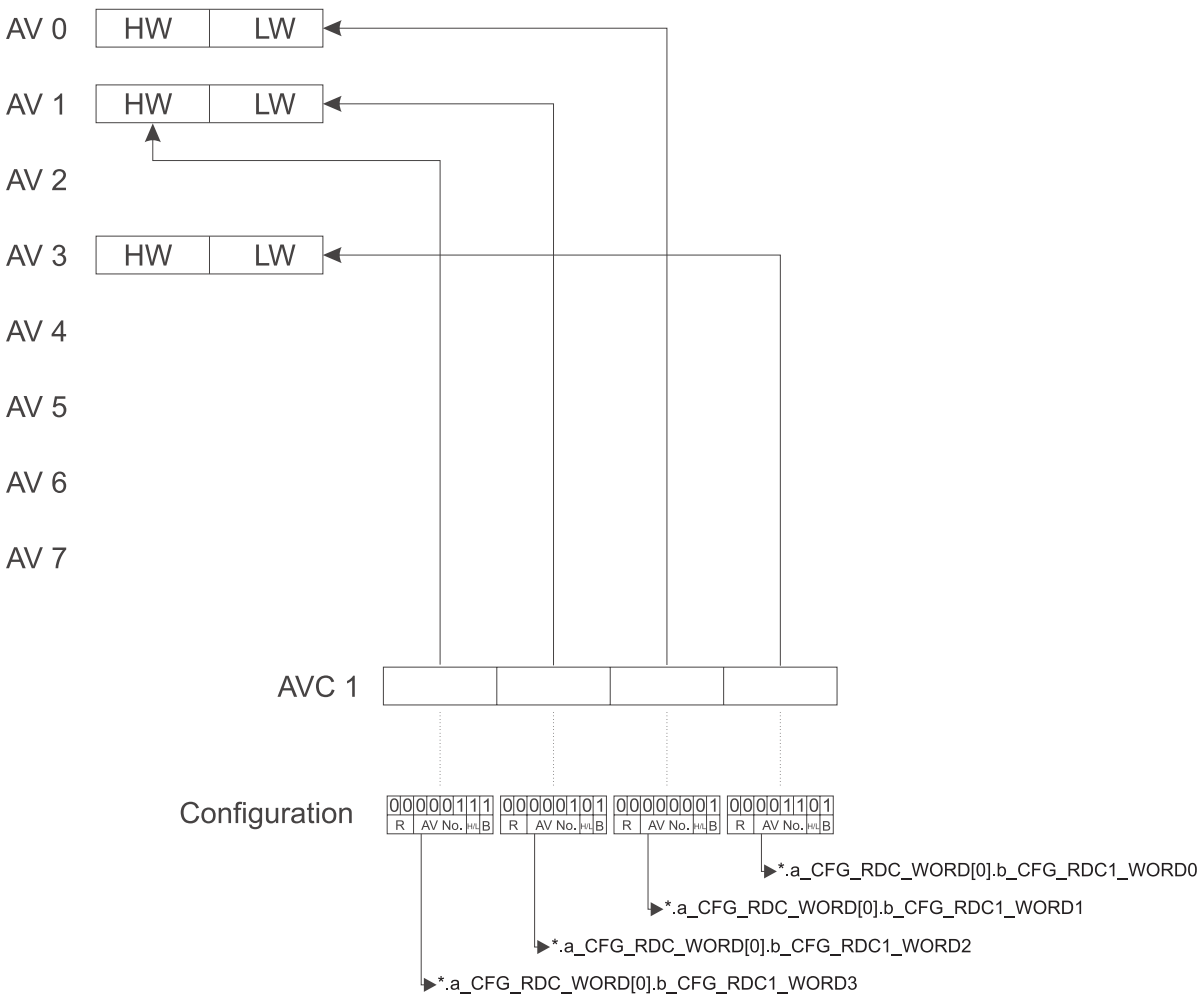
Your configuration tells the system which actual value it is to transfer at which location in the actual value message frame.

	Meaning
Bit 0	Assigned If = 1, the system uses this word of the message frame
Bit 1	Highword/lowword If = 1, the system reads the highword of the actual value
Bit 2	**Actual value number** Number of actual value 0 to 15
Bit 3	
Bit 4	
Bit 5	
Bit 6	Reserved
Bit 7	

NOTE

One word of the message frame is only not used if the following settings are made: actual value number = 0, highword/lowword = 0 and assigned = 0.

Example:



*.a_CFG_RDC_WORD[0].b_CFG_RDC1_WORD0 = 16#0D

0	0	0	0	1	1	0	1
R		AV No.				H/L	B

(* Corresponds, for example, to _CANsyncMaster_Ctrl_Slot_G)

This means that the system reads word 0 from actual value message frame 1 of CANsync slave 0 and writes it to the lowword of actual value 3.

Register	Contents
*.a_RD_BMARRAY[0][0]	Actual value 0 of slave 0
*.a_RD_BMARRAY[0][1]	Actual value 1 of slave 0
*.a_RD_BMARRAY[0][2]	Actual value 2 of slave 0
*.a_RD_BMARRAY[0][3]	Actual value 3 of slave 0
*.a_RD_BMARRAY[0][4]	Actual value 4 of slave 0
*.a_RD_BMARRAY[0][5]	Actual value 5 of slave 0
*.a_RD_BMARRAY[0][6]	Actual value 6 of slave 0

Register	Contents
*.a_RD_BMARRAY[0][7]	Actual value 7 of slave 0
*.a_RD_BMARRAY[0][8]	Reserved
...	...
*.a_RD_BMARRAY[0][15]	Reserved
*.a_RD_BMARRAY[1][0]	Actual value 0 of slave 1
*.a_RD_BMARRAY[1][1]	Actual value 1 of slave 1
*.a_RD_BMARRAY[1][2]	Actual value 2 of slave 1
*.a_RD_BMARRAY[1][3]	Actual value 3 of slave 1
*.a_RD_BMARRAY[1][4]	Actual value 4 of slave 1
*.a_RD_BMARRAY[1][5]	Actual value 5 of slave 1
*.a_RD_BMARRAY[1][6]	Actual value 6 of slave 1
*.a_RD_BMARRAY[1][7]	Actual value 7 of slave 1
*.a_RD_BMARRAY[1][8]	Reserved
...	...
*.a_RD_BMARRAY[1][15]	Reserved
...	...
*.a_RD_BMARRAY[31][0]	Actual value 0 of slave 31
*.a_RD_BMARRAY[31][1]	Actual value 1 of slave 31
*.a_RD_BMARRAY[31][2]	Actual value 2 of slave 31
*.a_RD_BMARRAY[31][3]	Actual value 3 of slave 31
*.a_RD_BMARRAY[31][4]	Actual value 4 of slave 31
*.a_RD_BMARRAY[31][5]	Actual value 5 of slave 31
*.a_RD_BMARRAY[31][6]	Actual value 6 of slave 31
*.a_RD_BMARRAY[31][7]	Actual value 7 of slave 31
*.a_RD_BMARRAY[31][8]	Reserved
...	...
*.a_RD_BMARRAY[31][15]	Reserved

(* Corresponds, for example, to _CANsyncMaster_Ctrl_Slot_G)

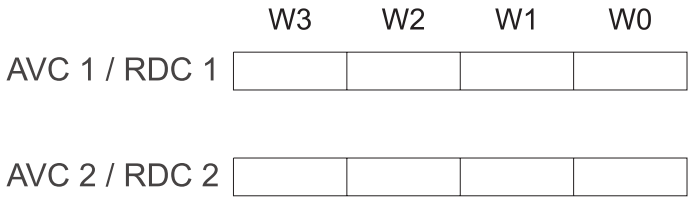
The actual values can be word or doubleword actual values.

NOTE

When displaying _CANsyncMaster_Ctrl_Slot_G in the PROPROG wt II watch window, a period may be between the square brackets.

Explanation of Using the Actual Value Channels

In synchronous operation, two actual value channels (channels 1 and 2; also RDC1 and RDC2) are available. Actual value message frames 1 and 2 are sent on actual value channels 1 and 2.



Both actual value message frames consist of four words each (W0 to W3). After CANsync initialization, you must state at least once the assignment of these words. This can be carried out in the initialization program.
To do this, you must enter in area *.a_CFG_RDC_WORD[0].b_CFG_RDC1_WORD0 to *.a_CFG_RDC_WORD[31].b_CFG_RDC2_WORD3 for each assigned word of the actual value message frames (of each CANsync slave) the actual value that is transferred at this position. Valid actual value numbers are in the range 0 to 7. For doubleword actual values, you must use two words in the message frame.

You can change the configuration even during active operation. The system applies the change in the next CANsync interval at the latest. (The CANsync interface module reads the configuration data when the corresponding actual value message frame is received).

The system carries out actual value message frame evaluation at the start of the CANsync event task. There are two evaluation methods: With the first one, you poll the status registers of the actual value channels (*.b_STATREG_RDC1 or *.b_STATREG_RDC2). The system enters in these registers the slave number of the CANsync slave from which the corresponding actual value message frame was received. Then, you can read out the actual values from registers (*.a_RD_BMARRAY[0][0] to *.a_RD_BMARRAY[31][7]) of this CANsync slave and set the status register to 16#7, for example, to be able to detect correctly the next entry.

With the second method, you poll directly actual value acknowledgement (*.a_STATREG_RDC[0].b_STATREG_RDC1 to *.a_STATREG_RDC[31].b_STATREG_RDC2) for one CANsync slave. In this acknowledgement register, the system marks with 16#02 reception of an actual value message frame. You can then read out the actual values from registers (*.a_RD_BMARRAY[0][0] to *.a_RD_BMARRAY[31][7]) of this CANsync slave. In this case too, the system must then write another value to the status register to detect reentry of the acknowledgement.

You must assign in accordance with the actual value configuration in the application program the actual values that must be read when an actual value message frame has been received.

The system uses the reference value message frame to request a CANsync slave to send its actual value message frame.

The number of the CANsync slave interface module is entered in control register actual value request (*.b_CTRLREG_RD_ORDER_RDC1 and *.b_CTRLREG_RD_ORDER_RDC2). Reference value message frame 1 requests actual value message frame 1, etc.

The request numbers for actual value message frame 1 and actual value message frame 2, etc. can be the same or different too; this means that actual value message frame 1 can be requested by CANsync slave x and actual value message frame 2 can be requested by CANsync slave y.

There are two options for the request number of the CANsync slave: The first one is to state directly the slave number in the control register. While this register is not changed, the system addresses the same CANsync slave in each CANsync interval.

If you enter 16#80 as the request number, the CANsync interface module automatically increments the slave number by 1 in each CANsync interval until the maximum slave number (*.b_MAX_SL_NR) is reached. Polling then starts again at slave number 0. This makes possible automatic requesting of the actual value message frames of all the CANsync slaves that are present. If you assign non-contiguous slave numbers to CANsync slaves, the system generates an actual value request for the CANsync slaves that are not present but this does not lead to any functional problems.

3.2.2.6 Command and Response Channel

Configuring the Command Channel

The command channel assignments are configured in the following registers.

Register	Contents
*.b_SL_NR_CONTROLWORD	Slave number of the CANsync slave for control word command
*.b_SL_NR_ PARAMETER	Slave number of the CANsync slave for parameter word command
*.b_SL_NR_UPDOWN	Slave number of the CANsync slave for the upload/download command

(* Corresponds, for example, to _CANsyncMaster_Ctrl_Slot_G)

Explanation of configuration of the command channel

Using configuration registers *.b_SL_NR_CONTROLWORD, *.b_SL_NR_PARAMETER and *.b_SL_NR_UPDOWN, you can set the use of the command channel. The normal setting is that all three registers are set to 16#80. In this case, the system cyclically increments the slave numbers of the CANsync slaves for which command message frames may be generated. The maximum slave number is the same as for actual value request (*.b_MAX_SL_NR).

As an alternative, you can explicitly specify the slave number. In this case, the system only checks whether the corresponding command should be generated for this CANsync slave.

In any one CANsync interval, it is only possible to send one command. The following scheme indicates the sequence in which the system polls particular areas in synchronous operation and whether an order for generating a commands is entered. The various commands are explained in the next few sections.

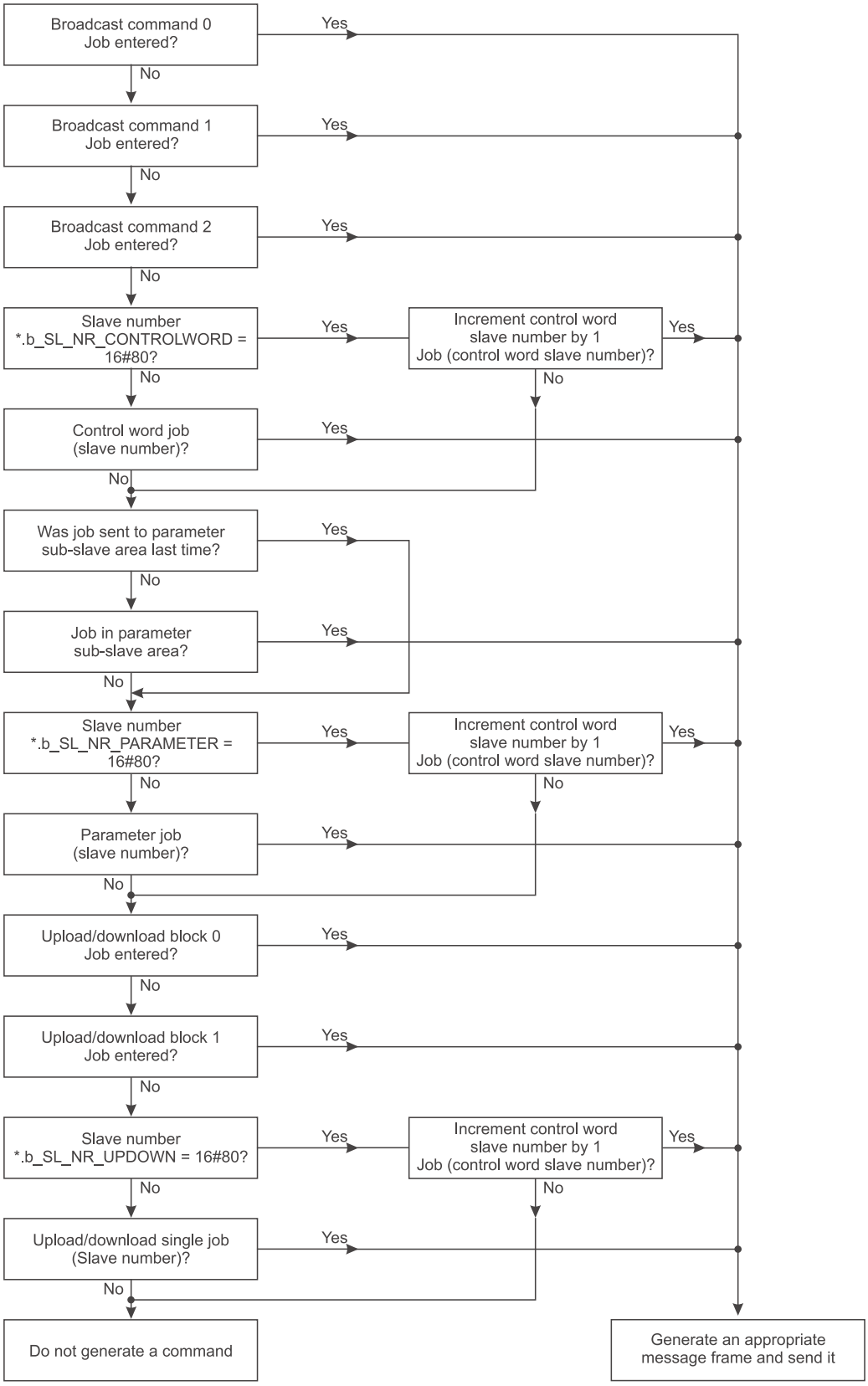


Figure 6: (* After initialization, corresponds to variable _CANsyncMaster_Ctrl_Slot_G, for example)

The scheme shows you which jobs are treated with higher priority (e.g. broadcast commands), which jobs can block other jobs and the number of CANsync intervals at the latest after which the system processes a job. Broadcast command 0 has the highest priority: it is sent immediately in this CANsync interval. The system only checks the next area if no job is entered in this area. The system ignores the pre-event history. This means that if a job is entered in broadcast area 0 in every CANsync interval, no other commands are sent.

Broadcast Command

Broadcast commands can be sent using the following registers:

Register	Contents
*.b_CTRL_REG_BC0	Control register of broadcast command 0
*.d_SL_MASK_BC0	Bit strip of broadcast command 0
*.b_CMD_BC0	Command byte of broadcast command 0
*.b_DATA_BYTE_BC0	Data byte 0 of broadcast command 0
*.w_DATA_WORD_BC0	Data word 1 of broadcast command 0 (DW)

(* Corresponds, for example, to _CANsyncMaster_Ctrl_Slot_G)

Register	Contents
*.b_CTRL_REG_BC1	Control register of broadcast command 1
*.d_SL_MASK_BC1	Bit strip of broadcast command 1
*.b_CMD_BC1	Command byte of broadcast command 1
*.b_DATA_BYTE_BC1	Data byte 0 of broadcast command 1
*.w_DATA_WORD_BC1	Data word 1 of broadcast command 1 (DW)

(* Corresponds, for example, to _CANsyncMaster_Ctrl_Slot_G)

Register	Contents
*.b_CTRL_REG_BC2	Control register of broadcast command 2
*.d_SL_MASK_BC2	Bit strip of broadcast command 2
*.b_CMD_BC2	Command byte of broadcast command 2
*.b_DATA_BYTE_BC2	Data byte 0 of broadcast command 2 (DB)
*.w_DATA_WORD_BC2	Data word 1 of broadcast command 2 (DW)

(* Corresponds, for example, to _CANsyncMaster_Ctrl_Slot_G)

In bit strip *.d_SL_MASK_BC0, *.d_SL_MASK_BC1, *.d_SL_MASK_BC2, you state the CANsync slaves for which the action command is specified. Each CANsync slave has a bit assigned to it (bit 0 = slave 0; bit 1 = slave 1; ... bit 31 = slave 31).

When the bit = TRUE, the associated CANsync slave carries out the command.

In addition to transferring a command, it is also possible to transfer a data byte (*.b_DATA_BYTE...) and a data word (*.w_DATA_WORD...).

To be able to send a command, 16#05 must be entered in the control register. The CANsync interface module then returns 16#04 to confirm sending.

List of Commands

Command byte	Command
16#01	Control word DB = Not used DW = Control word
16#02 - 16#FF	Reserved

Control word command

The control word command is sent using the following registers.

Register	Contents
*.a_STEUREG_CONTROLWORD[0]	Control register of control word for slave 0
*.a_STEUREG_CONTROLWORD[1]	Control register of control word for slave 1
*.a_STEUREG_CONTROLWORD[2]	Control register of control word for slave 2
*.a_STEUREG_CONTROLWORD[3]	Control register of control word for slave 3
...	...
*.a_STEUREG_CONTROLWORD[31]	Control register of control word for slave 31

(* Corresponds, for example, to _CANsyncMaster_Ctrl_Slot_G)

Register	Contents
*.a_CONTROLWORD_SL[0]	Control word for slave 0
*.a_CONTROLWORD_SL[1]	Control word for slave 1
*.a_CONTROLWORD_SL[2]	Control word for slave 2
*.a_CONTROLWORD_SL[3]	Control word for slave 3
...	...
*.a_CONTROLWORD_SL[31]	Control word for slave 31

(* Corresponds, for example, to _CANsyncMaster_Ctrl_Slot_G)

To be able to send a control word to a CANsync slave, the system must enter the control word in the area *.a_CONTROLWORD_SL[0] onwards and then enter 16#05 to the corresponding control register from *.a_STEUREG_CONTROLWORD[0] onwards. When the control word has been sent, the system returns 16#04 as the acknowledgement.

Example: Sending a control word to CANsync slave 3:

1. Enter control word in *.a_CONTROLWORD_SL[3].
2. Enter 16#05 in control register *.a_STEUREG_CONTROLWORD[3].

Acknowledgement after sending: 16#04 to *.a_STEUREG_CONTROLWORD[3].

On the CANsync bus, the control word command is mapped to an action command in which, in the bit mask (bit strip), only the bit for one CANsync slave is set, the command byte is set to 16#01 and the control word is entered as the data word (DW).

Parameter command

You can read and write parameters using the following registers:

Register	Contents
*.a_CTRLREG_PAR_CMD_SL[0].b_STEUREG_PAR_CMD	Control register of parameter command 0
*.a_CTRLREG_PAR_CMD_SL[0].b_STATREG_PAR_CMD	Status register of parameter command 0
*.a_CTRLREG_PAR_CMD_SL[1].b_STEUREG_PAR_CMD	Control register of parameter command 1
*.a_CTRLREG_PAR_CMD_SL[1].b_STATREG_PAR_CMD	Status register of subslave command 1
...	...
*.a_CTRLREG_PAR_CMD_SL[31].b_STEUREG_PAR_CMD	Control register of parameter command 31
*.a_CTRLREG_PAR_CMD_SL[31].b_STATREG_PAR_CMD	Status register of parameter command 31

(* Corresponds, for example, to _CANsyncMaster_Ctrl_Slot_G)

Meanings of the control and status register:

Bit 0	**Active** Must be set to 1 to start the command is set to 0 when the response has been received
Bit 1	**Read** =1: Read parameter
Bit 2	**Write** =1: Write parameter
Bit 3	**Send** =1: Indication that the command has been sent
Bit 4	**Format** 0: Parameter is of word format 1: Parameter is of doubleword format
Bit 5	**Error display** 1: An error has occurred 0: No error
Bit 6	**Busy** = 1 CANsync slave is processing the job, the data is not yet valid
Bit 7	Reserved

Register	Contents
*.a_DATA_PARAMETER[0].d_PAR_VALUE *.a_DATA_PARAMETER[0].w_PAR_NR *.a_DATA_PARAMETER[0].w_SUBSL_ADR	Data of CANsync slave 0 Parameter number 0 Reserved
*.a_DATA_PARAMETER[1].d_PAR_VALUE *.a_DATA_PARAMETER[1].w_PAR_NR *.a_DATA_PARAMETER[1].w_SUBSL_ADR	Data of CANsync slave 1 Parameter number 1 Reserved
...	...
*.a_DATA_PARAMETER[31].d_PAR_VALUE *.a_DATA_PARAMETER[31].w_PAR_NR *.a_DATA_PARAMETER[31].w_SUBSL_ADR	Data of CANsync slave 31 Parameter number 31 Reserved

(* Corresponds, for example, to _CANsyncMaster_Ctrl_Slot_G)

The system enters or receives the parameter value under Data of CANsync slave x. This value can be a word or a doubleword.

The parameter number selects a parameter in the addressed CANsync slave. Refer to the description of the CANsync slave for the meaning of the parameter.

Sequence of a Parameter Access

The parameter commands can be used in the CANsync event task as well as in the rest of the program. To ensure data consistency, the system allocates jobs and evaluates the response via the control register and the status register. To guarantee that the job is carried out without conflicts, first the control register and then the status register must always be read from the application program and if a new value is written, the program must always first set the status register and then the control register. (The control register is the crucial one for the CANsync interface module).

In the following explanation, the addresses refer to CANsync slave 0.

For a parameter write job, the system enters
the **data value** in *.a_DATA_PARAMETER[0].d_PAR_VALUE (doubleword) and
the **parameter number** in *.a_DATA_PARAMETER[0].w_PAR_NR.

After this you enter the value 16#05 in status register *.a_CTRLREG_PAR_CMD_SL[0].b_STATREG_PAR_CMD and in control register *.a_CTRLREG_PAR_CMD_SL[0].b_STEUREG_PAR_CMD if the system is to write a word parameter or 16#15 if it is to write a doubleword parameter.

When the CAN interface module has taken over the job (in accordance with ▹Configuring the Command Channel ◃ from page 62 onward) the system confirms by setting bit 3 (send) in the control register and in the status register.

If the CANsync slave has accepted the job but has not yet completed it, the system sets bit 6 (busy).

If an error occurs in the CANsync slave while it is carrying out the job, the system sets bit 5 (error display), clears bit 6 (busy) and bit 0 (active); and the error number can be read in data range *.a_DATA_PARAMETER[0].d_PAR_VALUE.

When the CANsync slave has completed the job the system sets bit 6 (busy) and clears bit 0 (active); in the byte, this results in 16x04 for write word or 04x14 for write doubleword.

The evaluation in the application program can be limited to polling bit 0; while the bit is set, the system continues to process the job and when it is reset, the job is completed.

It may well be that bit 6 (busy) is not set if the CANsync slave can respond to the job immediately.

The read parameter job runs in a similar way with the only differences being that the command is 16#03 for read word and 16#13 for read doubleword and that the system does not enter the data before the job, but rather that it is available for reading out after the system cleared bit 0 (active). In the byte the acknowledgement then reads 16#02 for word access and 16#12 for doubleword access. The system takes the format from the CANsync slave's response.

Error number of the CANsync slave

Value	Meaning
16#0000	No error occurred
16#FFFF	Error occurred
16#FFFE	Value less than minimum value
16#FFFD	Value greater than maximum value
16#FFFC	Element cannot be changed
16#FFFB	Element not present
16#FFFA	Data is not available (e.g. being processed)
16#FFF9	Error in data format

Upload/Download Block Area

You can control upload and download jobs using the following registers:

Register	Contents
*.b_STEUREG_UPDOWNBLK1	Control register of up/down block 1
*.b_STATREG_UPDOWNBLK1	Status register of up/down block 1
*.b_SL_NR_UPDOWNBLK1	CANsync slave number of up/down block 1
*.w_ERR_NR_UPDOWNBLK1	Error number of up/down block 1
*.w_SUBSL_NR_UPDOWNBLK1	Sub-slave number of up/down block 1
*.d_BASE_ADR_UPDOWNBLK1	Base address of up/down block 1
*.w_LENGTH_UPDOWNBLK1	Length in bytes of up/down block 1
*.w_COUNTER_UPDOWNBLK1	Counter of up/down block 1
*.a_DATA_UPDOWNBLK1[0 to 74]	Data block of up/down block 1 (75 doublewords)

(* Corresponds, for example, to _CANsyncMaster_Ctrl_Slot_G)

Meanings of the control and status register

Bit 0	**Active** Must be set to 1 to start the command is set to 0 when the command has been processed
Bit 1	**Send** =1: Indication that the command has been sent
Bit 2 Bit 3	**Mode** Bit3 Bit2 0 0: reserved 0 1: initialization 1 0: ongoing upload/download 1 1: End of block
Bit 4	**Upload/download** 0: Upload 1: Download

Bit 5	**Error display** 1: An error has occurred 0: No error
Bit 6	**Busy** = 1 CANsync slave is processing the job, the data is not yet valid
Bit 7	**Reset** Cancellation of a job

Sequence of an Upload/Download Job in Block 1

The upload/download commands can be used in the CANsync event task as well as in the rest of the program. To ensure data consistency, the system allocates jobs and evaluates the response via the control register and the status register. To guarantee that the job is carried out without conflicts, first the control register and then the status register must always be read from the application program and if a new value is written, the program must always first set the status register and then the control register. (The control register is the crucial one for the CANsync interface module).

For a download, the system first fills data block area *.a_DATA_UPDOWNBLK1[0] to *.a_DATA_UPDOWNBLK1[74] with the data to be transferred. The maximum block length that can be sent with one download job is 300 bytes (75 doublewords).

The number of the CANsync slaves that is to receive the download is entered in *.b_SL_NR_UPDOWNBLK1 and the base address is entered in *.d_BASE_ADR_UPDOWNBLK1.

Base addresses 16#0000_0000 to 16#0000_00FF are reserved for operating system jobs.

After this, you enter the value 16#15 in status register *.b_STATREG_UPDOWNBLK1 and in control register *.b_STEUREG_UPDOWNBLK1.

The CANsync interface module will now try to transfer the entire block to the CANsync slave. The system automatically generates the sequential modes. The progress can be read off the byte counter in *.w_COUNTER_UPDOWNBLK1.

When the CANsync interface module has taken over the job (see ▹Configuring the Command Channel ◃ from page 62 onward), the system confirms by setting bit 1 (send) in the control register and in the status register.

If the CANsync slave has accepted the job but has not yet completed it, the system sets bit 6 (busy).

If an error occurs in the CANsync slave while it is carrying out the job, the system sets bit 5 (error display), clears bit 6 (busy) and bit 0 (active); and the error number can be read in *.b_SL_NR_UPDOWNBLK1.

When the CANsync slave has completed the job the system sets bit 6 (busy) and clears bit 0 (active); in the control and status register, this results in 16#1C.

The evaluation in the application program can be limited to polling bit 0 (active). While the bit is set, the system continues to process the job and when it is reset, the job is completed.

It may well be that bit 6 (busy) is not set if the CANsync slave can respond to the individual modes of the job immediately.

The upload job runs in a similar way with the only differences being that the command is 16#05 and that the data is available for reading out after bit 0 (active) has been cleared rather than before the job has been entered. The acknowledgement in the control and status register then reads 16#0C.

You can cancel an upload/download job by setting bit 7 (reset) to TRUE. The system then sends to the CANsync slave an upload initialization message frame with address = 0 and length = 0.

Upload/Download Error Numbers

Error number	Meaning
16#0001	CANsync slave acknowledges wrong block number
16#0002	Entered length greater than 300 bytes
...	
16#0100	CANsync slave expects block with the number that is entered in the counter
16#0101	CANsync slave expects block end
16#0102	CANsync slave does not yet expect block end
16#0103	CANsync slave cancels upload/download
16#0104	Upload/download not possible
16#0105	Base address not allowed
16#0106	Reserved
16#0107	Block length > CANsync slave's maximum block length
16#0108	Message frame mode error (mode not allowed at this stage)

3.2.2.7 CANsync Slave Status

Register	Contents
*.a_STAT_SL[0]	CANsync slave status 0
*.a_STAT_SL[1]	CANsync slave status 1
*.a_STAT_SL[2]	CANsync slave status 2
...	...
*.a_STAT_SL[31]	CANsync slave status 31

(* Corresponds, for example, to _CANsyncMaster_Ctrl_Slot_G)

Meaning of CANsync slave status:

Bit 0	**Response** = 1: CANsync slave responds
Bit 1	**Synchronized** = 1: CANsync slave is synchronized

Bit 2	Reserved
Bit 3	
Bit 4	
Bit 5	
Bit 6	
Bit 7	

In set-up mode (see ▹Initialization ◃ from page 48 onward), the system polls the slave status of all the initialized CANsync slaves and enters it here. Bit 0 (response) indicates that the CANsync slave has logged on to the CANsync bus. Bit 1 (synchronized) indicates that the CANsync slave is synchronized and that synchronous operation can be started.

3.2.3 Register structure and function of the CANsync slave option module for b maXX PLC

Below, we will go into the register structure of communication RAM in the CANsync slave option module for b maXX PLC.

To allow you to access registers of the communication RAM in the PROPROG wt II project, data types are defined that map the register structure. The system uses these data types to declare variables that are assigned to the CANsync interface module's base address.

After this, it is possible to access the registers of communication RAM via the structure elements of the declared variables.

At initialization of the CANsync slave interface module, the registers in communication RAM have a different meaning than after initialization in cyclical operation.

This means that, for initialization there is the

CANsync_PLC_INIT_BMSTRUCT

and for cyclical operation, the

CANsync_PLC_SL_BMSTRUCT

These structures are defined from library BM_TYPES_20bd03 onwards. After you have integrated library BM_TYPES_20bd03 in the project, the data types are available.

