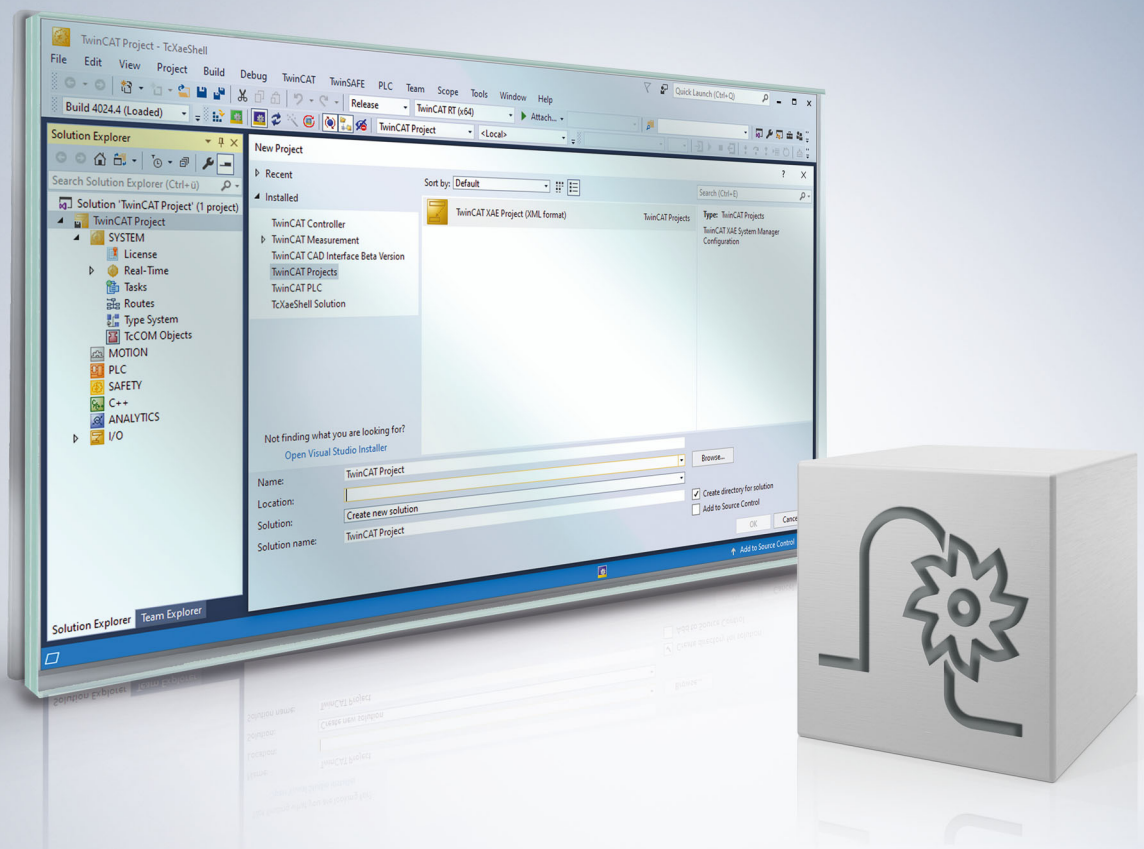
BECKHOFF New Automation Technology

Manual | EN

TF5200 | TwinCAT 3 CNC

External variables



6/5/2025 | Version: 1.16

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

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

List of figures

1 Overview of EXTV parameters

The overview of External Variable parameters is sorted into a 4-column table.

- Column 1 contains the unambiguous identifier of the start-up parameter called the “ID” which consists of the prefix “P-EXTV” and a unique 5-digit number,
  e.g. P-EXTV-00001.
- Column 2 represents the data structure which defines the parameter,
  e.g. var[i]
  The structure is a categorisation aid and is described in the following section. If an entry is missing in ‘structure’, this is not an error. The parameter in column 3 is then only valid on its own.
- Column 3 contains the “parameter” with its exact name,
  e.g. name
  The important thing is that “structure”+”parameter” always belong together and must be configured in exactly the same way in the External Variables list,
  e.g. var[i].name
- Column 4 contains the “functionality” in a summarised term/short description,
  e.g. name of external variable.

ID	Structure	Parameter	Functionality/short description
P-EXTV-00001 [▶ 23]	var[i].	name	Name of the External Variable
P-EXTV-00002 [▶ 23]	var[i].	byte_offset	Position of the External Variable in the memory
P-EXTV-00003 [▶ 23]	var[i].	type	Variable type
P-EXTV-00004 [▶ 24]	var[i].	scope	Validity of the variable
P-EXTV-00005 [▶ 24]	var[i].	synchronisation	Synchronisation type of the variable
P-EXTV-00006 [▶ 24]	var[i].	access_rights	Access rights of the variable
P-EXTV-00007 [▶ 25]	var[i].	array_elements	Number of elements in an array of structure elements.
P-EXTV-00008 [▶ 25]	var[i].	size	Size of the variable
P-EXTV-00009 [▶ 25]	var[i].	create_hmi_interface	Enabling access by HMI
P-EXTV-00010 [▶ 26]		number_used_variables	Number of configured external variables
P-EXTV-00011 [▶ 26]		check_overlapping_variables	Plausibility check of the memory layout
P-EXTV-00012 [▶ 26]		auto_memory_mode	automatic memory layout
P-EXTV-00013 [▶ 29]		init	Initialisation with default values
P-EXTV-00015 [▶ 19]	struct[i].	name	Name of the variable type
P-EXTV-00016 [▶ 20]	struct[i].element[j].	name	Name of the structure element
P-EXTV-00017 [▶ 20]	struct[i].element[j].	type	Type of the structure element
P-EXTV-00018 [▶ 20]	struct[i].element[j].	synchronisation	Synchronisation type of the structure element
P-EXTV-00019 [▶ 21]	struct[i].element[j].	access_rights	Access right of the structure element
P-EXTV-00020 [▶ 21]	struct[i].element[j].	array_elements	Array size of a structure element
P-EXTV-00021 [▶ 21]	struct[i].element[j].	size	Size of a structure element of type VSTRING
P-EXTV-00022 [▶ 21]		use_extended_string_var	Number of characters of a string variable
P-EXTV-00047 [▶ 25]	var[i].	suppress_export	Suppress variable export to the PLC description

ID	Structure	Parameter	Functionality/short description
P-EXTV-00048	var[i].	start	Variables with overlapping memory area

2 Function and characteristics

External variables describe a data memory which is used to exchange any values between the CNC (NC channel), the graphic user interface (GUI) and the PLC. Data can be used within the CNC either channel-specific (only accessible by one channel) or cross-channel (“global”). The meaning of the individual memory locations (here an external variable) is defined by the application (NC program, GUI and PLC).

The CNC itself only has the task of defining the memory layout and allowing the user access to the variables in the NC program. During the configuration phase, a name, a type and an access right (write and/or read) are assigned to each external variable. Optionally, memory addresses and variable size can be assigned by the user.

It is recommended to use the automated address assignment by the CNC to avoid problems caused by alignment or memory overlaps. A combination of the two address assignments is possible but this is also not recommended.

Defining the memory layout

The external variables are configured on the basis of an ASCII list only once during start-up.

User-defined structures can be defined besides “simple” variables. In addition, unidimensional arrays of simple variables or structures are possible. Access can be indexed to them.

A memory description is required to permit the GUI or a PLC to access the memory layout correctly - ultimately, this is defined by the CNC. This is made possible by the command #EXPORT VE. It is recommended to execute this command before the first start of the PLC. The command generates declaration files which can be directly integrated in the PLC programming environment. This obviates the need for an extremely (!) error-prone manual simulation of the memory layout by the GUI or PLC.

NOTICE

If addresses and types are not identical between the CNC and the PLC or GUI variables, the CNC has no possibility of checking this or preventing an error response.

The only way to protect the CNC from an invalid REAL64 parameter - which will cause a controller crash - is to check for "1.#INF", "-1.#INF" and "1.#SNAN” patterns. This check takes place when an external variable is read and, in case of error, it results in the output of messages with the ID 21820 or ID 21821.

The values of local variables apply throughout the lifetime of an NC program, i.e. they are not cleared, for instance when a new NC program is started.

Writing/reading instants

There are two options to determine the instant of writing/reading a variable from the NC program:

- Access to a variable is performed synchronously in the interpolator to execute the NC program, i.e. this ensures the temporal sequence of NC commands and variable access operations.
- Writing/reading a variable is executed at the instant of decoding (asynchronous with the execution of the NC program in the interpolator, i.e. “in advance”).

The user must evaluate the advantages and disadvantages of the two variants.

Synchronous read accesses cause the decoder to stop until the synchronous read value is available to the decoder. In addition, it is not permitted to read synchronous variables, e.g. with functions such as an enabled Tool radius compensation; the message is output with the ID 20651.

Access to the data by the GUI or PLC

- is not synchronised with NC program processing. This means that it cannot be ensured when data is read or written, i.e. when data is valid for the viewpoint of an application.
- In the same way, it cannot be guaranteed that data is consistently transferred in structures or arrays.

