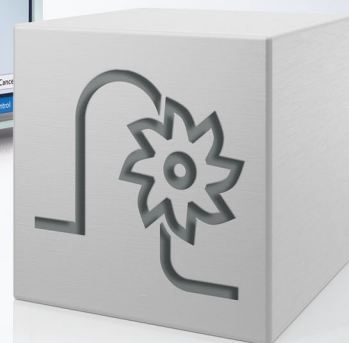
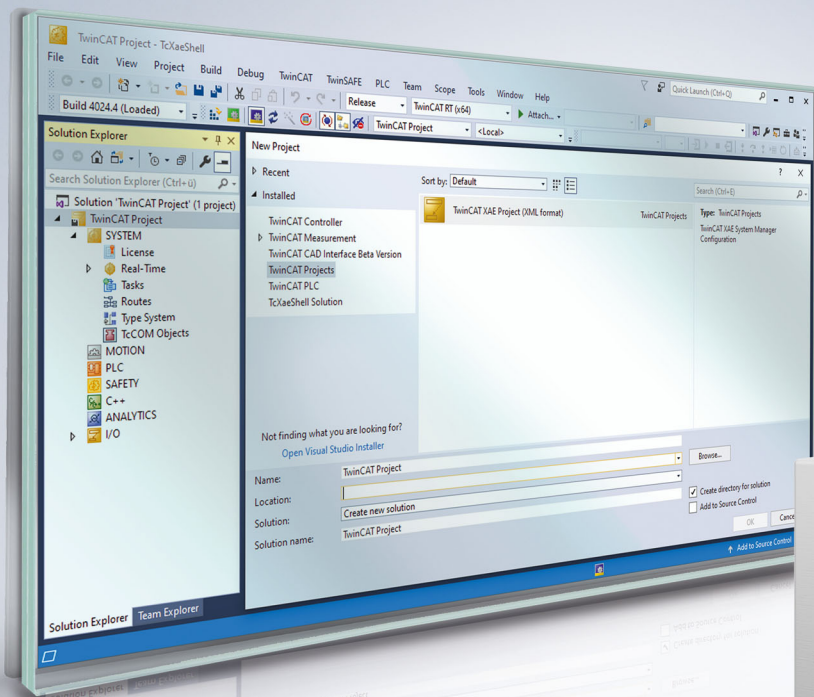


# BECKHOFF New Automation Technology

Manual | EN

## TF5270/71 | TwinCAT 3 CNC

Virtual NCK





## 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

|   |           |
|---|-----------|
| <b>Notes on the documentation.....</b>              | <b>3</b>  |
| <b>General and safety instructions .....</b>        | <b>4</b>  |
| <b>1 Simulation DLL for CNC kernel.....</b>         | <b>12</b> |
| <b>2 Description .....</b>                          | <b>13</b> |
| 2.1 DLL version .....                               | 13        |
| 2.2 Components .....                                | 13        |
| 2.3 Dependencies .....                              | 14        |
| 2.3.1 kernelv as of V300 .....                      | 14        |
| 2.4 Multiple instances.....                         | 14        |
| 2.4.1 Diagnostics with multiple instances.....      | 14        |
| 2.5 License protection .....                        | 15        |
| 2.5.1 Licensing under TwinCAT 2 .....               | 15        |
| 2.5.2 Licensing under TwinCAT 3 .....               | 15        |
| 2.6 General .....                                   | 17        |
| 2.6.1 Identification of axes and channels .....     | 17        |
| 2.6.2 Coordinate systems .....                      | 17        |
| 2.7 Use of kernelv .....                            | 18        |
| 2.7.1 Start.....                                    | 18        |
| 2.7.2 Cyclic operation.....                         | 18        |
| 2.7.3 End/restart.....                              | 18        |
| 2.7.4 Operation modes of kernelv .....              | 19        |
| 2.7.5 Acknowledgement of technology functions ..... | 20        |
| 2.7.6 Search path for NXC programs .....            | 20        |
| 2.7.7 kernelv demo application .....                | 20        |
| 2.8 Configuration.....                              | 24        |
| 2.8.1 Path of the configuration lists .....         | 24        |
| 2.9 Error message texts .....                       | 24        |
| 2.10 Tool management .....                          | 25        |
| <b>3 kernelv API functions .....</b>                | <b>26</b> |
| 3.1 kernelv_get_api_version().....                  | 26        |
| 3.2 kernelv_get_cnc_version() .....                 | 27        |
| 3.3 kernelv_get_cycletime() .....                   | 28        |
| 3.4 kernelv_startup() .....                         | 29        |
| 3.5 kernelv_startup_instance().....                 | 30        |
| 3.6 kernelv_do_cycle().....                         | 31        |
| 3.7 kernelv_ch_program_start() .....                | 32        |
| 3.8 kernelv_ch_reset() .....                        | 33        |
| 3.9 kernelv_ch_suspend().....                       | 34        |
| 3.10 kernelv_ch_resume() .....                      | 35        |
| 3.11 kernelv_ch_get_override() .....                | 36        |
| 3.12 kernelv_ch_set_override().....                 | 37        |
| 3.13 kernelv_ch_get_blocknumber().....              | 38        |
| 3.14 kernelv_ch_get_filename().....                 | 39        |

|      |  |    |
|------|--|----|
| 3.15 | kernelv_ch_get_programname()                   | 40 |
| 3.16 | kernelv_ch_get_state()                         | 41 |
| 3.17 | kernelv_ch_get_fileoffset()                    | 42 |
| 3.18 | kernelv_ch_get techno_data()                   | 43 |
| 3.19 | kernelv_ch_get_new techno_data()               | 44 |
| 3.20 | kernelv_ch_get techno_data2()                  | 45 |
| 3.21 | kernelv_ch_get_new techno_data2()              | 46 |
| 3.22 | kernelv_ch_get_finished_nc_lines()             | 47 |
| 3.23 | kernelv_ax_get techno_data()                   | 48 |
| 3.24 | kernelv_ax_get_new techno_data()               | 49 |
| 3.25 | kernelv_ax_get techno_data2()                  | 50 |
| 3.26 | kernelv_ax_get_new techno_data2()              | 51 |
| 3.27 | kernelv_ax_set_position()                      | 52 |
| 3.28 | kernelv_get_acs_command_positions()            | 53 |
| 3.29 | kernelv_get_acs0_command_positions()           | 54 |
| 3.30 | kernelv_get_acs_actual_positions()             | 55 |
| 3.31 | kernelv_get_acs_target_positions()             | 56 |
| 3.32 | kernelv_get_acs_start_positions()              | 57 |
| 3.33 | kernelv_get_wcs_command_positions()            | 58 |
| 3.34 | kernelv_get_wcs_target_positions()             | 59 |
| 3.35 | kernelv_get_wcs_start_positions()              | 60 |
| 3.36 | kernelv_get_prg_target_positions()             | 61 |
| 3.37 | kernelv_get_axis_channel_number()              | 62 |
| 3.38 | kernelv_ch_get_variable_value()                | 63 |
| 3.39 | kernelv_ch_set_variable_value()                | 65 |
| 3.40 | kernelv_get_channel_count()                    | 67 |
| 3.41 | kernelv_get_axis_count()                       | 68 |
| 3.42 | kernelv_sync_read_request()                    | 69 |
| 3.43 | kernelv_sync_write_request()                   | 70 |
| 3.44 | kernelv_sync_read_write_req()                  | 71 |
| 3.45 | kernelv_get_axis_names()                       | 72 |
| 3.46 | kernelv_control techno_func_duration()         | 74 |
| 3.47 | kernelv_ch_set techno_func_duration()          | 75 |
| 3.48 | kernelv_ch_set techno_func_user_ackn()         | 76 |
| 3.49 | kernelv_ch_ackn techno_func()                  | 77 |
| 3.50 | kernelv_ax_ackn techno_func()                  | 78 |
| 3.51 | kernelv_get_license_info()                     | 79 |
| 3.52 | kernelv_set_options()                          | 80 |
| 3.53 | kernelv_ch_get_decoder_positions()             | 81 |
| 3.54 | kernelv_ch_get_prog_start_mode()               | 82 |
| 3.55 | kernelv_ch_set_cont visu_grid()                | 83 |
| 3.56 | kernelv_ch_set_cont visu_rel curvature_error() | 84 |
| 3.57 | kernelv_ch_set_cont visu_abs curvature_error() | 85 |
| 3.58 | kernelv_ch_get_cont visu_data()                | 86 |
| 3.59 | kernelv_ch_get_active_g_codes()                | 87 |
| 3.60 | kernelv_get_active_g_group()                   | 88 |

|          |   |            |
|----------|---|------------|
| 3.61     | kernelv_ch_get_command_feed().....                                      | 89         |
| 3.62     | kernelv_ch_get_active_feed().....                                       | 90         |
| 3.63     | kernelv_set_call_ratio().....   | 91         |
| 3.64     | CNC error messages with kernelv .....                                   | 92         |
| 3.64.1   | Read out error message in the form of a string kernelv_get_error()..... | 92         |
| 3.64.2   | General information on error messages.....                              | 93         |
| 3.64.3   | Error messages caused by NC programs .....                              | 102        |
| 3.65     | Coordinate systems and offsets .....                                    | 107        |
| 3.65.1   | kernelv_ch_get_cs_name().....   | 108        |
| 3.65.2   | kernelv_ch_get_cs_rot_matrix().....                                     | 109        |
| 3.65.3   | kernelv_ch_get_cs_shift_vector() .....                                  | 110        |
| 3.65.4   | kernelv_ch_get_cs_count().....  | 111        |
| 3.65.5   | kernelv_ch_get_active_cs_index().....                                   | 111        |
| 3.65.6   | kernelv_ch_axis_get_offsets() .....                                     | 112        |
| 3.65.7   | kernelv_ch_get_total_cs_rot_matrix().....                               | 113        |
| 3.65.8   | kernelv_ch_get_total_cs_offset.....                                     | 114        |
| 3.65.9   | kernelv_ch_get_total_cs_def().....                                      | 115        |
| 3.65.10  | kernelv_ch_get_coord_sys_active().....                                  | 116        |
| 3.66     | Kinematic transformations.....  | 116        |
| 3.66.1   | kernelv_ch_get_kin_trafo_active() .....                                 | 116        |
| 3.66.2   | kernelv_ch_get_active_kin_id().....                                     | 117        |
| 3.67     | External measuring hardware .....                                       | 117        |
| 3.67.1   | kernelv_ax_get_ext_latch_command() .....                                | 118        |
| 3.67.2   | kernelv_ax_acknowledge_ext_latch_command() .....                        | 118        |
| 3.67.3   | kernelv_ax_set_ext_latch_event_pos().....                               | 119        |
| 3.67.4   | kernelv_ax_set_ext_latch_event() .....                                  | 119        |
| 3.68     | kernelv_ch_get_timer() .....  | 121        |
| 3.69     | kernelv_get_production_time().....                                      | 121        |
| 3.70     | kernelv_diagnosis_upload() .....  | 122        |
| <b>4</b> | <b>kernelv API types .....</b>  | <b>123</b> |
| 4.1      | Enum KERNELV_RETURN .....   | 123        |
| 4.2      | KERNELV_CHANNEL_STATE.....  | 125        |
| 4.3      | Enum E_KERNELV_TECHNO_TYPE .....  | 127        |
| 4.4      | Struct KERNELV_TECHNO_DATA .....  | 127        |
| 4.5      | KERNELV_CHANNEL_TECHNO_DATA_ARRAY .....                                 | 128        |
| 4.6      | KERNELV_CHANNEL_TECHNO_DATA_ARRAY2 .....                                | 129        |
| 4.7      | KERNELV_AXIS_TECHNO_DATA_ARRAY .....                                    | 130        |
| 4.8      | Struct KERNELV_TECHNO_DATA2 .....                                       | 130        |
| 4.9      | KERNELV_CHANNEL_TECHNO_DATA_ARRAY2 .....                                | 131        |
| 4.10     | KERNELV_AXIS_TECHNO_DATA_ARRAY2 .....                                   | 132        |
| 4.11     | Union U_KERNELV_TECHNO_PARAM .....                                      | 132        |
| 4.12     | Union U_KERNELV_TECHNO_PARAM2 .....                                     | 133        |
| 4.13     | Struct M_H_CODE_DATA .....  | 133        |
| 4.14     | Struct M_H_CODE_DATA2 .....   | 133        |
| 4.15     | Enum E_KERNELV_SPINDLE_TYPE.....  | 133        |
| 4.16     | Struct S_CODE_DATA .....  | 134        |

|         |   |     |
|---------|---|-----|
| 4.17    | Struct T_CODE_DATA.....                     | 135 |
| 4.18    | Struct KERNELV_NC_LINE_DATA.....            | 135 |
| 4.19    | Enum E_KERNELV_VAR_TYPE.....                | 136 |
| 4.20    | Union U_KERNELV_VAR_VALUE.....              | 137 |
| 4.21    | Struct KERNELV_VARIABLE.....                | 138 |
| 4.22    | Struct KERNELV_NC_LINE_DATA.....            | 138 |
| 4.23    | Struct KERNELV_LICENSE_INFO.....            | 139 |
| 4.24    | Struct KERNELV_DECODER_POSITION_HEADER..... | 139 |
| 4.25    | Struct KERNELV_DECODER_POSITION_DATA.....   | 140 |
| 4.26    | Enum E_KERNELV_PROG_START_MODE.....         | 140 |
| 4.27    | Struct ACTIVE_G_CODES.....                  | 140 |
| 4.28    | Enum E_KERNELV_G_GROUP_TYPE.....            | 141 |
| 4.29    | Data types of contour visualisation.....    | 143 |
| 4.29.1  | Struct CONTOUR_VISU.....                    | 143 |
| 4.29.2  | Union CONTOUR_VISU_DATA.....                | 144 |
| 4.29.3  | Struct CONTOUR_VISU_DATA_V0.....            | 144 |
| 4.29.4  | Struct CONTOUR_VISU_DATA_V1.....            | 145 |
| 4.29.5  | Struct CONTOUR_VISU_DATA_V2.....            | 145 |
| 4.29.6  | Struct CONTOUR_VISU_DATA_V3.....            | 145 |
| 4.29.7  | Struct CONTOUR_VISU_DATA_V4.....            | 146 |
| 4.29.8  | Struct CONTOUR_VISU_DATA_V5.....            | 146 |
| 4.29.9  | Struct CONTOUR_VISU_DATA_V6.....            | 146 |
| 4.29.10 | Struct CONTOUR_VISU_DATA_V7.....            | 147 |
| 4.29.11 | Struct CONTOUR_VISU_DATA_V8.....            | 147 |
| 4.29.12 | Struct CONTOUR_VISU_DATA_V9.....            | 147 |
| 4.29.13 | Struct CONTOUR_VISU_DATA_V10.....           | 148 |
| 4.29.14 | Struct CONTOUR_VISU_DATA_V11.....           | 148 |
| 4.29.15 | Struct CONTOUR_VISU_CH_DATA.....            | 149 |
| 4.29.16 | Struct CONTOUR_VISU_CH_DATA_V1.....         | 150 |
| 4.29.17 | Struct CONTOUR_VISU_CH_DATA_V2.....         | 151 |
| 4.29.18 | Struct CONTOUR_AXIS_DATA.....               | 151 |
| 4.29.19 | Struct CONTOUR_AXIS_DATA_V1.....            | 152 |
| 4.29.20 | Struct CONTOUR_AXIS_DATA_V2.....            | 152 |
| 4.29.21 | Enum E_CONTOUR_TECHNO_TYPE.....             | 152 |
| 4.29.22 | Struct CONTOUR_M_H_PROCESS.....             | 153 |
| 4.29.23 | Struct CONTOUR_M_H_PROCESS_V1.....          | 153 |
| 4.29.24 | Enum E_CONTOUR_S_CMD.....                   | 153 |
| 4.29.25 | Struct CONTOUR_S_PROCESS.....               | 154 |
| 4.29.26 | Struct CONTOUR_TOOL_PROCESS.....            | 154 |
| 4.29.27 | Struct CONTOUR_DATA_TECHNO.....             | 154 |
| 4.29.28 | Struct CONTOUR_DATA_TECHNO_V1.....          | 155 |
| 4.30    | Data types of error output.....             | 156 |
| 4.30.1  | Struct KERNELV_ERROR_VALUE.....             | 156 |
| 4.30.2  | KERNELV_ERROR_VALUE_ARRAY.....              | 156 |
| 4.30.3  | Enum E_KERNELV_ERR_VAL_TYPE.....            | 157 |
| 4.30.4  | Enum E_KERNELV_ERR_VAL_DIMENSION.....       | 158 |



|          |  |            |
|----------|--|------------|
| 4.30.5   | Enum E_KERNELV_ERR_VAL_MEANING .....       | 159        |
| 4.31     | Enum KERNELV_AXIS_OFFSET_TYPES .....       | 160        |
| 4.32     | External measuring hardware .....          | 161        |
| 4.32.1   | Struct KERNELV_EXT_LATCH_COMMAND_DATA..... | 161        |
| 4.32.2   | Enum E_KERNELV_EXT_LATCH_ORDER.....        | 161        |
| 4.32.3   | E_KERNELV_MEAS_ACTIVE_EDGE.....            | 161        |
| 4.33     | Production time calculation .....          | 162        |
| <b>5</b> | <b>kernelv API constants.....</b>          | <b>163</b> |
| 5.1      | KERNELV_VAR_STRING_LEN .....               | 163        |
| 5.2      | KERNELV_FILE_NAME_LENGTH.....              | 163        |
| 5.3      | KERNELV_VAR_NAME_LENGTH.....               | 163        |
| 5.4      | KERNELV_OPTION_LICENSE_CHECK_VERBOSE ..... | 163        |
| 5.5      | CONTOUR_MAX_DATA_V0 .....                  | 163        |
| 5.6      | CONTOUR_MAX_DATA_V1 .....                  | 164        |
| 5.7      | CONTOUR_MAX_DATA_V2 .....                  | 164        |
| 5.8      | CONTOUR_MAX_DATA_V3 .....                  | 164        |
| 5.9      | CONTOUR_MAX_DATA_V4 .....                  | 164        |
| 5.10     | CONTOUR_MAX_DATA_V5 .....                  | 164        |
| 5.11     | CONTOUR_MAX_DATA_V6 .....                  | 165        |
| 5.12     | CONTOUR_MAX_DATA_V7 .....                  | 165        |
| 5.13     | CONTOUR_MAX_DATA_V8 .....                  | 165        |
| 5.14     | CONTOUR_MAX_DATA_V9 .....                  | 165        |
| 5.15     | CONTOUR_MAX_DATA_V10 .....                 | 165        |
| 5.16     | CONTOUR_MAX_DATA_V11 .....                 | 166        |
| 5.17     | CONTOUR_MAX_M_H_DATA .....                 | 166        |
| 5.18     | CONTOUR_MAX_SPDL_DATA .....                | 166        |
| 5.19     | CONTOUR_AXIS_PER_CHANNEL .....             | 166        |
| 5.20     | KERNELV_ERROR_VALUE_COUNT .....            | 166        |
| 5.21     | KERNELV_ERR_MSG_STRING_LENGTH .....        | 167        |
| 5.22     | KERNELV_CHANNEL_TECHNO_DATA_COUNT.....     | 167        |
| 5.23     | KERNELV_AXIS_TECHNO_DATA_COUNT.....        | 167        |
| 5.24     | KERNELV_ERROR_VALUE_COUNT .....            | 167        |
| 5.25     | KERNELV_INSTANCE_PREFIX_MAX_LEN .....      | 168        |
| <b>6</b> | <b>Support and Service .....</b>           | <b>169</b> |
|          | <b>Index.....</b>                          | <b>170</b> |



## List of figures

|        |   |     |
|--------|---|-----|
| Fig. 1 | Select a TwinCAT 3 trial license .....          | 16  |
| Fig. 2 | Activate a TwinCAT 3 trial license.....         | 16  |
| Fig. 3 | The start screen of the demo application .....  | 22  |
| Fig. 4 | Display axis positions in demo application..... | 23  |
| Fig. 5 | State diagram of a CNC channel .....            | 126 |

# 1 Simulation DLL for CNC kernel

The ISG CNC kernel is integrated into a real-time environment for a standard CNC to arrive at real time-enabled, deterministic behaviour.

However, the real-time environment is not required for many applications in the environment of a CNC controller such as:

- production time calculation,
- advance collision checking or
- a visualisation.

The kernel's CNC simulation DLL provides users with an opportunity to use a virtual CNC within their own non-real time application.

With regard to existing functionality, the simulation DLL largely offers the same possibilities as the real-time kernel.

## **Known restrictions:**

- No access to hardware.
- The CNC treats axes as simulation axes.
- A PLC cannot be used and the API interfaces use the HLI to command the CNC kernel.
- No real time necessary or possible.

## ***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.

## 2 Description

### 2.1 DLL version



This description refers to the API version of the simulation DLL referred to in the document title. You can query the DLL version string by means of the `kernelv_get_api_version()` function.

---

### 2.2 Components

The simulation DLL consists of 3 components:

1. the simulation DLL `kernelv_mt.dll`
2. the header file `kernelv.h`
3. the lib file `kernelv_mt.lib` for implicit linking.

A valid license is also required to use the DLL.

#### Diagnosis

The “ahmi.exe” program is included for diagnostic purposes.

This program calls internal diagnostic data from the kernelv DLL and saves it to a text file.

The ahmi.exe program is only intended for diagnostic purposes and should not be used as a user interface for the kernelv DLL.

In addition, the internal diagnosis data is also stored in a file linked to the `kernelv_diagnosis_upload()` [[▶ 122](#)] function.

## 2.3 Dependencies

Use of the kernel DLL is linked to dependencies with components produced by other manufacturers.

### 2.3.1 kernelv as of V300

The following additional DLLs must be available on the application computer in order to use the kernelv DLL.

- TcAdsDII.DLL The corresponding version of the DLL is used (depending on the version of the kernelv DLL (32-bit or 64-bit)).
- VCRUNTIME140.DLL

The TcAdsDII.DLL file is automatically installed when TwinCAT is installed.

If VCRUNTIME140.DLL is missing, it indicates that “Visual C++ Re-distributable for Visual Studio 2015/2017/2019” should be installed.

These packets can be downloaded free of charge from Microsoft®.

Please check that you install the correct version (32-bit or 64-bit).

## 2.4 Multiple instances

Only one kernelv instance can be started per application. However, it is possible to run several applications on the same PC each with one kernelv instance.

If an attempt is made to start 2 instances of kernelv within an application, e.g. by 2 calls of `kernelv_startup()` or `kernelv_startup_instance()`, the start of the 2nd instance is prevented and the error code `ERR_DOUBLE_INSTANCE` is returned.

To start several kernelv instances, use the function `kernelv_startup_instance()`. In this case, the transferred instance identifier must be unique on the computer for every kernelv instance started.

If an attempt is made to start 2 kernelv instances with the same instance identifier, the start of the 2nd instance is prevented and the error code `ERR_DOUBLE_KERNEL` is returned..

### 2.4.1 Diagnostics with multiple instances

The `ahmi.exe` program can also be used for diagnostic purposes with several instances of kernelv DLL.

When the `ahmi.exe` program is started, it must be specified which kernelv DLL instance is to be connected.

There are 2 options for this:

1. Specify the instance identifier as command line parameter at program start. Here, use `kernelv_startup_instance()` [▶ 30] to specify the same instance identifier as used at the start of the kernelv instance. The kernelv instance is specified by the parameter `-instance_prefix`.  
**Example:** `ahmi.exe -instance_prefix 1_`
2. The command line parameter `-query_instance_prefix` specifies the instance identifier of the kernelv instance at program start.  
**Example:** `ahmi.exe -query_instance_prefix`  
The function `kernelv_diagnosis_upload()` [▶ 122] starts the upload of diagnosis data for the instance containing the function call.

## 2.5 License protection

---

● The DLL and all related functions are only available in TwinCAT systems.

**i**

---

● A valid license is required to use the DLL.

**i**

---

● As of CNC Build V3.1.3104.01 at least one basic license is required in order to use the DLL.

**i** Use only via an option package is not possible

---

The `kernelv_startup()` checks for the presence of a license when the simulation kernel is started. If a valid license is not found, the start of the simulation kernel is aborted and the error code `ERR_NO_LICENSE` is returned.

If errors occur on accessing license information, the error code `ERR_REGISTRY_ACCESS` is returned. This may occur if

- VNCK was incorrectly installed under TwinCAT 2 or
- no trial license was generated under TwinCAT 3.

In addition, license packs are also required to use certain functions, e.g. transformations or when the number of axes is  $> 8$ . The presence of these license packs is checked during kernelv runtime and if necessary a CNC error message is output.

### 2.5.1 Licensing under TwinCAT 2

Under TwinCAT 2 the license information is written to the Windows registry when the VNCK is installed. This information is then requested from there when the kernelv DLL is started.

After installation no further action need be taken by the user.

### 2.5.2 Licensing under TwinCAT 3

#### TwinCAT 3 trial license

As for all other TwinCAT 3 software modules, a 7-day trial license can be generated for the kernelv DLL. However, in this case a TwinCAT installation must be present on the computer.

Take the following actions to generate the trial license:

1. Start TwinCAT XAE.
2. Generate a new empty TwinCAT project.
3. Select the System\License node on the left-hand side of the TwinCAT XAE in the tree view.
4. In the tab view then displayed, select the 'Manage Licenses' tab and select the options 'TC3 CNC Virtual NCK Basis' (TF5270) and eventually 'TC3 CNC Virtual NCK Options' from the options list.
5. Activate the trial license by clicking the '7 Days Trial License' in the 'Order Information (Runtime)' tab and enter the security code required.

#### TwinCAT 3 perpetual license

Request a perpetual license using the standard practice in TwinCAT 3.

### Using kernelv DLL with license check

When the CNC kernel is started with the kernelv\_startup() function, the required license information is requested from the TwinCAT license server. Communication between the CNC kernel and the license server takes place via ADS. It is therefore necessary for the application using kernelv DLL to have access to the TcAdsDll.dll library. This library is included in the TwinCAT installation.

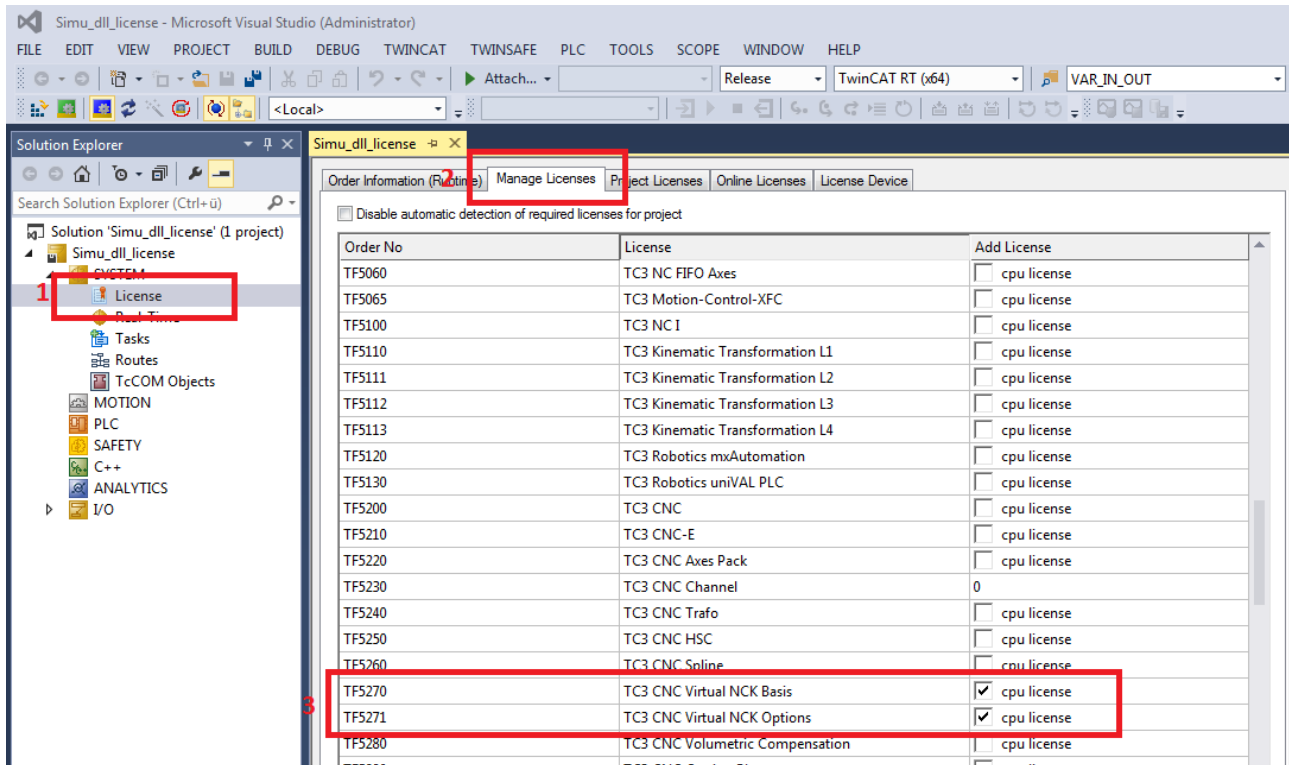


Fig. 1: Select a TwinCAT 3 trial license

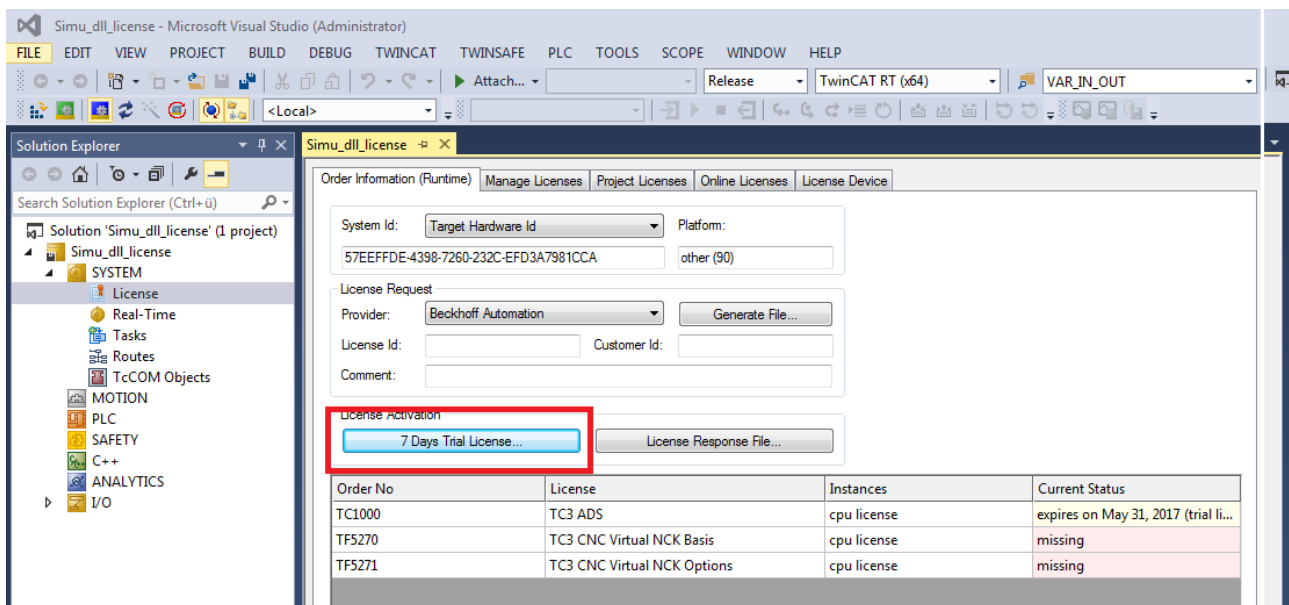


Fig. 2: Activate a TwinCAT 3 trial license



## 2.6 General

### 2.6.1 Identification of axes and channels

#### Channel identification

Channels are identified by their index on the PLC interface. The sequence of channels corresponds to the configuration sequence.

The channel first configured is addressed by the index 0 and the last channel configured out of n channels is addressed by the index n – 1.

If an invalid channel index is transferred to a function, the return value ERR\_INVALID\_CHAN (defined in kernelv.h) is returned.

If functions supply channel numbers as the return value, the associated channel index can be determined by means of the relationship

channel index = channel number - 1

.