These structures contain

- 8-bit elements,
- 16-bit elements,
- 32-bit elements,
- Structures from the elements mentioned above
- Fields (ARRAY) and structures from the elements and structures mentioned above

Short designations have been prepended to the data types (8-, 16-, 32-bit elements, structures and fields) that are used in a structure. This is for the sake of clarity when using the structures in programming.

Data type	Short designation	Number of bits
BYTE	b	8
WORD	w	16
DWORD (double word)	d	32
SINT (short integer)	si	8
DINT (double integer)	di	32
USINT (unsigned short integer)	us	8
UINT (unsigned integer)	u	16
UDINT (unsigned double integer)	ud	32
STRUCT	_ (underline)	-
ARRAY	a	-

Other data types that are not used in the structures include:

Data type	Short designation	Number of bits
BOOL (bit)	x	1
TIME	t	-

3.2.3.1 Explanation of declaring the global variables

For initialization, global variables of data type CANsync_PLC_INIT_BMSTRUCT are already declared for each option module slot in template "BM4-O-PLC01".

These are the following global variables

```
_CANsyncSlave_Init_Slot_G  AT  %MB3.2001792  :
                                    CANsync_PLC_INIT_BMSTRUCT;
_CANsyncSlave_Init_Slot_H  AT  %MB3.3001792  :
                                    CANsync_PLC_INIT_BMSTRUCT;
_CANsyncSlave_Init_Slot_J  AT  %MB3.4001792  :
                                    CANsync_PLC_INIT_BMSTRUCT;
_CANsyncSlave_Init_Slot_K  AT  %MB3.5001792  :
                                    CANsync_PLC_INIT_BMSTRUCT;
_CANsyncSlave_Init_Slot_L  AT  %MB3.6001792  :
                                    CANsync_PLC_INIT_BMSTRUCT;
_CANsyncSlave_Init_Slot_M  AT  %MB3.7001792  :
                                    CANsync_PLC_INIT_BMSTRUCT;
```

If these variables are no longer in the PROPROG wt II project, create – in dependence on the slot (G to M) – the global variable `_CANsyncSlave_Init_Slot_G` (to `_CANsyncSlave_Init_Slot_`M) of data type `CANsync_PLC_INIT_BMSTRUCT`.

At declaration of global variables, this variable must be connected to the base address of the CANsync interface module on the option module.

In dependence on the slot, the base address of the CANsync interface module on the CANsync slave option module for b maXX PLC (BM4-O-CAN-05, DIP switch 9 = ON) is as follows:

Slot	Base address of CAN-sync interface module
G	%MB3.2001792
H	%MB3.3001792
J	%MB3.4001792
K	%MB3.5001792
L	%MB3.6001792
M	%MB3.7001792

Example:

CANsync interface module on CANsync slave option module for b maXX PLC in the b maXX's slot G

```
_CANsyncSlave_Init_Slot_G  AT  %MB3.2001792  :
                                    CANsync_PLC_INIT_BMSTRUCT;
```

Where:

`_CANsyncSlave_Init_Slot_G`	is the variable name with the data type short designation "_" for Struct
`CANsync_PLC_BMSTRUCT`	is the data type
`%MB3.2001792`	is the base address of the CANsync interface module on the CANsync slave option module for b maXX PLC in the b maXX's slot G

For cyclic operation, global variables of data type CANsync_PLC_SL_BMSTRUCT are already declared for each option module slot in template "BM4-O-PLC01".

These are the following global variables

```
_CANsyncSlave_Ctrl_Slot_G  AT  %MB3.2001792  :
                                    CANsync_PLC_SL_BMSTRUCT;
_CANsyncSlave_Ctrl_Slot_H  AT  %MB3.3001792  :
                                    CANsync_PLC_SL_BMSTRUCT;
_CANsyncSlave_Ctrl_Slot_J  AT  %MB3.4001792  :
                                    CANsync_PLC_SL_BMSTRUCT;
_CANsyncSlave_Ctrl_Slot_K  AT  %MB3.5001792  :
                                    CANsync_PLC_SL_BMSTRUCT;
_CANsyncSlave_Ctrl_Slot_L  AT  %MB3.6001792  :
                                    CANsync_PLC_SL_BMSTRUCT;
_CANsyncSlave_Ctrl_Slot_M  AT  %MB3.7001792  :
                                    CANsync_PLC_SL_BMSTRUCT;
```

If these variables are no longer in the PROPROG wt II project, create – in dependence on the slot (G to M) – the global variable `_CANsyncSlave_Ctrl_Slot_G` (to `_CANsyncSlave_Ctrl_Slot_M`) of data type `CANsync_PLC_SL_BMSTRUCT`.

At declaration of global variables, this variable must be connected to the base address of the CANsync interface module on the option module.

In dependence on the slot, the base address of the CANsync interface module on the CANsync slave option module for b maXX PLC (BM4-O-CAN-05, DIP switch 9 = ON) is as follows:

Slot	Base address of CANsync interface module
G	%MB3.2001792
H	%MB3.3001792
J	%MB3.4001792
K	%MB3.5001792
L	%MB3.6001792
M	%MB3.7001792

Example:

CANsync interface module on the CANsync slave option module in the b maXX's slot G

```
_CANsyncSlave_Ctrl_Slot_G  AT  %MB3.2001792  :
                                    CANsync_PLC_SL_BMSTRUCT;
```

Where:

`_CANsyncSlave_Ctrl_Slot_G`	is the variable name with the data type short designation "_" for Struct
`CANsync_PLC_SL_BMSTRUCT`	is the data type
`%MB3.2001792`	is the base address of the CANsync interface module on the CANsync slave option module in the b maXX's slot G

NOTE

In the following tables, the variable name is replaced by an asterisk (*).
It is also assumed that the CANsync slave option module for b maXX PLC is fitted in slot G.

This means that you access register `*.w_CPU_CONTROL` via

`_CANsyncSlave_Init_Slot_G.w_CPU_CONTROL` zu,

you access `*.w_OPTION_STATUS` via

`_CANsyncSlave_Init_Slot_G.w_OPTION_STATUS` zu.

Where:

`_CANsyncSlave_Init_Slot_G`	is the variable name with the data type short designation "_" for Struct
`w_CPU_CONTROL`	is the control register of the CANsync interface module with data type short designation "w" for WORD

Registers *.w_CPU_CONTROL and *.w_OPTION_STATUS can also be triggered via the structure for cyclical operation. This makes possible access via

_CANsyncSlave_Ctrl_Slot_G.w_CPU_CONTROL and
_CANsyncSlave_Ctrl_Slot_G.w_OPTION_STATUS.

Where

_CANsyncSlave_Ctrl_Slot_G	is the variable name with the data type short designation "_" for STRUCT
w_CPU_CONTROL	is the status register of the CANsync interface module with data type short designation "w" for WORD

Example of accessing an element of a field that is used in the structure:

According to the table: *.a_WR_VALUE_RECEIVE[3]

Access: _CANsyncSlave_Ctrl_Slot_G.a_WR_VALUE_RECEIVE[3]

Where

_CANsyncSlave_Ctrl_Slot_G	is the variable name with the data type short designation "_" for STRUCT
a_WR_VALUE_RECEIVE[3]	is the register for reference value 3 with the data type short designation "a" for ARRAY. The data type of the elements of the field (of the reference values) is taken from the corresponding table and the description.

Example of accessing an element of a two-dimensional field that is used in the structure:

According to the table: *.a_RD_VALUE[5][7]

Access: _CANsyncSlave_Ctrl_Slot_G.a_RD_VALUE[5][7]

Where

_CANsyncSlave_Ctrl_Slot_G	is the variable name with the data type short designation "_" for STRUCT
a_RD_VALUE[5][7]	is the register for actual value 7 of CANsync slave 5 with the data type short designation "a" for ARRAY. The data type of the elements of the field (of the reference values) is taken from the corresponding table and the description.

Example of accessing an element a (sub) structure, which is itself a field that is used in the structure:

According to the table: *.a_CFG_RDC_WORD[31].b_CFG_RDC2_WORD3

Access:
_CANsyncSlave_Ctrl_Slot_G.a_CFG_RDC_WORD[31].b_CFG_RDC2_WORD3

Where

`_CANsyncSlave_Ctrl_Slot_G`	is the variable name with the data type short designation "_" for STRUCT
`a_CFG_RDC_WORD[31]`	is the field containing the configuration data for mapping the words of CANsync slave 31's actual value message frames with the data type short designation "a" for ARRAY
`b_CFG_RDC2_WORD3`	is the register for the configuration data for mapping the third word of actual value message frame 2 (of CANsync slave 31) with the data type short designation "b" for BYTE

Example of accessing an element of a (sub) structure that is used in the structure:

According to the table: `*._CFG_WRC_WORD.b_CFG_WRC1_WORD0`

Access: `_CANsyncSlave_Ctrl_Slot_G._CFG_WRC_WORD.b_CFG_WRC1_WORD0`

Where

`_CANsyncSlave_Ctrl_Slot_G`	is the variable name with the data type short designation "_" for STRUCT
`_CFG_WRC_WORD`	is the structure containing the configuration data for mapping the words of the reference value message frames with the data type short designation "_" for STRUCT
`b_CFG_WRC1_WORD0`	is the register for the configuration data for mapping the 0th word of reference value message frame 1 with the data type short designation "b" for BYTE

3.2.3.2 General Registers of the CANsync Interface Module

Register	Contents
*.w_CANsync_STATUS	CANsync status
*.w_SLAVE_NUMBER	The slave number set using a DIP switch
*.i_SW1_NR	Card software number
*.i_SW1_RELEASE	Software revision incompatible and compatible

CANsync Status

Each time the CANsync processor cycle is run through, the system outputs the CANsync status to *.w_CANsync_STATUS.

Meaning:

Bit No.	Meaning (bit = TRUE)
0	Reserved
1	Overrun: A CANsync message could not be received
2	CANsync send buffer is free
3	CANsync send job executed successfully
4	CANsync message currently being received
5	CANsync message currently being sent
6	Error present (warning)
7	CANsync node is deactivated (BUS off)
8-15	Reserved

Slave number

In register *.w_SLAVE_NUMBER, the system displays the slave number that you set using the DIP switches (see ▷Operating Instructions of the CANsync slave Option Module for b maXX◁).

Software number and software version

In register *.i_SW1_NR, the system displays the number of the CANsync software on the CANsync slave option module for b maXX PLC.

In register *.i_SW1_RELEASE, the system displays the compatible and the incompatible revision of the CANsync software on the CANsync slave option module for b maXX PLC.

3.2.3.3 Initialization

For initialization, global variables of data type CANsync_PLC_INIT_BMSTRUCT are already declared for each option module slot in template "BM4-O-PLC01".

These are the following global variables

```
_CANsyncSlave_Init_Slot_G  AT  %MB3.2001792  :
                                    CANsync_PLC_INIT_BMSTRUCT;
_CANsyncSlave_Init_Slot_H  AT  %MB3.3001792  :
                                    CANsync_PLC_INIT_BMSTRUCT;
_CANsyncSlave_Init_Slot_J  AT  %MB3.4001792  :
                                    CANsync_PLC_INIT_BMSTRUCT;
_CANsyncSlave_Init_Slot_K  AT  %MB3.5001792  :
                                    CANsync_PLC_INIT_BMSTRUCT;
_CANsyncSlave_Init_Slot_L  AT  %MB3.6001792  :
                                    CANsync_PLC_INIT_BMSTRUCT;
```

```
_CANsyncSlave_Init_Slot_M  AT  %MB3.7001792  :
                                    CANsync_PLC_INIT_BMSTRUCT;
```

If these variables are no longer in the PROPROG wt II project, create – in dependence on the slot (G to M) – the global variable `_CANsyncSlave_Init_Slot_G` (to `_CANsyncSlave_Init_Slot_M`) of data type `CANsync_PLC_INIT_BMSTRUCT`.

At declaration of global variables, this variable must be connected to the base address of the CANsync interface module on the option module.

In dependence on the slot, the base address of the CANsync interface module on the CANsync slave option module for b maXX PLC (BM4-O-CAN-05, DIP switch 9 = ON) is as follows:

Slot	Base address of CAN-sync interface module
G	%MB3.2001792
H	%MB3.3001792
J	%MB3.4001792
K	%MB3.5001792
L	%MB3.6001792
M	%MB3.7001792

Example:

CANsync interface module on CANsync slave option module for b maXX PLC in the b maXX's slot G

```
_CANsyncSlave_Init_Slot_G  AT  %MB3.2001792  :
                                    CANsync_PLC_INIT_BMSTRUCT;
```

Where:

`_CANsyncSlave_Init_Slot_G`	is the variable name with the data type short designation "_" for Struct
`CANsync_PLC_BMSTRUCT`	is the data type
`%MB3.2001792`	is the base address of the CANsync interface module on the CANsync slave option module for b maXX PLC in the b maXX's slot G

In addition, you carry out configuration for synchronous operation:

Meaning		Register	Value
Baud rate	e.g.: 1 Mbps	*.b_BT_0 *.b_BT_1	16#40 16#34
CANsync interval	e.g. 1 ms	*.b_TIME_PATTERN	16#01
Acceptance Code	All message frames	*.b_AC	16#FF
Acceptance Mask		*.b_AM	16#FF

Meaning		Register	Value
Output Control	16#FA	*.b_OUTPUT_CONTROL	16#FA
Clock Divider	16#07	*.b_CLOCK_DIVIDER	16#07
Slave/Master	Slave	*.b_MA_SL_MODE	16#01

You set the operating mode via register *.w_CPU_CONTROL. The system displays the currently active operating mode in register *.w_OPTION_STATUS. You can change the operating mode even after it has been started successfully.

Register	Contents
*.w_CPU_CONTROL	Control register of CANsync interface module
*.w_OPTION_STATUS	Status register of CANsync interface module

(* Corresponds at initialization to _CANsyncSlave_Init_Slot_G, for example, and after initialization to _CANsyncSlave_Ctrl_Slot_G), for example

Control register of CANsync interface module	Meaning
16#0000	Cold restart
16#0001	Test Handshake
16#0002	Take over initialization data
16#0012	Reserved
16#0013	Reserved
16#0020	Start synchronous operating mode
16#0040	Enable active operation
16#0080	(Bit 7 = TRUE) Reset CAN controller

Status register of CANsync interface module	Meaning
16#0001	Start up
16#0002	Take over waiting for initialization data
16#0003	Waiting for start
16#0011	Reserved
16#0012	Reserved
16#0013	Setting up synchronous operation of CANsync slave
16#0020	Synchronous operation of CANsync slave is active
16#0041	Reserved
16#0042	Reserved
16#0043	*Setting up synchronous operation of CANsync master*
16#0080	*Synchronous operation of CANsync master is active*

Initialization is carried out using commands 16#0000, 16#0001 and 16#0002 to *.w_CPU_CONTROL. This starts set-up mode. After this, you can enable active operation. This is done by setting bit 6 in *.w_CPU_CONTROL (*.w_CPU_CONTROL = 16#0040).

If you set bit 7 of *.w_CPU_CONTROL, the CANsync controller is reset and the bit is cleared. This makes it possible to reset the CANsync controller's BUS-OFF status and to send and receive CANsync message frames again. The system displays the BUS-OFF status in *.w_CANsync_STATUS (see ▹General Registers of the CANsync Interface Module ◃ from page 76 onward).

The following table lists the registers that can be operated at initialization.

Register	Contents
*.b_MA_SL_MODE	Operating mode: Master/slave (SYNC-OUT/SYNC-IN)
*.b_AC	Acceptance Code of the CANsync controller
*.b_AM	Acceptance Mask of the CANsync controller
*.b_BT_0	Bit timing register 0 of the CANsync controller
*.b_BT_1	Bit timing register 1 of the CANsync controller
*.b_OUTPUT_CONTROL	Output control register of the CANsync controller
*.b_CLOCK_DIVIDER	Clock divider of the CANsync controller
*.b_TIME_PATTERN	CANsync interval in ms

(* At initialization, corresponds to structure _CANsyncSlave_INIT_Slot_G, for example)

3.2.3.4 Reference values

Reference values are received in the CANsync event task.

Register	Contents
*.b_CTRLREG_WRC1	Status register of reference value channel 1
*.b_CTRLREG_WRC2	Status register of reference value channel 2
*.b_CTRLREG_WRC3	Reserved
*.b_CTRLREG_WRC4	Reserved

(* Corresponds, for example, to _CANsyncSlave_Ctrl_Slot_G)

In the status register of the reference value channels, 16#02 is used to report that the respective reference value message frame was received.

Reference value message frames are configured using the following registers.

Register	Contents
*._CFG_WRC_WORD.b_CFG_WRC1_WORD0	Configuration of reference value message frame 1 word 0
*._CFG_WRC_WORD.b_CFG_WRC1_WORD1	Configuration of reference value message frame 1 word 1
*._CFG_WRC_WORD.b_CFG_WRC1_WORD2	Configuration of reference value message frame 1 word 2
*._CFG_WRC_WORD.b_CFG_WRC1_WORD3	Configuration of reference value message frame 1 word 3
*._CFG_WRC_WORD.b_CFG_WRC2_WORD0	Configuration of reference value message frame 2 word 0
*._CFG_WRC_WORD.b_CFG_WRC2_WORD1	Configuration of reference value message frame 2 word 1
*._CFG_WRC_WORD.b_CFG_WRC2_WORD2	Configuration of reference value message frame 2 word 2
*._CFG_WRC_WORD.b_CFG_WRC2_WORD3	Configuration of reference value message frame 2 word 3
*._CFG_WRC_WORD.b_CFG_WRC3_WORD0	Reserved
*._CFG_WRC_WORD.b_CFG_WRC3_WORD1	Reserved
*._CFG_WRC_WORD.b_CFG_WRC3_WORD2	Reserved

*._CFG_WRC_WORD.b_CFG_WRC3_WORD3	Reserved
*._CFG_WRC_WORD.b_CFG_WRC4_WORD0	Reserved
*._CFG_WRC_WORD.b_CFG_WRC4_WORD1	Reserved
*._CFG_WRC_WORD.b_CFG_WRC4_WORD2	Reserved
*._CFG_WRC_WORD.b_CFG_WRC4_WORD3	Reserved

(* Corresponds, for example, to _CANsyncSlave_Ctrl_Slot_G)

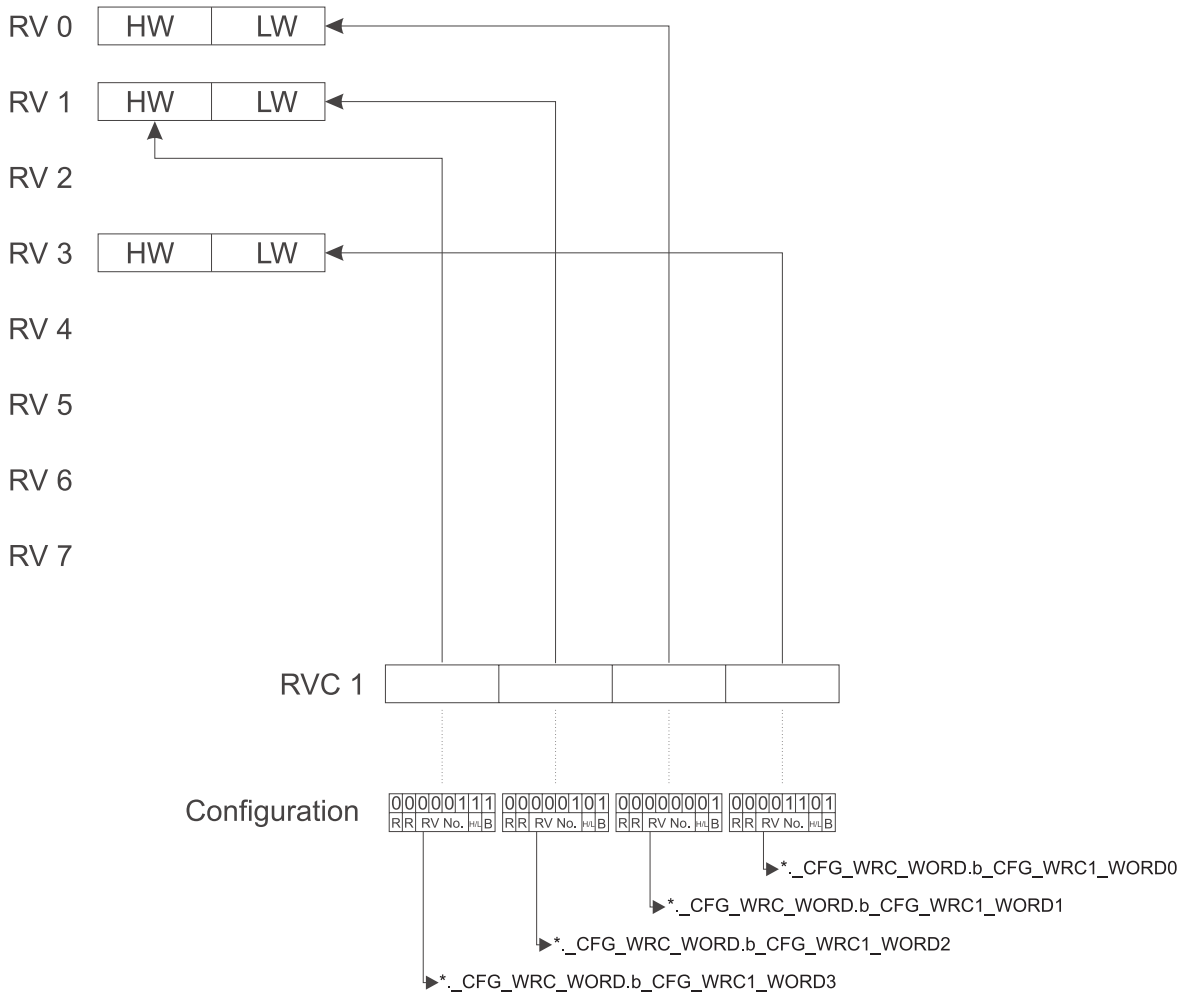
With the configuration, you state which reference value is to be read at what location (..._WORD0, ..., ..._WORD3) in the reference value message frame (..._WRC1_..., ..._WRC2_...).

	Meaning
Bit 0	**Assigned** If = 1, the system uses this word of the message frame
Bit 1	**Highword/lowword** If = 1, the system enters the highword of the reference value
Bit 2	**Reference value number** Number of reference value 0 to 15
Bit 3	
Bit 4	
Bit 5	
Bit 6	Reserved
Bit 7	

NOTE

One word of the message frame is only not used if the following settings are made: reference value number = 0, highword/lowword = 0 and assigned = 0.

Example:



*._CFG_WRC_WORD.b_CFG_WRC1_WORD0 = 16#0D

0	0	0	0	1	1	0	1
R	R	SV No.				H/L	B

(* Corresponds, for example, to _CANsyncSlave_Ctrl_Slot_G)

This setting enters word 0 of reference value message frame 1 as the lowword of reference value 3.

Register	Contents
*.a_WR_VALUE_RECEIVE[0]	Reference value 0
*.a_WR_VALUE_RECEIVE[1]	Reference value 1
*.a_WR_VALUE_RECEIVE[2]	Reference value 2
*.a_WR_VALUE_RECEIVE[3]	Reference value 3
*.a_WR_VALUE_RECEIVE[4]	Reference value 4
*.a_WR_VALUE_RECEIVE[5]	Reference value 5
*.a_WR_VALUE_RECEIVE[6]	Reference value 6

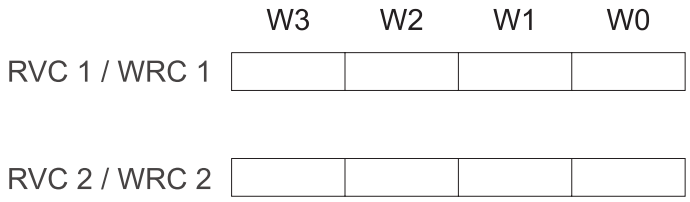
Register	Contents
*.a_WR_VALUE_RECEIVE[7]	Reference value 7
*.a_WR_VALUE_RECEIVE[8]	Reserved
...	...
*.d_WR_VALUE_RECEIVE[15]	Reserved

(* Corresponds, for example, to _CANsyncSlave_Ctrl_Slot_G)

The reference values can be word or doubleword reference values.

Explanation of Using the Reference Value Channels

In synchronous operation, two reference value channels (channels 1 and 2; also WRC1 and WRC2) are available. Reference value message frames 1 and 2 of the CANsync master are received on reference value channels 1 and 2.



Both reference value message frames consist of four words each (W0 to W3). After CANsync initialization, you must state at least once the assignment of these words.

This can be carried out in the initialization program. To do this, enter in areas
*._CFG_WRC_WORD.b_CFG_WRC1_WORD0 to
*._CFG_WRC_WORD.b_CFG_WRC2_WORD3
for each word of the reference value message frames the reference value that is to be received at this location. Valid reference value numbers are in the range 0 to 7. For doubleword reference values, you must use two words in the message frame.

You can change the configuration even during active operation. The system applies the change in the next CANsync interval at the latest. At the start of the CANsync interval, the CANsync interface module reads the configuration data.

The system carries out reference value message frame evaluation at the start of the CANsync interval. For this, you poll the status registers (control registers) of the reference value channels (*.b_CTRLREG_WRC1 to *.b_CTRLREG_WRC2). The system enters 16#02 in these registers if the corresponding reference value message frame was received. Then, you can read out the reference values from registers *.a_WR_VALUE_RECEIVE[0] to *.a_WR_VALUE_RECEIVE[7] and set the status register to 16#00, for example, to be able to detect correctly the next entry.

You must assign in accordance with the reference value configuration in the application program the reference values that must be read when a reference value message frame has been received.

If a reference value message frame fails, which contains a position reference value from the virtual leading axle, for example, the system must carry out extrapolation in the application program starting from the last position reference value. If the reference value fails several times in succession, a fault response must be triggered.

3.2.3.5 Actual values of the CANsync slave

Actual values are sent in the CANsync event task.

Register	Contents
*.b_CTRLREG_RD_RDC1	Control register of actual value channel 1
*.b_CTRLREG_RD_RDC2	Control register of actual value channel 2
*.b_CTRLREG_RD_RDC3	Reserved
*.b_CTRLREG_RD_RDC4	Reserved

(* Corresponds, for example, to _CANsyncSlave_Ctrl_Slot_G)

In the control register, the system marks with 16#05 the fact that new actual values for the respective actual value message frame were entered and that the message frame will be sent. The CANsync interface module acknowledges the command with 16#04.

Actual value message frames are configured using the following registers.

Register	Contents
*._CFG_RDC_WORD.b_CFG_RDC1_WORD0	Configuration of actual value message frame 1 word 0
*._CFG_RDC_WORD.b_CFG_RDC1_WORD1	Configuration of actual value message frame 1 word 1
*._CFG_RDC_WORD.b_CFG_RDC1_WORD2	Configuration of actual value message frame 1 word 2
*._CFG_RDC_WORD.b_CFG_RDC1_WORD3	Configuration of actual value message frame 1 word 3
*._CFG_RDC_WORD.b_CFG_RDC2_WORD0	Configuration of actual value message frame 2 word 0
*._CFG_RDC_WORD.b_CFG_RDC2_WORD1	Configuration of actual value message frame 2 word 1
*._CFG_RDC_WORD.b_CFG_RDC2_WORD2	Configuration of actual value message frame 2 word 2
*._CFG_RDC_WORD.b_CFG_RDC2_WORD3	Configuration of actual value message frame 2 word 3
*._CFG_RDC_WORD.b_CFG_RDC3_WORD0	Reserved
*._CFG_RDC_WORD.b_CFG_RDC3_WORD1	Reserved
*._CFG_RDC_WORD.b_CFG_RDC3_WORD2	Reserved
*._CFG_RDC_WORD.b_CFG_RDC3_WORD3	Reserved
*._CFG_RDC_WORD.b_CFG_RDC4_WORD0	Reserved
*._CFG_RDC_WORD.b_CFG_RDC4_WORD1	Reserved
*._CFG_RDC_WORD.b_CFG_RDC4_WORD2	Reserved
*._CFG_RDC_WORD.b_CFG_RDC4_WORD3	Reserved

(* Corresponds, for example, to _CANsyncSlave_Ctrl_Slot_G)

Your configuration tells the system which actual value is to be entered at which location in the actual value message frame.

Meaning

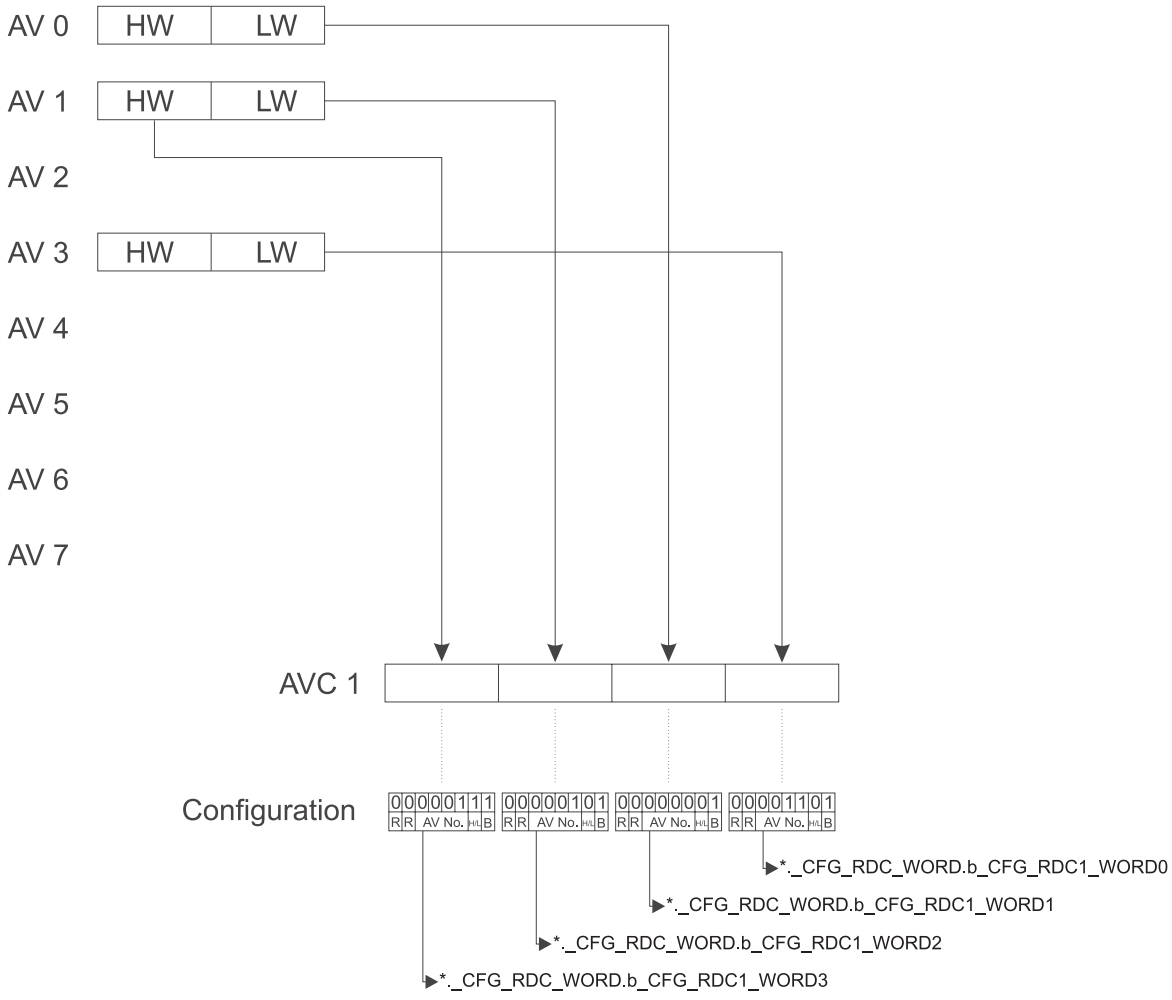
Bit 0	**Assigned** If = 1, the system uses this word of the message frame
Bit 1	**Highword/lowword** If = 1, the system enters the highword of the actual value

Bit 2	**Actual value number** Number of actual value 0 to 15
Bit 3	
Bit 4	
Bit 5	
Bit 6	Reserved
Bit 7	

NOTE

One word of the message frame is only not used if the following settings are made: actual value number = 0, highword/lowword = 0 and assigned = 0.