The only thing that can be guaranteed (by the processor) is correct access to the basic types of byte, word, doubleword and REAL64. From a programming viewpoint, the user must guarantee access protection with regard to instant and completeness.

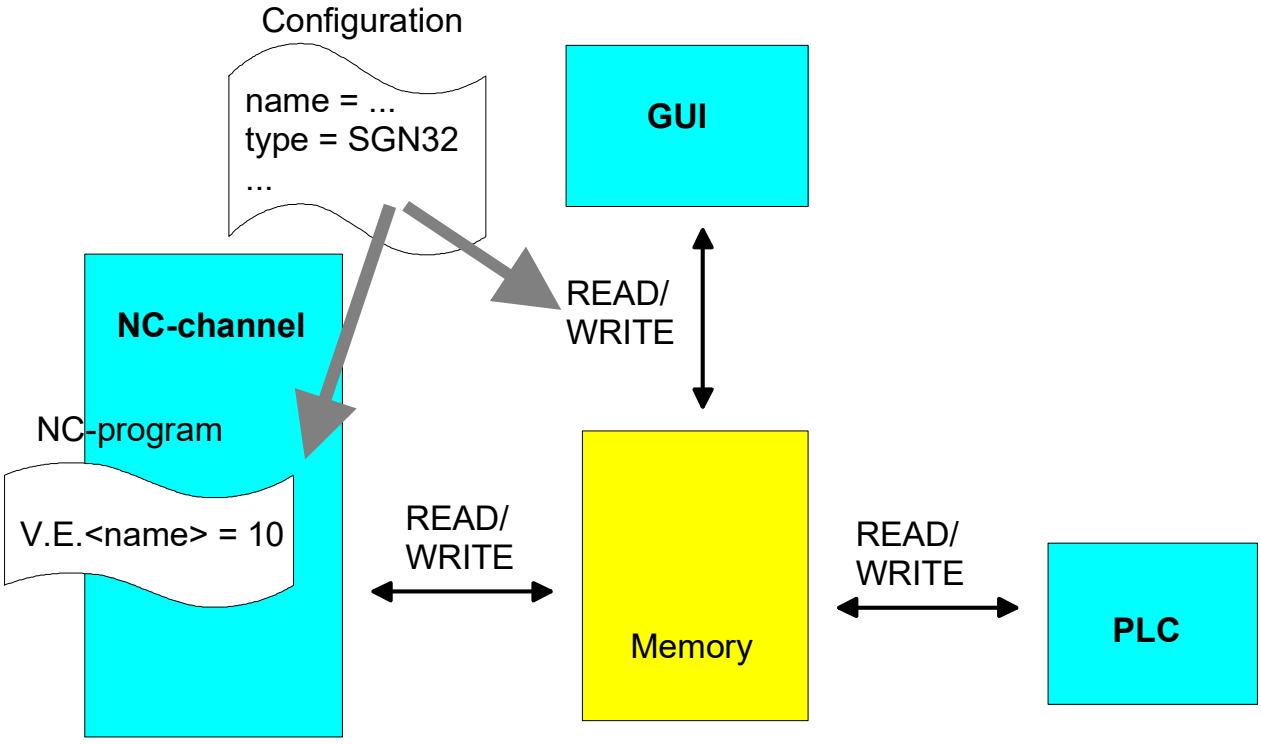


Fig. 1: Use of external variables

3 Configuration and initialization

3.1 Memory size

The entire memory available for the external variables of each channel must be defined before controller start-up.

With TwinCAT systems, dimensioning is defined in the VE entry of the HLI mask in the System Manager (CNC Task-GEO).

The value defined here determines the number of 24 byte blocks which compose the V.E. memory of each channel. The global external variables (cross-channel scope) use an own memory of equal size. The reserved V.E. memory of each channel and the global memory must always be larger than the memory area occupied by the memory layout [▶ 12] .

If the available memory is not large enough for the external variables configured in the list, the error message P-ERR-21519 is output. In this case, either the reserved memory must be expanded or the size and number of configured external variables must be reduced.

3.2 Memory layout

Each channel has access to two different memory areas: One of them is only available locally for the current NC channel and the other is cross-channel, i.e. it can be shared by all NC channels. The configuration of external variables is specified in a file for each NC channel and each of these files lists both global and channel-specific variables.

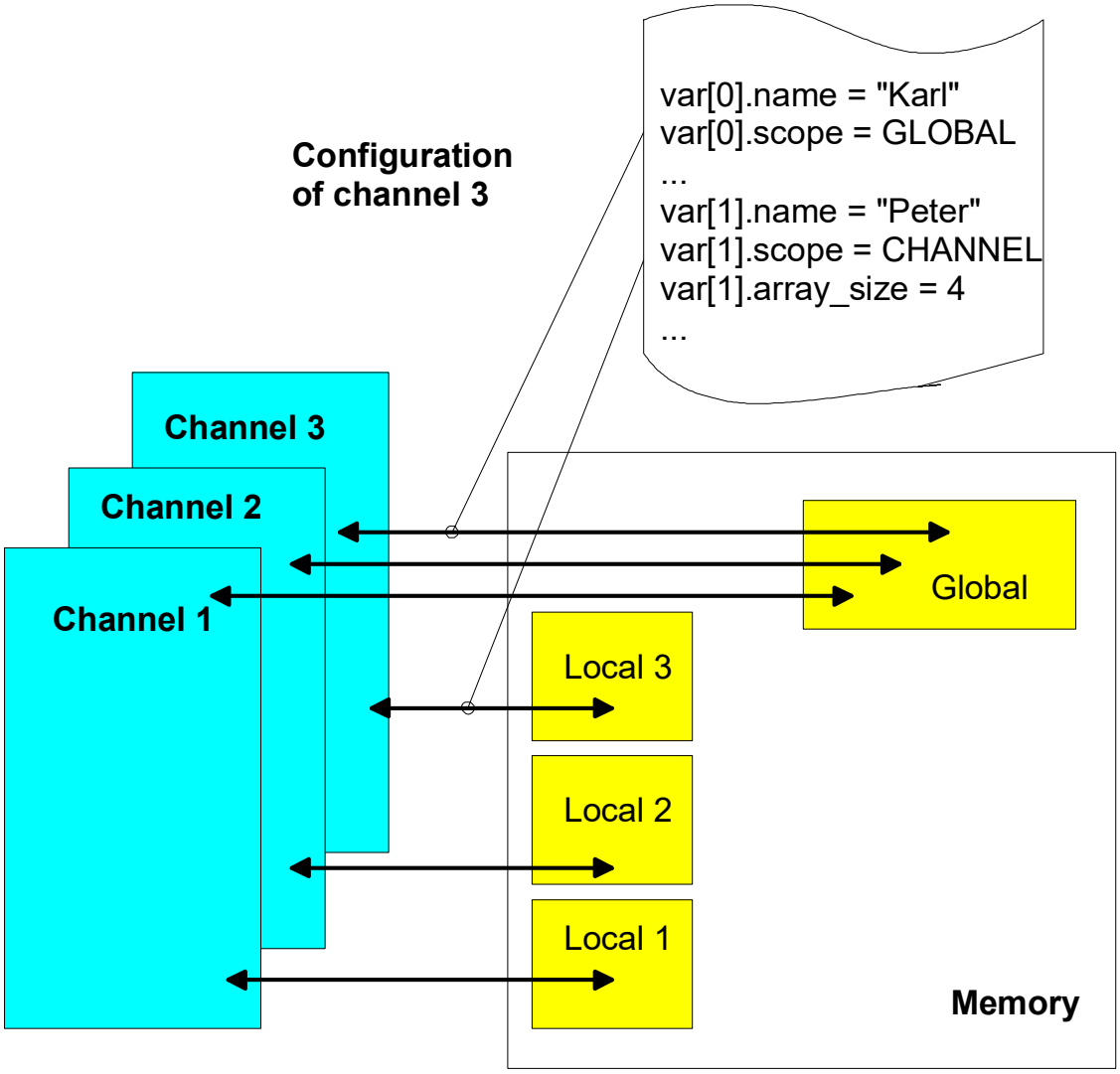


Fig. 2: Scope of validity of the memories

The memory location of a V.E. variable can be defined manually or the CNC defines the correct alignment and memory requirement. In the first case, the user is responsible for the error-free definition of the memory layout. In the second case, there is no risk of incorrect configuration.

If the memory layout is defined by manual configuration, V.E. variables can be assigned to any location in the V.E. memory by specifying an address offset (see P-EXTV-00002 [▶ 23]). The parameter P-EXTV-00008 [▶ 25] defines the alignment to one of the variables located behind it in the memory. With more complex structures, this may be important if, for example, the next variable is a REAL64 type.

If user-defined variable types are used, byte alignment must be used for the structures in the PLC.