#### Axis identification

Axes are identified by their index on the PLC interface. The sequence of axes corresponds to the configuration sequence.

The axis first configured is addressed by the index 0 and the last axis configured out of n axes is addressed by the index n – 1.

If an invalid axis index is transferred to a function, the return value ERR\_INVALID\_AX (defined in kernelv.h) is returned.

### 2.6.2 Coordinate systems

#### ACS coordinates

ACS coordinates (ACS = **a**xis **c**oordinate **s**ystem) are regarded as the coordinates of the physical axes.

Function names of functions that return ACS coordinates start with the prefix "kernelv\_get\_acs\_"

#### WCS coordinates

WCS coordinates (WCS = **w**orkpiece **c**oordinate **s**ystem) are regarded as the programming coordinate system.

The programming coordinate system can be shifted and rotated relative to the axis coordinate system, e.g. by reference point offsets and the definition of machining coordinate systems.

The names of functions that return WCS coordinates start with the prefix "kernelv\_get\_wcs\_"

#### Dimensions

Position values have a resolution of 0.1 µm for translatory axes or  $1 \times 10^{-4} \text{°}$  for spindles or modulo axes.

## 2.7 Use of kernelv

The DLL can be used by a user application by both implicit and explicit (dynamic) linking.

### Requirements for TwinCAT 3

With kernelv versions produced for a TwinCAT 3 environment (Build number of CNC version  $\geq 3000$ ), the application using the kernelv DLL must have access to the TwinCAT TcAdsDll.dll library. This library is installed when the TwinCAT is installed.

### 2.7.1 Start

#### kernelv\_startup()

The simulation CNC is started by calling the API function `kernelv_startup()`. Call parameters are the cycle time in microseconds and the number of axes and channels.

The function returns the value `RET_FINISHED` if the start of the simulation CNC was successful, otherwise an error code is displayed.

Only one instance of the simulation CNC can ever run on the same computer at the same time. The error code `ERR_DOUBLE_KERNEL` is returned whenever the `kernelv_startup()` function is called while an instance of the simulation CNC is still running.

A valid license is also required to use the simulation CNC. If no valid license is found when the simulation CNC is started, the start of the simulation CNC is cancelled and the error code `ERR_NO_LICENSE` is returned.

### 2.7.2 Cyclic operation

#### kernelv\_do\_cycle()

When the `kernelv_do_cycle()` function is called, one cycle of the simulation CNC is computed for all channels.

The cycle time transferred when the simulation CNC is started is used as the cycle time.

A call in real time is not necessary.

### 2.7.3 End/restart

Currently no API function is available to end the simulation CNC.

If the simulation CNC needs to be restarted, the DLL can be cancelled if it is linked dynamically and it can then be reloaded. The simulation CNC can then be restarted.

## 2.7.4 Operation modes of kernelv

An NC program can be started in 2 different channel operation modes. Depending on the operation mode, the kernelv DLL has different functions available. The operation mode can be specified for specific channels and this must be specified at program start.

Currently, the following operation modes are available:

| Name                                 | E_KERNELV_PROG_START_MODE       | Meaning   |
|--------------------------------------|---------------------------------|---|
| <b>Normal operation mode</b>         | KERNELV_START_MODE_NORMAL       | Normal operation mode; axis motions are interpolated at the correct velocity and the movement of the physical axes is simulated.  |
| <b>Command contour visualisation</b> | KERNELV_START_MODE_CONTOUR_VISU | Command contour visualisation operation mode; interpolation vertices are only calculated depending on the visualisation grid set. |

### Normal operation mode

In normal operation mode the NC program is interpolated at the cycle time specified at the start of the kernelv DLL. The interpolation data generated corresponds to the data of a controller running in real time. The dynamic limits set for the axes are maintained.

### Command contour visualisation operation mode

Compared to normal operation mode, the command contour visualisation operation mode generates an approximate vertex sequence in order to visualise the programmed contour rapidly.

The output grid can be set for straight and curved contour elements with the functions

```
kernelv_ch_set_cont_visu_grid(),
kernelv_ch_set_cont_visu_rel_curvature_error() and
kernelv_ch_set_cont_visu_abs_curvature_error()
```

. The end points of a motion block are always output, irrespective of the output grid set.

The approximate output grid is obtained by calculating the velocity at which the contour element is interpolated in order to obtain the output grid set. For this reason the dynamic axis data is not maintained. In the same way, the interpolated command values are not transferred to the position controller with the result that the ACS coordinates do not change in this operation mode.

### Select an operation mode

The operation mode must be specified when the kernelv\_ch\_program\_start() function is called.

```
kernelv_ch_program_start (0,
                          prog_name,
                          KERNELV_START_MODE_NORMAL);
```

### Contour visualisation output data

The vertices calculated in the contour visualisation operation mode are written to a FIFO memory and read by the function kernelv\_ch\_get\_cont\_visu\_data().

To avoid losing visualisation data, the interpolation is stopped when the internal FIFO is full. To execute an NC program as fast as possible, the kernelv\_ch\_get\_cont\_visu\_data() function must be called a sufficient number of times.

### Output format of contour visualisation

The data output when contour visualisation is active can be set system global by the parameter contour\_visu\_ifc\_version (P-STUP-00039) in the start-up list.

Depending on the interface version set, the format of the data returned by the function `kernelv_ch_get_cont_visu_data()` changes. See also the structure definition of `CONTOUR_VISU`.

Interpolated WCS coordinates can be read by the functions `kernelv_get_wcs_`.

## 2.7.5 Acknowledgement of technology functions

Technology functions (such as M or H functions) are used to exchange information between the NC program and the software components in the environment (such as PLC) during a real CNC control.

This includes the actual technology information output by the CNC as well as acknowledgements transferred by the software components in the environment to the CNC to synchronise program sequences within and outside of the CNC. When an NC program is processed, these synchronisations can lead to delays in the processing of the NC program.

### Default behaviour

All technology functions output are immediately automatically acknowledged by the `kernelV-DLL`.

### Execution time simulation

For the execution time to be taken into account in the execution of an NC program, the API function `kernelv_control techno_func_duration()` [▶ 74] can be used to activate the execution time simulation for the technology functions..

When the execution time simulation is activated, technology functions are automatically acknowledged after an adjustable time period.

The execution time can be set by means of the `kernelv_ch_set techno_func_duration()` [▶ 75] function or by means of entries in the channel parameter list (Time-out and process times of M functions (P-CHAN-00040), Time-out / process times of H functions (P-CHAN-00026)).

### User acknowledgement

By means of the user acknowledgement, the user himself has full control over the time of acknowledgement of the technology function.

For this purpose, the execution time simulation must first be acknowledged by invoking the function `kernelv_control techno_func_duration()` [▶ 74].

Technology functions for which user acknowledgement is to be activated must be marked using the function `kernelv_ch_set techno_func_user_ackn()` [▶ 76]. All other technology functions are acknowledged after the respective execution time is over.

The functions `kernelv_ch_get_new techno_data()` [▶ 44] or `kernelv_ax_get_new techno_data()` [▶ 49] have to be invoked to check if the respective technology function has been output.

At the time of acknowledgement, the technology function must be acknowledged by invoking the function `kernelv_ch_ackn techno_func()` [▶ 77] or `kernelv_ax_ackn techno_func()` [▶ 78].

## 2.7.6 Search path for NXC programs

NC programs are sought relative to the application's start directory. Additionally, search paths for programs and global subroutines can be specified in the start-up list (see CNC kernel documentation).

## 2.7.7 kernelv demo application

As an example for use of the `kernelv` DLL, a demo application including a demo parameter set is available in which the `kernelv` DLL is explicitly loaded. Users have the option to start NC programs and display axis positions via a simple ASCII user interface.

The demo application is delivered in a zip archive. The actual kernelv DLL is supplied in a separate installation and is not included in the zip archive.

### 2.7.7.1 Project structure

The following folder structure is created when the demo application is unpacked:

```
kernelv_demo
|
|----\listen
L----\prg
```

The kernelv\_demo directory contains the following files:

|                                       |   |
|---------------------------------------|---|
| kernelv_demo.dsw,<br>kernelv_demo.dsp | Project files, demo application Visual Studio 6               |
| kernelv_demo.c                        | C source text of the demo application                         |
| err_text_version.txt                  | File containing error message texts                           |
| kernelv_demo.bat                      | Batch file for starting the demo application (debug version). |

The configuration lists of the CNC kernel are saved in the subdirectory \listen. The subdirectory \prg contains a simple example program.

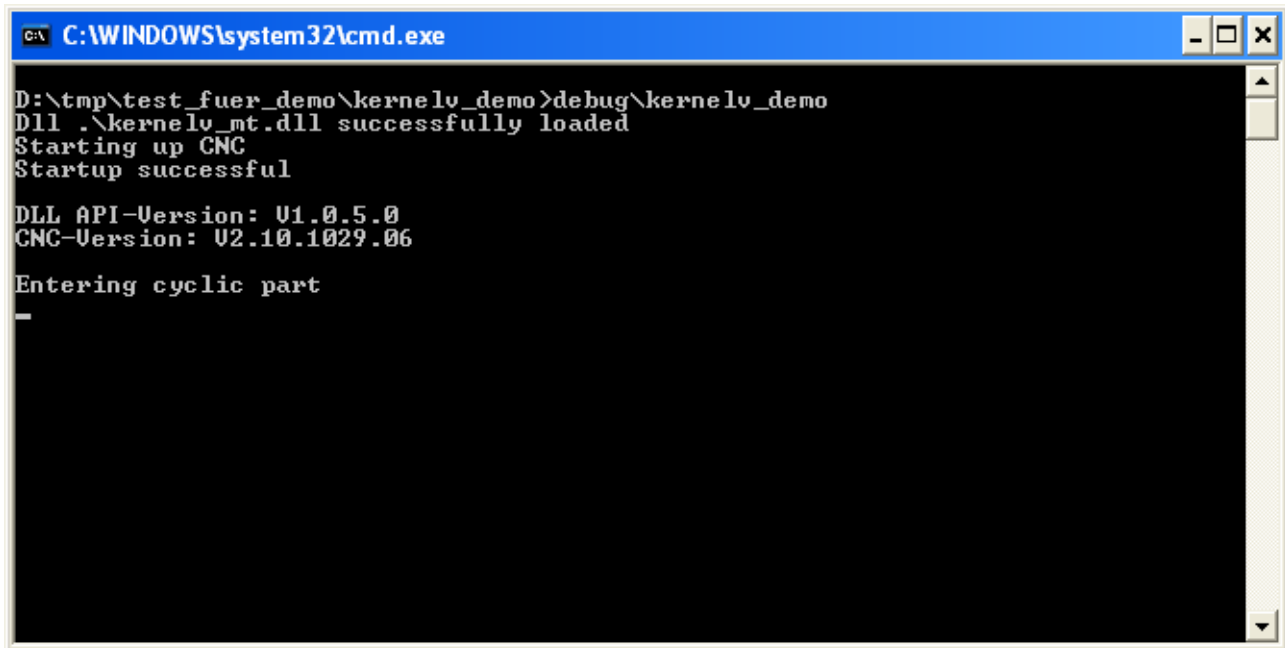
### 2.7.7.2 Use of the demo application

The following steps are required to use the demo application:

1. Unpack the demo.
2. Copy the kernelv DLL and kernelv.h to the kernelv\_demo directory that was created when unpacked.
3. Compile and start the project. It can be started either in the Visual Studio debugger or by means of the kernelv\_demo.bat file.

#### Start screen

After start, the screen below is displayed:



```

C:\WINDOWS\system32\cmd.exe
D:\tmp\test_fuer_demo\kernelv_demo>debug\kernelv_demo
Dll .\kernelv_mt.dll successfully loaded
Starting up CNC
Startup successful
DLL API-Version: U1.0.5.0
CNC-Version: U2.10.1029.06
Entering cyclic part
_

```

Fig. 3: The start screen of the demo application

#### Keyboard commands

| Input   | Effect  |
|---------|---|
| S       | Starts the program. If no program name is specified, an attempt is made to start the NC program 'test1.nc'. A search for this is made in the './prg' directory. |
| R       | Executes CNC reset.   |
| P       | Reads out and displays axis positions.  |
| Q       | Quits the application   |
| 'Enter' | Status display  |

### Display axis positions

The screen below appears after the supplied test program is executed and the axis positions are displayed:

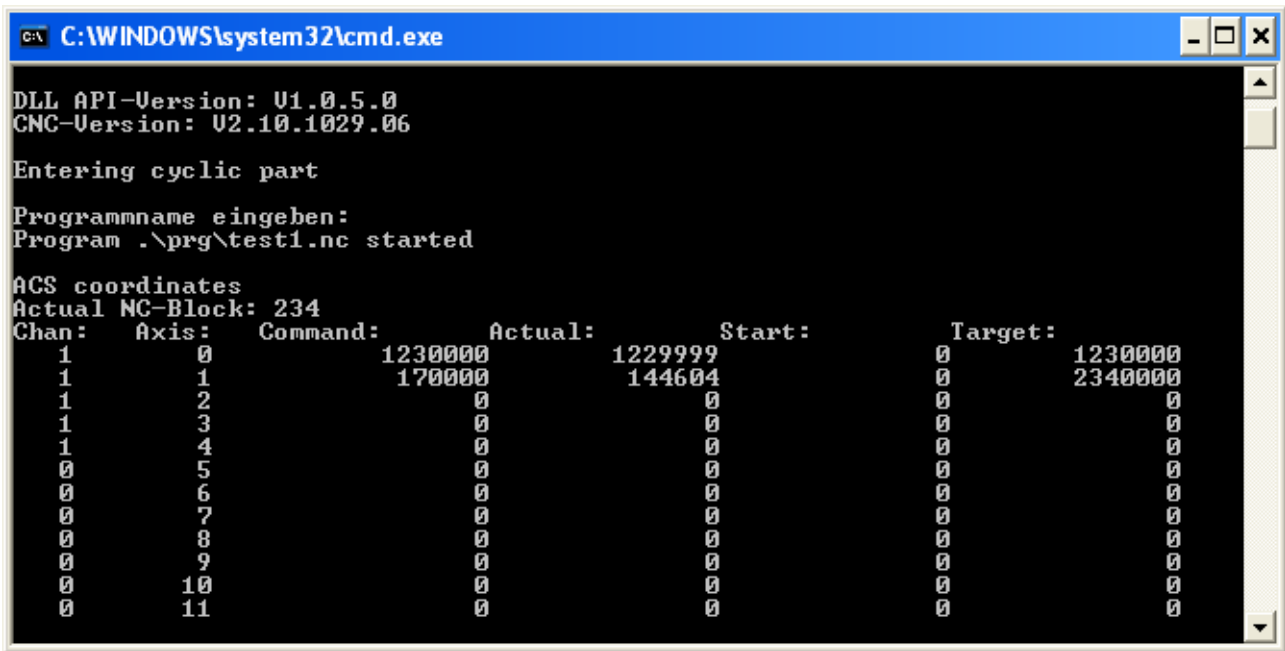


Fig. 4: Display axis positions in demo application

### 2.7.7.3 Explanations about demo application

All steps required to use the kernelv DLL are executed in the demo application source code. Refer to the respective compiler documentation for further information on the operating system functions called.

Load DLL:

```
hDll = LoadLibrary(dll_name);
```

Query function pointers:

```
if ( !(pCycle = (KERNELV_CYCLE) GetProcAddress(hDll,
                                                "kernelv_do_cycle")) )
{
    printf("Missing pointer to kernelv_do_cycle()\n");
    return -2;
}
```

If the requested symbol is not defined in the DLL, a message is output and the application ends. Refer to the section 'kernelv API' in this document for details on how to use individual functions.

Main loop for handling user inputs and cyclic call of kernelv\_do\_cycle(). One CNC cycle is computed:

```
(pCycle)();
```

## 2.8 Configuration

In the current version of the simulation DLL, the CNC kernel is configured by ASCII lists. These ASCII lists can be generated from a TwinCAT configuration file by the "Listenexporteur" (list exporter) program.

### 2.8.1 Path of the configuration lists

#### Start-up file

The start-up file contains the basic configuration of the CNC kernel, e.g.:

- number of axes,
- number of channels and
- the access path to the various axis- or channel-specific configuration files.

The path and file name of the start-up file (default: hochlauf.lis) must be transferred to the API function `kernelv_startup`.

You must specify the access paths to other parameter lists in the start-up list:

- Relative paths within the start-up list must be specified relative to the start-up list storage location.
- Absolute paths are adopted without change.



You are advised to generate the parameter lists either by exporting the lists in the TwinCAT system manager or by using the 'Listenexporteur' tool.

---

## 2.9 Error message texts

#### Assign error message number to error message text

When an error message is output, the CNC kernel only outputs an error message code together with several parameters. An error message text is not assigned until later.

Each error message number is assigned to an error message text by the 'err\_text\_version.txt' file. This file must reside in the work directory of the application that uses the simulation DLL.



## 2.10 Tool management

### Internal/external tool management

When the CNC kernel is used as simulation DLL, both internal and external tool management can be used (toggle by using P-CHAN-00016).

### Tool data

With internal tool management, tool data is saved to the tool data list channel-specific.

When external tool management is used, tool data is always managed globally for the entire CNC: In this case, tool data must be entered in the tool data list of the 1st channel.

The access path to the tool data list must be specified in the start-up list.

The tool data list format is described in [TOOL].

## 3 kernelv API functions

### 3.1 kernelv\_get\_api\_version()

#### Prototype

```
KERNELV_RETURN kernelv_get_api_version (char* versionString,
                                         unsigned long maxStringLength,
                                         unsigned long* returnSize);
```

#### Description

Reads the API version string.

#### Parameter

| Name            | Type           | Meaning   |
|-----------------|----------------|---|
| versionString   | char*          | Pointer to the version string storage location. The application must provide sufficient memory. |
| maxStringLength | unsigned long  | Length of the memory provided by the application (in bytes).                                    |
| returnSize      | unsigned long* | Length of the version string returned. The value 0 is returned if an error has occurred.        |

#### Return values

| Symbol             | Value | Meaning  |
|--------------------|-------|--|
| RET_FINISHED       | 0     | The function was executed without error.   |
| ERR_CNC_NOT_INIT   | -3    | The simulation CNC kernel was not yet initialised.   |
| ERR_CNC_RET_MEMORY | -4    | The return value(s) does/do not fit into the memory provided. The number of required bytes including the terminating zero is returned in the returnSize parameter. |

## 3.2 kernelv\_get\_cnc\_version()

### Prototype

```
KERNELV_RETUR kernelv_get_cnc_version(char* versionString,  
N unsigned long maxStringLength,  
unsigned long* returnSize);
```

### Description

Reads the CNC version string.

### Parameter

| Name            | Type           | Meaning   |
|-----------------|----------------|---|
| versionString   | char*          | Pointer to the version string storage location. The application must provide sufficient memory. |
| maxStringLength | unsigned long  | Length of the memory provided by the application (in bytes).                                    |
| returnSize      | unsigned long* | Length of the version string returned. The value 0 is returned if an error has occurred.        |

### Return values

| Symbol               | Value | Meaning  |
|----------------------|-------|--|
| RET_FINISHED         | 0     | The function was executed without error.   |
| err_text_version.txt | -3    | The simulation CNC kernel was not yet initialised.   |
| ERR_CNC_RET_MEMORY   | -4    | The return value(s) does/do not fit into the memory provided. The number of required bytes including the terminating zero is returned in the returnSize parameter. |

### 3.3 kernelv\_get\_cycletime()

#### Prototype

```
KERNELV_RETUR kernelv_get_cycletime (unsigned long* cycleTime);  
N
```

#### Description

Reads the CNC cycle time.

#### Parameter

| Name      | Type           | Meaning        |
|-----------|----------------|----------------|
| cycleTime | unsigned long* | Cycle time us. |

#### Return values

| Symbol           | Value | Meaning  |
|------------------|-------|--|
| RET_FINISHED     | 0     | The function was executed without error.           |
| ERR_CNC_NOT_INIT | -3    | The simulation CNC kernel was not yet initialised. |

### 3.4 kernelv\_startup()

**Prototype**

```
KERNELV_RETURN kernelv_startup( unsigned long cycleTime,
                                char* startupFile);
```

**Description**

Starts the simulation CNC.

**Parameter**

| Name        | Type          | Meaning                             |
|-------------|---------------|-------------------------------------|
| cycleTime   | unsigned long | Cycle time us.                      |
| startupFile | char *        | Path and name of the start-up file. |

**Return values**

| Symbol                | Value | Meaning  |
|-----------------------|-------|--|
| RET_FINISHED          | 0     | The function was executed without error.   |
| ERR_DOUBLE_KERNEL     | -6    | An instance of the simulation CNC is already running.  |
| ERR_SHM_STARTUP       | -7    | Internally used shared memories could not be created when the CNC kernel was started.  |
| ERR_STARTUP           | -8    | An error occurred on starting the simulation CNC. Possible causes are missing parameter lists or incorrect entries in parameter lists.   |
| ERR_NO_LICENSE        | -17   | No license was found for the use of the kernelv DLL.   |
| ERR_REGISTRY_ACCESS   | -19   | An error occurred when an attempt was made to read values from the Windows registry.   |
| ERR_PREFIX_TOO_LONG   | -23   | When the function kernelv_startu_prefix() was called, the transferred instance identifier is too long with the result that the internally generated names for the shared memories used no longer fit in the memory provided. |
| ERR_STARTUP_CHAN_INIT | -31   | When kernelv-Dll was started, it was not possible to execute the initialisation of the configured NC channels.   |



Depending on the number of axes and channels in the configuration used, the start of the CNC kernel may take 20 - 30 seconds.

## 3.5 kernelv\_startup\_instance()

### Prototype

```
KERNELV_RETUR kernelv_startup_instance (unsigned long cycleTime,
N                                     char* startupFile
                                       char* instancePrefix);
```

### Description

Start the simulation CNC:

This function permits several applications to run on one computer. Each application uses a single and therefore a separate instance of the kernelv DLL. Here, a unique instance identifier must be transferred in the call parameter instancePrefix and is valid for the entire computer.

The maximum length of the string that may be transferred as an instance identifier is defined by the constant KERNELV\_INSTANCE\_PREFIX\_MAX\_LEN. If a longer string is transferred, the start-up is not executed and the function returns the value ERR\_PREFIX\_TOO\_LONG (-23).

It is not possible to run several instances of the kernelv DLL in one application.

### Parameter

| Name           | Type          | Meaning                             |
|----------------|---------------|-------------------------------------|
| cycleTime      | unsigned long | Cycle time us.                      |
| startupFile    | char *        | Path and name of the start-up file. |
| instancePrefix | char *        | Unique instance identifier.         |

### Return values

| Symbol                | Value | Meaning  |
|-----------------------|-------|--|
| RET_FINISHED          | 0     | The function was executed without error.   |
| ERR_DOUBLE_KERNEL     | -6    | An instance of the simulation CNC is already running.  |
| ERR_SHM_STARTUP       | -7    | Internally used shared memories could not be created when the CNC kernel was started.  |
| ERR_STARTUP           | -8    | An error occurred on starting the simulation CNC. Possible causes are missing parameter lists or incorrect entries in parameter lists.   |
| ERR_NO_LICENSE        | -17   | No license was found for the use of the kernelv DLL.   |
| ERR_REGISTRY_ACCESS   | -19   | An error occurred when an attempt was made to read values from the Windows registry.   |
| ERR_PREFIX_TOO_LONG   | -23   | When the function kernelv_startu_prefix() was called, the transferred instance identifier is too long with the result that the internally generated names for the shared memories used no longer fit in the memory provided. |
| ERR_STARTUP_CHAN_INIT | -31   | When kernelev-Dll was started, it was not possible to execute the initialisation of the configured NC channels.  |



Depending on the number of axes and channels in the configuration used, the start of the CNC kernel may take 20 - 30 seconds.

## 3.6 kernelv\_do\_cycle()

### Prototype

```
KERNELV_RETUR kernelv_do_cycle();  
N
```

### Description

Computes one simulation CNC cycle.

The cycle time transferred as a parameter at the start is used for the internal calculations. If an error should occur in the simulation CNC, it can be checked by the `kernelv_get_error ()` function and the error message string can be read out.

### Parameter

None

### Return values

| Symbol           | Value | Meaning  |
|------------------|-------|--|
| RET_FINISHED     | 0     | The function was executed without error.             |
| ERR_CNC_NOT_INIT | -3    | The simulation CNC kernel was not yet initialised.   |
| ERR_NO_LICENSE   | -17   | No licence was found for the use of the kernelv DLL. |

## 3.7 kernelv\_ch\_program\_start()

### Prototype

```
KERNELV_RETURN kernelv_ch_program_start (unsigned long chanIndex,
                                         char* name,
                                         unsigned long mode);
```

### Description

Starts an NC program in the specified channel.

### Parameter

| Name      | Type           | Meaning   |
|-----------|----------------|---|
| chanIndex | unsigned long  | Channel index of the channel in which the program is to be started.   |
| name      | char*          | Name of the program to be started.  |
| mode      | unsigned long* | Execution mode in which the program is started.<br>Possible execution modes, see E_KERNELV_PROG_START_MODE. |

### Return values

| Symbol                 | Value | Meaning  |
|------------------------|-------|--|
| RET_FINISHED           | 0     | The function was executed without error.   |
| RET_BUSY               | 1     | The function is currently being executed, but has not yet been completed.<br>The API function must continue to be called.  |
| ERR_INVALID_CHAN       | -1    | The transferred channel index is higher than the number of configured channels - 1   |
| ERR_PROG_NAME_LENGTH   | -2    | The transferred program name is longer than permitted.   |
| ERR_CNC_NOT_INIT       | -3    | The simulation CNC kernel was not yet initialised.   |
| ERR_INVALID_START_MODE | -25   | An invalid start mode was transferred when the function kernelv_ch_program_start() was started.<br>For valid execution mode values, see E_KERNELV_PROG_START_MODE. |



## 3.8 kernelv\_ch\_reset()

### Prototype

```
KERNELV_RETURN kernelv_ch_reset (unsigned long chanIndex);
```

### Description

Executes a CNC reset in the specified channel.

This resets internal CNC errors. Any program running during reset is aborted.

### Parameter

| Name      | Type          | Meaning   |
|-----------|---------------|---|
| chanIndex | unsigned long | Channel index of the channel in which the program is to be started. |

### Return values

| Symbol           | Value | Meaning   |
|------------------|-------|---|
| RET_FINISHED     | 0     | The function was executed without error.  |
| RET_BUSY         | 1     | The function is currently being executed, but has not yet been completed.<br>The API function must continue to be called. |
| ERR_INVALID_CHAN | -1    | The transferred channel index is higher than the number of configured channels - 1  |
| ERR_CNC_NOT_INIT | -3    | The simulation CNC kernel was not yet initialised.  |

## 3.9 kernelv\_ch\_suspend()

### Prototype

KERNELV\_RETURN kernelv\_ch\_suspend (unsigned long int chanIndex);

### Description

The program currently running in the channel is stopped. The channel state changes to SIMU\_CNC\_STATE\_HOLD.

It is only possible to stop a program if an NC program is currently being executed in the channel, i.e. when the channel is in the SIMU\_CNC\_STATE\_ACTIVE state. The function returns the value ERR\_INVALID\_STATE if it is called while the channel is in another state.

### Parameter

| Name      | Type          | Meaning   |
|-----------|---------------|---|
| chanIndex | unsigned long | Channel index of the channel that is to be stopped. |

### Return values

| Symbol            | Value | Meaning   |
|-------------------|-------|---|
| RET_FINISHED      | 0     | The function was executed without error.  |
| RET_BUSY          | 1     | The function is currently being executed, but has not yet been completed.<br>The API function must continue to be called. |
| ERR_INVALID_CHAN  | -1    | The transferred channel index is higher than the number of configured channels - 1  |
| ERR_CNC_NOT_INIT  | -3    | The simulation CNC kernel was not yet initialised.  |
| ERR_INVALID_STATE | -5    | The CNC channel is in the wrong state to execute a function.  |

### 3.10 kernelv\_ch\_resume()

**Prototype**

KERNELV\_RETURN kernelv\_ch\_resume (unsigned long int chanIndex);

**Description**

The program currently running in the channel is continued. The channel state changes to SIMU\_CNC\_STATE\_ACTIVE.

A program can only be continued if the channel is in the SIMU\_CNC\_STATE\_HOLD state. The function returns the value ERR\_INVALID\_STATE if it is called while the channel is in another state.

**Parameter**

| Name      | Type          | Meaning   |
|-----------|---------------|---|
| chanIndex | unsigned long | Channel index of the channel that is to be continued. |

**Return values**

| Symbol            | Value | Meaning   |
|-------------------|-------|---|
| RET_FINISHED      | 0     | The function was executed without error.  |
| RET_BUSY          | 1     | The function is currently being executed, but has not yet been completed.<br>The API function must continue to be called. |
| ERR_INVALID_CHAN  | -1    | The transferred channel index is higher than the number of configured channels - 1  |
| ERR_CNC_NOT_INIT  | -3    | The simulation CNC kernel was not yet initialised.  |
| ERR_INVALID_STATE | -5    | The CNC channel or the axis is in the wrong state to execute a function.  |

## 3.11 kernelv\_ch\_get\_override()

### Prototype

```
KERNELV_RETURN kernelv_ch_get_override (unsigned long int chanIndex,
                                         unsigned short int *override);
```

### Description

Reads the current override value for the channel.

### Parameter

| Name      | Type            | Meaning  |
|-----------|-----------------|--|
| chanIndex | unsigned long   | Channel index of the channel whose override is to be read.   |
| override  | unsigned short* | Pointer to the value to which the override is to be written. The current override value is returned in 0.1%. |

### Return values

| Symbol           | Value | Meaning  |
|------------------|-------|--|
| RET_FINISHED     | 0     | The function was executed without error.   |
| ERR_INVALID_CHAN | -1    | The transferred channel index is higher than the number of configured channels -1. |
| ERR_CNC_NOT_INIT | -3    | The simulation CNC kernel was not yet initialised.                                 |

## 3.12 kernelv\_ch\_set\_override()

### Prototype

```
KERNELV_RETUR kernelv_ch_set_override (unsigned long int chanIndex,  
N unsigned short int override);
```

### Description

Sets the current override value for the channel.

### Parameter

| Name      | Type           | Meaning   |
|-----------|----------------|---|
| chanIndex | unsigned long  | Channel index of the channel whose override is to be set. |
| override  | unsigned short | Override value to be set in 0.1%.                         |

### Return values

| Symbol           | Value | Meaning  |
|------------------|-------|--|
| RET_FINISHED     | 0     | The function was executed without error.   |
| ERR_INVALID_CHAN | -1    | The transferred channel index is higher than the number of configured channels -1. |
| ERR_CNC_NOT_INIT | -3    | The simulation CNC kernel was not yet initialised.                                 |

### 3.13 kernelv\_ch\_get\_blocknumber()

#### Prototype

KERNELV\_RETURN kernelv\_ch\_get\_blocknumber (unsigned long int chanIndex,  
signed long int \*blocknumber);

#### Description

Reads the currently executed block number of an NC program.

#### Parameter

| Name        | Type          | Meaning  |
|-------------|---------------|--|
| chanIndex   | unsigned long | Channel index of the channel whose block number is to be read.   |
| blocknumber | signed long*  | Pointer to the value to which the block number is to be written. |

#### Return values

| Symbol           | Value | Meaning  |
|------------------|-------|--|
| RET_FINISHED     | 0     | The function was executed without error.   |
| ERR_INVALID_CHAN | -1    | The transferred channel index is higher than the number of configured channels -1. |
| ERR_CNC_NOT_INIT | -3    | The simulation CNC kernel was not yet initialised.                                 |

### 3.14 kernelv\_ch\_get\_filename()

**Prototype**

```
KERNELV_RETURN kernelv_ch_get_filename (unsigned long int chanIndex,
                                         char *filename,
                                         unsigned short int nameLength,
                                         unsigned short int* returnLength);
```

**Description**

Reads the file name of the program currently active in the channel.

