Programmers guide for PowerCube

Developing PC Programs for PowerCube Modules

Directory of Revisions			
Authorized by: Roland Tschakarow			
File name: Programmers guide for PowerCube.doc			
No.	Description	Revision	Date of change
001	1. Version	1.0	16.09.2002
002	Added SCHUNK FTS functions and renamed MP55 FS functions	1.1	10.01.2003
003	Revision	1.2	30.07.2004
004	Added new functions	1.3	15.07.2007

1 Developing PC Programs for PowerCube Modules

This document describes how to use PowerCube specific function calls with different compilers running on PCs. It shows the differences regarding compilers and operation systems. The document is useful for programmers developing applications by using Amtec's library of function calls for PowerCube.

2 System requirements

The function library for PowerCubes is available for different PC operation systems. It therefore has operation system specific features:

Operation sytem	Form	Supported Compilers
MS Windows 9x/NT/2000/XP	Dynamic Link Library (DLL)	Visual C/C++ Visual Basic National Instruments LabWindows CVI
SuSe Linux 6.4	Open Source	GNU C/C++
QNX 4.25	Open Source	Watcom C/C++ v. 10

The Windows DLL allows the use of a variety of compilers. This document includes only information on compilers Amtec has sucessfully tested with PowerCube.

3 Hardware requirements

The hardware requirements depend on the type of communication interface used. PowerCube modules provide 3 different types of interfaces:

PowerCube communication interface	Hardware requirements	Software requirements
CAN	CAN-Interface board installed	CAN-Interface driver installed
Profibus DP	DP-Interface board installed	DP-Interface driver installed
RS232	COM port available	COM-Port driver enabled (only Linux and QNX)

The Interface boards supported by Amtec are all listed in the document "First Steps with PowerCube". Additioonally you will find information on how to install and test the modules with the supplied demo software.

4 Visual C/C++ and PowerCube (Windows)

To use the M5APIW32.DLL with Visual C/C++ these files are required:

File	Description
M5APIW32.DLL	to be stored in the same directory as the exe file or in the path.
M5APIW32.H	Header file with function declarations and constants. To be used in the #include statement of all source modules using PowerCube functions. Store it in the project directory.
M5APIW32.LIB	Import library for Visual C/C++. Define as additional library module in the project settings.

The Header file M5APIW32.H holds all necessary function and constant declarations. The data types used are the standard C data types like „int“ and „float“.

This example shows how to open COM1 and address a module with ID 7 (RS232-Interface):

```
...
int ret = 0;
int dev = 0;
int modId = 7;
int numOfModules = 0;
```

```
float pos = 1.0;
float vel = 1.0;
float acc = 1.0;
char pInitString[] = "RS232:1,9600";
...

ret = PCube_openDevice( &dev, pInitString );
if( ret != 0 )
  // Error Handling ...

numOfModules = PCube_getModuleCount( dev );
printf( "Found %d PowerCubes\n", numOfModules );

ret = PCube_homeModule( dev, modId );
if( ret != 0 )
  // Error Handling ...

do
{ ret = PCube_getModuleState( dev, modId, &state );
  if( ret != 0 )
    // Error Handling ...
} while( !(state & STATEID_MOD_HOME) );

ret = PCube_moveRamp( dev, modId, pos, vel, acc );
if( ret != 0 )
  // Error Handling ...

ret = PCube_closeDevice( dev );
if( ret != 0 )
  // Error Handling ...
...
```

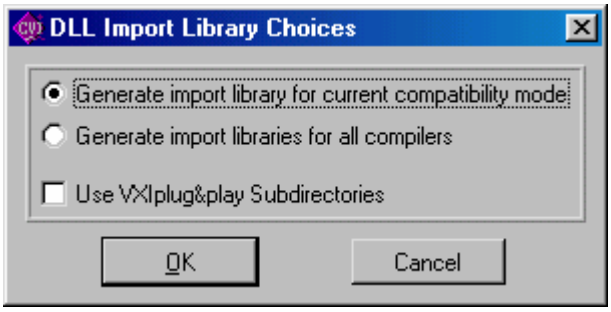
5 NI LabWindows CVI and PowerCube (Windows)

If you are planning to develop an application using LabWindows CVI, you need these files:

File	Description
M5APIW32.DLL	to be stored in the same directory as the exe file or in the path.
M5APIW32.H	Header file with function declarations and constants. To be used in the #include statement of all source modules using PowerCube functions. Store it in the project directory.

