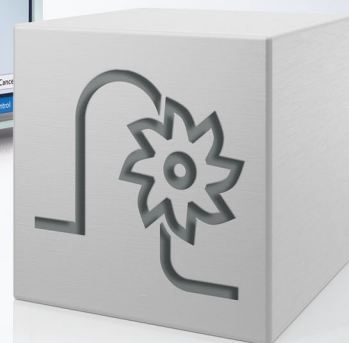
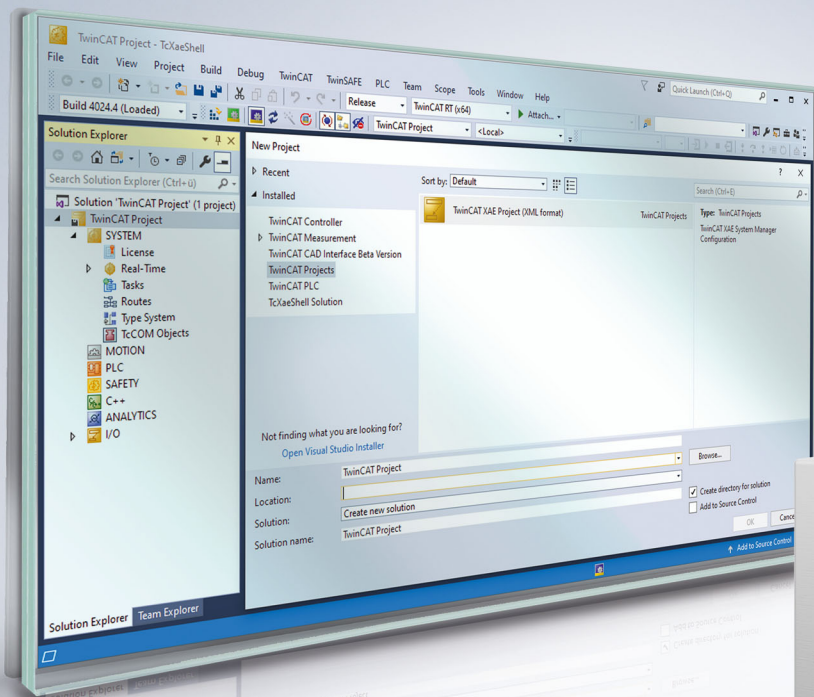


Functional description | EN

TF5200 | TwinCAT 3 CNC

ADS Access on CNC



Notes on the documentation

This description is only intended for the use of trained specialists in control and automation engineering who are familiar with the applicable national standards.

It is essential that the documentation and the following notes and explanations are followed when installing and commissioning the components.

It is the duty of the technical personnel to use the documentation published at the respective time of each installation and commissioning.

The responsible staff must ensure that the application or use of the products described satisfy all the requirements for safety, including all the relevant laws, regulations, guidelines and standards.

Disclaimer

The documentation has been prepared with care. The products described are, however, constantly under development.

We reserve the right to revise and change the documentation at any time and without prior announcement.

No claims for the modification of products that have already been supplied may be made on the basis of the data, diagrams and descriptions in this documentation.

Trademarks

Beckhoff®, TwinCAT®, TwinCAT/BSD®, TC/BSD®, EtherCAT®, EtherCAT G®, EtherCAT G10®, EtherCAT P®, Safety over EtherCAT®, TwinSAFE®, XFC®, XTS® and XPlanar® are registered trademarks of and licensed by Beckhoff Automation GmbH.

Other designations used in this publication may be trademarks whose use by third parties for their own purposes could violate the rights of the owners.

Patent Pending

The EtherCAT technology is patent protected, in particular by the following applications and patents:

EP1590927, EP1789857, EP1456722, EP2137893, DE102015105702

with corresponding applications or registrations in various other countries.



EtherCAT® is registered trademark and patented technology, licensed by Beckhoff Automation GmbH, Germany

Copyright

© Beckhoff Automation GmbH & Co. KG, Germany.

The reproduction, distribution and utilisation of this document as well as the communication of its contents to others without express authorisation are prohibited.

Offenders will be held liable for the payment of damages. All rights reserved in the event of the grant of a patent, utility model or design.

General and safety instructions

Icons used and their meanings

This documentation uses the following icons next to the safety instruction and the associated text. Please read the (safety) instructions carefully and comply with them at all times.

Icons in explanatory text

1. Indicates an action.

⇒ Indicates an action statement.

DANGER

Acute danger to life!

If you fail to comply with the safety instruction next to this icon, there is immediate danger to human life and health.

CAUTION

Personal injury and damage to machines!


If you fail to comply with the safety instruction next to this icon, it may result in personal injury or damage to machines.

NOTICE

Restriction or error

This icon describes restrictions or warns of errors.

Tips and other notes

 This icon indicates information to assist in general understanding or to provide additional information.

General example

Example that clarifies the text.

NC programming example

Programming example (complete NC program or program sequence) of the described function or NC command.

Specific version information


 Optional or restricted function. The availability of this function depends on the configuration and the scope of the version.

Table of contents

Notes on the documentation	3
General and safety instructions	4
1 Introduction	8
2 Description	12
2.1 COM task	13
2.1.1 Requesting object description	13
2.1.2 Data content.....	15
2.2 Task SDA, Task GEO	15
2.2.1 Classes	15
2.2.2 Administration, IndexGroup/Offset address	16
3 Dynamic access to variables	19
3.1 P parameter	20
3.2 V.L variables	20
3.3 V.P variables	20
3.4 V.S variables	20
4 Interface for variables with flexible data type (V.E, V.G, V.A, V.CYC)	22
4.1 General properties	22
4.2 Interfaces	23
4.2.1 Channel-specific interface.....	23
4.2.2 Axis-specific interface (e.g. for V.A variables).....	23
4.3 Transfer parameters and return values.....	25
4.3.1 “Number of variable elements” function	25
4.3.2 “Variable description by index” function	27
4.3.3 “Variable handle by name” function	30
4.3.4 “Read variable value by handle” function.....	32
4.3.5 “Read variable value by name” function.....	34
4.3.6 “Write variable value by handle” function	36
4.3.7 “Write variable value by name” function.....	38
4.3.8 “Extended description by index” function	40
4.3.9 “Variable handle by name” axis-specific function.....	42
4.3.10 “Read variable value by name” axis-specific function	44
4.3.11 “Write variable value by name” axis-specific function	45
5 ADS services	46
6 Support and Service	47
Index	48

List of figures

Fig. 1 Overview of CNC architecture..... 9

Fig. 2 Overview - addressing the number of parameters 19

1 Introduction

Mandatory note on references to other documents

For the sake of clarity, links to other documents and parameters are abbreviated, e.g. [PROG] for the Programming Manual or P-AXIS-00001 for an axis parameter.

For technical reasons, these links only function in the Online Help (HTML5, CHM) but not in pdf files since pdfs do not support cross-linking.

Access to CNC data is by means of CNC objects which are addressed by the index group and index offset. The following tasks are provided and must be addressed via separate ports:

- SDA,
- COM
- GEO

i Code page CP1252 is used.