Example:



*._CFG_RDC_WORD.b_CFG_RDC1_WORD0 = 16#0D

0	0	0	0	1	1	0	1
R	R	AV No.				H/L	B

(* Corresponds, for example, to _CANsyncSlave_Ctrl_Slot_G)

This setting enters the lowword of actual value 3 in actual value message frame 1 in word 0.

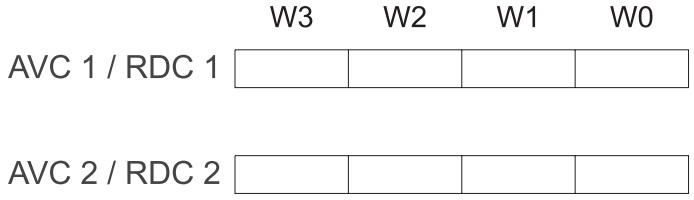
Register	Contents
*.a_RD_VALUE_SEND[0]	Actual value 0
*.a_RD_VALUE_SEND[1]	Actual value 1
*.a_RD_VALUE_SEND[2]	Actual value 2
*.a_RD_VALUE_SEND[3]	Actual value 3
*.a_RD_VALUE_SEND[4]	Actual value 4
*.a_RD_VALUE_SEND[5]	Actual value 5
*.a_RD_VALUE_SEND[6]	Actual value 6
*.a_RD_VALUE_SEND[7]	Actual value 7
*.a_RD_VALUE_SEND[8]	Reserved
...	...
*.a_RD_VALUE_SEND[15]	Reserved

(* Corresponds, for example, to _CANsyncSlave_Ctrl_Slot_G)

The actual values can be word or doubleword actual values.

Explanation of Using the Actual Value Channels

In synchronous operation, two actual value channels (channels 1 and 2; also RDC1 and RDC2) are available. Actual value message frames 1 and 2 of this CANsync slave are sent to the CANsync master on actual value channels 1 and 2.



Both actual value message frames consist of four words each (W0 to W3). After CANsync initialization, you must state at least once the assignment of these words.

This can be carried out in the initialization program. To do this, enter in areas *._CFG_RDC_WORD.b_CFG_RDC1_WORD0 to *._CFG_RDC_WORD.b_CFG_RDC2_WORD3 for each word of the actual value message frames the actual value that is to be transferred at this location. Valid actual value numbers are in the range 0 to 7. For doubleword actual values, you must use two words in the message frame.

You can change the configuration even during active operation. The system applies the change in the next CANsync interval at the latest. The system reads the CANsync interface module's configuration data when the corresponding actual value message frame is received.

CANsync slaves cannot themselves trigger sending of their actual values, but rather, the CANsync master uses the identifier of the reference value message frame to request a CANsync slave's actual values.

To ensure that the current actual values are always sent, the actual value entry must be made at the start of the CANsync event task. Approximately 70 µs elapse from the falling edge of the SYNC signal to execution of the application program in the CANsync event task. The system must enter the actual values for actual value message frame 1 by approximately 200 µs after the start of the CANsync event task to registers *.a_RD_VALUE_SEND[0] to *.a_RD_VALUE_SEND[7]. This is because the CANsync interface module starts preparing the actual value message frames at this time. To identify the fact that new actual values have been entered, the system must enter 16#05 in the appropriate control register of the actual value channel *.b_CTRLREG_RD_RDC1 to *.b_CTRLREG_RD_RDC2. This is the enable telling the CANsync interface module that the actual values can be read and that the actual value message frame is being generated. If this enable is not issued by 200 µs after the start of the CANsync event task, actual value message frame 1 is omitted in this CANsync interval. To acknowledge generation of the actual value message frame, the CANsync interface module enters 16#04 in the control register. This allows users to check whether the system finished entering the actual value in good time or not. If generation of the actual values takes longer, the system must always generate the actual values for the next CANsync interval. This means that at the beginning of the new CANsync interval the system only needs to copy the precalculated actual values to the corresponding registers.

The time by which the system must have entered the actual values for actual value message frame 2, is 450 µs after the start of the CANsync event task. Signalling in the control register is the same as for actual value message frame 1.

3.2.3.6 Actual values of other CANsync slaves

The actual values of the CANsync slaves are received in the CANsync event task.

Register	Contents
*.b_STATREG_RDC1	Status register of actual value channel 1
*.b_STATREG_RDC2	Status register of actual value channel 2
*.b_STATREG_RDC3	Reserved
*.b_STATREG_RDC4	Reserved

(* Corresponds, for example, to _CANsyncSlave_Ctrl_Slot_G)

The system enters in the status register the slave numbers of the CANsync slaves from which an actual value message frame was received.

Register	Contents
*.a_STATREG_RDC[0].b_STATREG_RDC1	Actual value acknowledgement of actual value channel 1 of slave 0
*.a_STATREG_RDC[0].b_STATREG_RDC2	Actual value acknowledgement of actual value channel 2 of slave 0
*.a_STATREG_RDC[0].b_STATREG_RDC3	Reserved
*.a_STATREG_RDC[0].b_STATREG_RDC4	Reserved
*.a_STATREG_RDC[1].b_STATREG_RDC1	Actual value acknowledgement of actual value channel 1 of slave 1
*.a_STATREG_RDC[1].b_STATREG_RDC2	Actual value acknowledgement of actual value channel 2 of slave 1

Register	Contents
*.a_STATREG_RDC[1].b_STATREG_RDC3	Reserved
*.a_STATREG_RDC[1].b_STATREG_RDC4	Reserved
...	...
*.a_STATREG_RDC[31].b_STATREG_RDC1	Actual value acknowledgement of actual value channel 1 of slave 31
*.a_STATREG_RDC[31].b_STATREG_RDC2	Actual value acknowledgement of actual value channel 2 of slave 31
*.a_STATREG_RDC[31].b_STATREG_RDC3	Reserved
*.a_STATREG_RDC[31].b_STATREG_RDC4	Reserved

(* Corresponds, for example, to _CANsyncSlave_Ctrl_Slot_G)

When the corresponding actual value message frame for a CANsync slave has arrived, the system enters 16#02 in the actual value acknowledgement register.

You use the following registers to configure the actual value message frames of the other CANsync slaves.

Register	Contents
*.a_CFG_RDC_WORD[0].b_CFG_RDC1_WORD0	Configuration of actual value message frame 1 word 0 slave 0
*.a_CFG_RDC_WORD[0].b_CFG_RDC1_WORD1	Configuration of actual value message frame 1 word 1 slave 0
*.a_CFG_RDC_WORD[0].b_CFG_RDC1_WORD2	Configuration of actual value message frame 1 word 2 slave 0
*.a_CFG_RDC_WORD[0].b_CFG_RDC1_WORD3	Configuration of actual value message frame 1 word 3 slave 0
*.a_CFG_RDC_WORD[0].b_CFG_RDC2_WORD0	Configuration of actual value message frame 2 word 0 slave 0
*.a_CFG_RDC_WORD[0].b_CFG_RDC2_WORD1	Configuration of actual value message frame 2 word 1 slave 0
*.a_CFG_RDC_WORD[0].b_CFG_RDC2_WORD2	Configuration of actual value message frame 2 word 2 slave 0
*.a_CFG_RDC_WORD[0].b_CFG_RDC2_WORD3	Configuration of actual value message frame 2 word 3 slave 0
*.a_CFG_RDC_WORD[0].b_CFG_RDC3_WORD0	Reserved
*.a_CFG_RDC_WORD[0].b_CFG_RDC3_WORD1	Reserved
*.a_CFG_RDC_WORD[0].b_CFG_RDC3_WORD2	Reserved
*.a_CFG_RDC_WORD[0].b_CFG_RDC3_WORD3	Reserved
*.a_CFG_RDC_WORD[0].b_CFG_RDC4_WORD0	Reserved
*.a_CFG_RDC_WORD[0].b_CFG_RDC4_WORD1	Reserved
*.a_CFG_RDC_WORD[0].b_CFG_RDC4_WORD2	Reserved
*.a_CFG_RDC_WORD[0].b_CFG_RDC4_WORD3	Reserved
*.a_CFG_RDC_WORD[1].b_CFG_RDC1_WORD0	Configuration of actual value message frame 1 word 0 slave 1
*.a_CFG_RDC_WORD[1].b_CFG_RDC1_WORD1	Configuration of actual value message frame 1 word 1 slave 1
*.a_CFG_RDC_WORD[1].b_CFG_RDC1_WORD2	Configuration of actual value message frame 1 word 2 slave 1
*.a_CFG_RDC_WORD[1].b_CFG_RDC1_WORD3	Configuration of actual value message frame 1 word 3 slave 1
*.a_CFG_RDC_WORD[1].b_CFG_RDC2_WORD0	Configuration of actual value message frame 2 word 0 slave 1
*.a_CFG_RDC_WORD[1].b_CFG_RDC2_WORD1	Configuration of actual value message frame 2 word 1 slave 1
*.a_CFG_RDC_WORD[1].b_CFG_RDC2_WORD2	Configuration of actual value message frame 2 word 2 slave 1
*.a_CFG_RDC_WORD[1].b_CFG_RDC2_WORD3	Configuration of actual value message frame 2 word 3 slave 1
...	...
*.a_CFG_RDC_WORD[31].b_CFG_RDC1_WORD0	Configuration of actual value message frame 1 word 0 slave 31

Register	Contents
*.a_CFG_RDC_WORD[31].b_CFG_RDC1_WORD1	Configuration of actual value message frame 1 word 1 slave 31
*.a_CFG_RDC_WORD[31].b_CFG_RDC1_WORD2	Configuration of actual value message frame 1 word 2 slave 31
*.a_CFG_RDC_WORD[31].b_CFG_RDC1_WORD3	Configuration of actual value message frame 1 word 3 slave 31
*.a_CFG_RDC_WORD[31].b_CFG_RDC2_WORD0	Configuration of actual value message frame 2 word 0 slave 31
*.a_CFG_RDC_WORD[31].b_CFG_RDC2_WORD1	Configuration of actual value message frame 2 word 1 slave 31
*.a_CFG_RDC_WORD[31].b_CFG_RDC2_WORD2	Configuration of actual value message frame 2 word 2 slave 31
*.a_CFG_RDC_WORD[31].b_CFG_RDC2_WORD3	Configuration of actual value message frame 2 word 3 slave 31
*.a_CFG_RDC_WORD[31].b_CFG_RDC3_WORD0	Reserved
*.a_CFG_RDC_WORD[31].b_CFG_RDC3_WORD1	Reserved
*.a_CFG_RDC_WORD[31].b_CFG_RDC3_WORD2	Reserved
*.a_CFG_RDC_WORD[31].b_CFG_RDC3_WORD3	Reserved
*.a_CFG_RDC_WORD[31].b_CFG_RDC4_WORD0	Reserved
*.a_CFG_RDC_WORD[31].b_CFG_RDC4_WORD1	Reserved
*.a_CFG_RDC_WORD[31].b_CFG_RDC4_WORD2	Reserved
*.a_CFG_RDC_WORD[31].b_CFG_RDC4_WORD3	Reserved

(* Corresponds, for example, to _CANsyncSlave_Ctrl_Slot_G)

Your configuration tells the system which actual value it to be transferred at which position in the actual value message frame (of the other CANsync slave).

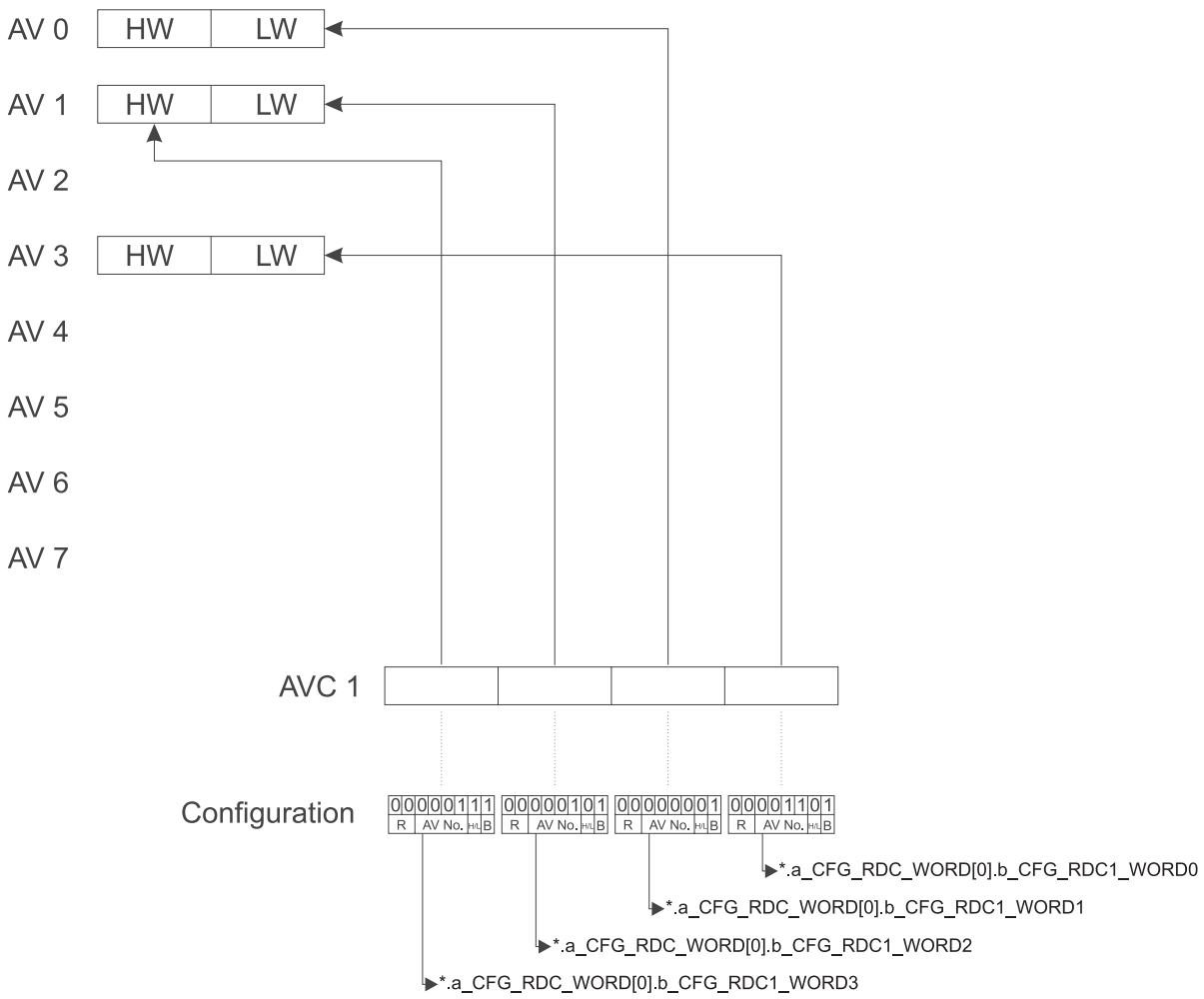
Meaning

Bit 0	**Assigned** If = 1, the system uses this word of the message frame
Bit 1	**Highword/lowword** If = 1, the system enters the highword of the actual value
Bit 2	**Actual value number** Number of actual value 0 to 7
Bit 3	
Bit 4	
Bit 5	
Bit 6	Reserved
Bit 7	

NOTE

One word of the message frame is only not used if the following settings are made: actual value number = 0, highword/lowword = 0 and assigned = 0.

Example:



*.a_CFG_RDC_WORD[0].b_CFG_RDC1_WORD0 = 16#0D

0	0	0	0	1	1	0	1
R		AV No.				H/L	B

(* Corresponds, for example, to _CANsyncSlave_Ctrl_Slot_G)

This means that the system reads word 0 from actual value message frame 1 of CANsync slave 0 and writes it to the lowword of actual value 3.

Register	Contents
*.a_RD_BMARRAY[0][0]	Actual value 0 of slave 0
*.a_RD_BMARRAY[0][1]	Actual value 1 of slave 0
*.a_RD_BMARRAY[0][2]	Actual value 2 of slave 0
*.a_RD_BMARRAY[0][3]	Actual value 3 of slave 0
*.a_RD_BMARRAY[0][4]	Actual value 4 of slave 0
*.a_RD_BMARRAY[0][5]	Actual value 5 of slave 0
*.a_RD_BMARRAY[0][6]	Actual value 6 of slave 0
*.a_RD_BMARRAY[0][7]	Actual value 7 of slave 0

Register	Contents
*.a_RD_BMARRAY[0][8]	Reserved
...	...
*.a_RD_BMARRAY[0][15]	Reserved
*.a_RD_BMARRAY[1][0]	Actual value 0 of slave 1
*.a_RD_BMARRAY[1][1]	Actual value 1 of slave 1
*.a_RD_BMARRAY[1][2]	Actual value 2 of slave 1
*.a_RD_BMARRAY[1][3]	Actual value 3 of slave 1
*.a_RD_BMARRAY[1][4]	Actual value 4 of slave 1
*.a_RD_BMARRAY[1][5]	Actual value 5 of slave 1
*.a_RD_BMARRAY[1][6]	Actual value 6 of slave 1
*.a_RD_BMARRAY[1][7]	Actual value 7 of slave 1
*.a_RD_BMARRAY[1][8]	Reserved
...	...
*.a_RD_BMARRAY[31][0]	Actual value 0 of slave 31
*.a_RD_BMARRAY[31][1]	Actual value 1 of slave 31
*.a_RD_BMARRAY[31][2]	Actual value 2 of slave 31
*.a_RD_BMARRAY[31][3]	Actual value 3 of slave 31
*.a_RD_BMARRAY[31][4]	Actual value 4 of slave 31
*.a_RD_BMARRAY[31][5]	Actual value 5 of slave 31
*.a_RD_BMARRAY[31][6]	Actual value 6 of slave 31
*.a_RD_BMARRAY[31][7]	Actual value 7 of slave 31
*.a_RD_BMARRAY[31][8]	Reserved
...	...
*.a_RD_BMARRAY[31][15]	Reserved

(* Corresponds, for example, to _CANsyncSlave_Ctrl_Slot_G)

The actual values can be word or doubleword actual values.

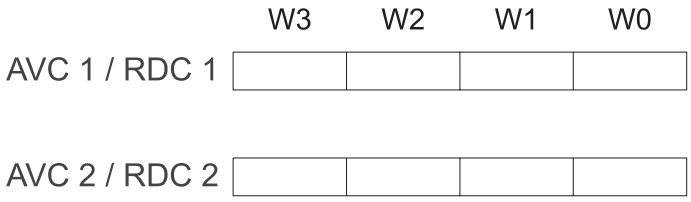
NOTE

When displaying _CANsyncSlave_Ctrl_Slot_G in the PROPROG wt II watch window, a period may be between the square brackets.

Explanation of Using the Actual Value Channels

A CANsync slave can also evaluate all the actual value message frames of the other CANsync slaves. However, it is only possible to request them from the CANsync master.

In synchronous operation, two actual value channels (channels 1 and 2; also RDC1 and RDC2) are available. Actual value message frames 1 and 2 are sent on actual value channels 1 and 2.



Both actual value message frames consist of four words each (W0 to W3). After CANsync initialization, you must state at least once the assignment of these words. This can be carried out in the initialization program.
To do this, you must enter in area *.a_CFG_RDC_WORD[0].b_CFG_RDC1_WORD0 to *.a_CFG_RDC_WORD[31].b_CFG_RDC2_WORD3 for each assigned word of the actual value message frames (of each CANsync slave) the actual value of the other CANsync slaves that is transferred at this position. Valid actual value numbers are in the range 0 to 7. For doubleword actual values, you must use two words in the message frame.

You can change the configuration even during active operation. The system applies the change in the next CANsync interval at the latest. The CANsync interface module reads the configuration data when the corresponding actual value message frame is received.

The system carries out actual value message frame evaluation at the start of the CANsync event task. There are two evaluation methods: With the first one, you poll the status registers of the actual value channels (*.b_STATREG_RDC1 or *.b_STATREG_RDC2). The system enters in these registers the slave number of the CANsync slave from which the corresponding actual value message frame was received. Then, you can read out the actual values from registers (*.a_RD_BMARRAY[0][0] to *.a_RD_BMARRAY[31][7]) of this CANsync slave and set the status register to 16#7, for example, to be able to detect correctly the next entry.

With the second method, you poll directly actual value acknowledgement (*.a_STATREG_RDC[0].b_STATREG_RDC1 to *.a_STATREG_RDC[31].b_STATREG_RDC2) for one CANsync slave (whose actual value message frames are to be monitored). In this acknowledgement register, the system marks with 16#02 reception of an actual value message frame. You can then read out the actual values from registers (*.a_RD_BMARRAY[0][0] to *.a_RD_BMARRAY[31][7]) of this CANsync slave. In this case too, the system must then write another value to the status register to detect reentry of the acknowledgement.

You must assign in accordance with the actual value configuration in the application program the actual values that must be read when an actual value message frame has been received.

The system uses the CANsync master's reference value message frame to request a CANsync slave to send its actual value message frame. You can only make the setting on the CANsync master.

3.2.3.7 Command and Response Channel

Action command

Action commands are reported in the following registers.

Register	Contents
*.b_STATREG_AKT_CMD	Status register of action command
*.b_AKT_CMD	Action command
*.b_DATA_BYTE_AKT_CMD	Data byte 0 (DB)
*.w_DATAWORD_AKT_CMD	Data word 1 (DW)
*.b_STATREG_CONTROLWORD	Status register of control word
*.w_CONTROLWORD	Control word

Register	Contents
*.b_STATREG_SYNC_MODUS	Status register of SYNC mode
*.b_SYNC_MODUS	SYNC mode

(* Corresponds, for example, to _CANsyncSlave_Ctrl_Slot_G)

The action command that the CANsync master transmits as a broadcast command is only reported if the CANsync slave is activated, i.e. if the corresponding bit is set in the bit strip (see ▹Register structure and function of the CANsync master option module for b maXX PLC ◃ from page 42 onward).

If the action command is a control word command (command byte = 16#01), the system enters the command's data word in *.w_CONTROLWORD as the control word and enters 16#02 in *.b_STATREG_CONTROLWORD as the reception indicator.

With all the other action commands, the system enters the data in *.b_DATA_BYTE_AKT_CMD and *.w_DATAWORD_AKT_CMD. In this case, the acknowledgement (reception indicator 16#02) is in *.b_STATREG_AKT_CMD. Since a new action command can be transferred in every CANsync interval, you should evaluate the action command's status register in every CANsync event task.

Parameter command

Parameter commands are reported in the following registers.

Register	Contents
*.b_STEUREG_PAR_CMD	Control register of parameter command
*.b_STATREG_PAR_CMD	Status register of parameter command
*.w_ERR_NR_PAR_CMD	Error number of parameter command
*.d_DATA_PAR_CMD	Data of parameter command
*.w_PAR_NR	Parameter number

(* Corresponds, for example, to _CANsyncSlave_Ctrl_Slot_G)

Meanings of the control and status registers

Bit 0	**Active** Is set to 1 when the job has been received Must be set to 0 when the job has been processed
Bit 1	**Read** =1: Write parameter
Bit 2	**Write** =1: Read parameter
Bit 3	**change** = 1: Indication that the CANsync master has changed the parameter command; must be acknowledged by reset
Bit 4	**Format** 0: Word parameter 1: Doubleword parameter

Bit 5	**Error display** Must be set to 1 if the job generates an error
Bit 6	**busy** = 1: response has not yet been sent = 0: response has been sent to the CANsync master
Bit 7	**Reset** = 1 confirms that the command change was seen at the change bit and that the old command will not be further-processed.

Sequence of a Parameter Access

Parameter commands can be processed both in the CANsync event task and in the rest of the program. To ensure data consistency, the system allocates jobs and evaluates the response via the control register and the status register. To guarantee that the job is carried out without conflicts, first the control register and then the status register must always be read from the application program and if a new value is written, the program must always first set the status register and then the control register. (The control register is the crucial one for the CANsync interface module).

When a write parameter job has been received, the value to be written is located in *.d_DATA_PAR_CMD and the parameter number is located in *.w_PAR_NR. In the control register (*.b_STEUREG_PAR_CMD) and in the status register (*.b_STATREG_PAR_CMD), the system reports a write word command with 16#45 and a write doubleword command with 16#55.

If the application program has written the requested parameter error-free, this must be indicated in the status register and the control register by clearing bit 0 (active).

As soon as the response message frame has been sent to the CANsync master, the CANsync interface module clears bit 6 (busy).

If the application program cannot carry out the write access, an error number must be entered in *.w_ERR_NR_PAR_CMD and the system must set in the registers bit 5 (error display) and clear bit 0 (active). Then, a response message frame containing the entered error number is sent to the CANsync master.

As soon as the response message frame has been sent to the CANsync master, the CANsync interface module clears bit 6 (busy).

The read parameter job runs in a similar way with the only differences being that, regardless of the format, the command is 16#43 and that as the response the data must be entered in *.d_DATA_PAR_CMD. Before acknowledging by deleting bit 0 (active), the system must enter the parameter's actual format in bit 4 (format) so that the response to the CANsync master can be generated correctly.

If the CANsync master changes the parameter command (parameter number), the system indicates this by setting bit 3 (change). In this case, the old command must not be responded to. The application program must detect the change and acknowledge this by setting bit 7 (reset). Then, the current parameter command is entered.

Error code

Value	Meaning
16#0000	No error occurred
16#FFFF	Error occurred
16#FFFE	Value less than minimum value
16#FFFD	Value greater than maximum value
16#FFFC	Element cannot be changed
16#FFFB	Element not present
16#FFFA	Data is not available (e.g. being processed)
16#FFF9	Error in data format

Upload/download command

The system displays upload and download commands are in the following registers.

Register	Contents
*.b_STEUREG_UPDOWNBLK1	Control register of upload/download block 1
*.b_STATREG_UPDOWNBLK1	Status register of upload/download block 1
*.b_EN_UPDOWNBLK1	Enable of block 1
*.w_ERR_NR_UPDOWNBLK1	Error number of upload/download block 1
*.d_BASE_ADR_UPDOWNBLK1	Base address of upload/download 1
...	...
*.w_LENGTH_UPDOWNBLK1	Length in bytes of upload/download block 1
*.w_COUNTER_UPDOWNBLK1	Counter of upload/download block 1
*.a_DATA_UPDOWNBLK1[0 to 74]	Data block of upload/download block 1

(* Corresponds, for example, to _CANsyncSlave_Ctrl_Slot_G)

Meanings of the control and status registers

Bit 0	**Active** Is set to 1 when the job has been received Must be set to 0 when the job has been processed
Bit 1	**change** = 1: Indication that the CANsync master has changed the parameter command; must be acknowledged by reset
Bit 2 Bit 3	**Mode** Bit3 Bit2 0 0: Reserved 0 1: initialization 1 0: ongoing upload/download 1 1: End of block
Bit 4	**Upload/download** =0: Upload =1: Download

Bit 5	**Error display** Must be set to 1 if the job generates an error
Bit 6	**busy** = 1: response has not yet been sent = 0: response has been sent to the CANsync master
Bit 7	**Reset** = 1 confirms that the command change was seen at the change bit and that the old command will not be further-processed.

Sequence of an Upload/Download Job in Block 1

The evaluation of the upload/download commands can be used in the CANsync event task as well as in the rest of the program. To ensure data consistency, the system allocates jobs and evaluates the response via the control register and the status register. To guarantee that the job is carried out without conflicts, first the control register and then the status register must always be read from the application program and if a new value is written, the program must always first set the status register and then the control register. (The control register is the crucial one for the CANsync interface module).

When a download is logged on, the system enters the base address in *.d_BASE_ADR_UPDOWNBLK1 and the block length in *.w_LENGTH_UPDOWNBLK1.

Base addresses 16#00000000 to 16#000000FF are reserved for operating system jobs.

The value 16#55 is reported in status register *.b_STATREG_UPDOWNBLK1 and in control register *.b_STEUREG_UPDOWNBLK1. If the download is allowed, the application program must clear bit 0 (active). Then, the CANsync master receives the entire block (you can read off the progress in the byte counter in *.w_COUNTER_UPDOWNBLK1) and at the end the system reports 16#5D in the registers. If the data is taken from area *.a_DATA_UPDOWNBLK1[0] to *.a_DATA_UPDOWNBLK1[74], you must clear bit 0 (active). To indicate that the response message frame has been sent to the CANsync master, the system clears bit 6 (busy).

If an error occurs in the CANsync slave while it is carrying out the job, the system must enter the error number in *.w_ERR_NR_UPDOWNBLK1, set bit 5 (error display) and clear bit 0 (active). Then a corresponding error message frame is sent to the CANsync master, which cancels the download.

Upload jobs run in a similar way with the only differences being that the command is 16#05 and that the data must be entered in the block area before acknowledgement of the initialization message frame.

If the CANsync master changes the upload/download command, the system indicates this by setting bit 1 (change). Then, the old command must be cancelled. The application program must detect the change and acknowledge this by setting bit 7 (reset). Then, the current upload/download command is entered.

The error numbers that the slave CANsync can enter are above 16#00FF.

Upload/download error numbers that the CANsync slave interface module sends to the CANsync master; display in the CANsync master only:

Error number	Meaning
16#0001	CANsync slave acknowledges wrong block number
16#0002	Entered length greater than 300 bytes
16#0100	CANsync slave expects block with the number that is entered in the counter
16#0101	CANsync slave expects block end
16#0102	CANsync slave does not yet expect block end
16#0103	CANsync slave cancels upload/download
16#0104	Upload/download not possible
16#0105	Base address not allowed
16#0106	Reserved
16#0107	Block length > CANsync slave's maximum block length
16#0108	Message frame mode error (mode not allowed at this stage)

4

CANsync Function Blocks

In this chapter, we will describe the individual function blocks of the CANsync library. You can also find this description in the online help for each respective function block.

4.1 Function Blocks for the Synchronized CAN Overview

In addition to the standard functions, you can use manufacturer-defined functions if you have logged on libraries in a project.

Note: Logging on of libraries is described in the general help.