Manual address assignment is not recommended since the memory could be then be fragmented and overlapping. This can only be checked on request (see P-EXTV-00011 [▶ 26]). On the other hand, when automatic address assignment is used, variables are saved one after the other in the V.E. Memory without gaps and with the correct alignment (see figure).

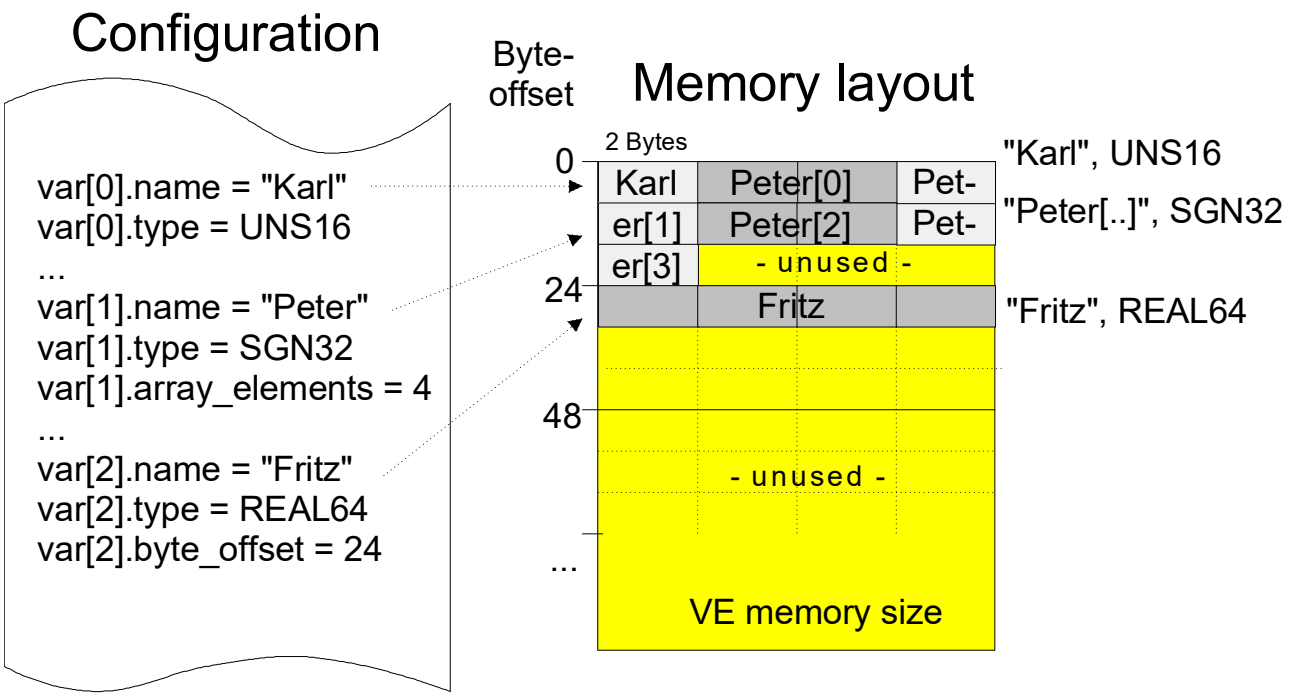


Fig. 3: Memory layout resulting from the given configuration

NOTICE

As of CNC Build V3.00.3019 and higher, the external variables and structures are created with an 8-byte alignment. If a fixed start offset is specified for a variable by the parameter byte_offset (P-EXTV-00002 [▶ 23]) or the index (P-EXTV-00038 [▶ 46]) or size (P-EXTV-00008 [▶ 25]) is used, the user is responsible for the correct address assignment. No alignment calculation takes place.

Compatibility of old configuration lists

Configuration lists configured with an older syntax may continue to be used.
See Memory layout up to V2.10.1025 [▶ 45]

With multichannel configurations, we recommend the use of the same global entries of V.E. variables in all channels.

3.3 Parameterising the memory layout of V.E. variables

General parameters

ID	Parameter	Meaning
P-EXTV-00010 [▶ 26]	number_used_variables	Number of configured external variables
P-EXTV-00011 [▶ 26]	check_overlapping_variables	Plausibility check of the memory layout
P-EXTV-00012 [▶ 26]	auto_memory_mode	automatic memory layout
P-EXTV-00013 [▶ 29]	init	Initialisation with default values
P-EXTV-00022 [▶ 21]	use_extended_string_var	Number of characters of a string variable

Parameter for V.E. structures

ID	Parameter	Meaning
P-EXTV-00015 [▶ 19]	struct[i].name	Name of the variable type
P-EXTV-00016 [▶ 20]	struct[i].element[j].name	Name of the structure element
P-EXTV-00017 [▶ 20]	struct[i].element[j].type	Type of the structure element
P-EXTV-00018 [▶ 20]	struct[i].element[j]. synchronisation	Synchronisation type of the structure element
P-EXTV-00019 [▶ 21]	struct[i].element[j]. access_rights	Access right of the structure element
P-EXTV-00020 [▶ 21]	struct[i].element[j]. array_elements	Array size of a structure element
P-EXTV-00021 [▶ 21]	struct[i].element[j].size	Size of a structure element of type VSTRING

V.E. Parameters of a variable

ID	Parameter	Meaning
P-EXTV-00001 [▶ 23]	var[i].name	Name of external variable
P-EXTV-00002 [▶ 23]	var[i].byte_offset	Position of external variable in the memory
P-EXTV-00003 [▶ 23]	var[i].type	Variable type
P-EXTV-00004 [▶ 24]	var[i].scope	Validity of the variable
P-EXTV-00005 [▶ 24]	var[i].synchronisation	Synchronisation type of the variable
P-EXTV-00006 [▶ 24]	var[i].access_rights	Access rights of the variable
P-EXTV-00007 [▶ 25]	var[i].array_elements	Number of elements in an array of structure elements
P-EXTV-00008 [▶ 25]	var[i].size	Size of the variable
P-EXTV-00009 [▶ 25]	var[i].create_hmi_interface	Enabling the access by HMI
P-EXTV-00047 [▶ 25]	var[i].suppress_export	Suppress variable export to the PLC description
P-EXTV-00048	var[i].start	Variables with overlapping memory area

3.4 syntax:

In the following description of the configuration of external variables, a distinction is made between the definition of variable types and the instantiation of variables.

3.4.1 Comments in the ASCII list file

Comments can be in an entire line or can be added at the end of a line.

With a comment spanning an entire line, the comment character "'#" must be placed at the start of the line and followed by a blank.

If a comment is to be inserted at the end of a line, only a blank is required before the comment. Blank lines are also possible.

Comments in the ASCII list file

```
# ****************************************************************************
# Data
# ****************************************************************************
#
# List comments after numerical values

dummy[1] 1 Comment
dummy[2] 1 # Comment
dummy[3] 1 ( Comment
dummy[4] 1 /* Comment
...
...
```

However, if a character string was assigned to the list parameter as a value in the line, any following comment must be opened by a bracket '('. The comment characters space, # and /* are not permitted.

If a '(' itself is part of the character string, the character string must be defined in inverted commas ″..″ (available as of V3.1.3081.0, V3.1.3108.0).

```
# List comments after strings

beispiel[0].bezeichnung STRING_1 (comment requires a '(' bracket!)

beispiel[1].bezeichnung″ STRING_(2)″ (comment requires a '(' bracket!)
```

3.4.2 Syntax and interpretation of ASCII list file

An interpreter copies the entries in the ASCII list file into identical internal structures which are then checked for plausibility. To ensure reliable controller start-up every time, any defective entries found by the plausibility check are replaced by default values.

Unknown entries are not taken over. These irregularities are displayed by warning messages. We advise you to investigate the cause for these warning messages and remove defective entries from the ASCII list file.

The following agreement applies to BOOLEAN data:

Value	Meaning
0	Definition of FALSE
1	Definition of TRUE

The following agreement applies to STRING data:

If a character string containing characters with a special meaning in ASCII lists (e.g. comment characters, spaces [▶ 16]) is assigned to a STRING type list parameter, this character string must be defined in inverted commas "..." (available as of V3.1.3081.0, V3.1.3108.0).

```
example[0].name "STRING_WITH_COMMENT( # /*)_CHARACTERS"
```

Trailing spaces are discarded on import. The entry..

```
example[0].name "STRING_WITH_POST_SPACES "
```

..has the same meaning as

```
example[0].name "STRING_WITH_POST_SPACES"
```

If the character string only contains characters without any special meaning, no inverted commas are required.

```
example[0].name STRING_WITH_STANDARD_CHARACTERS!
```

3.4.3 Data sets for definition of variable types (struct[i].*)

Besides user-defined variable types, the elementary data types (BOOLEAN, ..., REAL64) and the character strings of the STRING type can also be used.

