zub
machine control AG

ZB MOC

LIBRARY

Manual

Last Update:
December 2002

Imprint

Address zub machine control AG · Buzibachstrasse 23
CH-6023 Rothenburg

Telefon +41 41 54150-40
Telefax +41 41 54150-49

http://www.zub.ch
http://www.aposs.ch
info@zub.ch

Table of Contents

ZB MOC 1

Imprint 2

Table of Contents 3

Introduction 4

What Services are provided ? 4

How can the Library be used ? 4

Function Modules 5

Open Interface 5

Close Interface 6

Configuration of the Control Unit 6

Global and Axis Parameter Handling of the Control Unit 7

APOSS Program Handling 8

APOSS Configuration Handling 9

Exchanging Data with APOSS Programs 10

Reading of a User Array 10

Writing of an User Array 10

Reading of a User Variable 10

Transmitting of a User Variable 10

Data Types and Structures 11

Standardized Data Types 11

Overview of Structures 11

Overview of Error Codes 18

Alphabetical Function List and Specification 21

AutoClear up to WriteUserVar 21

Appendix 118

Index 118

Introduction

The ZBMOC Library offers a collection of simple to use functions for ...

- Configuration
- Control
- Supervising

of zub control units in production environment.
Every of the required functions can be accessed by your Windows application program.

What Services are provided ?

The ZBMOC Library offers a consistent application programming interface (API) for different control units of zub machine control AG. The programming interface is independent of the hardware interfaces and bus systems used by each application and control unit itself.

Slightly differences in control and functional range caused by different hardware and software versions of control units are compensated by the ZBMOC Library.

It is possible to address a control unit offering a new interface or bus systems without important modifications of existing application programs.

The ZBMOC Library is an highly efficient tool, which allows the application programmer to focus on the application programming itself instead of working through all details of bus systems, error handling and so on.

How can the Library be used ?

The ZBMOC Library is a Dynamic Link Library (DLL) available for 16bit and 32bit Windows (WIN32). Both versions offer full driver support of all supported interfaces.

It depends on the development environment how the DLLs can be used by your application program. If Microsoft Visual C++ is used for example, it is sufficient to include the header zbMoc.h and link the base library (zbMoc.lib or zbMoc32.lib) to use all functions of the library. The corresponding zbMoc.dll has to be placed somewhere in the search path which is used by the program.

There are some short examples available showing the zbMoc Library integration for the development environemts mentioned below. Please visit our websites www.zub.ch and www.aposs.ch.

Visual C++
Borland C++
Borland C++ Builder
Delphi

Function Modules

Open Interface

The interface has to be opened by calling one of the 'ZbMocOpen...' functions before any communication with the the control unit is possible.

The 'ZbMocOpen...' functions are responsible for the following internal steps:

1. Availability, compatibility and fitness for use check of the hardware, as well as the required drivers for the selected interface.
2. Setting of all required parameters necessary to open the interface.
3. Scan of all control units, which are connected and have a bus ID within the defined range.
4. Check of the communication to all control units and evaluation of the control unit type, as well as the particular hardware and software version.
5. Internal storage of all collected information about the scanned control units. This information can be queried by the application program later.
6. Returning an interface handle (0...n) or a negative error code..

The following functions are available for opening the interface:

CAN BUS Centronics Adapter

```
ZbMocOpenCanLpt (
                UNSIGNED16  lptnr,
                UNSIGNED16  irq,
                UNSIGNED16  baud,
                UNSIGNED16  mode,
                UNSIGNED16  scanfrom,
                UNSIGNED16  scanto
                BOOLEAN     breakall
                )
```

CAN BUS GIN Adapter

```
ZbMocOpenCanGin (
                UNSIGNED16  ioadr,
                UNSIGNED16  irq,
                UNSIGNED16  baud,
                UNSIGNED16  mode,
                UNSIGNED16  scanfrom,
                UNSIGNED16  scanto
                BOOLEAN     breakall
                )
```

V24 Serial Port

```
ZbMocOpenV24    (
                UNSIGNED16  comnr,
                char*       init,
                UNSIGNED16  mode
                BOOLEAN     breakall
                )
```

VLT Serial Port

```
ZbMocOpenVLT (
UNSIGNED16 comnr,
UNSIGNED16 baud,
char* drivername
UNSIGNED16 mode,
UNSIGNED16 scanfrom,
UNSIGNED16 scanto
BOOLEAN breakall
)
```

Close Interface

f the application program has to be closed or if the interface has to be opened up with new parameters, the interface has to be closed by using one of the functions 'ZbMocClose' or 'ZbMocCloseAll' in advance.

By closing the interface all system ressources are freed again, which were allocated during opening up.

Configuration of the Control Unit

Some important functions relevant for configuration of the control unit

Reset a control unit to the factory settings.

```
SIGNED16
ZbMocMoconClearFactory(UNSIGNED16 h, UNSIGNED16 id);
```

Query the configuration of a control unit identified by the control ID.

```
ZbMocMoconInfoS*
ZbMocMoconInfo(UNSIGNED16 h, UNSIGNED16 id);
```

.Setting of communication parameters for the CAN bus. This can be also done in a much more convenient way by using the V24 serial link of the control unit and the APOSS software.

```
SIGNED16
ZbMocMoconSetCanParameters(UNSIGNED16 h, UNSIGNED16 id,
UNSIGNED8 cannr, UNSIGNED8 baud);
```

Query the configuration of a control unit identified by the index (0..ZbMocGetNumber()-1).

```
ZbMocMoconInfoS*
ZbMocMoconInfoIdx(UNSIGNED16 h, UNSIGNED16 idx);
```

Persistent storage of programs and parameters in non-volatile memory. This is just necessary for control units without battery backup-ed RAM.

```
SIGNED16
ZbMocMoconSaveRam(UNSIGNED16 h, UNSIGNED16 id);
```

Global and Axis Parameter Handling of the Control Unit

Global and Axis Parameter Handling of the Control Unit

Some important functions relevant for parameter handling of the control unit:

Reset all parameters of a control unit to the factory settings.

```
SIGNED16
ZbMocMoconClearParameters(UNSIGNED16 hdl, UNSIGNED16 id);
```

Query the number of parameters of a control unit. This number depends on the hardware and software version of a control unit.

```
SIGNED16
ZbMocParamCount(UNSIGNED16 h, UNSIGNED16 id);
```

Reading out the global parameters of a control unit.

```
SIGNED16
ZbMocParamRead(UNSIGNED16 h, UNSIGNED16 id, ZbMocParamS*
para);
```

Setting the global parameters of a control unit.

```
SIGNED16
ZbMocParamWrite(UNSIGNED16 h, UNSIGNED16 id, ZbMocParamS*
para);
```

Query the number of axis parameters of a control unit. This number depends on the hardware and software version of a control unit.

```
SIGNED16
ZbMocAxisParamCount(UNSIGNED16 h, UNSIGNED16 id);
```

Reading out the axis parameters of a control unit.

```
SIGNED16
ZbMocAxisParamRead(UNSIGNED16 h, UNSIGNED16 id, UNSIGNED8
axnr, ZbMocAxisParamS* para);
```

Setting the axis parameters of a control unit.

```
SIGNED16
ZbMocAxisParamWrite(UNSIGNED16 h, UNSIGNED16 id, UNSIGNED8
axnr, ZbMocAxisParamS* para);
```

APOSS Program Handling

Some important functions relevant for program handling:

.Erasing all programs. The formerly used storage memory is freed.

```
SIGNED16
ZbMocMoconClearPrograms(UNSIGNED16 h, UNSIGNED16 id);
```

List all programs stored in the control unit. (The temporary stored program does not count.)

```
SIGNED16
ZbMocProgramList(UNSIGNED16 h, UNSIGNED16 id, ZbMocProgramS*
pl, UNSIGNED16 maxentries);
```

Loading a binary program into the control unit. The program is stored temporary at this stage. A binary program has to be generated out of the *.m source code by the APOSS compiler before. Easiest way to compile and download a program is to use the provided APOSS software.

```
SIGNED16
ZbMocProgramLoad(UNSIGNED16 h, UNSIGNED16 id, UNSIGNED16
prglen, UNSIGNED8 (*prg)(UNSIGNED32));
```

Storing the temporary downloaded program with a given name. A program number is assigned automatically to it.

```
SIGNED16
ZbMocProgramSave(UNSIGNED16 h, UNSIGNED16 id, const
UNSIGNED8* name);
```

Storing the temporary downloaded program with a given name and program number.

```
SIGNED16
ZbMocProgramSaveAs(UNSIGNED16 h, UNSIGNED16 id, const
UNSIGNED8* name, UNSIGNED16 prgnr);
```

Executing a stored program identified by the program number.

```
SIGNED16
ZbMocExec(UNSIGNED16 h, UNSIGNED16 id, UNSIGNED16 prgnr);
```

Executing the temporary program. It is possible to start the temporary program without storage directly after downloading.

```
SIGNED16
ZbMocExecTemp(UNSIGNED16 h, UNSIGNED16 id, BOOLEAN init);
```

.Query the execution state. Is the control unit still engaged in executing the program ?

```
SIGNED16
ZbMocMoconExecutionStatus(UNSIGNED16 h, UNSIGNED16 id,
BOOLEAN refresh_first, BOOLEAN* is_executing, SIGNED16*
errnum);
```

Interruption and stop of a running program as well as clearing a control unit error state. The control unit is set to a defined state.

```
SIGNED16
ZbMocBreak(UNSIGNED16 h, UNSIGNED16 id);
```

Identically to ZbMocBreak, but affects all control units.

```
SIGNED16
```

```
ZbMocBreakAll(UNSIGNED16 h);
```

Setting the 'Autostart' flag for a given stored program. The program with the 'Autostart' flag, starts up automatically at start-up of the control unit.

```
SIGNED16
ZbMocAutoSet(UNSIGNED16 h, UNSIGNED16 id, UNSIGNED8 prgnr);
```

Erasing the 'Autostart' flag. No program is executed automatically at start-up of the control unit.

```
SIGNED16
ZbMocAutoClear(UNSIGNED16 h, UNSIGNED16 id);
```

APOSS Configuration Handling

Reads out the parameter settings and arrays definitions of the given control unit and writes these configuration data into the configuration file (*.cnf) with the given name.

```
SIGNED16
ZbMocCNFSaveToFile(UNSIGNED16 h, UNSIGNED16 id, const char*
filename);
```

Downloads the data (parameters and content of arrays) of a configuration file (*.cnf) to the selected control unit.

```
SIGNED16
ZbMocCNFRestoreFromFile(UNSIGNED16 h, UNSIGNED16 id, const
char* filename);
```

What Services are provided ?

Exchanging Data with APOSS Programs

Some important functions relevant for data exchange of APOSS programs:

Reading of a User Array

The array might hold some input information used by the APOSS program, which is started afterwards.

```
SIGNED16
ZbMocReadUserArray(UNSIGNED16 h, UNSIGNED16 id, UNSIGNED16
arrnr, UNSIGNED16 mxlen, SIGNED32 *values, UNSIGNED16
*arrsize);
```

Writing of an User Array

The array might hold some data that was generated by the APOSS program running before.

```
SIGNED16
ZbMocWriteUserArray(UNSIGNED16 h, UNSIGNED16 id, UNSIGNED16
*arrnr, UNSIGNED16 *arrsize, UNSIGNED16 *rsize, SIGNED32
*werte);
```

Reading of a User Variable

Reading of a user defined variable of the APOSS program via CAN bus. The APOSS program transmits both numerical values via CAN bus during execution time. The PC can use the process data immediately for further calculation or process control.

```
SIGNED16
ZbMocReadUserVar(UNSIGNED16 h, UNSIGNED16 id, UNSIGNED16
*varnr, SIGNED32 *val, UNSIGNED32 timeout);
```

Transmitting of a User Variable

Transmission of a user defined variable to the APOSS program via CAN bus. The APOSS program reads both numerical values during execution time and can do any further processing.

```
SIGNED16
ZbMocWriteUserVar(UNSIGNED16 h, UNSIGNED16 id, UNSIGNED16
varnr, SIGNED32 val);
```

Data Types and Structures

Data Types and Structures

Specification of the data types, structures, error codes in use:

Standardized Data Types

There are some standardized data types defined in the ZbTypes.h header file to get a portability of the library as high as possible:

Signed Integer, 8, 16 and 32 bit long:

SIGNED8
SIGNED16
SIGNED32

Unsigned Integer, 8, 16 and 32 bit long:

UNSIGNED8
UNSIGNED16
UNSIGNED32

Boolean value, 0 or 1:

BOOLEAN

Overview of Structures

Structure holding information about connected control unit

ZbMocMoconInfoS:

The following structure contains information about one control unit. The 'ZbMocOpen...' functions retrieve this information automatically from all connected control units by calling 'ZbMocMoconInfoControlledRefresh' just after the connection has been established.

The ZbMoc library caches this information for each of the control units connected and returns a pointer to it to the application via the 'ZbMocMoconInfo' and 'ZbMocMoconInfoIdx' functions.

The application can force the ZbMoc library to requery the information from the control units by calling the function 'ZbMocMoconInfoControlledRefresh'.

```
#define MOC_FNAMELEN 10
typedef struct ZbMocMoconInfoS_
{

BOOLEAN    active;                  // unit accessible ok?
UNSIGNED16 interf;                  // interface type
UNSIGNED16 channelno;               // channel number
UNSIGNED16 full_version;            // full version number,
                                    // e.g. 4.50 = 450
UNSIGNED16 major_version;           // major version no, e.g. 4
UNSIGNED16 minor_version;           // minor version no, e.g. 50
UNSIGNED8  cpu_type;                // cpu type
UNSIGNED16 axis_controller_type;    // type of axis control:
                                    // 'L'=LM,'H'=HCTL,'S'=Software
UNSIGNED8  number_axis;             // number of axes
UNSIGNED8  option_code;             // option code
UNSIGNED8  board_revision;          // pcboard version
UNSIGNED16 bus_id;                  // ID on can bus
UNSIGNED8  unit_name[MOC_FNAMLEN+1]; // alphanumeric name of this unit
UNSIGNED32 regspeed;                // regulator speed, 8MHz=800
BOOLEAN    sdo_compatible;          // sdo compatible firmware

} ZbMocMoconInfoS;
```

Structure holding information about saved program

ZbMocProgramS:

The following structure contains the information about one program, as it has been permanently saved into control units memory. Use the 'ZbMocProgramList' function to query program information from a connected control unit.

```
typedef struct ZbMocProgramS_
{

UNSIGNED16 index;                // mocon program number
UNSIGNED8  name[25];             // program name
BOOLEAN    autostart;            // autostart YES/NO

} ZbMocProgramS;
```

Structure holding information about restored source code

ZbMocSourceInfoS:

The following structure contains the information about program source code that was read out of the control unit.

```
#define MOC_SOURCENAMELEN 80
typedef struct ZbMocSoureInfoS_
{

UNSIGNED8  day;    // Day of saving source into control unit
UNSIGNED8  month;  // Month of saving source into control unit
UNSIGNED16 year;   // Year of saving source into control unit
UNSIGNED8  hour;   // Hour of saving source into control unit
UNSIGNED8  minute; // Minute of saving source into control unit
char       filename[MOC_SOURCENAMELEN];  // Name of source file

} ZbMocSourceInfoS;
```

Structure holding status of control unit

ZbMocStatusS:

Use the function 'ZbMocMoconStatusGet' to retrieve the following status record from a connected control unit.

```
typedef struct ZbMocStatusS_
{

UNSIGNED8  userstatus;    // see controller doc
UNSIGNED8  systemstatus;
UNSIGNED8  flags;
UNSIGNED8  numberaxes;    // number of axis installed
UNSIGNED32 ax1pos;        // current position of axis #1

} ZbMocStatusS;
```

Structure holding global parameters of control unit

ZbMocParamS:

These are the control unit's parameters.

Read them with ZbMocParamRead and write them with the function 'ZbMocParamWrite'.