The following function blocks are available for synchronized CAN:

Function	Brief description
CANsync_BC_MA0	Sends the CANsync master's broadcast command in transmission range 0 (highest priority)
CANsync_BC_MA1	Sends the CANsync master's broadcast command in transmission range 1 (medium priority)
CANsync_BC_MA2	Sends the CANsync master's broadcast command in transmission range 2 (low priority)
CANsync_BC_SL	Receives the CANsync master's broadcast commands
CANsync_COMM_CONTROL_MA	Configures use of a CANsync master interface module's command channel
CANsync_CONTROLWORD_MA	Sends the CANsync master's control word commands
CANsync_CONTROLWORD_SL	Receives the CANsync master's control word in a CANsync slave interface module
CANsync_INIT	Initializes a CANsync interface module
CANsync_MODE_MA	Sets the operating mode of a CANsync master interface module
CANsync_MODE_SL	Sets the operating mode of a CANsync slave interface module
CANsync_PAR_READ_MA	The CANsync master requests via CANsync a parameter value from the CANsync slave

Function	Brief description
CANsync_PAR_SL	Detects the parameter request or the transferred parameter
CANsync_PAR_WRITE_MA	CANsync master sends via CANsync a parameter value to the CANsync slave
CANsync_PD_CFG_MA	Configures assignment of the CAN interface module's reference value channels for a CANsync master
CANsync_PD_CFG_READ_MA	Configures assignment of the CAN interface module's actual value channels for a CANsync master
CANsync_PD_CFG_READ_SL	Configures assignment of the CAN interface module's actual value channels for a CANsync slave
CANsync_PD_CFG_SL	Configures assignment of the CAN interface module's reference value and actual value channels for a CANsync slave
CANsync_PD_COMM_MA	Copies the process data (CANsync interface module's reference values and actual values) for a CANsync master
CANsync_PD_COMM_READ_MA	Copies in a (master) CANsync interface module the process data actual values of a CANsync slave
CANsync_PD_COMM_READ_SL	Copies in a (slave) CANsync interface module the process data actual values of a CANsync slave
CANsync_PD_COMM_SL	Copies the process data (reference values and actual values) for a CANsync slave
CANsync_SL_TYP_INIT	CANsync slave types (initialization)
CANsync_UPDOWNLOAD_MA	CANsync upload download Master
CANsync_UPDOWNLOAD_SL	CANsync upload download slave

4.2 CANsync_BC_MA0

Description

You can use this function block for CANsync to send a broadcast command of the CANsync master in transmission range 0 (highest priority).

NOTE

FB CANsync_BC_MA0 uses library BM_TYPES_20bd03 or above.

Parameter input	Data type	Description
_BASE	CANsync_PLC_MA_BMSTRUCT	Operating data for the CANsync interface module
us_BC_CMD_NR	USINT	Command number of the broadcast command
si_BC_BYTE	SINT	Data byte of the broadcast command
i_BC_WORD	INT	Data word of the broadcast command
d_SL_MASK	DWORD	Bit strip to which CANsync slaves the broadcast command is to be sent
x_EN	BOOL	Enable

Parameter output	Data type	Description
_BASE	CANsync_PLC_MA_BMSTRUCT	Operating data for the CANsync interface module
x_OK	BOOL	OK bit

With the CANsync, there are three transmission ranges for broadcast message frames with broadcast commands (transmission ranges 0 to 2). In any one CANsync interval, it is only possible to send one broadcast message frame. Transmission range 0 has the highest priority. FB CANsync_BC_MA0 uses transmission range 0 for sending the broadcast message frame.

Input/output _BASE:

At input/output `_BASE`, global variable `_CANsyncMaster_Ctrl_Slot_G` (to `_CANsyncMaster_Ctrl_Slot_M`) is connected in accordance with the slot (G to M) of the option module.

If this global variable is not present in the project, create – in dependence on the slot (G to M) – the global variable `_CANsyncMaster_Ctrl_Slot_G` (to `_CANsyncMaster_Ctrl_Slot_M`) of data type `CANsync_PLC_MA_BMSTRUCT`.

Using the declaration of global variables, this variable must be connected to the base address of the CANsync interface module on the option module.

In dependence on the slot, the base address of the CANsync interface module on the CANsync master option module (BM4-O-CAN-06) is as follows:

Slot	Base address of CANsync interface module
G	%MB3.2001792
H	%MB3.3001792
J	%MB3.4001792
K	%MB3.5001792
L	%MB3.6001792
M	%MB3.7001792

Example:

CANsync interface module on the CANsync master option module in the b maXX's slot G

```
_CANsyncMaster_Ctrl_Slot_G  AT  %MB3.2001792  :
                                      CANsync_PLC_MA_BMSTRUCT;
```

Where:

`_CANsyncMaster_Ctrl_Slot_G`	is the variable name with the data type short designation "_" for Struct
`CANsync_PLC_MA_BMSTRUCT`	is the data type
`%MB3.2001792`	is the base address of the CANsync interface module on the CANsync master option module in the b maXX's slot G

Input us_BC_CMD_NR:

At input us_BC_CMD_NR, you state the command number of the broadcast command.

Inputs si_BC_BYTE, i_BC_WORD:

At inputs si_BC_BYTE and i_BC_WORD, you must connect the associated data in dependence on the command number (see further below in the list of broadcast commands).

Input d_SL_MASK:

At input d_SL_MASK, you state which of the CANsync slaves is to receive the command. To do this, you must set to TRUE for each CANsync slave the bit number that matches its slave number. For the CANsync slave with slave number 0, you set bit 0 of d_SL_MASK to TRUE; for the CANsync slave with slave number 1, you set bit 1 of d_SL_MASK to TRUE, etc. If you want to address all the CANsync slaves, you must set d_SL_MASK = 16#FFFFFFFF.

Input x_EN:

If input x_EN is set to TRUE, the system enters the broadcast command for sending. The CANsync interface module then sends the broadcast command in the broadcast message frame to the selected CANsync slaves (input d_SL_MASK). While x_EN is set to TRUE, the system enters the broadcast command for sending every time the FB is called. However, this would mean that the CANsync's command channel would be busy all the time. You should therefore set input x_EN to TRUE for only one CANsync interval to send the broadcast command once.

Output x_OK:

Output x_OK indicates by TRUE that the last broadcast message frame has been sent. Output x_OK is FALSE if no broadcast message frame has been sent.

List of broadcast commands:

Command number	**Meaning**
1	Control word si_BC_BYTE: not used i_BC_WORD: control word
2 to 127	Reserved
128 – 255	Are available for users

4.3 CANsync_BC_MA1

Description

You can use this function block for CANsync to send a broadcast command of the CANsync master in transmission range 1 (medium priority).

NOTE

FB CANsync_BC_MA1 uses library BM_TYPES_20bd03 or above.

Parameter input	Data type	Description
_BASE	CANsync_PLC_MA_BMSTRUCT	Operating data for the CANsync interface module
us_BC_CMD_NR	USINT	Command number of the broadcast command
si_BC_BYTE	SINT	Data byte of the broadcast command
i_BC_WORD	INT	Data word of the broadcast command
d_SL_MASK	DWORD	Bit strip to which CANsync slaves the broadcast command is to be sent
x_EN	BOOL	Enable

Parameter output	Data type	Description
_BASE	CANsync_PLC_MA_BMSTRUCT	Operating data for the CANsync interface module
x_OK	BOOL	OK bit

With the CANsync, there are three transmission ranges for broadcast message frames with broadcast commands (transmission ranges 0 to 2). In any one CANsync interval, it is only possible to send one broadcast message frame. Transmission range 1 has the medium priority. FB CANsync_BC_MA1 uses transmission range 1 for sending the broadcast message frame.

Input/output _BASE:

At input/output `_BASE`, global variable `_CANsyncMaster_Ctrl_Slot_G` (to `_CANsyncMaster_Ctrl_Slot_M`) is connected in accordance with the slot (G to M) of the option module.

If this global variable is not present in the project, create – in dependence on the slot (G to M) – the global variable `_CANsyncMaster_Ctrl_Slot_G` (to `_CANsyncMaster_Ctrl_Slot_M`) of data type `CANsync_PLC_MA_BMSTRUCT`.

Using the declaration of global variables, this variable must be connected to the base address of the CANsync interface module on the option module.

In dependence on the slot, the base address of the CANsync interface module on the CANsync master option module (BM4-O-CAN-06) is as follows:

Slot	Base address of CANsync interface module
G	%MB3.2001792
H	%MB3.3001792
J	%MB3.4001792
K	%MB3.5001792
L	%MB3.6001792
M	%MB3.7001792

Example:

CANsync interface module on the CANsync master option module in the b maXX's slot G

```
_CANsyncMaster_Ctrl_Slot_G  AT  %MB3.2001792  :
                                    CANsync_PLC_MA_BMSTRUCT;
```

Where:

`_CANsyncMaster_Ctrl_Slot_G`	is the variable name with the data type short designation "_" for Struct
`CANsync_PLC_MA_BMSTRUCT`	is the data type
`%MB3.2001792`	is the base address of the CANsync interface module on the CANsync master option module in the b maXX's slot G

Input us_BC_CMD_NR:

At input us_BC_CMD_NR, you state the command number of the broadcast command.

Inputs si_BC_BYTE, i_BC_WORD:

At inputs si_BC_BYTE and i_BC_WORD, you must connect the associated data in dependence on the command number (see further below in the list of broadcast commands).

Input d_SL_MASK:

At input d_SL_MASK, you state which of the CANsync slaves is to receive the command. To do this, you must set to TRUE for each CANsync slave the bit number that matches its slave number. For the CANsync slave with slave number 0, you set bit 0 of d_SL_MASK to TRUE; for the CANsync slave with slave number 1, you set bit 1 of d_SL_MASK to TRUE, etc. If you want to address all the CANsync slaves, you must set d_SL_MASK = 16#FFFFFFFF.

Input x_EN:

If input x_EN is set to TRUE, the system enters the broadcast command for sending. The CANsync interface module then sends the broadcast command in the broadcast message frame to the selected CANsync slaves (input d_SL_MASK). While x_EN is set to TRUE, the system enters the broadcast command for sending every time the FB is called. However, this would mean that the CANsync's command channel would be busy all the time. You should therefore set input x_EN to TRUE for only one CANsync interval to send the broadcast command once.

Output x_OK:

Output x_OK indicates by TRUE that the last broadcast message frame has been sent. Output x_OK is FALSE if no broadcast message frame has been sent.

List of broadcast commands:

Command number	**Meaning**
1	Control word si_BC_BYTE: not used i_BC_WORD: control word
2 to 127	Reserved
128 to 255	Are available for users

4.4 CANsync_BC_MA2

Description

You can use this function block for CANsync to send a broadcast command of the CANsync master in transmission range 2 (lowest priority).

NOTE

FB CANsync_BC_MA2 uses library BM_TYPES_20bd03 or above.

Parameter input	Data type	Description
_BASE	CANsync_PLC_MA_BMSTRUCT	Operating data for the CANsync interface module
us_BC_CMD_NR	USINT	Command number of the broadcast command
si_BC_BYTE	SINT	Data byte of the broadcast command
i_BC_WORD	INT	Data word of the broadcast command
d_SL_MASK	DWORD	Bit strip to which CANsync slaves the broadcast command is to be sent
x_EN	BOOL	Enable

Parameter output	Data type	Description
_BASE	CANsync_PLC_MA_BMSTRUCT	Operating data for the CANsync interface module
x_OK	BOOL	OK bit

With the CANsync, there are three transmission ranges for broadcast message frames with broadcast commands (transmission ranges 0 to 2). In any one CANsync interval, it is only possible to send one broadcast message frame. Transmission range 2 has the lowest priority. FB CANsync_BC_MA2 uses transmission range 2 for sending the broadcast message frame.

Input/output _BASE:

At input/output `_BASE`, global variable `_CANsyncMaster_Ctrl_Slot_G` (to `_CANsyncMaster_Ctrl_Slot_M`) is connected in accordance with the slot (G to M) of the option module.

If this global variable is not present in the project, create – in dependence on the slot (G to M) – the global variable `_CANsyncMaster_Ctrl_Slot_G` (to `_CANsyncMaster_Ctrl_Slot_M`) of data type `CANsync_PLC_MA_BMSTRUCT`.

Using the declaration of global variables, this variable must be connected to the base address of the CANsync interface module on the option module.

In dependence on the slot, the base address of the CANsync interface module on the CANsync master option module (BM4-O-CAN-06) is as follows:

Slot	Base address of CANsync interface module
G	%MB3.2001792
H	%MB3.3001792
J	%MB3.4001792
K	%MB3.5001792
L	%MB3.6001792
M	%MB3.7001792

Example:

CANsync interface module on the CANsync master option module in the b maXX's slot G

```
_CANsyncMaster_Ctrl_Slot_G  AT  %MB3.2001792  :
                                      CANsync_PLC_MA_BMSTRUCT;
```

Where:

`_CANsyncMaster_Ctrl_Slot_G`	is the variable name with the data type short designation "_" for Struct
`CANsync_PLC_MA_BMSTRUCT`	is the data type
`%MB3.2001792`	is the base address of the CANsync interface module on the CANsync master option module in the b maXX's slot G

Input us_BC_CMD_NR:

At input us_BC_CMD_NR, you state the command number of the broadcast command.

Inputs si_BC_BYTE, i_BC_WORD:

At inputs si_BC_BYTE and i_BC_WORD, you must connect the associated data in dependence on the command number (see further below in the list of broadcast commands).

Input d_SL_MASK:

At input d_SL_MASK, you state which of the CANsync slaves is to receive the command. To do this, you must set to TRUE for each CANsync slave the bit number that matches its slave number. For the CANsync slave with slave number 0, you set bit 0 of d_SL_MASK to TRUE; for the CANsync slave with slave number 1, you set bit 1 of d_SL_MASK to TRUE, etc. If you want to address all the CANsync slaves, you must set d_SL_MASK = 16#FFFFFFFF.

Input x_EN:

If input x_EN is set to TRUE, the system enters the broadcast command for sending. The CANsync interface module then sends the broadcast command in the broadcast message frame to the selected CANsync slaves (input d_SL_MASK). While x_EN is set to TRUE, the system enters the broadcast command for sending every time the FB is called. However, this would mean that the CANsync's command channel would be busy all the time. You should therefore set input x_EN to TRUE for only one CANsync interval to send the broadcast command once.

Output x_OK:

Output x_OK indicates by TRUE that the last broadcast message frame has been sent. Output x_OK is FALSE if no broadcast message frame has been sent.

List of broadcast commands:

Command number	**Meaning**
1	Control word si_BC_BYTE: not used i_BC_WORD: control word
2 to 127	Reserved
128 to 255	Are available for users

4.5 CANsync_BC_SL

Description

You can use this function block for CANsync to use a CANsync slave to receive a broadcast command of the CANsync master.

NOTE

FB CANsync_BC_SL uses library BM_TYPES_20bd03 or above.

Parameter input	Data type	Description
_BASE	CANsync_PLC_SL_BMSTRUCT	Operating data for the CANsync interface module

Parameter output	Data type	Description
_BASE	CANsync_PLC_SL_BMSTRUCT	Operating data for the CANsync interface module
us_BC_CMD_NR	USINT	Command number of the broadcast command
si_BC_BYTE	SINT	Data byte of the broadcast command
i_BC_WORD	INT	Data word of the broadcast command
si_BC_RECEIVED	SINT 0, 2	Display indicating that a command was received

This FB indicates at si_BC_RECEIVED that a broadcast command of the CANsync master was received and outputs the command number (us_BC_CMD_NR) as well as the contents of the broadcast command (si_BC_BYTE, i_BC_WORD).

Input/output _BASE:

At input/output `_BASE`, global variable `_CANsyncSlave_Ctrl_Slot_G` (to `_CANsyncSlave_Ctrl_Slot_M`) is connected in accordance with the slot (G to M) of the option module.

If this global variable is not present in the project, create – in dependence on the slot (G to M) – the global variable `_CANsyncSlave_Ctrl_Slot_G` (to `_CANsyncSlave_Ctrl_Slot_M`) of data type `CANsync_PLC_SL_BMSTRUCT`.

Using the declaration of global variables, this variable must be connected to the base address of the CANsync interface module on the option module.

In dependence on the slot, the base address of the CANsync interface module on the CANsync slave option module for b maXX PLC (BM4-O-CAN-05, DIP switch 9 = ON) is as follows:

Slot	Base address of CANsync interface module
G	%MB3.2001792
H	%MB3.3001792
J	%MB3.4001792
K	%MB3.5001792
L	%MB3.6001792
M	%MB3.7001792

Example:

CANsync interface module on the CANsync slave option module in the b maXX's slot G

```
_CANsyncSlave_Ctrl_Slot_G  AT  %MB3.2001792  :
                                    CANsync_PLC_SL_BMSTRUCT;
```

Where:

`_CANsyncSlave_Ctrl_Slot_G`	is the variable name with the data type short designation "_" for Struct
`CANsync_PLC_SL_BMSTRUCT`	is the data type
`%MB3.2001792`	is the base address of the CANsync interface module on the CANsync slave option module in the b maXX's slot G

Output us_BC_CMD_NR:

At output us_BC_CMD_NR, the system outputs the command number of the broadcast command.

Outputs si_BC_BYTE, i_BC_WORD:

At outputs si_BC_BYTE and i_BC_WORD, the data must be read that is associated with the respective command number.

Output si_BC_RECEIVED:

At output si_BC_RECEIVED, the system indicates whether a broadcast message frame was received (with a broadcast command). In this case, si_BC_RECEIVED displays a 2. Otherwise, si_BC_RECEIVED displays a 0. The system does not read out the data associated with the broadcast message frame (us_BC_CMD_NR, si_BC_BYTE, i_BC_WORD) until the message frame has been received. Otherwise, the old values continue to be displayed.

List of broadcast commands:

Command number	Meaning
1	Control word si_BC_BYTE: not used i_BC_WORD: control word
2 to 127	Reserved
128 to 255	Are available for users

4.6 CANsync_COMM_CONTROL_MA

Description

You can use this function block for CANsync to configure the use of a CANsync interface module's command channel.

NOTE

FB CANsync_COMM_CONTROL_MA uses library BM_TYPES_20bd03 or above.

Parameter input	Data type	Description
_BASE	CANsync_PLC_MA_BMSTRUCT	Operating data for the CANsync interface module
si_SL_NR_CTRL	SINT -128, 0 to 31	Slave number of the CANsync slave with send control word job
x_EN_CTRL	BOOL	Enable for si_SL_NR_CTRL
si_SL_NR_PAR	SINT -128, 0 to 31	Slave number of the CANsync slave with parameter job
x_EN_PAR	BOOL	Enable for si_SL_NR_PAR
si_SL_NR_UDL	SINT -128, 0 to 31	Slave number of the CANsync slave with upload/download job
x_EN_UDL	BOOL	Enable for si_SL_NR_UDL
si_MAX_SL_NR [a)]	SINT -1, 0 to 31	Maximum slave number for automatic incrementing [a)]

a) This input corresponds to input si_MAX_SL_NR on FB CANsync_PD_COMM_MA as the maximum slave number for an automatic actual value message frame request.

Parameter output	Data type	Description
_BASE	CANsync_PLC_MA_BMSTRUCT	Operating data for the CANsync interface module

Using this FB, you state the slave number of the CANsync slave for which the system is to request whether send control word jobs (si_SL_NR_CTRL, x_EN_CTRL), parameter jobs (si_SL_NR_PAR, x_EN_PAR) and upload/download jobs (si_SL_NR_UDL, x_EN_UDL) are present.

NOTE

The CANsync master can send command message frame in every CANsync interval.

The system processes the various message frames on the command channel in a priority-based sequence:

Message frame type	Priority
Broadcast message frame 0	Highest
Broadcast message frame 1	↑
Broadcast message frame 2	|
Control word message frames	|
Parameter message frames	↓
Upload/download message frames	Lowest

As a result of these priorities, you cannot send another message frame if a higher-priority one is being transmitted. If you send the control word message frame in every CANsync interval, for example, you can never transmit a parameter message frame or an upload/download message frame!

If you need to poll several CANsync slaves for existing jobs, you set the CANsync interface module as follows:

In every CANsync interval the system automatically increments by 1 the slave number of the CANsync slaves for which polling for existing orders is being carried out. This incrementation is carried out up to si_MAX_SL_NR. After this, the system starts with polling for the CANsync slave with slave number 0, etc.

Input/output _BASE:

At input/output `_BASE`, global variable `_CANsyncMaster_Ctrl_Slot_G` (to `_CANsyncMaster_Ctrl_Slot_M`) is connected in accordance with the slot (G to M) of the option module.

If this global variable is not present in the project, create – in dependence on the slot (G to M) – the global variable `_CANsyncMaster_Ctrl_Slot_G` (to `_CANsyncMaster_Ctrl_Slot_M`) of data type `CANsync_PLC_MA_BMSTRUCT`.

Using the declaration of global variables, this variable must be connected to the base address of the CANsync interface module on the option module.

In dependence on the slot, the base address of the CANsync interface module on the CANsync master option module (BM4-O-CAN-06) is as follows:

Slot	Base address of CANsync interface module
G	%MB3.2001792
H	%MB3.3001792
J	%MB3.4001792
K	%MB3.5001792
L	%MB3.6001792
M	%MB3.7001792

Example:

CANsync interface module on the CANsync master option module in the b maXX's slot G

```
_CANsyncMaster_Ctrl_Slot_G  AT  %MB3.2001792  :
                                    CANsync_PLC_MA_BMSTRUCT;
```

Where:

`_CANsyncMaster_Ctrl_Slot_G`	is the variable name with the data type short designation "_" for Struct
`CANsync_PLC_MA_BMSTRUCT`	is the data type
`%MB3.2001792`	is the base address of the CANsync interface module on the CANsync master option module in the b maXX's slot G

Inputs si_SL_NR_CTRL, x_EN_CTRL:

At input si_SL_NR_CTRL, you state the slave number of the CANsync slave for which the system is to carry out polling for an existing send control word job (FB CANsync_CONTROLWORD_MA).

Entering si_SL_NR_CTRL = -128 tells the system in every CANsync interval to automatically increment by 1 the slave number of the CANsync slave for which the system is to carry out polling for an existing send control word job until si_MAX_SL_NR is reached. After this, the system starts with polling for the CANsync slave with slave number 0, etc.

The default setting is si_SL_NR_CTRL = -128, i.e. automatic incrementing until si_MAX_SL_NR.

The system only applies the setting at si_SL_NR_CTRL (even if it is not assigned) if x_EN_CTRL = TRUE. If x_EN_CTRL = FALSE, the system does not change this part of the command channel configuration.

Inputs si_SL_NR_PAR, x_EN_PAR:

At input si_SL_NR_PAR , you state the slave number of the CANsync slave for which the system is to carry out polling for an existing parameter job (FB CANsync_PAR_WRITE_MA or CANsync_PAR_READ_MA).

Entering si_SL_NR_CTRL = -128 tells the system in every CANsync interval to automatically increment by 1 the slave number of the CANsync slave for which the system is to carry out polling for an existing send parameter job until si_MAX_SL_NR is reached. After this, the system starts with polling for the CANsync slave with slave number 0, etc.

The default setting is si_SL_NR_PAR = -128, i.e. automatic incrementing until si_MAX_SL_NR.

The system only applies the setting at si_SL_NR_PAR (even if it is not assigned) if x_EN_PAR = TRUE. If x_EN_PAR = FALSE, the system does not change this part of the command channel configuration.

Inputs si_SL_NR_UDL, x_EN_UDL:

At input si_SL_NR_UDL, you state the slave number of the CANsync slave for which the system is to carry out polling for an existing upload/download job (FB CANsync_UPDOWNLOAD_MA).

Entering si_SL_NR_CTRL = -128 tells the system in every CANsync interval to automatically increment by 1 the slave number of the CANsync slave for which the system is to carry out polling for an existing upload/download job until si_MAX_SL_NR is reached. After this, the system starts with polling for the CANsync slave with slave number 0, etc.

The default setting is si_SL_NR_UDL = -128, i.e. automatic incrementing until si_MAX_SL_NR.

The system only applies the setting at si_SL_NR_UDL (even if it is not assigned) if x_EN_PAR = TRUE. If x_EN_UDL = FALSE, the system does not change this part of the command channel configuration.

Input si_MAX_SL_NR:

You state the highest slave number of a CANsync slave at input si_MAX_SL_NR. The system keeps polling up to this slave number whether send control word jobs, parameter jobs and upload/download jobs are available. For this, the system sets si_SL_NR_CTRL, si_SL_NR_PAR, si_SL_UDL not assigned and si_EN_CTRL, si_EN_PAR and si_EN_UDL to TRUE.

If si_MAX_SL_NR = -1, the value remains unchanged on the CANsync interface module. The default setting is si_MAX_SL_NR = -1, i.e. the value remains unchanged on the CANsync interface module.

NOTE

This input corresponds to input si_MAX_SL_NR on FB CANsync_PD_COMM_MA. This means that at input si_MAX_SL_NR on FB CANsync_COMM_CONTROL_MA **or** input si_MAX_SL_NR on FB CANsync_PD_COMM_MA, you state the highest slave number of a CANsync slave.

You must use only one of the two inputs!

4.7 CANsync_CONTROLWORD_MA

Description

You can use this function block for CANsync to receive a control word command of the CANsync master to a CANsync slave.

NOTE

To use this FB it is necessary for the command channel (CC) to release the control word command.
The system uses FB CANsync_COMM_CONTROL_MA to carry out releasing.

FB CANsync_CONTROLWORD_MA uses library BM_TYPES_20bd03 or above.

Parameter input	Data type	Description
_BASE	CANsync_PLC_MA_BMSTRUCT	Operating data for the CANsync interface module
si_SL_NR_CTRL	SINT 0 to 31	Slave number of the CANsync slaves to which the control word is to be sent
w_CONTROLWORD	WORD	Control word
x_EN	BOOL	Enable

Parameter output	Data type	Description
_BASE	CANsync_PLC_MA_BMSTRUCT	Operating data for the CANsync interface module
x_OK	BOOL	OK bit

The system transfers to the CANsync interface module the control word (w_CONTROLWORD), the enable for sending (x_EN) and the slave number of the CANsync slave to which the control word is to be sent (si_SL_NR_CTRL). After the CANsync interface module has detected the job (see FB CANsync_COMM_CONTROL_MA), the system sends the control word at the CANsync slave using a control word message frame and acknowledges sending at output x_OK .

Input/output _BASE:

At input/output `_BASE`, global variable `_CANsyncMaster_Ctrl_Slot_G` (to `_CANsyncMaster_Ctrl_Slot_M`) is connected in accordance with the slot (G to M) of the option module.

If this global variable is not present in the project, create – in dependence on the slot (G to M) – the global variable `_CANsyncMaster_Ctrl_Slot_G` (to `_CANsyncMaster_Ctrl_Slot_M`) of data type `CANsync_PLC_MA_BMSTRUCT`.

Using the declaration of global variables, this variable must be connected to the base address of the CANsync interface module on the option module.

In dependence on the slot, the base address of the CANsync interface module on the CANsync master option module (BM4-O-CAN-06) is as follows:

Slot	Base address of CANsync interface module
G	%MB3.2001792
H	%MB3.3001792
J	%MB3.4001792
K	%MB3.5001792
L	%MB3.6001792
M	%MB3.7001792

Example:

CANsync interface module on the CANsync master option module in the b maXX's slot G

```
_CANsyncMaster_Ctrl_Slot_G  AT  %MB3.2001792  :
                                    CANsync_PLC_MA_BMSTRUCT;
```

Where:

`_CANsyncMaster_Ctrl_Slot_G`	is the variable name with the data type short designation "_" for Struct
`CANsync_PLC_MA_BMSTRUCT`	is the data type
`%MB3.2001792`	is the base address of the CANsync interface module on the CANsync master option module in the b maXX's slot G

Input w_CONTROLWORD:

You connect the control word that is to be sent at input w_CONTROLWORD.

Input x_EN:

If Input x_EN is set to TRUE, the system enters the control word for sending. The CANsync interface module then sends the control word in the control word message frame to CANsync slave si_SL_NR_CTRL. While x_EN stays TRUE, the system enters the control word for sending every time the FB is called. The system carries this out again every time the FB is called while x_EN stays TRUE. However, this would mean that the CANsync's command channel would be busy all the time. You should therefore set input x_EN to TRUE for only one CANsync interval to send the control word once.

Output x_OK:

Output x_OK indicates by TRUE that the last control word message frame has been sent. Output x_OK is FALSE if no control word message frame has been sent.

One reason why the system does not send a control word message frame can be that the command channel is busy with higher-priority messages (broadcast commands; see ▷the operating instructions of the CANsync master option module for b maXX PLC,◁ in the section entitled command channel). Another possible cause is that the command channel is not released for control word commands (see FB ▷CANsync_COMM_CONTROL_MA◁ from page 113 onward).

4.8 CANsync_CONTROLWORD_SL

Description

You can use this function block for CANsync to receive a control word of the CANsync master in a CANsync slave interface module.

NOTE

FB CANsync_CONTROLWORD_SL uses library BM_TYPES_20bd03 or above.

Parameter input	Data type	Description
_BASE	CANsync_PLC_SL_BMSTRUCT	Operating data for the CANsync interface module

Parameter output	Data type	Description
_BASE	CANsync_PLC_SL_BMSTRUCT	Operating data for the CANsync interface module
w_CONTROLWORD	WORD	Control word
si_RECEIVED	SINT 0, 2	Display indicating that a control word was received

If the CANsync slave receives a control word message frame from the CANsync master, FB CANsync_CONTROLWORD_SL outputs the received control word at output w_CONTROLWORD.

Input/output _BASE:

At input/output `_BASE`, global variable `_CANsyncSlave_Ctrl_Slot_G` (to `_CANsyncSlave_Ctrl_Slot_M`) is connected in accordance with the slot (G to M) of the option module.

If this global variable is not present in the project, create – in dependence on the slot (G to M) – the global variable `_CANsyncSlave_Ctrl_Slot_G` (to `_CANsyncSlave_Ctrl_Slot_M`) of data type `CANsync_PLC_SL_BMSTRUCT`.

Using the declaration of global variables, this variable must be connected to the base address of the CANsync interface module on the option module.

In dependence on the slot, the base address of the CANsync interface module on the CANsync slave option module for b maXX PLC (BM4-O-CAN-05, DIP switch 9 = ON) is as follows:

Slot	Base address of CAN-sync interface module
G	%MB3.2001792
H	%MB3.3001792
J	%MB3.4001792
K	%MB3.5001792
L	%MB3.6001792
M	%MB3.7001792

Example:

CANsync interface module on the CANsync slave option module in the b maXX's slot G

```
_CANsyncSlave_Ctrl_Slot_G  AT  %MB3.2001792  :
                                    CANsync_PLC_SL_BMSTRUCT;
```

Where:

`_CANsyncSlave_Ctrl_Slot_G`	is the variable name with the data type short designation "_" for Struct
`CANsync_PLC_SL_BMSTRUCT`	is the data type
`%MB3.2001792`	is the base address of the CANsync interface module on the CANsync slave option module in the b maXX's slot G

Output w_CONTROLWORD:

The system outputs the received control word at output w_CONTROLWORD.

Output si_RECEIVED:

The system displays at output si_RECEIVED whether a control word message frame has been received. In this case, the output displays a 2. Otherwise the system displays a 0 at output si_RECEIVED.

The control word message frame is a special case of the broadcast message frame (with broadcast command number 1).

4.9 CANsync_INIT

Description

You can use this function block for CANsync to initialize a CANsync interface module (CANsync master or CANsync slave).

NOTE

FB CANsync_INIT uses library BM_TYPES_20bd03 or above.

Parameter input	Data type	Description
_BASE	CANsync_PLC_INIT_BMSTRUCT	Initialization data for the CANsync interface module
x_SL	BOOL	Selection of CANsync slave/CANsync master
x_SYNC_IN	BOOL	Configuration of send/receive SYNC signal
x_SYNC_MODE	BOOL	Set up synchronous operating mode
x_ASYNC_MODE0	BOOL	Reserved
x_ASYNC_MODE1	BOOL	Reserved
a_SL_TYP	BYTE_32_BMARRAY	Initialization data of slave types
b_ACCEPT_CODE	BYTE	Acceptance code
b_ACCEPT_MASK	BYTE	Acceptance mask
b_BIT_TIMING0	BYTE	Bus timing 0
b_BIT_TIMING1	BYTE	Bus timing 1
us_BAUDRATE	USINT 3, 4, 5, 6	Baud Rate
us_SYNC_INTERVAL	USINT 8, 4, 2, 1	CANsync interval (cycle scheme) in ms

Parameter output	Data type	Description
_BASE	CANsync_PLC_INIT_BMSTRUCT	Initialization data for the CANsync interface module
w_ERR	WORD	Error word
x_ERR	BOOL	Error bit
x_OK	BOOL	OK bit

FB CANsync_INIT offers several configuration options for initializing a CANsync interface module. You use the FB when initializing a CANsync master interface module or a CANsync slave interface module.