**Parameter**

| Name         | Type            | Meaning   |
|--------------|-----------------|---|
| chanIndex    | unsigned long   | Channel index of the channel whose file name is to be read.   |
| filename     | char*           | Pointer to the memory location for the file name.   |
| nameLength   | unsigned short  | Length of the memory area for the file name.  |
| returnLength | unsigned short* | Pointer to the value to which the actually returned number of bytes is to be written. The number of characters in the file name +1 is returned. |

**Return values**

| Symbol             | Value | Meaning   |
|--------------------|-------|---|
| RET_FINISHED       | 0     | The function was executed without error.  |
| ERR_INVALID_CHAN   | -1    | The transferred channel index is higher than the number of configured channels -1.  |
| ERR_CNC_NOT_INIT   | -3    | The simulation CNC kernel was not yet initialised.  |
| ERR_CNC_RET_MEMORY | -4    | The return value(s) does/do not fit into the memory provided. In this case, the required memory including the terminating 0 are returned in the returnLength parameter. |

## 3.15 kernelv\_ch\_get\_programname()

### Prototype

```
KERNELV_RETURN kernelv_ch_get_programname (unsigned long int chanIndex,
char *programname,
unsigned short int nameLength,
unsigned short int* returnLength);
```

### Description

Reads the file name of the program currently active in the channel. The program name is specified at the start of the NC program (see also the programming instructions).

### Parameter

| Name         | Type            | Meaning   |
|--------------|-----------------|---|
| chanIndex    | unsigned long   | Channel index of the channel whose program name is to be read.  |
| programname  | char*           | Pointer to the memory location for the program name.  |
| nameLength   | unsigned short  | Length of the memory area for the program name.   |
| returnLength | unsigned short* | Pointer to the value to which the actually returned number of bytes is to be written. The number of characters in the file name +1 is returned. |

### Return values

| Symbol             | Value | Meaning  |
|--------------------|-------|--|
| RET_FINISHED       | 0     | The function was executed without error.   |
| ERR_INVALID_CHAN   | -1    | The transferred channel index is higher than the number of configured channels -1.   |
| ERR_CNC_NOT_INIT   | -3    | The simulation CNC kernel was not yet initialised.   |
| ERR_CNC_RET_MEMORY | -4    | The return value(s) does/do not fit into the memory provided. In this case, the required memory is returned in the returnLength parameter. |



## 3.16 kernelv\_ch\_get\_state()

### Prototype

```
KERNELV_RETURN kernelv_ch_get_state (unsigned long int chanIndex,  
                                     KERNELV_CHANNEL_STATE *state);
```

### Description

Reads the channel's current state.

### Parameter

| Name      | Type                    | Meaning                              |
|-----------|-------------------------|--------------------------------------|
| chanIndex | unsigned long           | Channel index.                       |
| state     | KERNELV_CHANNEL_STATE * | Pointer to the state to be returned. |

### Return values

| Symbol             | Value | Meaning  |
|--------------------|-------|--|
| RET_FINISHED       | 0     | The function was executed without error.   |
| ERR_INVALID_CHAN   | -1    | The transferred channel index is higher than the number of configured channels -1. |
| ERR_CNC_NOT_INIT   | -3    | The simulation CNC kernel was not yet initialised.                                 |
| ERR_INTERNAL_ERROR | -11   | An internal error has occurred in the DLL.   |

## 3.17 kernelv\_ch\_get\_fileoffset()

### Prototype

```
KERNELV_RETURN kernelv_ch_get_fileoffset (unsigned long int chanIndex,
signed long int *fileoffset);
```

### Description

Returns the current file offset in the program file.

### Parameter

| Name       | Type              | Meaning   |
|------------|-------------------|---|
| chanIndex  | unsigned long     | Channel index   |
| fileoffset | signed long int * | Pointer to the value to which the file offset is to be written. |

### Return values

| Symbol           | Value | Meaning  |
|------------------|-------|--|
| RET_FINISHED     | 0     | The function was executed without error.   |
| ERR_INVALID_CHAN | -1    | The transferred channel index is higher than the number of configured channels -1. |
| ERR_CNC_NOT_INIT | -3    | The simulation CNC kernel was not yet initialised.                                 |

### 3.18 kernelv\_ch\_get techno\_data()

**Prototype**

```

KERNELV_RETURN kernelv_ch_get techno_data (unsigned long int chanindex,
N                                             KERNELV_TECHNO_DATA *
                                             technoData
                                             unsigned long int technoLength
                                             unsigned long int * returnLength);
    
```

**Description**

Returns the technology functions (M/H functions) acknowledged for the specified channel during the last call of kernelv\_do\_cycle().

**Parameter**

| Name         | Type                 | Meaning  |
|--------------|----------------------|--|
| chanIndex    | unsigned long        | Channel index  |
| technoData   | KERNELV_TECHNO_DATA* | Pointer to the memory area to which the technology data is to be written.          |
| technoLength | unsigned long        | Size of the provided memory area in bytes.   |
| returnLength | unsigned long*       | Pointer to the memory area to which the actually returned bytes are to be written. |

**Return values**

| Symbol             | Value | Meaning   |
|--------------------|-------|---|
| RET_FINISHED       | 0     | The function was executed without error.  |
| ERR_INVALID_CHAN   | -1    | The transferred channel index is higher than the number of configured channels -1.  |
| ERR_CNC_NOT_INIT   | -3    | The simulation CNC kernel was not yet initialised.  |
| ERR_CNC_RET_MEMORY | -4    | The return value(s) does/do not fit into the memory provided. The number of bytes required is returned in the returnLength parameter. |

## 3.19 kernelv\_ch\_get\_new techno\_data()

### Prototype

```
KERNELV_RETURN      kernelv_ch_get_new techno_data (unsigned long int chanindex,
                                                    KERNELV_TECHNO_DATA * technoData
                                                    unsigned long int technoLength
                                                    unsigned long int * returnLength);
```

### Description

Returns the technology functions (M/H functions) recently output during the last call of kernelv\_do\_cycle() for each specific channel.

### Parameter

| Name         | Type                 | Meaning  |
|--------------|----------------------|--|
| chanIndex    | unsigned long        | Channel index  |
| technoData   | KERNELV_TECHNO_DATA* | Pointer to the memory area to which the technology data is to be written.          |
| technoLength | unsigned long        | Size of the provided memory area in bytes.   |
| returnLength | unsigned long*       | Pointer to the memory area to which the actually returned bytes are to be written. |

### Return values

| Symbol             | Value | Meaning   |
|--------------------|-------|---|
| RET_FINISHED       | 0     | The function was executed without error.  |
| ERR_INVALID_CHAN   | -1    | The transferred channel index is higher than the number of configured channels - 1  |
| ERR_CNC_NOT_INIT   | -3    | The simulation CNC kernel was not yet initialised.  |
| ERR_CNC_RET_MEMORY | -4    | The return value(s) does/do not fit into the memory provided. The number of bytes required is returned in the returnLength parameter. |

## 3.20 kernelv\_ch\_get techno\_data2()

### Prototype

```
KERNELV_RETUR kernelv_ch_get techno_data2 (unsigned long int chanindex,
N
KERNELV_TECHNO_DATA2 * technoData2
unsigned long int technoLength
unsigned long int * returnLength);
```

### Description

Returns the technology functions (M/H functions) acknowledged for the specified channel during the last call of kernelv\_do\_cycle().

### Parameter

| Name         | Type                  | Meaning  |
|--------------|-----------------------|--|
| chanIndex    | unsigned long         | Channel index  |
| technoData   | KERNELV_TECHNO_DATA2* | Pointer to the memory area to which the technology data is to be written.          |
| technoLength | unsigned long         | Size of the provided memory area in bytes.   |
| returnLength | unsigned long*        | Pointer to the memory area to which the actually returned bytes are to be written. |

### Return values

| Symbol             | Value | Meaning   |
|--------------------|-------|---|
| RET_FINISHED       | 0     | The function was executed without error.  |
| ERR_INVALID_CHAN   | -1    | The transferred channel index is higher than the number of configured channels - 1  |
| ERR_CNC_NOT_INIT   | -3    | The simulation CNC kernel was not yet initialised.  |
| ERR_CNC_RET_MEMORY | -4    | The return value(s) does/do not fit into the memory provided. The number of bytes required is returned in the returnLength parameter. |

## 3.21 kernelv\_ch\_get\_new techno\_data2()

### Prototype

```
KERNELV_RETURN kernelv_ch_get_new techno_data2 (unsigned long int chanindex,
                                                KERNELV_TECHNO_DATA2 * technoData2
                                                unsigned long int technoLength
                                                unsigned long int * returnLength);
```

### Description

Returns the technology functions (M/H functions) recently output during the last call of kernelv\_do\_cycle() for each specific channel.

### Parameter

| Name         | Type                  | Meaning  |
|--------------|-----------------------|--|
| chanIndex    | unsigned long         | Channel index  |
| technoData   | KERNELV_TECHNO_DATA2* | Pointer to the memory area to which the technology data is to be written.          |
| technoLength | unsigned long         | Size of the provided memory area in bytes.   |
| returnLength | unsigned long*        | Pointer to the memory area to which the actually returned bytes are to be written. |

### Return values

| Symbol             | Value | Meaning   |
|--------------------|-------|---|
| RET_FINISHED       | 0     | The function was executed without error.  |
| ERR_INVALID_CHAN   | -1    | The transferred channel index is higher than the number of configured channels - 1  |
| ERR_CNC_NOT_INIT   | -3    | The simulation CNC kernel was not yet initialised.  |
| ERR_CNC_RET_MEMORY | -4    | The return value(s) does/do not fit into the memory provided. The number of bytes required is returned in the returnLength parameter. |

## 3.22 kernelv\_ch\_get\_finished\_nc\_lines()

### Prototype

```
KERNELV_RETURN kernelv_ch_get_finished_nc_lines(unsigned long int chanIndex,
                                                KERNELV_NC_LINE_DATA *ncLineData,
                                                unsigned long int maxByteSize,
                                                unsigned long int* returnLength);
```

### Description

Returns the NC lines executed for the specified channel during the last call of kernelv\_do\_cycle(). Up to 20 NC blocks can be executed for each call of kernelv\_do\_cycle().

An array of structures of the type KERNELV\_NC\_LINE\_DATA is returned.

The number of entries returned can be calculated by means of returnLength/sizeof(KERNELV\_NC\_LINE\_DATA).

The breakdown of the structure is described in the section entitled Struct KERNELV\_NC\_LINE\_DATA.

### Parameter

| Name         | Type                  | Meaning   |
|--------------|-----------------------|---|
| chanIndex    | unsigned long         | Channel index   |
| ncLineData   | KERNELV_NC_LINE_DATA* | Pointer to the memory area to which the data of the executed NC lines is to be written. |
| maxByteSize  | unsigned long         | Size of the provided memory area in bytes.  |
| returnLength | unsigned long*        | Pointer to the memory area to which the actually returned bytes are to be written.      |

### Return values

| Symbol             | Value | Meaning   |
|--------------------|-------|---|
| RET_FINISHED       | 0     | The function was executed without error.  |
| ERR_INVALID_CHAN   | -1    | The transferred channel index is higher than the number of configured channels - 1  |
| ERR_CNC_NOT_INIT   | -3    | The simulation CNC kernel was not yet initialised.  |
| ERR_CNC_RET_MEMORY | -4    | The return value(s) does/do not fit into the memory provided. In this case, the number of actually required bytes is returned in returnLength |

### 3.23 kernelv\_ax\_get techno\_data()

#### Prototype

```
KERNELV_RETURN      kernelv_ax_get techno_data (unsigned long int axisIndex,
                                             KERNELV_TECHNO_DATA * technoData
                                             unsigned long int technoLength
                                             unsigned long int *returnLength);
```

#### Description

Returns the technology functions (M/H functions) acknowledged for the specified axis during the last call of kernelv\_do\_cycle().

#### Parameter

| Name         | Type                 | Meaning  |
|--------------|----------------------|--|
| axisIndex    | unsigned long        | Index of the axis.   |
| technoData   | KERNELV_TECHNO_DATA* | Pointer to the memory area to which the technology data is to be written.          |
| technoLength | unsigned long        | Size of the provided memory area in bytes.   |
| returnLength | unsigned long*       | Pointer to the memory area to which the actually returned bytes are to be written. |

#### Return values

| Symbol             | Value | Meaning   |
|--------------------|-------|---|
| RET_FINISHED       | 0     | The function was executed without error.  |
| ERR_CNC_NOT_INIT   | -3    | The simulation CNC kernel was not yet initialised.  |
| ERR_CNC_RET_MEMORY | -4    | The return value(s) does/do not fit into the memory provided. The number of bytes required is returned in the returnLength parameter. |
| ERR_INVALID_AX     | -9    | The transferred axis index is higher than the number of configured axes - 1   |



### 3.24 kernelv\_ax\_get\_new techno\_data()

**Prototype**

```
KERNELV_RETURN      kernelv_ax_get_new techno_data (unsigned long int axisIndex,
                                                    KERNELV_TECHNO_DATA * technoData
                                                    unsigned long int technoLength
                                                    unsigned long int *returnLength);
```

**Description**

Returns the technology functions (M/H functions) recently output during the last call of kernelv\_do\_cycle() for each specific axis.

**Parameter**

| Name         | Type                 | Meaning  |
|--------------|----------------------|--|
| axisIndex    | unsigned long        | Index of the axis.   |
| technoData   | KERNELV_TECHNO_DATA* | Pointer to the memory area to which the technology data is to be written.          |
| technoLength | unsigned long        | Size of the provided memory area in bytes.   |
| returnLength | unsigned long*       | Pointer to the memory area to which the actually returned bytes are to be written. |

**Return values**

| Symbol             | Value | Meaning   |
|--------------------|-------|---|
| RET_FINISHED       | 0     | The function was executed without error.  |
| ERR_CNC_NOT_INIT   | -3    | The simulation CNC kernel was not yet initialised.  |
| ERR_CNC_RET_MEMORY | -4    | The return value(s) does/do not fit into the memory provided. The number of bytes required is returned in the returnLength parameter. |
| ERR_INVALID_AX     | -9    | The transferred axis index is higher than the number of configured axes - 1   |

## 3.25 kernelv\_ax\_get techno\_data2()

### Prototype

```
KERNELV_RETURN kernelv_ax_get techno_data2 (unsigned long int axisIndex,
                                             KERNELV_TECHNO_DATA2 * technoData
                                             unsigned long int technoLength
                                             unsigned long int *returnLength);
```

### Description

Returns the technology functions (M/H functions) acknowledged for the specified axis during the last call of kernelv\_do\_cycle().

### Parameter

| Name         | Type                  | Meaning  |
|--------------|-----------------------|--|
| axisIndex    | unsigned long         | Index of the axis.   |
| technoData   | KERNELV_TECHNO_DATA2* | Pointer to the memory area to which the technology data is to be written.          |
| technoLength | unsigned long         | Size of the provided memory area in bytes.   |
| returnLength | unsigned long*        | Pointer to the memory area to which the actually returned bytes are to be written. |

### Return values

| Symbol             | Value | Meaning   |
|--------------------|-------|---|
| RET_FINISHED       | 0     | The function was executed without error.  |
| ERR_CNC_NOT_INIT   | -3    | The simulation CNC kernel was not yet initialised.  |
| ERR_CNC_RET_MEMORY | -4    | The return value(s) does/do not fit into the memory provided. The number of bytes required is returned in the returnLength parameter. |
| ERR_INVALID_AX     | -9    | The transferred axis index is higher than the number of configured axes - 1   |

## 3.26 kernelv\_ax\_get\_new techno\_data2()

### Prototype

```
KERNELV_RETURN kernelv_ax_get_new techno_data (unsigned long int axisIndex,
                                                KERNELV_TECHNO_DATA2 * technoData
                                                unsigned long int technoLength
                                                unsigned long int *returnLength);
```

### Description

Returns the technology functions (M/H functions) recently output during the last call of kernelv\_do\_cycle() for each specific axis.

### Parameter

| Name         | Type                  | Meaning  |
|--------------|-----------------------|--|
| axisIndex    | unsigned long         | Index of the axis.   |
| technoData   | KERNELV_TECHNO_DATA2* | Pointer to the memory area to which the technology data is to be written.          |
| technoLength | unsigned long         | Size of the provided memory area in bytes.   |
| returnLength | unsigned long*        | Pointer to the memory area to which the actually returned bytes are to be written. |

### Return values

| Symbol             | Value | Meaning   |
|--------------------|-------|---|
| RET_FINISHED       | 0     | The function was executed without error.  |
| ERR_CNC_NOT_INIT   | -3    | The simulation CNC kernel was not yet initialised.  |
| ERR_CNC_RET_MEMORY | -4    | The return value(s) does/do not fit into the memory provided. The number of bytes required is returned in the returnLength parameter. |
| ERR_INVALID_AX     | -9    | The transferred axis index is higher than the number of configured axes - 1   |

## 3.27 kernelv\_ax\_set\_position()

### Prototype

```
KERNELV_RETURN    kernelv_ax_set_position (unsigned long int axisIndex,
                                           signed long int position);
```

### Description

Sets the actual position of the axis to the position specified in the position parameter. This function can only be executed if no NC program is active in the channel to which the axis currently belongs. If an attempt is made to set the axis position while the NC program is active, adoption of the position is refused and the return value ERR\_INVALID\_STATE is returned.

### Parameter

| Name      | Type          | Meaning                                    |
|-----------|---------------|--|
| axisIndex | unsigned long | Index of the axis.                         |
| position  | unsigned long | New actual position of the axis in 0.1 µm. |

### Return values

| Symbol            | Value | Meaning   |
|-------------------|-------|---|
| RET_FINISHED      | 0     | The function was executed without error.  |
| RET_BUSY          | 1     | The function is currently being executed, but has not yet been completed.<br>The API function must continue to be called. |
| ERR_CNC_NOT_INIT  | -3    | The simulation CNC kernel has not yet been initialised.   |
| ERR_INVALID_STATE | -5    | The CNC channel of the axis is in the wrong state for executing a function.   |
| ERR_INVALID_AX    | -9    | The transferred axis index is higher than the number of configured axes -1 or zero.                                       |
| ERR_INVALID_AX    | -9    | The transferred axis index is higher than the number of configured axes - 1   |
| ERR_AXIS_ERROR    | -10   | The CNC axis indicates an error. The CNC additionally issues an error message.  |

## 3.28 kernelv\_get\_acs\_command\_positions()

### Prototype

```
KERNELV_RETURN    kernelv_get_acs_command_positions (unsigned long* positions,
                                                    unsigned long maxByteSize,
                                                    unsigned long* returnSize);
```

### Description

The ACS command positions of all axes existing in the CNC are returned in an array. If an axis-specific command value transformation is configured for an axis, this function returns the transformed command value for each specific axis.

Position values have a resolution of 0.1 µm for translatory axes or  $1 \cdot 10^{-4}^\circ$  for spindles or modulo axes.

The error code ERR\_CNC\_RET\_MEMORY is returned if the memory provided by the calling application is not sufficient for returning all values.

The calling application must provide at least number of axes \* sizeof(signed long int) bytes for returning of all position values.

The order of the axis positions in the returned array is equal to the configuration order of the axes.

### Parameter

| Name        | Type           | Meaning   |
|-------------|----------------|---|
| positions   | unsigned long* | Pointer to the memory area for the axis positions to be returned. |
| maxByteSize | unsigned long  | Size of the memory area for the axis positions.                   |
| returnSize  | unsigned long* | Number of bytes returned in positions.                            |

### Return values

| Symbol             | Value | Meaning   |
|--------------------|-------|---|
| RET_FINISHED       | 0     | The function was executed without error.  |
| ERR_CNC_NOT_INIT   | -3    | The simulation CNC kernel was not yet initialised.  |
| ERR_CNC_RET_MEMORY | -4    | The return value(s) does/do not fit into the memory provided. The number of bytes required is returned in the returnSize parameter. |

## 3.29 kernelv\_get\_acs0\_command\_positions()

### Prototype

```
KERNELV_RETURN    kernelv_get_acs0_command_positions (unsigned long* positions,
                                                    unsigned long maxByteSize,
                                                    unsigned long* returnSize);
```

### Description

The ACS command positions of all axes existing in the CNC are returned in an array. If an axis-specific command value transformation is configured for an axis, this function returns the untransformed command value for each specific axis.

Position values have a resolution of 0.1  $\mu\text{m}$  for translatory axes or  $1 \cdot 10^{-4}^\circ$  for spindles or modulo axes.

The error code ERR\_CNC\_RET\_MEMORY is returned if the memory provided by the calling application is not sufficient for returning all values.

The calling application must provide at least number of axes \* sizeof(signed long int) bytes for returning of all position values.

The order of the axis positions in the returned array is equal to the configuration order of the axes.

### Parameter

| Name        | Type           | Meaning   |
|-------------|----------------|---|
| positions   | unsigned long* | Pointer to the memory area for the axis positions to be returned. |
| maxByteSize | unsigned long  | Size of the memory area for the axis positions.                   |
| returnSize  | unsigned long* | Number of bytes returned in positions.                            |

### Return values

| Symbol             | Value | Meaning   |
|--------------------|-------|---|
| RET_FINISHED       | 0     | The function was executed without error.  |
| ERR_CNC_NOT_INIT   | -3    | The simulation CNC kernel was not yet initialised.  |
| ERR_CNC_RET_MEMORY | -4    | The return value(s) does/do not fit into the memory provided. The number of bytes required is returned in the returnSize parameter. |

### 3.30 kernelv\_get\_acs\_actual\_positions()

**Prototype**

```
KERNELV_RETURN kernelv_get_acs_actual_positions (unsigned long* positions,
                                                unsigned long maxByteSize,
                                                unsigned long* returnSize);
```

**Description**

The actual ACS positions of all axes existing in the CNC are returned in an array.

To calculate actual positions, a position control loop is simulated internally in the CNC and the mechanical behaviour of the axis is simulated by a PT2 element.

Position values have a resolution of 0.1 µm for translatory axes or 1\*10<sup>-4</sup>° for spindles or modulo axes.

The error code ERR\_CNC\_RET\_MEMORY is returned if the memory provided by the calling application is not sufficient for returning all values.

The calling application must provide at least number of axes \* sizeof(signed long int) bytes for returning of all position values.

The order of the axis positions in the returned array is equal to the configuration order of the axes.

**Parameter**

| Name        | Type           | Meaning   |
|-------------|----------------|---|
| positions   | unsigned long* | Pointer to the memory area for the axis positions to be returned. |
| maxByteSize | unsigned long  | Size of the memory area for the axis positions.                   |
| returnSize  | unsigned long* | Number of bytes returned in positions.                            |

**Return values**

| Symbol             | Value | Meaning   |
|--------------------|-------|---|
| RET_FINISHED       | 0     | The function was executed without error.  |
| ERR_CNC_NOT_INIT   | -3    | The simulation CNC kernel was not yet initialised.  |
| ERR_CNC_RET_MEMORY | -4    | The return value(s) does/do not fit into the memory provided. The number of bytes required is returned in the returnSize parameter. |

### 3.31 kernelv\_get\_acs\_target\_positions()

#### Prototype

```
KERNELV_RETURN kernelv_get_acs_target_positions (unsigned long* positions,
                                                unsigned long maxByteSize,
                                                unsigned long* returnSize);
```

#### Description

The ACS target positions of all axes existing in the CNC are returned in an array.

The target position is the axis position at the end of the currently executed motion block.

Position values have a resolution of 0.1  $\mu\text{m}$  for translatory axes or  $1 \cdot 10^{-4}^\circ$  for spindles or modulo axes.

The error code ERR\_CNC\_RET\_MEMORY is returned if the memory provided by the calling application is not sufficient for returning all values.

The calling application must provide at least number of axes \* sizeof(signed long int) bytes for returning of all position values.

The order of the axis positions in the returned array is equal to the configuration order of the axes.

#### Parameter

| Name        | Type           | Meaning   |
|-------------|----------------|---|
| positions   | unsigned long* | Pointer to the memory area for the axis positions to be returned. |
| maxByteSize | unsigned long  | Size of the memory area for the axis positions.                   |
| returnSize  | unsigned long* | Number of bytes returned in positions.                            |

#### Return values

| Symbol             | Value | Meaning  |
|--------------------|-------|--|
| RET_FINISHED       | 0     | The function was executed without error.   |
| ERR_CNC_NOT_INIT   | -3    | The simulation CNC kernel was not yet initialised.   |
| ERR_CNC_RET_MEMORY | -4    | The return value(s) does/do not fit into the memory provided. In this case, the required memory is returned in the returnSize parameter. |



### 3.32 kernelv\_get\_acs\_start\_positions()

**Prototype**

```
KERNELV_RETURN kernelv_get_acs_start_positions (unsigned long* positions,
                                                unsigned long maxByteSize,
                                                unsigned long* returnSize);
```

**Description**

The ACS start positions of all axes existing in the CNC are returned in an array.

The start position is the axis position that the axis was in at the start of the currently executed motion block.

Position values have a resolution of 0.1 µm for translatory axes or 1\*10<sup>-4</sup>° for spindles or modulo axes.

The error code ERR\_CNC\_RET\_MEMORY is returned if the memory provided by the calling application is not sufficient for returning all values.

The calling application must provide at least number of axes \* sizeof(signed long int) bytes for returning of all position values.

The order of the axis positions in the returned array is equal to the configuration order of the axes.

**Parameter**

| Name        | Type           | Meaning   |
|-------------|----------------|---|
| positions   | unsigned long* | Pointer to the memory area for the axis positions to be returned. |
| maxByteSize | unsigned long  | Size of the memory area for the axis positions.                   |
| returnSize  | unsigned long* | Number of bytes returned in positions.                            |

**Return values**

| Symbol             | Value | Meaning   |
|--------------------|-------|---|
| RET_FINISHED       | 0     | The function was executed without error.  |
| ERR_CNC_NOT_INIT   | -3    | The simulation CNC kernel was not yet initialised.  |
| ERR_CNC_RET_MEMORY | -4    | The return value(s) does/do not fit into the memory provided. The number of bytes required is returned in the returnSize parameter. |

### 3.33 kernelv\_get\_wcs\_command\_positions()

#### Prototype

```
KERNELV_RETURN      kernelv_get_wcs_command_positions (unsigned long* positions,
                                                    unsigned long maxByteSize,
                                                    unsigned long* returnSize);
```

#### Description

The WCS command positions of all axes existing in the CNC are returned in an array.

Position values have a resolution of 0.1  $\mu\text{m}$  for translatory axes or  $1 \cdot 10^{-4}^\circ$  for spindles or modulo axes.

The error code ERR\_CNC\_RET\_MEMORY is returned if the memory provided by the calling application is not sufficient for returning all values.

The calling application must provide at least number of axes \* sizeof(signed long int) bytes for returning of all position values.

The order of the axis positions in the returned array is equal to the configuration order of the axes.

#### Parameter

| Name        | Type           | Meaning   |
|-------------|----------------|---|
| positions   | unsigned long* | Pointer to the memory area for the axis positions to be returned. |
| maxByteSize | unsigned long  | Size of the memory area for the axis positions.                   |
| returnSize  | unsigned long* | Number of bytes returned in positions.                            |

#### Return values

| Symbol             | Value | Meaning  |
|--------------------|-------|--|
| RET_FINISHED       | 0     | The function was executed without error.   |
| ERR_CNC_NOT_INIT   | -3    | The simulation CNC kernel was not yet initialised.   |
| ERR_CNC_RET_MEMORY | -4    | The return value(s) does/do not fit into the memory provided. In this case, the required memory is returned in the returnSize parameter. |

### 3.34 kernelv\_get\_wcs\_target\_positions()

**Prototype**

```
KERNELV_RETURN kernelv_get_wcs_target_positions (unsigned long* positions,
                                                unsigned long maxByteSize,
                                                unsigned long* returnSize);
```

**Description**

The WCS target positions of all axes existing in the CNC are returned in an array.

The target position is the axis position at the end of the currently executed motion block.

Position values have a resolution of 0.1 µm for translatory axes or 1\*10<sup>-4</sup>° for spindles or modulo axes.

The error code ERR\_CNC\_RET\_MEMORY is returned if the memory provided by the calling application is not sufficient for returning all values.

The calling application must provide at least number of axes \* sizeof(signed long int) bytes for returning of all position values.

The order of the axis positions in the returned array is equal to the configuration order of the axes.

**Parameter**

| Name        | Type           | Meaning   |
|-------------|----------------|---|
| positions   | unsigned long* | Pointer to the memory area for the axis positions to be returned. |
| maxByteSize | unsigned long  | Size of the memory area for the axis positions.                   |
| returnSize  | unsigned long* | Number of bytes returned in positions.                            |

**Return values**

| Symbol             | Value | Meaning  |
|--------------------|-------|--|
| RET_FINISHED       | 0     | The function was executed without error.   |
| ERR_CNC_NOT_INIT   | -3    | The simulation CNC kernel was not yet initialised.   |
| ERR_CNC_RET_MEMORY | -4    | The return value(s) does/do not fit into the memory provided. In this case, the required memory is returned in the returnSize parameter. |

### 3.35 kernelv\_get\_wcs\_start\_positions()

#### Prototype

```
KERNELV_RETURN    kernelv_get_wcs_start_positions (unsigned long* positions,
                                                    unsigned long maxByteSize,
                                                    unsigned long* returnSize);
```

#### Description

The WCS start positions of all axes existing in the CNC are returned in an array.

The start position is the axis position that the axis was in at the start of the currently executed motion block.

Position values have a resolution of 0.1  $\mu\text{m}$  for translatory axes or  $1 \cdot 10^{-4}^\circ$  for spindles or modulo axes.

The error code ERR\_CNC\_RET\_MEMORY is returned if the memory provided by the calling application is not sufficient for returning all values.

The calling application must provide at least number of axes \* sizeof(signed long int) bytes for returning of all position values.

The order of the axis positions in the returned array is equal to the configuration order of the axes.

#### Parameter

| Name        | Type           | Meaning   |
|-------------|----------------|---|
| positions   | unsigned long* | Pointer to the memory area for the axis positions to be returned. |
| maxByteSize | unsigned long  | Size of the memory area for the axis positions.                   |
| returnSize  | unsigned long* | Number of bytes returned in positions.                            |

#### Return values

| Symbol             | Value | Meaning  |
|--------------------|-------|--|
| RET_FINISHED       | 0     | The function was executed without error.   |
| ERR_CNC_NOT_INIT   | -3    | The simulation CNC kernel was not yet initialised.   |
| ERR_CNC_RET_MEMORY | -4    | The return value(s) does/do not fit into the memory provided. In this case, the required memory is returned in the returnSize parameter. |

### 3.36 kernelv\_get\_prg\_target\_positions()

**Prototype**

```
KERNELV_RETURN    kernelv_get_prg_target_positions (signed long int *positions,
                                                    unsigned long int maxByteSize,
                                                    unsigned long int *returnSize);
```

**Description**

The target positions of all axes existing in the CNC that are programmed in the NC program are returned in an array.

The returned positions are the positions programmed in the currently executed NC block.

If motion blocks are inserted by the CNC or programmed motion blocks are split (e.g. when contouring), the target position of the source block is output for all generated NC blocks.

Position values have a resolution of 0.1 µm for translatory axes or 1\*10<sup>-4°</sup> for spindles or rotary axes.

The error code ERR\_CNC\_RET\_MEMORY is returned if the memory provided by the calling application is not sufficient for returning all values.

The calling application must provide at least number of axes \* sizeof(signed long int) bytes for returning of all position values.

The order of the axis positions in the returned array is equal to the configuration order of the axes.

If an axis is currently not assigned to an NC channel, the value zero is returned for that axis.

**Parameter**

| Name        | Type           | Meaning   |
|-------------|----------------|---|
| positions   | unsigned long* | Pointer to the memory area for the axis positions to be returned. |
| maxByteSize | unsigned long  | Size of the memory area for the axis positions.                   |
| returnSize  | unsigned long* | Number of bytes returned in positions.                            |

**Return values**

| Symbol             | Value | Meaning  |
|--------------------|-------|--|
| RET_FINISHED       | 0     | The function was executed without error.   |
| ERR_CNC_NOT_INIT   | -3    | The simulation CNC kernel was not yet initialised.   |
| ERR_CNC_RET_MEMORY | -4    | The return value(s) does/do not fit into the memory provided. In this case, the required memory is returned in the returnSize parameter. |

### 3.37 kernelv\_get\_axis\_channel\_number()

#### Prototype

```
KERNELV_RETURN kernelv_get_axis_channel_number (unsigned short* chanNumbers,
                                                unsigned long maxByteSize,
                                                unsigned long* returnSize);
```

#### Description

By configuration or by axis replacement commands during an NC program, an axis can be moved by different CNC channels. By means of this function, the number of the channel moving the respective axis can be queried.