CVI requires an own Import library to enable the Link to M5APIW32.DLL. Please follow these steps:

1. Create a CVI project.
2. Open the file M5APIW32.H in the editor.
3. Create the CVI Import Library using „Options/Generate DLL Import Library"



4. Add the newly created file M5APIW32.LIB to your project: „Edit/Add files to Project..."

For a programming example please refer to the Visual C++ section.

6 Visual Basic and PowerCube (Windows)

These files are required to create a Visual Basic project for PowerCube:

File	Description
M5APIW32.DLL	to be stored in the same directory as the exe file or in the path.
M5APIW32.BAS	holds function and constant declarations. Use „Project/Add module...“ to add this file to your project.

This example shows how to open a device for controlling PowerCube modules using an InitString supplied from a Textbox named tInitString (the textbox is part of a VB form):

```
ret As Long
dev As Long
numOfModules As Long
modId As Long
state As Long
pos As Single
vel As Single
acc As Single

...
ret = PCube_openDevice( dev, tInitString.text )
If ret <> Then
  Rem Error handling ...
Else
  Rem Normal operation ...

numOfModules = PCube_getModuleCount( dev )

ret = PCube_homeModule( dev, modId );
If ret <> 0 Then
  Rem Error Handling ...

Do
  ret = PCube_getModuleState( dev, modId, state )
  If ret <> 0 Then
    Rem Error Handling ...
Loop While( state And STATEID_MOD_HOME <> 1 )

ret = PCube_moveRamp( dev, modId, pos, vel, acc )
If ret <> 0 Then
  Rem Error Handling ...
...

ret = PCube_closeDevice( dev )
If ret <> 0 Then
  Rem Error Handling ...
...
```

7 GNU C/C++ und PowerCube (Linux)

To work with PowerCube and Linux Amtec ships the complete source code for integration in your application program. This avoids any dependencies on Kernel versions or Linux distributions kits. For CAN bus users: Please make sure you have installed the CAN Interface driver suitable for the Linux Kernel version you are using. This driver has to be started before running a program with PowerCube. For RS232 users: Make sure your COM port is free and the driver is started as well.

The programmer can either chose a C++ class library interface or a standard ANSI-C interface (m5apiw32). There are sample programs for both variants.

These files are part of the project:

Directory	Description
c331	Linux driver for the CAN-Interface board C331 PCI by ESD. The driver has to be started before you begin to use PowerCube (Attention! Mind the Kernel version). Only root is allowed to start it: 1.) **su root** 2.) **cd c331** 3.) **insmod C331-2.2.14** After you finished work with PowerCube, remove the driver: 4.) **rmmod c331-2.2.14**
ComDef	Collection of globally used Header files
include	Collection of internally used Header files
lib	Collection of the library modules necessary to create the executable Device driver (M5 driver)
Util	Collection of additional C++ source modules for helper functions. Compile with: 1.) **cd Util** 2.) **make –f Makefile.linux**
Device	Collection of C++ source modules to create the executable Device driver (M5 driver). Compile with: 1.) **cd Device** 2.) **make –f Makefile.linux**
DeviceTest	C++ Test program using the Device class library. Compile with: 1.) **cd DeviceTest** 2.) **make –f Makefile.linux**
M5apiw32	ANSI-C Interface for the Device class library (like M5APIW32.DLL for Windows) . Compile with: 1.) **cd M5apiw32** 2.) **make –f Makefile.linux**
M5apiw32Test	ANSI-C Test program based on the M5apiw32 Interface (ANSI-C) . Compile with: 1.) **cd M5apiw32Test** 2.) **make –f Makefile.linux**
M5apitst2	Another ANSI-C Test program based on the M5apiw32 Interface (ANSI-C) . Compile with: 1.) **cd M5apitst2** 2.) **make –f Makefile.linux**

Make sure you call the compiler in the order given by the table above. For more programming examples please refer to the Visual C++ section.