Input/output _BASE:

At **initialization of the CANsync interface module on the CANsync master option module for b maXX PLC** connect at input/output `_BASE` global variable `_CANsyncMaster_Init_Slot_G` (to `_CANsyncMaster_Init_Slot_M`) in accordance with the slot (G to M) of the option module.

If this global variable is not present in the project, create – in dependence on the slot (G to M) – the global variable `_CANsyncMaster_Init_Slot_G` (to `_CANsyncMaster_Init_Slot_M`) of data type `CANsync_PLC_INIT_BMSTRUCT`.

Using the declaration of global variables, this variable must be connected to the base address of the CANsync interface module on the option module.

In dependence on the slot, the base address of the CANsync interface module on the CANsync master option module (BM4-O-CAN-06) is as follows:

Slot	Base address of CANsync interface module
G	%MB3.2001792
H	%MB3.3001792
J	%MB3.4001792
K	%MB3.5001792
L	%MB3.6001792
M	%MB3.7001792

Example:

CANsync interface module on the CANsync master option module in the b maXX's slot G

```
_CANsyncMaster_Init_Slot_G  AT  %MB3.2001792  :
                        CANsync_PLC_INIT_BMSTRUCT;
```

Where:

`_CANsyncMaster_Init_Slot_G`	is the variable name with the data type short designation "_" for Struct
`CANsync_PLC_INIT_BMSTRUCT`	is the data type
`%MB3.2001792`	is the base address of the CANsync interface module on the CANsync master option module in the b maXX's slot G

At **initialization of the CANsync interface module on the CANsync slave option module for b maXX PLC** connect at input/output `_BASE` global variable `_CANsyncSlave_Init_Slot_G` (to `_CANsyncSlave_Init_Slot_M`) in accordance with the slot (G to M) of the option module.

If this global variable is not present in the project, create – in dependence on the slot (G to M) – the global variable `_CANsyncSlave_Init_Slot_G` (to `_CANsyncSlave_Init_Slot_M`) of data type `CANsync_PLC_INIT_BMSTRUCT`.

Using the declaration of global variables, this variable must be connected to the base address of the CANsync interface module on the option module.

In dependence on the slot, the base address of the CANsync interface module on the CANsync slave option module for b maXX PLC (BM4-O-CAN-05, DIP switch 9 = ON) is as follows:

Slot	Base address of CANsync interface module
G	%MB3.2001792
H	%MB3.3001792
J	%MB3.4001792
K	%MB3.5001792
L	%MB3.6001792
M	%MB3.7001792

Example:

CANsync interface module on the CANsync slave option module in the b maXX's slot G

```
_CANsyncSlave_Init_Slot_G  AT  %MB3.2001792  :
                                    CANsync_PLC_INIT_BMSTRUCT;
```

Where:

`_CANsyncSlave_Init_Slot_G`	is the variable name with the data type short designation "_" for Struct
`CANsync_PLC_INIT_BMSTRUCT`	is the data type
`%MB3.2001792`	is the base address of the CANsync interface module on the CANsync slave option module for b maXX PLC in the b maXX's slot G

Input x_SL:

At input x_SL, you choose whether the system is to initialize the CANsync interface module as a master or a slave.

If x_SL = FALSE, the system initializes the CANsync interface module as a master; if x_SL = TRUE, the system initializes the CANsync interface module as a slave.

Input x_SYNC_IN:

This input is only assigned if the CANsync interface module is initialized as a master.

A CANsync master interface module can generate its own SYNC signal; you can, however, set the CANsync master interface module such that it uses an external SYNC signal (e.g. from a CANsync slave interface module in the same unit) as the SYNC signal.

If you are only operating the CANsync master interface module, it must generate its own SYNC signal and x_SYNC_IN stays FALSE.

When x_SYNC_IN = FALSE, the CANsync master interface module generates its own SYNC signal; when x_SYNC_IN = TRUE, the CANsync master interface module takes an external SYNC signal (e.g. the CANsync slave interface module's). You may need to make additional settings to be able to do this; refer to the technical descriptions in each case for more information.

The default setting is x_SYNC_IN = FALSE, i.e. if x_SYNC_IN is not assigned, the CANsync master interface module generates its own SYNC signal.

Inputs x_SYNC_MODE, x_ASYNC_MODE0, x_ASYNC_MODE1:

You set the CANsync interface module's operating mode at these three inputs. Only one of the three inputs may be TRUE.

Setting x_SYNC_MODE = TRUE sets synchronous operation.

Inputs x_ASYNC_MODE0 and x_ASYNC_MODE1 are reserved and are not assigned.

For an explanation of the operating mode, see ▹General ◃ from page 15 onward.

If all three inputs are FALSE, the system only transfers the initialization data to the CANsync interface module and does not set an operating mode. You can set the operating mode using FB CANsync_MODE_MA (CANsync master interface module) or CANsync_MODE_SL (CANsync slave interface module).

NOTE

The operating mode is enabled using FB CANsync_MODE_MA or CANsync_MODE_SL .

Input a_SL_TYP:

This input is only assigned if the CANsync interface module is initialized as a master. Here, you state which CANsync slaves are connected to the CANsync bus. You can also do this using FB CANsync_SL_TYP_INIT.

A variable of data type BYTE_32_BMARRAY is connected at input a_SL_TYP. Data type BYTE_32_BMARRAY is a field of 32 entries of data type byte:

```
BYTE_32_BMARRAY   : ARRAY [0..31] OF BYTE;
```

Example:

```
a_Slave_Typen     : BYTE_32_BMARRAY;
```

Where:

`a_Slave_Typen`	is the variable name with the data type short designation "a" for ARRAY
`BYTE_32_BMARRAY`	is the data type.

In the individual entries of the field, you enter the slave type of the CANsync slave on the CANsync bus. In entry [0], there is the slave type of the CANsync slave with slave number 0; in entry [1] there is the slave type of the CANsync slave with slave number 1, etc.

A 0 in entry [x] means that there is no CANsync slave with slave number x on the CANsync bus.

A value ≠0 in entry [x] means that there is one CANsync slave with slave number x on the CANsync bus.

Meanings of the slave types

Slave type	Meaning
0	No CANsync slave present with slave number x
1	b maXX present with CANsync slave option module with slave number x
2 to 255	Reserved

Inputs b_ACCEPT_MASK, b_ACCEPT_CODE:

At inputs b_ACCEPT_MASK and b_ACCEPT_CODE, you can set the acceptance filter of the CANsync interface module. If the inputs are not assigned, this yields default settings b_ACCEPT_MASK = 16#FF and b_ACCEPT_CODE = 16#FF, i.e. all the objects are taken into account.

No other settings are needed with the CANsync.

These inputs are present for reasons of compatibility.

Inputs b_BIT_TIMING0, b_BIT_TIMING1, us_BAUDRATE, us_SYNC_INTERVAL:

At input us_BAUDRATE, you set the baud rate for the CANsync bus. You must set the maximum baud rate that all the nodes on the CANsync bus can "understand".

NOTE

For limitations on the baud rate, refer to the respective technical description.

The bus timing is calculated for four different baud rates and it is transferred to the CANsync interface module by FB CANsync_INIT.

Baud rate	us_BAUDRATE	b_BIT_TIMING0	b_BIT_TIMING1
125 kbps	3	16#47	16#34
250 kbps	4	16#43	16#34
500 kbps	5	16#41	16#34
1 Mbps	6	16#40	16#34

If value us_BAUDRATE is less than 3 or greater than 6, the system sets the baud rate to 125 kbps and sets bit 1 in error word w_ERR to TRUE.

At inputs b_BIT_TIMING0 and b_BIT_TIMING1, you can set individually the bus timing of the CANsync interface module. For the values for this, refer to the respective technical description.

The system applies the settings of these inputs when input us_BAUDRATE = 0.

The system ignores the settings of these inputs when input us_BAUDRATE is assigned with a value from 3 to 6.

The default setting is us_BAUDRATE = 0, i.e. if us_BAUDRATE is not assigned, the system applies the settings of b_BIT_TIMING0 and b_BIT_TIMING1.

NOTE

The system only applies the values at inputs b_BIT_TIMING0 and b_BIT_TIMING1 if input us_BAUDRATE = 0 or it is not assigned.

At input us_SYNC_INTERVAL, you state the duration in milliseconds of the CANsync interval and the CANsync cycle time.

Together with input us_BAUDRATE, the following combinations are allowed:

CANsync baud rate and CANsync interval duration	us_BAUDRATE	us_SYNC_INTERVAL
1 Mbps and 1 ms	6	1
500 kbps and 2 ms	5	2
250 kbps and 4 ms	4	4
125 kbps and 8 ms	3	8

Output x_OK:

The system sets output x_OK to TRUE if the CANsync interface module was initialized successfully. Output x_OK stays FALSE if the CANsync interface module was not initialized or an error occurred at initialization.

Outputs x_ERR, w_ERR:

If an error occurs, the system sets error bit x_ERR to TRUE and outputs error word b_ERR. In this case output x_OK stays FALSE.

Error word w_ERR:

Bit No.	Error
0	Timeout handshaking with the CANsync interface module
1	Input error with b_BIT_TIMING0, b_BIT_TIMING1 or us_BAUDRATE
2 to 10	Reserved
11	Initialization of CANsync interface module not completed
12 to 15	Reserved

4.10 CANsync_MODE_MA

Description

You can use this function block for CANsync to set the operating mode of a CANsync interface module.

NOTE

FB CANsync_MODE_MA uses library BM_TYPES_20bd03 or above.

Parameter input	Data type	Description
_BASE	CANsync_PLC_MA_BMSTRUCT	Operating data for the CANsync interface module
x_RESET_SOFTWARE	BOOL	Software reset
x_SET_INIT_DATA	BOOL	Take over initialization data
x_CANsync_RUN	BOOL	Enable active operation
x_RESET_CANsync_CONTROLLER	BOOL	Reset CANsync controller (bus-off reset)
x_SYNC_MODE	BOOL	Set up synchronous operating mode
x_ASYNC_MODE0	BOOL	Reserved
x_ASYNC_MODE1	BOOL	Reserved

Parameter output	Data type	Description
_BASE	CANsync_PLC_MA_BMSTRUCT	Operating data for the CANsync interface module
x_HS_ACTIV	BOOL	Handshake is active
x_INIT_POSSIBLE	BOOL	Initialization possible
x_WAIT	BOOL	Waiting for command for setting the operating mode
x_PREPARE_ACTIV	BOOL	Setting the operating mode is active
x_CANsync_ACTIV	BOOL	Operation active
x_SYNC_MODE_ACTIV	BOOL	Synchronous operation
x_ASYNC_MODE_ACTIV	BOOL	Asynchronous operation
x_SL	BOOL	Slave operation
x_MA	BOOL	Master operation

FB CANsync_MODE_MA makes it possible to set the operating modes on the CANsync interface module. The inputs correspond to commands. The outputs display the current actual status. The signals are each active when they are TRUE. If all the inputs are FALSE, the system does not execute any commands and the last status stays active.

NOTE

FB CANsync_MODE_MA does not wait for the CANsync interface module's status message. This means that if the FB is called in the cold and warm boot task, the outputs may not be set. If you need a display of the status, the FB must be called again. The inputs must then be set to FALSE. The CANsync interface module's status is then displayed at the outputs.

Use in the cold and warm boot task:

It is possible to start an operating mode. To do this, you set an operating mode (e.g. set x_SYNC_MODE to TRUE for synchronous operation).

The set operating mode is started with x_CANsync_RUN = TRUE.

Use in the cyclical program:

The CANsync interface module can be reinitialized. To do this, you reset the CANsync interface module with x_RESET_SOFTWARE = TRUE (FB CANsync_MODE_MA).

After this, the system carries out reinitialization of the CANsync interface module (FB CANsync_INIT) and sets and enables an operating mode (FB CANsync_MODE_MA).

Input/output _BASE:

At input/output `_BASE`, global variable `_CANsyncMaster_Ctrl_Slot_G` (to `_CANsyncMaster_Ctrl_Slot_M`) is connected in accordance with the slot (G to M) of the option module.

If this global variable is not present in the project, create – in dependence on the slot (G to M) – the global variable `_CANsyncMaster_Ctrl_Slot_G` (to `_CANsyncMaster_Ctrl_Slot_M`) of data type `CANsync_PLC_MA_BMSTRUCT`.

Using the declaration of global variables, this variable must be connected to the base address of the CANsync interface module on the option module.

In dependence on the slot, the base address of the CANsync interface module on the CANsync master option module (BM4-O-CAN-06) is as follows:

Slot	Base address of CANsync interface module
G	%MB3.2001792
H	%MB3.3001792
J	%MB3.4001792
K	%MB3.5001792
L	%MB3.6001792
M	%MB3.7001792

Example:

CANsync interface module on the CANsync master option module in the b maXX's slot G

```
_CANsyncMaster_Ctrl_Slot_G  AT  %MB3.2001792  :
                                        CANsync_PLC_MA_BMSTRUCT;
```

Where:

`_CANsyncMaster_Ctrl_Slot_G`	is the variable name with the data type short designation "_" for Struct
`CANsync_PLC_MA_BMSTRUCT`	is the data type
`%MB3.2001792`	is the base address of the CANsync interface module on the CANsync master option module in the b maXX's slot G

Input x_RESET_CANsync_CONTROLLER:

With x_RESET_CANsync_CONTROLLER = TRUE, you reset the CANsync controller. This causes the CANsync controller to leave BUS-OFF status and it can be active again on the CANsync bus.

Input x_SET_INIT_DATA:

With x_SET_INIT_DATA = TRUE, the CANsync interface module applies new initialization data.

This input is only needed if you explicitly program initialization without FB CANsync_INIT.

Input x_CANsync_RUN:

With x_CANsync_RUN = TRUE, you activate the operating mode set under x_SYNC_MODE, x_ASYNC_MODE0 or x_ASYNC_MODE1.

Inputs x_SYNC_MODE, x_ASYNC_MODE0, x_ASYNC_MODE1:

You set the CANsync interface module's operating mode at these three inputs. Only one of the three inputs may be TRUE.

Setting x_SYNC_MODE = TRUE sets synchronous operation.

Inputs x_ASYNC_MODE0 and x_ASYNC_MODE1 are reserved and stay FALSE.

After setting the operating mode, you use x_CANsync_RUN = TRUE to enable active operation.

The system issues a TRUE checkback signal at the outputs. Otherwise, the outputs are FALSE.

Output x_HS_ACTIV:

Output x_HS_ACTIV indicates with TRUE that handshaking is active. This is used by FB CANsync_INIT.

Output x_INIT_POSSIBLE:

Output x_INIT_POSSIBLE indicates with TRUE that the CANsync interface module is in the initialization status. The module can then receive new initialization data or the command for setting the operating mode.

Output x_WAIT:

Output x_WAIT indicates with TRUE that the CANsync interface module has applied the initialization data and is waiting for the command for setting the operating mode.

Output x_PREPARE_ACTIV:

Output x_PREPARE_ACTIV indicates with TRUE that an operating mode is being set up.

Output x_CANsync_ACTIV:

Output x_CANsync_ACTIV indicates with TRUE that an operating mode is active.

Output x_SYNC_MODE_ACTIV:

Output x_SYNC_MODE_ACTIV indicates with TRUE that synchronous operation has been set up.

Output x_ASYNC_MODE_ACTIV:

Output x_ASYNC_MODE_ACTIV indicates with TRUE that asynchronous operation (Mode 0 or Mode 1) has been set up.

Output x_SL:

Output x_SL indicates with TRUE that the CANsync interface module has been configured as a slave.

Output x_MA:

Output x_MA indicates with TRUE that the CANsync interface module has been configured as a master.

It is also possible to set combinations of outputs. If x_CANsync_ACTIV, x_SYNC_MODE_ACTIV and x_MA are set to TRUE, for example, this means that the CANsync interface module is a master in active synchronous operation.

4.11 CANsync_MODE_SL

Description

You can use this function block for CANsync to set the operating mode of a CANsync interface module.

NOTE

FB CANsync_MODE_SL uses library BM_TYPES_20bd03 or above.

Parameter input	Data type	Description
_BASE	CANsync_PLC_SL_BMSTRUCT	Operating data for the CANsync interface module
x_RESET_SOFTWARE	BOOL	Software reset
x_SET_INIT_DATA	BOOL	Take over initialization data
x_CANsync_RUN	BOOL	Enable active operation
x_RESET_CANsync_CONTROLLER	BOOL	Reset CANsync controller (bus-off reset)
x_SYNC_MODE	BOOL	Set up synchronous operating mode
x_ASYNC_MODE0	BOOL	Reserved
x_ASYNC_MODE1	BOOL	Reserved

Parameter output	Data type	Description
_BASE	CANsync_PLC_SL_BMSTRUCT	Operating data for the CANsync interface module
x_HS_ACTIV	BOOL	Handshake is active
x_INIT_POSSIBLE	BOOL	Initialization possible
x_WAIT	BOOL	Waiting for command for setting the operating mode
x_PREPARE_ACTIV	BOOL	Setting the operating mode is active
x_CANsync_ACTIV	BOOL	Operation active
x_SYNC_MODE_ACTIV	BOOL	Synchronous operation
x_ASYNC_MODE_ACTIV	BOOL	Asynchronous operation
x_SL	BOOL	Slave operation
x_MA	BOOL	Master operation

FB CANsync_MODE_SL makes it possible to set the operating modes on the CANsync interface module. The inputs correspond to commands. The outputs display the current actual status. The signals are each active when they are TRUE. If all the inputs are FALSE, the system does not execute any commands and the last status stays active.

NOTE

FB CANsync_MODE_SL does not wait for the CANsync interface module's status message. This means that if the FB is called in the cold and warm boot task, the outputs may not be set. If you need a display of the status, the FB must be called again. The inputs must then be set to FALSE. The CANsync interface module's status is then displayed at the outputs.

Use in the cold and warm boot task:

It is possible to start an operating mode. To do this, you set an operating mode (e.g. set x_SYNC_MODE to TRUE for synchronous operation).

The set operating mode is started with x_CANsync_RUN = TRUE.

Use in the cyclical program:

The CANsync interface module can be reinitialized. To do this, you reset the CANsync interface module with x_RESET_SOFTWARE = TRUE (FB CANsync_MODE_SL).

After this, the system carries out reinitialization of the CANsync interface module (FB CANsync_INIT) and sets and enables an operating mode (FB CANsync_MODE_SL).

Input/output _BASE:

At input/output `_BASE`, global variable `_CANsyncSlave_Ctrl_Slot_G` (to `_CANsyncSlave_Ctrl_Slot_M`) is connected in accordance with the slot (G to M) of the option module.

If this global variable is not present in the project, create – in dependence on the slot (G to M) – the global variable `_CANsyncSlave_Ctrl_Slot_G` (to `_CANsyncSlave_Ctrl_Slot_M`) of data type `CANsync_PLC_SL_BMSTRUCT`.

Using the declaration of global variables, this variable must be connected to the base address of the CANsync interface module on the option module.

In dependence on the slot, the base address of the CANsync interface module on the CANsync slave option module for b maXX PLC (BM4-O-CAN-05, DIP switch 9 = ON) is as follows:

Slot	Base address of CANsync interface module
G	%MB3.2001792
H	%MB3.3001792
J	%MB3.4001792

Slot	Base address of CAN-sync interface module
K	%MB3.5001792
L	%MB3.6001792
M	%MB3.7001792

Example:

CANsync interface module on the CANsync slave option module in the b maXX's slot G

```
_CANsyncSlave_Ctrl_Slot_G  AT  %MB3.2001792  :
                                    CANsync_PLC_SL_BMSTRUCT;
```

Where:

`_CANsyncSlave_Ctrl_Slot_G`	is the variable name with the data type short designation "_" for Struct
`CANsync_PLC_SL_BMSTRUCT`	is the data type
`%MB3.2001792`	is the base address of the CANsync interface module on the CANsync slave option module in the b maXX's slot G

Input x_RESET_CANsync_CONTROLLER:

With x_RESET_CANsync_CONTROLLER = TRUE, you reset the CANsync controller. This causes the CANsync controller to leave BUS-OFF status and it can be active again on the CANsync bus.

Input x_SET_INIT_DATA:

With x_SET_INIT_DATA = TRUE, the CANsync interface module applies new initialization data. This input is only needed if you explicitly program initialization without FB CANsync_INIT.

Input x_CANsync_RUN:

With x_CANsync_RUN = TRUE, you activate the operating mode set under x_SYNC_MODE, x_ASYNC_MODE0 or x_ASYNC_MODE1.

Inputs x_SYNC_MODE, x_ASYNC_MODE0, x_ASYNC_MODE1:

You set the CANsync interface module's operating mode at these three inputs. Only one of the three inputs may be TRUE.

Setting x_SYNC_MODE = TRUE sets synchronous operation.

Inputs x_ASYNC_MODE0 and x_ASYNC_MODE1 are reserved and stay FALSE.

After setting the operating mode, you use x_CANsync_RUN = TRUE to enable active operation.

The system issues a TRUE checkback signal at the outputs. Otherwise, the outputs are FALSE.

Output x_HS_ACTIV:

Output x_HS_ACTIV indicates with TRUE that handshaking is active. This is used by FB CANsync_INIT.

Output x_INIT_POSSIBLE:

Output x_INIT_POSSIBLE indicates with TRUE that the CANsync interface module is in the initialization status. The module can then receive new initialization data or the command for setting the operating mode.

Output x_WAIT:

Output x_WAIT indicates with TRUE that the CANsync interface module has applied the initialization data and is waiting for the command for setting the operating mode.

Output x_PREPARE_ACTIV:

Output x_PREPARE_ACTIV indicates with TRUE that an operating mode is being set up.

Output x_CANsync_ACTIV:

Output x_CANsync_ACTIV indicates with TRUE that an operating mode is active.

Output x_SYNC_MODE_ACTIV:

Output x_SYNC_MODE_ACTIV indicates with TRUE that synchronous operation has been set up.

Output x_ASYNC_MODE_ACTIV:

Output x_ASYNC_MODE_ACTIV indicates with TRUE that asynchronous operation (Mode 0 or Mode 1) has been set up.

Output x_SL:

Output x_SL indicates with TRUE that the CANsync interface module has been configured as a slave.

Output x_MA:

Output x_MA indicates with TRUE that the CANsync interface module has been configured as a master.

It is also possible to set combinations of outputs. If x_CANsync_ACTIV, x_SYNC_MODE_ACTIV and x_SL are set to TRUE, for example, this means that the CANsync interface module is a slave in active synchronous operation.

4.12 CANsync_PAR_READ_MA

Description

You can use this function block for CANsync to request from the CANsync master via CANsync a parameter value from the CANsync slave.

NOTE

You can instantiate this FB several times if different CANsync slaves are addressed in each case.

To use this FB it is necessary for the command channel (CC) to release the parameter commands.
The system uses FB CANsync_COMM_CONTROL_MA to carry out releasing.

FB CANsync_PAR_READ_MA uses library BM_TYPES_20bd03 or above.

Parameter input	Data type	Description
_BASE	CANsync_PLC_MA_BMSTRUCT	Operating data for the CANsync interface module
si_SL_NR	SINT 0 to 31	Slave number of the CANsync slave to which the read parameter job is addressed
u_PAR_NR	UINT	Parameter number
x_PAR_FORMAT	BOOL	Parameter format
i_SUB_SL	INT 0 to 31	Sub-slave address (reserved)
t_TIME	TIME	Monitoring time
x_EN	BOOL	Enable
x_RESET	BOOL	Reset

Parameter output	Data type	Description
_BASE	CANsync_PLC_MA_BMSTRUCT	Operating data for the CANsync interface module
ud_PAR_VALUE	UDINT	Read parameter value
x_PAR_FORMAT_READ	BOOL	Read parameter format
x_BUSY	BOOL	Communication is active
b_ERR	BYTE	Error byte
i_ERR	INT	Error number
x_ERR	BOOL	Error bit
x_OK	BOOL	OK bit

FB CANsync_PAR_READ_MA transfers with the values of inputs u_PAR_NR, x_PAR_FORMAT and i_SUB_SL a read parameter job to the CANsync slave with slave number si_SL_NR. The CANsync slave processes the read parameter job and returns the result of communication. The read parameter value is output at output ud_PAR_VALUE.

Input/output _BASE:

At input/output `_BASE`, global variable `_CANsyncMaster_Ctrl_Slot_G` (to `_CANsyncMaster_Ctrl_Slot_M`) is connected in accordance with the slot (G to M) of the option module.

If this global variable is not present in the project, create – in dependence on the slot (G to M) – the global variable `_CANsyncMaster_Ctrl_Slot_G` (to `_CANsyncMaster_Ctrl_Slot_M`) of data type `CANsync_PLC_MA_BMSTRUCT`.

Using the declaration of global variables, this variable must be connected to the base address of the CANsync interface module on the option module.

In dependence on the slot, the base address of the CANsync interface module on the CANsync master option module (BM4-O-CAN-06) is as follows:

Slot	Base address of CANsync interface module
G	%MB3.2001792
H	%MB3.3001792
J	%MB3.4001792
K	%MB3.5001792
L	%MB3.6001792
M	%MB3.7001792

Example:

CANsync interface module on the CANsync master option module in the b maXX's slot G

```
_CANsyncMaster_Ctrl_Slot_G  AT  %MB3.2001792  :
                                    CANsync_PLC_MA_BMSTRUCT;
```

Where:

`_CANsyncMaster_Ctrl_Slot_G`	is the variable name with the data type short designation "_" for Struct
`CANsync_PLC_MA_BMSTRUCT`	is the data type
`%MB3.2001792`	is the base address of the CANsync interface module on the CANsync master option module in the b maXX's slot G

Input si_SL_NR:

At input si_SL_NR, you state the slave number of the CANsync slave on the CANsync bus from which the parameter value is read.

Input u_PAR_NR:

You state the parameter number for the read parameter job at input u_PAR_NR.

Input x_PAR_FORMAT:

You set the format of the requested parameter value at input x_PAR_FORMAT. x_PAR_FORMAT = FALSE means word format; x_PAR_FORMAT = TRUE means doubleword format.

Input i_SUB_SL:

This input is reserved and is not assigned.

Input t_TIME:

At input t_TIME, you state the monitoring time within which the system is to carry out the read parameter job. If the read parameter job is not completed within the monitoring time, the system sets bit 1 of error byte b_ERR to TRUE.

The default setting for t_TIME is 3 s.

One reason why the system does not completely process a read parameter job can be that the command channel is busy with higher-priority messages (broadcast commands, send control word jobs, etc.) see the technical description of the CANsync master option module for b maXX PLC in the section entitled command channel). Another possible cause is that the command channel is not released for parameter jobs (see FB CANsync_COMM_CONTROL_MA).

Input x_EN:

Communication is started by means of x_EN = TRUE. Input x_EN must not be reset to FALSE until output x_BUSY drops to FALSE after communication is completed. Otherwise, it is assumed that communication was cancelled deliberately and you must reset the FB (x_RESET = TRUE).

Input x_RESET:

FB CANsync_PAR_READ_MA is reset by means of x_RESET = TRUE. This is necessary after aborting communication (by means of x_EN = FALSE) or after an error message, for example. After this, you must set x_RESET back to FALSE.

Output ud_PAR_VALUE:

The system makes available the read parameter value at output ud_PAR_VALUE.

Output x_BUSY:

Output x_BUSY indicates by TRUE that communication is active.

Output x_OK:

Output x_OK is set to TRUE if the read parameter job was executed correctly. Output x_OK is FALSE if the system did not execute a read parameter job or it was not executed correctly.

Outputs x_ERR, b_ERR, i_ERR:

If an error occurs, the system sets error bit x_ERR to TRUE and outputs error byte b_ERR. In this case output x_OK stays FALSE.

If the CANsync slave reports an error number, the system outputs an error number at output i_ERR. The contents of the error number are determined by the application in the CANsync slave.

Error byte b_ERR:

Bit No.	Error
0	Communications error, error number is in i_ERR
1	Timeout
2	Doubleword format (TRUE) was expected at input x_PAR_FORMAT
3	Word format (FALSE) was expected at input x_PAR_FORMAT
4	Invalid slave number of the CANsync slaves on the CANsync bus
5 - 7	Reserved

4.13 CANsync_PAR_SL

Description

You can use this function block for CANsync to detect a read parameter job or a write parameter job. The FB is suitable for use with BACI requirements data FBs BACI_PAR_READ and BACI_PAR_WRITE.

NOTE

FB CANsync_PAR_SL uses library BM_TYPES_20bd03 or above.

Parameter input	Data type	Description
_BASE	CANsync_PLC_SL_BMSTRUCT	Operating data for the CANsync interface module
ud_PAR_VALUE_READ	UDINT	Parameter value (read)
x_PAR_FORMAT_READ	BOOL	Parameter format (read)
i_ERR	INT	Error number (application)
x_ERR_IN	BOOL	Error bit (application)
x_OK_IN	BOOL	OK bit (application)
x_EN	BOOL	Enable
x_RESET	BOOL	Reset

Parameter output	Data type	Description
_BASE	CANsync_PLC_SL_BMSTRUCT	Operating data for the CANsync interface module
x_RESET_JOB	BOOL	Reset (change of job)
x_READ	BOOL	Read parameter job
x_WRITE	BOOL	Write parameter job
u_PAR_NR	UINT	Parameter number
x_PAR_FORMAT_WRITE	BOOL	Parameter format (write)
ud_PAR_VALUE_WRITE	UDINT	Parameter value (write)
x_ACTIV	BOOL	Job logged on, waiting for result of application
x_BUSY	BOOL	Communication is active
x_OK	BOOL	OK bit

FB CANsync_PAR_SL detects a read parameter job or a write parameter job and makes available the respective data at the outputs. The application processes the jobs and transfers the results to the FB via the inputs. The FB then reports the results to the CANsync slave interface module and indicates that the results were sent to the CANsync master.

In the case of a read parameter job, the system indicates the job with x_READ = TRUE and outputs the parameter number at u_PAR_NR. The application expects the parameter value at ud_PAR_VALUE_READ and the parameter format at x_PAR_FORMAT_READ. With x_OK_IN = TRUE, the system takes the parameter value and the parameter format and sends them to the CANsync master. If the application reports an error, the system can connect an error number at i_ERR and set x_ERR_IN = TRUE. In this case, the error number is sent to the CANsync master.

In the case of a write parameter job, the system indicates the job with x_WRITE = TRUE, and outputs the parameter number at u_PAR_NR, the parameter format at x_PAR_FORMAT_WRITE and the parameter value at ud_PAR_VALUE_WRITE. The application expects the result of communication. With x_OK_IN = TRUE, the system reports to the CANsync master that the write parameter job was executed successfully. If the application reports an error, the system can connect an error number at i_ERR and set x_ERR_IN = TRUE. In this case, the error number is sent to the CANsync master.

NOTE

In the b maXX PLC with a CANsync slave option module, it is possible to forward read and write parameter jobs. The FBs of BACI requirements data communication are used for this. In the following section, some of the inputs and outputs have listed in brackets the respective input or output of the FBs of BACI-requirements data communication.

Input/output _BASE:

At input/output `_BASE`, global variable `_CANsyncSlave_Ctrl_Slot_G` (to `_CANsyncSlave_Ctrl_Slot_M`) is connected in accordance with the slot (G to M) of the option module.

If this global variable is not present in the project, create – in dependence on the slot (G to M) – the global variable `_CANsyncSlave_Ctrl_Slot_G` (to `_CANsyncSlave_Ctrl_Slot_M`) of data type `CANsync_PLC_SL_BMSTRUCT`.