The following relationship exists between the channel number and the channel index:

Channel index = channel number – 1

#### Parameter

| Name        | Type            | Meaning  |
|-------------|-----------------|--|
| chanNumbers | unsigned short* | Pointer to the memory area for the channel numbers to be returned. |
| maxByteSize | unsigned long   | Size of the memory area for the axis positions.                    |
| returnSize  | unsigned long*  | Number of bytes returned in positions.                             |

#### Return values

| Symbol             | Value | Meaning  |
|--------------------|-------|--|
| RET_FINISHED       | 0     | The function was executed without error.   |
| ERR_CNC_NOT_INIT   | -3    | The simulation CNC kernel was not yet initialised.   |
| ERR_CNC_RET_MEMORY | -4    | The return value(s) does/do not fit into the memory provided. In this case, the required memory is returned in the returnSize parameter. |

### 3.38 kernelv\_ch\_get\_variable\_value()

**Prototype**

```
KERNELV_RETURN kernelv_ch_get_variable_value (unsigned long chanIndex,
char* varName,
KERNELV_VARIABLE *var);
```

**Description**

Reads a variable out of the CNC kernel. Currently, the following variable types can be read.

| Type                                       | Example         |
|--|-----------------|
| External variables                         | V.E.VAR_1       |
| Global variables, see (1)                  | V.G.CNC_RELEASE |
| Axis-specific variables                    | V.A.+SWE.X      |
| Program global, self-defined variables     | V.P.VAR_1       |
| Cross-program, self-defined variables      | V.S.VAR_1       |
| Program non-global, self-defined variables | V.L.VAR_1       |

Currently, it is not possible to read Type V.G.WZ[] variables.

The variable to be read is identified on the basis of its name and the channel index.

The complete name (including the V.E. prefix and the array index in the case of array variables) must be specified as the name.

Example: "V.E.VAR\_FLOAT\_ARRAY[3]"

The value is returned in the KERNELV\_VARIABLE \*var structure.

An error code is returned if an error occurred while reading the variables.

**Parameter**

| Name      | Type              | Meaning  |
|-----------|-------------------|--|
| chanIndex | unsigned long     | Channel index of the channel from which the variable is to be read.              |
| varName   | char*             | Pointer to the variable name.  |
| var       | KERNELV_VARIABLE* | Pointer to structure to which the variable value and the type are to be written. |

**Return values**

| Symbol               | Value | Meaning   |
|----------------------|-------|---|
| RET_FINISHED         | 0     | The function was executed without error.  |
| ERR_INVALID_CHAN     | -1    | The transferred channel index is higher than the number of configured channels - 1                              |
| ERR_CNC_NOT_INIT     | -3    | The simulation CNC kernel was not yet initialised.  |
| ERR_CNC_RET_MEMORY   | -4    | The return value(s) does/do not fit into the memory provided.   |
| ERR_INTERNAL_ERROR   | -11   | An internal error has occurred in the DLL.  |
| ERR_UNKNOWN_VARIABLE | -12   | The variable name is not known in the CNC kernel.   |
| ERR_VARIABLE_SYNTAX  | -13   | The variable name is syntactically incorrect, e.g. no closing bracket in the case of an array variable.         |
| ERR_VAR_NAME_LENGTH  | -18   | The variable name transferred to the function exceeds the maximum permissible length (KERNELV_VAR_NAME_LENGTH). |

|                         |     |   |
|-------------------------|-----|---|
| ERR_ARRAY_NOT_SUPPORTED | -21 | With many CNC real-time variants, an array can be read 'en block' by omitting the array index on access. The kernelv DLL does not currently support this access type. |
|-------------------------|-----|---|



### 3.39 kernelv\_ch\_set\_variable\_value()

**Prototype**

```
KERNELV_RETURN kernelv_ch_set_variable_value (unsigned long int chanIndex,
                                              char* varName,
                                              KERNELV_VARIABLE *var);
```

**Description**

Writes variables defined for a CNC kernel.

Writing a variable is only possible if it is writeable by the NC program. External variables can always be written irrespective of the access type configured.

Currently, the following variable types can be written.

| Type                                       | Example      |
|--|--------------|
| External variables                         | V.E.VAR_1    |
| Global variables, see (1)                  | V.G.WZ_AKT.R |
| Axis-specific variables                    | V.A.WCS.X    |
| Program global, self-defined variables     | V.P.VAR_1    |
| Cross-program, self-defined variables      | V.S.VAR_1    |
| Program non-global, self-defined variables | V.L.VAR_1    |

Currently, it is not possible to read Type V.G.WZ[] variables.

The variable to be written is identified on the basis of its name and the channel index.

The complete name (including the V.E. prefix and the array index in the case of array variables) must be specified as the name.

Example: "V.E.VAR\_FLOAT\_ARRAY[3]"

The variable value to be written must be transferred in the KERNELV\_VARIABLE \*var structure.

An error code is returned if an error occurred while writing the variables.

**Parameter**

| Name      | Type              | Meaning  |
|-----------|-------------------|--|
| chanIndex | unsigned long     | Channel index of the channel from which the variable is to be read.              |
| varName   | char*             | Pointer to the variable name.  |
| var       | KERNELV_VARIABLE* | Pointer to structure to which the variable value and the type are to be written. |

**Return values**

| Symbol               | Value | Meaning   |
|----------------------|-------|---|
| RET_FINISHED         | 0     | The function was executed without error.  |
| ERR_INVALID_CHAN     | -1    | The transferred channel index is higher than the number of configured channels - 1                      |
| ERR_CNC_NOT_INIT     | -3    | The simulation CNC kernel was not yet initialised.  |
| ERR_CNC_RET_MEMORY   | -4    | The return value(s) does/do not fit into the memory provided.   |
| ERR_INTERNAL_ERROR   | -11   | An internal error has occurred in the DLL.  |
| ERR_UNKNOWN_VARIABLE | -12   | The variable name is not known in the CNC kernel.   |
| ERR_VARIABLE_SYNTAX  | -13   | The variable name is syntactically incorrect, e.g. no closing bracket in the case of an array variable. |

|                         |     |  |
|-------------------------|-----|--|
| ERR_DATA_TYPE_MISMATCH  | -14 | On write access to a variable, the transferred data type does not match the data type used internally in the CNC.  |
| ERR_VAR_NAME_LENGTH     | -18 | The variable name passed on to the function exceeds the maximum permissible length (KERNELV_VAR_NAME_LENGTH).  |
| ERR_ARRAY_NOT_SUPPORTED | -21 | With many CNC real-time variants, an array can be read or written 'en block' by omitting the array index at access. The kernelv DLL does not currently support this access type.   |
| ERR_VAR_NOT_WRITEABLE   | -22 | An attempt was made to write a non-writeable variant. For write access to variables, the same access rules apply as within an NC program. The only exception here are V.E. variables. They can always be written irrespective of the access rights configured. |

## 3.40 kernelv\_get\_channel\_count()

### Prototype

```
KERNELV_RETURN kernelv_get_channel_count (unsigned long int* channelCount);
```

### Description

Reads the number of configured channels of the CNC control.

An error code is returned if the function is called while the CNC kernel has not yet started.

### Parameter

| Name         | Type           | Meaning  |
|--------------|----------------|--|
| channelCount | unsigned long* | Pointer to the memory for the number of channels to be returned. |

### Return values

| Symbol           | Value | Meaning  |
|------------------|-------|--|
| RET_FINISHED     | 0     | The function was executed without error.           |
| ERR_CNC_NOT_INIT | -3    | The simulation CNC kernel was not yet initialised. |

## 3.41 kernelv\_get\_axis\_count()

### Prototype

```
KERNELV_RETURN kernelv_get_axis_count (unsigned long int* axisCount);
```

### Description

Reads the number of configured axes of the CNC controller.

An error code is returned if the function is called while the CNC kernel has not yet started.

### Parameter

| Name      | Type           | Meaning  |
|-----------|----------------|--|
| axisCount | unsigned long* | Pointer to the memory for the number of axes to be returned. |

### Return values

| Symbol           | Value | Meaning  |
|------------------|-------|--|
| RET_FINISHED     | 0     | The function was executed without error.           |
| ERR_CNC_NOT_INIT | -3    | The simulation CNC kernel was not yet initialised. |

### 3.42 kernelv\_sync\_read\_request()

**Prototype**

```
signed long int    kernelv_sync_read_request (unsigned short port,
                                             unsigned long int indexGroup,
                                             unsigned long int indexOffset,
                                             unsigned long int length,
                                             void* data);
```

**Description**

Synchronous reading of variables. The variables are identified by Port, indexGroup and indexOffset.

At present, the following ports are supported:

|             |          |
|-------------|----------|
| Port number | CNC task |
| 551         | GEO task |
| 552         | SDA task |
| 553         | COM task |

The meaning of indexGroup and indexOffset depends on the addressed port and can be found in the documentation.

An error code is returned if the function is called while the CNC kernel has not yet started.

**Parameter**

| Name        | Type           | Meaning   |
|-------------|----------------|---|
| port        | unsigned short | Port from which the data is to be read.               |
| indexGroup  | unsigned long* | Index group of the data item to be read               |
| indexOffset | unsigned long* | Index offset of the data item to be read              |
| length      | unsigned long* | Size of the memory in bytes for the value to be read. |
| data        | void*          | Pointer to memory for the value to be read            |

**Return values**

| Symbol | Value | Meaning  |
|--------|-------|--|
|        | 0     | The function was executed without error.                       |
|        | -3    | The simulation CNC kernel was not yet initialised.             |
|        | 6     | The transferred port number is not known.                      |
|        | 0x701 | The requested service is not supported.                        |
|        | 0x702 | Index group is invalid   |
|        | 0x703 | Index offset is invalid  |
|        | 0x704 | The variable addressed must not be read.                       |
|        | 0x705 | The memory provided is too small for the value to be returned. |

### 3.43 kernelv\_sync\_write\_request()

#### Prototype

```
signed long int      kernelv_sync_write_request (unsigned short port,
                                                unsigned long int indexGroup,
                                                unsigned long int indexOffset,
                                                unsigned long int length,
                                                void* data);
```

#### Description

Synchronous writing of variables. The variables are identified by Port, indexGroup and indexOffset.

At present, the following ports are supported:

|             |          |
|-------------|----------|
| Port number | CNC task |
| 551         | GEO task |
| 552         | SDA task |
| 553         | COM task |

The meaning of indexGroup and indexOffset depends on the addressed port and can be found in the documentation.

An error code is returned if the function is called while the CNC kernel has not yet started.

A write operation may last several NC cycles irrespective of the index group and index offset used. In this case the function returns the value 1 (RET\_BUSY).

If this is the case, the function must be called again every time kernelv\_do\_cycle() is called until either the return value returned is 0 (RET\_FINISHED) or an error message is returned.

#### Parameter

| Name        | Type           | Meaning  |
|-------------|----------------|--|
| port        | unsigned short | Port via which the data is to be written.                |
| indexGroup  | unsigned long  | Index group of the datum to be written.                  |
| indexOffset | unsigned long  | Index offset of the datum to be written.                 |
| length      | unsigned long* | Size of the memory in bytes for the value to be written. |
| data        | void*          | Pointer to memory for the value to be written            |

#### Return values

| Symbol       | Value | Meaning   |
|--------------|-------|---|
| RET_FINISHED | 0     | The function was executed without error.  |
| RET_BUSY     | 1     | The function is currently being executed, but has not yet been completed.<br>The API function must continue to be called. |
|              | 6     | The transferred port number is not known.   |
|              | 0x701 | The requested service is not supported.   |
|              | 0x702 | Index group is invalid  |
|              | 0x703 | Index offset is invalid   |
|              | 0x704 | The variable addressed must not be written.   |
|              | 0x705 | The data to be written does not match the addressed variable.   |

### 3.44 kernelv\_sync\_read\_write\_req()

**Prototype**

```
signed long int kernelv_sync_write_req (unsigned short port,
                                       unsigned long int indexGroup,
                                       unsigned long int indexOffset,
                                       unsigned long int *readLength,
                                       unsigned long int writeLength,
                                       void* data);
```

**Description**

Synchronous writing and reading of variables. The variables are identified by Port, indexGroup and indexOffset.

When called, the data pointer contains the data to be written. Specify in writeLength the length of the data area to be written in bytes. Enter the number of bytes to be read in the readLength pointer.

The data read is saved to the memory to which the data pointer points and the number of bytes written is written to \*readLength.

At present, the following ports are supported:

|             |          |
|-------------|----------|
| Port number | CNC task |
| 551         | GEO task |
| 552         | SDA task |

The meaning of indexGroup and indexOffset depends on the addressed port and can be found in the documentation.

An error code is returned if the function is called while the CNC kernel has not yet started.

**Parameter**

| Name        | Type           | Meaning   |
|-------------|----------------|---|
| port        | unsigned short | Port from which the data is to be read.                       |
| indexGroup  | unsigned long  | Index group of the data item to be read.                      |
| indexOffset | unsigned long  | Index offset of the data item to be read.                     |
| readLength  | unsigned long* | Pointer to the return value for the number of bytes returned. |
| writeLength | unsigned long  | Length of the data area to be written.                        |
| data        | void*          | Pointer to memory for the data to be read or written.         |

**Return values**

| Symbol             | Value | Meaning   |
|--------------------|-------|---|
|                    | 0     | The function was executed without error.                      |
|                    | -3    | The simulation CNC kernel was not yet initialised.            |
| ERR_CNC_RET_MEMORY | -4    | The return value(s) does/do not fit into the memory provided. |
|                    | 6     | The transferred port number is not known.                     |
|                    | 0x701 | The requested service is not supported.                       |
|                    | 0x702 | Index group is invalid  |
|                    | 0x703 | Index offset is invalid                                       |
|                    | 0x704 | The variable addressed must not be written.                   |
|                    | 0x705 | The data to be written does not match the addressed variable. |

## 3.45 kernelv\_get\_axis\_names()

### Prototype

```
KERNELV_RETURN kernelv_get_axis_names(char* axisNames,
                                     unsigned long int maxByteSize,
                                     unsigned long int *returnSize,
                                     unsigned long int *nAxis);
```

### Description

Returns the axis names of an axis used in the respective channels. The axis name is the name with which the axis is addressed in an NC program. Only axes that are in a channel have an axis name.

An array of strings with a fixed length is returned. The length of one single string (including terminating zeroes) can be determined as follows:

Length of name = returnSize / number of configured axes

The number of configured axes can be determined with the kernelv\_get\_axis\_count() function.

The indexes of the returned axis names correspond to the configuration sequence. See also Section 1.2.3.1.

If the axis name is shorter than the name length -1 character, the string is filled with the value \0.

The indexes of the returned axis names correspond to the configuration sequence. See also Section 1.2.3.1.

If the memory provided by the calling application is too small for the values to be returned, the value ERR\_CNC\_RET\_MEMORY is returned and, in this case, the memory in bytes minimally required for returning is entered in the returnSize variable.

### Parameter

| Name        | Type           | Meaning   |
|-------------|----------------|---|
| axisNames   | char*          | Pointer to the memory area for the axis names to be returned. |
| maxByteSize | unsigned long  | Size of the memory area for the axis names.                   |
| returnSize  | unsigned long* | Number of bytes returned in axisNames.                        |
| nAxis       | unsigned long* | Number of axis names returned in axisNames.                   |

### Return values

| Symbol             | Value | Meaning   |
|--------------------|-------|---|
| RET_FINISHED       | 0     | The function was executed without error.  |
| ERR_CNC_NOT_INIT   | -3    | The simulation CNC kernel was not yet initialised.  |
| ERR_CNC_RET_MEMORY | -4    | The return value(s) does/do not fit into the memory provided. The number of bytes required is returned in the returnSize parameter. |

Four axes are configured, and only the axes with the indexes 0 (axis name 'X2') and 2 (axis name 'C') are assigned to a channel: A call of kernelv\_get\_axis\_names() returns the following values:

| Parameter  | Value | Comment   |
|------------|-------|---|
| returnSize | 80    | Configured axes * name length                           |
| nAxis      | 2     | Number of axes that are currently assigned to a channel |
| axisNames  |       | See the table below.                                    |

The length (nameLength) of a single string is thus:

namelength = returnSize / number of configured axes = 80 / 4 = 20



axisNames as the following structure:

| Offset             | Value                     |
|--------------------|---------------------------|
| 0                  | 'X', '2'; \0, \0, \0 .... |
| 20 (1* nameLength) | \0, \0, \0, .....         |
| 40 (2* nameLength) | 'C'; \0, \0, \0 ....      |
| 60 (3* nameLength) | \0, \0, \0, .....         |

## 3.46 kernelv\_control techno\_func\_duration()

### Prototype

KERNELV\_RETURN kernelv\_control techno\_func\_duration (unsigned char onOff);

### Description

Activates or deactivates execution time simulation for technology functions.

By default, execution time simulation is off, i.e. technology functions are acknowledged immediately. When execution time simulation is active, acknowledgement of the technology functions is delayed to simulate their actual execution time.

The execution time can be set by means of the kernelv\_ch\_set techno\_func\_duration() function or by means of entries in the channel parameter list (P-CHAN-00040, P-CHAN-00026).

If execution time simulation is deactivated while still unacknowledged technology functions are pending, they are acknowledged immediately regardless of the set execution time.

### Parameter

| Name  | Type          | Meaning  |
|-------|---------------|--|
| onOff | unsigned char | In the case of a value of onOff > 0, the execution time is activated and a value of 0 deactivates execution time simulation. |

### Return values

| Symbol       | Value | Meaning                                  |
|--------------|-------|--|
| RET_FINISHED | 0     | The function was executed without error. |

### 3.47 kernelv\_ch\_set techno\_func\_duration()

**Prototype**

KERNELV\_RETURN kernelv\_ch\_set techno\_func\_duration (unsigned long int chanIndex, E\_KERNELV\_TECHNO\_TYPE type, unsigned long int number, unsigned long int time\_us);

**Description**

Sets the execution time for the transferred technology function.

The execution time must be specified in microseconds.

**Parameter**

| Name      | Type                  | Meaning  |
|-----------|-----------------------|--|
| chanIndex | unsigned long int     | Channel index.   |
| type      | E_KERNELV_TECHNO_TYPE | Type of the technology function whose execution time is to be written.   |
| number    | unsigned long int     | Number of the technology function whose execution time is to be written. |
| time_us   | unsigned long int     | Execution time in us.  |

**Return values**

| Symbol                   | Value | Meaning   |
|--------------------------|-------|---|
| RET_FINISHED             | 0     | The function was executed without error.  |
| ERR_INVALID_CHAN         | -1    | The transferred channel index is higher than the number of configured channels - 1  |
| ERR_INTERNAL_ERROR       | -11   | An internal error has occurred in the DLL. The value could not be written.  |
| ERR_INVALID_TECHNO_PARAM | -16   | An invalid parameter was transferred when the execution time was set for a technology function, e.g. transferred number of the M or H function is higher than the maximum permitted number. |
| ERR_UNKNOWN_TECHNO_TYPE  | -15   | An invalid type was specified for a technology function when the execution time was set.  |

## 3.48 kernelv\_ch\_set techno\_func\_user\_ackn()

### Prototype

```
KERNELV_RETURN kernelv_ch_set techno_func_user_ackn (unsigned long int chanIndex,
                                                    E_KERNELV_TECHNO_TYPE type,
                                                    unsigned long int number);
```

### Description

Deactivates automatic acknowledgement when execution time simulation is active after expiry of the execution time set. The user must acknowledge the technology function by calling one of the functions `kernelv_ch_ackn techno_func()` or `kernelv_ax_ackn techno_func()`.

User acknowledgement is only possible if execution time simulation is active.

### Parameter

| Name      | Type                  | Meaning   |
|-----------|-----------------------|---|
| chanIndex | unsigned long int     | Channel index.  |
| type      | E_KERNELV_TECHNO_TYPE | Type of technology function to be acknowledged by the user. |
| number    | unsigned long int     | Number of the technology function.                          |

### Return values

| Symbol                   | Value | Meaning   |
|--------------------------|-------|---|
| RET_FINISHED             | 0     | The function was executed without error.  |
| ERR_INVALID_CHAN         | -1    | The transferred channel index is higher than the number of configured channels -1   |
| ERR_INTERNAL_ERROR       | -11   | An internal error has occurred in the DLL. The value could not be written.  |
| ERR_INVALID_TECHNO_PARAM | -16   | An invalid parameter was transferred when user acknowledgement was activated, e.g. transferred number of the M or H function is higher than the maximum permitted number. |
| ERR_UNKNOWN_TECHNO_TYPE  | -15   | An invalid type was specified for a technology function when the user acknowledgement was activated.  |

### 3.49 kernelv\_ch\_ackn techno\_func()

**Prototype**

```
KERNELV_RETURN kernelv_ch_ackn techno_func(unsigned long int chanIndex,
                                             E_KERNELV_TECHNO_TYPE type,
                                             unsigned long int number);
```

**Description**

Acknowledges a technology function for which the user acknowledgement was activated.

If the specified technology function is not open (acknowledgement pending), the return value ERR\_TECHNO\_NOT\_FOUND is returned, otherwise RET\_FINISHED.

**Parameter**

| Name      | Type                  | Meaning   |
|-----------|-----------------------|---|
| chanIndex | unsigned long int     | Channel index.                                  |
| type      | E_KERNELV_TECHNO_TYPE | Type of technology function to be acknowledged. |
| number    | unsigned long int     | Number of the technology function.              |

**Return values**

| Symbol               | Value | Meaning   |
|----------------------|-------|---|
| RET_FINISHED         | 0     | The technology function was acknowledged.   |
| ERR_INVALID_CHAN     | -1    | The transferred channel index is higher than the number of configured channels -1 |
| ERR_CNC_NOT_INIT     | -3    | The simulation CNC kernel was not yet initialised.                                |
| ERR_TECHNO_NOT_FOUND | -29   | The specified technology function was not found.                                  |

## 3.50 kernelv\_ax\_ackn techno\_func()

### Prototype

```
KERNELV_RETURN kernelv_ax_ackn techno_func(unsigned long int axIndex,
                                             E_KERNELV_TECHNO_TYPE type,
                                             unsigned long int number);
```

### Description

Acknowledges a technology function for which the user acknowledgement was activated.

If the specified technology function is not open (acknowledgement pending), the return value ERR\_TECHNO\_NOT\_FOUND is returned, otherwise RET\_FINISHED.

### Parameter

| Name   | Type                  | Meaning   |
|--------|-----------------------|---|
| axdex  | unsigned long int     | Axis index.                                     |
| type   | E_KERNELV_TECHNO_TYPE | Type of technology function to be acknowledged. |
| number | unsigned long int     | Number of the technology function.              |

### Return values

| Symbol               | Value | Meaning  |
|----------------------|-------|--|
| RET_FINISHED         | 0     | The technology function was acknowledged.                                  |
| ERR_INVALID_CHAN     | -1    | The transferred axis index is higher than the number of configured axes -1 |
| ERR_CNC_NOT_INIT     | -3    | The simulation CNC kernel was not yet initialised.                         |
| ERR_TECHNO_NOT_FOUND | -29   | The specified technology function was not found.                           |

## 3.51 kernelv\_get\_license\_info()

### Prototype

```
KERNELV_RETURN kernelv_get_license_info (KERNELV_LICENSE_INFO *pLicenseInfo);
```

### Description

Reads the existing license information.

The function can only be used after the function `kernelv_startup()` is called.

The function writes the existing license information to the memory referenced by the call parameter `pLicenseInfo`.

### Parameter

| Name                      | Type                               | Meaning  |
|---------------------------|------------------------------------|--|
| <code>pLicenseInfo</code> | <code>KERNELV_LICENSE_INFO*</code> | Memory for the license information to be returned. |

### Return values

| Symbol                        | Value | Meaning  |
|-------------------------------|-------|--|
| <code>RET_FINISHED</code>     | 0     | The function was executed without error.           |
| <code>ERR_CNC_NOT_INIT</code> | -3    | The simulation CNC kernel was not yet initialised. |

## 3.52 kernelv\_set\_options()

### Prototype

KERNELV\_RETURN kernelv\_set\_options (unsigned long int optionsMask);

### Description

This function activates internal kernelv functions, e.g. the output of messages during license check.

Depending on the function to be activated, this function may have to be called by the kernelv\_startup() function.

Each function is assigned a bit in the call parameter of this function. The function is activated by setting the appropriate bit.

If several functions are to be activated, the resulting bit array is generated from the OR operation of the individual bit masks.

If bits are set in the bit array transferred to the function and they are not assigned to any function, the unknown bits are ignored and the value ERR\_UNKNOWN\_OPTION is returned. In this case, the known bits are evaluated and the assigned functions are activated.

### Parameter

| Name        | Type          | Meaning                 |
|-------------|---------------|-------------------------|
| optionsMask | unsigned long | Bit array with options. |

Possible values for the bit array are:

| Symbol                               | Value | Meaning  |
|--------------------------------------|-------|--|
| KERNELV_OPTION_LICENSE_CHECK_VERBOSE | 0x1   | During the license check at start-up, progress messages are output relating to the license check. Acts as error search to handle licensing problems. |

### Return values

| Symbol           | Value | Meaning  |
|------------------|-------|--|
| RET_FINISHED     | 0     | The function was executed without error.           |
| ERR_CNC_NOT_INIT | -3    | The simulation CNC kernel was not yet initialised. |



### 3.53 kernelv\_ch\_get\_decoder\_positions()

**Prototype**

```
KERNELV_RETURN kernelv_ch_get_decoder_positions (unsigned long int chanIndex,
                                                void* ret_buffer,
                                                unsigned long int buffer_size,
                                                unsigned long int *ret_length));
```

**Description**

This function requests the decoder for the currently active axis positions in the machine coordinates and in the programming coordinates.

The values returned by the function are written to the memory area that points to the buffer. The size of this memory area must be specified in the buffer\_size variable. The actual size of the returned data is returned in ret\_length. If the returned data does not fit in the memory area provided, the error code ERR\_CNC\_RET\_MEMORY is returned. In this case ret\_length contains the memory size required for the return.

The returned data has the following structure. The individual data elements of the structures are packed in the memory:

| Byte index | Type                            | Meaning  |
|------------|---------------------------------|--|
| 0          | KERNELV_DECODER_POSITION_HEADER | For general data about the decoder positions supplied, see 2.2.24 [▶ 139]. Enter the number of the following structures of type KERNELV_DECODER_POSITION_DATA in the axis_count element. |
| 11         | KERNELV_DECODER_POSITION_DATA   | Axis positions and axis number of the 1st axis. For structure definition, see 2.2.25 [▶ 140]   |
| 25         | KERNELV_DECODER_POSITION_DATA   | Axis positions and axis number of the 2nd axis. For structure definition, see 2.2.25 [▶ 140]   |
| 39         | ...                             | ...  |

Since the maximum possible number of axes in the channel is currently limited to 32, the maximum size of the return value is 459 bytes.

**Parameter**

| Name        | Type           | Meaning  |
|-------------|----------------|--|
| chanIndex   | unsigned long  | Index of the channel   |
| ret_buffer  | void*          | Pointer to the memory area to which the return data is to be written.              |
| buffer_size | unsigned long  | Size of the provided memory area in bytes.   |
| ret_Length  | unsigned long* | Pointer to the memory area to which the actually returned bytes are to be written. |

**Return values**

| Symbol             | Value | Meaning   |
|--------------------|-------|---|
| RET_FINISHED       | 0     | The function was executed without error.  |
| ERR_CNC_NOT_INIT   | -3    | The simulation CNC kernel was not yet initialised.                                |
| ERR_INVALID_CHAN   | -1    | The transferred channel index is higher than the number of configured channels -1 |
| ERR_CNC_RET_MEMORY | -4    | The return value(s) does/do not fit into the memory provided.                     |

### 3.54 kernelv\_ch\_get\_prog\_start\_mode()

#### Prototype

```
KERNELV_RETURN kernelv_ch_get_prog_start_mode (unsigned long chanIndex,
                                              E_KERNELV_PROG_START_MODE* mode);
```

#### Description

Reads the execution mode of the current program running.

#### Parameter

| Name      | Type                       | Meaning   |
|-----------|----------------------------|---|
| chanIndex | unsigned long              | Channel index of the channel from which the variable is to be read. |
| mode      | E_KERNELV_PROG_START_MODE* | Points to Enum for the returned execution mode.                     |

#### Return values

| Symbol           | Value | Meaning   |
|------------------|-------|---|
| RET_FINISHED     | 0     | The function was executed without error.  |
| ERR_INVALID_CHAN | -1    | The transferred channel index is higher than the number of configured channels -1 |
| ERR_CNC_NOT_INIT | -3    | The simulation CNC kernel was not yet initialised.                                |

## 3.55 kernelv\_ch\_set\_cont\_visu\_grid()

### Prototype

```
KERNELV_RETURN kernelv_ch_set_cont_visu_grid (unsigned long int chanIndex,  
                                              unsigned long int grid);
```

### Description

Sets the output grid for linear blocks for command contour visualisation.

### Parameter

| Name      | Type          | Meaning   |
|-----------|---------------|---|
| chanIndex | unsigned long | Channel index of the channel from which the variable is to be read. |
| grid      | unsigned long | Output grid for linear blocks in 0.1 um.                            |

### Return values

| Symbol           | Value | Meaning   |
|------------------|-------|---|
| RET_FINISHED     | 0     | The function was executed without error.  |
| ERR_INVALID_CHAN | -1    | The transferred channel index is higher than the number of configured channels -1 |
| ERR_CNC_NOT_INIT | -3    | The simulation CNC kernel was not yet initialised.                                |

## 3.56 kernelv\_ch\_set\_cont\_visu\_rel\_curvature\_error()

### Prototype

```
KERNELV_RETURN kernelv_ch_set_cont_visu_rel_curvature_error
                (unsigned long int chanIndex,
                 unsigned long int rel_error);
```

### Description

Sets the relative curvature error of the command contour visualisation for curved contours (circle segments and polynomials).

The relative curvature error indicates the permitted secant error for tracing the curved contour as a percentage of the curve radius.

Example: For a circle, the curve radius is identical to the circle radius; for a circle radius of 100 mm and a relative curvature error of 1%, the permitted secant error is  $100 \text{ mm} * 1\% = 1 \text{ mm}$ .

The effective secant error to trace a contour element is determined by the minimum value between the absolute and relative secant errors.

### Parameter

| Name      | Type          | Meaning   |
|-----------|---------------|---|
| chanIndex | unsigned long | Channel index of the channel from which the variable is to be read. |
| rel_error | unsigned long | Relative curvature error in 0.1%                                    |

### Return values

| Symbol           | Value | Meaning   |
|------------------|-------|---|
| RET_FINISHED     | 0     | The function was executed without error.  |
| ERR_INVALID_CHAN | -1    | The transferred channel index is higher than the number of configured channels -1 |
| ERR_CNC_NOT_INIT | -3    | The simulation CNC kernel was not yet initialised.                                |

## 3.57 kernelv\_ch\_set\_cont\_visu\_abs\_curvature\_error()

### Prototype

```
KERNELV_RETURN kernelv_ch_set_cont_visu_abs_curvature_error
(unsigned long int chanIndex,
unsigned long int rel_error);
```

### Description

Sets the absolute curvature error of the command contour visualisation for curved contours (circle segments and polynomials).

The absolute curvature error indicates the permitted secant error for tracing the curved contour.

The effective secant error to trace a contour element is determined by the minimum value between the absolute and relative secant errors.

### Parameter

| Name      | Type          | Meaning   |
|-----------|---------------|---|
| chanIndex | unsigned long | Channel index of the channel from which the variable is to be read. |
| rel_error | unsigned long | Relative curvature error in 0.1%                                    |

### Return values

| Symbol           | Value | Meaning   |
|------------------|-------|---|
| RET_FINISHED     | 0     | The function was executed without error.  |
| ERR_INVALID_CHAN | -1    | The transferred channel index is higher than the number of configured channels -1 |
| ERR_CNC_NOT_INIT | -3    | The simulation CNC kernel was not yet initialised.                                |

## 3.58 kernelv\_ch\_get\_cont\_visu\_data()

### Prototype

```
KERNELV_RETURN kernelv_ch_get_cont_visu_data (unsigned long int chanIndex,
                                              unsigned char* pData,
                                              unsigned long int maxByteSize,
                                              unsigned long int* retLength);
```

### Description

Reads the contour visualisation of a channel. The format of the returned data is set by the parameter P-STUP-00039. A structure of type CONTOUR\_VISU is returned followed by structures of type COUNTOUR\_VISU\_DATA\_V0 ... \_V8.