The number of parameters implemented in a specific control unit depends on the software version of the unit. See the contoller documentation. Also, the function 'ZbMocParamCount' returns the number of parameters supported by a specific control unit.

```
typedef struct ZbMocParamS_
{

UNSIGNED32 can_nr;
UNSIGNED32 can_baud;
UNSIGNED32 prgpar;
UNSIGNED32 i_prgstart;
UNSIGNED32 i_prgchoice;
UNSIGNED32 i_break;
UNSIGNED32 i_continue;
UNSIGNED32 i_errclr;
UNSIGNED32 o_error;

} ZbMocParamS;
```

Structure holding axis parameters of control unit

ZbMocAxisParamS:

These are the control unit's AXIS parameters.

Read them with the function 'ZbMocAxisParamRead' and write them with the function 'ZbMocAxisParamWrite'.

The number of axis parameters implemented in a specific control unit depends on the software version of the unit. See the control unit documentation. Also, function 'ZbMocAxisParamCount' returns the number of parameters supported by a specific control unit.

Please note that some of the parameters have different meanings, depending on control unit chip and software version.

It is recommended, to use the APOSS software to set an axis parameter, and permanently save them to control unit's memory.

APOSS automatically handles all hardware and software control unit versions correctly.

For the meaning of individual parameters, please see the control unit's documentation.

```
typedef struct ZbMocAxisParamS_
{

UNSIGNED32 function_inputs;     // #0
UNSIGNED32 function_outputs;    // #18
  SIGNED32 sensor_counts;       // #2
UNSIGNED32 max_velocity;        // #9
UNSIGNED32 max_acceleration;    // #10
UNSIGNED32 timer;               // #14
UNSIGNED32 gain;                // #11
UNSIGNED32 zero;                // #12
UNSIGNED32 pole;                // #13
UNSIGNED32 pullgap;             // #15
UNSIGNED32 test_window;         // #8
UNSIGNED32 org_velocity;        // #16
UNSIGNED32 org_acceleration;    // #17
UNSIGNED32 home_velocity;       // #7
UNSIGNED32 home_enforce;        // #3
UNSIGNED32 negative_limit;      // #4
UNSIGNED32 positive_limit;      // #5
UNSIGNED32 integration_limit;   // #21
UNSIGNED32 use_neg_limit;       // #19
UNSIGNED32 use_pos_limit;       // #20
UNSIGNED32 velocity_divisor;    // #22
UNSIGNED32 user_factor;         // #23
UNSIGNED32 rpm;                 // #1
UNSIGNED32 mnu_point;           // #6
UNSIGNED32 test_duration;       // #24
UNSIGNED32 test_limit;          // #25
UNSIGNED32 usrteil;             // ???
UNSIGNED32 encodertyp;          // #27
UNSIGNED32 posdrct;             // #28
UNSIGNED32 posunits;            // #29
UNSIGNED32 mencoder;            // #30
UNSIGNED32 mencodertype;        // #67
UNSIGNED32 ramp_min;            // #31
UNSIGNED32 ramptype;            // #32
UNSIGNED32 dfltvel;             // #33
```

```
UNSIGNED32 dfltacc;             // #34
UNSIGNED32 bandwidth;           // #35
UNSIGNED32 ffvel;               // #33
UNSIGNED32 ffacc;               // #37
UNSIGNED32 regwmax;             // #38
UNSIGNED32 regwmin;             // #39
UNSIGNED32 hometype;            // #40
UNSIGNED32 home_ramp;           // #41
UNSIGNED32 originoffs;          // #42
UNSIGNED32 errcond;             // #43
UNSIGNED32 endswmod;            // #44
UNSIGNED32 i_refswitch;         // #45
UNSIGNED32 i_poslimitsw;        // #46
UNSIGNED32 i_neglimitsw;        // #47
UNSIGNED32 o_brake;             // #48
UNSIGNED32 syncfactm;           // #49
UNSIGNED32 syncfacts;           // #50
UNSIGNED32 synctype;            // #51
UNSIGNED32 syncmarkm;           // #52
UNSIGNED32 syncmarks;           // #53
UNSIGNED32 syncposoffs;         // #54
UNSIGNED32 syncaccuracy;        // #55
UNSIGNED32 syncready;           // #56
UNSIGNED32 syncfault;           // #57
UNSIGNED32 syncmpulsm;          // #58
UNSIGNED32 syncmpulss;          // #59
UNSIGNED32 syncmtypm;           // #60
UNSIGNED32 syncmtyps;           // #61
UNSIGNED32 syncmstart;          // #62
UNSIGNED32 syncvftime;          // #65
UNSIGNED32 syncvelrel;          // #66
UNSIGNED32 revers;              // #63
UNSIGNED32 o_axmove;            // #64
UNSIGNED32 posfact_n;           // #26
UNSIGNED32 syncmwinm;           // #68
UNSIGNED32 syncmwins;           // #69
UNSIGNED32 esccond;             // #70

} ZbMocAxisParamS;
```

Structure holding data of a test run

ZbMocTestrunParamS:

The following structure contains all parameters needed to set up a testrun.

Testrun is used by the APOSS testrun function, which allows to check and tune axis parameters. It is not intended for use by application programs.

```
typedef struct ZbMocTestrunParamS_
{

  SIGNED32 distance;
UNSIGNED32 velocity;
UNSIGNED32 acceleration;
UNSIGNED32 deceleration;
UNSIGNED32 datapoints;
UNSIGNED32 parameters;
UNSIGNED32 sample_timer;

UNSIGNED32 proportional_factor;
UNSIGNED32 differential_factor;
UNSIGNED32 integral_factor;
UNSIGNED32 integration_limit;
UNSIGNED32 pid_timer;
UNSIGNED32 bandwidth;
UNSIGNED32 ff_acceleration;
UNSIGNED32 ff_velocity;

UNSIGNED32 velmax;
UNSIGNED32 accmax;
UNSIGNED32 encoder_resolution;
UNSIGNED32 posfact_z;
UNSIGNED32 posfact_n;
  SIGNED32 posdrct;

} ZbMocTestrunParamS;
```

Source code Information

ZbMocBlobInfo:

Program source code can be stored in the control unit in so-called Blobs (= Binary large objects). Each Blob is identified by a number, which is unique in the list of all Blobs. This number is used to get access to the Blob, e.g. program source code, by using the function 'ZbMocBlobRead'.

```
#define BLOB_INFO_LEN 19
typedef struct ZbMocBlobInfoS_
{

UNSIGNED16 num;
 UNSIGNED8 info[BLOB_INFO_LEN];

} ZbMocBlobInfoS;
```

Overview of Error Codes

The following error codes are defined in the 'zbmocerr.h' Header:

```
// Error codes used in the ZBMoc Library
// All error states are negative.
// So it is easy to detect, if a function succeeded or not.

// If no error is present or a function succeeded
// ZBMOC_OK (= No error) is returned
#define ZBMOC_OK                            0

// General error messages,
// e.g. concerning general function,
//      control unit addressing or parameter problems
// Operation not available
#define ZBMOC_E_UNAVAILABLE              -100
// No new handles are available
#define ZBMOC_E_ALLBUSY                  -101
// Given handle is not valid
#define ZBMOC_E_BADHANDLE                -102
// Given index is not valid
#define ZBMOC_E_BADINDEX                 -103
// Function is for CAN connections only
#define ZBMOC_E_CANONLY                  -104
// Illegal parameters
#define ZBMOC_E_PARAMETERS               -105
// Error in reading parameters
#define ZBMOC_E_PARAMETER_READOUT        -106
// Controller is busy...
#define ZBMOC_E_IS_BUSY                  -107
// Function not available
#define ZBMOC_E_FUNCTION_UNAVAILABLE     –108
// ID does not exist
#define ZBMOC_E_BADID                    -109
// No controller found
#define ZBMOC_E_NOCONTROLLER             -110
// Compression error
#define ZBMOC_E_COMPRESSION_ERROR        -111
// Problem with Blob, e.g. Blob number is not valid
#define ZBMOC_E_BADBLOB                  -112
// CAN Open internal error
#define ZBMOC_E_CANOPEN_INTERNAL         -113
// Array is not existing
#define ZBMOC_E_BADARRAYNO               -114
// Array size mismatch
#define ZBMOC_E_BADARRAYSIZE             -115
// 'Break' is required for execution of the requested function
#define ZBMOC_E_BREAK_REQUIRED           -116
// Not all control units respond to a request
#define ZBMOC_E_CONTROLLER_MISSING       -117
// Size of provided buffer is too small
#define ZBMOC_E_RXBUFFER_TOO_SMALL       -118
// File access error, e.g. file not found
#define ZBMOC_E_FILEACCESS               -119
// Content of file not valid
#define ZBMOC_E_FILECONTENT              -120
// Running out of memory
#define ZBMOC_E_LOWMEMORY                -121
// Array size is too big
#define ZBMOC_E_ARRAYTOOBIG              -122
```

```
// CAN specific error messages
// concerning CAN bus problems
#define ZBMOC_E_CAN_BUSY                  -201
#define ZBMOC_E_CAN_TIMEOUT               -202
#define ZBMOC_E_CAN_NO_RESET              -203
#define ZBMOC_E_CAN_NO_HALT               -204
#define ZBMOC_E_CAN_NO_START              -205
#define ZBMOC_E_CAN_NO_HARDWARE           -206
#define ZBMOC_E_CAN_NO_MEMORY             -207
#define ZBMOC_E_CAN_NO_USER               -208
#define ZBMOC_E_CAN_NO_CPU_ACCESS         -209
#define ZBMOC_E_CAN_NO_POLLSLAVES         -210
#define ZBMOC_E_CAN_TOO_MANY_SLAVES       -211
#define ZBMOC_E_CAN_NO_SLAVE              -212
#define ZBMOC_E_CAN_NO_INIT               -213
#define ZBMOC_E_CAN_SLAVES_EXIST          -214
#define ZBMOC_E_CAN_NO_POLLING            -215
#define ZBMOC_E_CAN_POLLING_IS_RUN        -216

// VLT specific error messages
#define ZBMOC_E_VLT_SCAN_IN_PROGRESS      -217
#define ZBMOC_E_VLT_SCAN_ERROR            -218
#define ZBMOC_E_VLT_SCAN_NO_SLAVES        -219
#define ZBMOC_E_VLT_UNIT_NOT_AVAILABLE    -220
#define ZBMOC_E_VLT_ILLEGAL_ID            -221
#define ZBMOC_E_VLT_OPEN                  -222
#define ZBMOC_E_VLT_TRANSMIT              -223
#define ZBMOC_E_VLT_FRAMETYPE             -224
#define ZBMOC_E_VLT_MESSAGETYPE           -225
#define ZBMOC_E_VLT_STATUS_READ_ERROR     -226
#define ZBMOC_E_VLT_RXBCC                 -227
#define ZBMOC_E_VLT_NORXBUFFER            -228
#define ZBMOC_E_VLT_INTERNAL              -229
#define ZBMOC_E_VLT_RXBADLENGTH           -230
#define ZBMOC_E_VLT_UNKNOWN               -231
#define ZBMOC_E_VLT_TIMEOUT               -232
#define ZBMOC_E_VLT_LOSTCONNECTION        -233

// COM port specific error messages
#define ZBMOC_E_COM_ILLEGALPORT           -234
#define ZBMOC_E_COM_OPENERROR             -235
#define ZBMOC_E_COM_TXERROR               -236
#define ZBMOC_E_COM_TIMEOUT               -237
#define ZBMOC_E_COM_ERROR                 -238

// Some more error messages
#define ZBMOC_E_CAN_COMERROR              -239

#define ZBMOC_E_CAN_TIMER_UNAVAILABLE     -240
#define ZBMOC_E_CAN_STATUS_TIMEOUT        -241

#define ZBMOC_E_COM_XON_ERROR             -242
#define ZBMOC_E_COM_CLOSED                -243
#define ZBMOC_E_COM_COMMAND               -244
#define ZBMOC_E_COM_PROTOCOL_LEN          -245
#define ZBMOC_E_COM_GLOBAL_TIMEOUT        -246
#define ZBMOC_E_COM_COMMAND_TIMEOUT       -247
```

```
#define ZBMOC_E_COM_PORT                -248

#define ZBMOC_E_NULL_POINTER            -249
#define ZBMOC_E_SLAVE_NOT_FOUND         -250
#define ZBMOC_E_NO_PARAMETERS           -251

#define ZBMOC_MESSAGE_READ              -252

#define ZBMOC_E_COM_FRAME_FORMAT_ERROR  -253
#define ZBMOC_E_COM_FRAME_TEMP_OVERFLOW -254
```

Alphabetical Function List and Specification

AutoClear up to WriteUserVar

ZbMoc AutoClear
(
UNSIGNED16 h,
UNSIGNED16 id
)

Summary Clears the 'Autostart' flag of the control unit with the given ID. No program is started anymore after switch-on.

Parameter

h	Handle of the interface
id	Identification number (ID) of the control unit

Return value SIGNED16 Error code

Remarks Even if no 'Autostart' flag was set before there is no error reported.

It is possible to call up this function preventive to make sure that no 'Autostart' flag is set anymore, nevertheless if an 'Autostart' flag was set before or not.

Portability

zbmoc.dll	Version 6.0.0 and higher
Control unit	Version 2.7 and higher

ZbMoc AutoSet

```
(
UNSIGNED16 h,
UNSIGNED16 id,
UNSIGNED8 prgnr
)
```

Summary

Sets the 'Autostart' flag of the given control unit for the program defined by the given program number.

Parameter

h	Handle of the interface
id	Identification number (ID) of the control unit
prgnr	Program number

Return value

SIGNED16	Error code

Remarks

If the 'Autostart' flag was given to another program before, this flag is removed and given to the program number defined now. An existing 'Autostart' flag is removed even, if the given program number is not in use.

Just one program per control unit can be tagged with an 'Autostart' flag.

The program owning the 'Autostart' flag starts automatically after switch-on or reset of the control unit.

Portability

zbmoc.dll	Version 6.0.0 and higher
Control unit	Version 2.7 and higher

ZbMoc AxisParamCount

```
(
UNSIGNED16 h,
UNSIGNED16 id
)
```

Summary Returns the number of axis parameters of the selected control unit.

Parameter

h	Handle of the interface
id	Identification number (ID) of the control unit

Return value

SIGNED16	>= 0:	Number of axis parameters
	<0:	Error code

Remarks The number of axis parameters depends on the type and version of the connected control unit.

Portability

zbmoc.dll	Version 6.0.0 and higher
Control unit	Version 2.7 and higher

ZbMoc AxisParamRead

```
(
UNSIGNED16 h,
UNSIGNED16 id,
UNSIGNED8 axnr,
ZbMocAxisParamS * para
)
```

Summary Reads out the axis parameters of the selected control unit and axis number. The parameters are decoded and copied into the 'ZbMocAxisParamS' structure.

Parameter

h	Handle of the interface
id	Identification number (ID) of the control unit
axnr	Axis number (1...n) of the control unit
para	Pointer to a structure taking the read out axis parameters

Return value

SIGNED16	Error code

Remarks All axis parameters exist on its own for every axis of the control unit.

This function reads out all axis parameters which are supported by the connected control unit and version. All other parameters are set to zero.

If there is any conversion necessary to get compatible results corresponding to the version of the control unit, this is done internally.

The structure 'ZbMocAxisParamS' is specified in section “Data Types and Structures”.

Portability

zbmoc.dll	Version 6.0.0 and higher Version 6.4.3 and higher: Usage of SDO-based communication
Control unit	Version 2.7 and higher Version 6.01 and higher: Usage of SDO-based communication => Function can be processed independent of control unit's execution state, e.g. while APOSS program executes

ZbMoc AxisParam ReadRaw

```
(
UNSIGNED16 h,
UNSIGNED16 id,
UNSIGNED8 axnr,
UNSIGNED32* param,
UNSIGNED16 first,
UNSIGNED16 last
)
```

Summary

Reads out the defined number of parameters of the selected control unit and axis number. The parameters are copied into the given array without any modifications, i.e. no conversion or decoding takes place.

Parameter

h	Handle of the interface
id	Identification number (ID) of the control unit.
axnr	Axis number (1...n) of the control unit
param	Pointer to an array taking the read out parameters
first	Number of the first parameter to read out: 0..n-1
last	Number of the last parameter to read out: 0..n-1

Return value

SIGNED16	Error code

Remarks

It has to be made sure, that the requested axis parameters exist for the connected control unit and version !