Using the declaration of global variables, this variable must be connected to the base address of the CANsync interface module on the option module.

In dependence on the slot, the base address of the CANsync interface module on the CANsync slave option module for b maXX PLC (BM4-O-CAN-05, DIP switch 9 = ON) is as follows:

Slot	Base address of CANsync interface module
G	%MB3.2001792
H	%MB3.3001792
J	%MB3.4001792
K	%MB3.5001792
L	%MB3.6001792
M	%MB3.7001792

Example:

CANsync interface module on the CANsync slave option module in the b maXX's slot G

```
_CANsyncSlave_Ctrl_Slot_G  AT  %MB3.2001792  :
                                    CANsync_PLC_SL_BMSTRUCT;
```

Where:

`_CANsyncSlave_Ctrl_Slot_G`	is the variable name with the data type short designation "_" for Struct
`CANsync_PLC_SL_BMSTRUCT`	is the data type
`%MB3.2001792`	is the base address of the CANsync interface module on the CANsync slave option module in the b maXX's slot G

Input ud_PAR_VALUE_READ:

In the case of a read parameter job, the system expects the parameter value at ud_PAR_VALUE_READ. The application program makes available this parameter value. (You can connect input ud_PAR_VALUE_READ via FB DINT_TO_UDINT to FB BACI_PAR_READ, output di_PAR_VALUE.)

Input x_PAR_FORMAT_READ:

In the case of a read parameter job, the system expects the parameter format at x_PAR_FORMAT_READ. With x_PAR_FORMAT_READ = FALSE, you state that the parameter value at ud_PAR_VALUE_READ is of WORD format (16-bit); with x_PAR_FORMAT_READ = TRUE, you state that the parameter value at ud_PAR_VALUE_READ is of DOUBLEWORD format (32-bit).

The application program makes available the parameter format. (Input x_PAR_FORMAT_READ can be connected to FB BACI_PAR_READ, output x_PAR_FORMAT).

Inputs x_ERR_IN, i_ERR:

If the application program cannot read or fulfill the CANsync master's parameter job, it is possible to state an error number at i_ERR and set x_ERR_IN = TRUE. In this case, the error number is sent to the CANsync master.

The application program makes available the error number, i_ERR, and the error bit x_ERR_IN. (Input x_ERR_IN can be linked to FB BACI_PAR_WRITE, output x_ERR and/or FB BACI_PAR_READ, output x_ERR. (Input i_ERR can be linked to FB BACI_PAR_WRITE, output i_ERR_COMM and/or FB BACI_PAR_READ, output i_ERR_COMM. See ▹Figure7 on page 143: No. 1)

Input x_OK_IN:

If the application program has fulfilled the CANsync master's parameter job, the system sets input x_OK_IN = TRUE.

In the case of a read parameter job, the system expects the read parameter value at ud_PAR_VALUE_READ and the format of the read parameter at x_PAR_FORMAT_READ. In the case of a write parameter job, the system does not expect any other values.

The application program must make available the OK bit. (Input x_OK can be linked to FB BACI_PAR_WRITE, output x_OK and/or FB BACI_PAR_READ, output x_OK. See ▹Figure7 on page 143: No. 2)

Input x_EN:

FB CANsync_PAR_SL is activated with x_EN = TRUE . The system only reports parameter jobs and sends answers to the CANsync master when the FB is activated.

If FB CANsync_PAR_SL is deactivated (x_EN = FALSE), the system must wait until the last parameter job has been processed and sent to the CANsync master (x_BUSY = FALSE). Otherwise, it is assumed that communication was cancelled deliberately and you must then reset the FB with x_RESET = TRUE.

Input x_RESET:

You can use x_RESET = TRUE to reset the FB. This is necessary after aborting communication (by means of x_EN = FALSE) or after an error message, for example. After this, you must set x_RESET back to FALSE.

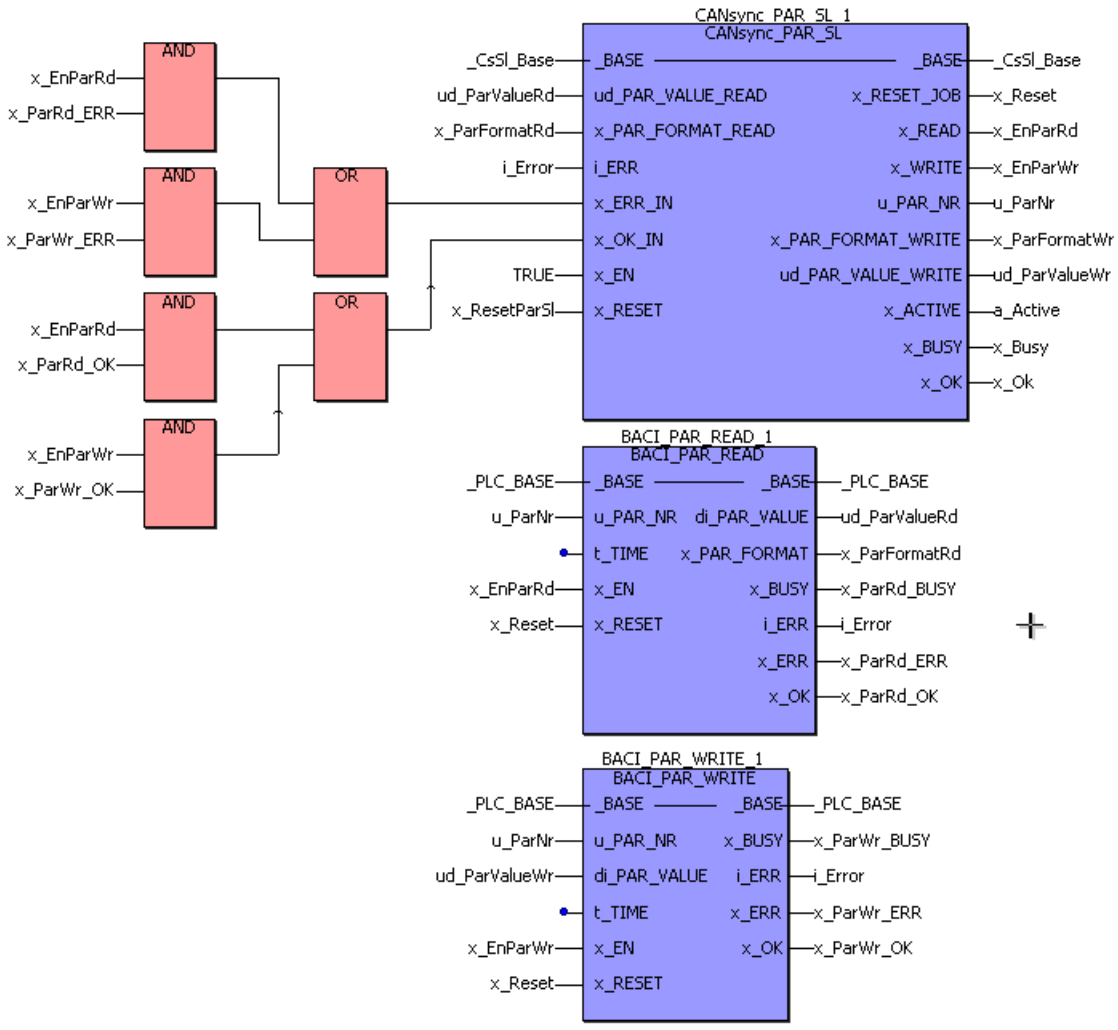


Figure 7: Example of using FB CANsync_PAR_SL in conjunction with FBs BACI_PAR_READ and BACI_PAR_WRITE

Output x_RESET_JOB:

The CANsync master can cancel a parameter job . In this case, the system sets output x_RESET_JOB to TRUE. Output x_RESET_JOB is set back to FALSE when the CANsync master starts a new parameter job. (Output x_RESET_JOB can be linked to FB BACI_PAR_WRITE, input x_RESET and/or FB BACI_PAR_READ, input x_RESET.

Output x_READ:

Output x_READ is set to TRUE if a read parameter job is pending. (Output x_READ can be connected to FB BACI_PAR_READ, input x_EN).

Output x_READ is set to FALSE if no read parameter job is pending.

Output x_WRITE:

Output x_WRITE is set to TRUE if a write parameter job is pending. (Output x_WRITE can be connected to FB BACI_PAR_WRITE, input x_EN). Output x_WRITE is set to FALSE if no write parameter job is pending.

Output u_PAR_NR:

The parameter number of the parameter job is output at output u_PAR_NR. (Output u_PAR_NR can be linked to FB BACI_PAR_WRITE, input u_PAR_NR or FB BACI_PAR_READ, input u_PAR_NR).

Output x_PAR_FORMAT:

The system outputs at output x_PAR_FORMAT the parameter format of parameter u_PAR_NR in the case of a write parameter job.

x_PAR_FORMAT = FALSE means WORD format (16-bit) and x_PAR_FORMAT = TRUE means DOUBLEWORD format (32-bit). (Output x_PAR_FORMAT is of no consequence for FB BACI_PAR_WRITE, since the system automatically detects the parameter format that is stored in the controller when writing parameters via the BACI to the controller.)

Output ud_PAR_VALUE:

The system outputs at output x_PAR_VALUE the parameter value of parameter u_PAR_NR in the case of a write parameter job. (You can connect output ud_PAR_VALUE via FB DINT_TO_UDINT to FB BACI_PAR_WRITE, input di_PAR_VALUE.)

Output x_ACTIV:

Output x_ACTIV indicates with TRUE that FB CANsync_PAR_SL is waiting during a parameter job for the result of the parameter job (input x_OK_IN or x_ERR_IN). Otherwise, output x_ACTIV is set to FALSE.

Output x_BUSY:

Output x_BUSY indicates with TRUE that FB CANsync_PAR_SL is waiting for the result of the read or write parameter job and that the answer to the CANsync master is ready but the CANsync master has not requested it. Otherwise, output x_BUSY is set to FALSE.

Output x_OK:

The system sets output x_OK to TRUE if the CANsync master has fetched the answer. In this connection, it does not matter whether the answer in question is the error answer or the answer for correct processing of the job. Output x_OK is FALSE if no parameter job has yet been executed, the parameter job has not been executed or the CANsync master has not fetched the answer.

4.14 CANsync_PAR_WRITE_MA

Description

You can use this function block for CANsync to send a parameter value from the CANsync master via the CANsync bus to the CANsync slave.

NOTE

You can instantiate this FB several times if different CANsync slaves are addressed in each case.

To use this FB it is necessary for the command channel (CC) to release the parameter commands.
The system uses FB CANsync_COMM_CONTROL_MA to carry out releasing.

FB CANsync_PAR_WRITE_MA uses library BM_TYPES_20bd03 or above.

Parameter input	Data type	Description
_BASE	CANsync_PLC_MA_BMSTRUCT	Operating data for the CANsync interface module
si_SL_NR	SINT 0 to 31	Slave number of the CANsync slave to which the write parameter job is addressed
u_PAR_NR	UINT	Parameter number
x_PAR_FORMAT	BOOL	Parameter format
i_SUB_SL	INT 0 to 31	Sub-slave address (reserved)
ud_PAR_VALUE	UDINT	Parameter value to be written
t_TIME	TIME	Monitoring time
x_EN	BOOL	Enable
x_RESET	BOOL	Reset

Parameter output	Data type	Description
_BASE	CANsync_PLC_MA_BMSTRUCT	Operating data for the CANsync interface module
x_BUSY	BOOL	Communication is active
i_ERR	INT	Error number
b_ERR	BYTE	Error byte
x_ERR	BOOL	Error bit
x_OK	BOOL	OK bit

FB CANsync_PAR_WRITE_MA transfers with the values of inputs u_PAR_NR, x_PAR_FORMAT, i_SUB_SL and ud_PAR_VALUE a write parameter job to the CANsync slave with slave number si_SL_NR. The CANsync slave processes the write parameter job and returns the result of communication.

Input/output _BASE:

At input/output `_BASE`, global variable `_CANsyncMaster_Ctrl_Slot_G` (to `_CANsyncMaster_Ctrl_Slot_M`) is connected in accordance with the slot (G to M) of the option module.

If this global variable is not present in the project, create – in dependence on the slot (G to M) – the global variable `_CANsyncMaster_Ctrl_Slot_G` (to `_CANsyncMaster_Ctrl_Slot_M`) of data type `CANsync_PLC_MA_BMSTRUCT`.

Using the declaration of global variables, this variable must be connected to the base address of the CANsync interface module on the option module.

In dependence on the slot, the base address of the CANsync interface module on the CANsync master option module (BM4-O-CAN-06) is as follows:

Slot	Base address of CANsync interface module
G	%MB3.2001792
H	%MB3.3001792
J	%MB3.4001792
K	%MB3.5001792
L	%MB3.6001792
M	%MB3.7001792

Example:

CANsync interface module on the CANsync master option module in the b maXX's slot G

```
_CANsyncMaster_Ctrl_Slot_G  AT  %MB3.2001792  :
                                    CANsync_PLC_MA_BMSTRUCT;
```

Where:

`_CANsyncMaster_Ctrl_Slot_G`	is the variable name with the data type short designation "_" for Struct
`CANsync_PLC_MA_BMSTRUCT`	is the data type
`%MB3.2001792`	is the base address of the CANsync interface module on the CANsync master option module in the b maXX's slot G

Input si_SL_NR:

At input si_SL_NR, you state the slave number of the CANsync slave on the CANsync bus to which the parameter value is sent.

Input u_PAR_NR:

You state the parameter number for the write parameter job at input u_PAR_NR.

Input x_PAR_FORMAT:

You set the format of the parameter value to be transferred at input x_PAR_FORMAT. x_PAR_FORMAT = FALSE means word format; x_PAR_FORMAT = TRUE means doubleword format.

Input i_SUB_SL:

This input is reserved and is not assigned.

Input ud_PAR_VALUE:

You state the parameter value to be transferred/written at input ud_PAR_VALUE.

Input t_TIME:

At input t_TIME, you state the monitoring time within which the system is to carry out the write parameter job. If the write parameter job is not completed within the monitoring time, the system sets bit 1 of error byte b_ERR to TRUE.

The default setting for t_TIME is 3 s.

One reason why the system does not completely process a write parameter job can be that the command channel is busy with higher-priority messages (broadcast commands, send control word jobs, etc.) see ▷the operating instructions of the CANsync master option module for b maXX PLC◁ in the section entitled command channel). Another possible cause is that the command channel is not released for parameter jobs (see FB ▷CANsync_COMM_CONTROL_MA ◁ from page 113 onward).

Input x_EN:

Communication is started by means of x_EN = TRUE. Input x_EN must not be reset to FALSE until output x_BUSY drops to FALSE after communication is completed. Otherwise, it is assumed that communication was cancelled deliberately and you must reset the FB (x_RESET = TRUE).

Input x_RESET:

FB CANsync_PAR_WRITE_MA is reset by means of x_RESET = TRUE. This is necessary after aborting communication (by means of x_EN = FALSE) or after an error message, for example. After this, you must set x_RESET back to FALSE.

Output x_BUSY:

Output x_BUSY indicates by TRUE that communication is active. Otherwise, x_BUSY = FALSE.

Output x_OK:

Output x_OK is set to TRUE if the write parameter job was executed correctly. Output x_OK is FALSE if the system did not execute a write parameter job or it was not executed correctly.

Outputs x_ERR, b_ERR, i_ERR:

If an error occurs, the system sets error bit x_ERR to TRUE and outputs error byte b_ERR. In this case output x_OK stays FALSE.

If the CANsync slave reports an error number, the system outputs an error number at output i_ERR. The contents of the error number are determined by the application in the CANsync slave.

Error byte b_ERR:

Bit No.	Error
0	Communications error, error number is in i_ERR
1	Timeout
2, 3	Reserved
4	Invalid slave number of the CANsync slaves on the CANsync bus
5 - 7	Reserved

4.15 CANsync_PD_CFG_MA

Description

You can use this function block für CANsync for CANsync to configure the assignment of the CANsync interface module's reference value message frames for a CANsync master.

NOTE

FB CANsync_PD_CFG_MA uses library BM_TYPES_20bd03 or above.

Parameter input	Data type	Description
_BASE	CANsync_PLC_MA_BMSTRUCT	Operating data for the CANsync interface module
a_WRC1	SINT_4_BMARRAY	Reference value numbers for reference value message frame 1
a_HL_WRC1	BOOL_4_BMARRAY	Assignment of highword or lowword for reference value message frame 1
a_WRC2	SINT_4_BMARRAY	Reference value numbers for reference value message frame 2
a_HL_WRC2	BOOL_4_BMARRAY	Assignment of highword or lowword for reference value message frame 2

Parameter output	Data type	Description
_BASE	CANsync_PLC_MA_BMSTRUCT	Operating data for the CANsync interface module

Using FB CANsync_PD_CFG_MA, you:

- assign eight 32-bit reference values to reference value message frames 1 and 2 (send)

in a CANsync master.

In the CANsync master, the application can write eight 32-bit reference values. The reference value numbers are from 0 to 7.

A reference value is composed of a lowword (bits 0 to 15) and a highword (bit 16 to 31).

In the two reference value message frames, it is possible to send (in each case) 4 * 16-bit data to CANsync slaves. That is four words numbered 0 to 3.

Using FB CANsync_PD_CFG_MA, you specify
which data is to be entered in the four words (words 0 to 3) of reference value message frame 1
and which is to be entered in the four words (words 0 to 3) of reference value message frame 2.

You can choose the data for reference value message frame 1 from reference values 0 to 3 and the data for reference value message frame 2 from reference values 4 to 7.

You can assign to each word in a reference value message frame a lowword or a high-word of a reference value. Two words are needed in the reference value message frame when transferring a 32-bit reference value.

Input/output _BASE:

At input/output `_BASE`, global variable `_CANsyncMaster_Ctrl_Slot_G` (to `_CANsyncMaster_Ctrl_Slot_M`) is connected in accordance with the slot (G to M) of the option module.

If this global variable is not present in the project, create – in dependence on the slot (G to M) – the global variable `_CANsyncMaster_Ctrl_Slot_G` (to `_CANsyncMaster_Ctrl_Slot_M`) of data type `CANsync_PLC_MA_BMSTRUCT`.

Using the declaration of global variables, this variable must be connected to the base address of the CANsync interface module on the option module.

In dependence on the slot, the base address of the CANsync interface module on the CANsync master option module (BM4-O-CAN-06) is as follows:

Slot	Base address of CANsync interface module
G	%MB3.2001792
H	%MB3.3001792
J	%MB3.4001792
K	%MB3.5001792
L	%MB3.6001792
M	%MB3.7001792

Example:

CANsync interface module on the CANsync master option module in the b maXX's slot G

```
_CANsyncMaster_Ctrl_Slot_G  AT  %MB3.2001792  :
                                    CANsync_PLC_MA_BMSTRUCT;
```

Where:

`_CANsyncMaster_Ctrl_Slot_G`	is the variable name with the data type short designation "_" for Struct
`CANsync_PLC_MA_BMSTRUCT`	is the data type
`%MB3.2001792`	is the base address of the CANsync interface module on the CANsync master option module in the b maXX's slot G

Input a_WRC1:

Assignment of

"Reference value (0 to 3)" -> "word in reference value message frame 1" is carried out in a_WRC1.

a_WRC1[word number] := reference value number

Input a_HL_WRC1:

Assignment of

"Lowword or highword of the selected reference value" -> "word in reference value message frame 1" is carried out in a_HL_WRC1.

a_HL_WRC1[word number] := FALSE (if lowword)

a_HL_WRC1[word number] := TRUE (if highword)

Example:

Word number in reference value message frame 1	Selected reference value	Connection at input a_WRC1	Connection at input a_HL_WRC1
0	Reference value 1 lowword	a_WRC1[0] = SINT#1	a_HL_WRC1[0] = FALSE
1	Reference value 1 highword	a_WRC1[1] = SINT#1	a_HL_WRC1[1] = TRUE
2	Reference value 0 word	a_WRC1[2] = SINT#0	a_HL_WRC1[2] = FALSE or open
3	Reference value 2 word	a_WRC1[3] = SINT#2	a_HL_WRC1[3] = FALSE or open

Input a_WRC2:

Assignment of

"Reference value (4 to 7)" -> "word in reference value message frame 2" is carried out in a_WRC2.

a_WRC2[word number] := reference value number

Input a_HL_WRC2:

Assignment of

"Lowword or highword of the selected reference value" -> "word in reference value message frame 2" is carried out in a_HL_WRC2.

a_HL_WRC2[word number] := FALSE (if lowword)

a_HL_WRC2[word number] := TRUE (if highword)

If you do not want to assign a reference value to a word in reference value message frame 1 or 2, enter -1 as the reference value number at the corresponding entry in a_WRC1 or a_WRC2 respectively.

In this case, the corresponding setting in a_HL_WRC1 or a_HL_WRC2 is meaningless.

a_WRC1[word number] := SINT#-1

a_WRC2[word number] := SINT#-1

Example:
No reference value is to be assigned to word 1 of reference value message frame 2.

Word number in reference value message frame 2	Selected reference value	Connection at input a_WRC2	Connection at input a_HL_WRC2
1	None	a_WRC2[1] = SINT# -1	a_HL_WRC2[1] meaningless

Reference value message frames are assigned to reference values in the CANsync slave. By default, this assignment is carried out in a similar way to the CANsync master. The system does not cross-check the assignment in the CANsync master and the CANsync slave, since there are also reasonable applications for assignments that are different.

4.16 CANsync_PD_CFG_READ_MA

Description

You can use this function block für CANsync to configure the assignment of the actual value message frames of the CANsync slave in the CANsync interface module for a CANsync master.

NOTE

FB CANsync_PD_CFG_READ_MA uses library BM_TYPES_20bd03 or above.

Parameter input	Data type	Description
_BASE	CANsync_PLC_MA_BMSTRUCT	Operating data for the CANsync interface module
si_SL_NR	SINT 0 to 31	Slave number of the CANsync slaves from which actual values are received
a_RDC1	SINT_4_BMARRAY	Actual value numbers for actual value message frame 1
a_HL_RDC1	BOOL_4_BMARRAY	Assignment of highword or lowword for actual value message frame 1
a_RDC2	SINT_4_BMARRAY	Actual value numbers for actual value message frame 2
a_HL_RDC2	BOOL_4_BMARRAY	Assignment of highword or lowword for actual value message frame 2

Parameter output	Data type	Description
_BASE	CANsync_PLC_MA_BMSTRUCT	Operating data for the CANsync interface module

Using FB CANsync_PD_CFG_READ_MA, you:

- assign actual value message frames 1 and 2 of a CANsync slave (receive) to eight 32-bit actual values

in a CANsync master.

In the CANsync master, the application can read eight 32-bit reference values of each CANsync slave.

The CANsync slaves have numbers 0 to 31.

The actual value numbers are from 0 to 7.

An actual value is composed of a lowword (bits 0 to 15) and a highword (bit 16 to 31).

In the CANsync master, it is possible to receive for each CANsync slave two actual value message frames (each) containing 4 * 16 bit data. That is four words (in each case) numbered 0 to 3.

Using FB CANsync_PD_CFG_READ_MA, you specify for a CANsync slave the actual values to which data from the four words
(words 0 to 3) of the CANsync slave's actual value message frame 1 is to be assigned and the actual values to which the data from the four words (words 0 to 3) of its actual value message frame 2 is to be assigned.

You can assign the data from actual value message frame 1 to actual values 0 to 3 and the data for actual value message frame 2 to actual values 4 to 7.

You can assign each word in an actual value message frame to only one lowword or highword of an actual value.

When transferring a 32-bit actual value, two words are needed in the actual value message frame (one word from the actual value message frame is assigned to the lowword of an actual value and another word in this actual value message frame is assigned to the highword of this actual value).

Input/output _BASE:

At input/output `_BASE`, global variable `_CANsyncMaster_Ctrl_Slot_G` (to `_CANsyncMaster_Ctrl_Slot_M`) is connected in accordance with the slot (G to M) of the option module.

If this global variable is not present in the project, create – in dependence on the slot (G to M) – the global variable `_CANsyncMaster_Ctrl_Slot_G` (to `_CANsyncMaster_Ctrl_Slot_M`) of data type `CANsync_PLC_MA_BMSTRUCT`.

Using the declaration of global variables, this variable must be connected to the base address of the CANsync interface module on the option module.

In dependence on the slot, the base address of the CANsync interface module on the CANsync master option module (BM4-O-CAN-06) is as follows:

Slot	Base address of CANsync interface module
G	%MB3.2001792
H	%MB3.3001792
J	%MB3.4001792
K	%MB3.5001792
L	%MB3.6001792
M	%MB3.7001792

Example:

CANsync interface module on the CANsync master option module in the b maXX's slot G

```
_CANsyncMaster_Ctrl_Slot_G  AT  %MB3.2001792  :
                                    CANsync_PLC_MA_BMSTRUCT;
```

Where:

`_CANsyncMaster_Ctrl_Slot_G`	is the variable name with the data type short designation "_" for Struct
`CANsync_PLC_MA_BMSTRUCT`	is the data type
`%MB3.2001792`	is the base address of the CANsync interface module on the CANsync master option module in the b maXX's slot G

Input si_SL_NR:

At input si_SL_NR, you state the slave number of the CANsync slave on the CANsync bus whose actual value message frames are to be configured.

Input a_RDC1:

Assignment of

"Word in actual value message frame 1" -> "actual value (0 to 3)" is carried out in a_RDC1.

a_RDC1[word number] := actual value number

Input a_HL_RDC1:

Assignment of

"Word in actual value message frame 1" -> "lowword or highword of the selected actual value" is carried out in a_HL_RDC1.

a_HL_RDC1[word number] := FALSE (if lowword)

a_HL_RDC1[word number] := TRUE (if highword)

Example:

Word number in actual value message frame 1	Selected actual value	Connection at input a_RDC1	Connection at input a_HL_RDC1
0	Actual value 1 lowword	a_RDC1[0] = SINT#1	a_HL_RDC1[0] = FALSE
1	Actual value 1 highword	a_RDC1[1] = SINT#1	a_HL_RDC1[1] = TRUE
2	Actual value 0 word	a_RDC1[2] = SINT#0	a_HL_RDC1[2] = FALSE
3	Actual value 2 word	a_RDC1[3] = SINT#2	a_HL_RDC1[3] = FALSE

Input a_RDC2:

Assignment of

"Word in actual value message frame 2" -> "actual value (4 to 7)" is carried out in a_RDC2.

a_RDC2[word number] := actual value number

Input a_HL_RDC2:

Assignment of

"Word in actual value message frame 2" -> "lowword or highword of the selected actual value" is carried out in a_HL_RDC2.

a_HL_RDC2[word number] := FALSE (if lowword)

a_HL_RDC2[word number] := TRUE (if highword)

If you do not want to assign an actual value to a word in actual value message frame 1 or 2, enter -1 as the reference value number at the corresponding entry in a_RDC1 or a_RDC2 respectively. In this case, the corresponding setting in a_HL_RDC1 or a_HL_RDC1 is meaningless.

a_RDC1[word number] := SINT#-1

a_RDC2[word number] := SINT#-1

Example:
Word 1 of actual value message frame 2 is not to be assigned to an actual value.

Word number in actual value message frame 2	Selected actual value	Connection at input a_RDC1	Connection at input a_HL_RDC1
1	None	a_RDC2[1] = SINT# -1	a_HL_RDC2[1] meaningless

Actual values are assigned to the words of the actual value message frames in the CANsync slave. By default, this assignment is carried out in a similar way to the CANsync master. The system does not cross-check the assignment in the CANsync master and the CANsync slave, since there are also reasonable applications for assignments that are different.

4.17 CANsync_PD_CFG_READ_SL

Description

You can use this function block für CANsync to configure the assignment of the actual value message frames of the CANsync slave in the CANsync interface module for a CANsync slave.

NOTE

FB CANsync_PD_CFG_READ_SL uses library BM_TYPES_20bd03 or above.

Parameter input	Data type	Description
_BASE	CANsync_PLC_SL_BMSTRUCT	Operating data for the CANsync interface module
si_SL_NR	SINT 0 to 31	Slave number of the CANsync slaves from which actual values are received
a_RDC1	SINT_4_BMARRAY	Actual value numbers for actual value message frame 1
a_HL_RDC1	BOOL_4_BMARRAY	Assignment of highword or lowword for actual value message frame 1
a_RDC2	SINT_4_BMARRAY	Actual value numbers for actual value message frame 2
a_HL_RDC2	BOOL_4_BMARRAY	Assignment of highword or lowword for actual value message frame 2

Parameter output	Data type	Description
_BASE	CANsync_PLC_SL_BMSTRUCT	Operating data for the CANsync interface module

Using FB CANsync_PD_CFG_READ_SL, you:

- assign actual value message frames 1 and 2 of another CANsync slave (receive) to eight 32-bit actual values

in a CANsync slave.

Every CANsync slave can monitor the actual value message frames of the other CANsync slaves and use the data in the actual value message frames.

In the CANsync slave, the application can read eight 32-bit actual values of every other CANsync slave on the CANsync bus.

The CANsync slaves have numbers 0 to 31.

The actual value numbers are from 0 to 7.

An actual value is composed of a lowword (bits 0 to 15) and a highword (bit 16 to 31).

In the CANsync slave, it is possible to receive the two actual value message frames of each CANsync slave, which each contain 4 * 16 bit data. That is four words each numbered 0 to 3.

Using FB CANsync_PD_CFG_READ_SL, you specify for another CANsync slave the actual values to which data from the four words (words 0 to 3) of actual value message frame 1 is to be assigned and the actual values to which the data from the four words (words 0 to 3) of the other CANsync slave's actual value message frame 2 is to be assigned.

You can assign the data from actual value message frame 1 to actual values 0 to 3 and the data for actual value message frame 2 to actual values 4 to 7.

You can assign each word in an actual value message frame to only one lowword or highword of an actual value.

When transferring a 32-bit actual value, two words are needed in the actual value message frame (one word from the actual value message frame is assigned to the lowword of an actual value and another word in this actual value message frame is assigned to the highword of this actual value).

Input/output _BASE:

At input/output `_BASE`, global variable `_CANsyncSlave_Ctrl_Slot_G` (to `_CANsyncSlave_Ctrl_Slot_M`) is connected in accordance with the slot (G to M) of the option module.

If this global variable is not present in the project, create – in dependence on the slot (G to M) – the global variable `_CANsyncSlave_Ctrl_Slot_G` (to `_CANsyncSlave_Ctrl_Slot_M`) of data type `CANsync_PLC_SL_BMSTRUCT`.

Using the declaration of global variables, this variable must be connected to the base address of the CANsync interface module on the option module.

In dependence on the slot, the base address of the CANsync interface module on the CANsync slave option module for b maXX PLC (BM4-O-CAN-05, DIP switch 9 = ON) is as follows:

Slot	Base address of CANsync interface module
G	%MB3.2001792
H	%MB3.3001792
J	%MB3.4001792
K	%MB3.5001792
L	%MB3.6001792
M	%MB3.7001792

Example:

CANsync interface module on the CANsync slave option module in the b maXX's slot G

```
_CANsyncSlave_Ctrl_Slot_G  AT  %MB3.2001792  :
                                    CANsync_PLC_SL_BMSTRUCT;
```

Where:

`_CANsyncSlave_Ctrl_Slot_G`	is the variable name with the data type short designation "_" for Struct
`CANsync_PLC_SL_BMSTRUCT`	is the data type
`%MB3.2001792`	is the base address of the CANsync interface module on the CANsync slave option module in the b maXX's slot G

Input si_SL_NR:

At input si_SL_NR, you state the slave number of the CANsync slave on the CANsync bus whose actual value message frames are to be configured.