If the memory provided by the calling application is not sufficient to return the structure CONTOUR\_VISU, ERR\_CNC\_RET\_MEMORY is returned and the minimum size of the required memory area is returned in retLength.

### Parameter

| Name        | Type           | Meaning   |
|-------------|----------------|---|
| chanIndex   | unsigned long  | Channel index of the channel from which the variable is to be read. |
| pData       | unsigned char* | Pointer to the memory area for return values.                       |
| maxByteSize | unsigned long  | Size of the return memory.  |
| retLength   | unsigned long* | Number of bytes returned in pData.                                  |

### Return values

| Symbol             | Value | Meaning   |
|--------------------|-------|---|
| RET_FINISHED       | 0     | The function was executed without error.  |
| ERR_INVALID_CHAN   | -1    | The transferred channel index is higher than the number of configured channels -1   |
| ERR_CNC_NOT_INIT   | -3    | The simulation CNC kernel was not yet initialised.  |
| ERR_CNC_RET_MEMORY | -4    | The return value(s) does/do not fit into the memory provided. The number of bytes required is returned in the returnSize parameter. |

### 3.59 kernelv\_ch\_get\_active\_g\_codes()

**Prototype**

KERNELV\_RETURN kernelv\_ch\_get\_active\_g\_codes (unsigned long int chanIndex,  
ACTIVE\_G\_CODES\* pGCodes);

**Description**

Reads the G function groups active in the specified channel. A structure of type ACTIVE\_G\_CODES is returned. This structure contains the active G function groups. If the value -1 is in an entry, it means that the entry is unassigned.

**Parameter**

| Name      | Type            | Meaning   |
|-----------|-----------------|---|
| chanIndex | unsigned long   | Channel index of the channel from which the variable is to be read. |
| pGCodes   | ACTIVE_G_CODES* | Pointer to the structure to which the values are returned.          |

**Return values**

| Symbol           | Value | Meaning   |
|------------------|-------|---|
| RET_FINISHED     | 0     | The function was executed without error.  |
| ERR_INVALID_CHAN | -1    | The transferred channel index is higher than the number of configured channels -1 |
| ERR_CNC_NOT_INIT | -3    | The simulation CNC kernel has not yet been initialised.                           |

## 3.60 kernelv\_get\_active\_g\_group()

### Prototype

```
signed short          kernelv_get_active_g_group (ACTIVE_G_CODES* pGCodes,
                                                E_KERNELV_G_GROUP_TYPE Type);
```

### Description

Receives a structure of type ACTIVE\_G\_CODES and returns the content of the group specified in E\_KERNELV\_G\_GROUP\_TYPE.

Utility function to evaluate the structure returned by kernelv\_ch\_get\_active\_g\_codes().

### Parameter

| Name    | Type                   | Meaning  |
|---------|------------------------|--|
| pGCodes | ACTIVE_G_CODES*        | Pointer to the structure from which a G group is to be read. |
| Type    | E_KERNELV_G_GROUP_TYPE | Type ID of the G group which is to be read.                  |

### Return values

short int: Active G function of the requested group or -1 if no valid value is entered in the requested group.



## 3.61 kernelv\_ch\_get\_command\_feed()

### Prototype

```
KERNELV_RETURN kernelv_ch_get_command_feed (unsigned long int chanIndex,  
                                             signed long int* command_feed);
```

### Description

Returns the programmed feed.

### Parameter

| Name         | Type             | Meaning   |
|--------------|------------------|---|
| chanIndex    | unsigned long    | Channel index of the channel from which the variable is to be read. |
| command_feed | signed long int* | Pointer to the variable for the return value.                       |

### Return values

| Symbol           | Value | Meaning   |
|------------------|-------|---|
| RET_FINISHED     | 0     | The function was executed without error.  |
| ERR_INVALID_CHAN | -1    | The transferred channel index is higher than the number of configured channels -1 |
| ERR_CNC_NOT_INIT | -3    | The simulation CNC kernel was not yet initialised.                                |

## 3.62 kernelv\_ch\_get\_active\_feed()

### Prototype

```
KERNELV_RETURN kernelv_ch_get_active_feed (unsigned long int chanIndex,  
signed long int* command_feed);
```

### Description

Returns the actually traversed feed in  $\mu\text{m/s}$ .

### Parameter

| Name         | Type             | Meaning   |
|--------------|------------------|---|
| chanIndex    | unsigned long    | Channel index of the channel from which the variable is to be read. |
| command_feed | signed long int* | Pointer to the variable for the return value.                       |

### Return values

| Symbol           | Value | Meaning   |
|------------------|-------|---|
| RET_FINISHED     | 0     | The function was executed without error.  |
| ERR_INVALID_CHAN | -1    | The transferred channel index is higher than the number of configured channels -1 |
| ERR_CNC_NOT_INIT | -3    | The simulation CNC kernel was not yet initialised.                                |

### 3.63 kernelv\_set\_call\_ratio()

#### Prototype

```
KERNELV_RETURN kernelv_set_call_ratio (unsigned short int dec_calls,
                                       unsigned short int interpolator_calls);
```

#### Description

Defines the ratio of decoder calls to interpolator calls. One interpolator call is executed for every kernelv\_do\_cycle() call. In many real-time environments the cycle time for the path preparation task can be set irrespective of the interpolator cycle time. The function sets a ratio between path preparation cycles and interpolator cycles for the kernelv DLL.

The kernelv\_set\_call\_ratio(5, 2) function sets a ratio of 5 path preparation calls to 2 interpolator calls.

The parameters dec\_calls and interpolator\_calls may not both be 0. The ratio between the two parameters must be within the range [0,05 , 20].

#### Parameter

| Name               | Type           | Meaning                            |
|--------------------|----------------|------------------------------------|
| dec_calls          | unsigned short | Number of path preparation cycles. |
| interpolator_calls | unsigned short | Number of interpolator cycles.     |

#### Return values

| Symbol                | Value | Meaning   |
|-----------------------|-------|---|
| RET_FINISHED          | 0     | The function was executed without error.  |
| ERR_INVALID_PARAMETER | -30   | One of the parameters transferred is invalid. The following conditions apply:<br>dec_calls, interpolator_calls <> 0<br>$0.05 \leq \text{dec\_calls}/\text{interpolator\_calls} \leq 20$ . |

## 3.64 CNC error messages with kernelv

The kernelv DLL can read out and display error messages output by the CNC. There are 2 options for this:

1. Read out fully formatted error message as string. The error message is returned in the same format as entered in the error message log and in the form of a string. The `kernelv_get_error()` function can be used for this.
2. Internal storage of the error message and request for parts of the error message for further processing and display in a user interface. The remaining functions described in this section are used for this purpose.

### 3.64.1 Read out error message in the form of a string `kernelv_get_error()`

#### Prototype

```
KERNELV_RETURN kernelv_get_error (unsigned long* errorId,
                                   char* messageString,
                                   unsigned long maxStringLength,
                                   unsigned long* returnLength);
```

#### Description

Reads out error messages of the simulation CNC: Error messages are read out for all channels. As several error messages can occur during one CNC cycle, the function must be called cyclically until it returns the error ID 0.

If the memory provided by the calling application is too small for the string to be returned, the return value `ERR_CNC_RET_MEMORY` is returned. In this case, the `returnLength` parameter contains the size in bytes required for returning the string.

In any case, a value is assigned to the "errorId" parameter.

If this function is used at the same time as `kernelv_read_error()`, please note that the function also calls `kernelv_read_error()` internally. The function `kernelv_get_error_message_string()` should be used to read an error string when the function `kernelv_read_error()` is used.

#### Parameter

| Name            | Type           | Meaning  |
|-----------------|----------------|--|
| errorId         | unsigned long* | Pointer to error message ID.   |
| messageString   | char*          | Pointer to string for error message string. The calling application must provide the memory.   |
| maxStringLength | unsigned long* | Size of memory for the error message string. No string is returned if the error message string generated by the simulation CNC is longer than the memory provided. |
| returnLength    | unsigned long* | Size of the memory expected.   |

#### Return values

| Symbol             | Value | Meaning  |
|--------------------|-------|--|
| RET_FINISHED       | 0     | The function was executed without error.   |
| ERR_CNC_NOT_INIT   | -3    | The simulation CNC kernel was not yet initialised.   |
| ERR_CNC_RET_MEMORY | -4    | The return value(s) does/do not fit into the memory provided. The number of bytes required is returned in the <code>returnSize</code> parameter. |

## 3.64.2 General information on error messages

Proceed as follows to request specific information on an error message displayed:

Call the function `kernelv_read_error()`. This checks whether there is a current error message. If so, it is stored temporarily for further evaluation in the DLL.

If there is a current error message, you can request further information about it.

### 3.64.2.1 `kernelv_read_error()`

#### Prototype

```
KERNELV_RETURN kernelv_read_error (void);
```

#### Description

Checks whether there is a current error message from the simulation CNC and stores it temporarily. The error messages of all channels are checked.

If there is a current error message, further functions can be used to request details about this specific error message. When this function is called again, a new error message may be read out. Then it is no longer possible to request the error message properties of a previously pending error message.

As several error messages may occur during one CNC cycle, the function must be called cyclically until it returns `RET_FINISHED`.

#### Parameter

-

#### Return values

| Name                          | Value | Meaning  |
|-------------------------------|-------|--|
| <code>RET_FINISHED</code>     | 0     | No error message was read.                         |
| <code>RET_BUSY</code>         | 1     | An error message was read.                         |
| <code>ERR_CNC_NOT_INIT</code> | -3    | The simulation CNC kernel was not yet initialised. |

### 3.64.2.2 kernelv\_get\_error\_id()

#### Prototype

```
unsigned long int    kernelv_get_error_id(void);
```

#### Description

Returns the error number of an error message previously read by `kernelv_read_error()`. If `kernelv_read_error()` was not called or there is no current error message, the value 0 is returned.

#### Parameter

-

#### Return values

Type: unsigned long int

0 if there is no current error message; otherwise the error number.

The individual errors numbers are described in the CNC diagnostics manual ([DIAG]).

### 3.64.2.3 kernelv\_get\_error\_reaction()

**Prototype**

signed short int

kernelv\_get\_error\_reaction(void);

**Description**

Returns the error reaction class of an error message previously read by kernelv\_read\_error(). If no error message was read or there is no current error message, the value -1 is returned.

**Parameter**

-

**Return values**

Type: signed short int

-1 if there is no current error message; otherwise the error reaction class (see also [DIAG]).

| Error reaction class | Internal error reaction  |
|----------------------|--|
| 1                    | <b>No reaction.</b><br>Only possible in the event of a warning (error class 1).  |
| 2                    | <b>NC program execution aborted and transition to error state.</b><br>If an error occurs in the NC block preparation area, the interpolator executes the NC blocks that have already been prepared. In this case, the time between the occurrence of the error and machine standstill depends on the type and number of buffered NC blocks. The BF reporting the error assumes an error state. |
| 3                    | <b>Job execution aborted and transition to normal state.</b><br>After an error message, BFs that provide services for other BFs (servers), for example AXIS ADMINISTRATION, FILE ADMINISTRATION, MANUAL MODE, etc., abort job execution and return to normal state.  |
| 4                    | <b>Motion stopped (feedhold) for the entire axis group and transition to error state.</b>  |
| 5                    | <b>Abrupt axis stop for defective axis, feedhold for the other axes in the axis group and transition to an error state.</b>  |
| 6                    | <b>Abrupt axis stop for all axes and transition to an error state.</b><br>Position control assumes an error state.   |
| 7                    | <b>Closed-loop controlled axis stop for defective axis, feedhold for the other axes in the axis group and transition to an error state.</b>  |
| 8                    | <b>Open-loop controlled axis stop for defective axis, feedhold for the other axes in the axis group and transition to an error state.</b><br>The position control loop of the defective axis is opened.  |

### 3.64.2.4 kernelv\_get\_error\_severity()

#### Prototype

signed short int

```
kernelv_get_error_severity(void);
```

#### Description

Returns the error remedy class of an error message previously read by `kernelv_read_error()`. If no error message was read or there is no current error message, the value -1 is returned.

#### Parameter

-

#### Return values

Type: signed short int

-1 if there is no current error message; otherwise the error remedy class (see also [DIAG]).

| Error remedy class | Internal error remedy   |
|--------------------|---|
| 0                  | Error message acts as warning; this causes <b>automatically</b> an <b>internal error remedy</b> . |
| 2                  | Requires complete <b>reset</b> of the NC channel.   |
| 5                  | Requires complete <b>reset</b> of the NC channel.   |
| 6                  | Requires <b>restart</b> of the CNC channel.   |
| 7                  | Requires <b>restart</b> of the entire NC kernel after switch-off.                                 |



### 3.64.2.5 kernelv\_get\_error\_channel()

**Prototype**

signed short int

```
kernelv_get_error_channel(void);
```

**Description**

Returns the channel number of the channel in which an error message previously read by `kernelv_read_error()` occurred. If no error message was read or there is no current error message, the value -1 is returned.

**Parameter**

-

**Return values**

Type: signed short int

0 if there is no current error message; otherwise the channel number.

### 3.64.2.6 kernelv\_get\_error\_message\_string()

#### Prototype

```
KERNELV_RETURN kernelv_get_error_message_string(char * string,
                                                unsigned long int *length);
```

#### Description

Returns the formatted string of an error message previously read by `kernelv_read_error()` in the form of a string. The returned string is identical to the string returned by the function `kernelv_get_error()`.

When called, the 'string' parameter transfers a pointer to the memory for the return string. The size of this memory area is contained in 'length'. If the memory provided is sufficient to return the string, the length of the returned string is returned in 'length'.

If the memory provided by the calling application is too small for the string to be returned, the return value `ERR_CNC_RET_MEMORY` is returned. In this case, the 'length' parameter contains the size in bytes required to return the string.

#### Parameter

| Name   | Type           | Meaning   |
|--------|----------------|---|
| string | char*          | Pointer to string for error message string. The calling application must provide the memory.  |
| length | unsigned long* | Size of memory for the error message string. No string is returned if the error message string generated by the simulation CNC is longer than the memory provided. Either the length of the returned string is returned or the size of the memory area required for the return. |

#### Return values

| Symbol             | Value | Meaning   |
|--------------------|-------|---|
| RET_FINISHED       | 0     | The function was executed without error   |
| ERR_CNC_NOT_INIT   | -3    | The simulation CNC kernel was not yet initialised.  |
| ERR_CNC_RET_MEMORY | -4    | The return value(s) does/do not fit into the memory provided. The number of bytes required is returned in the 'returnSize' parameter. |

### 3.64.2.7 kernelv\_get\_error\_id\_text()

**Prototype**

```
KERNELV_RETUR kernelv_get_error_message_string(char * string,
N unsigned long int *length);
```

**Description**

Returns the error message text belonging to the current error number of an error message previously read by kernelv\_read\_error(). When called, the 'string' parameter transfers a pointer to the memory for the return string, The size of this memory area is contained in 'length'. If the memory provided is sufficient to return the string, the length of the returned string is returned in 'length'.

If the memory provided by the calling application is too small for the string to be returned, the return value ERR\_CNC\_RET\_MEMORY is returned In this case, the 'length' parameter contains the size in bytes required to return the string.

**Parameter**

| Name   | Type           | Meaning   |
|--------|----------------|---|
| string | char*          | Pointer to string for error message string. The calling application must provide the memory.  |
| length | unsigned long* | Size of memory for the error message string. No string is returned if the error message string generated by the simulation CNC is longer than the memory provided. Either the length of the returned string is returned or the size of the memory area required for the return. |

**Return values**

| Symbol             | Value | Meaning   |
|--------------------|-------|---|
| RET_FINISHED       | 0     | The function was executed without error.  |
| ERR_CNC_NOT_INIT   | -3    | The simulation CNC kernel was not yet initialised.  |
| ERR_CNC_RET_MEMORY | -4    | The return value(s) does/do not fit into the memory provided. The number of bytes required is returned in the 'returnSize' parameter. |

### 3.64.2.8 kernelv\_get\_error\_message\_values()

#### Prototype

```
KERNELV_RETURN kernelv_get_error_message_string (KERNELV_ERROR_VALUE * p_values,
                                                unsigned long int *length);
```

#### Description

Returns the values output in an error message.

Before using this function, a check must first be made whether there is a current error by calling the function `kernelv_read_error()`.

The values are returned in an array of structures of type `KERNELV_ERROR_VALUE`. The array size is `KERNELV_ERROR_VALUE_COUNT`. Specify the size of the memory areas for the error message values in the 'length' call parameter.

If the return array does not fit in the memory provided, the return value `ERR_CNC_RET_MEMORY` is output and 'length' contains the memory size in bytes required for the return.

#### Parameter

| Name     | Type                 | Meaning  |
|----------|----------------------|--|
| p_values | KERNELV_ERROR_VALUE* | Pointer to string for error message values. The calling application must provide the memory.   |
| length   | unsigned long*       | Size of memory for the error message values. If the memory provided is not large enough, the required memory size in bytes is returned; otherwise the number of bytes is returned. |

#### Return values

| Symbol             | Value | Meaning   |
|--------------------|-------|---|
| RET_FINISHED       | 0     | The function was executed without error.  |
| ERR_CNC_NOT_INIT   | -3    | The simulation CNC kernel was not yet initialised.  |
| ERR_CNC_RET_MEMORY | -4    | The return value(s) does/do not fit into the memory provided. The number of bytes required is returned in the 'length' parameter. |

### 3.64.2.9 kernelv\_get\_error\_cycle\_time()

**Prototype**

```
unsigned long long int kernelv_get_error_cycle_time(void);
```

**Description**

Returns the CNC cycle in which an error message previously read by `kernelv_read_error()` occurred. If no error message was read or there is no current error message, the value 0 is returned.

**Parameter**

-

**Return values**

Type: unsigned long int

0 if there is no current error message; otherwise the CNC cycle in which the error message occurred.

### 3.64.3 Error messages caused by NC programs

With error messages caused by an NC program, it is possible to request additional information about the NC program in which the error occurred. This information can then be used to localise the error in the NC program.

A check is first made whether the NC program contains specific information by calling the function `kernelv_is_program_err()`. If this function returns the value 1, the current error message was caused by an NC program and further information about the error in the NC program can be called by calling the functions below.

#### 3.64.3.1 `kernelv_error_is_program_error()`

##### Prototype

unsigned char

`kernelv_error_is_program_error(void);`

##### Description

Indicates whether the current error was caused by a CNC program.

##### Parameter

-

##### Return values

| Value | Meaning   |
|-------|---|
| 0     | The current error message was not caused by an NC program or there is no error. |
| 1     | The current error message was caused by an NC program.                          |

### 3.64.3.2 kernelv\_program\_error\_get\_path

**Prototype**

```
KERNELV_RETURN kernelv_program_error_get_path(char *return_string,
                                              unsigned long * return_length);
```

**Description**

Returns the program path used to start the active program.

If the program was started by the absolute specification of a file name, an empty string is returned; if the current active program is a manual block, “-” is returned (less the inverted commas).

A zero terminated string is returned. The number of returned bytes is returned in return\_length, i.e. including the terminating zero.

If the memory provided by the calling application is too small for the string to be returned, the return value ERR\_CNC\_RET\_MEMORY is returned. In this case, the returnLength parameter contains the size in bytes required for returning the string.

**Parameter**

| Name          | Type           | Meaning  |
|---------------|----------------|--|
| return_string | char*          | Pointer to string for the program path. The calling application must provide the memory.   |
| return_length | unsigned long* | Size of memory for the return value. If the return string is longer than the memory provided, nothing is returned.<br><br>Either the length of the returned string is returned or the size of the memory area required for the return. |

**Return values**

| Symbol             | Value | Meaning   |
|--------------------|-------|---|
| RET_FINISHED       | 0     | The function was executed without error.  |
| ERR_CNC_RET_MEMORY | -4    | The return value(s) does/do not fit into the memory provided. The number of bytes required is returned in the 'length' parameter. |
| ERR_CNC_NO_DATA    | -28   | The requested data is not available.<br><br>There is no available NC program specific data relating to an CNC error.              |

### 3.64.3.3 kernelv\_program\_error\_get\_program\_name

#### Prototype

```
KERNELV_RETURN kernelv_program_error_get_program_name
                (char *return_string,
                 unsigned long * returnLength);
```

#### Description

Returns the program name of the active NC program.

The program name is specified at the start of the main program by a "%" character; for more information see [PROG]. If no program name was specified for the NC program, an empty string is returned.

A zero terminated string is returned. The number of returned bytes is returned in return\_length, i.e. the terminating zero is included.

If the memory provided by the calling application is too small for the string to be returned, the return value ERR\_CNC\_RET\_MEMORY is returned. In this case, the returnLength parameter contains the size in bytes required for returning the string.

#### Parameter

| Name          | Type           | Meaning  |
|---------------|----------------|--|
| return_string | char*          | Pointer to string for the program name. The calling application must provide the memory.   |
| return_length | unsigned long* | Size of memory for the return value. If the return string is longer than the memory provided, nothing is returned.<br>Either the length of the returned string is returned or the size of the memory area required for the return. |

#### Return values

| Symbol             | Value | Meaning   |
|--------------------|-------|---|
| RET_FINISHED       | 0     | The function was executed without error.  |
| ERR_CNC_RET_MEMORY | -4    | The return value(s) does/do not fit into the memory provided. The number of bytes required is returned in the 'length' parameter. |
| ERR_CNC_NO_DATA    | -28   | The requested data is not available.  |



### 3.64.3.4 kernelv\_program\_error\_get\_file\_name

**Prototype**

```
KERNELV_RETURN kernelv_program_error_get_file_name(char *return_string,
                                                    unsigned long * returnLength);
```

**Description**

Returns the file name of the active NC program.

If the program was started by specifying an absolute program name, the complete program name is returned.

If the CNC controller opened the NC program by using a search path, this function returns the file name specified at program start. The search path used can be requested by the function kernelv\_program\_error\_get\_path().

The function returns the number of returned bytes in returnLength, i.e. including the zero terminating the string.

If the memory provided by the calling application is too small for the string to be returned, the return value ERR\_CNC\_RET\_MEMORY is returned. In this case, the returnLength parameter contains the size in bytes required for returning the string.

**Parameter**

| Name          | Type           | Meaning  |
|---------------|----------------|--|
| return_string | char*          | Pointer to string for the program path. The calling application must provide the memory.   |
| return_length | unsigned long* | Size of memory for the return value. If the return string is longer than the memory provided, nothing is returned.<br>Either the length of the returned string is returned or the size of the memory area required for the return. |

**Return values**

| Symbol             | Value | Meaning   |
|--------------------|-------|---|
| RET_FINISHED       | 0     | The function was executed without error.  |
| ERR_CNC_RET_MEMORY | -4    | The return value(s) does/do not fit into the memory provided. The number of bytes required is returned in the 'length' parameter. |
| ERR_CNC_NO_DATA    | -28   | The requested data is not available. There is no available NC program specific data relating to an CNC error.                     |

### 3.64.3.5 kernelv\_program\_error\_get\_fileoffset

#### Prototype

```
KERNELV_RETURN kernelv_program_error_get_fileoffset(unsigned long * fileoffset);
```

#### Description

Returns the file offset of the error in the program file.

#### Parameter

Pointer to memory location for the file offset.

#### Return values

| Symbol          | Value | Meaning   |
|-----------------|-------|---|
| RET_FINISHED    | 0     | The function was executed without error.  |
| ERR_CNC_NO_DATA | -28   | The requested data is not available. There is no available NC program specific data relating to an CNC error. |

### 3.64.3.6 kernelv\_program\_error\_get\_lineoffset

#### Prototype

```
KERNELV_RETURN kernelv_program_error_get_linenoffset(unsigned short int*);
```

#### Description

Returns the offset of the error in the NC line

#### Parameter

Pointer to the memory location for line offset.

#### Return values

| Symbol          | Value | Meaning   |
|-----------------|-------|---|
| RET_FINISHED    | 0     | The function was executed without error.  |
| ERR_CNC_NO_DATA | -28   | The requested data is not available. There is no available NC program specific data relating to an CNC error. |

### 3.64.3.7 kernelv\_program\_error\_get\_tokenoffset

**Prototype**

KERNELV\_RETURN kernelv\_program\_error\_get\_tokenoffset(unsigned short int\*);

**Description**

Returns the offset of the error in the parsed NC\_token.

**Parameter**

Pointer to the memory location for token offset.

**Return values**

| Symbol          | Value | Meaning   |
|-----------------|-------|---|
| RET_FINISHED    | 0     | The function was executed without error.  |
| ERR_CNC_NO_DATA | -28   | The requested data is not available. There is no available NC program specific data relating to an CNC error. |

### 3.64.3.8 kernelv\_program\_error\_get\_linenummer

**Prototype**

KERNELV\_RETURN kernelv\_program\_error\_get\_linenummer(signed long \* linenummer);

**Description**

Returns the programmed BNC line number of the error in the program file.

**Parameter**

Pointer to the memory location for line number

**Return values**

| Symbol          | Value | Meaning   |
|-----------------|-------|---|
| RET_FINISHED    | 0     | The function was executed without error.  |
| ERR_CNC_NO_DATA | -28   | The requested data is not available. There is no available NC program specific data relating to an CNC error. |

## 3.65 Coordinate systems and offsets

The programming coordinate system in an NC program can be adapted to the given requirements by NC commands, e.g. by rotating or shifting the coordinate system. The NC commands used here include #CS DEF/#CS ON or #ACS DEF/#ACS ON. For details please see the Programming Manual, Section 'Coordinate systems'.

Coordinate systems can also be concatenated by the repeated use of these commands. The resulting coordinate system is then formed by a stack of subordinate coordinate systems.

Other coordinate systems can be added to this coordinate system stack by other NC commands (#CS ADD\*) without them having any effect, i.e. influencing the resulting coordinate system. The command #CS SELECT\* defines the coordinate system to be used.

Within each coordinate system, offsets can be defined for each of the axes in the channel, e.g. by the commands G54 ... G59 (zero offset) or G92 (reference point offset).

The following information is read out by the API functions provided:

Number of defined coordinate systems

Index of the active coordinate system in the coordinate system stack

Information about a coordinate system at a specific index in the stack

Information about axis-specific offsets within a coordinate system in the stack

\*This NC command is not available in all versions.

### 3.65.1 kernelv\_ch\_get\_cs\_name()

#### Prototype

```
KERNELV_RETURN kernelv_ch_get_cs_name(unsigned long int chanIndex,
                                       unsigned short csIndex,
                                       char *name,
                                       unsigned long int bufferSize,
                                       unsigned long int *retBytes);
```

#### Description

Returns the name of the coordinate system defined in the NC program.

If no coordinate system is defined in the coordinate system stack at the location defined by csIndex, an empty string is returned and the return value of the function is ERR\_CNC\_NO\_DATA.

#### Parameter

| Name         | Type           | Meaning  |
|--------------|----------------|--|
| chanIndex    | unsigned long  | Channel index of the channel.  |
| csIndex      | unsigned short | Index of the coordinate system in the coordinate system stack.   |
| name         | char*          | Pointer to the memory location for the coordinate system name.   |
| nameLength   | unsigned long  | Length of the memory area for the coordinate system name.  |
| returnLength | unsigned long* | Pointer to the value to which the actually returned number of bytes is to be written. The number of characters is returned in the file name +1.<br><br>If the transferred memory is too small for the return value, the return value ERR_CNC_RET_MEMORY is returned and this parameter returns the required memory size. |

#### Return values

| Symbol                | Value | Meaning   |
|-----------------------|-------|---|
| RET_FINISHED          | 0     | The function was executed without error.  |
| ERR_INVALID_CHAN      | -1    | The transferred channel index is higher than the number of configured channels -1   |
| ERR_CNC_NOT_INIT      | -3    | The simulation CNC kernel was not yet initialised.  |
| ERR_CNC_RET_MEMORY    | -4    | The return value(s) does/do not fit into the memory provided. In this case, the minimum size required for the return is returned in returnLength.   |
| ERR_READ_ERR          | -26   | An error occurred on reading data from the kernelv DLL.   |
| ERR_CNC_NO_DATA       | -28   | The requested data is not available. No coordinate system is defined in the coordinate system stack at the location specified by the call parameter csIndex. An empty string is returned. |
| ERR_INVALID_PARAMETER | -30   | An invalid parameter was transferred:   |

|  |  |
|--|--|
|  | The coordinate system index transferred in the parameter csIndex is higher than the maximum possible index in the coordinate system stack. |
|--|--|

### 3.65.2 kernelv\_ch\_get\_cs\_rot\_matrix()

**Prototype**

```
KERNELV_RETURN kernelv_ch_get_cs_rot_matrix(unsigned long int chanIndex,
                                             unsigned short csIndex,
                                             double *matrix,
                                             unsigned long int bufferSize,
                                             unsigned long int *retBytes);
```

**Description**

Returns the rotation matrix of the coordinate system specified by the parameter csIndex.

A 3x3 rotation matrix is returned. It can be used to generate the coordinate system from the subordinate coordinate system.

The rotation matrix generated by the parameters  $\phi 1$ ,  $\phi 2$   $\phi 3$  of the following NC command is returned:

**#CS DEF [CS1] [ <v1>,<v2>,<v3>,< $\phi 1$ >,< $\phi 2$ >,< $\phi 3$ > ]**

If no coordinate system is defined at the specific index in the coordinate system stack, a standard matrix is returned and the return value of the function is ERR\_CNC\_NO\_DATA.

**Parameter**

| Name         | Type           | Meaning  |
|--------------|----------------|--|
| chanIndex    | unsigned long  | Channel index of the channel.  |
| csIndex      | unsigned short | Index of the coordinate system in the coordinate system stack.   |
| matrix       | double*        | Pointer to the memory location for the rotation matrix.  |
| bufferSize   | unsigned long  | Length of the memory area for the rotation matrix must be at least 3 x 3 x sizeof(double).   |
| returnLength | unsigned long* | Pointer to the value to which the actually returned number of bytes is to be written.<br><br>If the transferred memory is too small for the return value, the return value ERR_CNC_RET_MEMORY is returned and this parameter returns the required memory size. |

**Return values**

| Symbol                | Value | Meaning   |
|-----------------------|-------|---|
| RET_FINISHED          | 0     | The function was executed without error.  |
| ERR_INVALID_CHAN      | -1    | The transferred channel index is higher than the number of configured channels -1   |
| ERR_CNC_NOT_INIT      | -3    | The simulation CNC kernel was not yet initialised.  |
| ERR_CNC_RET_MEMORY    | -4    | The return value(s) does/do not fit into the memory provided. In this case, the minimum size required for the return is returned in returnLength.   |
| ERR_READ_ERR          | -26   | An error occurred on reading data from the kernelv DLL.   |
| ERR_CNC_NO_DATA       | -28   | The requested data is not available. No coordinate system is defined in the coordinate system stack at the location specified by the call parameter csIndex. The standard matrix is returned. |
| ERR_INVALID_PARAMETER | -30   | An invalid parameter was transferred:   |

|  |  |
|--|--|
|  | The coordinate system index transferred in the parameter csIndex is higher than the maximum possible index in the coordinate system stack. |
|--|--|

### 3.65.3 kernelv\_ch\_get\_cs\_shift\_vector()

#### Prototype

```
KERNELV_RETURN kernelv_ch_get_cs_shift_vector(unsigned long int chanIndex,
                                              unsigned short csIndex,
                                              double *vector,
                                              unsigned long int bufferSize,
                                              unsigned long int *retBytes);
```

#### Description

Returns the shift of the origin of the coordinate system specified by the parameter csIndex.

A vector with three elements is returned, specifying the shift of the coordinate system origin to the origin of the subordinate coordinate system.

The shift vector that is generated by the parameters **v1**, **v2**, **v3** of the following NC command is returned:

```
#CS DEF [CS1] [ <v1>,<v2>,<v3>,<φ1>,<φ2>,<φ3> ]
```

If no coordinate system is defined at the specific index in the coordinate system stack, a zero vector is returned and the return value of the function is ERR\_CNC\_NO\_DATA.