CNC architecture

CNC objects can be accessed via ADS. The element is then defined via the ADS NetId address, IndexGroup and IndexOffset.

CNC objects can be accessed via IndexGroup and IndexOffset.

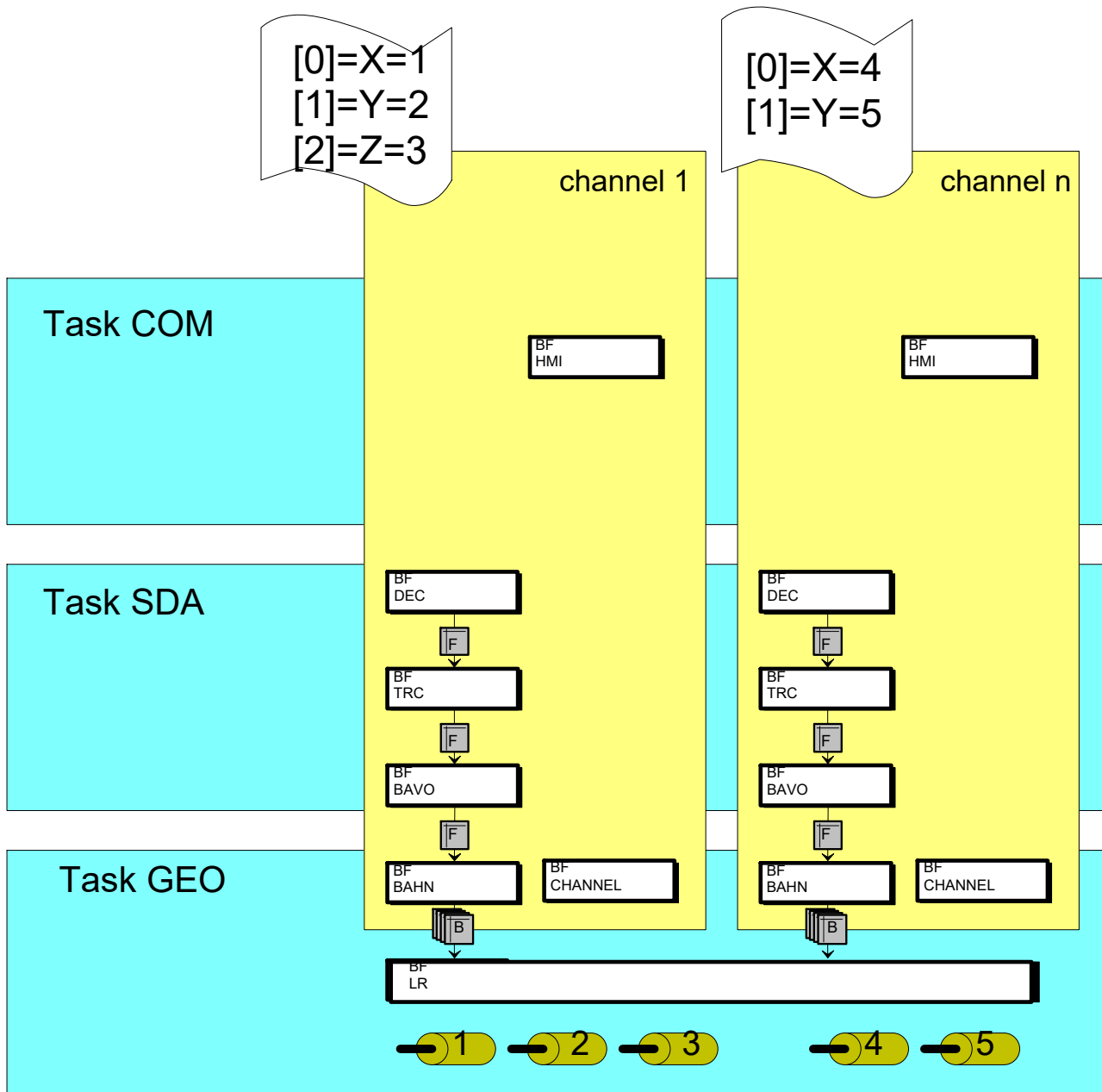


Fig. 1: Overview of CNC architecture

The figure above lists the following CNC classes as an example:

- Decoding (BF DEC)
- Path preparation (BF BAVO)
- Interpolator (BF BAHN)
- Position control (BF LR)
- Spindle (BF SPINDEL)

Access attributes

CNC class instances provide objects that have different attributes. Depending on the CNC configuration, each CNC class creates instances and links them to a CNC architecture. Access can then be made to these object attributes via services. Either:

- read
- write
- read/write

Platform, channel and axis

Depending on the CNC class, the instance can be global throughout the platform (channel number = 0, e.g. position controller) or can be created as a channel-specific instance (channel number > 0).

In addition, an instance can provide both its own elements and also axis-specific elements.

Platform axes

The user creates all controller system axes for the control platform. This produces a sequence of axes. The calculation of the object address used to address the individual axes is derived from the axis placement in this axis sequence. It corresponds to the placement index in this sequence. The first axis in this sequence has the index 1. This address can be used at any time to access the axis.

Channel axes

The channel axes are addressed via the index in the channel. Assigning an axis in the channel can be modified or defined to suit requirements or defined by the channel parameters and axis exchange.

If a channel axis index is currently not occupied by an axis, it can be viewed by reading "blank" data content. In this particular case, the logical axis number = 0.

Spindles

In analogy to a CNC channel which is responsible for the coordinated motion of several axes, the BF SPINDEL is responsible for moving a single independent axis (single axis interpolator).

You can also access generated spindles via CNC objects..

● HLD



The high-level driver (HLD, BF CHANNEL) is a special case here. The HLD of the first channel drives all platform-global axes in the interface to the PLC in addition to channel-specific elements. Therefore, the addresses of the axes are not changed by an axis exchange but are fixed by the start-up description.

● COM task base address



IndexGroup = 0x20100 and 0x20200 are used as the basic address to access internal CNC data of the COM task.

Individual channels or axes can be queried via different values that are additive to this base address (IndexGroup).

● GEO task basic address



IndexGroup = 0x20300 is defined as the basic address to access internal CNC data of the SDA and GEO tasks.

Individual attributes of an element can be queried via different values that are additive to this base address (IndexGroup).

Static CNC objects available

The available static CNC objects are dependent on the CNC Build used.

A list of all available objects can be created using the ISG object browser.