Input a_RDC1:

Assignment of

"Word in actual value message frame 1" -> "actual value (0 to 3)" is carried out in a_RDC1.

a_RDC1[word number] := actual value number

Input a_HL_RDC1:

Assignment of

"Word in actual value message frame 1" -> "lowword or highword of the selected actual value" is carried out in a_HL_RDC1.

a_HL_RDC1[word number] := FALSE (if lowword)

a_HL_RDC1[word number] := TRUE (if highword)

Example:

Word number in actual value message frame 1	Selected actual value	Connection at input a_RDC1	Connection at input a_HL_RDC1
0	Actual value 1 lowword	a_RDC1[0] = SINT#1	a_HL_RDC1[0] = FALSE
1	Actual value 1 highword	a_RDC1[1] = SINT#1	a_HL_RDC1[1] = TRUE
2	Actual value 0 word	a_RDC1[2] = SINT#0	a_HL_RDC1[2] = FALSE
3	Actual value 2 word	a_RDC1[3] = SINT#2	a_HL_RDC1[3] = FALSE

Input a_RDC2:

Assignment of

"Word in actual value message frame 2" -> "actual value (4 to 7)" is carried out in a_RDC2.

a_RDC2[word number] := actual value number

Input a_HL_RDC2:

Assignment of

"Word in actual value message frame 2" -> "lowword or highword of the selected actual value" is carried out in a_HL_RDC2.

a_HL_RDC2[word number] := FALSE (if lowword)

a_HL_RDC2[word number] := TRUE (if highword)

If you do not want to assign an actual value to a word in actual value message frame 1 or 2, enter -1 as the reference value number at the corresponding entry in a_RDC1 or a_RDC2 respectively. In this case, the corresponding setting in a_HL_RDC1 or a_HL_RDC1 is meaningless.

a_RDC1[word number] := SINT# -1

a_RDC2[word number] := SINT# -1

Example:
Word 1 of actual value message frame 2 is not to be assigned to an actual value.

Word number in actual value message frame 2	Selected actual value	Connection at input a_RDC1	Connection at input a_HL_RDC1
1	None	a_RDC2[1] = SINT# -1	a_HL_RDC2[1] meaningless

In the other CANsync slave, actual values are assigned to the words of the actual value message frames. By default, this assignment is carried out in a similar way to the (receiving) CANsync slave. The system does not cross-check the assignment in the (receiving) CANsync slave and the other CANsync slave, since there are also reasonable applications for assignments that are different.

4.18 CANsync_PD_CFG_SL

Description

You can use this function block for CANsync to configure the assignment of the CANsync interface module's specified and actual value message frames for a CANsync slave.

NOTE

FB CANsync_PD_CFG_SL uses library BM_TYPES_20bd03 or above.

Parameter input	Data type	Description
_BASE	CANsync_PLC_SL_BMSTRUCT	Operating data for the CANsync interface module
a_WRC1	SINT_4_BMARRAY	Reference value numbers for reference value message frame 1
a_HL_WRC1	BOOL_4_BMARRAY	Assignment of highword or lowword for reference value message frame 1
a_WRC2	SINT_4_BMARRAY	Reference value numbers for reference value message frame 2
a_HL_WRC2	BOOL_4_BMARRAY	Assignment of highword or lowword for reference value message frame 2
a_RDC1	SINT_4_BMARRAY	Actual value numbers for actual value message frame 1
a_HL_RDC1	BOOL_4_BMARRAY	Assignment of highword or lowword for actual value message frame 1
a_RDC2	SINT_4_BMARRAY	Actual value numbers for actual value message frame 2
a_HL_RDC2	BOOL_4_BMARRAY	Assignment of highword or lowword for actual value message frame 2

Parameter output	Data type	Description
_BASE	CANsync_PLC_SL_BMSTRUCT	Operating data for the CANsync interface module

Using FB CANsync_PD_CFG_SL, you:

- assign reference value message frames 1 and 2 (receive) to eight 32-bit reference values

and

- assign eight 32-bit actual values to actual value message frames 1 and 2 (send)

in a CANsync slave.

Receiving reference values:

In the CANsync slave, the application can write eight 32-bit reference values. The reference value numbers are from 0 to 7.

A reference value is composed of a lowword (bits 0 to 15) and a highword (bit 16 to 31).

In the CANsync slave, it is possible to receive two reference value message frames each containing 4 * 16 bit data. That is four words each numbered 0 to 3.

Using FB CANsync_PD_CFG_MA, you specify the reference values to which the data from the four words (words 0 to 3) of reference value message frame 1 is to be assigned and the reference values to which the data from the four words (words 0 to 3) of reference value message frame 2 is to be assigned.

You can assign the data from reference value message frame 1 to reference values 0 to 3 and the data from reference value message frame 2 can be assigned to reference values 4 to 7.

You can assign each word in a reference value message frame to only one lowword or highword of an actual value.

When transferring a 32-bit reference value, two words are needed in the reference value message frame (one word from the reference value message frame is assigned to the lowword of a reference value and another word in this reference value message frame is assigned to the highword of this reference value).

Sending actual values:

In the CANsync slave, the application can also write eight 32-bit actual values. The actual value numbers are from 0 to 7.

An actual value is composed of a lowword (bits 0 to 15) and a highword (bit 16 to 31).

In the two actual value message frames, it is possible to send 4 * 16-bit data each to the CANsync master. That is four words numbered 0 to 3.

Using FB CANsync_PD_CFG_SL, you specify which data is to be entered in the four words (words 0 to 3) of actual value message frame 1 and which is to be entered in the four words (words 0 to 3) of actual value message frame 2.

You can choose the data for actual value message frame 1 from actual values 0 to 3 and the data for actual value message frame 2 from actual values 4 to 7.

You can assign to each word in an actual value message frame a lowword or a highword of an actual value. Two words are needed in the actual value message frame when transferring a 32-bit actual value.

Input/output _BASE:

At input/output `_BASE`, global variable `_CANsyncSlave_Ctrl_Slot_G` (to `_CANsyncSlave_Ctrl_Slot_M`) is connected in accordance with the slot (G to M) of the option module.

If this global variable is not present in the project, create – in dependence on the slot (G to M) – the global variable `_CANsyncSlave_Ctrl_Slot_G` (to `_CANsyncSlave_Ctrl_Slot_M`) of data type `CANsync_PLC_SL_BMSTRUCT`.

Using the declaration of global variables, this variable must be connected to the base address of the CANsync interface module on the option module.

In dependence on the slot, the base address of the CANsync interface module on the CANsync slave option module for b maXX PLC (BM4-O-CAN-05, DIP switch 9 = ON) is as follows:

Slot	Base address of CAN-sync interface module
G	%MB3.2001792
H	%MB3.3001792
J	%MB3.4001792
K	%MB3.5001792
L	%MB3.6001792
M	%MB3.7001792

Example:

CANsync interface module on the CANsync slave option module in the b maXX's slot G

```
_CANsyncSlave_Ctrl_Slot_G  AT  %MB3.2001792  :
                                    CANsync_PLC_SL_BMSTRUCT;
```

Where:

`_CANsyncSlave_Ctrl_Slot_G`	is the variable name with the data type short designation "_" for Struct
`CANsync_PLC_SL_BMSTRUCT`	is the data type
`%MB3.2001792`	is the base address of the CANsync interface module on the CANsync slave option module in the b maXX's slot G

Input a_WRC1:

Assignment of

"Word in reference value message frame 1" -> "reference value (0 to 3)" is carried out in a_WRC1.

a_WRC1[word number] := reference value number

Input a_HL_WRC1:

Assignment of

"Word in reference value message frame 1" -> "Lowword or highword of the selected reference value" is carried out in a_HL_WRC1.

a_HL_WRC1[word number] := FALSE (if lowword)

a_HL_WRC1[word number] := TRUE (if highword)

Example:

Word number in reference value message frame 1	Selected reference value	Connection at input a_WRC1	Connection at input a_HL_WRC1
0	Reference value 1 lowword	a_WRC1[0] = SINT#1	a_HL_WRC1[0] = FALSE
1	Reference value 1 highword	a_WRC1[1] = SINT#1	a_HL_WRC1[1] = TRUE
2	Reference value 0 word	a_WRC1[2] = SINT#0	a_HL_WRC1[2] = FALSE
3	Reference value 2 word	a_WRC1[3] = SINT#2	a_HL_WRC1[3] = FALSE

Input a_WRC2:

Assignment of

"Word in reference value message frame 2" -> "reference value (4 to 7)" is carried out in a_WRC2.

a_WRC2[word number] := reference value number

Input a_HL_WRC2:

Assignment of

"Word in reference value message frame 2" -> "Lowword or highword of the selected reference value" is carried out in a_HL_WRC2.

a_HL_WRC2[word number] := FALSE (if lowword)

a_HL_WRC2[word number] := TRUE (if highword)

If you do not want to assign a word from reference value reference value message frame 1 or 2 to a reference value, enter -1 as the reference value number at the corresponding entry in a_WRC1 or a_WRC2 respectively.

In this case, the corresponding setting in a_HL_WRC1 or a_HL_WRC2 is meaningless.

a_WRC1[word number] := SINT#-1

a_WRC2[word number] := SINT#-1

Example:

You do not want to assign word 1 of reference value message frame 2 to a reference value.

Word number in reference value message frame 2	Selected reference value	Connection at input a_WRC2	Connection at input a_HL_WRC2
1	None	a_WRC2[1] = SINT# -1	a_HL_WRC2[1] meaningless

Reference values are assigned to the words of the reference value message frames in the CANsync master. By default, this assignment is carried out in a similar way to the

CANsync slave. The system does not cross-check the assignment in the CANsync master and the CANsync slave, since there are also reasonable applications for assignments that are different.

Input a_RDC1:

Assignment of

"Actual value (0 to 3)" -> "word in actual value message frame 1" is carried out in a_RDC1.

a_RDC1[word number] := actual value number

Input a_HL_RDC1:

Assignment of

"Lowword or highword of the selected actual value" -> "word in actual value message frame 1" is carried out in a_HL_RDC1.

a_HL_RDC1[word number] := FALSE (if lowword)

a_HL_RDC1[word number] := TRUE (if highword)

Example:

Word number in actual value message frame 1	Selected actual value	Connection at input a_RDC1	Connection at input a_HL_RDC1
0	Actual value 1 lowword	a_RDC1[0] = SINT#1	a_HL_RDC1[0] = FALSE
1	Actual value 1 highword	a_RDC1[1] = SINT#1	a_HL_RDC1[1] = TRUE
2	Actual value 0 word	a_RDC1[2] = SINT#0	a_HL_RDC1[2] = FALSE
3	Actual value 2 word	a_RDC1[3] = SINT#2	a_HL_RDC1[3] = FALSE

Input a_RDC2:

Assignment of

"Actual value (4 to 7)" -> "Word in actual value message frame 2" is carried out in a_RDC2.

a_RDC2[word number] := actual value number

Input a_HL_RDC2:

Assignment of

"Lowword or highword of the selected actual value" -> "word in actual value message frame 2" is carried out in a_HL_RDC2.

a_HL_RDC2[word number] := FALSE (if lowword)

a_HL_RDC2[word number] := TRUE (if highword)

If you do not want to assign an actual value to a word in actual value message frame 1 or 2, enter -1 as the reference value number at the corresponding entry in a_RDC1 or

a_RDC2 respectively. In this case, the corresponding setting in a_HL_RDC1 or a_HL_RDC1 is meaningless.

a_RDC1[word number] := SINT#-1

a_RDC2[word number] := SINT#-1

Example:
You do not want to assign an actual value to word 1 of actual value message frame 2.

Word number in actual value message frame 2	Selected actual value	Connection at input a_RDC1	Connection at input a_HL_RDC1
1	None	a_RDC2[1] = SINT# -1	a_HL_RDC2[1] meaningless

In the CANsync master, the words of the actual value message frames are assigned to actual values. By default, this assignment is carried out in a similar way to the CANsync slave. The system does not cross-check the assignment in the CANsync master and the CANsync slave, since there are also reasonable applications for assignments that are different.

4.19 CANsync_PD_COMM_MA

Description

You can use this function block for CANsync to carry out process data communication (reference and actual values) of the CANsync master interface module.

NOTE

FB CANsync_PD_COMM_MA uses library BM_TYPES_20bd03 or above.

Parameter input	Data type	Description
_BASE	CANsync_PLC_MA_BMSTRUCT	Operating data for the CANsync interface module
a_RD_ARRAY	CANsync_RD_BMARRAY	Output array for the actual values of the CANsync slaves
a_WR_VALUES_SEND	DINT_8_BMARRAY	Reference values that are to be sent
si_WRC1_SEND	SINT 5	Command for reference value message frame 1 to be sent
si_WRC2_SEND	SINT 5	Command for reference value message frame 2 to be sent
si_RD_SL_NR1_RECEIVE	SINT -1, 0 to 31	Slave number of the CANsync slave from which actual value message frame 1 is to be requested
si_RD_SL_NR2_RECEIVE	SINT -1, 0 to 31	Slave number of the CANsync slave from which actual value message frame 2 is to be requested
si_MAX_SL_NR	SINT 0 to 31	Maximum slave number for automatic incrementing [a)]
x_COPY_TO_RD_ARRAY	BOOL	Indication of whether actual values are to be copied into RD_ARRAY

a) This input corresponds to input si_MAX_SL_NR am FB CANsync_COMM_CONTROL_MA as the maximum slave number for automatic polling for send control word jobs, parameter jobs and/or upload/download jobs.

Parameter output	Data type	Description
_BASE	CANsync_PLC_MA_BMSTRUCT	Operating data for the CANsync interface module
a_RD_ARRAY	CANsync_RD_BMARRAY	Output array for the actual values of the CANsync slaves

Parameter output	Data type	Description
si_RD_SL_NR1_RECEIVED	SINT -1, 0 to 31	Display of the slave number of the CANsync slave from which actual value message frame 1 was received
si_RD_SL_NR2_RECEIVED	SINT -1, 0 to 31	Display of the slave number of the CANsync slave from which actual value message frame 2 was received

Using this FB, the system transfers reference values (a_WR_VALUES_SEND) to the CANsync interface module that are sent to the CANsync slave by means of the reference value message frames [1)]. Apart from this, the CANsync slaves request [1)] the actual value message frames and output the actual values (in a_RD_ARRAY).

In each CANsync interval, the system sends reference value message frames 1 and 2 to the CANsync slaves.

In each CANsync interval, CANsync slave number si_RD_SL_NR1_RECEIVE requests actual value message frame 1; and in each CANsync interval, CANsync slave number si_RD_SL_NR2_RECEIVE requests actual value message frame 2.

The CANsync slaves' request for the actual value message frames runs automatically as follows:

In each CANsync interval, the system automatically increments by one the slave number of the CANsync slave (from which actual value message frames 1 and 2 are requested). This incrementation is carried out up to si_MAX_SL_NR. After this, the system starts with the request for the CANsync slave with slave number 0, etc. (si_RD_SL_NR1_RECEIVE and si_RD_SL_NR2_RECEIVE not then assigned).

Input/output _BASE:

At input/output `_BASE`, global variable `_CANsyncMaster_Ctrl_Slot_G` (to `_CANsyncMaster_Ctrl_Slot_M`) is connected in accordance with the slot (G to M) of the option module.

If this global variable is not present in the project, create – in dependence on the slot (G to M) – the global variable `_CANsyncMaster_Ctrl_Slot_G` (to `_CANsyncMaster_Ctrl_Slot_M`) of data type `CANsync_PLC_MA_BMSTRUCT`.

Using the declaration of global variables, this variable must be connected to the base address of the CANsync interface module on the option module.

In dependence on the slot, the base address of the CANsync interface module on the CANsync master option module (BM4-O-CAN-06) is as follows:

[1)] Assuming you carried out corresponding configuration for reference value message frames 1 and 2 using FB CANsync_PD_CFG_MA for actual value message frames 1 and 2 of each CANsync slave using FB CANsync_PD_CFG_READ_MA.

Slot	Base address of CAN-sync interface module
G	%MB3.2001792
H	%MB3.3001792
J	%MB3.4001792
K	%MB3.5001792
L	%MB3.6001792
M	%MB3.7001792

Example:

CANsync interface module on the CANsync master option module in the b maXX's slot G

```
_CANsyncMaster_Ctrl_Slot_G  AT  %MB3.2001792  :
                                        CANsync_PLC_MA_BMSTRUCT;
```

Where:

`_CANsyncMaster_Ctrl_Slot_G`	is the variable name with the data type short designation "_" for Struct
`CANsync_PLC_MA_BMSTRUCT`	is the data type
`%MB3.2001792`	is the base address of the CANsync interface module on the CANsync master option module in the b maXX's slot G

Input/output a_RD_ARRAY (actual values 0 to 7 of CANsync slaves 0 to 31):

A variable of data type CANsync_RD_BMARRAY is connected at a_RD_ARRAY. Data type CANsync_RD_BMARRAY is a two-dimensional field of 32 (CANsync slaves) with 16 actual values each [1].

This means that data type CANsync_RD_BMARRAY is a field of 32 entries of data type DINT_16_BMARRAY. Data type DINT_16_BMARRAY is a field of 16 entries of data type double integer:

```
DINT_16_BMARRAY      : ARRAY [0..15] OF DINT;

CANsync_RD_BMARRAY   : ARRAY [0..31] OF DINT_16_BMARRAY
```

Example:

```
a_Istwerte        : CANsync_RD_BMARRAY;
```

Where:

`a_Istwerte`	is the variable name with the data type short designation "a" for ARRAY
`CANsync_RD_BMARRAY`	is the data type

[1] Currently, actual values 0 to 3 (actual value message frame 1) and actual values 4 to 7 (actual value message frame 2) are supported.

The system accesses the individual actual values according to this pattern:

Variable name[slave number of the CANsync slave][number of the actual value]

NOTE

There is no period between the variable name and the square brackets or between the square brackets themselves.

Example: The system writes (in structured text (ST)) variable di_Istwert_21_6 with actual value 6 of the CANsync slave with slave number 21:

```
di_Istwert_21_6 := a_Istwerte[21][6];
```

Inputs a_WR_VALUES_SEND, si_WRC1_SEND and si_WRC2_SEND:

A variable of data type DINT_8_BMARRAY is connected at input a_WR_VALUES_SEND. Data type DINT_8_BMARRAY is a field with 8 entries of data type double integer:

```
DINT_8_BMARRAY  : ARRAY [0..7] OF DINT;
```

Example:

```
a_Sollwerte     : DINT_8_BMARRAY;
```

Where:

`a_Sollwerte`	is the variable name with the data type short designation "a" for ARRAY
`DINT_8_BMARRAY`	is the data type

The system then expects reference values 0 to 7 in field elements `a_Sollwerte[0]` to `a_Sollwerte[7]`, for example.

At input si_WRC1_SEND, the system states when reference values 0 to 3 are valid. Reference value message frame 1 can then be sent A value of 5 indicates that reference values 0 to 3 are valid, whereas any other value indicates that reference values 0 to 3 are not valid.

At input si_WRC2_SEND, the system states when reference values 4 to 7 are valid. Reference value message frame 2 can then be sent A value of 5 indicates that reference values 4 to 7 are valid, whereas any other value indicates that reference values 4 to 7 are not valid.

NOTE

In reference value message frame 1, the system sends reference values 0 to 3; and in reference value message frame 2, it sends reference values 4 to 7. Gaps in reference value numbers are permissible.

Inputs si_RD_SL_NR1_RECEIVE, si_RD_SL_NR2_RECEIVE and si_MAX_SL_NR:

At input si_RD_SL_NR1_RECEIVE, you state the slave number of the CANsync slave from which actual value message frame 1 is requested.

For automatic requesting of the CANsync slaves' actual value message frame 1, si_RD_SL_NR1_RECEIVE is not assigned (or set equal to -128) and the system states the highest slave number at si_MAX_SL_NR.

At input si_RD_SL_NR2_RECEIVE, you state the slave number of the CANsync slave from which actual value message frame 2 is requested.

For automatic requesting of the CANsync slaves' actual value message frame 2, si_RD_SL_NR2_RECEIVE is not assigned (or set equal to -128) and the system states the highest slave number at si_MAX_SL_NR.

In every CANsync interval, the system then increments by one the slave number of the CANsync slave (from which actual value message frame 1 and/or 2 is/are requested automatically) until the number at input si_MAX_SL_NR is reached.

After this, the system starts with polling for the CANsync slave with slave number 0, etc.

It is also possible to mix explicit specification and automatic incrementing for actual value message frames 1 and 2.

The highest slave number of a CANsync slave from which actual value message frames are requested (input si_MAX_SL_NR) is also used for requirements data communication (control word, parameters, upload/download).

If si_MAX_SL_NR = -1, the value remains unchanged on the CANsync interface module.

The default setting is si_MAX_SL_NR = -1, i.e. the value remains unchanged on the CANsync interface module.

NOTE

This input corresponds to input si_MAX_SL_NR on FB CANsync_PD_COMM_CONTROL_MA. This means that at input si_MAX_SL_NR on FB CANsync_COMM_CONTROL_MA **or** input si_MAX_SL_NR on FB CANsync_PD_COMM_MA, you state the highest slave number of a CANsync slave.
You must use only one of the two inputs!

Input x_COPY_RD_ARRAY:

At input x_COPY_TO_RD_ARRAY, the system indicates with TRUE that the received actual values are entered in the two-dimensional field at a_RD_ARRAY.

If the system indicates FALSE at x_COPY_TO_RD_ARRAY or x_COPY_TO_RD_ARRAY is not assigned, the received actual values are not entered in the two-dimensional field at a_RD_ARRAY.

Outputs si_RD_SL_NR1_RECEIVED, si_RD_SL_NR2_RECEIVED:

At output si_RD_SL_NR1_RECEIVED, the system displays the slave number of the CANsync slave from which actual value message frame 1 was received in the last CANsync interval. If no actual value message frame 1 was received in a CANsync interval, the system displays -128 at si_RD_SL_NR1_RECEIVED.

At output si_RD_SL_NR2_RECEIVED, the system displays the slave number of the CANsync slave from which actual value message frame 2 was received in the last CANsync interval. If no actual value message frame 2 was received in a CANsync interval, the system displays -128 at si_RD_SL_NR2_RECEIVED.

4.20 CANsync_PD_COMM_READ_MA

NOTE

FB CANsync_PD_COMM_READ_MA is present for reasons of compatibility and you should not use it in new projects. The system uses FB CANsync_PD_COMM_MA to output the actual values of all the CANsync slaves to a_RD_BMARRAY.

Description

You can use this function block for CANsync to output in a CANsync master interface module the process data actual values of a CANsync slave.

NOTE

FB CANsync_PD_COMM_READ_MA uses library BM_TYPES_20bd03 or above.

Parameter input	Data type	Description
_BASE	CANsync_PLC_MA_BMSTRUCT	Operating data for the CANsync interface module
si_SL_NR	SINT 0 to 31	Slave number of the CANsync slaves from which actual values are read

Parameter output	Data type	Description
_BASE	CANsync_PLC_MA_BMSTRUCT	Operating data for the CANsync interface module
a_RD_VALUES	DINT_8_BMARRAY	Output array for actual values that were received from the CANsync slave
si_RD_SL_NR1_RECEIVED	SINT	Display that actual value message frame 1 was received
si_RD_SL_NR2_RECEIVED	SINT	Display that actual value message frame 2 was received

Using this FB, the system, outputs the actual values of a CANsync slave (a_RD_VALUES). The CANsync slave's request for the actual value message frames must be made via FB CANsync_PD_COMM_MA.

Using FB CANsync_PD_COMM_READ_MA is only reasonable if the actual values are not output at FB CANsync_PD_COMM_MA, i.e. when FB CANsync_PD_COMM_MA, input x_COPY_TO_RD_ARRAY = FALSE.

Input/output _BASE:

At input/output `_BASE`, global variable `_CANsyncMaster_Ctrl_Slot_G` (to `_CANsyncMaster_Ctrl_Slot_M`) is connected in accordance with the slot (G to M) of the option module.

If this global variable is not present in the project, create – in dependence on the slot (G to M) – the global variable `_CANsyncMaster_Ctrl_Slot_G` (to `_CANsyncMaster_Ctrl_Slot_M`) of data type `CANsync_PLC_MA_BMSTRUCT`.

Using the declaration of global variables, this variable must be connected to the base address of the CANsync interface module on the option module.

In dependence on the slot, the base address of the CANsync interface module on the CANsync master option module (BM4-O-CAN-06) is as follows:

Slot	Base address of CANsync interface module
G	%MB3.2001792
H	%MB3.3001792
J	%MB3.4001792
K	%MB3.5001792
L	%MB3.6001792
M	%MB3.7001792

Example:

CANsync interface module on the CANsync master option module in the b maXX's slot G

```
_CANsyncMaster_Ctrl_Slot_G  AT  %MB3.2001792  :
                                    CANsync_PLC_MA_BMSTRUCT;
```

Where:

`_CANsyncMaster_Ctrl_Slot_G`	is the variable name with the data type short designation "_" for Struct
`CANsync_PLC_MA_BMSTRUCT`	is the data type
`%MB3.2001792`	is the base address of the CANsync interface module on the CANsync master option module in the b maXX's slot G

Input si_SL_NR:

At input si_SL_NR, you state the slave number of the CANsync slave on the CANsync bus whose actual values are to be output.

Input/output a_RD_VALUES (actual values 0 to 7 of CANsync slave si_SL_NR):

A variable of data type DINT_8_BMARRAY is connected at a_RD_VALUES. Data type DINT_8_BMARRAY is a field of 8 entries of data type double integer:

```
DINT_8_BMARRAY  : ARRAY [0..7] OF DINT;
```

Example:

```
a_Istwerte_3    : DINT_8_BMARRAY;
```

Where:

`a_Istwerte_3`	is the variable name with the data type short designation "a" for ARRAY
`DINT_8_BMARRAY`	is the data type

The system accesses the individual actual values according to this pattern:

Variable name[number of the actual value]

Example: The system writes (in structured text (ST)) variable `di_Istwert_3_6` with actual value 6 of the CANsync slave with slave number 3:

```
di_Istwert_3_6 := a_Istwerte_3[6];
```

NOTE

Currently, actual values 0 to 3 (actual value message frame 1) and actual values 4 to 7 (actual value message frame 2) are supported.

Outputs si_RD_SL_NR1_RECEIVED, si_RD_SL_NR2_RECEIVED:

At output si_RD_SL_NR1_RECEIVED, the system indicates with 1 whether actual value message frame 1 was received. If actual value message frame 1 was not received, si_RD_SL_NR1_RECEIVED = 0. The new actual values 0 to 3 are only output at a_RD_VALUES when actual value message frame 1 was received.

At output si_RD_SL_NR2_RECEIVED, the system indicates with 2 whether actual value message frame 2 was received. If actual value message frame 2 was not received, si_RD_SL_NR2_RECEIVED = 0. The new actual values 4 to 7 are only output at a_RD_VALUES when actual value message frame 2 was received.

4.21 CANsync_PD_COMM_READ_SL

NOTE

FB CANsync_PD_COMM_READ_SL is present for reasons of compatibility and you should not use it in new projects. The system uses FB CANsync_PD_COMM_MA to output the monitored actual values to a_RD_BMARRAY of the other CANsync slaves.

Description

You can use this function block for CANsync to output in a CANsync slave interface module the process data actual values of a CANsync slave.

NOTE

FB CANsync_PD_COMM_READ_SL uses library BM_TYPES_20bd03 or above.

Parameter input	Data type	Description
_BASE	CANsync_PLC_SL_BMSTRUCT	Operating data for the CANsync interface module
si_SL_NR	SINT 0 to 31	Slave number of the CANsync slaves from which actual values are monitored

Parameter output	Data type	Description
_BASE	CANsync_PLC_SL_BMSTRUCT	Operating data for the CANsync interface module
a_RD_VALUES	DINT_8_BMARRAY	Output array for actual values that were received from the CANsync slave
si_RD_SL_NR1_RECEIVED	SINT	Display that actual value message frame 1 was received
si_RD_SL_NR2_RECEIVED	SINT	Display that actual value message frame 2 was received

Using this FB, the system, outputs the actual values that were monitored of another CANsync slave (a_RD_VALUES). The CANsync master must use FB CANsync_PD_COMM_MA to request the actual value message frames of the other CANsync slave.

Using FB CANsync_PD_COMM_READ_SL is only reasonable if the actual values are not output at FB CANsync_PD_COMM_SL, i.e. when FB CANsync_PD_COMM_SL, input x_COPY_TO_RD_ARRAY = FALSE.

Input/output _BASE:

At input/output `_BASE`, global variable `_CANsyncSlave_Ctrl_Slot_G` (to `_CANsyncSlave_Ctrl_Slot_M`) is connected in accordance with the slot (G to M) of the option module.

If this global variable is not present in the project, create – in dependence on the slot (G to M) – the global variable `_CANsyncSlave_Ctrl_Slot_G` (to `_CANsyncSlave_Ctrl_Slot_M`) of data type `CANsync_PLC_SL_BMSTRUCT`.

Using the declaration of global variables, this variable must be connected to the base address of the CANsync interface module on the option module.

In dependence on the slot, the base address of the CANsync interface module on the CANsync slave option module for b maXX PLC (BM4-O-CAN-05, DIP switch 9 = ON) is as follows:

Slot	Base address of CANsync interface module
G	%MB3.2001792
H	%MB3.3001792
J	%MB3.4001792
K	%MB3.5001792
L	%MB3.6001792
M	%MB3.7001792

Example:

CANsync interface module on the CANsync slave option module in the b maXX's slot G

```
_CANsyncSlave_Ctrl_Slot_G  AT  %MB3.2001792  :
                                        CANsync_PLC_SL_BMSTRUCT;
```

Where:

`_CANsyncSlave_Ctrl_Slot_G`	is the variable name with the data type short designation "_" for Struct
`CANsync_PLC_SL_BMSTRUCT`	is the data type
`%MB3.2001792`	is the base address of the CANsync interface module on the CANsync slave option module in the b maXX's slot G

Input si_SL_NR:

At input si_SL_NR, you state the slave number of the CANsync slave on the CANsync bus whose monitored actual values are to be output.

Input/output a_RD_VALUES (actual values 0 to 7 of CANsync slave si_SL_NR):

A variable of data type DINT_8_BMARRAY is connected at a_RD_VALUES. Data type DINT_8_BMARRAY is a field of 8 entries of data type double integer:

```
DINT_8_BMARRAY  : ARRAY [0..7] OF DINT;
```

Example:

```
a_Istwerte_3    : DINT_8_BMARRAY;
```

Where:

`a_Istwerte_3`	is the variable name with the data type short designation "a" for ARRAY
`DINT_8_BMARRAY`	is the data type

The system accesses the individual actual values according to this pattern:

Variable name[number of the actual value]

Example: The system writes (in structured text (ST)) variable `di_Istwert_22_6` with actual value 6 of the CANsync slave with slave number 22:

```
di_Istwert_22_6 := a_Istwerte_22[6];
```

NOTE

Currently, actual values 0 to 3 (actual value message frame 1) and actual values 4 to 7 (actual value message frame 2) are supported.

Outputs si_RD_SL_NR1_RECEIVED, si_RD_SL_NR2_RECEIVED:

At output si_RD_SL_NR1_RECEIVED, the system indicates with 1 whether actual value message frame 1 was received. If actual value message frame 1 was not received, si_RD_SL_NR1_RECEIVED = 0. The new actual values 0 to 3 are only output at a_RD_VALUES when actual value message frame 1 was received.

At output si_RD_SL_NR2_RECEIVED, the system indicates with 2 whether actual value message frame 2 was received. If actual value message frame 2 was not received, si_RD_SL_NR2_RECEIVED = 0. The new actual values 4 to 7 are only output at a_RD_VALUES when actual value message frame 2 was received.