The array size has to be big enough to take all read out parameters. The array size is not checked by this function !

This function is intended for internal use only !

Portability

zbmoc.dll	Version 6.0.0 and higher Version 6.4.3 and higher: Usage of SDO-based communication
Control unit	Version 2.7 and higher Version 6.01 and higher: Usage of SDO-based communication => Function can be processed independent of control unit's execution state, e.g. while APOSS program executes

ZbMoc AxisParamWrite

```
(
UNSIGNED16 h,
UNSIGNED16 id,
UNSIGNED8 axnr,
ZbMocAxisParamS* para
)
```

Summary

Writes the axis parameters of the given axis number into the selected control unit. The parameters are taken out of the 'ZbMocAxisParamS' structure, prepocessed (if necessary) and saved in the control unit.

Parameter

h	Handle of the interface
id	Identification number (ID) of the control unit
axnr	Axis number (1...n) of the control unit
para	Pointer to a data section of type 'ZbMocAxisParamS' holding the parameters

Return value

SIGNED16	Error code

Remarks

All axis parameters exist on its own for every axis of the control unit.

This function sets all axis parameters which are supported by the connected control unit and version. All other given parameters are ignored.

If there is any conversion necessary to get compatible results corresponding to the version of the control unit, this is done internally before writing the setting.

The structure 'ZbMocAxisParamS' is specified in section “Data Types and Structures”.

Portability

zbmoc.dll	Version 6.0.0 and higher Version 6.4.3 and higher: Usage of SDO-based communication
Control unit	Version 2.7 and higher Version 6.01 and higher: Usage of SDO-based communication => Function can be processed independent of control unit's execution state, e.g. while APOSS program executes

ZbMoc AxisParam WriteRaw

```
(
UNSIGNED16 h,
UNSIGNED16 id,
UNSIGNED8 axnr,
UNSIGNED32* param,
UNSIGNED16 first,
UNSIGNED16 last
)
```

Summary

Writes the defined number of parameters of the given axis number into the selected control unit. The parameter values are copied into the control unit exactly in the way they are given in the array. There is no conversion or decoding.

Parameter

h	Handle of the interface
id	Identification number (ID) of the control unit.
axnr	Axis number (1...n) of the control unit
param	Pointer to the start of an array holding the parameter values to write
first	Number of the first parameter to write: 0..n-1
last	Number of the last parameter to write: 0..n-1

Return value

SIGNED16	Error code

Remarks

It has to be made sure, that the requested parameters exist for the connected control unit and version ! If not, there is an error.

The array holding the parameter values has to have the correct size. The array size is not checked by this function !

Due to the fact that the parameters are not preprocessed (like the 'ZbMocAxisParamWrite' function does), it is a must to take care of the control units version. This is especially important for upgrades of control units. It is strongly recommended to use the 'ZbMocAxisParamWrite' function instead to avoid such problems.

This function is intended for internal use only !

Portability

zbmoc.dll	Version 6.0.0 and higher Version 6.4.3 and higher: Usage of SDO-based communication
Control unit	Version 2.7 and higher Version 6.01 and higher: Usage of SDO-based communication => Function can be processed independent of control unit's execution state, e.g. while APOSS program executes

ZbMoc BlobInfoRequest

```
(
UNSIGNED16 h,
UNSIGNED16 id,
ZbMocBlobInfoS* blobinfos,
UNSIGNED16* numentries,
UNSIGNED16 maxentries
)
```

Summary

Gives the internal program information headers of the source code stored in the selected control unit.

Parameter

h	Handle of the interface
id	Identification number (ID) of the control unit
blobinfos	Pointer to an array of structures taking the program information
numentries	Pointer to a variable taking the actual number of program information headers in the array
maxentries	Maximum number of program information structures in the array provided for taking program information

Return value

SIGNED16	Error code

Remarks

Program source code (or any other kind of data) can be stored in the control unit in so-called Blobs (= Binary large objects). Each Blob is identified by a number, which is unique in the list of all Blobs. This function returns the identification number of each Blob as part of the Program Information Header. This number can be used to get access to the program source code by using the function 'ZbMocBlobRead'.

The APOSS application program uses the 'ZbMocBlob...' functions to manage the storage of source code on request. The source code is therefor compressed in case of downloading and decompressed in case of uploading again. The compression is part of the application program and not of the ZbMoc functions. The APOSS application program uses the Archive Library 2.12 by Greenleaf Software Inc. for this task.

There is no internal link between the binary program code and the corresponding source code. So, the application is responsible to maintain a fixed method to manage corresponding program binary and source code. APOSS does this by defining an identical number for the program binary code (as shown in the program list) and the Blob holding the program source code.

This function is intended for internal use only !

This function is not available for serial (V24) communication below DLL-version 6.4.3. and control unit version 6.0.1. !

Portability

zbmoc.dll	Version 6.0.0 and higher Version 6.4.3 and higher: Usage of SDO-based communication
Control unit	Version 5.17 and higher Version 6.01 and higher: Usage of SDO-based communication => Function can be processed independent of control unit's execution state, e.g. while APOSS program executes

ZbMoc BlobRead

```
(
UNSIGNED16 h,
UNSIGNED16 id,
UNSIGNED16 blobnum,
UNSIGNED8* blobdata
UNSIGNED32* blobdatalen,
UNSIGNED32 buflen
)
```

Summary Reads out program source code, that was saved in the selected control unit.

Parameter

h	Handle of the interface
id	Identification number (ID) of the control unit
blobnum	Identification number of the Blob holding the program source code as given in the program information header
blobdata	Pointer to a byte array taking the received compressed program source code or the required length, if the provided buffer was too small
blobdatalen	Pointer to a variable receiving the length of the compressed program source code in bytes
buflen	Size of the byte array provided for taking the program source

Return value

SIGNED16	Error code

Remarks

The parameter 'blobnum' is used as an identification number of the Blob (= Binary large object) holding the program source code. This number is part of the structure that is returned by the 'ZbMocBlobInfoRequest' function.

There is no internal link between binary program code and source code. So, the application is responsible to maintain a fixed method to manage corresponding program binary and source code. APOSS does this by defining an identical number for the program binary code (as shown in the program list) and the Blob holding the program source code.

The read out source code has a compressed format in case it was originally written by the APOSS application software. The source code has to be decompressed using the corresponding function of the Archive Library 2.12 by Greenleaf Software Inc.

The provided buffer has to be at least 8 bytes bigger than the actual blob size. If the provided buffer is too small, an error code (e.g. ZBMOC_E_RXBUFFER_TOO_SMALL) is returned and the actual blob size is given in blobdatalen.

This function is intended for internal use only !

This function is not available for serial (V24) communication below DLL-version 6.4.3. and control unit version 6.0.1. !

Portability

zbmoc.dll	Version 6.0.0 and higher Version 6.4.3 and higher: Usage of SDO-based communication
Control unit	Version 5.17 and higher Version 6.01 and higher: Usage of SDO-based communication => Function can be processed independent of control unit's execution state, e.g. while APOSS program executes

ZbMoc BlobWrite

```
(
UNSIGNED16 h,
UNSIGNED16 id,
UNSIGNED16 blobnum,
UNSIGNED8* blobdata
UNSIGNED32 blobdatalen
)
```

Summary Transmits program source code for storage in the selected control unit.

Parameter

h	Handle of the interface
id	Identification number (ID) of the control unit
blobnum	Identification number of the Blob, which has to be used to save the program source code
blobdata	Pointer to a byte array holding the compressed program source code data to transmit
blobdatalen	Number of compressed program source code bytes in the given array. Exactly this number of bytes are transmitted.

Return value

SIGNED16	Error code

Remarks

The parameter 'blobnum' is used as an identifier for the Blob and the program source code stored in it and has to be unique for all Blobs (= Binary large objects). It is recommended to use the same number for the binary program (in the program list) and the program source code Blob, which gives an obvious relation between binary and source code.

It is recommended to use the compressing function of the Archive Library 2.12 by Greenleaf Software Inc before calling up this function. This gives full compatibility with source code written into the control unit by the APOSS application software and saves a lot of memory space.

There is no internal link between binary program code and source code. So, the application is responsible to maintain a fixed method to manage corresponding program binary and source code. APOSS does this by defining an identical number, as explained above, for the program binary code (as shown in the program list) and the Blob holding the program source code.

This function is intended for internal use only !

This function is not available for serial (V24) communication below DLL-version 6.4.3. and control unit version 6.0.1. !

Portability

zbmoc.dll	Version 6.0.0 and higher Version 6.4.3 and higher: Usage of SDO-based communication
Control unit	Version 5.17 and higher Version 6.01 and higher: Usage of SDO-based communication => Function can be processed independent of control unit's execution state, e.g. while APOSS program executes

ZbMoc Break

```
(
UNSIGNED16 h,
UNSIGNED16 id
)
```

Summary Transmits a 'Break' command to the selected control unit, which initiates the following actions:
* Stops program execution (and also autostart)
* Stops all moving axis
* Clears an error state, if the control unit is in error state

Parameter

h	Handle of the interface
id	Identification number (ID) of the control unit

Return value

SIGNED16	Error code

Remarks Depending on the type of interfaces and control unit it might be necessary to call this function repeated to stop a program, which has the 'Autostart' flag set. The reason is that the first 'Break' command results in a restart. A 'Break' command acting on an 'autostart' program results in an error state no.19.

Portability

zbmoc.dll	Version 6.0.0 and higher
Control unit	Version 2.7 and higher

ZbMocBreakAll

(

UNSIGNED16 h

)

Summary Transmits a 'Break' command to ALL control units connected to the selected interface, i.e. it is kind of a 'Break' broadcast message.

From the point of view of a single control unit this function is identical to 'ZbMocBreak'.

Parameter

h	Handle of the interface

Return value

SIGNED16	Error code

Remarks Control units connected to CAN bus or other multi point interfaces are only concerned, if their IDs are within the range defined in the 'ZbMocOpen...' function. Other control units which may be connected to the same bus are not influenced at all.

Portability

zbmoc.dll	Version 6.0.0 and higher
Control unit	Version 2.7 and higher

ZbMoc CanOpen SlaveReadSDO

```
(
UNSIGNED16 h,
UNSIGNED16 id,
UNSIGNED16 index,
UNSIGNED8 subindex
UNSIGNED8* value,
UNSIGNED32 len
UNSIGNED32 *len_rx_data
)
```

Summary Reads data out of the SDO of the selected control unit, which has to be configured as a CAN open slave.

Parameter

h	Handle of the interface
id	Identification number (ID) of the control unit
index	Index of the SDO
subindex	Subindex of the SDO
value	Pointer to a byte array taking the received SDO data
len	Size of the provided array for taking the received data
len_rx_data	Pointer to a variable taking the actual number of received bytes

Return value SIGNED16 Error code

Remarks The array taking the received values has to have a size of typically 4 bytes, which is the maximum number of received bytes. If the provided array is smaller (according to the given length 'len') any additonal received bytes are ignored. If the returned value in 'len_rx_data' is bigger than the given parameter 'len', part of the received SDO data was ignored.

Portability

zbmoc.dll	Version 6.0.0 and higher
Control unit	Version 5.21 and higher

ZbMoc
CanOpen
SlaveStart

(
UNSIGNED16 h,
UNSIGNED16 id
)

Summary Configures and starts the selected control unit as a CAN open slave.

Parameter

h	Handle of the interface
id	Identification number (ID) of the control unit

Return value

SIGNED16	Error code

Portability

zbmoc.dll	Version 6.0.0 and higher
Control unit	Version 5.21 and higher

ZbMoc CanOpen SlaveWriteSDO

```
(
UNSIGNED16 h,
UNSIGNED16 id,
UNSIGNED16 index,
UNSIGNED8 subindex
UNSIGNED8* value,
UNSIGNED8 len
)
```

Summary Transmits data to the SDO of the selected control unit configured as a CAN open slave.

Parameter

h	Handle of the interface
id	Identification number (ID) of the control unit
index	Index of the SDO
subindex	Subindex of the SDO
value	Pointer to array of bytes, holding the data, which has to be transmitted to the SDO
len	Size of the array, i.e. number of bytes in the array

Return value SIGNED16 Error code

Remarks The array holding the data to transmit has to have a size of typically 4 bytes, which is the maximum number of data bytes.

There are always 4 data bytes transmitted to the SDO. It is the responsibility of the application to provide in minumum the correct number of data bytes for the selected SDO. The contents of any additional, not required data bytes are ignored by the SDO. Thereby it is possible to use always an array of 4 bytes length.

Portability

zbmoc.dll	Version 6.0.0 and higher
Control unit	Version 5.21 and higher

ZbMoc CanReadRaw

```
(
UNSIGNED16 h,
UNSIGNED16 id,
UNSIGNED8* msg,
UNSIGNED32 timeout
)
```

Summary Reads a single (raw) CAN message of the selected control unit.

Parameter

h	Handle of the interface
id	Identification number (ID) of the control unit
msg	Pointer to a data section of 8 bytes length taking the received CAN message data
timeout	Maximum period of time [ms] for receiving the message 0 = Do not wait, i.e. a message has to present immediately If there is no message received within the defined time slot, the function returns an error code.

Return value

SIGNED16	Error code: ZBMOC_OK = Message was received and is copied to the data buffer ZBMOC_E_CAN_TIMEOUT = No message received within time slot

Remarks This function is only available for CAN bus based communication.

The user data section of the CAN message has a length of 8 bytes. The data is copied exactly the way it had been received into the given 'msg' data buffer.

Portability

zbmoc.dll	Version 6.0.0 and higher
Control unit	Version 2.7 and higher

ZbMoc CanWriteRaw

```
(
UNSIGNED16 h,
UNSIGNED16 id,
UNSIGNED8* msg
)
```

Summary

Transmits a single CAN message to the selected control unit.

The user data section of the CAN message has a length of 8 bytes. The data is transmitted exactly the way it is given in the 'msg' data buffer.

Parameter

h	Handle of the interface
id	Identification number (ID) of the control unit
msg	Pointer to a data section of 8 bytes length, holding the user data of the CAN message to transmit

Return value

SIGNED16	Error code

Remarks

This function is only available for CAN bus based communication.

Portability

zbmoc.dll	Version 6.0.0 and higher
Control unit	Version 2.7 and higher

ZbMoc ClearError

```
(
UNSIGNED16 h,
UNSIGNED16 id
)
```

Summary Clears an error state of the communication layer of the given interface.

Parameter

h	Handle of the interface
id	Identification number (ID) of the control unit

Return value

SIGNED16	Error code

Remarks If an error state was present, the communication layer is resetted, i.e. the transmit and receive buffers of the interface are cleared.

Portability

zbmoc.dll	Version 6.0.0 and higher
Steuerung	Version 2.7 and higher

ZbMoc Close (
UNSIGNED16 h
)

Summary Closes the given interface.

Parameter

h	Handle of the interface

Return value

SIGNED16	Error code

Remarks This function should be called before an application program exits.

Portability

zbmoc.dll	Version 6.0.0 and higher
Control unit	Version 2.7 and higher

ZbMoc
CloseAll
(
void
)

Summary Closes all open interfaces.

Parameter –

Return value SIGNED16 Error code

Remarks This function can be used instead of 'ZbMocClose'. It does not make any difference, because the ZbMoc library only supports one connection at a time.

Portability

zbmoc.dll	Version 6.0.0 and higher
Steuerung	Version 2.7 and higher

ZbMoc CNFRestore FromFile

```
(
UNSIGNED16 h,
UNSIGNED16 id,
const char* filename
)
```

Summary Downloads the data (parameters and content of arrays) of a configuration file (*.cnf) to the selected control unit.

Parameter

h	Handle of the interface
id	Identification number (ID) of the control unit
filename	Pointer to a string holding the filename and path information of the CNF-file, whichs data has to be written into the control unit

Return value

SIGNED16	Error code

Remarks Due to the usage of the SDO-based communication the function can be executed independent of the control unit's execution state. It has to be taken into account that saving of huge arrays (i.e. with all together more than 1000 elements) might influence the speed of program execution significantly, because there is no APOSS command and interrupt or error handling processed during the final persistent saving of the arrays. It is recommended to stop time, process or safty critical programs in advance by using the 'ZbMocBreak' function !