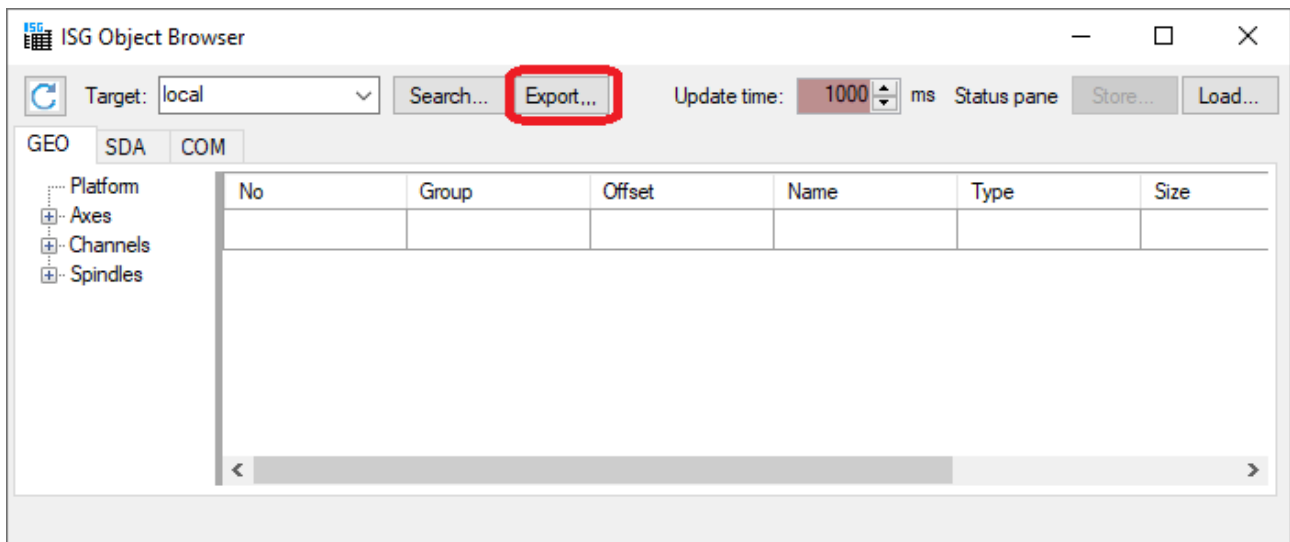
Requirements:

ISG object browser is installed

Configuration is active in TwinCAT and TwinCAT is in RUNNING mode

Execute the following steps:

- Start ISG object browser



- Confirm the Export button
- Define the storage location and the filename of the list generated.

The generated list contains all the static objects available in the CNC version.

2 Description

Notes on addressing

<C_{ID}> Channel or channel ID starting with 1

<A_{ID}> Axis ID starting with 1

<S_{ID}> Spindle index starting with 0. The spindle index is a component part of the index group.

Addressing modes of "IndexGroup"

The index group can be specified for the CNC objects of channels and axes both with "logical" and "index-based" addressing.

"Logical" addressing is selected by the additional bit 0x100000. If this bit is missing, access to the objects is index-based. The example below shows the significance of this bit.

Spindles are generally accessed index-based and the index entry is a component part of the index group. For example, 0x25302 for the index group of the **third** spindle.

Example: There are only two channels with logical numbers 1 and 5.

	Channel "1"	Channel "6"
Logical access	IdxGrp = 0x121301	IdxGrp = 0x12130 6
Index-based access	IdxGrp = 0x21301	IdxGrp = 0x2130 2

It is advisable to use the logical access. The advantage here is that access to channel 6 remains unchanged if channel 1 is deactivated. With index-based access, the index group of channel 6 changes from 0x21302 to 0x21301 in this case and it becomes the 'first' channel.

i It is advisable to use the logical access.

All the examples of addressing in the sections below are executed using logical addressing.

2.1 COM task

Only one instance exists in the COM task and it administers all elements (platform, channels and axes).

Instance classes	Addressing
Platform data	IndexGroup = 0x120101, IndexOffset = <Element>
Platform axes	IndexGroup = 0x120200 +<axis>, IndexOffset = <element>
Channel data	IndexGroup = 0x120100 +<channel>, IndexOffset = <element>
Channel axes	0x120100 +<channel>, IndexOffset = 0x1000 * <axis> + <element>



The first channel (IndexGroup = 0x120101) drives platform data.

2.1.1 Requesting object description

The description of a TASK COM object can be requested by accessing the object with bit identifier 0x30000.

An object is described by the following 84 bytes.

-The description of an object of the task COM can be requested by the bit identifier Object Access (ISGADS_IGR_OBJECT_ACCESS = 0x30000) in the index group-

Data type	Bit	Date	Description
DWORD	0-3	ID	Internal unique ID of an object
DWORD	4-7	Size	Size of object in bytes
WORD	8-9	Write access	TRUE is object is describable.
WORD	10-11	-	Not assigned
DWORD	12-15	Index group	for direct object access to the content
DWORD	16-19	Index offset	for direct object access to the content
STRING(32)	20-51	Name	Name of object
STRING(32)	52-83	Type	Object data type BOOL, BYTE, SINT, WORD, INT, DWORD, DINT, LWORD, LINT, REAL, LREAL, STRING

The index group = 0x13010<ch> is used for objects of the first channel (<ch> = 1). If several channels are configured, the required channel can be selected accordingly via the logic channel ID [1; max]. The platform data can optionally be addressed under channel ID = 0 or 1.

- Total number of all objects is requested via index offset = 0.
- Reading an object description Index offset > 0 supplies object description with index [1; number of objects]; the return memory must be a total of 84 bytes per object.

Reading several object descriptions with one access

- If exactly a multiple of 84 bytes is provided as return memory, object descriptions are returned in sequence starting from the transferred index.

Example

Index group = 0x130101, index offset = 0 returns the number of objects in the first channel

Index group = 0x130101, index offset = 1 returns the object description in the first object (channel=1)

Requesting a single object description

Index group = 0x130101, index offset = 0x10 supplies a single object description with 84 bytes.

Example listing

- ID= 82 (4 bytes)
- Size = 1 (4 bytes)
- Write access = 0, i.e. read only (2 bytes)
- Free (2 bytes)
- Index group = 0x00120101 (4 bytes)
- Index offset = 0x00000003 (4 bytes)
- Name = mc_active_single_block_r (32Byte)
- Type = BOOL (32 bytes)

2.1.2 Data content

The actual read/write access to the data content of the objects takes place via the bit identifier ISGADS_IGR_ELEMENT_ACCESS = 0x20000 in the IndexGroup.

- Platform 0x120100, 0x251
- Axis 0x120200, 0x10001
- Channel 0x120101, 0x1
- Channel axis 0x120101, 0x10001

Later access to object data can then take place via indexGroup = 0x120101, indexOffset = 0x1 "mc_command_single_block_w".