#### Parameter

| Name         | Type           | Meaning  |
|--------------|----------------|--|
| chanIndex    | unsigned long  | Channel index of the channel.  |
| csIndex      | unsigned short | Index of the coordinate system in the coordinate system stack.   |
| vector       | double*        | Pointer to the memory location for the shift vector.   |
| bufferSize   | unsigned long  | Length of the memory area for the rotation matrix must be at least 3 x sizeof(double).   |
| returnLength | unsigned long* | Pointer to the value to which the actually returned number of bytes is to be written.<br><br>If the transferred memory is too small for the return value, the return value ERR_CNC_RET_MEMORY is returned and this parameter returns the required memory size. |

#### Return values

| Symbol                | Value | Meaning   |
|-----------------------|-------|---|
| RET_FINISHED          | 0     | The function was executed without error.  |
| ERR_INVALID_CHAN      | -1    | The transferred channel index is higher than the number of configured channels -1   |
| ERR_CNC_NOT_INIT      | -3    | The simulation CNC kernel was not yet initialised.  |
| ERR_CNC_RET_MEMORY    | -4    | The return value(s) does/do not fit into the memory provided. In this case, the minimum size required for the return is returned in returnLength.                                       |
| ERR_READ_ERR          | -26   | An error occurred on reading data from the kernelv DLL.   |
| ERR_CNC_NO_DATA       | -28   | The requested data is not available. No coordinate system is defined in the coordinate system stack at the location specified by the call parameter csIndex. A zero vector is returned. |
| ERR_INVALID_PARAMETER | -30   | An invalid parameter was transferred:   |

|  |  |
|--|--|
|  | The coordinate system index transferred in the parameter csIndex is higher than the maximum possible index in the coordinate system stack. |
|--|--|

### 3.65.4 kernelv\_ch\_get\_cs\_count()

**Prototype**

```
KERNELV_RETURN kernelv_ch_get_cs_count(unsigned long int chanIndex,
                                         unsigned short *count);
```

**Description**

Returns the number of defined coordinate systems.

Even if no coordinate system is defined in the NC program, there is always a basic coordinate system at coordinate system stack level 0. Its rotation matrix is the standard matrix and its shift vector is the zero vector.

**Parameter**

| Name      | Type            | Meaning  |
|-----------|-----------------|--|
| chanIndex | unsigned long   | Channel index of the channel.  |
| count     | unsigned short* | Pointer to the memory location for the number of coordinate systems. |

**Return values**

| Symbol           | Value | Meaning   |
|------------------|-------|---|
| RET_FINISHED     | 0     | The function was executed without error.  |
| ERR_INVALID_CHAN | -1    | The transferred channel index is higher than the number of configured channels -1 |
| ERR_CNC_NOT_INIT | -3    | The simulation CNC kernel was not yet initialised.                                |
| ERR_READ_ERR     | -26   | An error occurred on reading data from the kernelv DLL.                           |

### 3.65.5 kernelv\_ch\_get\_active\_cs\_index()

**Prototype**

```
KERNELV_RETURN kernelv_ch_get_active_cs_index(unsigned long int chanIndex,
                                                unsigned short *csIndex);
```

**Description**

Returns the index of the active coordinate system.

Even if no coordinate system is defined in the NC program, there is always a basic coordinate system at coordinate system stack level 0. Its rotation matrix is the standard matrix and its translation vector is the zero vector.

**Parameter**

| Name      | Type            | Meaning   |
|-----------|-----------------|---|
| chanIndex | unsigned long   | Channel index of the channel.                                       |
| csIndex   | unsigned short* | Pointer to the memory location for the index of coordinate systems. |

**Return values**

| Symbol           | Value | Meaning   |
|------------------|-------|---|
| RET_FINISHED     | 0     | The function was executed without error.  |
| ERR_INVALID_CHAN | -1    | The transferred channel index is higher than the number of configured channels –1 |
| ERR_CNC_NOT_INIT | -3    | The simulation CNC kernel was not yet initialised.                                |
| ERR_READ_ERR     | -26   | An error occurred on reading data from the kernelv DLL.                           |

**3.65.6 kernelv\_ch\_axis\_get\_offsets()****Prototype**

```
KERNELV_RETURN kernelv_ch_axis_get_offsets(unsigned long int chanIndex,
                                           unsigned long int axisIndex,
                                           unsigned short int csIndex,
                                           signed long int *offsets,
                                           unsigned long int bufferSize,
                                           unsigned long int *retBytes);
```

**Description**

Returns the axis-specific offsets in the coordinate system specified by the parameter csIndex.

The parameter axisIndex identifies the index of the axis in the NC channel specified by ChanIndex.

A vector with eight elements is returned. It lists the offsets imported by the various NC commands for the axis defined by axisIndex.

The enumeration KERNELV\_AXIS\_OFFSET\_TYPES specifies the assignment of the index in the vector to the various offset types.

**Parameter**

| Name         | Type           | Meaning  |
|--------------|----------------|--|
| chanIndex    | unsigned long  | Channel index of the channel.  |
| axisIndex    | unsigned long  | Index of the axis in the NC channel.   |
| csIndex      | unsigned short | Index of the coordinate system in the coordinate system stack.   |
| offsets      | signed long*   | Pointer to the memory location for the offset vector.  |
| nameLength   | unsigned long  | Length of the memory area for the rotation matrix must be at least 8 x sizeof (signed long int).   |
| returnLength | unsigned long* | Pointer to the value to which the actually returned number of bytes is to be written.<br><br>If the transferred memory is too small for the return value, the return value ERR_CNC_RET_MEMORY is returned and this parameter returns the required memory size. |

**Return values**

| Symbol             | Value | Meaning   |
|--------------------|-------|---|
| RET_FINISHED       | 0     | The function was executed without error.  |
| ERR_INVALID_CHAN   | -1    | The transferred channel index is higher than the number of configured channels –1   |
| ERR_CNC_NOT_INIT   | -3    | The simulation CNC kernel was not yet initialised.  |
| ERR_CNC_RET_MEMORY | -4    | The return value(s) does/do not fit into the memory provided. In this case, the minimum size required for the return is returned in returnLength. |



|                       |     |   |
|-----------------------|-----|---|
| ERR_READ_ERR          | -26 | An error occurred on reading data from the kernelv DLL.   |
| ERR_CNC_NO_DATA       | -28 | The requested data is not available. No coordinate system is defined in the coordinate system stack at the location specified by the call parameter csIndex. A zero vector is returned. |
| ERR_INVALID_PARAMETER | -30 | An invalid parameter was transferred:<br>The coordinate system index transferred in the parameter csIndex is higher than the maximum possible index in the coordinate system stack.     |

### 3.65.7 kernelv\_ch\_get\_total\_cs\_rot\_matrix()

**Prototype**

```
KERNELV_RETURN kernelv_ch_get_total_cs_rot_matrix(unsigned long int chanIndex,
                                                    double *matrix,
                                                    unsigned long int bufferSize,
                                                    unsigned long int *retBytes);
```

**Description**

Returns the rotation matrix generated by concatenating all active coordinate systems.

The rotation matrix generated by the parameters  $\phi 1$ ,  $\phi 2$   $\phi 3$  of the following NC command is returned:

**#CS DEF [CS1] [ <v1>,<v2>,<v3>,< $\phi 1$ >,< $\phi 2$ >,< $\phi 3$ > ]**

If no coordinate system is active, the standard matrix is returned.

**Parameter**

| Name         | Type           | Meaning  |
|--------------|----------------|--|
| chanIndex    | unsigned long  | Channel index of the channel.  |
| matrix       | double*        | Pointer to the memory location for the rotation matrix.  |
| bufferSize   | unsigned long  | Length of the memory area for the rotation matrix must be at least 3 x 3 x sizeof(double).   |
| returnLength | unsigned long* | Pointer to the value to which the actually returned number of bytes is to be written.<br>If the transferred memory is too small for the return value, the return value ERR_CNC_RET_MEMORY is returned and this parameter returns the required memory size. |

**Return values**

| Symbol             | Value | Meaning   |
|--------------------|-------|---|
| RET_FINISHED       | 0     | The function was executed without error.  |
| ERR_INVALID_CHAN   | -1    | The transferred channel index is higher than the number of configured channels -1   |
| ERR_CNC_NOT_INIT   | -3    | The simulation CNC kernel was not yet initialised.  |
| ERR_CNC_RET_MEMORY | -4    | The return value(s) does/do not fit into the memory provided. In this case, the minimum size required for the return is returned in returnLength. |

### 3.65.8 kernelv\_ch\_get\_total\_cs\_offset

#### Prototype

```
KERNELV_RETURN kernelv_ch_get_total_cs_offset(unsigned long int chanIndex,
                                              double *vector,
                                              unsigned long int bufferSize,
                                              unsigned long int *retBytes);
```

#### Description

Returns the shift of the origin of the coordinate system resulting from concatenating all active coordinate systems.

A vector with three elements is returned, specifying the shift of the coordinate system origin to the origin of the basic coordinate system.

The shift vector that is generated by the parameters **v1**, **v2**, **v3** of the following NC command is returned:

```
#CS DEF [CS1] [ <v1>,<v2>,<v3>,<φ1>,<φ2>,<φ3> ]
```

If no coordinate system is active, a zero vector is returned.

#### Parameter

| Name         | Type           | Meaning  |
|--------------|----------------|--|
| chanIndex    | unsigned long  | Channel index of the channel.  |
| vector       | double*        | Pointer to the memory location for the translation vector.   |
| bufferSize   | unsigned long  | Length of the memory area for the rotation matrix must be at least 3 x sizeof(double).   |
| returnLength | unsigned long* | Pointer to the value to which the actually returned number of bytes is to be written.<br><br>If the transferred memory is too small for the return value, the return value ERR_CNC_RET_MEMORY is returned and this parameter returns the required memory size. |

#### Return values

| Symbol             | Value | Meaning   |
|--------------------|-------|---|
| RET_FINISHED       | 0     | The function was executed without error.  |
| ERR_INVALID_CHAN   | -1    | The transferred channel index is higher than the number of configured channels -1   |
| ERR_CNC_NOT_INIT   | -3    | The simulation CNC kernel was not yet initialised.  |
| ERR_CNC_RET_MEMORY | -4    | The return value(s) does/do not fit into the memory provided. In this case, the minimum size required for the return is returned in returnLength. |

### 3.65.9 kernelv\_ch\_get\_total\_cs\_def()

**Prototype**

```
KERNELV_RETURN kernelv_ch_get_total_cs_def(unsigned long int chanIndex,
                                           double *vector,
                                           unsigned long int bufferSize,
                                           unsigned long int *retBytes);
```

**Description**

Returns the shift and rotation angle of the coordinate system resulting from concatenating all active coordinate systems.

A vector with six elements is returned. The first three elements contain the shift of the coordinate system origin to the origin of the basic coordinate system. The three following vector elements contain the rotation angle in degrees required to generate the coordinate system from the basic coordinate system. Similar to the documentation of the #CS documentation, the sequence to execute the rotations is  $\phi_3$ ,  $\phi_2$ ,  $\phi_1$  in this order.

**#CS DEF [CS1] [ <v1>,<v2>,<v3>,< $\phi$ 1>,< $\phi$ 2>,< $\phi$ 3> ]**

| Index | Meaning in the #CS command |
|-------|----------------------------|
| 0     | <v1>                       |
| 1     | <v2>                       |
| 2     | <v3>                       |
| 3     | < $\phi$ 1>                |
| 4     | < $\phi$ 2>                |
| 5     | < $\phi$ 3>                |

If no coordinate system is active, a zero vector is returned.

**Parameter**

| Name         | Type           | Meaning  |
|--------------|----------------|--|
| chanIndex    | unsigned long  | Channel index of the channel.  |
| vector       | double*        | Pointer to the memory location for the translation vector.   |
| bufferSize   | unsigned long  | Length of the memory area for the rotation matrix must be at least 3 x sizeof(double).   |
| returnLength | unsigned long* | Pointer to the value to which the actually returned number of bytes is to be written.<br><br>If the transferred memory is too small for the return value, the return value ERR_CNC_RET_MEMORY is returned and this parameter returns the required memory size. |

**Return values**

| Symbol             | Value | Meaning   |
|--------------------|-------|---|
| RET_FINISHED       | 0     | The function was executed without error.  |
| ERR_INVALID_CHAN   | -1    | The transferred channel index is higher than the number of configured channels -1   |
| ERR_CNC_NOT_INIT   | -3    | The simulation CNC kernel was not yet initialised.  |
| ERR_CNC_RET_MEMORY | -4    | The return value(s) does/do not fit into the memory provided. In this case, the minimum size required for the return is returned in returnLength. |

### 3.65.10 kernelv\_ch\_get\_coord\_sys\_active()

#### Prototype

```
KERNELV_RETURN kernelv_ch_get_coord_sys_active(unsigned long int chanIndex,
                                                unsigned char *active);
```

#### Description

Returns the number of the current active kinematic transformation. If no kinematic transformation is active, a zero is returned.

#### Parameter

| Name      | Type            | Meaning                                       |
|-----------|-----------------|---|
| chanIndex | unsigned long   | Channel index of the channel.                 |
| active    | unsigned short* | Pointer to the memory area for return values. |

#### Return values

| Symbol           | Value | Meaning   |
|------------------|-------|---|
| RET_FINISHED     | 0     | The function was executed without error.  |
| ERR_INVALID_CHAN | -1    | The transferred channel index is higher than the number of configured channels -1 |
| ERR_CNC_NOT_INIT | -3    | The simulation CNC kernel was not yet initialised.                                |

## 3.66 Kinematic transformations

### 3.66.1 kernelv\_ch\_get\_kin\_trafo\_active()

#### Prototype

```
KERNELV_RETURN kernelv_ch_get_kin_trafo_active(unsigned long int chanIndex,
                                                unsigned char *active);
```

#### Description

Returns whether a coordinate system is active in the specified channel or not.

#### Parameter

| Name      | Type           | Meaning                                       |
|-----------|----------------|---|
| chanIndex | unsigned long  | Channel index of the channel.                 |
| active    | unsigned char* | Pointer to the memory area for return values. |

#### Return values

| Symbol           | Value | Meaning   |
|------------------|-------|---|
| RET_FINISHED     | 0     | The function was executed without error.  |
| ERR_INVALID_CHAN | -1    | The transferred channel index is higher than the number of configured channels -1 |
| ERR_CNC_NOT_INIT | -3    | The simulation CNC kernel was not yet initialised.                                |

### 3.66.2 kernelv\_ch\_get\_active\_kin\_id()

**Prototype**

KERNELV\_RETURN kernelv\_ch\_get\_active\_kin\_id(unsigned long int chanIndex,  
unsigned short \*kin\_id);

**Description**

Returns the number of the current active kinematic transformation. If no kinematic transformation is active, a zero is returned.

**Parameter**

| Name      | Type            | Meaning                                       |
|-----------|-----------------|---|
| chanIndex | unsigned long   | Channel index of the channel.                 |
| kin_id    | unsigned short* | Pointer to the memory area for return values. |

**Return values**

| Symbol           | Value | Meaning   |
|------------------|-------|---|
| RET_FINISHED     | 0     | The function was executed without error.  |
| ERR_INVALID_CHAN | -1    | The transferred channel index is higher than the number of configured channels -1 |
| ERR_CNC_NOT_INIT | -3    | The simulation CNC kernel was not yet initialised.                                |

## 3.67 External measuring hardware

By default, measurement runs are simulated in the kernelv DLL for the measurement simulation configured for each axis.

With the measurement variant “Measuring with external measuring hardware”, the interface on the HLI to the external measuring hardware is made accessible to the user via API functions. The user then has the possibility to use measurement methods for real-time control implemented in the PLC with the kernelv DLL.

The interface to the external measuring hardware consists of the following parts:

Command interface from the CNC to the PLC or to the DLL user.

Acknowledgement interface from the PLC or from the user to the CNC:

Trigger interface for the measuring event and possibly the measured value from the PLC to the CNC.

The sequence of a measurement run using the external measurement interface is as follows:

The command interface is queried by the function **kernelv\_ax\_get\_ext\_latch\_command()**The command is acknowledged by **kernelv\_ax\_acknowledge\_ext\_latch\_command()**The latch event is set and if required the latch value is tracked by **kernelv\_ax\_set\_ext\_latch\_event()** or by **kernelv\_ax\_set\_ext\_latch\_event\_pos()**.



When the external measurement interface is used, the internal measurement simulation for the axis must be deactivated by setting the axis parameter P-AXIS-00112 to the value 0.

### 3.67.1 kernelv\_ax\_get\_ext\_latch\_command()

#### Prototype

```
KERNELV_RETURN kernelv_ch_get_active_kin_id(unsigned long int axIndex,
                                           KERNELV_EXT_LATCH_COMMAND_DATA *data);
```

#### Description

Returns the content of the command interface for the external measuring hardware to the structure KERNELV\_EXT\_LATCH\_COMMAND\_DATA.

#### Parameter

| Name    | Type                             | Meaning  |
|---------|----------------------------------|--|
| axIndex | unsigned long                    | Index of the axis.   |
| data    | KERNELV_EXT_LATCH_COMMAND_DATA * | Pointer to the structure of type KERNELV_EXT_LATCH_COMMAND_DATA to return the value. |

#### Return values

| Symbol           | Value | Meaning  |
|------------------|-------|--|
| RET_FINISHED     | 0     | The function was executed without error.   |
| REST_BUSY        | 1     | The function is currently being executed, but has not yet been completed. The API function must continue to be called. |
| ERR_INVALID_AX   | -9    | The transferred axis index is higher than the number of configured axes -1   |
| ERR_CNC_NOT_INIT | -3    | The simulation CNC kernel was not yet initialised.   |

### 3.67.2 kernelv\_ax\_acknowledge\_ext\_latch\_command()

#### Prototype

```
KERNELV_RETURN kernelv_ax_acknowledge_ext_latch_command(unsigned long int
                                                         axIndex);
```

#### Description

Acknowledges any pending measuring command to the external measuring hardware. If a command is active and was acknowledged, the return value is the function RET\_FINISHED. Otherwise the return value is ERR\_NO\_DATA.

#### Parameter

| Name    | Type          | Meaning            |
|---------|---------------|--------------------|
| axIndex | unsigned long | Index of the axis. |

#### Return values

| Symbol       | Value | Meaning  |
|--------------|-------|--|
| RET_FINISHED | 0     | The function was executed without error.   |
| RET_BUSY     | 1     | The function is currently being executed, but has not yet been completed. The API function must continue to be called. |

|                  |     |  |
|------------------|-----|--|
| ERR_INVALID_AX   | -9  | The transferred axis index is higher than the number of configured axes -1 |
| ERR_CNC_NOT_INIT | -3  | The simulation CNC kernel was not yet initialised.                         |
| ERR_CNC_NO_DATA  | -28 | The requested data is not available.                                       |

### 3.67.3 kernelv\_ax\_set\_ext\_latch\_event\_pos()

**Prototype**

```
KERNELV_RETURN kernelv_ax_set_ext_latch_event_pos(unsigned long int axIndex, signed long int DATA position);
```

**Description**

Indicates the occurrence of the latch event to the CNC and transfers the latched position.

The latched position must be transferred in 0.1 um or 1E-4°.

**Parameter**

| Name     | Type              | Meaning  |
|----------|-------------------|--|
| axIndex  | unsigned long int | Index of the axis.   |
| position | signed long int   | Latch position in 0.1 µm for translatory axes or 1*10 <sup>-4</sup> ° for spindles or rotary axes. |

**Return values**

| Symbol           | Value | Meaning   |
|------------------|-------|---|
| RET_FINISHED     | 0     | The function was executed without error.  |
| ERR_INVALID_AX   | -9    | The transferred channel index is higher than the number of configured channels -1 |
| ERR_CNC_NO_DATA  | -28   | No latch command is active.   |
| ERR_CNC_NOT_INIT | -3    | The simulation CNC kernel was not yet initialised.                                |

### 3.67.4 kernelv\_ax\_set\_ext\_latch\_event()

**Prototype**

```
KERNELV_RETURN kernelv_ax_set_ext_latch_event(unsigned long int axIndex);
```

**Description**

Indicates the occurrence of the latch event to the CNC as a latched value; the current actual position of the axis is transferred.

**Parameter**

| Name    | Type              | Meaning            |
|---------|-------------------|--------------------|
| axIndex | unsigned long int | Index of the axis. |

**Return values**

| Symbol       | Value | Meaning                                  |
|--------------|-------|--|
| RET_FINISHED | 0     | The function was executed without error. |

|                  |     |   |
|------------------|-----|---|
| ERR_INVALID_AX   | -9  | The transferred channel index is higher than the number of configured channels –1 |
| ERR_CNC_NO_DATA  | -28 | No latch command is active.   |
| ERR_CNC_NOT_INIT | -3  | The simulation CNC kernel was not yet initialised.                                |



### 3.68 kernelv\_ch\_get\_timer()

**Prototype**

KERNELV\_RETURN kernelv\_ch\_get\_timer (unsigned long int chanIndex, unsigned short int timerId, unsigned long int\* time);

**Description**

This returns the time in ms and is stored in the specified channel in the variable V.G.TIMER[<counter\_number>].

**Parameter**

| Name      | Type               | Meaning   |
|-----------|--------------------|---|
| chanIndex | unsigned long int  | Channel index   |
| timerId   | unsigned short int | Counter number for the variable V.G.TIMER[]                 |
| Time      | unsigned long int* | Pointer to the memory area where the time in ms is written. |

**Return values**

| Symbol                | Value | Meaning  |
|-----------------------|-------|--|
| RET_FINISHED          | 0     | The function was executed without error.   |
| ERR_INVALID_CHAN      | -1    | The transferred channel index is higher than the number of configured channels -1.               |
| ERR_CNC_NOT_INIT      | -3    | The simulation CNC kernel was not yet initialised.   |
| ERR_INVALID_PARAMETER | -30   | The transferred parameter timerId is invalid.<br>Valid range for timerID:<br>0 <= timerID <= 127 |
| ER_NULL_PARAMETER     | -35   | The timer pointer is not referenced.   |

### 3.69 kernelv\_get\_production\_time()

**Prototype**

KERNELV\_RETURN kernelv\_get\_production\_time(double\* productionTime, KERNELV\_PT\_FILE files);

**Description**

Starts an NC program in the corresponding channel for each string that is not empty in the KERNELV\_PT\_FILES structure. The first element is started in the first channel, the second element in the second channel, etc.

If no program is started in a channel, the string must be 0 at the appropriate index.

Only configured programs can be started in channels.

The total processing time of all started NC programs is returned.

**Parameter**

| Name           | Type              | Meaning   |
|----------------|-------------------|---|
| productionTime | double*           | Pointer to the memory area where the production time is written in s. |
| files          | KERNELV_PT_FILE S | Structure where the names of NC programs are stored.                  |

**Return values**

| Symbol             | Value | Meaning  |
|--------------------|-------|--|
| RET_FINISHED       | 0     | The transferred NC programs have all been successfully ended.                      |
| ERR_INVALID_CHAN   | -1    | The transferred channel index is higher than the number of configured channels -1. |
| ERR_CNC_NOT_INIT   | -3    | The simulation CNC kernel was not yet initialised.                                 |
| ERR_INTERNAL_ERROR | -11   | An internal error has occurred in the DLL. The value could not be written.         |
| ERR_NC_PROGRAM     | -32   | An error has occurred in the NC program.   |
| ERR_CH_ERROR_STATE | -33   | The channel is in error state.   |

## 3.70 kernelv\_diagnosis\_upload()

**Prototype**

KERNELV\_RETURN kernelv\_diagnosis\_upload(char\* filename);

**Description**

Starts uploading the internal diagnosis data of the DLL and writes it to the file with the name filename. If no filename is specified, the data is saved to diag\_data.txt. If filename contains no path, the file is written to the current path.

**Parameter**

| Name     | Type  | Meaning                                    |
|----------|-------|--|
| Filename | char* | Name of the file to which data is written. |

**Return values**

| Symbol           | Value | Meaning  |
|------------------|-------|--|
| RET_FINISHED     | 0     | The upload is finished.                            |
| ERR_CNC_NOT_INIT | -3    | The simulation CNC kernel was not yet initialised. |

## 4 kernelv API types

All listed types are defined in the file `kernelv.h`.

### 4.1 Enum KERNELV\_RETURN

#### Description

Return values and error codes of API functions.

| Symbol                 | Value | Meaning   |
|------------------------|-------|---|
| RET_FINISHED           | 0     | The function was executed without error.  |
| RET_BUSY               | 1     | The function is currently being executed, but has not yet been completed.<br>The API function must continue to be called.   |
| ERR_INVALID_CHAN       | -1    | The transferred channel index is higher than the number of configured channels -1.  |
| ERR_PROG_NAME_LENGTH   | -2    | The transferred program name is longer than permitted.  |
| ERR_CNC_NOT_INIT       | -3    | The simulation CNC kernel was not yet initialised.  |
| ERR_CNC_RET_MEMORY     | -4    | The return value(s) does/do not fit into the memory provided.   |
| ERR_INVALID_STATE      | -5    | The CNC channel is in the wrong state for executing a function.   |
| ERR_DOUBLE_KERNEL      | -6    | An instance of the simulation CNC is running and uses the same instance prefix. This may occur if 2 instances of kernelv were started with the function call <code>kernelv_startup()</code> or the same instance prefix was used several times when <code>kernelv_startup_instance()</code> was called. |
| ERR_SHM_STARTUP        | -7    | Internally used shared memories could not be created when starting.   |
| ERR_STARTUP            | -8    | An error occurred on starting the simulation CNC. Possible causes are missing parameter lists or incorrect entries in parameter lists.  |
| ERR_INVALID_AX         | -9    | The transferred axis index is higher than the number of configured axes -1  |
| ERR_AXIS_ERROR         | -10   | The CNC axis indicates an error. The CNC additionally issues an error message.  |
| ERR_INTERNAL_ERROR     | -11   | An internal error has occurred in the DLL.  |
| ERR_UNKNOWN_VARIABLE   | -12   | The variable name is not known in the CNC kernel.   |
| ERR_VARIABLE_SYNTAX    | -13   | The variable name is syntactically incorrect, e.g. no closing bracket in the case of an array variable.   |
| ERR_DATA_TYPE_MISMATCH | -14   | On write access to a variable, the transferred data type does not match the data type used internally in the CNC.   |

| Symbol                       | Value | Meaning   |
|------------------------------|-------|---|
| ERR_UNKNOWN_TECHNO_TYPE      | -15   | An invalid type was specified for a technology function when its execution time was set.  |
| ERR_INVALID_TECHNO_PARAMETER | -16   | An invalid parameter was transferred when the execution time was set for a technology function, e.g. transferred number of the M or H function is higher than the maximum permitted number. |
| ERR_NO_LICENSE               | -17   | No license was found for the use of the kernelv DLL.  |
| ERR_VAR_NAME_LENGTH          | -18   | The variable name passed on to the function exceeds the maximum permissible length ( <code>KERNELV_VAR_NAME_LENGTH</code> ).  |
| ERR_REGISTRY_ACCESS          | -19   | An error occurred when an attempt was made to read values from the Windows registry.  |
| ERR_UNKNOWN_OPTION           | -20   | An unknown option was transferred to the function <code>kernelv_set_options()</code> .  |

|                         |     |   |
|-------------------------|-----|---|
| ERR_ARRAY_NOT_SUPPORTED | -21 | With many CNC real-time variants, an array can be read or written 'en block' by omitting the array index at access. The kernelv DLL does not currently support this access type.  |
| ERR_VAR_NOT_WRITEABLE   | -22 | An attempt was made to write a non-writeable variant. For write access to variables, the same access rules apply as within an NC program. The only exception here are V.E. variables. They can always be written irrespective of the access rights configured.  |
| ERR_PREFIX_TOO_LONG     | -23 | When the function kernelv_startu_prefix() was called, the transferred instance identifier is too long with the result that the internally generated names for the shared memories used no longer fit in the memory provided. . The permitted length is defined in the preprocessor constant<br>KERNELV_INSTANCE_PREFIX_MAX_LEN. |
| ERR_DOUBLE_INSTANCE     | -24 | A kernelv instance was already started from this DLL. It is not possible to start several kernelv instances from an application.  |
| ERR_INVALID_START_MODE  | -25 | An invalid start mode was transferred when the function kernelv_ch_program_start() was started. For valid execution mode values, see E_KERNELV_PROG_START_MODE.   |
| ERR_READ_ERR            | -26 | An error occurred on reading data from the kernelv DLL.   |
| ERR_WRITE_ERR           | -27 | An error occurred when data is written to the kernelv DLL.  |
| ERR_CNC_NO_DATA         | -28 | The requested data is not available.  |
| ERR_TECHNO_NOT_FOUND    | -29 | The specified technology function was not found.  |
| ERR_INVALID_PARAMETER   | -30 | An invalid parameter was transferred.   |
| ERR_STARTUP_CHAN_INIT   | -31 | When kernelv-DLL was started, it was not possible to execute the initialisation of the configured NC channels.  |

| Symbol             | Value | Meaning   |
|--------------------|-------|---|
| ERR_NC_PROGRAM     | -32   | An error has occurred in the NC program.          |
| ERR_CH_ERROR_STATE | -33   | The channel is in error state.                    |
| ERR_TIME_OUT       | -34   | The function could not be ended within the limit. |
| ERR_NULL_PARAMETER | -35   | An unreferenced pointer was transferred.          |

## 4.2 KERNELV\_CHANNEL\_STATE

Enumeration CNC\_SIMU\_CHANNEL\_STATE describes the state of a CNC channel.