Size of default variable types

Variable type	Size
BOOLEAN, UNS08, SGN08	1 byte
UNS16, SGN16	2 bytes
UNS32, SGN32	4 bytes
REAL64	8 bytes
STRING	128 bytes As of CNC Build V2.10.1025.00 and higher, string variables may contain up to 128 characters including the termination mark. For downgrade compatibility reasons, the parameter use_extended_string_var must be set to TRUE in order to use string variables with more than 20 characters the parameter (P-EXTV-00022 [▶ 21]). Depending on the parameter use_extended_string_var, string variables are limited to the following length (including the termination mark): • use_extended_string_var = FALSE: 21 bytes (default) • use_extended_string_var = TRUE: 128 Byte
VSTRING	String with variable size between 1 and 800 bytes. From CNC version V.3.1.3039.00 onwards the CNC supports the data type VSTRING which can be used for the definition of string variables with individual size. The required size of the character string including zero termination is specified for structure elements in the parameter P-EXTV-00021 [▶ 21] or for variables in the parameter P-EXTV-00008 [▶ 25]. The default size of the VSTRING data type is 128 bytes.

NOTICE

Later configuration to 128 byte string variables

If there is a later change to 128 byte size string variables (use_extended_string_var [▶ 21] = TRUE) within an existing configuration, the addresses of the external variables in the memory (index, byte offset) and in the PLC must be adapted to avoid overwriting the following variables.

The variable types are defined in the same configuration list as the definition of an instance of an external variable.

In the configuration file, comments can be whole lines or can be added at the end of a line. With a comment spanning an entire line, the comment character "'#" must be placed at the start of the line followed by a blank.

If a comment is to be inserted at the end of a line, there only needs to be a blank before the comment. However, if a string was defined in the line, the comment must be preceded by the comment character "(".

Blank lines are also possible.

The key word 'struct' replaces the common token 'type' used in older CNC versions. However, due to downward compatibility, the token 'type' continues to be supported.

Structure name	Index
struct[i]	i = 0 ... 49 (Maximum number of data sets for the definition of external variable types)

3.4.3.1 Name of the variable type (P-EXTV-00015)

P-EXTV-00015	Name of the variable type
Description	The variable type is identified by the name. Upper case letters and lower case letters are differentiated.
Parameter	struct[i].name
Data type	STRING
Data range	Maximum 26 characters (length of variable name as of CNC Build 2.10.1504 and higher *)
Dimension	----
Default value	-
Remarks	* Maximum 20 characters before CNC Build 2.10.1504

3.4.4 Data sets for definition of the elements of a variable type (struct[i].element[j].*)

Structure name	Index
struct[i].element[j]	j = 0 ... 49 (Maximum number of data sets for definition of the elements of a variable type)

3.4.4.1 Name of the structure element (P-EXTV-00016)

P-EXTV-00016	Name of the structure element
Description	The structure element is identified by the name. Upper case and lower case letters are recognised.
Parameter	struct[i].element[j].name
Data type	STRING
Data range	Maximum of 20 characters
Dimension	----
Default value	-
Remarks	

3.4.4.2 Type of the structure element (P-EXTV-00017)

P-EXTV-00017	Type of the structure element
Description	The type indicates the data type of the structure element. Besides elementary data types (SGN08, ..., REAL64), the data type STRING or a user-defined data type can also be defined here.
Parameter	struct[i].element[j].type
Data type	STRING
Data range	BOOLEAN, SGN08, UNS08, SGN16, UNS16, SGN32; UNS32, REAL64, STRING, VSTRING or user-defined
Dimension	----
Default value	UNS32
Remarks	

3.4.4.3 Synchronisation type of the structure element (P-EXTV-00018)

P-EXTV-00018	Synchronisation type of the structure element
Description	By default, all structure elements inherit the synchronisation type for write/read access (see Access synchronisation by NC channel [▶ 37]) from variable instantiation. This parameter defines the synchronisation type for each structure element.
Parameter	struct[i].element[j].synchronisation
Data type	BOOLEAN
Data range	TRUE, FALSE
Dimension	----
Default value	TRUE *
Remarks	* The synchronisation type is inherited from variable instantiation (see Data sets to define external variables (var[i].*) [▶ 23])

3.4.4.4 Access right of the structure element (P-EXTV-00019)

P-EXTV-00019	Access right of the structure element
Description	By default, all structure elements inherit the access rights from variable instantiation. This parameter specifies the individual definition of the access right for the structure element.
Parameter	struct[i].element[j].access_rights
Data type	STRING
Data range	READ_WRITE, READ_ONLY, WRITE_ONLY, INHERIT_ACCESS
Dimension	----
Default value	INHERIT_ACCESS
Remarks	The access right is inherited from variable instantiation (see Data sets to define external variables (var[i].*) [▶ 23])

3.4.4.5 Array size of the structure element (P-EXTV-00020)

P-EXTV-00020	Number of elements in an array of structure elements
Description	If the structure element is an array composed of structure elements, the number of elements must be specified. If the size is 0, the structure element is a single variable and not an array.
Parameter	struct[i].element[j].array_elements
Data type	UNS16
Data range	1 ... MAX(UNS16)
Unit	----
Default value	0
Remarks	

3.4.4.6 Size of a structure element of type VSTRING (P-EXTV-00021)

P-EXTV-00021	Size of a structure element of type VSTRING (Data type with variable string length)
Description	As of CNC Build V3.1.3039.00 and higher, a new data type VSTRING can be used to define external variables. The size of a string element can be defined separately for this data type. The desired size is specified in this parameter inclusive Null-termination byte meaning a value of size = 128 leads to a string element with 127 usable characters.
Parameter	struct[i].element[j].size
Data type	UNS32
Data range	1 ... 800 bytes
Dimension	Byte
Default value	128 bytes (127 usable characters and zero termination byte)
Remarks	This parameter is available as of CNC Build V3.1.3039.00 and higher. This parameter is only used to specify the size of string elements of data type VSTRING and has no significance for other data types.

3.4.5 Definition of the number of characters of a string variable (P-EXTV-00022)

P-EXTV-00022	Number of characters of a string variable

Description	This parameter can increase the permissible number of characters of string variables from 21 to 128 characters (each including the termination mark). If the addresses of the V.E. variable is specified in 24-byte blocks (see Memory layout [▶ 12]), make sure that 128-byte variables of the STRING type are assigned several 24-byte blocks in the memory layout and that the index is incremented accordingly (cf. variable arrays).
Parameter	use_extended_string_var
Data type	BOOLEAN
Data range	TRUE, FALSE
Dimension	----
Default value	FALSE
Remarks	

3.4.6 Data sets to define external variables (var[i].*)

Structure name	Index
var[i]	i = 0 ... 214 (Maximum number of data sets for the variable definition)

3.4.6.1 Variable name (P-EXTV-00001)

P-EXTV-00001	Name of external variable
Description	The variable is identified by its name. The name is used with the prefix "V.E." to create a complete name for the NC channel and the GUI (e.g. **V.E.<name>**). This complete name is then used to program variables in the NC program. Uppercase and lowercase letters are distinguished.
Parameter	var[i].name
Data type	STRING
Data range	Maximum 26 characters (length of variable name as of CNC Build 2.10.1504 and higher *)
Dimension	----
Default value	-
Remarks	* Maximum 20 characters before CNC Build 2.10.1504

3.4.6.2 Byte offset (P-EXTV-00002)

P-EXTV-00002	Position of external variable in the memory
Description	In deviation from the data size implicitly defined by "type" (P-EXTV-00003 [▶ 23]) or the defined data size of a structure, (see Data sets for definition of the elements of a variable type (struct[i].element[j].*) [▶ 20]), the position of a variable can be defined in the memory. The indication of this parameter is optional. All variables with byte_offset = –1 are concatenated without gaps (starting with offset 0) in the V.E.memory.
Parameter	var[i].byte_offset
Data type	SGN32
Data range	0 ... MAX(SGN32)
Dimension	----
Default value	-1
Remarks	It is recommended to leave address calculations up to the controller and to not use the parameter. Any memory overlaps caused by incorrect configuration can only be detected by the parameter P-EXTV-00011 [▶ 26].

3.4.6.3 Variable type (P-EXTV-00003)

P-EXTV-00003	Type of external variable
Description	The identifier indicates the data type of the variable. Besides elementary data types (SGN08, ..., REAL64) and the STRING data type, a user-defined variable type can also be defined here (P-EXTV-00015 [▶ 19]).
Parameter	var[i].type
Data type	STRING
Data range	BOOLEAN, SGN08, UNS08, SGN16, UNS16, SGN32; UNS32, REAL64, STRING, VSTRING or user-defined
Dimension	----
Default value	UNS32

Remarks	With automatic address assignment, the parameter defines implicitly the address of the **next** variable. This considers the computer architecture and the alignment strategies saved in the controller. Read or write access to a variable always takes place using the saved type **irrespective of manual or automatic address assignment**.