8 Watcom C/C++ and PowerCube (QNX)

To work with PowerCube and QNX there are two interfaces available:
- C++ class library (Device).
- ANSI-C library (M5apiw32).

The library ships with the complete source code to simplify integration in your application programs.

These files are part of the project:

Directory	Description
c331	QNX driver for the CAN-Interface board C331 PCI by ESD. The driver has to be started before you begin to use PowerCube. Only root is allowed to start it: 1.) **su root** 2.) **cd c331** 3.) **c331** After you finished work with PowerCube, remove the driver by terminating the process (eg. Ctrl-C)
ComDef	Collection of globally used Header files
include	Collection of internally used Header files
lib	Collection of the library modules necessary to create the executable Device driver (M5 driver)
Util	Collection of additional C++ source modules for helper functions. Compile with: 1.) **cd Util** 2.) **make –f Makefile.qnx**
Device	Collection of C++ source modules to create the executable Device driver (M5 driver). Compile with: 1.) **cd Device** 2.) **make –f Makefile.qnx**
DeviceTest	C++ Test program using the Device class library. Compile with: 1.) **cd DeviceTest** 2.) **make –f Makefile.qnx**
M5apiw32	ANSI-C Interface for the Device class library (like M5APIW32.DLL for Windows) . Compile with: 1.) **cd M5apiw32** 2.) **make –f Makefile.qnx**
M5apiw32Test	ANSI-C Test program based on the M5apiw32 Interface (ANSI-C) . Compile with: 1.) **cd M5apiw32Test** 2.) **make –f Makefile.qnx**
M5apitst2	Another ANSI-C Test program based on the M5apiw32 Interface (ANSI-C) . Compile with: 1.) **cd M5apitst2** 2.) **make –f Makefile.qnx**

Make sure you call the compiler in the order given by the table above. For more programming examples please refer to the Visual C++ section.

9 Error Codes of Function Calls

Value	Define	Description
-201	ERRID_DEV_FUNCTIONNOTAVAILABLE	The function called is not available.
-202	ERRID_DEV_NOINITSTRING	The InitString is missing during initialization.
-203	ERRID_DEV_NODEVICENAME	The device name specified in InitString is wrong or invalid.
-204	ERRID_DEV_BADINITSTRING	The InitString is incomplete or wrong.
-205	ERRID_DEV_INITERROR	Initialization of the interface failed. Check hardware and driver setup.
-206	ERRID_DEV_NOTINITIALIZED	The function was called before initializing the device.
-207	ERRID_DEV_WRITEERROR	Error during an attempt to write data to the interface.
-208	ERRID_DEV_READERROR	Error during an attempt to read data from the interface.
-209	ERRID_DEV_WRITETIMEOUT	Timeout while sending data on the bus.
-210	ERRID_DEV_READTIMEOUT	Timeout while reading data from a module.
-211	ERRID_DEV_WRONGMESSAGEID	The message received has an unexpected MessageID.
-212	ERRID_DEV_WRONGCOMMANDID	The message received has an unexpected CommandID.
-213	ERRID_DEV_WRONGPARAMETERID	The message received has an unexpected ParameterID.
-214	ERRID_DEV_EXITERROR	Error occured while closing the interface.
-215	ERRID_DEV_NOMODULES	No module found during initialization of the interface.
-216	ERRID_DEV_WRONGDEVICEID	The given DeviceID is wrong.
-217	ERRID_DEV_NOLIBRARY	A DLL file is missing to execute the function call.
-218	ERRID_DEV_ISINITIALIZED	The Interface has been already initialized.
-219	ERRID_DEV_WRONGEMSMODULEID	The given EMS module ID does not exist.
-220	ERRID_DEV_EMSNOTINITIALIZED	The EMS module has not been initialized.
-221	ERRID_DEV_EMSMAXNUMBER	The maximum number of EMS modules has been reached.
-222	ERRID_DEV_EMSINITERROR	Error initializing an EMS module.
-223	ERRID_DEV_WRONGEMSTYPE	This function is intended to use with a different EMS module type.
-224	ERRID_DEV_WRONGEMSCHANNELID	The given channel ID of the EMS module does not exist.
-225	ERRID_DEV_WRONGMP55MODULEID	The given MP55 module ID does not exist.
-226	ERRID_DEV_WRONGSCHUNKMODULEID	The given SCHUNK module ID does not exist.