2.2 Task SDA, Task GEO

2.2.1 Classes

IndexGroup base of CNC

IndexGroup = 0x120300

IndexGroup of classes

The following classes of the CNC instances can currently be addressed by the corresponding IndexGroup:

- 0x120300 : position controller (once per CNC, channel-independent with platform-global axes)
- 0x121300 : interpolator (channel-specific with channel axes)
- 0x122300 : decoder (channel-specific with channel axes)
- 0x123300 : HLD (channel-specific, HLD of the first channel additionally contains axes that are platform-global)
- 0x124300 : Path preparation (channel-specific)
- 0x25300 : Spindle

The individual classes have the following additive values on the CNC IndexGroup:

- 0x0000 : position controller (once per CNC, channel-independent with platform-global axes)
- 0x1000 : interpolator (channel-specific with channel axes)
- 0x2000 : decoder (channel-specific with channel axes)
- 0x3000 : HLD (channel-specific, HLD of the first channel additionally contains axes that are platform-global)
- 0x4000 : Path preparation (channel-specific)
- 0x5000 : Spindle

element attributes

The following attributes of an element can be queried (additive values to the IndexGroup):

- 0x000 : data content (depending on data type)
- 0x100 : name as string
- 0x200 : type as string [UNS08;SGN08; ... ;SGN32; REAL32, REAL64]
- 0x300 : length of data in bytes
- 0x400 : unit as string
- 0x500 : BOOLEAN flag indicating whether write access is permitted
- 0x600 : Object address: In (channel, axis), Out (IndexGroup, IndexOffset)

2.2.2 Administration, IndexGroup/Offset address

To determine which instances of a class exist, you can query the object address of the first element (IndexOffset = 0x0) by means of a READ&WRITE access.

The IndexGroup and the IndexOffset are returned. No instance of the class exists if (0, 0) is returned as the address.

If an instance exists, all further existing elements of the instance can be determined by incrementing the IndexOffset.

Object query

Query of object address of the 1st channel

IndexGroup = 0x123900 comprising

<CNC-Basis>	+ <Class = HLD>	+ <Attribute=Address>
0x20300	+ 0x3000	+ 0x600

Input : Channel = 1, Axis = 0

Output : IndexGroup = 0x123301, IndexOffset = 0x0

Query of object address of the 2nd channel

<CNC-Basis>	+ <Class = HLD>	+ <Attribute=Address>
0x20300	+ 0x3000	+ 0x600

Input : Channel = 2, Axis = 0

Output : IndexGroup = 0x123302, IndexOffset = 0x0

Query of 1st platform axis

<CNC-Basis>	+ <Class = HLD>	+ <Attribute=Address>
0x120300	+ 0x3000	+ 0x600

Input : Channel = 0, Axis = 1

Output : IndexGroup = 0x123300, IndexOffset = 0x1000

Query of 2nd decoder axis in 1st channel

<CNC-Basis>	+ <Class = HLD>	+ <Attribute=Address>
0x120300	+ 0x2000	+ 0x600

Input : Channel = 1, Axis = 2

Output : IndexGroup = 0x122301, IndexOffset = 0x20000

Number of element types of a class

The number of existing element types of an instance can be queried by the value content of the first element (IndexOffset = 0).

The name of this element also describes the instance itself.

Position controller

The position controller is not channel-specific, i.e. channel = 0 always applies. The axes are numbered from 1 up to the number of the configured axes.

Interpolator, decoder, path preparation

The channel number must be specified within the range [1; number of configured channels].

Axes are channel-specific and must lie within the range [0; <BAHN_BAVO_AXMAX>].

Channel, HLI driver

The channel number must be specified within the range [1; <number of configured channels>].

If axes are referenced, they must lie within the range [1; <number of configured axes>].

Spindle

The spindle index for spindles must be specified as “channel” entry. The spindle index is numbered from 0 up to the number of the configured axes - 1.

Querying element attributes

The query function is illustrated by the example of the element feedhold.enable in Channel 1. The element IndexOffset is 0x01. The composition of the IndexGroup is shown; the element IndexOffset remains unchanged.

Data content attribute:

<CNC-Basis>	+ <Class = HLD>	+ <Attribute=Address>	<Channel>
0x120300	+ 0x3000	+ 0x0	+ 1

Input : IndexGroup = 0x123301, IndexOffset = 0x01

Output : 0/1 (dependent on element data type)

Name attribute:

<CNC-Basis>	+ <Class = HLD>	+ <Attribute=Address>	<Channel>
0x120300	+ 0x3000	+ 0x100	+ 1

Input : IndexGroup = 0x123401, IndexOffset = 0x01

Output : feedhold.enable

Type attribute:

<CNC-Basis>	+ <Class = HLD>	+ <Attribute=Address>	<Channel>
0x20300	+ 0x3000	+ 0x200	+ 1

Input : IndexGroup = 0x123501, IndexOffset = 0x01

Output : BOOLEAN

Length attribute in bytes:

<CNC-Basis>	+ <Class = HLD>	+ <Attribute=Address>	<Channel>
0x120300	+ 0x3000	+ 0x300	+ 1

Input : IndexGroup = 0x123601, IndexOffset = 0x01

Output : 1

Unit attribute:

<CNC-Basis>	+ <Class = HLD>	+ <Attribute=Address>	<Channel>
0x120300	+ 0x3000	+ 0x400	+ 1

Input : IndexGroup = 0x123701, IndexOffset = 0x01

Output : -

Write protection attribute:

<CNC-Basis>	+ <Class = HLD>	+ <Attribute=Address>	<Channel>
0x120300	+ 0x3000	+ 0x500	+ 1

Input : IndexGroup = 0x123801, IndexOffset = 0x01

Output : 0

3 Dynamic access to variables

CNC objects are principally static and unambiguously identifiable by their IndexGroup and IndexOffset.

Exceptions to this are the following groupings; these variables or parameters are read dynamically.

GEO task

- AEP parameters

SDA task

- V.A variables
- V.E variables
- V.CYC variables
- V.G variables
- [P parameter \[► 20\]](#)
- [V.L variables \[► 20\]](#)
- [V.P variables \[► 20\]](#)
- [V.S variables \[► 20\]](#)

Dynamic elements

While the NC program is in the decoding process, variables can be created dynamically and then deleted. It is also possible to access these variables.

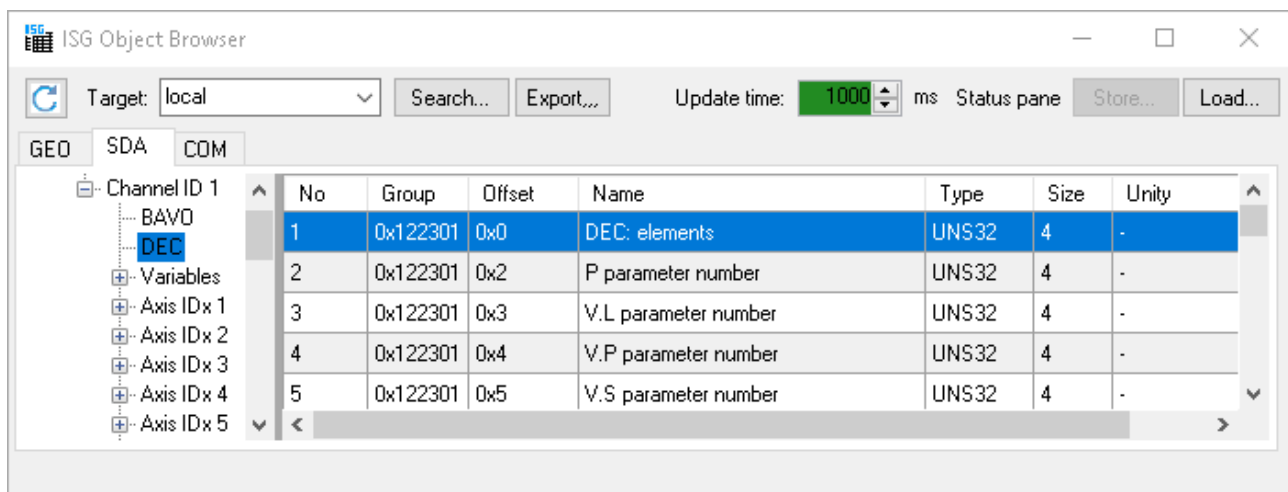


Fig. 2: Overview - addressing the number of parameters

This affects the following:

- [P parameter \[► 20\]](#)
- [V.L variables \[► 20\]](#)
- [V.P variables \[► 20\]](#)
- [V.S variables \[► 20\]](#)

The following process applies to connecting these variables:

1. Read the current number of parameters n_{max}
2. Read the name of each parameter [1; n_{max}]
3. Read the current parameter content



The value of a parameter should always be read by its name.

It is not recommended to read a parameter by the index since the index of a parameter can change when elements are created dynamically and deleted.



The maximum length of a variable name is 96 bytes. This applied to all V. variables and P parameters.

3.1 P parameter

Group	Offset	Identifier	Type	Length
0x122301	0x2	Number of P parameters	UNS32	4
0x122301	0x1B	P-name by index	UNS32/STRING	96
0x122301	0x1C	P-value by name	STRING/REAL64	8
0x122301	0x1D	P-value by index	UNS32/REAL64	8

Read a P parameter after previously querying the number of parameters

1. Read the number of P parameters in 1st channel
Input :: IndexGroup = 0x122301, IndexOffset = 0x02
Output : 1
2. Read the name of the P parameter by the parameter index
Input : IndexGroup = 0x122301, IndexOffset = 0x1b , Parameter = 1
Output : P1
3. Read the value of a P parameter by its name
Input : IndexGroup = 0x122301, IndexOffset = 0x1c , Parameter : P1
Output : 4711

3.2 V.L variables

Group	Offset	Identifier	Type	Length
0x122301	0x3	V.L parameter number	UNS32	4
0x122301	0x1E	V.L-name by index	UNS32/STRING	96
0x122301	0x1F	V.L-value by name	STRING/REAL64	8
0x122301	0x20	V.L-value by index	UNS32/REAL64	8

3.3 V.P variables

Group	Offset	Identifier	Type	Length
0x122301	0x4	V.P parameter number	UNS32	4
0x122301	0x21	V.P-name by index	UNS32/STRING	96
0x122301	0x22	V.P-value by name	STRING/REAL64	8
0x122301	0x23	V.P-value by index	UNS32/REAL64	8

3.4 V.S variables

Group	Offset	Identifier	Type	Length
0x122301	0x5	V.S parameter number	UNS32	4
0x122301	0x24	V.S-name by index	UNS32/STRING	96

0x122301	0x25	V.S-value by name	STRING/REAL64	8
0x122301	0x26	V.S-value by index	UNS32/REAL64	8

4 Interface for variables with flexible data type (V.E, V.G, V.A, V.CYC)

4.1 General properties

- There is a common channel-specific interface to read and write V.E, V.G, V.A and V.CYC variables.
- There is also an axis-specific interface for axis-specific variables (V.A). In the channel-specific interface, the axis name for axis-specific variables must be affixed to the variable names to address the required axis. In the axis-specific interface on the other hand, the variable value of the current axis currently assigned to this position must be read implicitly. Example:
Axis-specific interface: V.A.ACT_POS
Channel-specific interface: V.A.ACT_POS.X
- Access to the variable value can take place by name or handle, whereby using the handle is faster. The exceptions here are V.CYC variables. They are only accessible by name. The variable handle is variable (dependent on the configuration). For this reason, a query can be made by variable name.
- In order to access the variable without “pre-knowledge”, the number of variable entries can be determined and the variable descriptions can be polled (by index).
- Variable arrays with an elementary type (including multi-dimensional fields and “end nodes” of variable structures) can be treated as a single unit. Example:
V.E.array_sgn32, V.E.struct.array_uns32, V.G.NP[0].V, V.G.NP[1].V
When an array variable is read or written, access can be made either to a single element or a complete field. Example:
V.G.NP[0].V[1] -> single element
V.G.NP[0].V -> complete array V.G.NP[0].V[0]... V.G.NP[0].V[n]
- Variable structures are resolved into their single elements, i.e. the variable name is composed of structure names, element name and array indices. Example:
V.G.WZ_AKT.SPDL_AX_NR
V.E.trajectory.vector.x, V.E.trajectory.vector.y, V.E.trajectory.vector.z
V.E.struct1[0].struct2[0].array_sgn32, ..., V.E.struct1[5].struct2[8].array_sgn32
(end node “array_sgn32” is treated as a single unit).

4.2 Interfaces



Only V.CYC variables can be used n-dimensionally.

4.2.1 Channel-specific interface

IndexGroup: IdxGrp = 0x122300 + Channel_No , e.g. for Channel 1: 0x122301:

IdxGrp	IdxOffs	Function	Description	Input	Output
0x122301	0x44	<u>Number of variable elements</u> [▶ 25]	Number of existing variable entries of selected variable type. If no variable type is specified → total number of all available variables V.E + V.G + V.A (Largest index = number of elements – 1)	STRING max. 255 characters	UNS32
0x122301	0x45	<u>Variable description by index</u> [▶ 27]	Variable description (name, handle, variable type, access, size, array dimensions, largest index)	BYTE[2048] variable length	BYTE[2048] variable length
0x122301	0x46	<u>Variable handle by name</u> [▶ 30]	Handle for access to the variable	STRING max. 255 characters	Handle (UNS32)
0x122301	0x47	<u>Read variable value by handle</u> [▶ 32]	Read variable value by variable handle	BYTE[2048] variable length	BYTE[2048] variable length
0x122301	0x48	<u>Read variable value by name</u> [▶ 34]	Read variable value by variable name	STRING max. 255 characters	BYTE[2048] variable length
0x122301	0x49	<u>Write variable value by handle</u> [▶ 36]	Write variable value by variable handle	BYTE[2048] variable length	-
0x122301	0x4A	<u>Write variable value by name</u> [▶ 38]	Write variable value by variable name	BYTE[2048] variable length	-
0x122301	0x4B	<u>Extended variable description by index</u> [▶ 40]	Extended variable description, e.g. for V.E variable synchronisation and validity range	BYTE[2048] variable length	BYTE[2048] variable length
0x122301	0x4C-0x4E	reserved	Reserved	-	-

NOTICE

The following functions are not available for V.CYC variables.