3.4.6.4 Scope of validity (P-EXTV-00004)

P-EXTV-00004	**Scope of validity of external variable**
Description	In the case of the scope of validity, a distinction is made between a channel-specific and a cross-channel global scope.
Parameter	var[i].scope
Data type	STRING
Data range	GLOBAL, CHANNEL
Dimension	----
Default value	CHANNEL
Remarks	

3.4.6.5 Synchronisation type (P-EXTV-00005)

P-EXTV-00005	**Synchronisation type of external variable**
Description	Write/read access is normally performed synchronously with processing. In individual cases, this implicit synchronisation can be suppressed (see Access synchronisation by NC channel [▶ 37]). If the variable is a variable structure, all subordinated structure elements inherit the synchronisation type. In addition, the synchronisation type can be individually defined for each structure element in the type definition (P-EXTV-00018 [▶ 20]).
Parameter	var[i].synchronisation
Data type	BOOLEAN
Data range	TRUE, FALSE
Dimension	----
Default value	TRUE
Remarks	Synchronous accesses always cause the decoder to stop during a read operation until the synchronous read value is available to the decoder. In addition, it is not permitted to read synchronous variables, e.g. with functions such as an enabled Tool radius compensation; the message is output with ID 20651.

3.4.6.6 Access right (P-EXTV-00006)

P-EXTV-00006	**Access right of external variable**
Description	Write/read access to variables is possible in the basic setting and can be restricted by access protection. If the variable is a variable structure, the access right is inherited to all subordinated structure elements. In addition, for type definition, the access right can be set individually for each structure element (P-EXTV-00019 [▶ 21]).
Parameter	var[i].access_rights
Data type	STRING
Data range	READ_WRITE, READ_ONLY, WRITE_ONLY
Dimension	----
Default value	READ_WRITE
Remarks	The access right only applies to the CNC, not to the PLC. A variable defined by the access right READ_ONLY can only be read in the NC program. This variable can be written in the PLC.

3.4.6.7 Array size (P-EXTV-00007)

P-EXTV-00007	Number of elements in an array of structure elements
Description	If an external variable is required not only once but as an array of this variable, the number of array elements must be defined. If the variable is not an array, then specify 0.
Parameter	var[i].array_elements
Data type	UNS16
Data range	1 ... MAX(UNS16)
Dimension	----
Default value	0
Remarks	

3.4.6.8 Variable size (P-EXTV-00008)

P-EXTV-00008	Size of the external variable
Description	Deviates from the data size implicitly defined by "type" (P-EXTV-00003 [▶ 23]) or the defined data size of a structure, see Data sets for definition of the elements of a variable type (struct[i].element[j].*) [▶ 20]) and offset and thus the **start position of the next** global variable in the memory can be “moved”. Specifying the variable size in only necessary if additional alignment bytes must be considered. With arrays the parameter indicates the size of a single element. This parameter defines the size of string variables (including zero termination byte) for variables of the VSTRING type. A different size can be selected for each variable. The default size of the variable is 128 bytes.
Parameter	var[i].size
Data type	UNS32
Data range	0 ... MAX(UNS32)
Dimension	----
Default value	0
Remarks	If this parameter is set with the value 0, its value is derived from the data type set in P-EXTV-00003 [▶ 23]. **Important**: If the value is smaller than the actually required memory, the variable is overwritten by the next variable.

3.4.6.9 Enabling of HMI access (P-EXTV-00009)

P-EXTV-00009	Enabling the access by HMI
Description	If the flag is set to TRUE, access is also permitted over the user interface to each variable by a corresponding communication object.
Parameter	var[i].create_hmi_interface
Data type	BOOLEAN
Data range	TRUE, FALSE
Dimension	----
Default value	FALSE
Remarks	This parameter has no effect on variables with a user-defined data type.

3.4.6.10 Variable export to PLC-description (P-EXTV-00047)

P-EXTV-00047	Support variable export to PLC description

Description	The NC command #EXPORT VE (see [FCT-C22]) can be used to export the CNC description of the external variables to an equivalent PLC description. If this parameter is set to TRUE, the export of the variable is suppressed. Alignment bytes are then added to the PLC description.
Parameter	var[i].suppress_export
Data type	BOOLEAN
Data range	TRUE, FALSE
Dimension	----
Default value	FALSE
Remarks	This parameter is available as of CNC Builds 2.11.2027.01, V.2.11.2807.18 or V3.1.3052.01 and higher.

3.4.7 Number of configured external variables (P-EXTV-00010)

P-EXTV-00010	**Number of configured external variables**
Description	If *var[i].** is assigned without gaps, enter the index of the variable last defined +1 here (i_{last}+1). If *var[i].** is assigned with gaps, enter the highest index of the defined variable + 1 here (i_{max}+1). If the value is smaller that the number of variables actually configured, only the variables up to this value are available after controller start-up.
Parameter	number_used_variables
Data type	UNS16
Data range	0 ... MAX(UNS16)
Dimension	----
Default value	0
Remarks	*anzahl_belegt (old syntax up to V2.11.2034.0)*

3.4.8 Plausibility check of the memory layout (P-EXTV-00011)

P-EXTV-00011	**Plausibility check of the memory layout**
Description	The parameter var[i].byte_offset P-EXTV-00002 [▶ 23] can set external variables to any, possibly incorrect, memory address. This parameter can activate a plausibility check of the memory layout for external variables. If variables overlap in the memory, the CNC outputs the error message P-ERR-21848 at controller start-up and the overlapping variable is deleted.
Parameter	check_overlapping_variables
Data type	BOOLEAN
Data range	TRUE, FALSE
Dimension	----
Default value	FALSE
Remarks	This parameter is available as of CNC Builds 2.11.2027.01, V.2.11.2807.18 or V3.1.3052.01 and higher.

3.4.9 Method for automatic memory layout (P-EXTV-00012)

P-EXTV-00012	**Method for automatic memory layout**

Description	As of CNC Build V2.10.1025.00 and higher, the CNC can automatically create the external variables in the memory without gaps. External variables can also be assigned to any memory address by using the parameter var[i].byte_offset P-EXTV-00002 [▶ 23] or by selecting a 24-byte are via var[i].index P-EXTV-00038. If the automatic and manual address selection is combined, this parameters defines how the CNC generates the automatic addresses for the variables.
Parameter	auto_memory_mode
Data type	STRING
Data range	START_VE_MEMORY: All variables with automatic determined addresses are set to the start of the external memory area without gaps. LAST_USED_ADDRESS: The external variable with automatic address assignment is always created after the last memory area is occupied by the preceding variables.
Dimension	----
Default value	START_VE_MEMORY
Remarks	This parameter is available as of CNC Builds 2.11.2027.01, V.2.11.2807.18 or V3.1.3052.01 and higher. Older CNC builds always use the START_VE_MEMORY setting.

Example for auto_memory_mode = START_VE_MEMORY:

```
auto_memory_mode START_VE_MEMORY

var[0].name                     var_1
var[0].type                     SGN32
var[0].scope                    GLOBAL
var[0].synchronisation          FALSE
var[0].access_rights            READ_WRITE
#
var[1].name                     var_2
var[1].type                     REAL64
var[1].scope                    GLOBAL
var[1].synchronisation          TRUE
var[1].access_rights            READ_WRITE
var[1].byte_offset              16
#
var[2].name                     var_3
var[2].type                     SGN32
var[2].scope                    GLOBAL
var[2].synchronisation          TRUE
var[2].access_rights            READ_WRITE
```

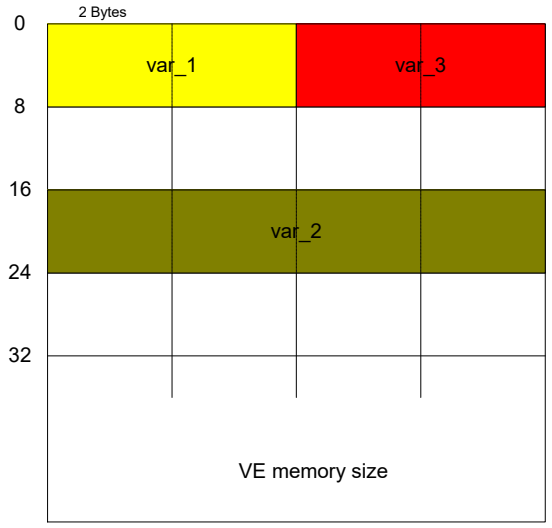


Fig. 4: Resulting memory layout:

Example for auto_memory_mode = LAST_USED_ADDRESS:

```
auto_memory_mode LAST_USED_ADDRESS

var[0].name                     var_1
var[0].type                     SGN32
var[0].scope                    GLOBAL
var[0].synchronisation          FALSE
var[0].access_rights            READ_WRITE
#
var[1].name                     var_2
var[1].type                     REAL64
var[1].scope                    GLOBAL
var[1].synchronisation          TRUE
var[1].access_rights            READ_WRITE
var[1].byte_offset              16
#
var[2].name                     var_3
var[2].type                     SGN32
var[2].scope                    GLOBAL
var[2].synchronisation          TRUE
var[2].access_rights            READ_WRITE
```