10 Observing the Module state

The module state is important for observation of all drive functions. This section describes the module state in detail.

10.1 State flags

The module state is a result of these function calls:
- PCube_getModuleState
- PCube_getStateDioPos
- PCube_moveRampExtended
- PCube_moveVelExtended
- PCube_moveCurExtended
- PCube_moveStepExtended
- PCube_movePosExtended.

The module state should be checked by the controlling program in every communication cycle.

Flag	Bit	Value	Meaning
STATE_HOME_OK	1	0x00000002	This flag is set after a successful homing procedure. It means that the drive has

Flag	Bit	Value	Meaning
			successfully found its zero position. All limitations for the operation range are valid now. If the user sends another Home-Command the flag will be reset until the homing procedure has been finished successfully.
STATE_HALTED	2	0x00000004	This flag is set in conjunction with an emergency stop. It means that the cube is in a secure state, not moving and not accepting motion commands. Only after a reset command which resets this flag, the module will return to the normal operation mode. An emergency stop can be caused automatically by the module in case of an error or by the user when sending a Halt command.
STATE_SWR	6	0x00000040	This flag shows the state of the home switch. Flag set means home switch is active, This is no error flag.
STATE_SW1	7	0x00000080	This flag shows the state of the Limit switch 1. Flag set means limit switch 1 is active. This is no error flag.
STATE_SW2	8	0x00000100	This flag shows the state of the Limit switch 2. Flag set means limit switch 2 is active. This is no error flag.
STATE_BRAKEACTIVE	9	0x00000200	This flag shows the state of the brake. Flag set means brake is active and servo loop is open. This is no error flag and it is used only if a brake is installed.
STATE_CURLIMIT	10	0x00000400	This flag is a warning of the servo loop. It has reached the maximum current output. The drive is working at its limits. This flag can be reset by the Reset command. It is no error flag
STATE_MOTION	11	0x00000800	This flag indicates the drive is in motion. It is set and reset automatically.
STATE_RAMP_ACC	12	0x00001000	This flag indicates the drive is in acceleration when controlled by ramp motion commands. It is automatically reset when the ramp motion profile has ended.
STATE_RAMP_STEADY	13	0x00002000	This flag indicates the drive is moving at constant speed when controlled by ramp motion commands. It is automatically reset when the ramp motion profile has ended.
STATE_RAMP_DEC	14	0x00004000	This flag indicates the drive is in deceleration when controlled by ramp motion commands. It is automatically reset when the ramp motion profile has ended.
STATE_RAMP_END	15	0x00008000	This flag indicates the end of a ramp motion profile. The drive is not moving.
STATE_INPROGRESS	16	0x00010000	This flag is only used in Step motion control. It indicates a Step motion command is in progress.
STATE_FULLBUFFER	17	0x00020000	This flag is only used in Step motion control. It indicates a Step motion command was pushed to the command stack. This happens when the module receives a Step motion command while STATE_INPROGRESS is set. Upon completion of the currently executed step command, the buffered one will automatically be executed.
STATE_ERROR	0	0x00000001	An error occured. The module stop immediately and does not accept motion commands anymore. The reason for the error state can be found reading the error flags. In many cases the error state can be reset by the user sending a Reset command. After a successful Reset the module is ready again to accept motion commands.