The state of a CNC channel is described by the following state diagram:

| Symbol                    | Value | Meaning   |
|---------------------------|-------|---|
| KERNELV_STATE_DESELECTED  | 1     | Channel state is DESELECTED.  |
| KERNELV_STATE_SELECTED    | 2     | Channel state is SELECTED.  |
| KERNELV_STATE_READY       | 3     | Channel state is READY.   |
| KERNELV_STATE_ACTIVE      | 4     | Channel state is ACTIVE, and an NC program is currently being executed. |
| KERNELV_STATE_HOLD        | 5     | Channel state is HOLD. An NC program has been started and then stopped. |
| KERNELV_STATE_ERROR       | 6     | Channel state is ERROR.   |
| KERNELV_STATE_SELECTING   | 7     | Channel state is SELECTING.   |
| KERNELV_STATE_DESELECTING | 8     | Channel state is DESELECTING.   |
| KERNELV_STATE_PREPARING   | 9     | Channel state is PREPARING.   |
| KERNELV_STATE_CLEARING    | 10    | Channel state is CLEARING.  |
| KERNELV_STATE_STARTING    | 11    | Channel state is STARTING. An NC program is currently being started.    |
| KERNELV_STATE_ABORTING    | 12    | Channel state is ABORTING. An NC program is currently being aborted.    |
| KERNELV_STATE_STOPPING    | 13    | Channel state is STOPPING. A running NC program is stopped.             |
| KERNELV_STATE_RESUMING    | 14    | Channel state is RESUMING. A stopped NC program is resumed.             |
| KERNELV_STATE_RESETTING   | 15    | Channel state is RESETTING. A reset is currently taking place.          |

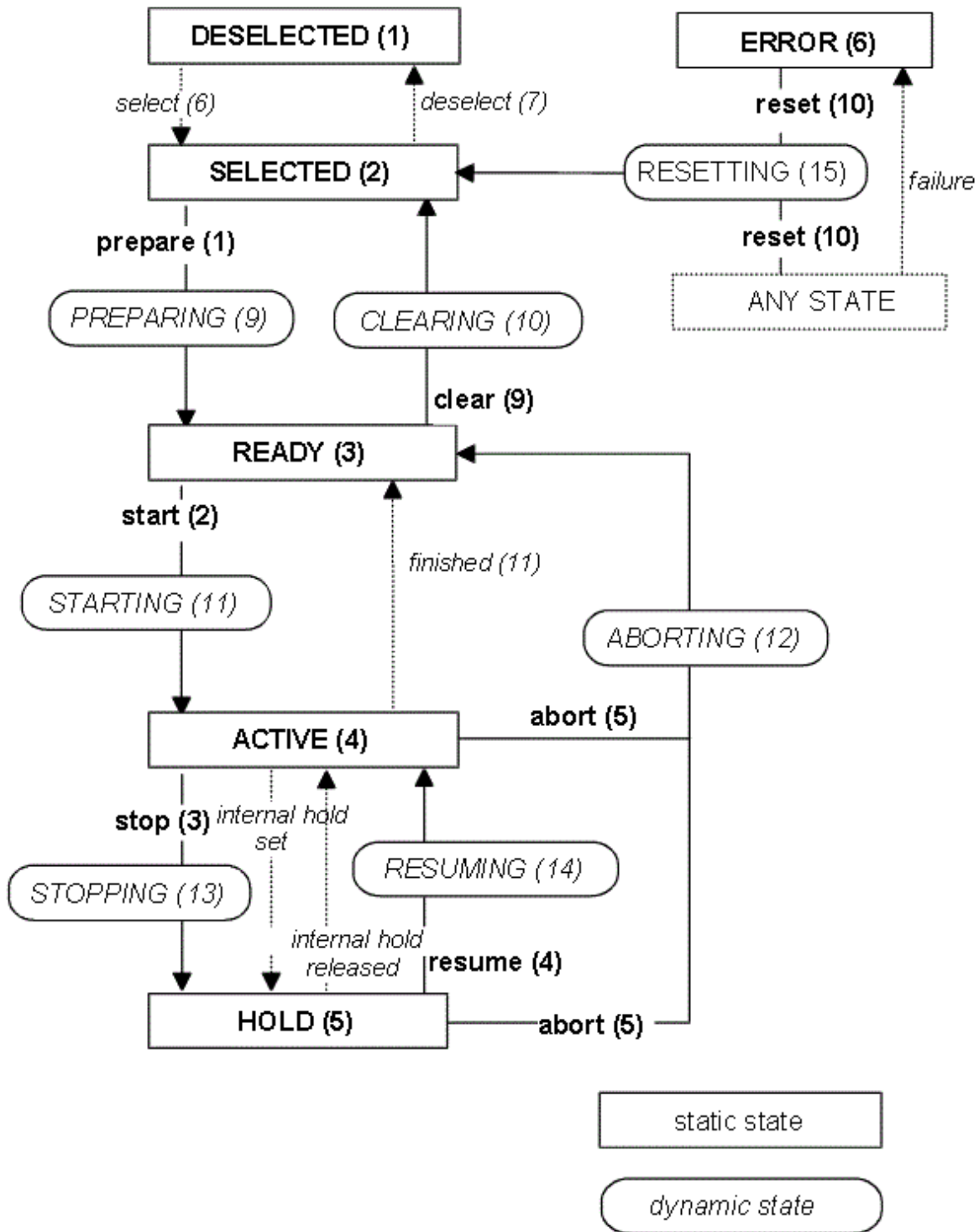


Fig. 5: State diagram of a CNC channel

## 4.3 Enum E\_KERNELV\_TECHNO\_TYPE

### Description

Type of a technology function stored in the KERNELV\_TECHNO\_DATA structure.

| Symbol                | Value | Meaning  |
|-----------------------|-------|--|
| KERNELV_TECHNO_EMPTY  | 0     | The structure does not contain any valid values.                     |
| KERNELV_TECHNO_M_CODE | 1     | The structure contains an M function.                                |
| KERNELV_TECHNO_H_CODE | 2     | The structure contains an H function.                                |
| KERNELV_TECHNO_S_CODE | 3     | The structure contains a spindle techno function, e.g. M3, M19 or S. |
| KERNELV_TECHNO_T_CODE | 4     | The structure contains a tool techno function.                       |

## 4.4 Struct KERNELV\_TECHNO\_DATA

### Description

Structure with technology data acknowledged by the CNC kernel.

### Memory orientation

The individual structure elements are packed in the memory

| Element | Type                   | Meaning  |
|---------|------------------------|--|
| type    | E_KERNELV_TECHNO_TYPE  | Type of the technology function stored in the param element. |
| param   | U_KERNELV_TECHNO_PARAM | Union with the data of the technology function.              |

## 4.5 KERNELV\_CHANNEL\_TECHNO\_DATA\_ARRAY

### Description

Defines an array of size `KERNELV_CHANNEL_TECHNO_DATA_COUNT` for structures of the type `KERNELV_TECHNO_DATA`.

```
typedef KERNELV_TECHNO_DATA
KERNELV_CHANNEL_TECHNO_DATA_ARRAY[KERNELV_CHANNEL_TECHNO_DATA_COUNT];
```

A variable of this type can be transferred to the functions `kernelv_ch_get techno_data()` or `kernelv_ch_get_new techno_data()` to read technology information.

```
KERNELV_CHANNEL_TECHNO_DATA_ARRAY ch_techno;

unsigned long int          techno_len;

if ( kernelv_ch_get techno_data(0,
                               ch_techno,
                               sizeof(ch_techno),
                               &techno_len) == RET_FINISHED)

{
for (int i = 0; i < KERNELV_CHANNEL_TECHNO_DATA_COUNT, i++)
.  .printf("Type: %d\n", ch_techno[i].type);
}
}
```



## 4.6 KERNELV\_CHANNEL\_TECHNO\_DATA\_ARRAY2

### Description

Defines an array of size `KERNELV_CHANNEL_TECHNO_DATA_COUNT` for structures of type `KERNELV_TECHNO_DATA2`.

```
typedef KERNELV_TECHNO_DATA2
```

```
KERNELV_CHANNEL_TECHNO_DATA_ARRAY2[KERNELV_CHANNEL_TECHNO_DATA_COUNT];
```

A variable of this type can be transferred to the functions `kernelv_ch_get techno_data2()` or `kernelv_get_new techno_data2()` to read technology information.

```
KERNELV_CHANNEL_TECHNO_DATA_ARRAY2 ch_techno;
```

```
unsigned long int techno_len;
```

```
if ( kernelv_ch_get techno_data2(0,
                                ch_techno,
                                sizeof(ch_techno),
                                &techno_len) == RET_FINISHED)
{
for (int i =0; i < KERNELV_CHANNEL_TECHNO_DATA_COUNT, i++)
. .printf("Type: %d\n", ch_techno[i].type);
}
```

## 4.7 KERNELV\_AXIS\_TECHNO\_DATA\_ARRAY

### Description

Defines an array of size `KERNELV_AXIS_TECHNO_DATA_COUNT` for structures of the type `KERNELV_TECHNO_DATA`.

```
typedef KERNELV_TECHNO_DATA
KERNELV_AXIS_TECHNO_DATA_ARRAY[KERNELV_AXIS_TECHNO_DATA_COUNT];
```

A variable of this type can be transferred to the functions `kernelv_ch_get techno_data()` or `kernelv_ch_get_new techno_data()` to read technology information.

```
KERNELV_CHANNEL_TECHNO_DATA_ARRAY ax techno;

unsigned long int techno_len;
```

```
if ( kernelv_ax_get techno_data(0,
                               ax techno,
                               sizeof(ch techno),
                               &techno_len) == RET_FINISHED)

{
for (int i =0; i < KERNELV_CHANNEL_TECHNO_DATA_COUNT ,i++)
. .printf("Type: %d\n", ax techno[i].type);
}
```

## 4.8 Struct KERNELV\_TECHNO\_DATA2

### Description

Structure with technology data acknowledged by the CNC kernel.

### Memory orientation

The individual structure elements are packed in the memory

| Element | Type                    | Meaning  |
|---------|-------------------------|--|
| type    | E_KERNELV_TECHNO_TYPE   | Type of the technology function stored in the param element. |
| param   | U_KERNELV_TECHNO_PARAM2 | Union with the data of the technology function.              |

## 4.9 KERNELV\_CHANNEL\_TECHNO\_DATA\_ARRAY2

### Description

Defines an array of size `KERNELV_CHANNEL_TECHNO_DATA_COUNT` for structure of type `KERNELV_TECHNO_DATA2`.

```
typedef KERNELV_TECHNO_DATA2
KERNELV_CHANNEL_TECHNO_DATA_ARRAY2[KERNELV_CHANNEL_TECHNO_DATA_COUNT];
```

A variable of this type can be transferred to the functions `kernelv_ch_get techno_data2()` or `kernelv_ch_get_new techno_data2()` to read technology information.

```
KERNELV_CHANNEL_TECHNO_DATA_ARRAY2 ch_techno;
unsigned long int techno_len;
if ( kernelv_ch_get techno_data2(0,
                                ch_techno,
                                sizeof(ch_techno),
                                &techno_len) == RET_FINISHED)
{
for (int i =0; i < KERNELV_CHANNEL_TECHNO_DATA_COUNT ,i++)
. .printf("Type: %d\n", ch_techno[i].type);
}
```

## 4.10 KERNELV\_AXIS\_TECHNO\_DATA\_ARRAY2

### Description

Defines an array of size `KERNELV_AXIS_TECHNO_DATA_COUNT` for structures of type `KERNELV_TECHNO_DATA2`.

```
typedef KERNELV_TECHNO_DATA2
KERNELV_AXIS_TECHNO_DATA_ARRAY2[KERNELV_AXIS_TECHNO_DATA_COUNT];
```

A variable of this type can be transferred to the functions `kernelv_ax_get techno_data2()` or `kernelv_ax_get_new techno_data2()` to read technology information.

```
KERNELV_CHANNEL_TECHNO_DATA_ARRAY2 ax_techno;

unsigned long int          techno_len;

if ( kernelv_ax_get techno_data2(0,
                                ax_techno,
                                sizeof(ch_techno),
                                &techno_len) == RET_FINISHED)

{
for (int i =0; i < KERNELV_AXIS_TECHNO_DATA_COUNT ,i++)
. .printf("Type: %d\n", ch_techno[i].type);
}
```

## 4.11 Union U\_KERNELV\_TECHNO\_PARAM

### Description

Union with the data of a technology function.

### Memory orientation

The individual structure elements are packed in the memory.

| Element | Type          | Meaning                               |
|---------|---------------|---------------------------------------|
| m_h     | M_H_CODE_DATA | Data of an M/H technology function    |
| spindle | S_CODE_DATA   | Data of a spindle technology function |
| tool    | T_CODE_DATA   | Data of a tool function.              |

## 4.12 Union U\_KERNELV\_TECHNO\_PARAM2

### Description

Union with the data of a technology function.

### Memory orientation

The individual structure elements are packed in the memory.

| Element | Type           | Meaning                                |
|---------|----------------|--|
| m_h     | M_H_CODE_DATA2 | Data of an M/H technology function.    |
| spindle | S_CODE_DATA    | Data of a spindle technology function. |
| tool    | T_CODE_DATA    | Data of a tool function.               |

## 4.13 Struct M\_H\_CODE\_DATA

### Description

Data of an M/H function.

### Memory orientation

The individual structure elements are packed in the memory.

| Element  | Type              | Meaning   |
|----------|-------------------|---|
| nr       | unsigned long int | Number of the M/H function.                                   |
| duration | unsigned long int | Set execution time in us when execution simulation is active. |

## 4.14 Struct M\_H\_CODE\_DATA2

### Description

Data of an M/H function.

### Memory orientation

The individual structure elements are packed in the memory.

| Element   | Type              | Meaning  |
|-----------|-------------------|--|
| nr        | unsigned long int | Number of the M/H function.                                    |
| duration  | unsigned long int | Set execution time in us when execution simulation is active.  |
| add_value | signed long int   | Value additionally programmed in the NC program (M4711 = 123). |
| fillup    | unsigned long int | Fill bytes for 8-byte alignment.                               |

## 4.15 Enum E\_KERNELV\_SPINDLE\_TYPE

### Description

Type of a technology function stored in the S\_CODE\_DATA structure.

| Symbol                 | Value | Meaning  |
|------------------------|-------|--|
| KERNELV_TECHNO_S_EMPTY | 0     | The structure does not contain any valid values. |

|                      |   |   |
|----------------------|---|---|
| KERNELV_TECHNO_S_M3  | 1 | The structure contains an M3 spindle function.  |
| KERNELV_TECHNO_S_M4  | 2 | The structure contains an M4 spindle function.  |
| KERNELV_TECHNO_S_M5  | 3 | The structure contains an M5 spindle function.  |
| KERNELV_TECHNO_S_M19 | 4 | The structure contains an M19 spindle function. |

## 4.16 Struct S\_CODE\_DATA

### Description

The structure contains the data belonging to a spindle technology function.

### Memory orientation

The individual structure elements are packed in the memory.

| Element     | Type                   | Meaning  |
|-------------|------------------------|--|
| type        | E_KERNELV_SPINDLE_TYPE | Data type of the spindle technology function.                        |
| axis_number | unsigned short int     | Axis number of the axis to which the technology function was output. |
| revolutions | unsigned long int      | Spindle speed  |
| position    | signed long int        | Target position of the spindle if the spindle is positioned.         |
| duration    | unsigned long int      | Set execution time in us when execution simulation is active.        |

## 4.17 Struct T\_CODE\_DATA

### Description

The structure contains the data belonging to a tool technology function.

### Memory orientation

The individual structure elements are packed in the memory.

| Element | Type            | Meaning                    |
|---------|-----------------|----------------------------|
| basic   | signed long int | Basic number of the tool.  |
| sister  | signed long int | Number of the sister tool. |
| variant | signed long int | Variant number of the tool |

## 4.18 Struct KERNELV\_NC\_LINE\_DATA

### Description

The structure of the data belonging to an executed NC program line.

### Memory orientation

The individual structure elements are packed in the memory.

| Element      | Type                               | Meaning   |
|--------------|------------------------------------|---|
| fileoffset   | unsigned long int                  | File offset of the executed line in the currently active NC program file  |
| ncLineNumber | unsigned long int                  | NC line number of the executed NC line, if programmed. If no NC line number was programmed, the last line number programmed is entered. |
| filename     | char[KERNELV_FILE_NAME_LENGTH + 1] | File name of the file that contains the NC line just executed.  |

## 4.19 Enum E\_KERNELV\_VAR\_TYPE

### Description

Data type of a data item stored in the structure U\_KERNELV\_VAR\_VALUE.

| Symbol                   | Value | Meaning   |
|--------------------------|-------|---|
| KERNELV_VAR_TYPE_UNKNOWN | 1     | The structure does not contain any valid data type.   |
| KERNELV_VAR_TYPE_BOOLEAN | 2     | Data type is unsigned char. Possible values 0 or 1.   |
| KERNELV_VAR_TYPE_UNSO8   | 3     | Data type is unsigned char.   |
| KERNELV_VAR_TYPE_SGN08   | 4     | Data type is signed char.   |
| KERNELV_VAR_TYPE_UNSO16  | 5     | Data type is unsigned short int.  |
| KERNELV_VAR_TYPE_SGN16   | 6     | Data type is signed short int.  |
| KERNELV_VAR_TYPE_UNSO32  | 7     | Data type is unsigned long int.   |
| KERNELV_VAR_TYPE_SGN32   | 8     | Data type is signed long int.   |
| KERNELV_VAR_TYPE_DOUBLE  | 9     | Data type is double.  |
| KERNELV_VAR_TYPE_STRING  | 10    | The data type is a string with a maximum of 127 (KERNELV_VAR_STRING_LEN) characters (total length including terminating 0 KERNELV_VAR_STRING_LEN +1 character). |



## 4.20 Union U\_KERNELV\_VAR\_VALUE

### Description

Union with the possible values of a CNC variable.

### Memory orientation

The individual structure elements are packed in the memory.

| Element | Type identifier<br>E_KERNELV_VAR_TYPE | Meaning  |
|---------|---------------------------------------|--|
| boolean | KERNELV_VAR_TYPE_BOOLEAN              | Data type is unsigned char. Possible values 0 or 1.  |
| uns08   | KERNELV_VAR_TYPE_UNSO8                | Data type is unsigned char.  |
| sgn08   | KERNELV_VAR_TYPE_SGN08                | Data type is signed char.  |
| uns16   | KERNELV_VAR_TYPE_UNSI16               | Data type is unsigned short int.   |
| sgn16   | KERNELV_VAR_TYPE_SGNI16               | Data type is signed short int.   |
| uns32   | KERNELV_VAR_TYPE_UNSI32               | Data type is unsigned long int.  |
| sgn32   | KERNELV_VAR_TYPE_SGNI32               | Data type is signed long int.  |
| real64  | KERNELV_VAR_TYPE_DOUBLE               | Data type is double.   |
| string  | KERNELV_VAR_TYPE_STRING               | The data type is a string with a maximum of 127 (KERNELV_VAR_STRING_LEN) characters (total length including terminating zero KERNELV_VAR_STRING_LEN +1 character). |

## 4.21 Struct KERNELV\_VARIABLE

### Description

The structure contains the value and type of a CNC variable.

### Memory orientation

The individual structure elements are packed in the memory.

| Element | Type                | Meaning  |
|---------|---------------------|--|
| type    | E_KERNELV_VAR_TYPE  | Data type of the data item in the value structure element. |
| value   | U_KERNELV_VAR_VALUE | Union with the data types possible for CNC variables.      |

## 4.22 Struct KERNELV\_NC\_LINE\_DATA

### Description

The structure contains information about the NC program lines executed in the current cycle.

### Memory orientation

The individual structure elements are packed in the memory.

| Element      | Type                               | Meaning  |
|--------------|------------------------------------|--|
| fileoffset   | unsigned long int                  | File offset of the lines in the NC program   |
| ncLineNumber | unsigned long int                  | NC line number of the line. The line number defined in the NC program (N followed by a numeric value) is returned. If no NC line number was specified in the executed line, the last programmed line number is returned. |
| filename     | char[KERNELV_FILE_NAME_LENGTH + 1] | File name of the NC program that is currently executed.  |

## 4.23 Struct KERNELV\_LICENSE\_INFO

### Description

The structure contains information about the licensed options.

### Memory orientation

The individual structure elements are packed in the memory.

| Element     | Type              | Meaning   |
|-------------|-------------------|---|
| CncSysIdOK  | unsigned long int | Valid license exists.   |
| CncAxesPack | unsigned long int | Maximum configurable number of axes.  |
| CncChannels | unsigned long int | Maximum configurable number of channels.  |
| CncTrafo    | unsigned long int | Transformation pack is licensed.  |
| CncSpline   | unsigned long int | Spline pack is licensed.  |
| CncSpline   | unsigned long int | The license is an export license (function restrictions due to export regulations). |
| CncDll      | unsigned long int | Use of the kernelv DLL is licensed.   |

## 4.24 Struct KERNELV\_DECODER\_POSITION\_HEADER

### Description

the structure contains general data about the axis positions read by the decoder.

When the function `kernelv_ch_get_decoder_positions()` is called, a structure of type `KERNELV_DECODER_POSITION_HEADER` is first returned. This is followed by a structure of type `KERNELV_DECODER_POSITION_DATA` for each axis present in the channel.

### Memory orientation

The individual structure elements are packed in the memory.

| Element     | Type               | Meaning  |
|-------------|--------------------|--|
| data_valid  | unsigned char      | Validity identifier for the following data.  |
| line_number | unsigned long int  | NC line number.  |
| Block_count | unsigned long int  | Block counter.   |
| axis_count  | unsigned short int | Number of axes in the channel, number of the following structures of type <code>KERNELV_DECODER_POSITION_DATA</code> . |

## 4.25 Struct KERNELV\_DECODER\_POSITION\_DATA

### Description

The structure contains the decoder axis positions of an axis.

### Memory orientation

The individual structure elements are packed in the memory.

| Element           | Type               | Meaning   |
|-------------------|--------------------|---|
| mcs_pos           | unsigned long int  | Axis position in the machine coordinate system.                     |
| pcs_pos_no_offset | unsigned long int  | Axis position in the programming coordinate system without offsets. |
| pcs_pos_offset    | unsigned long int  | Axis position in the programming coordinate system with offsets..   |
| axis_number       | unsigned short int | Axis number of the axis.  |

## 4.26 Enum E\_KERNELV\_PROG\_START\_MODE

### Description

Indicates the execution mode of the program start.

| Symbol                          | Value | Meaning  |
|---------------------------------|-------|--|
| KERNELV_START_MODE_NORMAL       | 0     | Normal execution                                 |
| KERNELV_START_MODE_CONTOUR_VISU | 1     | Execution mode of command contour visualisation. |

## 4.27 Struct ACTIVE\_G\_CODES

### Description

Contains an array with the active G functions of each G function group.

| Element                    | Type      | Meaning                |
|----------------------------|-----------|------------------------|
| group[KERNELV_G_MAX_G_GRP] | short int | Array of the G groups. |

## 4.28 Enum E\_KERNELV\_G\_GROUP\_TYPE

### Description

Enumeration with identifiers for the various G function types.

| Symbol                      | Value | Meaning  |
|-----------------------------|-------|--|
| KERNELV_G_PATH_COND         | 0     | Group 0, path condition, possible active G functions: G00, G01, G02, G03, G04, G33, G63, G74, G98, G99, G301, G302, G160 |
| KERNELV_G_PATH_FEED         | 1     | Group 1, possible active G functions: G08, G193  |
| KERNELV_G_DEC               | 2     | Group 2, possible active G functions: G09, G900, G901  |
| KERNELV_G_FEED_ADAPT        | 3     | Group 3, possible active G functions: G09, G900, G901  |
| KERNELV_G_ACTIVE_PLANE      | 4     | Group 4, possible active G functions: G17, G18, G19  |
| KERNELV_G_MIRROR            | 5     | Group 5, possible active G functions: G20, G21, G22, G23, G24, G351  |
| KERNELV_G_TRC_TRANSITION    | 6     | Group 6, possible active G functions: G25, G26   |
| KERNELV_G_TOOL_RADIUS_COMP  | 7     | Group 7, possible active G functions: G40, G41, G42  |
| KERNELV_G_DIAMETER_PROG     | 8     | Group 8, possible active G functions: G40, G41, G42  |
| KERNELV_G_ZERO_POS_SHIFT    | 9     | Group 9, possible active G functions: G53-G59, G159  |
| KERNELV_G_EXACT_STOP        | 10    | Group 10, possible active G functions: G60, G359, G360, G260, G261   |
| KERNELV_G_OVERRRIDE_100     | 11    | Group 11, possible active G functions: G166  |
| KERNELV_G_UNIT              | 12    | Group 12, possible active G functions: G70, G71  |
| KERNELV_G_SUB_CALL          | 13    | Group 13, possible active G functions: G80-G89, G800-G819  |
| KERNELV_G_ABS_REL           | 14    | Group 14, possible active G functions: G90, G91  |
| KERNELV_G_POS_SHIFT         | 15    | Group 15, possible active G functions: G92   |
| KERNELV_G_FEED_PROG         | 16    | Group 16, possible active G functions: G93, G94, G95, G194   |
| KERNELV_G_SPINDEL_FEED      | 17    | Group 17, possible active G functions: G96, G97, G196  |
| KERNELV_G_GEAR_CHANGE       | 18    | Group 18, possible active G functions: G112  |
| KERNELV_G_LOOKAHEAD         | 19    | Group 19, possible active G functions: G115, G116, G117  |
| KERNELV_G_ACC_WEIGHT        | 20    | Group 20, possible active G functions: G130, G131  |
| KERNELV_G_FEEDFORWARD       | 21    | Group 21, possible active G functions: G135, G136, G137  |
| KERNELV_G_TRC_SELECTION     | 22    | Group 22, possible active G functions: G05, G138, G139, G237, G238, G239   |
| KERNELV_G_CIRCE_CENTER      | 23    | Group 23, possible active G functions: G161, G162  |
| KERNELV_G_RADIUS_PROGR      | 24    | Group 24, possible active G functions: G163  |
| KERNELV_G_CIRC_CENTER_CORR  | 25    | Group 25, possible active G functions: G164, G165  |
| KERNELV_G_MANUAL_MODE       | 26    | Group 26, possible active G functions: G200, G201, G202  |
| KERNELV_G_RAMP_TIME_WEIGHT  | 27    | Group 27, possible active G functions: G132, G133, G134  |
| KERNELV_G_SPLINE            | 28    | Group 28, possible active G functions: G150, G151  |
| KERNELV_G_PROBING           | 29    | Group 29, possible active G functions: G100, G101, G102, G106, G107, G108  |
| KERNELV_G_CORNER_DECEL      | 30    | Group 30, possible active G functions: G12, G13  |
| KERNELV_G_CONTOUR_MASKING   | 31    | Group 31, possible active G functions: G140, G141  |
| KERNELV_G_PROBING_INTERRUPT | 32    | Group 32, possible active G functions: G310  |
| KERNELV_G_SPINDLE_OVERRIDE  | 33    | Group 33, possible active G functions: G167  |
| KERNELV_G_RAPID_FEED_WEIGHT | 34    | Group 34, possible active G functions: G129  |

|                      |    |   |
|----------------------|----|---|
| KERNELV_G_CONTOUR    | 35 | Group 35, possible active G functions: G301, G302 |
| KERNELV_G_CYCLE_SYNC | 36 | Group 36, possible active G functions: G66        |
| KERNELV_G_MAX_G_GRP  | 37 | Number of groups, size of array                   |

## 4.29 Data types of contour visualisation

### 4.29.1 Struct CONTOUR\_VISU

**Description**

Data of contour visualisation, return value of the function kernelv\_ch\_get\_cont\_visu\_data().

| Element     | Type                 | Meaning  |                           |
|-------------|----------------------|--|---------------------------|
| count       | signed long int      | Number of the entries with visualisation data contained in the element.                  |                           |
| ifc_version | unsigned long int    | Interface version of the visualisation data; set by the start-up parameter P-STUP-00039. |                           |
|             |                      | ifc_version  | Data type in CONTOUR_VISU |
|             |                      | 0  | CONTOUR_VISU_DATA_V0      |
|             |                      | 1  | CONTOUR_VISU_DATA_V1      |
|             |                      | 2  | CONTOUR_VISU_DATA_V2      |
|             |                      | 3  | CONTOUR_VISU_DATA_V3      |
|             |                      | 4  | CONTOUR_VISU_DATA_V4      |
|             |                      | 5  | CONTOUR_VISU_DATA_V5      |
|             |                      | 6  | CONTOUR_VISU_DATA_V6      |
| 7           | CONTOUR_VISU_DATA_V7 |  |                           |
| 8           | CONTOUR_VISU_DATA_V8 |  |                           |
| data        | CONTOUR_VISU_DATA    | Contour visualisation data corresponding to the interface type specified in ifc_version. |                           |

## 4.29.2 Union CONTOUR\_VISU\_DATA

### Description

Union with possible values of contour visualisation.

| Element       | Type                    | Meaning  |
|---------------|-------------------------|--|
| visu_data_v0  | CONTOUR_VISU_DATA_V0[]  | Contour visualisation data for use of interface version 0, array size CONTOUR_MAX_DATA_V0    |
| visu_data_v1  | CONTOUR_VISU_DATA_V1[]  | Contour visualisation data for use of interface version 1, array size CONTOUR_MAX_DATA_V1.   |
| visu_data_v2  | CONTOUR_VISU_DATA_V2[]  | Contour visualisation data for use of interface version 2, array size CONTOUR_MAX_DATA_V2.   |
| visu_data_v3  | CONTOUR_VISU_DATA_V3[]  | Contour visualisation data for use of interface version 3, array size CONTOUR_MAX_DATA_V3.   |
| visu_data_v4  | CONTOUR_VISU_DATA_V4[]  | Contour visualisation data for use of interface version 4, array size CONTOUR_MAX_DATA_V4.   |
| visu_data_v5  | CONTOUR_VISU_DATA_V5[]  | Contour visualisation data for use of interface version 5, array size CONTOUR_MAX_DATA_V5.   |
| visu_data_v6  | CONTOUR_VISU_DATA_V6[]  | Contour visualisation data for use of interface version 6, array size CONTOUR_MAX_DATA_V6.   |
| visu_data_v7  | CONTOUR_VISU_DATA_V7[]  | Contour visualisation data for use of interface version 7, array size CONTOUR_MAX_DATA_V7.   |
| visu_data_v8  | CONTOUR_VISU_DATA_V8[]  | Contour visualisation data for use of interface version 8, array size CONTOUR_MAX_DATA_V8.   |
| visu_data_v9  | CONTOUR_VISU_DATA_V9[]  | Contour visualisation data for use of interface version 9, array size CONTOUR_MAX_DATA_V9.   |
| visu_data_v10 | CONTOUR_VISU_DATA_V10[] | Contour visualisation data for use of interface version 10, array size CONTOUR_MAX_DATA_V10. |
| visu_data_v11 | CONTOUR_VISU_DATA_V11[] | Contour visualisation data for use of interface version 11, array size CONTOUR_MAX_DATA_V11. |

## 4.29.3 Struct CONTOUR\_VISU\_DATA\_V0

### Description

Channel-specific data of contour visualisation for use of interface version 0.

| Element   | Type                 | Meaning   |
|-----------|----------------------|---|
| ch_data   | CONTOUR_VISU_CH_DATA | Contour visualisation of a channel.                                       |
| ax_data[] | CONTOUR_AXIS_DATA    | Axis-specific visualisation data.<br>Array size: CONTOUR_AXIS_PER_CHANNEL |



### 4.29.4 Struct CONTOUR\_VISU\_DATA\_V1

**Description**

Channel-specific data of contour visualisation for use of interface version 1.

| Element   | Type                   | Meaning   |
|-----------|------------------------|---|
| ch_data   | CONTOUR_VISU_CH_DATA   | Contour visualisation of a channel.                                       |
| file_name | CONTOUR_VISU_FILE_NAME | File name of the active NC program.<br>Array size: FILE_NAME_LN + 1       |
| ax_data[] | CONTOUR_AXIS_DATA      | Axis-specific visualisation data.<br>Array size: CONTOUR_AXIS_PER_CHANNEL |

### 4.29.5 Struct CONTOUR\_VISU\_DATA\_V2

**Description**

Channel-specific data of contour visualisation for use of interface version 2.

| Element    | Type                    | Meaning   |
|------------|-------------------------|---|
| ch_data_v1 | CONTOUR_VISU_CH_DATA_V1 | Contour visualisation of a channel.                                       |
| file_name  | CONTOUR_VISU_FILE_NAME  | File name of the active NC program.<br>Array size: FILE_NAME_LN + 1       |
| ax_data[]  | CONTOUR_AXIS_DATA       | Axis-specific visualisation data.<br>Array size: CONTOUR_AXIS_PER_CHANNEL |

### 4.29.6 Struct CONTOUR\_VISU\_DATA\_V3

**Description**

Channel-specific data of contour visualisation for use of interface version 3.

| Element      | Type                 | Meaning   |
|--------------|----------------------|---|
| ch_data      | CONTOUR_VISU_CH_DATA | Contour visualisation of a channel.                                       |
| ax_data_v1[] | CONTOUR_AXIS_DATA_V1 | Axis-specific visualisation data.<br>Array size: CONTOUR_AXIS_PER_CHANNEL |

## 4.29.7 Struct CONTOUR\_VISU\_DATA\_V4

### Description

Channel-specific data of contour visualisation for use of interface version 4.

| Element      | Type                   | Meaning   |
|--------------|------------------------|---|
| ch_data      | CONTOUR_VISU_CH_DATA   | Contour visualisation of a channel.                                       |
| file_name    | CONTOUR_VISU_FILE_NAME | File name of the active NC program.<br>Array size: FILE_NAME_LN + 1       |
| ax_data_v1[] | CONTOUR_AXIS_DATA_V1   | Axis-specific visualisation data.<br>Array size: CONTOUR_AXIS_PER_CHANNEL |

## 4.29.8 Struct CONTOUR\_VISU\_DATA\_V5

### Description

Channel-specific data of contour visualisation for use of interface version 5.

| Element      | Type                    | Meaning   |
|--------------|-------------------------|---|
| ch_data_v1   | CONTOUR_VISU_CH_DATA_V1 | Contour visualisation of a channel.                                       |
| file_name    | CONTOUR_VISU_FILE_NAME  | File name of the active NC program.<br>Array size: FILE_NAME_LN + 1       |
| ax_data_v1[] | CONTOUR_AXIS_DATA_V1    | Axis-specific visualisation data.<br>Array size: CONTOUR_AXIS_PER_CHANNEL |

## 4.29.9 Struct CONTOUR\_VISU\_DATA\_V6

### Description

Channel-specific data of contour visualisation for use of interface version 6.