Portability

zbmoc.dll	Version 6.4.3 and higher
Control unit	Version 2.7 and higher Version 6.01 and higher: Usage of SDO-based communication => Function can be processed independent of control unit's execution state, e.g. while APOSS program executes

ZbMoc CNFSave ToFile

```
(
UNSIGNED16 h,
UNSIGNED16 id,
const char* filename
)
```

Summary Reads out the parameter settings and arrays definitions of the given control unit and writes these configuration data into the configuration file (*.cnf) with the given name.

Parameter

h	Handle of the interface
id	Identification number (ID) of the control unit
filename	Pointer to a string holding the filename and path information of the CNF-file into which the configuration read out of the control unit is written

Return value

SIGNED16	Error code

Remarks If the given CNF-file exists, it is overwritten !

Portability

zbmoc.dll	Version 6.4.3 and higher
Control unit	Version 2.7 and higher Version 6.01 and higher: Usage of SDO-based communication => Function can be processed independent of control unit's execution state, e.g. while APOSS program executes

ZbMoc Continue

```
(
UNSIGNED16 h,
UNSIGNED16 id
)
```

Summary Continues a program that was interrupted by a 'Break' command before.

Parameter

h	Handle of the interface
id	Identification number (ID) of the control unit

Return value

SIGNED16	Error code

Portability

zbmoc.dll	Version 6.0.0 and higher
Control unit	Version 2.7 and higher

ZbMoc
DebugPrintf (
UNSIGNED8* format,
...
)

Summary Inserts a line in the debug log file.

Parameter

format	Format string corresponding to the standard C 'printf' function
…	Variable parameter list corresponding to the standard C 'printf' function

Return value -

Portability

zbmoc.dll	Version 6.0.0 and higher
Control unit	Version 2.7 and higher

ZbMoc Exec (
UNSIGNED16 h,
UNSIGNED16 id,
UNSIGNED16 prgnr
)

Summary Starts the program with the given program number of the selected control unit.

Parameter

h	Handle of the interface
id	Identification number (ID) of the control unit
prgnr	Program number

Return value

SIGNED16	Error code

Portability

zbmoc.dll	Version 6.0.0 and higher
Control unit	Version 2.7 and higher

ZbMoc ExecTemp

```
(
UNSIGNED16 h,
UNSIGNED16 id,
BOOLEAN init
)
```

Summary Starts the program that was temporarily downloaded before.

Parameter

h	Handle of the interface
id	Identification number (ID) of the control unit
init	TRUE (recommended), if internal variables (e.g. MNU point, etc.) have to be initialized first

Return value

SIGNED16	Error code

Remarks The temporary program is the program, that was downloaded last and is not stored under a program number up to now.

See also the 'ZbMocProgramSave' and 'ZbMocProgramSaveAs' functions.

Portability

zbmoc.dll	Version 6.0.0 and higher
Control unit	Version 2.7 and higher

ZbMoc GetAxisPosition

```
(
UNSIGNED16 hdl,
UNSIGNED16 id,
UNSIGNED8 axnr,
SIGNED32* position
)
```

Summary Reads the position of the given axis of the selected control unit.

Parameter

h	Handle of the interface
id	Identification number (ID) of the control unit
axnr	Number of the axis: 1…n
position	Pointer to a variable taking the returned position value

Return value

SIGNED16	Error code

Remarks The application program calling up this function is responsible that the given axis number exists in the selected control unit.

Portability

zbmoc.dll	Version 6.0.0 and higher
Control unit	Version 2.7 and higher

ZbMoc GetDllVersion ( )

Summary Returns the version string of the ZbMoc library.

Parameter –

Return value

UNSIGNED8*	Pointer to the version string, e.g. “ZBMOC32.DLL Interface Driver 6.2.2”

Portability

zbmoc.dll	Version 6.0.0 and higher
Control unit	Version 2.7 and higher

ZbMoc
GetError (
UNSIGNED16 h
)

Summary Returns the error state of the communication layer of the given interface.

Parameter

h	Handle of the interface.

Return value

SIGNED16	Error code

Remarksr An error state of the communication layer can be cleared by using the 'ZbMocClearError' function, which also resets the transmit and receive buffers.

Portability

zbmoc.dll	Version 6.0.0 and higher
Control unit	Version 2.7 and higher

ZbMoc GetIFDriverID

```
(
UNSIGNED8* driverid
)
```

Summary Gives the version string of the interface low level driver currently in use, if so.

Parameter

driverid	Pointer to a string of at least 80 characters taking the identification string of the low level driver.

Return value

SIGNED16	Error code

Remarks The given identification string depends on the interface currently open.
If no interface is open or if the interface is based on an additional driver (that is part of the operating system) or if the driver does not provide access to the version information, a 'ZBMOC_E_UNAVAILABLE' error is returned.

Portability

zbmoc.dll	Version 6.4.3 and higher
Control unit	Version 2.7 and higher

ZbMoc GetNumber

(
UNSIGNED16 h
)

Summary Returns the number of connected control units of the given interface.

Parameter

h	Handle of the interface.

Return value

SIGNED16	Number of control units

Remarks Control units, which have an ID, that was not within the ID range defined during execution of the 'ZbMocOpen...' function for the interface, do not count for the returned number.

For serial (V24) interface the number of control units returned is always 1.

See also the functions 'ZbMocMoconInfoIdx' and 'ZbMocMoconInfo', which can be used for querying more detailed information about the connected control units.

Portability

zbmoc.dll	Version 6.0.0 and higher
Control unit	Version 2.7 and higher

ZbMoc GetTextResult

```
(
UNSIGNED16 h,
UNSIGNED8* message,
UNSIGNED16 messagelen
)
```

Summary Gives the result of the last executed function in plain text.

Parameter

h	Handle of the interface
message	Pointer to a data section for taking the plain text message
messagelen	Length of the data section

Return value

BOOLEAN	TRUE or FALSE, depending if result is available or not

Remarks This function is intended for internal use only !

Portability

zbmoc.dll	Version 6.0.0 and higher
Control unit	Version 2.7 and higher

ZbMoc Mocon ChangeOutput Port

```
(
UNSIGNED16 h,
UNSIGNED16 id,
UNSIGNED16 newport
)
```

Summary Changes the default output port of the selected control unit.

Parameter

h	Handle of the interface
id	Identification number (ID) of the control unit
newport	Output Port: ZBMOC_SET_OUTPUT_SERIAL or ZBMOC_SET_OUTPUT_CAN

Return value

SIGNED16	Error code

Remarks The output port of the connected control unit is set automatically by the 'ZbMocOpen...' functions of the available interfaces. There is no modification necessary.

This function is intended for internal use only !

Portability

zbmoc.dll	Version 6.0.0 and higher
Control unit	Version 2.7 and higher

ZbMoc
Mocon
ClearError

```
(
UNSIGNED16 h,
UNSIGNED16 id
)
```

Summary Clears the error state of the selected control unit.

Parameter

h	Handle of the interface
id	Identification number (ID) of the control unit

Return value

SIGNED16	Error code

Remarks This function is not implemented up to now and returns always the error code 'ZBMOC_E_FUNCTION_UNAVAILABLE' !!!

It is recommended to use the function 'ZbMocBreak' instead to clear any error states of the control unit.

```
ZbMoc
Mocon
ClearFactory
(
UNSIGNED16 h,
UNSIGNED16 id
)
```

Summary Resets the selected control unit to the factory settings. Every parameter and program of the control unit is overwritten by default values respectively erased.

Parameter

h	Handle of the interface
id	Identification number (ID) of the control unit

Return value

SIGNED16	Error code

Remarks Every parameter and program of the control unit is overwritten by default values respectively erased.

It is not (!) recommended to use this function on MACS or VLT/SyncPos control units !! Due to hardware limitations of the MACS and VLT/SyncPos control units, it is not possible to reset these control units completely by a software command. If this function is used on one of these units, it is a must that the control unit is switched off and on again afterwards before any other function is executed !

It is recommended to reset the complete memory content of a control unit by calling up the function 'ZbMocMoconDeleteEeprom' and switching the unit off and on afterwards !

Portability

zbmoc.dll	Version 6.0.0 and higher
Control unit	Version 2.7 and higher

ZbMoc
Mocon
ClearParameters

```
(
UNSIGNED16 h,
UNSIGNED16 id
)
```

Summary Resets all parameters (i.e. global and axis) of the selected control unit to the factory settings.

Parameter

h	Handle of the interface
id	Identification number (ID) of the control unit

Return value

SIGNED16	Error code

Remarks Every parameter of the control unit is overwritten by default values.

Portability

zbmoc.dll	Version 6.0.0 and higher
Steuerung	Version 2.7 and higher

```
ZbMoc
Mocon
ClearPrograms
(
UNSIGNED16 h,
UNSIGNED16 id
)
```

Summary Erases all programs of the selected control unit.

Parameter	h	Handle of the interface
	id	Identification number (ID) of the control unit
Return value	SIGNED16	Error code
Portability	zbmoc.dll	Version 6.0.0 and higher
	Control unit	Version 2.7 and higher

ZbMoc
Mocon
DeleteEeprom

(
UNSIGNED16 h,
UNSIGNED16 id
)

Summary Erases the content of the non-volatile memory (EEPROM) of the control unit.

Parameter

h	Handle of the interface
id	Identification number (ID) of the control unit

Return value

SIGNED16	Error code

Remarks It is recommended to switch the control unit off and on after calling up 'ZbMocMoconDeleteEeprom'. This function does not erase the content of the volatile memory (RAM). Switching the unit off and on afterwards, makes sure that the erased content of the EEPROM is copied into the RAM again and both memory units are consistent. Otherwise there is the risk that the still valid content of the RAM is copied again into the non-volatile memory (EEPROM) by accident, e.g. saving a program.

Portability

zbmoc.dll	Version 6.0.0 and higher
Control unit	Version 2.7 and higher

ZbMoc
Mocon
ErrorGet
(
UNSIGNED16 h,
UNSIGNED16 id
)

Summary Returns the error state of the selected control unit.

Parameter

h	Handle of the interface
id	Identification number (ID) of the control unit

Return value

SIGNED16	Error code

Portability

zbmoc.dll	Version 6.0.0 and higher
Control unit	Version 2.7 and higher

ZbMoc
Mocon
ErrorGetString

```
(
UNSIGNED16 h,
UNSIGNED8* message
)
```

Summary Gives the result in plain text of the last error state message, which was received of a control unit.

Parameter

h	Handle of the interface
message	Pointer to a data section for taking the plain text message

Return value

BOOLEAN	TRUE or FALSE, depending if result is available or not

Remarks There is always a 'FALSE' returned, because this function is not fully implemented up to now !

Portability

zbmoc.dll	Version 6.0.0 and higher
Control unit	Version 2.7 and higher

ZbMoc Mocon ExecutionStatus

```
(
UNSIGNED16 h,
UNSIGNED16 id,
BOOLEAN refresh_first,
BOOLEAN* is_executing,
SIGNED16* errnum
)
```

Summary Gives the program execution state of the selected control unit.

Parameter

h	Handle of the interface
id	Identification number (ID) of the control unit
refresh_first	TRUE: Wait until updated state is available FALSE: Return buffered state
is_executing	Pointer to a variable taking the program execution state TRUE: Control unit is executing a program FALSE: Control unit does not execute a program
errnum	Pointer to a variable taking the error code of the control unit or 0, if control unit is not in error state

Return value SIGNED16 Error code

Remarks In case of a CAN bus communication the state is requested with every polling command.

It is important to set the parameter 'refresh_first' to TRUE, in case of a connection to a VLT. Otherwise the state is not up-to-date ! Nevertheless it has to considered that calling up this function frequently may cause a performance problem. If the function 'ZbMocPollMessage' is used for frequent polling anyway, it is possible to set 'refresh_first' to FALSE, because the function 'ZbMocPollMessage' also updates the status.

Portability

zbmoc.dll	Version 6.0.0 and higher	
Control unit	Version 2.7 and higher:	No support of the V24 serial interface
	Version 5.18 and higher:	Support for all types of interfaces

ZbMocMoconInfo

```
(
UNSIGNED16 h,
UNSIGNED16 id
)
```

Summary Returns information (= description sector) about the selected control unit.

Parameter

h	Handle of the interface
id	Identification number (ID) of the control unit

Return value

ZbMocMoconInfoS*	Pointer to a structure holding information about the control unit or NULL, if no control unit with the given ID exists

Remarks The returned information (description sector) is read out of the control unit during execution of the 'ZbMocOpen...' function. The information can be read out again (e.g. for update purposes) by calling up the function 'ZbMocMoconInfoControlledRefresh'. In either case there are only control units considered, which are within the ID-range defined in the 'ZbMocOpen...' function.

Portability

zbmoc.dll	Version 6.0.0 and higher
Control unit	Version 2.7 and higher

```
ZbMoc
Mocon
InfoIdx
(
UNSIGNED16 h,
UNSIGNED16 idx
)
```

Summary Returns information (= description sector) about the selected control unit.

Parameter

h	Handle of the interface
idx	Index of the control unit: 0…ZbMocGetNumber()-1

Return value

ZbMocMoconInfoS*	Pointer to a structure holding information about the control unit or NULL, if no control unit with the given ID exists

Remarks The returned description sector is read out of the control unit during execution of the 'ZbMocOpen...' function. The information can be read out again (e.g. for update purposes) by calling up the function 'ZbMocMoconInfoControlledRefresh'. In either case there are only control units considered, which are within the ID-range defined in the 'ZbMocOpen...' function.

Portability

zbmoc.dll	Version 6.0.0 and higher
Control unit	Version 2.7 and higher

ZbMoc Mocon Info ControlledRefresh

```
(
UNSIGNED16 h
BOOLEAN breakall
)
```

Summary Reads out the information (description sector) of all connected control units. The information can be requested afterwards by using the 'ZbMocMoconInfo' and 'ZbMocMoconInfoIdx' functions.

Parameter

h	Handle of the interface
breakall	TRUE: All control units are stopped before retrieving the info FALSE: No stop is executed during the attempt to refresh the info

Return value

SIGNED16	Number of control units or Error code

Remarks Naturally it is not necessary to call up this function, because it is executed automatically by the 'ZbMocOpen...' functions.

If parameter 'breakall' is set to FALSE, just the SDO-based communication is used to retrieve the information about the control units. This might fail, if there is at least one control unit, which is not capable of the SDO-based communication or SDO-based communication is blocked due to the configuration (see ZbMocDefineUsage) or SDO-based communication fails out of unknown reasons. If SDO-based communication ios not capable to retrieve the information of each control unit, a 'ZBMOC_E_BREAK_REQUIRED' is returned.If no control unit at all is responding a 'ZBMOC_E_NOCONTROLLER' error code is returned.

If not all of the connected control units respond a 'ZBMOC_E_CONTROLLER_MISSING' error code is reported.
The 'ZBMOC_E_NOCONTROLLER' or 'ZBMOC_E_CONTROLLER_MISSING' can take place, if a APOSS program is executing on a control unit and the control unit is not capable of the newer SDO-based communication or the SDO-based communication is blocked by the 'ZbMocSDODefineUsage(ZBMOC_SDO_NONE)' function. Naturally these error messages can also take place in case of a noisy communication line or in fact there is no control unit present. If one of these errors is reported, it is recommended to execute the 'ZbMocBreakAll' function and try 'ZbMocMoconInfoControlledRefresh' again.

Portability

zbmoc.dll	Version 6.4.11 and higher
Control unit	Version 2.7 and higher Version 6.01 and higher: Usage of SDO-based communication => Function can be processed independent of control unit's execution state, e.g. while APOSS program executes

ZbMoc Mocon InfoRefresh (
UNSIGNED16 h
)

Summary Reads out the information (description sector) of all connected control units. The information can be requested afterwards by using the 'ZbMocMoconInfo' and 'ZbMocMoconInfoIdx' functions.

Parameter

h	Handle of the interface

Return value

SIGNED16	Number of control units or Error code

Remarks Naturally it is not necessary to call up this function, because it is executed automatically by the 'ZbMocOpen...' functions. If refreshing of the information is wanted out of whatever reason, it is recommended to use 'ZbMocMoconInfoControlledRefresh' instead of this function.