STATE_POWERFAULT	3	0x00000008	This flag defines an error of the servo amplifier. This flag si set in conjuction with STATE_ERROR. In most cases the module needs to be switched off to reset this error. One of the flags 18 through 23 will be set to explain the cause.
STATE_TOW_ERROR	4	0x00000010	Tow error: The servo loop was not able to follow the target position within the given limit. The maximum tow can be adjsuted using the parameter „MaxDeltaPos". Check if the module was overloaded.
STATE_COMM_ERROR	5	0x00000020	This error flag is raised if the watchdog has been enabled only. When enabled the watchdog must be refreshed in a given period of time by the external control. If the external control fails to do so, the drive will follow the emergency Stopp routine and enter an error state.
STATE_POW_VOLT_ERR	18	0x00040000	This flag is set in conjunction with STATE_POWERFAULT. It indicates a voltage drop or an overvoltage occurred in the motor supply. This error can be reset after the normal voltage level has been restored. Check your power supply.
STATE_POW_FET_TEMP	19	0x00080000	This flag is set in conjunction with STATE_POWERFAULT. The power transistors have overheated and the servo loop has been disabled. Power must be switched of to reset this error. It is due to overload or too high ambient temperature.
STATE_POW_INTEGRAL-ERR	23	0x00800000	This flag is set in conjunction with STATE_POWERFAULT. The drive has been overloaded and the servo loop has been disabled. Power must be switched off to reset this error. Check your apllication and the load situations of the drive.
STATE_BEYOND_HARD	25	0x02000000	This flag indicates the module has reached the hard limit. An emergency stop has been executed automatically. To remove the module from this position you need to follow the procedure described in "PowerCube™ Operation System: Disorder".
STATE_BEYOND_SOFT	26	0x04000000	This flag indicates the module has reached the soft limit. An emergency stop has been executed automatically. This flag can be reset by a Reset command.
STATE_LOGIC_VOLT	27	0x08000000	The voltage of the logic power supply has either dropped or an overvoltage occured. The drive will be disabled. This error can be reset.
STATE_POW_WDG_TEMP	20	0x00100000	This flag is set in conjunction with STATE_POWERFAULT. The motor has overhea-

Flag	Bit	Value	Meaning
			ted and the servo loop has been disabled. Power must be switched of to reset this error. It is due to overload or too high ambient temperature.
STATE_POW_SHORTCUR	21	0x00200000	This flag is set in conjunction with STATE_POWERFAULT. A short curcuit occured. The servo loop has been disabled. The power must be switched of to reset this error. The module has been overlaoded. If this error cannot be reset consult your service partner.
STATE_POW_HALLERR	22	0x00400000	This flag is set in conjunction with STATE_POWERFAULT. An error occured in reading the hall effect sensors of the motor. The motor has been overheated. Power must be switched off to reset this error.
STATE_CPU_OVERLOAD	24	0x01000000	Communication breakdown between CPU and current controller. Power must be switched off. Please consult your service partner.
STATE_POW_SETUP_ERR	27	0x08000000	Error in initializing the current controller. Module settings disaccord with controller configuration (5A/10A types). Power must be switched off. Please consult your service partner. Available from version 3.5.14 through 3.5.1D.

[These flags describe an error status]
[These flags provide useful information on the module status]
[These flags are obsolete]

10.2 Digital In-/Output state (IO-state)

The IO-state is the result of these function calls:
- PCube_getDefDioData
- PCube_getDioData

The data type returned is a "long" (4 Byte).

- PCube_getStateDioPos
- PCube_moveRampExtended
- PCube_moveVelExtended
- PCube_moveCurExtended
- PCube_moveStepExtended
- PCube_movePosExtended.