| Element      | Type                 | Meaning   |
|--------------|----------------------|---|
| ch_data      | CONTOUR_VISU_CH_DATA | Contour visualisation of a channel.                                       |
| ax_data_v2[] | CONTOUR_AXIS_DATA_V2 | Axis-specific visualisation data.<br>Array size: CONTOUR_AXIS_PER_CHANNEL |

### 4.29.10 Struct CONTOUR\_VISU\_DATA\_V7

**Description**

Channel-specific data of contour visualisation for use of interface version 7.

| Element      | Type                   | Meaning   |
|--------------|------------------------|---|
| ch_data      | CONTOUR_VISU_CH_DATA   | Contour visualisation of a channel.                                       |
| file_name    | CONTOUR_VISU_FILE_NAME | File name of the active NC program.<br>Array size: FILE_NAME_LN + 1       |
| ax_data_v2[] | CONTOUR_AXIS_DATA_V2   | Axis-specific visualisation data.<br>Array size: CONTOUR_AXIS_PER_CHANNEL |

### 4.29.11 Struct CONTOUR\_VISU\_DATA\_V8

**Description**

Channel-specific data of contour visualisation for use of interface version 8.

| Element      | Type                    | Meaning   |
|--------------|-------------------------|---|
| ch_data_v1   | CONTOUR_VISU_CH_DATA_V1 | Contour visualisation of a channel.                                       |
| file_name    | CONTOUR_VISU_FILE_NAME  | File name of the active NC program.<br>Array size: FILE_NAME_LN + 1       |
| ax_data_v2[] | CONTOUR_AXIS_DATA_V2    | Axis-specific visualisation data.<br>Array size: CONTOUR_AXIS_PER_CHANNEL |

### 4.29.12 Struct CONTOUR\_VISU\_DATA\_V9

**Description**

Channel-specific data of contour visualisation for use of interface version 9.

| Element    | Type                    | Meaning   |
|------------|-------------------------|---|
| ch_data_v2 | CONTOUR_VISU_CH_DATA_V2 | Contour visualisation of a channel.                                       |
| file_name  | CONTOUR_VISU_FILE_NAME  | File name of the active NC program.<br>Array size: FILE_NAME_LN + 1       |
| ax_data[]  | CONTOUR_AXIS_DATA       | Axis-specific visualisation data.<br>Array size: CONTOUR_AXIS_PER_CHANNEL |

### 4.29.13 Struct CONTOUR\_VISU\_DATA\_V10

#### Description

Channel-specific data of contour visualisation for use of interface version 10.

| Element      | Type                        | Meaning   |
|--------------|-----------------------------|---|
| ch_data_v2   | CONTOUR_VISU_<br>CH_DATA_V2 | Contour visualisation of a channel.                                       |
| file_name    | CONTOUR_VISU_<br>FILE_NAME  | File name of the active NC program.<br>Array size: FILE_NAME_LN + 1       |
| ax_data_v1[] | CONTOUR_AXIS_<br>DATA_V1    | Axis-specific visualisation data.<br>Array size: CONTOUR_AXIS_PER_CHANNEL |

### 4.29.14 Struct CONTOUR\_VISU\_DATA\_V11

#### Description

Channel-specific data of contour visualisation for use of interface version 7.

| Element      | Type                        | Meaning   |
|--------------|-----------------------------|---|
| ch_data_v2   | CONTOUR_VISU_<br>CH_DATA_V2 | Contour visualisation of a channel.                                       |
| file_name    | CONTOUR_VISU_<br>FILE_NAME  | File name of the active NC program.<br>Array size: FILE_NAME_LN + 1       |
| ax_data_v2[] | CONTOUR_AXIS_<br>DATA_V2    | Axis-specific visualisation data.<br>Array size: CONTOUR_AXIS_PER_CHANNEL |

## 4.29.15 Struct CONTOUR\_VISU\_CH\_DATA

### Description

Channel-specific data of contour visualisation for use of interface versions 0, 1, 3, 4, 6, 7.

| Element                | Type               | Meaning   |
|------------------------|--------------------|---|
| nc_block_nr            | signed long int    | NC block number   |
| fileoffset             | signed long int    | File offset in current program file   |
| channel_nr             | unsigned short int | Channel number  |
| g_function             | signed short int   | Interpolation type:<br>0: Rapid traverse<br>1: Linear interpolation<br>2; 3: Circular interpolation<br>61: Polynomial interpolation<br>-1: Interpolation type not assigned. |
| circle_radius          | unsigned long int  | Circle radius for circulation interpolation in 0.1 um   |
| circle_center_point[2] | double             | Circle centre point for circulation interpolation in 0.1 um.  |

## 4.29.16 Struct CONTOUR\_VISU\_CH\_DATA\_V1

### Description

Channel-specific data of contour visualisation for use of interface versions 2, 5, 8.

| Element                | Type                     | Meaning  |
|------------------------|--------------------------|--|
| nc_block_nr            | signed long int          | NC block number  |
| fileoffset             | signed long int          | File offset in current program file  |
| channel_nr             | unsigned short int       | Channel number   |
| g_function             | signed short int         | Interpolation type:<br>0: Rapid traverse<br>1: Linear interpolation<br>2; 3 Circular interpolation<br>61: Polynomial interpolation<br>-1. Interpolation type not assigned. |
| circle_radius          | unsigned long int        | Circle radius for circulation interpolation in 0.1 um  |
| circle_center_point[2] | double                   | Circle centre point for circulation interpolation in 0.1 um.   |
| v_prog                 | signed long int          | Programmed velocity in 1 um/s.   |
| techno                 | CONTOUR_VISU_DATA_TECHNO | Technology data.   |
| fillup                 | unsigned long int        | In kernelv versions with Build number > 3000, fill bytes for structure alignment.  |

## 4.29.17 Struct CONTOUR\_VISU\_CH\_DATA\_V2

### Description

Channel-specific data of contour visualisation for use of interface versions 9, 10, 11

| Element                | Type                        | Meaning   |
|------------------------|-----------------------------|---|
| nc_block_nr            | signed long int             | NC block number   |
| fileoffset             | signed long int             | File offset in current program file   |
| channel_nr             | unsigned short int          | Channel number  |
| g_function             | signed short int            | Interpolation type:<br>0: Rapid traverse<br>1: Linear interpolation<br>2; 3: Circular interpolation<br>61: Polynomial interpolation<br>-1: Interpolation type not assigned. |
| circle_radius          | unsigned long int           | Circle radius for circulation interpolation in 0.1 um   |
| circle_center_point[2] | double                      | Circle centre point for circulation interpolation in 0.1 um.  |
| v_prog                 | signed long int             | Programmed velocity in 1 um/s.  |
| techno                 | CONTOUR_VISU_DATA_TECHNO_V1 | Technology data.  |
| fillup                 | unsigned long int           | Alignment bytes to force 8-byte alignment.  |

## 4.29.18 Struct CONTOUR\_AXIS\_DATA

### Description

Axis-specific data of contour visualisation.

| Element     | Type               | Meaning                            |
|-------------|--------------------|------------------------------------|
| act_cmd_pos | signed long int    | ACS command position of the axis.  |
| axis_nbr    | unsigned short int | Axis number.                       |
| fillup      | unsigned short int | Fill bytes for structure alignment |

### 4.29.19 Struct CONTOUR\_AXIS\_DATA\_V1

#### Description

Axis-specific data of contour visualisation.

| Element          | Type               | Meaning  |
|------------------|--------------------|--|
| act_cmd_pos      | signed long int    | ACS command position of the axis.  |
| act_cmd_pos_wcs0 | signed long int    | WCS_0 command position of the axis.<br>Only calculated if the channel parameter P-CHAN-00145 has the value 1 and P-CHAN-00032 has a value > 0. |
| axis_nbr         | unsigned short int | Axis number.   |
| fillup           | unsigned short int | Fill bytes for structure alignment   |

### 4.29.20 Struct CONTOUR\_AXIS\_DATA\_V2

#### Description

Axis-specific data of contour visualisation.

| Element          | Type               | Meaning  |
|------------------|--------------------|--|
| act_cmd_pos      | signed long int    | ACS command position of the axis.  |
| act_cmd_pos_wcs0 | signed long int    | WCS_0 command position of the axis. (Command position of the axis in the Cartesian basic coordinate system of the machine).<br>Only calculated if the channel parameter P-CHAN-00145 has the value 1 and P-CHAN-00032 has a value > 0. |
| act_cmd_pos_wcs  | signed long int    | WCS command position of the axis in the current active coordinate system.<br>Only calculated if the channel parameter P-CHAN-00145 has the value 1 and P-CHAN-00032 has a value > 0.   |
| axis_nbr         | unsigned short int | Axis number  |
| fillup           | unsigned short int | Fill bytes for structure alignment.  |

### 4.29.21 Enum E\_CONTOUR\_TECHNO\_TYPE

#### Description

Type of technology function

| Symbol               | Value | Meaning            |
|----------------------|-------|--------------------|
| TECHNO_UNKNOWN_TYPE_ | 0     | Entry not assigned |
| TECHNO_M_TYPE        | 1     | M function         |
| TECHNO_H_TYPE        | 2     | H function         |



### 4.29.22 Struct CONTOUR\_M\_H\_PROCESS

**Description**

Data of a specified technology function.

| Element | Type              | Meaning  |
|---------|-------------------|--|
| nr      | unsigned long int | Number of the technology function.                                 |
| sync    | unsigned long int | Synchronisation mode of the technology function.                   |
| type    | unsigned long int | Type of the technology function according to E_CONTOUR_TECHNO_TYPE |

### 4.29.23 Struct CONTOUR\_M\_H\_PROCESS\_V1

**Description**

Data of a specified technology function.

| Element   | Type              | Meaning  |
|-----------|-------------------|--|
| nr        | unsigned long int | Number of the technology function.                                 |
| sync      | unsigned long int | Synchronisation mode of the technology function.                   |
| type      | unsigned long int | Type of the technology function according to E_CONTOUR_TECHNO_TYPE |
| add_value | signed long int   | Value additionally programmed in the NC program (M4711 = 123).     |

### 4.29.24 Enum E\_CONTOUR\_S\_CMD

**Description**

Type of a spindle technology function.

| Symbol           | Value | Meaning   |
|------------------|-------|---|
| SPDL_CMD_UNKNOWN | 0     | Entry not assigned.                             |
| SPDL_CMD_M3      | 3     | M3 function (rotate spindle clockwise).         |
| SPDL_CMD_M4      | 4     | M4 function (rotate spindle counter-clockwise). |
| SPDL_CMD_M5      | 5     | M5 function (stop spindle).                     |
| SPDL_CMD_M19     | 19    | M19 function (position spindle).                |

### 4.29.25 Struct CONTOUR\_S\_PROCESS

#### Description

Data of a specified spindle function.

| Element    | Type               | Meaning  |
|------------|--------------------|--|
| axis_nr    | unsigned short int | Number of the spindle axis.                                      |
| cmd        | unsigned short int | Spindle command according to E_CONTOUR_S_CMD.                    |
| sync       | unsigned long int  | Synchronisation mode of the spindle function.                    |
| position   | signed long int    | Target position to position spindle in $0.1 \cdot 10^{-3} \circ$ |
| revolution | signed long int    | Spindle speed in $1 \cdot 10^{-3} / s$                           |

### 4.29.26 Struct CONTOUR\_TOOL\_PROCESS

#### Description

Data of a T function output.

| Element | Type            | Meaning  |
|---------|-----------------|--|
| basic   | signed long int | Basic number of the tool.                                  |
| sister  | signed long int | Sister number of the tool. The value -1 means unassigned.  |
| variant | signed long int | Variant number of the tool. The value -1 means unassigned. |

### 4.29.27 Struct CONTOUR\_DATA\_TECHNO

#### Description

Data of the technology function output.

| Element                    | Type                 | Meaning   |
|----------------------------|----------------------|---|
| axis_nr                    | unsigned short int   | Axis number with axis-specific technology functions output, 0 for channel-specific technology functions output. |
| fillup                     | unsigned short int   | Fill bytes for structure alignment.   |
| m_h_count                  | unsigned long int    | Number of assigned entries in the m_h_data[] array.   |
| m_h_data[ ]                | CONTOUR_M_H_PROCESS  | M/H technology data.<br>Array size: CONTOUR_MAX_M_H_DATA  |
| s_count                    | signed long int      | Number of assigned entries in the s_procm_h_data[] array.   |
| s_data[CONTOUR_SPDL_COUNT] | CONTOUR_S_PROCESS    | Data of the spindle technology functions output.<br>Array size: CONTOUR_MAX_SPDL_DATA                           |
| tool                       | CONTOUR_TOOL_PROCESS | Technology data of a tool.  |

## 4.29.28 Struct CONTOUR\_DATA\_TECHNO\_V1

### Description

Data of the technology function output.

| Element                        | Type                     | Meaning   |
|--------------------------------|--------------------------|---|
| axis_nr                        | unsigned short int       | Axis number with axis-specific technology functions output, 0 for channel-specific technology functions output. |
| fillup                         | unsigned short int       | Fill bytes for structure alignment.   |
| m_h_count                      | unsigned long int        | Number of assigned entries in the m_h_data[] array.   |
| m_h_data[ ]                    | CONTOUR_M_H_PROCESS_V1   | M/H technology data.<br>Array size: CONTOUR_MAX_M_H_DATA  |
| s_count                        | signed long int          | Number of assigned entries in the s_procm_h_data[] array.   |
| s_data[CONTOUR_SPDL_CO<br>UNT] | CONTOUR_S_PROCESS        | Data of the spindle technology functions output.<br>Array size: CONTOUR_MAX_SPDL_DATA                           |
| tool                           | CONTOUR_TOOL_PROCES<br>S | Technology data of a tool.  |

## 4.30 Data types of error output

### 4.30.1 Struct KERNELV\_ERROR\_VALUE

#### Description

Additional values output in the error message.

| Element   | Type                        | Meaning   |
|-----------|-----------------------------|---|
| type      | E_KERNELV_ERR_VAL_TYPE      | Data type of the value contained in the data element. |
| dimension | E_KERNELV_ERR_VAL_DIMENSION | Dimension of the value contained in the data element. |
| meaning   | E_KERNELV_ERR_VAL_MEANING   | Meaning of the value contained in the data element.   |
| fillup    | unsigned long int           | Fill bytes for structure alignment.                   |
| data      | U_KERNELV_ERR_VAL_DATA      | Union with actual groove data.                        |

### 4.30.2 KERNELV\_ERROR\_VALUE\_ARRAY

#### Description

Defines an array of size KERNELV\_ERROR\_VALUE\_COUNT for structures of type KERNELV\_KERNELV\_ERROR\_VALUE.

```
typedef KERNELV_ERROR_VALUE
KERNELV_ERROR_VALUE_ARRAY[KERNELV_ERROR_VALUE_COUNT];
```

A variable of this type can be transferred to the functions kernelv\_get\_error\_message\_values() to read error message values.

### 4.30.3 Enum E\_KERNELV\_ERR\_VAL\_TYPE

#### Description

Identifiers for the data type of the value in U\_KERNELV\_ERR\_VAL\_DATA.

| Symbol                   | Value | Meaning  |
|--------------------------|-------|--|
| ERR_VAL_TYPE_NONE        | -1    | Unknown data type.   |
| ERR_VAL_TYPE_BOOLEAN     | 0     | Data type is unsigned char 1(8 bits). Possible values 0 or.                            |
| ERR_VAL_TYPE_UNSN08      | 1     | Data type is unsigned char (8 bits).   |
| ERR_VAL_TYPE_SGN08       | 2     | Data type is signed char (8 bits).   |
| ERR_VAL_TYPE_UNSN16      | 3     | Data type is unsigned short int (16 bits).   |
| ERR_VAL_TYPE_SGN16       | 4     | Data type is signed short int (16 bits).   |
| ERR_VAL_TYPE_UNSN32      | 5     | Data type is unsigned long int (32 bits).  |
| ERR_VAL_TYPE_SGN32       | 6     | Data type is signed long int (32 bits).  |
| ERR_VAL_TYPE_UNSN64      | 7     | Data type is unsigned long long int (64 bits).   |
| ERR_VAL_TYPE_SGN64       | 8     | Data type is signed long long int (64 bits).   |
| ERR_VAL_TYPE_REAL64      | 9     | Data type is a 64-bit floating point number.   |
| ERR_VAL_TYPE_REAL32      | 10    | Data type is a 32-bit floating point number.   |
| ERR_VAL_TYPE_BOOLEAN     | 11    | Data type is a character   |
| ERR_VAL_TYPE_STRING      | 12    | Data type is a string of length KERNELV_ERR_MSG_STRING_LENGTH (without terminating 0). |
| ERR_VAL_TYPE_ADRESSE     | 13    | Data type is an address.   |
| ERR_VAL_TYPE_IGNORE      | 14    | Data type not assigned.  |
| ERR_VAL_TYPE_A3_REAL64   | 15    | Data type is an array with 3 64-bit floating point number.                             |
| ERR_VAL_TYPE_BITARRAY_32 | 16    | Data type is bit array with 32 bits.   |
| ERR_VAL_TYPE_BITARRAY_16 | 17    | Data type is bit array with 16 bits.   |

#### 4.30.4 Enum E\_KERNELV\_ERR\_VAL\_DIMENSION

##### Description

Dimension of the value in U\_KERNELV\_ERR\_VAL\_DATA.

| Symbol                        | Value | Meaning   |
|-------------------------------|-------|---|
| ERR_VAL_DIM_UNKNOWN           | -1    | Unknown dimension identifier.                                     |
| ERR_VAL_TYPE_IGNORE           | 0     | No dimension specified.   |
| ERR_VAL_DIM_POSITION          | 1     | Position in [ $10^{-4}$ mm or $^{\circ}$ ].                       |
| ERR_VAL_DIM_POSITION_HIG_RES  | 2     | Position in [ $10^{-7}$ mm or $^{\circ}$ ]                        |
| ERR_VAL_DIM_VELOCITY          | 3     | Velocity in $10^{-3}$ mm/s or $10^{-3}$ $^{\circ}$ /s.            |
| ERR_VAL_DIM_ACCELERATION      | 4     | Acceleration in mm/s <sup>2</sup> or $^{\circ}$ /s <sup>2</sup> . |
| ERR_VAL_DIM_JERK              | 5     | Jerk in mm/s <sup>3</sup> or $^{\circ}$ /s <sup>3</sup> .         |
| ERR_VAL_DIM_TIME              | 6     | Time in $\mu$ s.  |
| ERR_VAL_DIM_PERMILL           | 7     | Factorial specification in 1/1000 (per mill).                     |
| ERR_VAL_DIM_INKREMENTS        | 8     | Encoder increments.   |
| ERR_VAL_DIM_REV_FEED          | 9     | Revolution feed in $10^{-4}$ mm/rev.                              |
| ERR_VAL_DIM_CUTTING_SPEED     | 10    | Cutting speed $10^{-3}$ mm/s.                                     |
| ERR_VAL_DIM_PATH_RESOLUTION   | 11    | Path resolution in increments / $10^{-4}$ mm.                     |
| ERR_VAL_DIM_INCR_PER_REV      | 12    | increments/revolution   |
| ERR_VAL_DIM_BYTE              | 13    | Byte.   |
| ERR_VAL_DIM_PROPORTIONAL_GAIN | 14    | Proportional gain 0.01/s.   |
| ERR_VAL_DIM_FREQUENCY         | 15    | Frequency in Hz.  |
| ERR_VAL_DIM_LOAD              | 16    | Load in kg or kg*m <sup>2</sup> .                                 |

### 4.30.5 Enum E\_KERNELV\_ERR\_VAL\_MEANING

**Description**

Meaning of the value in U\_KERNELV\_ERR\_VAL\_DATA.

| Symbol                       | Value | Meaning                        |
|------------------------------|-------|--------------------------------|
| ERR_VAL_MEAN_UNKNOWN         | -1    | Unknown meaning.               |
| ERR_VAL_MEAN_LIMIT           | 0     | Limit value                    |
| ERR_VAL_MEAN_ACT_VAL         | 1     | Current value                  |
| ERR_VAL_MEAN_ERR_VAL         | 2     | Error value                    |
| ERR_VAL_MEAN_EXPECT_VAL      | 3     | Expected value                 |
| ERR_VAL_MEAN_CORR_VAL        | 4     | Corrected value                |
| ERR_VAL_MEAN_LOG_AXIS_NR     | 5     | logical axis number            |
| ERR_VAL_MEAN_DRIVE_TYPE      | 6     | Drive type                     |
| ERR_VAL_MEAN_LOG_BED_ELEM_NR | 7     | logical control element number |
| ERR_VAL_MEAN_STATE           | 8     | State                          |
| ERR_VAL_MEAN_TRANSITION      | 9     | Transition                     |
| ERR_VAL_MEAN_SENDER          | 10    | Sender                         |
| ERR_VAL_MEAN_CLASS           | 11    | Class                          |
| ERR_VAL_MEAN_INSTANCE        | 12    | Instance                       |
| ERR_VAL_MEAN_IDENT_NR        | 13    | Identification number          |
| ERR_VAL_MEAN_STATUS          | 14    | Status                         |
| ERR_VAL_MEAN_RING_NR         | 15    | Ring number                    |
| ERR_VAL_MEAN_SATZ_NR         | 16    | Block number                   |
| ERR_VAL_MEAN_MIN_LIMIT       | 17    | Lower limit value              |
| ERR_VAL_MEAN_MAX_LIMIT       | 18    | Upper limit value              |
| ERR_VAL_MEAN_START_VAL       | 19    | Initial value                  |
| ERR_VAL_MEAN_TARGET_VAL      | 20    | Final value                    |
| ERR_VAL_MEAN_FILENAME        | 21    | File name                      |
| ERR_VAL_MEAN_LINE            | 22    | Line (text) in a file          |
| ERR_VAL_MEAN_LINE_NUMBER     | 23    | Line number in a file          |
| ERR_VAL_MEAN_COLUMN_NUMBER   | 24    | Column number in a file        |
| ERR_VAL_MEAN_ARGUMENT        | 25    | Argument                       |
| ERR_VAL_MEAN_PARAMETER       | 26    | Parameter                      |
| ERR_VAL_MEAN_AXIS            | 27    | Axis (string)                  |
| ERR_VAL_MEAN_COMPENSATION    | 28    | Compensation index             |
| ERR_VAL_MEAN_IDENTIFIER      | 29    | Identifier                     |
| ERR_VAL_MEAN_CHAIN           | 30    | Chain                          |

## 4.31 Enum KERNELV\_AXIS\_OFFSET\_TYPES

### Description

Assign different offset types to the index of the offset vector of the function `kernelv_ch_axis_get_offsets()`.

| Symbol                              | Value | Meaning   |
|-------------------------------------|-------|---|
| KERNELV_AXIS_OFFSET_UNKNOWN         | -1    | Unknown offset type.                                |
| KERNELV_AXIS_OFFSET_ZERO            | 0     | Offset caused by zero offset (G54 ... G59).         |
| KERNELV_AXIS_OFFSET_ADDED_ZERO      | 1     | Offset caused by reference point offset (G92).      |
| KERNELV_AXIS_OFFSET_PRESET          | 2     | Offset caused by position preset (# PSET).          |
| KERNELV_AXIS_OFFSET_CLAMP           | 3     | Offset caused by clamp position offset data.        |
| KERNELV_AXIS_OFFSET_TOOL            | 4     | Offset caused by tool data.                         |
| KERNELV_AXIS_OFFSET_MEASURE         | 5     | Offset caused by G101 (Include measurement offset). |
| KERNELV_AXIS_OFFSET_MANUAL_OP       | 6     | Offset caused by manual mode.                       |
| KERNELV_AXIS_OFFSET_TRACK_CS_OFFSET | 7     | Offset caused by #CS TRACK'.                        |
| KERNELV_AXIS_OFFSET_MAX             | 8     | Data type is signed long long int (64 bits).        |

\*This offset is not available in all versions.



## 4.32 External measuring hardware

### 4.32.1 Struct KERNELV\_EXT\_LATCH\_COMMAND\_DATA

**Description**

The structure contains information about commands output by the CNC to the external measuring hardware.

**Memory orientation**

The individual structure elements are packed in the memory.

| Element    | Type                           | Meaning  |
|------------|--------------------------------|--|
| order_type | E_KERNELV_EXT_LAT<br>CH_ORDER  | Type of command to the external latch hardware.  |
| input      | unsigned long int              | Number of the measurement input of the external measuring hardware used for the measurement. |
| edge       | E_KERNELV_MEAS_A<br>CTIVE_EDGE | Edge of the probe signal used for the measurement.   |

### 4.32.2 Enum E\_KERNELV\_EXT\_LATCH\_ORDER

**Description**

Type of the measuring command to be executed.

| Symbol                  | Value | Meaning                    |
|-------------------------|-------|----------------------------|
| E_KERNELV_NO_ORDER      | 0     | No command active          |
| E_KERNELV_ENABLE_PROBE  | 1     | Enable measuring hardware  |
| E_KERNELV_DISABLE_PROBE | 2     | Disable measuring hardware |

### 4.32.3 E\_KERNELV\_MEAS\_ACTIVE\_EDGE

**Description**

Edge of the probe signal to be evaluated.

| Symbol                                | Value | Meaning                   |
|---------------------------------------|-------|---------------------------|
| E_KERNELV_MEAS_SIGNAL_<br>LOW_ACTIVE  | 1     | Measure on negative edge. |
| E_KERNELV_MEAS_SIGNAL_<br>HIGH_ACTIVE | 2     | Measure on positive edge. |

## 4.33 Production time calculation

### Description

The structure contains the filenames of the programs that should be started in the individual channels in production time mode.

KERNELV\_PT\_FILE\_NAME is of type CHAR[KERNELV\_PT\_PRG\_NAME\_LEN]

### Memory orientation

The individual structure elements are packed in the memory.

| Element | Type  | Meaning  |
|---------|---|--|
| file    | KERNELV_PT_FILE_NAME<br>[KERNELV_PT_MAX_CHAN<br>] | The filenames of the programs that should be started in the individual channels in production time mode. |

## 5 kernelv API constants

All listed constants are defined in the file `kernelv.h`.

### 5.1 KERNELV\_VAR\_STRING\_LEN

#### Description

Maximum number of characters in a string of the union `U_KERNELV_VAR_VALUE`. The total length of the string (including terminating zeroes) is `KERNELV_VAR_STRING_LEN + 1` byte.

#### Value

127

### 5.2 KERNELV\_FILE\_NAME\_LENGTH

#### Description

Maximum number of the characters in a string of the structure `KERNELV_NC_LINE_DATA`. The total length of the string (including terminating zeroes) is `KERNELV_FILE_NAME_LENGTH + 1` byte.

#### Value

83

### 5.3 KERNELV\_VAR\_NAME\_LENGTH

#### Description

Maximum permissible length of the variable name to read and write variables with the `kernelv_ch_get_variable_value()/kernelv_ch_get_variable_value()` functions.

#### Value

255

### 5.4 KERNELV\_OPTION\_LICENSE\_CHECK\_VERBOSE

#### Description

Bit mask to enable additional outputs during the license check. Must be set before calling `kernelv_startup()`.

#### Value

1 (0x1)

### 5.5 CONTOUR\_MAX\_DATA\_V0

#### Description

Array size of the union element `visu_data_v0[]` of the union `CONTOUR_VISU_DATA`.

**Value**

15

## 5.6 CONTOUR\_MAX\_DATA\_V1

**Description**

Array size of the union element visu\_data\_v1[] of the union CONTOUR\_VISU\_DATA.

**Value**

10

## 5.7 CONTOUR\_MAX\_DATA\_V2

**Description**

Array size of the union element visu\_data\_v2[] of the union CONTOUR\_VISU\_DATA.

**Value**

5

## 5.8 CONTOUR\_MAX\_DATA\_V3

**Description**

Array size of the union element visu\_data\_v3[] of the union CONTOUR\_VISU\_DATA.

**Value**

10

## 5.9 CONTOUR\_MAX\_DATA\_V4

**Description**

Array size of the union element visu\_data\_v4[] of the union CONTOUR\_VISU\_DATA.

**Value**

7

## 5.10 CONTOUR\_MAX\_DATA\_V5

**Description**

Array size of the union element visu\_data\_v5[] of the union CONTOUR\_VISU\_DATA.

**Value**

7

## 5.11 CONTOUR\_MAX\_DATA\_V6

### Description

Array size of the union element visu\_data\_v6[] of the union CONTOUR\_VISU\_DATA.

### Value

6

## 5.12 CONTOUR\_MAX\_DATA\_V7

### Description

Array size of the union element visu\_data\_v7[] of the union CONTOUR\_VISU\_DATA.

### Value

5

## 5.13 CONTOUR\_MAX\_DATA\_V8

### Description

Array size of the union element visu\_data\_v8[] of the union CONTOUR\_VISU\_DATA.

### Value

3

## 5.14 CONTOUR\_MAX\_DATA\_V9

### Description

Array size of the union element visu\_data\_v9[] of the union CONTOUR\_VISU\_DATA.

### Value

5

## 5.15 CONTOUR\_MAX\_DATA\_V10

### Description

Array size of the union element visu\_data\_v10[] of the union CONTOUR\_VISU\_DATA.

### Value

4

## 5.16 CONTOUR\_MAX\_DATA\_V11

### Description

Array size of the union element visu\_data\_v8[] of the union CONTOUR\_VISU\_DATA.

### Value

3

## 5.17 CONTOUR\_MAX\_M\_H\_DATA

### Description

Array size of the element m\_h\_data[] of the structure CONTOUR\_TECHNO\_DATA.

### Value

20

## 5.18 CONTOUR\_MAX\_SPDL\_DATA

### Description

Array size of the element s\_data[] of the structure CONTOUR\_TECHNO\_DATA.

### Value

6

## 5.19 CONTOUR\_AXIS\_PER\_CHANNEL

### Description

Array size of the element ax\_data[] or ax\_data\_v1[] of the structures CONTOUR\_VISU\_DATA\_V0 ... CONTOUR\_VISU\_DATA\_V8.

### Value

32

## 5.20 KERNELV\_ERROR\_VALUE\_COUNT

### Description

Array size of the array ERROR\_VALUE\_ARRAY, number of structures of the KERNELV\_ERROR\_VALUE type returned by the function kernelv\_get\_error\_values().

### Value

5

## 5.21 KERNELV\_ERR\_MSG\_STRING\_LENGTH

### Description

Length of a string in the union U\_KERNELV\_ERR\_VAL\_DATA without terminating zero.

### Value

23

## 5.22 KERNELV\_CHANNEL\_TECHNO\_DATA\_COUNT

### Description

Number of elements in a KERNELV\_CHANNEL\_TECHNO\_DATA\_ARRAY or KERNELV\_CHANNEL\_TECHNO\_DATA\_ARRAY2 array.

Maximum number of elements which return the functions kernelv\_ch\_get\_(new\_)techno\_data() or kernelv\_ch\_get\_(new\_)techno\_data2().

### Value

30

## 5.23 KERNELV\_AXIS\_TECHNO\_DATA\_COUNT

### Description

Number of elements in a KERNELV\_AXIS\_TECHNO\_DATA\_ARRAY or KERNELV\_AXIS\_TECHNO\_DATA\_ARRAY2 array.

Maximum number of elements which return the functions kernelv\_ax\_get\_(new\_)techno\_data() or kernelv\_ax\_get\_(new\_)techno\_data2().

### Value

30

## 5.24 KERNELV\_ERROR\_VALUE\_COUNT

### Description

Number of elements in a KERNELV\_ERROR\_VALUE\_ARRAY array.

Maximum number of elements which return the function kernelv\_get\_error\_message\_values()

### Value

5

## 5.25 KERNELV\_INSTANCE\_PREFIX\_MAX\_LEN

### Description

Maximum length of the string which may be transferred as prefix to call the `kernelv_startup_instance()` function.

### Value

60



## 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](http://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](mailto: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](mailto: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](mailto:info@beckhoff.com)  
web: [www.beckhoff.com](http://www.beckhoff.com)

# Index



More Information:

[www.beckhoff.com/TF5270/71](http://www.beckhoff.com/TF5270/71)

Beckhoff Automation GmbH & Co. KG  
Hülshorstweg 20  
33415 Verl  
Germany  
Phone: +49 5246 9630  
[info@beckhoff.com](mailto:info@beckhoff.com)  
[www.beckhoff.com](http://www.beckhoff.com)