If no control unit at all is responding a 'ZBMOC_E_NOCONTROLLER' error code is returned. If not all of the connected control units respond a 'ZBMOC_E_CONTROLLER_MISSING' error code is reported.
The 'ZBMOC_E_NOCONTROLLER' or 'ZBMOC_E_CONTROLLER_MISSING' can take place, if a APOSS program is executing on a control unit and the control unit is not capable of the newer SDO-based communication or the SDO-based communication is blocked by the 'ZbMocSDODefineUsage(ZBMOC_SDO_NONE)' function. Naturally these error messages can also take place in case of a noisy communication line or in fact there is no control unit present. If one of these errors is reported, it is recommended to execute the 'ZbMocBreakAll' function and try 'ZbMocMoconInfoRefresh' again.

Portability

zbmoc.dll	Version 6.0.0 and higher Version 6.4.3 and higher: Usage of SDO-based communication
Control unit	Version 2.7 and higher Version 6.01 and higher: Usage of SDO-based communication => Function can be processed independent of control unit's execution state, e.g. while APOSS program executes

ZbMoc
Mocon
NameSet

```
(
UNSIGNED16 hdl,
UNSIGNED16 id,
const UNSIGNED8 *name
)
```

Summary Defines the ASCII-based project name of the selected control unit.

Parameter

hdl	Handle of the interface
id	Identification number (ID) of the control unit
name	Pointer to a data section holding the name

Return value

SIGNED16	Error code

Remarks It is not allowed to use any special characters for the name. It can have a length of 10 characters in maximum.

Portability

zbmoc.dll	Version 6.0.0 and higher Version 6.4.3 and higher: Usage of SDO-based communication
Control unit	Version 2.7 and higher Version 6.01 and higher: Usage of SDO-based communication => Function can be processed independent of control unit's execution state, e.g. while APOSS program executes

```
ZbMoc
Mocon
SaveRam
(
UNSIGNED16 h,
UNSIGNED16 id
)
```

Summary Saves the content of the RAM in non-volatile memory of the selected control unit.

Parameter

h	Handle of the interface
id	Identification number (ID) of the control unit

Return value

SIGNED16	Error code

Remarks The execution of this command might take some time (up to 20 seconds), depending on the type of non-volatile memory used in the control unit.

If the control unit has battery-buffered RAM, it is not necessary to execute this function and it even has any effect at all.

Portability

zbmoc.dll	Version 6.0.0 and higher
Control unit	Version 2.7 and higher

ZbMoc Mocon SaveRamSection

```
(
UNSIGNED16 h,
UNSIGNED16 id
UNSIGNED8 section
)
```

Summary Saves the content of the specified RAM sectionin non-volatile memory of the selected control unit.

Parameter

h	Handle of the interface	
id	Identification number (ID) of the control unit	
section	ID of the RAM section for saving:	
	MOC_RAMSECTION_AXISPAR	-> Save axis parameters
	MOC_RAMSECTION_GLOBALPAR	-> Save global parameters
	MOC_RAMSECTION_PROGRAMS	-> Save programs
	MOC_RAMSECTION_ARRAYS	-> Save arrays
	MOC_RAMSECTION_ALL	-> Save all

Return value SIGNED16 Error code

Remarks This function and the selective saving of a RAM section is available only for control units with an internal firmware version equal or higher V6.4.3 and SDO-based communication is not blocked (see 'ZbMocSDODefineUsage' and 'ZbMocSDOGetUsage' functions).

Due to the usage of the SDO-based communication the function can be executed independent of the control unit's execution state. It has to be taken into account that saving of programs, arrays or the complete content of RAM influences the speed of program execution strongly. There is no APOSS command and interrupt or error handling processed during execution of this function, this means that the prorgam might not react for some seconds depending on the selected RAM section.

It is strongly recommended not to call this function for saving of the program, an array section or complete RAM while the control unit is in execution state !! Use the 'ZbMocBreak' function to stop execution state in advance !

If the control unit has an older firmware version that does not support the required functionality or if SDO-based communication is blocked, the function 'ZbMocMoconSaveRamSection' returns 'ZBMOC_E_UNAVAILABLE'. In this case the function 'ZbMocMoconSaveRam' has to be used to save the entire content of the RAM persistently instead.

The execution of this command might take some seconds, depending on the type of non-volatile memory used in the control unit and the RAM section selected.

If the control unit has battery-buffered RAM, it is not necessary to execute this function and it even has any effect at all.

Portability

zbmoc.dll	Version 6.4.3 and higher
Control unit	Version 6.0.1 and higher

ZbMoc
Mocon
SetCanParameters

(
UNSIGNED16 h,
UNSIGNED16 id,
UNSIGNED8 cannr,
UNSIGNED8 baud
)

Summary Defines the CAN communication parameters of the selected control unit.

Parameter

h	Handle of the interface
id	Identification number (ID) of the control unit
cannr	New CAN ID
baud	New CAN baud rate: 0 = 5kbps, 1 = 10kbps, 2 = 20kbps, 3 = 50kbps, 4 = 100kbps, 5 = 125kbps, 6 = 250kbps, 7 = 500kbps, 8 = 1000kbps

Return value

SIGNED16	Error code

Remarks It is recommended to use this function only, if communication via serial port is in use. Any modifications to the CAN parameters are valid immediately. This results in the fact that the control unit is not accessible for the library with the new defined CAN ID until a new 'ZbMocOpen...' function on the CAN bus takes place.

Naturally these parameters are set just during initial operation using the APOSS Integrated Development Environment.

Portability

zbmoc.dll	Version 6.0.0 and higher
Control unit	Version 2.7 and higher

ZbMocMoconStatusGet

```
(
UNSIGNED16 h,
UNSIGNED16 id,
ZbMocStatusS* s
)
```

Summary Gives the status information of the selected control unit.

Parameter

h	Handle of the interface
id	Identification number (ID) of the control unit
s	Pointer to a structure taking the status information

Return value

SIGNED16	Error code

Portability

zbmoc.dll	Version 6.0.0 and higher
Control unit	Version 2.7 and higher

ZbMoc
Mocon
WaitForString

```
(
UNSIGNED16 h,
UNSIGNED16 id,
char* target,
UNSIGNED32 timeout
)
```

Summary Waits until the given string is received via serial port of the selected control unit.

Parameter

h	Handle of the interface
id	Identification number (ID) of the control unit
target	Pointer to a character array, holding the string to wait for
timeout	Maximum period of time [ms] for receiving the given string 0 = Do not wait, i.e. a string has to present immediately If the string is not received within the defined time slot, the function returns an error code.

Return value

SIGNED16	Error code

Remarks Strings differing from the defined one are ignored.

This function is only available, if serial (V24) communication is used.

Portability

zbmoc.dll	Version 6.0.0 and higher
Control unit	Version 2.7 and higher

ZbMoc MoveStart

```
(
UNSIGNED16 hdl,
UNSIGNED16 id,
UNSIGNED8 axnr,
SIGNED32 accel,
SIGNED32 velo
)
```

Summary Starts a continuous movement of the given axis and control unit.

Parameter

h	Handle of the interface
id	Identification number (ID) of the control unit
axnr	Axis number (1...n) of the control unit
accel	Acceleration
velo	Velocity

Return value

SIGNED16	Error code

Remarks The scaling of the parameters acceleration 'accel' and velocity 'velo' depends on the setting of the axis parameter no.22 'VELRES'. If the default setting (100) of this parameter is untouched, the parameters 'accel' and 'velo' represent a percentage value of the absolute maximum values also defined by the corresponding parameters.

Portability

zbmoc.dll	Version 6.0.0 and higher
Control unit	Version 2.7 and higher

ZbMoc MoveStop

```
(
UNSIGNED16 hdl,
UNSIGNED16 id,
UNSIGNED8 axnr,
SIGNED32* newpos
)
```

Summary Stops a movement of the given axis and control unit.

Parameter

h	Handle of the interface
id	Identification number (ID) of the control unit
axnr	Achsen Nummer
newpos	Pointer to a variable taking the position value after stop

Return value

SIGNED16	Error code

Portability

zbmoc.dll	Version 6.0.0 and higher
Control unit	Version 2.7 and higher

ZbMoc OpenCanGin

```
(
UNSIGNED16 ioadr,
UNSIGNED16 irq,
UNSIGNED16 baud,
UNSIGNED16 mode,
UNSIGNED16 scanfrom,
UNSIGNED16 scanto
BOOLEAN breakall
)
```

Summary

Opens the CAN interface using the GIN adapter board.

Parameter

ioadr	IO address used by the adapter board	
irq	Interrupt used by the adapter board	
baud	Baud rate of the CAN communication: 0 = 5kbps, 1 = 10kbps, 2 = 20kbps, 3 = 50kbps, 4 = 100kbps, 5 = 125kbps, 6 = 250kbps, 7 = 500kbps, 8 = 1000kbps	
mode	Always 0, up to now	
scanfrom	First ID, checked for presence on the CAN bus	
scanto	Last ID, checked for presence on the CAN bus	
breakall	TRUE:	All control units in execution state are stopped and the connection is established, if possible
	FALSE:	If there is at least one (old) control unit, which requires a stop to establish connection, the opening is not performed and returns with an error state

Return value

SIGNED16	>= 0:	Handle of the interface connection
	< 0:	Error code

Remarks

It is not possible to open more than one interface, up to now.

Control units, which have an ID outside the defined range, are ignored.

All control units (within the defined ID range) are addressed and configured automatically for usage by the library. The version information and configuration of the each control unit is read out (via an internal call up of the function 'ZbMocMoconInfoControlledRefresh') and is saved for later access by the application program via the 'ZbMocMoconInfo' function.

If control units with a firmware version (typically V6.0 or higher) that support the zub CAN Open SDO object dictionary are connected, it is possible to query control unit information without interrupting an executing APOSS program. For older control unit firmware versions it is necessary that the 'ZbMocOpenCanGin' function executes the 'ZbMocBreakAll' function before querying control unit data. Otherwise it might happen that the control unit might not respond.
If the parameter 'breakall' is set to 'FALSE', the 'ZbMocBreakAll' function is not executed in advance. This means that APOSS programs executing on the control units are not interrupted. If the usage of SDO-based communication is disabled (see 'ZbMocSDODefineUsage' function) or one of the connected control units might not support the new zub SDO object dictionary (due to a too old firmware version) a 'ZBMOC_E_BREAK_REQUIRED' error might be reported and the connection is closed again. In this case it is possible to call the 'ZbMocOpenCanGin' function again, but setting the parameter 'breakall' to 'TRUE' this time.
If the parameter 'breakall' is set to 'TRUE', the 'ZbMocBreakAll' function is executed in advance (i.e. running programs and moving axis are stopped). If no controller at all is responding a 'ZBMOC_E_NOCONTROLLER' error code is returned and the connection is

closed again. Otherwise the handle of the connection that was established is returned.

Portability

zbmoc.dll	Version 6.0.0 and higher
Control unit	Version 2.7 and higher Version 6.01 and higher: Usage of SDO-based communication => Function can be processed independent of control unit's execution state, e.g. while APOSS program executes

ZbMoc OpenCanLpt

```
(
UNSIGNED16 lptnr,
UNSIGNED16 irq,
UNSIGNED16 baud,
UNSIGNED16 mode,
UNSIGNED16 scanfrom,
UNSIGNED16 scanto
BOOLEAN breakall
)
```

Summary

Opens the CAN interface using CAN-adapter for PC's auxiliary (centronics) port.

Parameter

lptnr	Number of the auxiliary (centronics) port: 0 or 1 –1 = Check all ports automatically and evaluate corresponding IRQ.	
irq	Interrupt used by the auxiliary (centronics) port	
baud	Baud rate of the CAN communication: 0 = 5kbps, 1 = 10kbps, 2 = 20kbps, 3 = 50kbps, 4 = 100kbps, 5 = 125kbps, 6 = 250kbps, 7 = 500kbps, 8 = 1000kbps	
mode	Always 0, up to now	
scanfrom	First ID, checked for presence on the CAN bus	
scanto	Last ID, checked for presence on the CAN bus	
breakall	TRUE:	All control units in execution state are stopped and the connection is established, if possible
	FALSE:	If there is at least one (old) control unit, which requires a stop to establish connection, the opening is not performed and returns with an error state

Return value

SIGNED16	>= 0:	Handle of the interface connection
	< 0:	Error code

Remarks

It is not possible to open more than one interface, up to now.

Control units, which have an ID outside the defined range, are ignored.

All control units (within the defined ID range) are addressed and configured automatically for usage by the library. The version information and configuration of the each control unit is read out (via an internal call up of the function 'ZbMocMoconInfoControlledRefresh') and is saved for later access by the application program via the 'ZbMocMoconInfo' function.

If control units with a firmware version (typically V6.0 or higher) that support the zub CAN Open SDO object dictionary are connected, it is possible to query control unit information without interrupting an executing APOSS program. For older control unit firmware versions it is necessary that the 'ZbMocOpenCanLpt' function executes the 'ZbMocBreakAll' function before querying control unit data. Otherwise it might happen that the control unit might not respond.
If the parameter 'breakall' is set to 'FALSE', the 'ZbMocBreakAll' function is not executed in advance. This means that APOSS programs executing on the control units are not interrupted.
If the usage of SDO-based communication is disabled (see 'ZbMocSDODefineUsage' function) or one of the connected control units might not support the new zub SDO object dictionary (due to a too old firmware version) a 'ZBMOC_E_BREAK_REQUIRED' error might be reported and the connection is closed again. In this case it is possible to call the 'ZbMocOpenCanLpt' function again, but setting the parameter 'breakall' to 'TRUE' this time.
If the parameter 'breakall' is set to 'TRUE', the 'ZbMocBreakAll' function is executed in

advance (i.e. running programs and moving axis are stopped). If no controller at all is responding a 'ZBMOC_E_NOCONTROLLER' error code is returned and the connection is closed again. Otherwise the handle of the connection that was established is returned.

Portability

zbmoc.dll	Version 6.0.0 and higher
Control unit	Version 2.7 and higher Version 6.01 and higher: Usage of SDO-based communication => Function can be processed independent of control unit's execution state, e.g. while APOSS program executes

ZbMoc OpenV24

```
(
UNSIGNED16 comnr,
char* init,
UNSIGNED16 mode
BOOLEAN breakall
)
```

Summary Opens the V24 serial interface for control unit communication.

Parameter

comnr	Number of the COM port: 0, 1, …	
init	Baud rate as a string (e.g. “9600”)	
mode	Always 0, up to now	
breakall	TRUE:	All control units in execution state are stopped and the connection is established, if possible
	FALSE:	If there is at least one (old) control unit, which requires a stop to establish connection, the opening is not performed and returns with an error state

Return value

SIGNED16	>= 0:	Handle of the interface connection
	< 0:	Error code

Remarks It is not possible to open more than one interface, up to now.

The default V24 setting typically in use is:
9600 baud, 8 data bits, 1 stop bit, no parity

The detected control unit is configured automatically for usage by the library. The version information and configuration of the control unit is read out (via an internal call up of the function 'ZbMocMoconInfoControlledRefresh') and is saved for later access by the application program via the 'ZbMocMoconInfo' function.

If a control unit with a firmware version (typically V6.0 or higher) that supports the zub CAN Open SDO object dictionary is connected, it is possible to query control unit information without interrupting an executing APOSS program. For older control unit firmware versions it is necessary that the 'ZbMocOpenV24' function executes the 'ZbMocBreakAll' function before querying control unit data. Otherwise it might happen that the control unit might not respond. If the parameter 'breakall' is set to 'FALSE', the 'ZbMocBreakAll' function is not executed in advance. This means that an APOSS program executing on the control unit is not interrupted. If the usage of SDO-based communication is disabled (see 'ZbMocSDODefineUsage' function) or the connected control unit might not support the new zub SDO object dictionary (due to a too old firmware version) a 'ZBMOC_E_BREAK_REQUIRED' error might be reported and the connection is closed again. In this case it is possible to call the 'ZbMocOpenV24' function again, but setting the parameter 'breakall' to 'TRUE' this time.
If the parameter 'breakall' is set to 'TRUE', the 'ZbMocBreakAll' function is executed in advance (i.e. running programs and moving axis are stopped). If no controller at all is responding a 'ZBMOC_E_NOCONTROLLER' error code is returned and the connection is closed again. Otherwise the handle of the connection that was established is returned..