- "Variable handle by name"
- "Read variable value by handle"
- "Write variable value by handle"

4.2.2 Axis-specific interface (e.g. for V.A variables)

IndexGroup: IdxGrp = 0x122300 + Channel ID

Index offset of other axes: $\text{IdxOffs} += 0x10000 * \text{axis index}$

Example Channel 1, Axis 1: $\text{IdxGrp} = 0x122301$

IdxGrp	IdxOffs	Function	Description	Input	Output
0x122301	0x10004	<u>Variable handle by name</u> [▶ 42]	Handle for access to the variable	STRING max. 255 characters	Handle (UNS32)
0x122301	0x10005	<u>Read variable value by name</u> [▶ 44]	Read variable value by variable name	STRING max. 255 characters	BYTE[2048] variable length
0x122301	0x10006	<u>Write variable value by name</u> [▶ 45]	Write variable value by variable name	BYTE[2048] variable length	-
0x122301	0x10007-0x10009	reserved	Reserved	-	-

Instance identifier

In the axis-specific interface, the variable names contain no instance identifier.

Example: V.A.ACT_POS

The axis-specific interface always supplies the variable value of the current axis currently assigned to this position. In case of axis exchange the variable value displayed therefore changes.

4.3 Transfer parameters and return values



Transfer parameters and return values are treated as streams.

4.3.1 "Number of variable elements" function

Index group: IdxGrp = 0x122300 + Channel ID

Index offset: IdxOffs = 0x44

Number of variable entries	This function supplies the number of existing variable entries for the selected variable type. If no variable type is specified, the total number of all available variables V.E + V.G + V.A is returned. The "Variable description by index" function requests the variable description in each case.
-----------------------------------	--

Transfer parameter (here with stream offset specified):

0
Variable type identifier

where

Contents	Type	Size
Variable type identifier with zero termination	STRING "V.G" - Global variables "V.E" - External Variable "V.A" - Axis-specific Variables	variable, max. 256 bytes

=> stream length max. 256 bytes

Return value (here with stream offset specified):

0
Number of existing variable entries

where

Contents	Type	Size
Number of existing variable entries	UNS32	fixed, max. 4 bytes

=> streaming length max. 4 bytes



The variable type identifier, e.g. V.E, may not be followed by a dot.

Application examples

Transfer parameters:

V.E	0
-----	---

V.E identifier of variable type: V.E

0 zero termination

Return values:

123

123 V.E. variables are parameterised in Channel 1.

4.3.2 “Variable description by index” function

IndexGroup: IdxGrp = 0x122300 + Channel ID

Index offset: IdxOffs = 0x45

Variable description	This function supplies the associated variable description for specified index and variable type. The total number of existing variable entries can be queried by the “Variable element number” function. The valid range of the index is from 0 to (number of elements – 1).
-----------------------------	--

Transfer parameter (here with stream offset specified):

0	4
Variable index	Identifier for variable type with zero termination

where

Contents	Type	Size
Variable index	UNS32	fixed, 4 bytes
Variable type identifier	STRING “V.G” - Global variables “V.E” - External Variable “V.A” - Axis-specific variables	variable, max. 256 bytes

=> stream length max. 260 bytes

Return value (here with stream offset specified):

0	4	8	12	16	(20)	(24)	...	20 + 4 * m
Handle on the variable	Variable type	Access	Size	Number of array dimensions m	Number of elements in dimension 1	Number of elements in dimension 2	...	Variable name

where

Contents	Variable type	Size
Handle on the variable	UNS32	fixed, 4 bytes
Variable type	Enumeration (UNS32) (*)	fixed, 4 bytes
Access	Enumeration (UNS32) 1 = READ_ONLY 2 = READ_WRITE	fixed, 4 bytes
Size	UNS32	fixed, 4 bytes
Number of array dimensions m	UNS32 (= 0, elementary variable)	fixed, 4 bytes
Number of elements Index: 0 – (n-1) per array dimension	UNS32	variable, 4 bytes * m
Variable name	String	variable, max. 256 bytes

=> stream length 276 bytes + number of array dimensions n * 4 bytes, max. 2048 bytes

(*) Enumeration for variable type:

1 = BOOLEAN	4 = SGN16	7 = UNS32
2 = SGN08	5 = UNS16	8 = REAL64
3 = UNS08	6 = SGN32	9 = STRING



The variable type identifier, e.g. V.E, may not be followed by a dot.

Application examples

The example below reads the description of the V.E variable with index 2.

IndexGrp: 0x122301 for the first channel

IndexOffs: 0x45

Transfer parameters:

2	V.E	0
---	-----	---

2 for the variable with index entry 2

V.E identifier of variable type: V.E

0 zero termination

Return values:

0x2000002	1	1	2	1	2	V.E.Variable_3
-----------	---	---	---	---	---	----------------

0x2000002 Handle

1 type of variable: Boolean

1 access rights: Read-Only

2 total size in bytes

1 array dimension

2 number of array elements

V.E.Variable_3 Variable name

4.3.3 “Variable handle by name” function

IndexGroup: IdxGrp = 0x122300 + Channel ID

Index offset: IdxOffs = 0x46

Handle by name	<p>Transfers the handle belonging to the variable name for access to the variable. With axis-specific variables the handle is dependent on the position (axis index) of the axis to which it is currently assigned in the channel. Therefore, if the axis is assigned to a different axis index by exchange, the handle must again be determined.</p> <p>With structures such as user-defined V.E variables, the structure element name must be specified. The structure is resolved into its single elements.</p>
-----------------------	--



The handle is only valid during controller runtime. It must be read again when the controller is restarted.

Transfer parameter (here with stream offset specified):

0
Variable name with zero termination

where

Contents	Type	Size
Variable name (with instance identifier for axis-specific variables)	STRING	variable, max. 256 bytes

=> stream length max. 256 bytes

Return value (here with stream offset specified):

0
Handle on the variable

where

Contents	Type	Size
Handle on the variable	UNS32	fixed, max. 4 bytes

=> streaming length max. 4 bytes

Application examples

Transfer parameters:

V.E.vartest	0
-------------	---

V.E.vartest variable name

0 zero termination

Return value:

0x2000004

0x2000004 Handle of requested V.E variable

4.3.4 “Read variable value by handle” function

Index group: IdxGrp = 0x122300 + Channel ID

Index offset: IdxOffs = 0x47

Read by handle	This function reads the value of a variable by its handle.
-----------------------	--

Transfer parameter (here with stream offset specified):

0	4	8	12	8 + 4 Byte * n
Handle on the variable	Number n Array dimensions	1. index	2nd index	...

where

Contents	Type	Size
Handle on the variable	UNS32	fixed, 4 bytes
Number n of array dimensions	UNS32	fixed, 4 bytes
1. – n. array index	UNS32	variable, n * 4 bytes

=> stream length 8 bytes + number of array dimensions n * 4 bytes, max. 2048 bytes

Return value (here with stream offset specified):

0
variable value

where

Contents	Type	Size
variable value	String, REAL64, SGN32, REAL64[], SGN32[][], ...	variable, max. 2048 bytes

=> stream length max. 2048 bytes

With an array variable, a single value as well as the complete field in which no array indices are transferred can be read.