4.22 CANsync_PD_COMM_SL

Description

You can use this function block for CANsync to carry out process data communication (reference and actual values) of the CANsync slave interface module.

NOTE

FB CANsync_PD_COMM_SL uses library BM_TYPES_20bd03 or above.

Parameter input	Data type	Description
_BASE	CANsync_PLC_SL_BMSTRUCT	Operating data for the CANsync interface module
a_RD_ARRAY	CANsync_RD_BMARRAY	Output array for the actual values of the CANsync slaves
a_RD_VALUES_SEND	DINT_8_BMARRAY	Actual values that are sent to the CANsync master
si_RDC1_SEND	SINT 5	Command for reference value message frame 1 to be sent
si_RDC2_SEND	SINT 5	Command for reference value message frame 2 to be sent
x_COPY_TO_RD_ARRAY	BOOL	Indication of whether received actual values are to be copied into RD_ARRAY

Parameter output	Data type	Description
_BASE	CANsync_PLC_SL_BMSTRUCT	Operating data for the CANsync interface module
a_RD_ARRAY	CANsync_RD_BMARRAY	Output array for the actual values of the CANsync slaves
a_WR_VALUES_RECEIVED	DINT_8_BMARRAY	Reference values that were received by the CANsync master
si_WRC1_RECEIVED	SINT	Display that reference value message frame 1 was received
si_WRC2_RECEIVED	SINT	Display that reference value message frame 2 was received
si_RD_SL_NR1_RECEIVED	SINT	Display of the slave number of the CANsync slave from which actual value message frame 1 was received

Parameter output	Data type	Description
si_RD_SL_NR2_RECEIVED	SINT	Display of the slave number of the CANsync slave from which actual value message frame 2 was received

Using this FB, the system transfers actual values (a_RD_VALUES_SEND) to the CANsync interface module that are sent to the CANsync master by means of the actual value message frames [1]). In addition, the other CANsync slaves on the CANsync bus monitor the actual value message frames [1]) and output the actual values (in a_RD_ARRAY).

In each CANsync interval, the system sends reference value message frames 1 and 2 to the CANsync slaves and this FB outputs them to a_WR_VALUES_RECEIVED.

In each CANsync interval, the CANsync master requests actual value message frame 1 from a CANsync slave. The system only sends actual value message frame 1 when the CANsync master requests it from this CANsync slave.

In each CANsync interval, the CANsync master requests actual value message frame 2 from a CANsync slave. The system only sends actual value message frame 2 when the CANsync master requests it from this CANsync slave.

This FB can monitor and evaluate actual value message frames 1 and 2 of the other CANsync slaves when the CANsync master requests these actual value message frames.

NOTE

A CANsync slave can evaluate the actual value message frames of other CANsync slaves but not request them!

Input/output _BASE:

At input/output `_BASE`, global variable `_CANsyncSlave_Ctrl_Slot_G` (to `_CANsyncSlave_Ctrl_Slot_M`) is connected in accordance with the slot (G to M) of the option module.

If this global variable is not present in the project, create – in dependence on the slot (G to M) – the global variable `_CANsyncSlave_Ctrl_Slot_G` (to `_CANsyncSlave_Ctrl_Slot_M`) of data type `CANsync_PLC_SL_BMSTRUCT`.

Using the declaration of global variables, this variable must be connected to the base address of the CANsync interface module on the option module.

In dependence on the slot, the base address of the CANsync interface module on the CANsync slave option module for b maXX PLC (BM4-O-CAN-05, DIP switch 9 = ON) is as follows:

[1]) Assuming you carried out corresponding configuration for reference value message frames 1 and 2 using FB CANsync_PD_CFG_SL for actual value message frames 1 and 2 of each other CANsync slave (except this one) using FB CANsync_PD_CFG_READ_SL .

Slot	Base address of CAN-sync interface module
G	%MB3.2001792
H	%MB3.3001792
J	%MB3.4001792
K	%MB3.5001792
L	%MB3.6001792
M	%MB3.7001792

Example:

CANsync interface module on the CANsync slave option module in the b maXX's slot G

```
_CANsyncSlave_Ctrl_Slot_G  AT  %MB3.2001792  :
                                        CANsync_PLC_SL_BMSTRUCT;
```

Where:

`_CANsyncSlave_Ctrl_Slot_G`	is the variable name with the data type short designation "_" for Struct
`CANsync_PLC_SL_BMSTRUCT`	is the data type
`%MB3.2001792`	is the base address of the CANsync interface module on the CANsync slave option module in the b maXX's slot G

Input/output a_RD_ARRAY (actual values 0 to 7 of [the other] CANsync slaves 0 to 31):

A variable of data type CANsync_RD_BMARRAY is connected at a_RD_ARRAY.

Data type CANsync_RD_BMARRAY is a two-dimensional field of 32 (CANsync slaves) with 16 actual values each [1]).

This means that data type CANsync_RD_BMARRAY is a field of 32 entries of data type DINT_16_BMARRAY. Data type DINT_16_BMARRAY is a field of 16 entries of data type double integer:

```
DINT_16_BMARRAY   : ARRAY [0..15] OF DINT;
CANsync_RD_BMARAY : ARRAY [0..31] OF DINT_16_BMARRAY
```

Example:

```
a_Istwerte      : CANsync_RD_BMARRAY;
```

Where:

`a_Istwerte`	is the variable name with the data type short designation "a" for ARRAY
`CANsync_RD_BMARRAY`	is the data type

[1]) Currently, actual values 0 to 3 (actual value message frame 1) and actual values 4 to 7 (actual value message frame 2) are supported.

The system accesses the individual actual values according to this pattern:

Variable name[slave number of the CANsync slave][number of the actual value]

NOTE

There is no period between the variable name and the square brackets or between the square brackets themselves.

Example: The system writes (in structured text (ST)) variable `di_Istwert_21_6` with actual value 6 of the CANsync slave with slave number 21:

```
di_Istwert_21_6 := a_Istwerte[21][6];
```

NOTE

The system enters the monitored actual values of the other CANsync slaves in a_RD_ARRAY, but not the actual values of this CANsync slave.

Inputs a_RD_VALUES_SEND, si_RDC1_SEND and si_RDC2_SEND:

A variable of data type DINT_8_BMARRAY is connected at input a_RD_VALUES_SEND. Data type DINT_8_BMARRAY is a field with 8 entries of data type double integer:

```
DINT_8_BMARRAY  : ARRAY [0..7] OF DINT;
```

Example:

```
a_Istwerte_Senden : DINT_8_BMARRAY;
```

Where:

`a_Istwerte_Senden`	is the variable name with the data type short designation "a" for ARRAY
`DINT_8_BMARRAY`	is the data type

The system enters actual values 0 to 7 in entries 0 to 7 (e.g. `a_Istwerte_Senden[0]` to `a_Istwerte_Senden[7]`).

At input si_RDC1_SEND, the system states when actual values 0 to 3 are valid. The system can then send actual value message frame 1, assuming that the CANsync master requests it. A value of 5 indicates that actual values 0 to 3 are valid, whereas any other value indicates that actual values 0 to 3 are not valid.

At input si_RDC2_SEND, the system states when reference values 4 to 7 are valid. The system can then send actual value message frame 2, assuming that the CANsync master requests it. A value of 5 indicates that actual values 4 to 7 are valid, whereas any other value indicates that actual values 4 to 7 are not valid.

NOTE

In actual value message frame 1, the system sends actual values 0 to 3; and in actual value message frame 2, it sends actual values 4 to 7. Gaps in actual value numbers are permissible.

Input x_COPY_RD_ARRAY:

At input x_COPY_TO_RD_ARRAY, the system indicates with TRUE that the monitored actual values are entered in the two-dimensional field at a_RD_ARRAY.

If the system indicates FALSE at x_COPY_TO_RD_ARRAY or x_COPY_TO_RD_ARRAY is not assigned, the monitored actual values are not entered in the two-dimensional field at a_RD_ARRAY.

Outputs a_WR_VALUES_RECEIVED, si_WRC1_RECEIVED, si_WRC2_RECEIVED:

A variable of data type DINT_8_BMARRAY is connected at output a_WR_VALUES_RECEIVED. Data type DINT_8_BMARRAY is a field with 8 entries of data type double integer:

```
DINT_8_BMARRAY  : ARRAY [0..7] OF DINT;
```

Example:

```
a_Sollwerte     : DINT_8_BMARRAY;
```

Where:

`a_Sollwerte`	is the variable name with the data type short designation "a" for ARRAY
`DINT_8_BMARRAY`	is the data type

The system then enters reference values 0 to 7 that the CANsync master received in field elements `a_Sollwerte[0]` to `a_Sollwerte[7]`, for example. The reference values can be of word or double-word format.

At output si_WRC1_RECEIVED, the system indicates with 1 whether reference value message frame 1 was received. If reference value message frame 1 was not received, si_WRC1_RECEIVED = 0. The new reference values 0 to 3 are only output at a_WR_VALUES_RECEIVED when reference value message frame 1 was received.

At output si_WRC2_RECEIVED, the system indicates with 2 whether reference value message frame 2 was received. If reference value message frame 2 was not received, si_WRC2_RECEIVED = 0. The new reference values 4 to 7 are only output at a_WR_VALUES_RECEIVED when reference value message frame 2 was received.

NOTE

In reference value message frame 1, the system receives reference values 0 to 3; and in reference value message frame 2, it receives reference values 4 to 7. Gaps in reference value numbers are permissible.

Outputs si_RD_SL_NR1_RECEIVED, si_RD_SL_NR2_RECEIVED:

At output si_RD_SL_NR1_RECEIVED, the system displays the slave number of the CANsync slave from which actual value message frame 1 was monitored in the last CANsync interval. If no actual value message frame 1 was monitored in a CANsync interval, the system displays -128 at si_RD_SL_NR1_RECEIVED.

At output si_RD_SL_NR2_RECEIVED, the system displays the slave number of the CANsync slave which monitored actual value message frame 2 in the last CANsync interval. If no actual value message frame 2 was monitored in a CANsync interval, the system displays -128 at si_RD_SL_NR2_RECEIVED.

4.23 CANsync_SL_TYP_INIT

Description

You can use this function block for CANsync to output the slave types for CANsync initialization.

NOTE

This FB is used together with FB CANsync_INIT.

FB CANsync_SL_TYP_INIT uses library BM_TYPES_20bd03 or above.

Parameter input	Data type	Description
us_SL_TYP0	USINT	Slave type 0
us_SL_TYP1	USINT	Slave type 1
us_SL_TYP2	USINT	Slave type 2
...	...	...
us_SL_TYP31	USINT	Slave type 31

Parameter output	Data type	Description
a_SL_TYP	BYTE_32_BMARRAY	Initialization data of slave types

Using this FB, the system outputs the slave types of the CANsync slaves on the CANsync bus. The data is entered in a field (output a_SL_TYP). This field is connected at input a_SL_TYP of the FB CANsync_INIT.

Inputs us_SL_TYP0 to us_SL_TYP31:

For the CANsync slave with slave number 0, you state the slave type at input us_SL_TYP0.

- us_SL_TYP0 = 0 means that no CANsync slave is present with slave number 0
- us_SL_TYP0 = 1 means that a CANsync slave is present with slave number 0

For the CANsync slave with slave number 1, you state the slave type at input us_SL_TYP1.

- us_SL_TYP1 = 0 means that no CANsync slave is present with slave number 1
- us_SL_TYP1 = 1 means that a CANsync slave is present with slave number 1

etc.

For the CANsync slave with slave number 31, you state the slave type at input us_SL_TYP31.

- us_SL_TYP31 = 0 means that no CANsync slave is present with slave number 31
- us_SL_TYP31 = 1 means that a CANsync slave is present with slave number 31

Meanings of the slave types

Slave type	Meaning
0	No CANsync slave present with slave number x
1	b maXX present with CANsync slave option module with slave number x
2 - 255	Reserved

Output a_SL_TYP:

A variable of data type BYTE_32_BMARRAY is connected at output a_SL_TYP. Data type BYTE_32_BMARRAY is a field of 32 entries of data type byte:

```
BYTE_32_BMARRAY   : ARRAY [0..31] OF BYTE;
```

Example:

```
a_Slave_Typen     : BYTE_32_BMARRAY;
```

Where:

`a_Slave_Typen`	is the variable name with the data type short designation "a" for ARRAY
`BYTE_32_BMARRAY`	is the data type

In the individual entries of the field is located the slave type of the CANsync slave on the CANsync bus. In entry [0], there is the slave type of the CANsync slave with slave number 0; in entry [1] there is the slave type of the CANsync slave with slave number 1, etc.

A 0 in entry [x] means that there is no CANsync slave with slave number x on the CANsync bus.

A value ≠0 in entry [x] means that there is one CANsync slave with slave number x on the CANsync bus.

NOTE

This variable is connected at input a_SL_TYP of the FB CANsync_INIT.

4.24 CANsync_UPDOWNLOAD_MA

Description

You can use this function block for CANsync to carry out an upload or a download in the block 1 range (see ▹Upload/Download Block Area ◃ from page 68 onward). It is specially designed for use with FB CANsync_UPDOWNLOAD_SL in the CANsync slave.

NOTE

To use this FB it is necessary for the command channel (CC) to release the upload and download command.
The system uses FB CANsync_COMM_CONTROL_MA to carry out releasing.

FB CANsync_UPDOWNLOAD_MA uses library BM_TYPES_20bd03 or above.

Parameter input	Data type	Description
_BASE	CANsync_PLC_MA_BMSTRUCT	Operating data for the CANsync interface module
a_DOWNLOAD	CANsync_UPDOWN_BMARRAY	Download values
a_UPLOAD	CANsync_UPDOWN_BMARRAY	Upload values
si_SL_NR	SINT 0 to 31	Slave number of the CANsync slave to which the upload or download job is addressed
x_UPDOWN	BOOL	Upload or download
u_LENGTH	UINT 1 to 2048	Length of the up down block
us_MAX_NR_OF_COMM	USINT	Maximum number of communications attempts
t_TIME	TIME	Monitoring time
x_EN	BOOL	Enable
x_RESET	BOOL	Reset

Parameter output	Data type	Description
_BASE	CANsync_PLC_MA_BMSTRUCT	Operating data for the CANsync interface module
a_DOWNLOAD	CANsync_UPDOWN_BMARRAY	Download values
a_UPLOAD	CANsync_UPDOWN_BMARRAY	Upload values
x_BUSY	BOOL	Communication is active
w_ERR_SL	WORD	Error word (CANsync slave -> CANsync master)
x_ERR_SL	BOOL	Error bit (group error bit of w_ERR_SL)

Parameter output	Data type	Description
b_ERR	BYTE	Error byte
x_ERR	BOOL	Error bit (group error bit of b_ERR)
x_OK	BOOL	OK bit

FB CANsync_UPDOWNLOAD_MA carries out an upload or a download in the block 1 area. FB CANsync_UPDOWNLOAD_MA is specially designed for use with FB CANsync_UPDOWNLOAD_SL in the CANsync slave.

FB CANsync_UPDOWN_MA transfers download data to the CANsync slave with slave number si_SL_NR.

The download job consists of u_LENGTH values from field a_DOWNLOAD.

(FB CANsync_UPDOWN_MA carries out a download job to the CANsync slave in several download message frame blocks. The system sends per download message frame block a maximum of 75 values from field a_DOWNLOAD to the CANsync slave).

The CANsync slave processes the download job and returns the result of communication.

FB CANsync_UPDOWN_MA requests upload data from the CANsync slave with slave number si_SL_NR.

The system requests u_LENGTH values from the CANsync slave.

(FB CANsync_UPDOWN_MA carries out an upload job to the CANsync slave in several upload message frame blocks. The system receives per upload message frame block a maximum of 75 values from the CANsync slave).

The CANsync slave processes the upload job and returns the result of communication. The values are output at a_UPLOAD.

Input/output _BASE:

At input/output `_BASE`, global variable `_CANsyncMaster_Ctrl_Slot_G` (to `_CANsyncMaster_Ctrl_Slot_M`) is connected in accordance with the slot (G to M) of the option module.

If this global variable is not present in the project, create – in dependence on the slot (G to M) – the global variable `_CANsyncMaster_Ctrl_Slot_G` (to `_CANsyncMaster_Ctrl_Slot_M`) of data type `CANsync_PLC_MA_BMSTRUCT`.

Using the declaration of global variables, this variable must be connected to the base address of the CANsync interface module on the option module.

In dependence on the slot, the base address of the CANsync interface module on the CANsync master option module (BM4-O-CAN-06) is as follows:

Slot	Base address of CANsync interface module
G	%MB3.2001792
H	%MB3.3001792
J	%MB3.4001792

Slot	Base address of CANsync interface module
K	%MB3.5001792
L	%MB3.6001792
M	%MB3.7001792

Example:

CANsync interface module on the CANsync master option module in the b maXX's slot G

```
_CANsyncMaster_Ctrl_Slot_G  AT  %MB3.2001792  :
                                        CANsync_PLC_MA_BMSTRUCT;
```

Where:

`_CANsyncMaster_Ctrl_Slot_G`	is the variable name with the data type short designation "_" for Struct
`CANsync_PLC_MA_BMSTRUCT`	is the data type
`%MB3.2001792`	is the base address of the CANsync interface module on the CANsync master option module in the b maXX's slot G

Input/output a_DOWNLOAD:

A variable of data type CANsync_UPDOWN_BMARRAY is connected at a_DOWNLOAD. Data type CANsync_UPDOWN_BMARRAY is a field of 2048 entries of data type double integer:

```
CANsync_UPDOWN_BMARRAY        : ARRAY [0..2047] OF DINT;
```

Example:

```
a_Downloadwerte : CANsync_UPDOWN_BMARRAY;
```

Where:

`a_Downloadwerte`	is the variable name with the data type short designation "a" for ARRAY
`CANsync_UPDOWN_BMARRAY`	is the data type

The system then expects the data for the download in field elements `a_Downloadwerte[0]` to `a_Downloadwerte[2047]`, for example.

Input/output a_UPLOAD:

A variable of data type CANsync_UPDOWN_BMARRAY is connected at a_DOWNLOAD. Data type CANsync_UPDOWN_BMARRAY is a field of 2048 entries of data type double integer:

```
CANsync_UPDOWN_BMARRAY        : ARRAY [0..2047] OF DINT;
```

Example:

```
a_Uploadwerte : CANsync_UPDOWN_BMARRAY;
```

Where:

`a_Uploadwerte`	is the variable name with the data type short designation "a" for ARRAY
`CANsync_UPDOWN_BMARRAY`	is the data type

At uploading, the system then outputs the data in field elements `a_Uploadwerte[0]` to `a_Uploadwerte[2047]`, for example.

Input si_SL_NR:

At input si_SL_NR, you state the slave number of the CANsync slave on the CANsync bus from which the system is to upload data or to which it is to download it.

NOTE

The upload/download to/from this CANsync slave must be enabled via FB CANsync_COMM_CONTROL_MA (see description of FB CANsync_COMM_CONTROL_MA).

Input x_UPDOWN:

At input x_UPDOWN, you set an upload job with x_UPDOWN = FALSE; and with x_UPDOWN = TRUE, you set a download job.

Input u_LENGTH:

At input u_LENGTH, you state the length of the upload/download area to be transferred. The system transfers a maximum of 2048 32-bit values. If you enter a 0 at u_LENGTH, the system sets bit 1 in error byte b_ERR to TRUE.

The system transfers the data block-by-block in message frame blocks. One message frame block is 75 32-bit values in size. This means that with an upload or download job of more than 75 32-bit values, the system needs to send correspondingly more message frame blocks.

Input us_MAX_NR_OF_COMM:

At input us_MAX_NR_OF_COMM, you can specify how often a block of the upload/download area is to be repeated if the following cases apply: a) the monitoring time (of the CANsync master) has expired; and b) the CANsync slave did not report an error (the default setting is us_MAX_NR_OF_COMM = 1). The system does not issue the timeout message (see input t_TIME) until time us_MAX_NR_OF_COMM * t_TIME has expired and up to this instant no answer from the CANsync slave is present.

Input t_TIME:

You state the monitoring time at input t_TIME. If the download job or the upload job is not completely processed within the monitoring time, the system sets bit 2 in the error byte (see also input us_MAX_NR_OF_COMM).

Incomplete processing of an upload or download job can be due to the command channel being busy with higher-priority messages (broadcast commands, send control word jobs,

parameter jobs; see ▹Requirements Data ◃ from page 23 onward).Another possible cause is that the command channel is not released for upload/download jobs (see FB CANsync_COMM_CONTROL_MA).

Input x_EN:

With x_EN = TRUE, the system enables FB CANsync_UPDOWNLOAD_MA, i.e. upload or download jobs can be sent to the CANsync slave.

If x_EN is set to FALSE before the upload or download job is completed, it is assumed that the job was cancelled deliberately. You must then reset FB CANsync_UPDOWNLOAD_MA with x_RESET = TRUE to start a new upload or download job via the specified length (input u_LENGTH).

Input x_RESET:

You can use x_RESET = TRUE to reset FB CANsync_UPDOWNLOAD_MA. This is necessary after cancelling the upload or download job (using x_EN = FALSE) or following an error message, for example. After this, you must set x_RESET back to FALSE.

Output x_BUSY:

Output x_BUSY indicates with TRUE that FB CANsync_UPDOWNLOAD_MA is processing a job.

Output x_OK:

Output x_OK indicates with TRUE that the upload or download job has been carried out correctly. Output x_OK is FALSE if the system did not execute an upload or download job or it was not executed correctly.

Outputs x_ERR_SL, w_ERR_SL:

If an error occurs in the CANsync slave while the upload or download job is being carried out, the system sets error bit x_ERR_SL to TRUE and outputs error word w_ERR_SL (see below). In this case output x_OK stays FALSE.

Outputs x_ERR, b_ERR:

If an error occurs while the upload or download job is being carried out, the system sets error bit x_ERR to TRUE and outputs error byte b_ERR (see below). In this case output x_OK stays FALSE.

Error byte b_ERR:

Bit No.	Meaning
0	CANsync slave reports an error when processing the upload/download job
1	Length of the upload/download area to be transferred is equal to 0 (input u_LENGTH)
2	Timeout
3 to 7	Reserved

Error word w_ERR_SL:

w_ERR_SL	Meaning
16#0000	Reserved
16#0001	CANsync slave acknowledges wrong block number
16#0002	Entered transfer block length > 300 bytes / 75 doublewords
16#0003 to 16#00FF	Reserved
16#0100	CANsync slave expects transfer block with the number that is entered in the counter
16#0101	CANsync slave expects transfer block end
16#0102	CANsync slave does not yet expect transfer block end
16#0103	CANsync slave cancels job
16#0104	Job not possible
16#0105	Base address not allowed
16#0106	Reserved
16#0107	Upload/download area CANsync master > Upload/download area CANsync slave
16#0108	Message frame mode error (mode not allowed at this stage)
16#0109 to 16#FFFF	Reserved

4.25 CANsync_UPDOWNLOAD_SL

Description

Depending on the CANsync master job, you can use this function block for CANsync to carry out an upload or a download in the block 1 range (see ▷Upload/download command ◁ from page 95 onward). It is specially designed for use with FB CANsync_UPDOWNLOAD_MA in the CANsync master.

NOTE

FB CANsync_UPDOWNLOAD_SL uses library BM_TYPES_20bd03 or above.

Parameter input	Data type	Description
_BASE	CANsync_PLC_SL_BMSTRUCT	Operating data for the CANsync interface module
a_DOWNLOAD	CANsync_UPDOWN_BMARRAY	Download values
a_UPLOAD	CANsync_UPDOWN_BMARRAY	Upload values
t_TIME	TIME	Monitoring time
x_EN	BOOL	Enable
x_RESET	BOOL	Reset

Parameter output	Data type	Description
_BASE	CANsync_PLC_SL_BMSTRUCT	Operating data for the CANsync interface module
a_DOWNLOAD	CANsync_UPDOWN_BMARRAY	Download values
a_UPLOAD	CANsync_UPDOWN_BMARRAY	Upload values
x_UPDOWN	BOOL	Display of upload or download
x_ORDER_ACTIV	BOOL	Display of pending job
u_LENGTH	UINT	Length of the transferred block
x_BUSY	BOOL	Display of FB is active
w_ERR	WORD	Error word
x_ERR	BOOL	Error bit
x_OK	BOOL	OK bit

FB CANsync_UPDOWNLOAD_SL carries out a download job or an upload job of the to the CANsync master. The system outputs download values (a_DOWNLOAD) and sends upload values (a_UPLOAD) to the CANsync master.

The system transfers the data in several message frame blocks. A maximum of 75 32-bit values are transferred per message frame block.

FB CANsync_UPDOWNLOAD_SL is specially designed for use with FB CANsync_UPDOWNLOAD_MA in the CANsync master.

Input/output _BASE:

At input/output `_BASE`, global variable `_CANsyncSlave_Ctrl_Slot_G` (to `_CANsyncSlave_Ctrl_Slot_M`) is connected in accordance with the slot (G to M) of the option module.

If this global variable is not present in the project, create – in dependence on the slot (G to M) – the global variable `_CANsyncSlave_Ctrl_Slot_G` (to `_CANsyncSlave_Ctrl_Slot_M`) of data type `CANsync_PLC_SL_BMSTRUCT`.

Using the declaration of global variables, this variable must be connected to the base address of the CANsync interface module on the option module.

In dependence on the slot, the base address of the CANsync interface module on the CANsync slave option module for b maXX PLC (BM4-O-CAN-05, DIP switch 9 = ON) is as follows:

Slot	Base address of CANsync interface module
G	%MB3.2001792
H	%MB3.3001792
J	%MB3.4001792
K	%MB3.5001792
L	%MB3.6001792
M	%MB3.7001792

Example:

CANsync interface module on the CANsync slave option module in the b maXX's slot G

```
_CANsyncSlave_Ctrl_Slot_G  AT  %MB3.2001792  :
                                        CANsync_PLC_SL_BMSTRUCT;
```

Where:

`_CANsyncSlave_Ctrl_Slot_G`	is the variable name with the data type short designation "_" for Struct
`CANsync_PLC_SL_BMSTRUCT`	is the data type
`%MB3.2001792`	is the base address of the CANsync interface module on the CANsync slave option module in the b maXX's slot G

Input/output a_DOWNLOAD:

A variable of data type CANsync_UPDOWN_BMARRAY is connected at a_DOWNLOAD. Data type CANsync_UPDOWN_BMARRAY is a field of 2048 entries of data type double integer:

```
CANsync_UPDOWN_BMARRAY        : ARRAY [0..2047] OF DINT;
```

Example:

```
a_Downloadwerte : CANsync_UPDOWN_BMARRAY;
```

Where:

`a_Downloadwerte`	is the variable name with the data type short designation "a" for ARRAY
`CANsync_UPDOWN_BMARRAY`	is the data type

At downloading, the system then outputs the data in field elements `a_Downloadwerte[0]` to `a_Downloadwerte[2047]`, for example.

Input/output a_UPLOAD:

A variable of data type CANsync_UPDOWN_BMARRAY is connected at a_DOWNLOAD. Data type CANsync_UPDOWN_BMARRAY is a field of 2048 entries of data type double integer:

```
CANsync_UPDOWN_BMARRAY        : ARRAY [0..2047] OF DINT;
```

Example:

```
a_Uploadwerte : CANsync_UPDOWN_BMARRAY;
```

Where:

`a_Uploadwerte`	is the variable name with the data type short designation "a" for ARRAY
`CANsync_UPDOWN_BMARRAY`	is the data type

The system then expects the data for the upload in field elements `a_Uploadwerte[0]` to `a_Uploadwerte[2047]`, for example.

Input t_TIME:

You state the monitoring time at input t_TIME. If the download job or the upload job is not completely processed within the monitoring time, the system sets bit 0 in the error word. If input t_TIME is not assigned, this yields a preset monitoring time of 30 s.

Incomplete processing of an upload or download job can be due to the command channel being busy with higher-priority messages (broadcast commands, send control word jobs, parameter jobs; see ▹Requirements Data ◃ from page 23 onward). Another possible cause is that the command channel is not released for upload/download jobs (see FB ▹CANsync_COMM_CONTROL_MA ◃ from page 113 onward).

Input x_EN:

With x_EN = TRUE, the system enables FB CANsync_UPDOWNLOAD_SL, i.e. upload or download jobs can be processed.

If x_EN is set to FALSE before the upload or download job is completed, it is assumed that it was cancelled deliberately. You must then reset FB CANsync_UPDOWNLOAD_SL with x_RESET = TRUE to be able to carry out a new upload or download job.

Input x_RESET:

You can use x_RESET = TRUE to reset FB CANsync_UPDOWNLOAD_SL. This is necessary after cancelling the upload or download job (using x_EN = FALSE) or following an error message, for example. After this, you must set x_RESET back to FALSE.

Outputs x_UPDOWN, x_ORDER_ACTIV:

Output x_ORDER_ACTIV indicates with TRUE that a job is present. Output x_UPDOWN displays the type of job. In the case of an upload job, x_UPDOWN = FALSE; with a download job, x_UPDOWN = TRUE.

Output u_LENGTH:

Output u_LENGTH displays the number of 32-bit values of the transferred message frame block. One message frame block is a maximum of 75 32-bit values in size. In the case of an upload or download job that is larger than 75 32-bit values, the system transfers correspondingly more message frame blocks and displays at u_LENGTH the number of 32-bit values of the currently transferred message frame block.

Output x_BUSY:

Output x_BUSY indicates with TRUE that FB CANsync_UPDOWNLOAD_SL is processing an upload or download job.

Output x_OK:

Output x_OK indicates with TRUE that the upload or download job has been carried out correctly. Output x_OK is FALSE if the system did not execute an upload or download job or it was not executed correctly.

Outputs x_ERR, w_ERR:

If an error occurs, the system sets error bit x_ERR to TRUE and outputs error word b_ERR. In this case output x_OK stays FALSE.

Error word w_ERR:

Bit No.	Meaning
0	Timeout
1 to 15	Reserved

APPENDIX A - ABBREVIATIONS

16#	Prefix for hexadecimal numbers
AVC	Actual value channel
BACI	Baumüller Component Interface
BUB	Ballast unit
BUC	Baumüller feed/return feed unit
BUG	Baumüller converter basic feed unit
BUM	Baumüller individual power unit
BUS	Baumüller power module
CAN	Controller Area Network
CANsync	Synchronized CAN
CC	Command Channel
CPU	Central Processing Unit
DB	Data byte
DC	d.c. current
DP-RAM	Dual-port RAM
DIN	Deutsches Institut für Normung e.V. (German Standards Institute)
EMC	Electromagnetic compatibility
EN	European standard
EPROM	Erasable Programmable Read-Only Memory
ESD	Electrostatic sensitive devices
FB	Function block
I/O	Input/Output
ISO	International Organization for Standardization
IWK	Actual value channel (Read Channel, RDC)
LED	Light Emitting Diode
RAM	Random Access Memory
RC	Response Channel
RDC	Read Channel (actual value channel, IWK)
RVC	Reference value channel
SWK	Reference value channel (Write Channel, WRC)
t_{RSPTO}	Response Timeout
UL	Underwriter Laboratories Inc.
USS	USS protocol function module
USS®	Trademark of Siemens, universal serial interface
VDE	Verband deutscher Elektrotechniker (German Association of Electrical Engineers)
WRC	Write Channel (reference value channel, SWK)

Index

be in motion

Baumüller Nürnberg Electronic GmbH & Co. KG Ostendstraße 80-90 90482 Nuremberg Tel: +49(0)911-5432-0 Fax: +49(0)911-5432-130 **www.baumueller.de**