Portability

zbmoc.dll	Version 6.0.0 and higher
Control unit	Version 2.7 and higher Version 6.01 and higher: Usage of SDO-based communication => Function can be processed independent of control unit's execution state, e.g. while APOSS program executes

ZbMoc OpenVLT

```
(
UNSIGNED16 comnr,
UNSIGNED16 baud,
const char* drivername
UNSIGNED16 mode,
UNSIGNED16 scanfrom,
UNSIGNED16 scanto
BOOLEAN breakall

)
```

Summary

Opens the VLT interface using the RS485 or DCOM communication.

Parameter

comnr	Number of the COM port: 0, 1, ...	
baud	Baud rate [bps] of the VLT / RS485 communication: 300, 600, 1200, 2400, 4800, 9600	
drivername	Pointer to a string holding the MT10 communication driver name	
mode	Operating mode of the VLT interface: 0 = VLT_STANDARD (VLT/RS485 communication) 1 = VLT_TSTDCOM (Reserved for internal usage only) 2 = VLT_USEDCOM (VLT communication via DCOM)	
scanfrom	First ID, checked for presence on the VLT bus	
scanto	Last ID, checked for presence on the VLT bus	
breakall	TRUE:	All control units in execution state are stopped and the connection is established, if possible
	FALSE:	If there is at least one (old) control unit, which requires a stop to establish connection, the opening is not performed and returns with an error state

Return value

SIGNED16	>= 0:	Handle of the interface connection In case of mode = 'VLT_TSTDCOM': usability information (0 or 1)
	< 0:	Error code

Remarks

It is not possible to open more than one interface, up to now.

The default VLT setting typically in use (for the RS485 interface) is:
9600 baud, 8 data bits, 1 stop bit, even parity

The mode 'VLT_TSTDCOM' is reserved for internal usage only and returns a ZBMOC_E_UNAVAILABLE for all released ZbMoc library versions.

In case of 'VLT_USEDCOM' mode, the parameters 'comnr' and 'baud' have no meaning.

In case of 'VLT_USEDCOM' mode, the parameter 'drivername' is passed to the VLT DCOM. Up to now the following names are valid: "FC", "DPV1".

Control units, which have an ID outside the defined range, are ignored.

All control units (within the defined ID range) are addressed and configured automatically for usage by the library. The version information and configuration of the each control unit is read out (via an internal call up of the function 'ZbMocMoconInfoControlledRefresh') and is saved for later access by the application program via the 'ZbMocMoconInfo' function.

If control units with a firmware version (typically V6.0 or higher) that support the zub CAN Open SDO object dictionary are connected, it is possible to query control unit information without interrupting an executing APOSS program. For older control unit firmware versions it is necessary that the 'ZbMocOpenCanLpt' function executes the 'ZbMocBreakAll' function

before querying control unit data. Otherwise it might happen that the control unit might not respond.
If the parameter 'breakall' is set to 'FALSE', the 'ZbMocBreakAll' function is not executed in advance. This means that APOSS programs executing on the control units are not interrupted. If the usage of SDO-based communication is disabled (see 'ZbMocSDODefineUsage' function) or one of the connected control units might not support the new zub SDO object dictionary (due to a too old firmware version) a 'ZBMOC_E_BREAK_REQUIRED' error might be reported and the connection is closed again. In this case it is possible to call the 'ZbMocOpenCanLpt' function again, but setting the parameter 'breakall' to 'TRUE' this time.
If the parameter 'breakall' is set to 'TRUE', the 'ZbMocBreakAll' function is executed in advance (i.e. running programs and moving axis are stopped). If no controller at all is responding a 'ZBMOC_E_NOCONTROLLER' error code is returned and the connection is closed again. Otherwise the handle of the connection that was established is returned.

Portability

zbmoc.dll	Version 6.0.0 and higher	
Control unit	Version 2.7 and higher:	Support of mode 'VLT_STANDARD'
	Version 6.01 and higher:	Usage of SDO-based communication
	=> Function can be processed independent of control unit's execution state, e.g. while APOSS program executes	
	Version 6.4.4 and higher (i.e. Danfoss version number 4.2.1 and higher): Additional support of modes 'VLT_TSTDCOM' and 'VLT_USEDCOM'	

ZbMoc ParamCount

```
(
UNSIGNED16 h,
UNSIGNED16 id
)
```

Summary Returns the number of global parameters of the selected control unit.

Parameter

h	Handle of the interface
id	Identification number (ID) of the control unit

Return value

SIGNED16	>= 0:	Number of global parameters
	<0:	Error code

Remarks The number of global parameters depend on the type of control unit and the installed software version.

It is not necessary to know the number of global parameters for using the 'ZbMocParamRead' function. This function checks and takes the number of parameters in account on its own.

Portability

zbmoc.dll	Version 6.0.0 and higher
Control unit	Version 2.7 and higher

ZbMoc ParamRead

```
(
UNSIGNED16 h,
UNSIGNED16 id,
ZbMocParamS* para
)
```

Summary

Reads out all global parameters of the selected control unit. The parameters are decoded and copied into the 'ZbMocParamS' structure.

Parameter

h	Handle of the interface
id	Identification number (ID) of the control unit
para	Pointer to a structure taking the parameters

Return value

SIGNED16	Error code

Remarks

This function reads out all global parameters which are supported by the selected control unit. All other parameters are set to zero.

If there is any conversion necessary to get compatible results corresponding to the version of the control unit, this is done internally.

The structure 'ZbMocParamS' is specified in section “Data Types and Structures”.

Portability

zbmoc.dll	Version 6.0.0 and higher Version 6.4.3 and higher: Usage of SDO-based communication
Control unit	Version 2.7 and higher Version 6.01 and higher: Usage of SDO-based communication => Function can be processed independent of control unit's execution state, e.g. while APOSS program executes

ZbMoc ParamReadRaw

```
(
UNSIGNED16 h,
UNSIGNED16 id,
SIGNED32* param,
UNSIGNED16 first,
UNSIGNED16 last
)
```

Summary

Reads out the defined number of global parameters of the selected control unit. The parameters are copied into the given array without any modifications, i.e. no conversion or decoding takes place.

Parameter

h	Handle of the interface
id	Identification number (ID) of the control unit
param	Pointer to an array taking the read out parameter values
first	Number of the first parameter to read out: 0..n-1
last	Number of the last parameter to read out: 0..n-1

Return value

SIGNED16	Error code

Remarks

It has to be made sure, that the requested parameter exists for the connected control unit and version !

The array size has to be big enough to take all read out parameters. The array size is not checked by this function !

This function is intended for internal use only !

Portability

zbmoc.dll	Version 6.0.0 and higher Version 6.4.3 and higher: Usage of SDO-based communication
Control unit	Version 2.7 and higher Version 6.01 and higher: Usage of SDO-based communication => Function can be processed independent of control unit's execution state, e.g. while APOSS program executes

ZbMoc ParamWrite

```
(
UNSIGNED16 h,
UNSIGNED16 id,
ZbMocParamS* para
)
```

Summary Writes all global parameters of the selected control unit. The parameters are taken out of the 'ZbMocParamS' structure, preprocessed if necessary and saved in the control unit.

Parameter

h	Handle of the interface
id	Identification number (ID) of the control unit
para	Pointer to structure holding the parameters to write

Return value

SIGNED16	Error code

Remarks This function sets all global parameters which are supported by the selected control unit. All other given parameters are ignored.

If there is any conversion necessary to get compatible results corresponding to the version of the control unit, this is done internally before writing the settings.

The structure 'ZbMocParamS' is specified in section “Data Types and Structures”.

Portability

zbmoc.dll	Version 6.0.0 and higher Version 6.4.3 and higher: Usage of SDO-based communication
Control unit	Version 2.7 and higher Version 6.01 and higher: Usage of SDO-based communication => Function can be processed independent of control unit's execution state, e.g. while APOSS program executes

ZbMoc ParamWriteRaw

```
(
UNSIGNED16 h,
UNSIGNED16 id,
SIGNED32* param,
UNSIGNED16 first,
UNSIGNED16 last
)
```

Summary

Writes the defined number of global parameters of the selected control unit. The array values are copied into the global parameters of the control unit without any modifications, i.e. no conversion or decoding takes place.

Parameter

h	Handle of the interface
id	Identification number (ID) of the control unit
param	Pointer to an array holding the parameter values to write
first	Number of the first parameter to write: 0..n-1
last	Number of the last parameter to write: 0..n-1

Return value

SIGNED16	Error code

Remarks

It has to made sure, that the requested parameters exist for the connected control unit and version ! If not, there is an error.

The array holding the parameter values has to have the correct size. The array size is not checked by ths function !

Due to the fact that the parameters are not preprocessed (like the 'ZbMocParamWrite' function does), it is a must to take care of the control units version. This is especially important for upgrades of control units. It is strongly recommended to use the 'ZbMocParamWrite' function instead to avoid such problems.

This function is intended for internal use only !

Portability

zbmoc.dll	Version 6.0.0 and higher Version 6.4.3 and higher: Usage of SDO-based communication
Control unit	Version 2.7 and higher Version 6.01 and higher: Usage of SDO-based communication => Function can be processed independent of control unit's execution state, e.g. while APOSS program executes

ZbMoc PollMessage

```
(
UNSIGNED16 h,
SIGNED16 id,
UNSIGNED8* message
)
```

Summary

Gives a message which was generated by a 'print' command of the APOSS program running or by a system message generated by the selected control unit itself.

The message is copied as plain text into the given message buffer.

Parameter

h	Handle of the interface
id	Identification number (ID) of the control unit
message	Pointer to a buffer taking the message

Return value

SIGNED16	Error code

Remarks

This function offers full functionality just for the V24 serial and VLT interface. If a CAN interface is used, it is only possible to get system messages. Messages generated by the APOSS 'print' command are not available on the CAN bus.

The message buffer has to be big enough. Internally the buffer has a size of 2 kBytes.

Portability

zbmoc.dll	Version 6.0.0 and higher
Control unit	Version 2.7 and higher

ZbMoc ProgramList

```
(
UNSIGNED16 h,
UNSIGNED16 id,
ZbMocProgramS * pl,
UNSIGNED16 maxentries
)
```

Summary Gives a list of all program stored in the selected control unit.

Parameter

h	Handle of the interface
id	Identification number (ID) of the control unit
pl	Pointer to an array of 'ZbMocProgramS' structures taking the program numbers, names and the state of the 'Autostart' flag
maxentries	Number of elements in the array

Return value

SIGNED16	>=0:	Number of programs in the array (0 ... maxentries)
	<0:	Error code

Remarks The returned list of programs just covers all programs, which were stored using one of the 'ZbMocProgramSave' or 'ZbMocProgramSaveAs' functions before. If a program is just downloaded, it is stored temporarily and is not part of the list.

If there are more programs stored in the control unit than can be handled by the given array according to the size defined by parameter 'maxentries', any additional programs are ignored. The return value of programs found, is hereby always in the range of 0 up to the maximum array size defined by parameter 'maxentries', if no error took place before.

Portability

zbmoc.dll	Version 6.0.0 and higher Version 6.4.3 and higher: Usage of SDO-based communication
Control unit	Version 2.7 and higher Version 6.01 and higher: Usage of SDO-based communication => Function can be processed independent of control unit's execution state, e.g. while APOSS program executes

ZbMoc ProgramLoad

```
(
UNSIGNED16 h,
UNSIGNED16 id,
UNSIGNED16 prglen,
UNSIGNED8 (*prg)(UNSIGNED32)
)
```

Summary Downloads a binary (i.e. compiled APOSS) program to the selected control unit.

Parameter

h	Handle of the interface
id	Identification number (ID) of the control unit
prglen	Length of the binary program in number of bytes
prg	Pointer to a function returning always one byte of program code

Return value

SIGNED16	Error code

Remarks

The APOSS source program (*.m) has to be compiled by the APOSS compiler before a download can take place. The binary code has to be downloaded to the control unit. It is not possible to download APOSS source code directly.

This function 'ZbMocProgramLoad' calls up the function '(*prg)()' to get one byte of binary program code. The function '(*prg)()' is called up for each byte corresponding to the number of bytes defined by parameter 'prglen'. The function '(*prg)()' gets the index of the requested byte as the only parameter. The return value of '(*prg)()' is the corresponding binary program code byte.

The program is stored afterwards as a so-called temporary program. This means that the program has no number and no name. Such a temporary program can be started using the function 'ZbMocExecTemp' or it can be registered in the program list by using one of the 'ZbMocProgramSave' or 'ZbMocProgramSaveAs' functions. Depending on the control unit it may also be necessary to call the function 'ZbMocMoconSaveRam' finally to save the program persistently.

Portability

zbmoc.dll	Version 6.0.0 and higher
Control unit	Version 2.7 and higher

ZbMoc ProgramLoadAll

```
(
UNSIGNED16 h,
UNSIGNED16 prglen,
UNSIGNED8 (*prg)(UNSIGNED32)
)
```

Summary Downloads a binary (i.e. compiled APOSS) program to all units connected to the interface.

Parameter

h	Handle of the interface
prglen	Length of the binary program in number of bytes
prg	Pointer to a function returning always one byte of program code

Return value

SIGNED16	Error code

Remarks Control units, which have an ID, that was not within the ID range defined during execution of the 'ZbMocOpen...' function for the interface are not taken into account.

The APOSS source program (*.m) has to be compiled by the APOSS compiler before a download can take place. The binary code has to be downloaded to the control unit. It is not possible to download APOSS source code directly.

This function 'ZbMocProgramLoad' calls up the function '(*prg)()' to get one byte of binary program code. The function '(*prg)()' is called up for each byte corresponding to the number of bytes defined by parameter 'prglen'. The function '(*prg)()' gets the index of the requested byte as the only parameter. The return value of '(*prg)()' is the corresponding binary program code byte.

The program is stored afterwards as a so-called temporary program. This means that the program has no number and no name. Such a temporary program can be started using the function 'ZbMocExecTemp' or it can be registered in the program list by using one of the 'ZbMocProgramSave' or 'ZbMocProgramSaveAs' functions. Depending on the control unit it may also be necessary to call the function 'ZbMocMoconSaveRam' finally to save the program persistently.

This function is not implemented up to now and returns always the error code 'ZBMOC_E_FUNCTION_UNAVAILABLE' !!!

Portability

zbmoc.dll	Version 6.0.0 and higher
Control unit	Version 2.7 and higher

ZbMoc ProgramSave

```
(
UNSIGNED16 h,
UNSIGNED16 id,
const UNSIGNED8* name
)
```

Summary Registers the temporary program in the program list using the next available number.

Parameter

h	Handle of the interface
id	Identification number (ID) of the control unit
name	Pointer to a character array holding the program name

Return value

SIGNED16	>=0:	Number, which was used in the program list
	<0:	Error code

Remarks A program has to be downloaded using the function 'ZbMocProgramLoad' before it can be registered using the function 'ZbMocProgramSave'.

The program name can consist of maximum 8 capital letters without any special characters.

The number of the program slot which is used as an identifier later, is picked automatically by the control unit. If the program number should be defined manually or an existing program has to be overwritten, the function 'ZbMocProgramSaveAs' has to be used instead.

Depending on the control unit it may also be necessary to call the function 'ZbMocMoconSaveRam' afterwards to save the program persistently, if wanted.

Portability

zbmoc.dll	Version 6.0.0 and higher
Control unit	Version 2.7 and higher

ZbMoc ProgramSaveAs

```
(
UNSIGNED16 h,
UNSIGNED16 id,
const UNSIGNED8* name,
UNSIGNED16 prgnr
)
```

Summary Registers the temporary program in the program list using the given program number.

Parameter

h	Handle of the interface
id	Identification number (ID) of the control unit
name	Pointer to the character array holding the program name
prgnr	Number (0...94) used to register the temporary program in the list

Return value

SIGNED16	Error code

Remarks

A program has to be downloaded using the function 'ZbMocProgramLoad' before it can be registered using the function 'ZbMocProgramSaveAs'.

The program name can consist of maximum 8 capital letters without any special characters.

If a program with a number identical to the defined one exists, this (old) program is erased and the temporary program takes its place in the list.