Application examples

1. Example of reading an array element of a unidimensional array:

Transfer parameters:

Handle on the variable	Number of array dimensions	1. index	Return value
Handle on the array 0x20012341	1	4	Value of 5th element V.G.array[4]

0x20012341 Handle on the variable

1 number of array dimensions

4 for Index 4 therefore the value of the 5th Element of the variable V.E.test[] → V.E.test[4]

Return value:

333

333 variable value

2nd Example of reading a complete array with 5 elements:

Handle on the variable	Number of array dimensions	1. index	Return value
Handle on the array 0x2001234	0		Value of all array elements 0 to (n-1)

0x2001234 Handle on the variable

0 number of array dimensions

Return values:

123	234	345	456	567
-----	-----	-----	-----	-----

123 value of 1st element

234 value of 2nd element

345 value of 3rd element

456 value of 4th element

567 value of 5th element

4.3.5 “Read variable value by name” function

IndexGroup: IdxGrp = 0x122300 + Channel ID

Index offset: IdxOffs = 0x48

Read by name	This function reads the value of a variable by its name.
---------------------	--

Transfer parameters: (here with stream offset specified)::

0
Variable name with zero termination

where

Contents	Type	Size
Variable name (with instance identifier for axis-specific variables)	STRING	variable, max. 256 bytes

=> stream length max. 256 bytes

Return value: (here with stream offset specified)::

0
variable value

where

Contents	Type	Size
variable value	String, REAL64, SGN32, REAL64[], SGN32[][], ...	variable, max. 2048 bytes

=> stream length max. 2048 bytes

With an array variable, a single value as well as the complete field in which no indices are specified in the variable name can be read.

Example of variable V.G.array[0 .. 9]:

Variable name	Return value
V.G.array[4]	Value of 5th element V.G.array[4]
V.G.array	Value or array elements 0 - 9

Application examples

Example of reading an array element of a 1-dimensional array:

Transfer parameters:

V.E.Testvar[3]	0
----------------	---

V.E.Testvar[3] Variable name

0 zero termination

Return value:

333

333 variable value of the element

Example of reading a complete array with 5 elements:

Transfer parameters:

V.E.Testvar	0
-------------	---

V.E.Testvar Name of the variable

0 zero termination

Return values:

123	234	345	456	567
-----	-----	-----	-----	-----

123 value of 1st element

234 value of 2nd element

345 value of 3rd element

456 value of 4th element

567 value of 5th element

4.3.6 “Write variable value by handle” function

IndexGroup: IdxGrp = 0x122300 + Channel ID

Index offset: IdxOffs = 0x49

Write by handle	This function changes the value of a variable by its handle.
------------------------	--

Transfer parameters: (here with stream offset specified)::

0	4	8	12	...	8 + 4 * n
Handle on the variable	Number n Array indices	1st index	2nd index	...	variable value

where

Contents	Type	Size
Handle on the variable	UNS32	fixed, 4 bytes
Number n of transferred array indices	UNS32	fixed, 4 bytes
1st – nth array index	UNS32	variable, n * 4 bytes
variable value	String, REAL64, SGN32, REAL64[], SGN32[][], ...	variable, max. 2040 bytes

=> stream length max. 2048 bytes

This function has no return value.

With an array variable, a single value as well as the complete field in which no array indices are transferred can be written.

Example of variable V.G.array[0 .. 9]:

Handle on the variable	Number of array indices	1st index	variable value
Handle on the array	1	4	Value of 5th element V.G.array[4]
Handle on the array	0		Value or array elements 0 - 9

Application examples

1. Example of writing an array element of a unidimensional array:

Transfer parameters:

0x2001234	1	4	123
-----------	---	---	-----

0x2001234 Handle of variable

1 number of array dimensions

4 for the index 4 therefore the 5th element of the V.E.test[] → V.E.test4[]

123 new value of 5th element of the variable

Return value: -

2nd Example of writing a complete array with 5 elements:

0x2001234	0	123	234	345	456	567
-----------	---	-----	-----	-----	-----	-----

0x2001234 Handle of variable

0 number of array dimensions

123 new value of 1st element of the variable

234 new value of 2nd element of the variable

345 new value of 3rd element of the variable

456 new value of 4th element of the variable

567 new value of 5th element of the variable

Return value: -

4.3.7 “Write variable value by name” function

IndexGroup: IdxGrp = 0x122300 + Channel ID

Index offset: IdxOffs = 0x4A

Write via Name	This function changes the value of a variable by its name. With axis-specific variables the axis name must contain the instance identifier (e.g. axis name V.A.ACT_POS.X).
-----------------------	--

Transfer parameter (here with stream offset specified):

0	1...256
Variable name with zero termination	variable value

where

Contents	Type	Size
Variable name (with instance identifier for axis-specific variables)	String	variable, max. 256 bytes
variable value	String, REAL64, SGN32, REAL64[], SGN32[][], ...	variable, max. 2047 bytes

=> stream length max. 2048 bytes

This function has no return value.

With an array variable, a single value as well as the complete field in which no indices are specified in the variable name can be written.

Example of variable V.G.array[0 .. 9]:

Variable name	variable value
V.G.array[4]	Value of 5th element V.G.array[4]
V.G.array	Value or array elements 0 - 9

Application examples

1. Example of writing an array element of a unidimensional array:

Transfer parameters:

V.E.test[4]	0	123
-------------	---	-----

V.E.Testvar[4] Variable name

0 zero termination

123 new value of 5th element of the variable

Return value: -

2nd Example of writing a complete array with 5 elements:

V.E.test	0	123	234	345	456	567
----------	---	-----	-----	-----	-----	-----

V.E.Testvar[4] Variable name

0 zero termination

123 new value of 1st element of the variable

234 new value of 2nd element of the variable

345 new value of 3rd element of the variable

456 new value of 4th element of the variable

567 new value of 5th element of the variable

Return values: -

4.3.8 “Extended description by index” function

Index group: IdxGrp = 0x122300 + Channel ID

Index offset: IdxOffs = 0x4B

Extended information:	This function supplies extended information about the variable. The transferred data is dependent on the variable type. The valid range of the index is from 0 to (number of elements – 1)
------------------------------	---

Transfer parameter (here with stream offset specified):

0	4
Variable index	Variable type identifier

where

Contents	Type	Size
Variable index	UNS32	fixed, 4 bytes
Variable type identifier	STRING “V.G” - Global variables “V.E” - External Variable “V.A” - Axis-specific variables	variable, max. 256 bytes

=> stream length max. 260 bytes

Return value: (here with stream offset specified):

The return value is dependent on the variable type:

) V.G, V.A variables -> no return value since no extended description exists.

b.) V.E. variables:

0	4
Synchronisation	Validity range

where

Contents	Type	Size
Synchronisation	Enumeration (UNS32) 1 = NO_SYNC 2 = SYNCHRONISED	fixed, 4 bytes
Validity range	Enumeration (UNS32) 1 = CHANNEL 2 = GLOBAL	fixed, 4 bytes

=> stream length of V.E. Variable 8 bytes max. 2048 bytes



The variable type identifier, e.g. V.E, may not be followed by a dot.