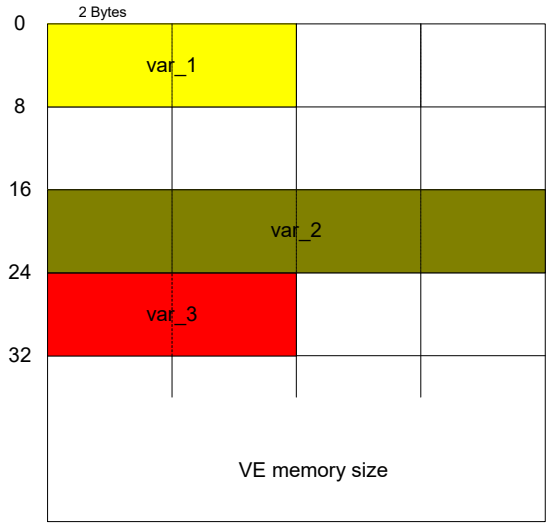


Fig. 5: Resulting memory layout:

3.4.10 Initialisation at CNC start (P-EXTV-00013)

P-EXTV-00013	Initialisation of external variables with default values
Description	This parameter can assign initial values to external variables at controller start-up. The controller only evaluates this parameter at start-up after the memory layout of the external variables is defined. Behind the keyword init, the CNC expects a syntax string conforming to an NC program with value assignment to an external variable (see [PROG//13-Variables and calculation of variables]). The keyword init may be used several times in the configuration file of the external variables. The initial instructions are processed in the sequence in which they occur in the configuration file. In the value assignment it is possible to use arithmetic operations, string operations and other external variables. All other decoder variables and instructions are not permissible.
Parameter	init
Data type	STRING
Data range	Syntax string conforming to the NC program with value assignment to an external variable.
Dimension	----
Default value	-

Remarks	This parameter is available as of CNC Builds 2.11.2027.01, V.2.11.2807.18 or V3.1.3052.01 and higher.

NOTICE

The initialisation values of the external variables P-EXTV-00013 are only evaluated at controller start-up. The values are not adopted if an external variable list is reloaded.

Example of the initialisation of ext. Variables at controller start-up:

```
init      V.E.var1_real64 = 1234.5
init      V.E.var1_sgn32 = ROUND[1 + 10 / 3]
init      V.E.var1_string = "Hello" + " world!"
init      V.E.var2_real64 = 2.0 * V.E.var1_real64

init      V.E.arr_sgn32[0] = 1
init      V.E.arr_sgn32[1] = 2

init      V.E.vector.x = 10.0
init      V.E.vector.y = 20.0
init      V.E.vector.z = 30.0
```

3.4.11 Example of a configuration list

Examples for type definitions:

```
use_extended_string_var                 TRUE #Strings with 128 characters

struct[0].name                          VECTOR_T
struct[0].element[0].name               x
struct[0].element[0].type               REAL64
struct[0].element[1].name               y
struct[0].element[1].type               REAL64
struct[0].element[2].name               z
struct[0].element[2].type               REAL64
#
struct[1].name                          TARGET_POINT_T
struct[1].element[0].name               point
struct[1].element[0].type               VECTOR_T
struct[1].element[1].name               valid
struct[1].element[1].type               BOOLEAN
struct[1].element[1].access_rights      READ_ONLY
struct[1].element[1].synchronisation    TRUE
#
struct[2].name                          TRAJEKTORIE_T
struct[2].element[0].name               nbr_points
struct[2].element[0].type               SGN32
struct[2].element[1].name               name
struct[2].element[1].type               STRING
struct[2].element[2].name               points
struct[2].element[2].type               TARGET_POINT_T
struct[2].element[2].array_elements     10
```

Examples for variable definitions:

```
number_used_variables                   5
#
var[0].name                             var_global_1
var[0].type                             UNS32
var[0].scope                            GLOBAL
var[0].synchronisation                  FALSE
var[0].access_rights                    READ_WRITE
#
var[1].name                             var_chan_1
var[1].type                             SGN32
var[1].scope                            CHANNEL
var[1].synchronisation                  TRUE
var[1].access_rights                    READ_WRITE
#
var[2].name                             array_chan_1
var[2].type                             SGN16
var[2].scope                            CHANNEL
var[2].synchronisation                  TRUE
var[2].access_rights                    READ_WRITE
var[2].array_elements                   20
#
var[3].name                             var_chan_2
var[3].type                             STRING
var[3].scope                            CHANNEL
var[3].synchronisation                  TRUE
var[3].access_rights                    READ_WRITE
#
var[4].name trajectory
var[4].type                             TRAJEKTORIE_T
var[4].scope                            CHANNEL
var[4].synchronisation                  FALSE
var[4].access_rights                    READ_WRITE
```

3.4.12 Example of V.E. structures

This example explains by means of a V.E. List how to handle V.E. structures.

Task:

A position curve is given a name and contains a defined number of positions.

Each of these positions consists of X, Y, Z and a validity flag.

5 different curves are possible and each of these curves has a maximum of 12 points

Curve structure:

- Position
- Name

Position structure

- X
- Y
- Z
- Validity flag

```
#*********************************************
# TC_CHANNEL_DESC_5: External variables
#*********************************************
use_extended_string_var                        1
# ----Definition of structures -------
# -------Position curve structure -------
#
struct[0].name                                  typcurve
struct[0].element[0].name                       point
struct[0].element[0].type                       typ_pos
struct[0].element[0].array_elements             12
struct[0].element[1].name                       curve_name
struct[0].element[1].type                       STRING

#------- Spatial position structure -------
struct[1].name                                  typ_pos
struct[1].element[0].name                       X
struct[1].element[0].type                       REAL64
struct[1].element[1].name                       Y
struct[1].element[1].type                       REAL64
struct[1].element[2].name                       Z
struct[1].element[2].type                       REAL64
struct[1].element[3].name                       pos_is_valid
struct[1].element[3].type                       BOOLEAN
#
#------- Variables -------
number_used_variables                          1
#
var[0].name                                    curve
var[0].type                                    typcurve
var[0].scope                                   GLOBAL
var[0].synchronisation                         FALSE
var[0].access_rights                           READ_WRITE
var[0].array_size                              5
#
End
```

The entry of structures and variables is case-sensitive.

If the type with string is specified for the data type instead of STRING, the error P-ERR-21441 is output.

The applicable data types are listed in the parameter P-EXTV-00003 [▶ 23].

On the structures used are checked for correct syntax.

The check is executed at controller start-up.

(assignment of a point from the NC program

%Setpoint.nc

N020 V.E.curve[0].point[2].X=11

N030 V.E.curve[0].point[2].Y=22

N040 V.E.curve[0].point[2].Z=33

N080 M30

3.5 Integration in NC boot-up

The configuration of the external variables is declared to each NC-channel separately by means of an ASCII list. For each channel, a file name is specified in the system's central boot description, and this file name defines the configuration of the external variables of this channel.

Example: Extract from the start-up description for two channels

```
// --------------------------------------
# Configuration data lists
// --------------------------------------
#
Listen                          ASCII
default_sda_mds                 ..\listen\default_sda.lis
hand_mds                        ..\listen\hand_mds.lis
rtconf_lis                      ..\listen\rtconf.lis
#
sda_mds[0]                      ..\listen\sda_mds1.lis
werkz_data[0]                   ..\listen\werkz_d1.lis
nullp_data[0]                   ..\listen\nullp_d1.lis
pzv_data[0]                     ..\listen\pzv_d1.lis
ve_var[0]                       ..\lists\ext_var1.lis
hmi[0].objects                  default
channel[0].objects              default
#
sda_mds[1]                      ..\listen\sda_mds2.lis
werkz_data[1]                   ..\listen\werkz_d2.lis
nullp_data[1]                   ..\listen\nullp_d2.lis
pzv_data[1]                     ..\listen\pzv_d2.lis
ve_var[1]                       ..\lists\ext_var2.lis
hmi[1].objects                  default
channel[1].objects              default
```

4 Application and access to variables

4.1 NC program

The NC channel accesses the external variables by using the write/read instruction to the variable **V.E.<name>** in the NC program. The variables available in the NC program consist of the prefix **V.E.** and the name specified in the variable configuration list **<name>** . **V.E.** Variables may only consist of 30 characters.

VE variable access in the CNC

```
N100 $IF V.E.CHANNEL_WR >= 100     (corresp. value of V.E.CHANNEL_WR)
                                   (branched to)
                                   (various cases)
N110 G01 X100 Y100 F1000

N120 $ELSE
N130 G01 X100 YV.E.CHANNEL_WR F1000   (Linear interpolation in)
                                      (Y direction with the value)
                                      (of CHANNEL_WR)
N140 $ENDIF

N150 V.E.GLOBAL_SWR = V.A.ABS.X    (The external variable is assigned)
                                   (absolute X coordinates)
N160 G01 XV.E.GLOBAL_SWR           (Straight interpolation in X direction)
                                   (with the value of V.E.GLOBAL_SWR)
```

VE variable access in the CNC as of Version V2.10.1025.00 and higher

```
N010 $IF V.E.trajektorie.name != „"
N020   V.E.name = V.E.trajektorie.name
N030   P1 = 0
N040   $WHILE P1 < V.E.trajektorie.nbr_points
N050     $IF V.E.trajektorie.points[P1].valid == TRUE
N060       G0 X = V.E.trajektorie.points[P1].point.x
              Y = V.E.trajektorie.points[P1].point.y
              Z = V.E.trajektorie.points[P1].point.z
N070     $ENDIF
N080     P1 += 1
N090   $ENDWHILE
N100 $ENDIF
N110 V.E.name = „"
N120 M30
```

4.1.1 Access synchronisation by NC channel

With a synchronous read or write variable access, the user may expect a temporal sequence as specified in the NC program. This behaviour can be determined by the synchronisation type of the variables.