The program turns up in the program list with the defined number. This number has to be used as an identifier to start the program.

Depending on the control unit it may also be necessary to call the function 'ZbMocMoconSaveRam' afterwards to save the program persistently, if wanted.

Portability

zbmoc.dll	Version 6.0.0 and higher
Control unit	Version 2.7 and higher

ZbMoc ReadMemPage

```
(
UNSIGNED16 h,
UNSIGNED16 id,
UNSIGNED32 pageaddress,
UNSIGNED16* crc,
UNSIGNED8* values
)
```

Summary Reads a memory page out of the selected control unit.

Parameter

h	Handle of the interface
id	Identification number (ID) of the control unit
pageaddress	Number of the memory page
crc	Pointer to a variable taking the CRC
values	Pointer to an array taking the received 256 bytes of data

Return value

SIGNED16	Error code

Remarks Each memory page has a size of 256 bytes.

This function is only available, if serial (V24) communication is used.

Portability

zbmoc.dll	Version 6.0.0 and higher
Control unit	MVS Version only

ZbMoc ReadUserArray

```
(
UNSIGNED16 h,
UNSIGNED16 id,
UNSIGNED16 arrnr,
UNSIGNED16 mxlen,
SIGNED32 *values,
UNSIGNED16 *arrsize
)
```

Summary Reads an user-defined array out of the selected control unit.

Parameter

h	Handle of the interface
id	Identification number (ID) of the control unit
arrnr	Number of the array (0...n)
mxlen	Size of the provided array, which takes the read out values
values	Pointer to an array taking the received user data
arrsize	Pointer to a variable taking the actual size of the read out user array

Return value SIGNED16 Error code

Remarks The complete array is read out of the control unit. To make sure that no range violation of the given array can take place, the maximum length of the given array has to be provided.

Arrays are numbered consecutive, starting with 0.

The actual size of the array read out of the control unit is copied to the 'arrsize' variable. If the array does not existing, an array size of 0 is returned. Checking for an array size of 0 can be used to detect the last valid array number, which is the current array number minus 1.

Portability

zbmoc.dll	Version 6.0.0 and higher Version 6.4.3 and higher: Usage of SDO-based communication
Control unit	Version 2.7 and higher Version 6.01 and higher: Usage of SDO-based communication => Function can be processed independent of control unit's execution state, e.g. while APOSS program executes

ZbMoc ReadUserVar

```
(
UNSIGNED16 h,
UNSIGNED16 id,
UNSIGNED16 *varnr,
SIGNED32 *val,
UNSIGNED32 timeout
)
```

Summary

Reads out a data message of the selected control unit. A message can be send in an APOSS program by using the 'OUTMSG' command.

The message contains always one 16 bit data value and a 32 bit data value.

Parameter

h	Handle of the interface
id	Identification number (ID) of the control unit
varnr	Pointer to a variable taking the 16 bit value
val	Pointer to a variable taking the 32 bit value
timeout	Maximum period of time [ms] for receiving the message 0 = Do not wait, i.e. a message has to present immediately If there is no message received within the defined time slot, the function returns an error code.

Return value

SIGNED16	Error code

Remarks

This function is not available for serial (V24) communication !

Portability

zbmoc.dll	Version 6.0.0 and higher
Control unit	Version 2.7 and higher

ZbMoc SDOBlockUsage

(
BOOLEAN block
)

Summary Blocks the internal usage of the SDO-based communication, which would be used otherwise for ZBMoc function processing in case of a control unit supporting the zub SDO object dictionary.

Parameter

block	TRUE:	do not use SDOs, independent of control unit's version
	FALSE:	use SDOs,if supported by the control unit

Return value _

Remarks A lot of functions switch to the internal usage of SDO-based comunication for control units supporting the zub SDO object dictionary. If internal SDO-based communication has to be blocked out of application reasons, it is possible to call up this function with parameter set to 'TRUE'. If internal SDO-based communication is blocked, the "old-fashioned" communication is used independent of control unit's version.

It is recommended to use the function 'ZbMocSDODefineUsage' instead of 'ZbMocSDOBlockUsage'. 'ZbMocSDODefineUsage' allows to define different levels of the usage of SDO-based communication.

If 'ZbMocSDOBlockUsage' is called up with parameter 'TRUE', the level of SDO usage is set to 'ZBMOC_SDO_NONE'. If 'ZbMocSDOBlockUsage' is called up with parameter 'FALSE', the level of SDO usage keeps unchanged, if it was already set to one of the “SDO in use” levels. Otherwise, if it was set to 'ZBMOC_SDO_NONE', it is set to 'ZBMOC_SDO_OPTIMISED'.

Portability

zbmoc.dll	Version 6.4.3 and higher
Control unit	Version 6.0 and higher

ZbMoc SDODefineUsage

(
UNSIGNED8 level
)

Summary Defines the internal usage of the SDO-based communication, which can be used for some of the ZBMoc functions in case of a control unit supporting the zub SDO object dictionary.

Parameter

level	ZBMOC_SDO_NONE (= 0): => SDOs are not used, independent of control unit's version ZBMOC_SDO_OPTIMISED (= 1): => SDOs are used only, if control unit is in execution state ZBMOC_SDO_ALWAYS (= 2): => SDOs are used always, if supported by the control unit

Return value

UNSIGNED8	Old setting of the SDO-based communication usage

Remarks A lot of ZbMoc functions offer two different ways of communication. The "older" communication path requires that the control unit is not in execution state while processing a ZbMoc request. This communication is compatible to all old and new control units. The other, so-called SDO-based communication path offers the possibility of command execution even while an APOSS program is running on the control unit. The 'ZbMocSDODefineUsage' function defines under which circumstances the SDO-based communication has to be used, if the control unit is capable of it.

SDO-based communication needs more small data packages for communication, this results in a bigger overhead for communication and remarkable less performance for communication lines with a low data thruput (e.g. RS232, DCOM-based VLT). If the level is set to ZBMOC_SDO_OPTIMISED (which is also the default setting), the SDO-based communication is only used in case of the control unit is in execution state and the "old" communication path would not allow command execution through this.

If the SDO-based communication is actually used, depends on different influences:
- SDO-based communication compatibility of the control unit
- Setting done by this command 'ZbMocSDODefineUsage'
- Execution status of the control unit (in case of ZBMOC_SDO_OPTIMISED setting)

Portability

zbmoc.dll	Version 6.4.4 and higher
Control unit	Version 6.0 and higher

ZbMoc SDOGetUsage

```
(
UNSIGNED16 h,
UNSIGNED16 id
)
```

Summary Returns, if the SDO-based communication is used internally for ZbMoc function processing.

Parameter

h	Handle of the interface
id	Identification number (ID) of the control unit

Return value

BOOLEAN	TRUE,	if internal SDO-based communication is in use
	FALSE,	if internal SDO-based communication is not used

Remarks If a 'FALSE' is returned, i.e. SDO-based communication for ZbMoc function processing is not in use, this can be due to a control unit not supporting the zub SDO object dictionary or due to blocking the SDO-based communication by a previous call-up of the function 'ZbMocSDODefineUsage' with parameter set to 'ZBMOC_SDO_NONE'.

More detailed information, if the SDO-based communication would currently be used and if the control unit is capable of it, can be retrieved by the function 'ZbMocSDOGetUsageDetails'.

Portability

zbmoc.dll	Version 6.4.3 and higher
Control unit	Version 6.0 and higher

ZbMoc SDOGetUsage Details

```
(
UNSIGNED16 h,
UNSIGNED16 id,
UNSIGNED8* level
BOOLEAN* sdosupport
)
```

Summary

Returns, if the SDO-based communication would be currently used (if possible) for ZbMoc function processing of the given control unit.

Parameter

h	Handle of the interface
id	Identification number (ID) of the control unit
level	Pointer to a variable taking the level of usage
sdosupport	Pointer to a variable taking the information, if the selected control unit supports SDO-based communication

Return value

BOOLEAN	TRUE	=> SDO-based comm. would be currently used (if possible)
	FALSE	=> SDO-based communication would not be used

Remarks

The return value depends on different influences:
- SDO-based communication compatibility of the control unit
- SDO usage level defined by the command 'ZbMocSDODefineUsage'
- Execution status of the control unit (in case of ZBMOC_SDO_OPTIMISED setting)

If SDO-based communication is actually used (or not) for processing of the next ZbMoc function call depends on the function call and the currently execution state of the control unit.

Portability

zbmoc.dll	Version 6.4.4 and higher
Control unit	Version 6.0 and higher

ZbMoc SetCallback GetStrings

(
void(*cb)(UNSIGNED16 sid,UNSIGNED8* s)
)

Summary Defines a callback pointer of a function, which delivers language dependent text strings.

Parameter

cb	Pointer to a function This function copies the requested text (identified by an ID) into the character array pointed to by 's'.

Return value –

Remarks This function registers a callback function for retrieving strings and formatting templates. The callback function gets the string number and returns the string into the specified buffer.

This function is intended for internal use only !

Portability

zbmoc.dll	Version 6.0.0 and higher
Control unit	Version 2.7 and higher

ZbMoc
SetCallback
Messageout

(
void(*cb)(UNSIGNED16 id,UNSIGNED8* s)
)

Summary Defines a callback pointer of a function, which delivers messages.

Parameter

cb	Pointer to a function This function copies the requested message (identified by an ID) into the character array pointed to by 's'.

Return value –

Remarks This function registers a callback function for printing messages. The ZbMoc library calls this callback function, to get messages printed which it has received from mocons.

This function is intended for internal use only !

Portability

zbmoc.dll	Version 6.0.0 and higher
Control unit	Version 2.7 and higher

ZbMoc SetDebugFile

```
(
const char* filename
)
```

Summary Defines the log file (complete name incl. path), in which debugging messages are written.

Parameter

filename	Pointer to a string holding the filename and path information

Return value –

Remarks The given string holding the filename and path information can be 128 characters long in maximum !

Portability

zbmoc.dll	Version 6.0.0 and higher
Control unit	Version 2.7 and higher

ZbMoc SetDebugLevel

(
UNSIGNED16 level
)

Summary Defines the debugging level.

Parameter

level	Debugging level (0 ... 9)
	0 = switch debug protocol off
	1..9 = the higher the value, the more detailed and the more extensive ist the protocol.

Return value _

Remarks The default value of the debug level is 0, i.e. no debugging protocol is generated.

If the debug level is set to a value higher than 0 (valid values are 0-9), a debugging protocol is written to the 'zbmoc.dm' file, which resides in the current directory. The debugging protocol is an ASCII text format file with time stamps for every entry. If debugging level is set to '1' for example, all calls of the ZBMoc functions are logged including the time stamp and the parameters of the function call.

The execution time of a program and library functions can be increased by setting the debug level to a value higher than 0.

Portability

zbmoc.dll	Version 6.0.0 and higher
Control unit	Version 2.7 and higher

ZbMoc SourceRestore FromController

```
(
UNSIGNED16 h,
UNSIGNED16 id,
UNSIGNED16 prognum,
const char* filename
ZbMocSourceInfoS* sourceinfo
)
```

Summary Reads out source code of the selected control unit.

Parameter

h	Handle of the interface
id	Identification number (ID) of the control unit
prognum	Number, which identifies the selected program in the control unit
filename	Pointer to a string holding the filename and path information of the file into which the restored source code has to be written
sourceinfo	Pointer to a structure taking the information (e.g. original file name, date and time of saving) about the read out source code

Return value

SIGNED16	Error code

Remarks

If the given file exists, it is overwritten !!

If the control unit supports the SDO-based communication and it is not blocked due to an application setting (see 'ZbMocSDODefineUsage'), the function can be executed independent of execution state. Otherwise (i.e. if SDO-based communication is not available) the execution state is checked by the function and a 'ZBMOC_E_IS_BUSY' error is returned, if the control unit is in execution state.

The source code has to be saved in the control unit by using the function 'ZbMocSourceSaveInController' sometimes before. It is not possible to restore source code out of a (binary) program that was just saved for execution.

There is no guarantee that the source code stored in the control unit corresponds to the binary program. Both types can be stored independent by using the 'ZbMocProgramSaveAs' and 'ZbMocSourceSaveInController' functions. There is no internal link in between the binary code and the source code. It is in the responsibility of the application program to make sure that both versions are consistent and stored by using the same number.

If the given file can not be accessed for writing or an error happens during any write operation, a 'ZBMOC_E_FILEACCESS' error is returned.
The source data is stored in a compressed data format in the control unit. If the decompression fails, a 'ZBMOC_E_COMPRESSION_ERROR' is returned.
Due to the fact that this command is a compound function with a lot of communication, any communication or memory allocation error can take place and be returned.

Portability

zbmoc.dll	Version 6.4.3 and higher
Control unit	Version 5.17 and higher Version 6.01 and higher: Usage of SDO-based communication => Function can be processed independent of control unit's execution state, e.g. while APOSS program executes

ZbMocSourceSaveInController

```
(
UNSIGNED16 h,
UNSIGNED16 id,
UNSIGNED16 prognum,
const char* filename
)
```

Summary

Writes the source code of the given file in a compressed data format into the selected control unit.

Parameter

h	Handle of the interface
id	Identification number (ID) of the control unit
prognum	Number, which is used for identification of the program in the control unit
filename	Pointer to a string holding the filename and path information of the source code file which has to be stored in the control unit

Return value

SIGNED16	Error code

Remarks

If the control unit supports the SDO-based communication and it is not blocked due to an application setting (see 'ZbMocSDODefineUsage'), the function can be executed independent of execution state. Otherwise (i.e. if SDO-based communication is not available) the execution state is checked by the function and a 'ZBMOC_E_IS_BUSY' error is returned, if the control unit is in execution state.

If the given file can not be accessed for reading or an error happens during the read operations, a 'ZBMOC_E_FILEACCESS' error is returned.
It is not possible to save empty files in the control unit. In this case the error code 'ZBMOC_E_FILECONTENT' is returned.
The file content is saved as compressed data in the controller. If the compression fails, a 'ZBMOC_E_COMPRESSION_ERROR' is returned.
Due to the fact that this command is a compound function with a lot of communication, any communication or memory allocation error can take place and be returned.

The given program number has to be equal to the number of the corresponding binary program, that was defined by using the 'ZbMocProgramSaveAs' function. The reason is that there is no internal link between the binary program code and the corresponding source code. So, the application is responsible to maintain a fixed method to manage corresponding program binary and source code. It is strongly recommended to do this by defining an identical number for the program binary code (as shown in the program list) and the program source code. Otherwise there is no compatibility with the application programs offered by zub machine control.

Portability

zbmoc.dll	Version 6.4.3 and higher
Control unit	Version 5.17 and higher Version 6.01 and higher: Usage of SDO-based communication => Function can be processed independent of control unit's execution state, e.g. while APOSS program executes

ZbMoc Step

```
(
UNSIGNED16 hdl,
UNSIGNED16 id,
UNSIGNED8 axnr,
SIGNED32 accel,
SIGNED32 velo,
SIGNED32 distance,
SIGNED32* newpos
)
```

Summary Moves the selected axis according to the given parameters.

Parameter

h	Handle of the interface
id	Identification number (ID) of the control unit
axnr	Axis number (1...n) of the control unit
accel	Acceleration
velo	Velocity
distance	Distance of the movement [user-defined units]
newpos	Pointer to a variable taking the new, up-to-date position value after the motion is finished

Return value

SIGNED16	Error code

Remarks The scaling of the parameters acceleration 'accel' and velocity 'velo' depends on the setting of the axis parameter no.22 'VELRES'. If the default setting (100) of this parameter is untouched, the parameters 'accel' and 'velo' represent a percentage value of the absolute maximum values also defined by the corresponding parameters.

Portability

zbmoc.dll	Version 6.0.0 and higher
Control unit	Version 2.7 and higher

ZbMoc Testrun

```
(
UNSIGNED16 hdl,
UNSIGNED16 id,
UNSIGNED8 axnr,
ZbMocTestrunParamS* para,
SIGNED32* messwerte
)
```

Summary Executes a test run of the selected control unit and axis number using the given parameters.