Application examples

Example of reading the extended properties of a V.E. Variable:

Transfer parameters:

3	V.E	0
---	-----	---

3 for the variable with index entry 3

V.E identifier of variable type: V.E

0 zero termination

Return values:

2	2
---	---

2 Synchronisation: SYNCHRONIZED

2 Validity range: GLOBAL

4.3.9 “Variable handle by name” axis-specific function

Index group: IdxGrp = 0x122300 + Channel ID

Index offset: IdxOffs = 0x10004 for axis 1

= 0x20004 for axis 2

Handle by name	This function returns the handle for the specified variable name which belong to an axis assigned to this position (axis index). For this reason an exchange can change the variable value read by the handle.
-----------------------	--

Transfer parameter (here with stream offset specified)::

0
Variable name

where

Contents	Type	Size
Variable name (without instance identifier) with zero termination.	STRING	variable, max. 256 bytes

=> stream length max. 256 bytes

Return value (here with stream offset specified):

0 4
Handle on the variable

where

Contents	Type	Size
Handle on the variable	UNS32	fixed, max. 4 bytes

=> streaming length max. 4 bytes

Application examples

Example of reading a handle of an axis-specific variable.
The index offset is important here.

Transfer parameters:

Index offset: 0x20004

V.A.WCS	0
---------	---

V.A.WCS name of the variable (without instance identifier V.A.WCS.Y)

0 zero termination

Return values:

0x0102001b

0x0102001b Handle of the variable, here the 2nd axis

4.3.10 “Read variable value by name” axis-specific function

IndexGroup: IdxGrp = 0x122300 + Channel ID

Index offset: IdxOffs = 0x10005 for axis 1

= 0x20005 for axis 2

Read by name	This function reads the value of a variable by its variable name. The variable that belongs to the axis assigned to this position (axis index) is then always accessed.
---------------------	---

Transfer parameter (here with stream offset specified):

0
Variable name with zero termination

where

Contents	Type	Size
Variable name (without instance identifier)	STRING	variable, max. 256 bytes

=> stream length max. 256 bytes

Return value (here with stream offset specified):

0
variable value

where

Contents	Type	Size
variable value	String, REAL64, SGN32, REAL64[], SGN32[][], ...	variable, max. 2048 bytes

=> stream length max. 2048 bytes

Application examples

Example of reading an axis-specific variable. The variable V.A.WCS.Y, i.e. the 2nd axis is read by the index offset 0x20005

Transfer parameters:

V.A.WCS	0
---------	---

V.A.WCS name of the variable (without instance identifier V.A.WCS.Y)

0 zero termination

Return values:

1234.2

1234.2 Value of the variable of the 2nd axis

4.3.11 “Write variable value by name” axis-specific function

Index group: IdxGrp = 0x122300 + Channel ID

Index offset: IdxOffs = 0x10006 for axis 1

= 0x20006 for axis 2

Writing by name	This function changes the value of a variable by its variable name. The variable that belongs to the axis assigned to this position (axis index) is then always accessed.
------------------------	---

Transfer parameter (here with stream offset specified):

0	1- 256
Variable name with zero termination	variable value

where

Contents	Type	Size
Variable name e.g. V.A.WCS (without instance identifier V.A.WCS.Y)	String	variable, max. 256 bytes
variable value	String, REAL64, SGN32, REAL64[], SGN32[][], ...	variable, max. 2047 bytes

=> stream length max. 2048 bytes

This function has no return value.

Application examples

Example of writing an axis-specific variable. The variable V.A.WCS.Y, i.e. the 2nd axis is written:

Index offset: 0x20006

Transfer parameters:

V.A.WCS	0	123,456
---------	---	---------

V.A.WCS name of the variable (without instance identifier V.A.WCS.Y)

0 zero termination

123.456 New value of the variable

Return value: -

5 ADS services

Interface for ADS services

The CNC instances offer an interface for access to the internal elements by the following ADS services:

Data is addressed by the Index Group and the Index Offset.

ADS Read

The ADS Read function read data from an ADS unit.

In	IndexGroup, 4 bytes, index group of the data to be read. IndexOffset, 4 bytes, index offset of the data to be read. Length, 4 bytes, length in bytes of the data to be read.
Out	Result, 4 bytes, ADS error number Length, 4 bytes, length in bytes of the data returned. Data, n bytes, data returned

ADS Write

Data is written to an ADS unit with ADS Write.

In	IndexGroup, 4 bytes, index group of the data to be read. IndexOffset, 4 bytes, index offset of the data to be read. ReadLength, 4 bytes, length in bytes of the data to be read. WriteLength, 4 bytes, length in bytes of the data to be written. Data, n bytes, data to be written
Out	Result, 4 bytes, ADS error number

ADS Read & Write

The *ADS ReadWrite* function writes data to an ADS unit. Data can also be read out of the ADS unit.

In	IndexGroup, 4 bytes, index group of the data to be read. IndexOffset, 4 bytes, index offset of the data to be read. ReadLength, 4 bytes, length in bytes of the data to be read. WriteLength, 4 bytes, length in bytes of the data to be written. Data, n bytes, data to be written
Out	Result, 4 bytes, ADS error number Length, 4 bytes, length in bytes of the data returned. Data, n bytes, data returned

ADS – Automation Device Specification (source: Beckhoff Help)

6 Support and Service

Beckhoff and their partners around the world offer comprehensive support and service, making available fast and competent assistance with all questions related to Beckhoff products and system solutions.

Download finder

Our [download finder](#) contains all the files that we offer you for downloading. You will find application reports, technical documentation, technical drawings, configuration files and much more.

The downloads are available in various formats.

Beckhoff's branch offices and representatives

Please contact your Beckhoff branch office or representative for [local support and service](#) on Beckhoff products!

The addresses of Beckhoff's branch offices and representatives round the world can be found on our internet page: www.beckhoff.com

You will also find further documentation for Beckhoff components there.

Beckhoff Support

Support offers you comprehensive technical assistance, helping you not only with the application of individual Beckhoff products, but also with other, wide-ranging services:

- support
- design, programming and commissioning of complex automation systems
- and extensive training program for Beckhoff system components

Hotline: +49 5246 963-157
e-mail: support@beckhoff.com

Beckhoff Service

The Beckhoff Service Center supports you in all matters of after-sales service:

- on-site service
- repair service
- spare parts service
- hotline service

Hotline: +49 5246 963-460
e-mail: service@beckhoff.com

Beckhoff Headquarters

Beckhoff Automation GmbH & Co. KG

Huelshorstweg 20
33415 Verl
Germany

Phone: +49 5246 963-0
e-mail: info@beckhoff.com
web: www.beckhoff.com

Index

More Information:
www.beckhoff.com/TF5200

Beckhoff Automation GmbH & Co. KG
Hülshorstweg 20
33415 Verl
Germany
Phone: +49 5246 9630
info@beckhoff.com
www.beckhoff.com