An asynchronous read/write access (see figure, case b)) is executed directly when the NC program is decoded.

Since NC program decoding is performed in a planning phase before the actual execution of the NC commands, synchronous variable access may not simply be performed during program decoding. It must be synchronised with current program execution (axis movement). Synchronous access is ensured as follows:

Synchronous READ (see figure, case a))**:** When reading, decoding is stopped until program processing reaches the last decoded NC program line (implicit # FLUSH WAIT [PROG// Flushing NC channel]). The value is then read and made available to the decoder. Only then is decoding continued. Since # FLUSH WAIT is not allowed during certain cross-block NC functionalities (e.g. active spline interpolation, active tool radius compensation), synchronous READ during this functionality is not possible either. In such cases, the error ID 20651 is output.

Synchronous WRITE (see figure, case a))**:** Write access is scheduled in the same way as any other NC command during decoding and is only performed later during NC machining.

Since this synchronisation has undesirable run-time effects, in particular stopping the decoding process during a Read access, this implicit synchronisation can be deactivated when defining variables. Of course, this is only possible if the moment of read access need not be synchronous with processing or this is ensured anyway by an explicit synchronisation point in the NC program (e.g. explicit # FLUSH WAIT).

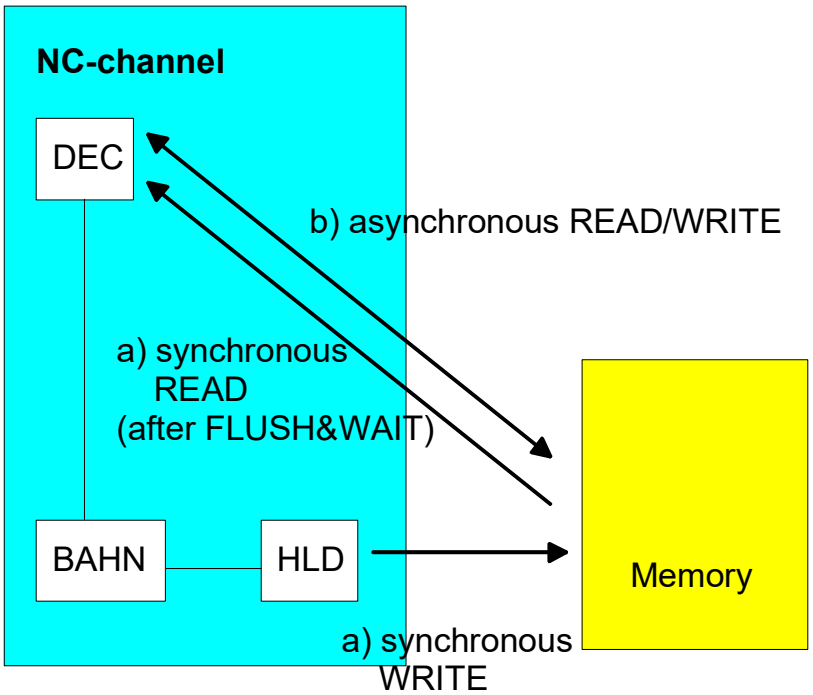


Fig. 6: Asynchronous/synchronous access of decoding (DEC) and processing (BAHN) via PLC driver (HLD)

NOTICE

The output of synchronised V.E. variables takes place in the controller with the MOS synchronisation type over the same interface as the output of technology functions (M/H/T functions). The user must therefore ensure that all data in the queue of this interface are read out by the PLC, otherwise synchronicity for output to the NC channel is not guaranteed.

4.2 Graphical user interface (P-EXTV-00030 - P-EXTV-00037)

The graphical user interface has access to NC interfaces and NC data via so-called HMI objects. The access protocol is encapsulated in a DLL which offers a Windows application write/read access to HMI objects. Interface objects are created automatically according to the configuration list for the graphical user interface if this is configured accordingly for the variable.

Two objects are then created for each specified variable, one for write access and one for read access. This occurs independently of the access rights of the NC channel, i.e. even if the NC channel is only assigned read access to the variable (e.g. access_rights = READ_ONLY), the graphical user interface can still write to the variable. With an array, a GUI object is created for each array element.

If required, the name format of GUI objects can be adapted as required by specifying corresponding templates in the list (as of Version V254). In this case, the placeholder %s must be specified for the name followed by a placeholder %d for the array index.

P-EXTV-00030	**HMI read access with global array**
Description	Format of the HMI object name by specifying a corresponding template.
Parameter	name_rd_global_array
Default value	**cnc_ve_**%s**_rd[**%d**]**
Remarks	Placeholder %s for the name and followed by %d for the array index

P-EXTV-00031	**HMI write access with global array**
Description	Format of the HMI object name by specifying a corresponding template.
Parameter	name_wr_global_array
Default value	**cnc_ve_**%s**_wr[**%d**]**
Remarks	Placeholder %s for the name and followed by %d for the array index

P-EXTV-00032	**HMI read access with channel-specific array**
Description	Format of the HMI object name by specifying a corresponding template.
Parameter	name_rd_channel_array
Default value	**mc_ve_**%s**_rd[**%d**]**
Remarks	Placeholder %s for the name and followed by %d for the array index

P-EXTV-00033	**HMI read access with channel-specific array**
Description	Format of the HMI object name by specifying a corresponding template.
Parameter	name_wr_channel_array
Default value	**mc_ve_**%s**_wr[**%d**]**
Remarks	Placeholder %s for the name and followed by %d for the array index

P-EXTV-00034	**HMI read access with global variables**
Description	Format of the HMI object name by specifying a corresponding template.
Parameter	name_rd_global
Default value	**cnc_ve_**%s**_rd**
Remarks	Placeholder %s for the name

P-EXTV-00035	**HMI write access with global variables**
Description	Format of the HMI object name by specifying a corresponding template.
Parameter	name_wr_global
Default value	**cnc_ve_**%s**_wr**
Remarks	Placeholder %s for the name

P-EXTV-00036	**HMI read access with channel-specific variables**
Description	Format of the HMI object name by specifying a corresponding template.
Parameter	name_rd_channel
Default value	**mc_ve_**%s**_rd**
Remarks	Placeholder %s for the name

P-EXTV-00037	**HMI write access with channel-specific variables**
Description	Format of the HMI object name by specifying a corresponding template.
Parameter	name_wr_channel
Default value	**mc_ve_**%s**_wr**
Remarks	Placeholder %s for the name

Example: Assigning the name to the GUI

```
# ************************************
#
# ************************************
#
name_rd_global_array              cnc_test1_%s_rd[%d]
name_wr_global_array              cnc_test1_%s_wr[%d]
name_rd_channel                   mc_test2_%s_rd
name_wr_channel                   mc_test2_%s_wr

...

var[0].name                       G_ARRAY5
var[0].type                       SGN32
var[0].scope                      GLOBAL
var[0].synchronisation            FALSE
var[0].access_rights              READ_WRITE
var[0].array_size                 5
var[0].create_hmi_interface TRUE # HMI object is created
#

var[1].name                       L_BOOLEAN
var[1].type                       BOOLEAN
var[1].scope                      CHANNEL
var[1].synchronisation            FALSE
var[1].access_rights              READ_WRITE
var[1].array_size                 1
var[1].create_hmi_interface TRUE # HMI object is created
```

The following HMI objects are created from the above extract of the configuration list:

```
cnc_test1_G_ARRAY5_rd[0]
cnc_test1_G_ARRAY5_wr[0]
cnc_test1_G_ARRAY5_rd[1]
cnc_test1_G_ARRAY5_wr[1]
cnc_test1_G_ARRAY5_rd[2]
cnc_test1_G_ARRAY5_wr[2]
cnc_test1_G_ARRAY5_rd[3]
cnc_test1_G_ARRAY5_wr[3]
cnc_test1_G_ARRAY5_rd[4]
cnc_test1_G_ARRAY5_wr[4]

mc_test2_L_BOOLEAN_wr
mc_test2_L_BOOLEAN_rd
```

4.3 PLC

After controller start-up, the PLC run-time system receives access to the shared memory areas for V.E. variables between CNC and PLC A distinction is made between in channel and global memory areas.

gpVE[iChannelIndex]r

gpVEGlob

In order to simplify the simulation of the memory area for the PLC, it is recommended to export the variable after it is created using the # EXPORT VE command .
For more information see [FCT-C22// Description]

Manual simulation of the variable structure in the PLC

This PLC example below shows how V.E. variables can be used in the PLC by simulating variable structures.