Parameter

hdl	Handle of the interface
id	Identification number (ID) of the control unit
axnr	Axis number (1...n) of the control unit
para	Pointer to a structure holding the parameters of the test run
messwerte	Pointer to an array taking the sampled measurements during the test run

Return value

SIGNED16	Error code

Remarks Commanded and actual position, velocity and motor current values are sampled and logged for evaluation of the control reaction.

The logged measurement values are written into the data section, to which the 'messwerte' pointer refers. It is must, that this data section is big enough to take all values. This is not checked by the function itself !

The motion and control parameters defined by the given structure are just valid for the test run. The current settings of the control unit keeps untouched. If the parameters should be set as the actual parameters of the control unit after a successful optimization, it is necessary to call up the function 'ZbMocTestrunParametersWrite' and set its parameter 'permanent' to TRUE.

Portability

zbmoc.dll	Version 6.0.0 and higher
Control unit	Version 2.7 and higher

ZbMoc Testrun ParametersRead

```
(
UNSIGNED16 hdl,
UNSIGNED16 id,
UNSIGNED8 axnr,
ZbMocTestrunParamS* para
)
```

Summary Reads out all parameters of the control unit, relevant for the execution of a test run.

Parameter

h	Handle of the interface
id	Identification number (ID) of the control unit
axnr	Axis number (1...n) of the control unit
para	Pointer to a structure taking the parameters

Return value

SIGNED16	Error code

Remarks This function retrieves all required current settings, which can be used to run a test run afterwards.

Portability

zbmoc.dll	Version 6.0.0 and higher
Control unit	Version 2.7 and higher

ZbMoc Testrun ParametersWrite

```
(
UNSIGNED16 hdl,
UNSIGNED16 id,
UNSIGNED8 axnr,
ZbMocTestrunParamS* para,
BOOLEAN permanent
)
```

Summary

Writes all parameters, which are relevant of a test run, into the parameters section of the control unit.

Parameter

hdl	Handle of the interface	
id	Identification number (ID) of the control unit	
axnr	Axis number (1...n) of the control unit	
para	Pointer to a structure taking the parameters	
permanent	TRUE:	parameters are copied into the control parameter section
	FALSE:	parameters are just stored temporarily until the next program executes

Return value

SIGNED16	Error code

Remarks

This function (with its parameter 'permanent' set to TRUE) is used to save optimized test run parameters, evaluated by using the 'ZbMocTestrun' function. The control parameters are copied into the general parameter section of the control unit. In case of a control unit using an internal software version > 5 (MACS), the parameters are even saved persistently. For all other versions it is necessary to call up the 'ZbMocMoconSaveRam' function afterwards for persistent storage.

Portability

zbmoc.dll	Version 6.0.0 and higher
Control unit	Version 2.7 and higher

ZbMoc Trace

```
(
UNSIGNED16 h,
UNSIGNED16 id,
UNSIGNED32 *line
)
```

Summary Executes a single line of APOSS program code, which was compiled using the debug option.

Parameter

h	Handle of the interface
id	Identification number (ID) of the control unit
line	Pointer to a variable taking the next source code line number after execution of this function. The returned line number is the line number, which is executed with the next call of the 'ZbMocTrace' function.

Return value

SIGNED16	Error code

Remarks In program execution debug mode this function has to be called up for each line of program code.

This function is intended for internal use only !

Portability

zbmoc.dll	Version 6.0.0 and higher
Control unit	Version 2.7 and higher

ZbMoc V24ReadRawChar

```
(
UNSIGNED16 h,
UNSIGNED16 id,
UNSIGNED8* rxd,
UNSIGNED32 timeout
)
```

Summary Reads a single character received via the serial interface.

Parameter

h	Handle of the interface
id	Identification number (ID) of the control unit
rxd	Pointer to a variable taking the received character
timeout	Maximum period of time [ms] for receiving a character 0 = Do not wait, i.e. a character has to present immediately If there is no character received within the defined time slot, the function returns an error code.

Return value

SIGNED16	Error code

Remarks This function is used for the implementation of applications, which receive byte or character based data of the control unit via the V24 interface.

This function is only available, if serial (V24) communication is used.

Portability

zbmoc.dll	Version 6.0.0 and higher
Control unit	Version 2.7 and higher

ZbMoc V24Read RawString

```
(
UNSIGNED16 h,
UNSIGNED16 id,
UNSIGNED8* rxbuf,
UNSIGNED16 rxbuflen,
UNSIGNED32 timeout
)
```

Summary Reads a string received via the serial interface.

Parameter

h	Handle of the interface
id	Identification number (ID) of the control unit
rxbuf	Pointer to a data section taking the received string
rxbuflen	Length of the data section
timeout	Maximum period of time [ms] for receiving a string 0 = Do not wait, i.e. a string has to present immediately If there is no (complete) string received within the defined time slot, the function returns an error code.

Return value

SIGNED16	Error code

Remarks This function is used for the implementation of applications, which receive string data of the control unit via the V24 interface.

According to the 'C' language convention the string is null terminated.

The string can be send by the control unit using the APOSS 'print' command. The string is terminated on the serial interface line with a <CR><LF>.

This function is only available, if serial (V24) communication is used.

Portability

zbmoc.dll	Version 6.0.0 and higher
Control unit	Version 2.7 and higher

ZbMoc V24WriteRaw

```
(
UNSIGNED16 h,
UNSIGNED16 id,
UNSIGNED8* buf,
UNSIGNED16 buflen
)
```

Summary Writes a string to the serial interface buffer for transmission.

Parameter

h	Handle of the interface
id	Identification number (ID) of the control unit
buf	Pointer to a data section holding the characters to transmit
buflen	Number of characters to transmit (Exactly this number of characters is transmitted.)

Return value

SIGNED16	Error code

Remarks This function is used for the implementation of applications, which want to transmit data, single characters or strings to the control unit (and/or APOSS program) via the V24 interface.

All the characters are transmitted just as provided by the given data array. Null characters are processed like any other characters.

This function is only available, if serial (V24) communication is used.

Portability

zbmoc.dll	Version 6.0.0 and higher
Control unit	Version 2.7 and higher

ZbMoc WriteArrayAll

```
(
UNSIGNED16 h,
UNSIGNED16 *arrnr,
UNSIGNED16 *arrsize,
UNSIGNED16 *rsize,
SIGNED32 *werte
)
```

Summary Writes an user-defined array into all connected control units.

Parameter

h	Handle of the interface
arrnr	Pointer to a variable holding the number of the array (0...n)
arrsize	Pointer to a variable holding the number of array values to write
rsize	Pointer to a variable taking the actual size of the array stored in the control unit (if already present)
werte	Pointer to an array holding the values to write

Return value SIGNED16 Error code

Remarks This function is not implemented up to now and returns always the error code 'ZBMOC_E_FUNCTION_UNAVAILABLE' !!!

Portability

zbmoc.dll	Version 6.0.0
Control unit	Version 2.7

ZbMoc WriteMemPage

```
(
UNSIGNED16 h,
UNSIGNED16 id,
UNSIGNED32 pageaddress,
UNSIGNED16 crc,
UNSIGNED8* values
)
```

Summary Writes the content of a memory page into the selected control unit.

Parameter

h	Handle of the interface
id	Identification number (ID) of the control unit
pageaddress	Number of the memory page
crc	CRC to transmit
values	Pointer to a data section holding the 256 bytes of data to transmit

Return value

SIGNED16	Error code

Remarks Each memory page has a size of 256 bytes.

This function is not implemented up to now and returns always the error code 'ZBMOC_E_FUNCTION_UNAVAILABLE' !!!

Portability

zbmoc.dll	Version 6.0.0 and higher
Control unit	MVS versions only

ZbMoc WriteUserArray

```
(
UNSIGNED16 h,
UNSIGNED16 id,
UNSIGNED16 *arrnr,
UNSIGNED16 *arrsize,
UNSIGNED16 *rsize,
SIGNED32 *values
)
```

Summary Writes an user-defined array into the selected control unit.

Parameter

h	Handle of the interface
id	Identification number (ID) of the control unit
arrnr	Pointer to a variable holding the number of the array (0...n)
arrsize	Pointer to a variable holding the number of array values to write
rsize	Pointer to a variable taking the actual size of the array stored in the control unit (if already present)
values	Pointer to an array holding the values to write

Return value SIGNED16 Error code

Remarks The complete given array is written into the control unit.

Arrays are numbered consecutive, starting with 0.

It is possible to write new arrays or overwrite existing arrays.
If a new (not existing) array is written, its number has to be consecutive to the last array number in use. !!! If enough free memory is available, the new array is allocated.
If an existing array is overwritten, its (old) array size has to be equal or bigger than the one given as part of the function call parameter. The actual size of the array which resides in the control unit is copied into the variable 'rsize'.

Portability

zbmoc.dll	Version 6.0.0 and higher Version 6.4.3 and higher: Usage of SDO-based communication
Control unit	Version 2.7 and higher Version 6.01 and higher: Usage of SDO-based communication => Function can be processed independent of control unit's execution state, e.g. while APOSS program executes

ZbMoc WriteUserVar

```
(
UNSIGNED16 h,
UNSIGNED16 id,
UNSIGNED16 varnr,
SIGNED32 val
)
```

Summary Transmits a data message to the selected control unit. This message can be received by an APOSS program using the 'INMSG' command.

The message contains always one 16 bit data value and a 32 bit data value.

Parameter

h	Handle of the interface
id	Identification number (ID) of the control unit
varnr	16 bit value to transmit
val	32 bit value to transmit

Return value

SIGNED16	Error code

Portability

zbmoc.dll	Version 6.0.0 and higher
Control unit	Version 2.7 and higher

ZbMoc WriteUsrVarAll

```
(
UNSIGNED16 h,
UNSIGNED16 varnr,
SIGNED32 val
)
```

Summary Transmits a data message to all connected control units. This message can be received by an APOSS program using the 'INMSG' command.

The message contains always one 16 bit data value and a 32 bit data value.

Parameter

h	Handle of the interface
varnr	16 bit value to transmit
val	32 bit value to transmit

Return value

SIGNED16	Error code

Remarks This function is not implemented up to now and returns always the error code 'ZBMOC_E_FUNCTION_UNAVAILABLE' !!!

Portability

zbmoc.dll	Version 6.0.0 and higher
Control unit	Version 2.7 and higher

Index

Appendix

A

Address 2
Anschrift 2
ASCII name
  Set 65
Autostart
  Clear 9
  Set 9, 21
Autotsart
  Clear 20

B

Break command
  Broadcast 31
  Continue 42
  Transmit 30

C

Callback Pointer
  Set 99, 100
CAN bus
  Centronics adapter 5
  GIN adapter 5
  Setting communication parameters 6
CAN Interface
  Open using Centronics adapter 75
  Open using GIN adapter 73
  Set parameters 68
CAN Message
  Read 35
  Transmit 36
CAN Open
  Read SDO of slave 32
  Start slave 33
  Write SDO of slave 34
Change default output port 52
Communication
  Read a character of the serial port 110
  Read a string of the serial port 111
  Write a string to the serial port 112
Configuration
  of the Control Unit 6
  Query using control unit ID 6
  Query using index 6
  Restore from file 9
  Save to file 9
Configuration file
  Restore from file 40
  Save to file 41
Control Unit
  Get Status 69
  Query information 61, 62, 63, 64
Copyright 2

D

Data Exchange with APOSS Programs 10
Debugging
  Insert line in log-file 43
  Set level 102
  Set log file 101
Drivers/DLLs
  Read identification 49
  Read version 47

E

EEPROM
  Clear 57
Error
  Clear 37, 53
  Read state 48, 58
Error Codes Overview 17
Error State
  Clear 8

F

Function Module
  APOSS program handling 8
  Close interface 6
  Configuration of the control unit 6
  Exchanging data with APOSS programs 10
  Global and. Axis Parameter Handling 7
  Open interface 5
Function Modules 5

H

How can the Library be used ? 4

I

Imprint 2
Interface
  Open 5
Interface
  Close 6, 38
  Close all 39
Interruption of a running program 8
Introduction 4

M

Memory Page
  Read 92
Message
  Read output of 'print' command 86
Motion
  Move stepwise 105
  Start 71
  Stop 72

P

Parameters
  Reset all 7
  Save 6
Parameters (axis)

Query number of 7
Query number of... 22
Read out 7, 23, 24
Write 7, 25, 26
Parameters (global)
Query number of 7
Query number of... 81
Read out 7, 82
Read out raw 83
Write 7, 84
Write raw 85
Program
Clear all 8, 56
Execute 8
Execute a line of code 109
Get list 87
List all 8
Load 8, 88, 89
Loading and Handling 8
Read execution state 60
Save 6
Start 44
Stop 8
Program (temporary)
Execute 8
Register in list 90
Register in list using number 91
Save using a name 8
Save using a name and number 8
Start 45
Program Source Code
Query info 27
Read out 28
Write 29
Project name
Set 65

Q

Query execution state 8

R

RAM
Save 66
Save Section 67
Read axis position 46
Read number of control units 50
Read plain state text 59
Read result of last function 51
Reset 6, 7, 54
Clear all parameters 55
Reset to factory settings 6

S

Saving of parameters and programs 6
SDO-based communication
Block usage 95
Define usage 96
Get usage 97
Get usage details 98
Source code
Read out of control unit 103
Save in control unit 104
Standardized Data Types 11
Structure
Axis parameters 14
Control unit information 11
Control unit status 12
Global parameters 12
Program information 12
Source code information 12
Source code information 16
Test run parameters 16
Structures Overview 11

T

Table of Contents 3
Testrun
Execute 106
Read parameters 107
Write parameters 108
Trademark 2

U

User Array
Read 93
Reading 10
Write 115
Write all 113
Writing 10
User Variable
Read 94
Reading 10
Transmitting 10
Write 116

V

V24 Interface 5
Open 77
VLT Interface
Open 79
VLT Serial Port 6

W

Wait for a string 70
What Services are provided ? 4

Z

ZbMoc
AutoClear 20
AutoSet 21
AxisParamCount 22
AxisParamRead 23
AxisParamReadRaw 24
AxisParamWrite 25
AxisParamWriteRaw 26
BlobInfoRequest 27
BlobRead 28
BlobWrite 29
Break 30
BreakAll 31
CanOpenSlaveReadSDO 32
CanOpenSlaveStart 33

CanOpenSlaveWriteSDO 34
CanReadRaw 35
CanWriteRaw 36
ClearError 37
Close 38
CloseAll 39
CNFRestoreFromFile 40
CNFSaveToFile 41
Continue 42
DebugPrintf 43
Exec 44
ExecTemp 45
GetAxisPosition 46
GetDllVersion 47
GetError 48
GetIFDriverID 49
GetNumber 50
GetTextResult 51
MoconChangeOutputPort 52
MoconClearError 53
MoconClearFactory 54
MoconClearParameters 55
MoconClearPrograms 56
MoconDeleteEeprom 57
MoconErrorGet 58
MoconErrorGetString 59
MoconExecutionStatus 60
MoconInfo 61
MoconInfoControlledRefresh 63
MoconInfoIdx 62
MoconInfoRefresh 64
MoconNameSet 65
MoconSaveRam 66
MoconSaveRamSection 67
MoconSetCanParameters 68
MoconStatusGet 69
MoconWaitForString 70
MoveStart 71
MoveStop 72
OpenCanGin 73
OpenCanLpt 75
OpenV24 77
OpenVLT 79
ParamCount 81
ParamRead 82
ParamReadRaw 83
ParamWrite 84
ParamWriteRaw 85
PollMessage 86
ProgramList 87
ProgramLoad 88
ProgramLoadAll 89
ProgramSave 90
ProgramSaveAs 91
ReadMemPage 92
ReadUserArray 93
ReadUserVar 94
SDOBlockUsage 95
SDODefineUsage 96
SDOGetUsage 97
SDOGetUsageDetails 98
SetCallbackGetStrings 99
SetCallbackMessageout 100
SetDebugFile 101
SetDebugLevel 102
SourceRestoreFromController 103
SourceSaveInController 104
Step 105
Testrun 106
TestrunParametersRead 107
TestrunParametersWrite 108
Trace 109
V24ReadRawChar 110
V24ReadRawString 111
V24WriteRaw 112
WriteArrayAll 113
WriteMemPage 114
WriteUserArray 115
WriteUserVar 116
WriteUsrVarAll 116