Access to V.E. variables in the PLC

Definition of variable structures for V.E. variables:

```
TYPE VECTOR_T :
STRUCT
  x : LREAL;
  y : LREAL;
  z : LREAL;
END_STRUCT
END_TYPE

TYPE TARGET_POINT_T :
STRUCT
  point : VECTOR_T;
  valid : BOOL;
END_STRUCT
END_TYPE

TYPE TRAJEKTORIE_T :
STRUCT
  nbr_points : DINT;
  name : STRING(127);
  points : ARRAY [0..9] OF TARGET_POINT_T;
END_STRUCT
END_TYPE
```

Definition of structures for the complete assigned V.E. memory area:

```
TYPE VE_GLOBAL:
STRUCT
  var_global_1 : DINT;
END_STRUCT
END_TYPE

TYPE VE_CHAN_1:
STRUCT
  var_chan_1 : DINT;
  array_chan_1: ARRAY [0..19] OF INT;
  name : STRING(127);
  Trajectory: TRAJEKTORIE_T;
END_STRUCT
END_TYPE
```

PLC program to access V.E. variables of channel 1:

```
PROGAM V_E
  VAR
    p_ve_chan_1 : POINTER TO VE_CHAN_1;
  END_VAR

(* Channel 1 with index 0 - gpVECh[0]*)
p_ve_chan_1 := ADR (gpVECh[0]^.ext_var32[0]);

IF (p_ve_chan_1^.name = ,')
THEN
  p_ve_chan_1^.trajektorie.name := 'My Path!';
  p_ve_chan_1^.trajektorie.nbr_points := 2;
```

```
  p_ve_chan_1^.trajektorie.points[0].valid := TRUE;
  p_ve_chan_1^.trajektorie.points[0].point.x := 100.0;
  p_ve_chan_1^.trajektorie.points[0].point.y := 200.0;
  p_ve_chan_1^.trajektorie.points[0].point.z := 300.0;

  p_ve_chan_1^.trajektorie.points[1].valid := TRUE;
  p_ve_chan_1^.trajektorie.points[1].point.x := 200.0;
  p_ve_chan_1^.trajektorie.points[1].point.y := 400.0;
  p_ve_chan_1^.trajektorie.points[1].point.z := 600.0;
END_IF;
```

5 Appendix

5.1 Configuration syntax up to V2.10.1025

Example of an ASCII list:

```
# ************************************
# External variables V254
# ************************************
#
number_used_variables            2
#
var[0].name                      GLOBAL_SWR (Global VE)
var[0].index                     0
var[0].type                      SGN16
var[0].scope                     GLOBAL
var[0].synchronisation           FALSE
var[0].access_rights             READ_ONLY
var[0].array_size                0
var[0].size                      2 # 2 bytes per element
var[0].create_hmi_interface      FALSE
#
var[1].name                      CHANNEL_WR (channel VE)
var[1].index                     0
var[1].type                      SGN32
var[1].scope                     CHANNEL
var[1].synchronisation           FALSE
var[1].access_rights             READ_ONLY
var[1].array_size                10
var[1].size                      4 # 4 bytes per element
var[1].create_hmi_interface      FALSE
#
End
```

Identifier	Value range	Default	Meaning
number_used_variables *(Old syntax up to V2.11.2034.0: anzahl_belegt)*	[0; MAX_UNS16]	0	Assuming gapless assignment in *var[i].** enter the index of the last variable defined +1 here (i_{last}+1). Assuming assignment with gaps in *var[i].** enter the highest index of the defined variable + 1 here (i_{max}+1).
*var[i].**	i:= [0; 214]		Data records to define variables.
name	ASCII string		See P-EXTV-00001
index	[0; MAX_SGN32]	-1	The index defines the location in the memory where the variable is saved. The entire memory is structured as an array of units of 24 bytes each. An index of –1 indicates that the entry is not used.
type	[BOOLEAN, SGN08, UNS08, SGN16, UNS16, SGN32; UNS32, REAL64, STRING]	UNS32	See P-EXTV-00003
scope	[GLOBAL; CHANNEL]	CHANNEL	See P-EXTV-00004
synchronisation	[TRUE, FALSE]	TRUE	See P-EXTV-00005

access_rights	[READ_WRITE, READ_ONLY, WRITE_ONLY]	READ_WRITE	See P-EXTV-00006
array_size	[0; MAX_UNS16]	0	If an external variable is required not only once but as an array of this variable, the number of elements must be defined. If the variable is not an array, then specify 0.
size	[0; MAX_UNS32]	4	See P-EXTV-00008
create_hmi_interface	[TRUE, FALSE]	FALSE	See P-EXTV-00009

5.1.1 Memory layout up to V2.10.1025

The configuration defines the view of the channel onto the memory and, thus, its logical structuring. The entire memory is structured as an array of 24 bytes blocks (union of the types of all contents). When defining each variable, the start position is specified by specifying the block index in this array.

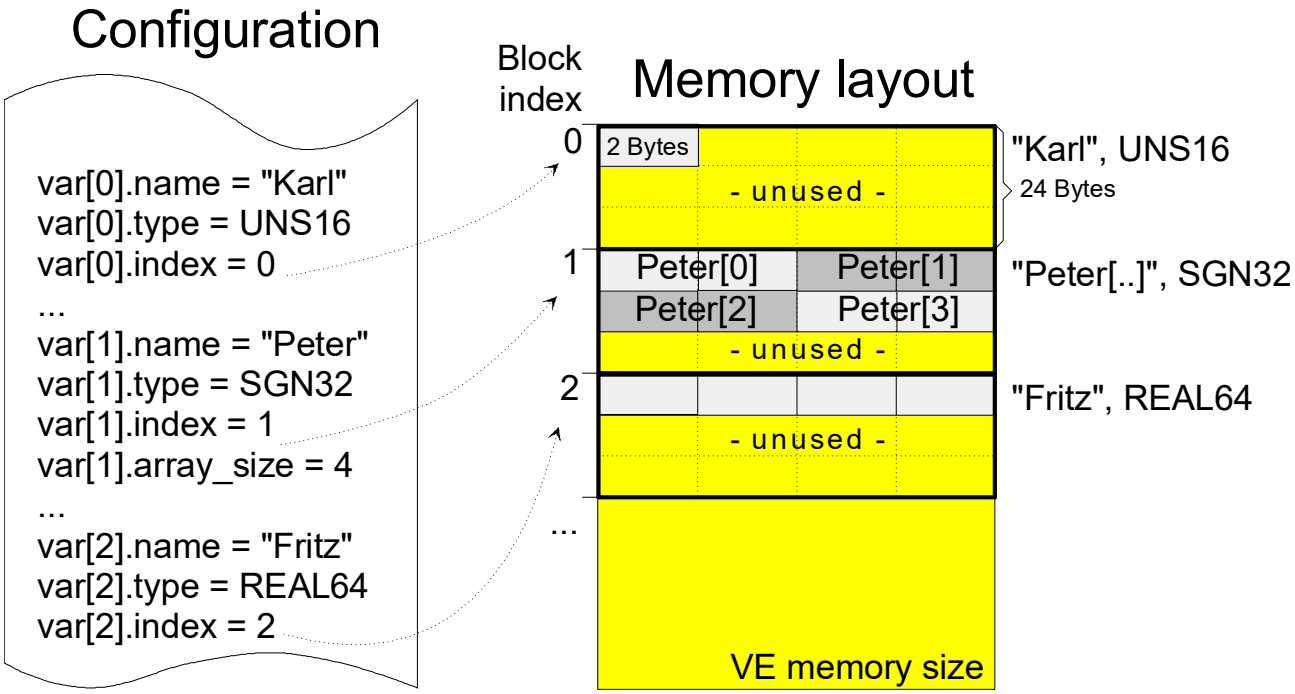


Fig. 7: Memory layout resulting from the given configuration

A VE variable (also VE array) is saved contiguously from the specified start position. If it is larger than a 24 byte grid, the following memory area (grid) must also be used. Basically, it is also possible to place several logical variables at the identical memory location, i.e. it is possible to enable several views onto a memory location. The NC does not monitor whether the individual variables overlap.

If individual variables are placed in the memory, an unused memory area is left over for each memory grid depending on the size of the variable. This unused area is no more addressable.

5.1.2 Memory block index (P-EXTV-00038)

P-EXTV-00038	**Memory block index of the external variable in the memory.**
Description	The index defines the location in the memory where the variable is saved. The entire memory is structured as an array of units of 24 bytes each. An index of -1 indicates that the entry is not used. See Memory layout up to V2.10.1025 [▶ 45]
Parameter	var[i].index
Data type	SGN32
Data range	0 ... MAX(SGN32)
Dimension	----
Default value	-1
Remarks	

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

P

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