These function calls return data of type "long" (4 Byte) too. Only the state of the Home and Limit switches is not included in this word.

Value	Define	Description
0x00000001L	DIOID_MOD_INBIT0	State of Input bit 0.
0x00000002L	DIOID_MOD_INBIT1	State of Input bit 1.
0x00000004L	DIOID_MOD_INBIT2	State of Input bit 2.
0x00000008L	DIOID_MOD_INBIT3	State of Input bit 3.
0x00000010L	DIOID_MOD_OUTBIT0	State of Output bit 0.
0x00000020L	DIOID_MOD_OUTBIT1	State of Output bit 1.
0x00000040L	DIOID_MOD_OUTBIT2	State of Output bit 2.
0x00000080L	DIOID_MOD_OUTBIT3	State of Output bit 3.
0x00000100L	DIOID_MOD_INSWR	State of Home switch. Valid only for PCube_getDioData.
0x00000200L	DIOID_MOD_INSW1	State of Limit switch 1. Valid only for PCube_getDioData.
0x00000400L	DIOID_MOD_INSW2	State of Limit switch 2. Valid only for PCube_getDioData.

11 Module configuration

The drive can be configured by the user after Power On. This concerns range of operation, speed and acceleration as well as error behaviour.

11.1 Configuration word

The configuration word is a result of the function calls PCube_getDefConfig and PCube_getConfig. It is a parameter of the call PCube_setConfig.

Value	Define	Description
0x00000008L	CONFIGID_MOD_BRAKE_PRESENT	1 = Brake present
0x00000010L	CONFIGID_MOD_BRAKE_AT_POWERON	0 = Brake will be released at Power On
0x00000020L	CONFIGID_MOD_SWR_WITH_ENCODERZERO	1 = Encoder Index used for Homing
0x00000040L	CONFIGID_MOD_SWR_AT_FALLING_EDGE	1 = Homing finishes on falling edge of homing switch
0x00000080L	CONFIGID_MOD_CHANGE_SWR_TO_LIMIT	1 = Homing switch converts to limit switch after Homing is finished
0x00000100L	CONFIGID_MOD_SWR_ENABLED	1 = Homing switch is enabled
0x00000200L	CONFIGID_MOD_SWR_LOW_ACTIVE	1 = Homing switch is low active
0x00000400L	CONFIGID_MOD_SWR_USE_EXTERNAL	1 = The external homing switch will be used
0x00000800L	CONFIGID_MOD_SW1_ENABLED	1 = Limit switch 1 is enabled
0x00001000L	CONFIGID_MOD_SW1_LOW_ACTIVE	1 = Limit switch 1 is low active
0x00002000L	CONFIGID_MOD_SW1_USE_EXTERNAL	1 = The external limit switch 1 will be used
0x00004000L	CONFIGID_MOD_SW2_ENABLED	1 = Limit switch 2 is enabled
0x00008000L	CONFIGID_MOD_SW2_LOW_ACTIVE	1 = Limit switch 2 is low active
0x00010000L	CONFIGID_MOD_SW2_USE_EXTERNAL	1 = The external limit switch 2 will be used
0x00020000L	CONFIGID_MOD_LINEAR	1 = Module is of linear type
0x00080000L	CONFIGID_MOD_ALLOW_FULL_CUR	0 = The max. cur commanded with PCube_moveCur will be limited to the nominal current.
0x00100000L	CONFIGID_MOD_M3_COMPATIBLE	1 = Module is MoRSE3 compatible. This concerns CAN communication and behaviour of PCube_moveStep. The module does not accept motion commands unless Homing is finished successfully.
0x00200000L	CONFIGID_MOD_LINEAR_SCREW	1 = Module is linear module with ball screw actuator.
0x00800000L	CONFIGID_MOD_DISABLE_ON_HALT	1 = On error the motor is set to zero current.
0x01000000L	CONFIGID_MOD_WATCHDOG_ENABLE	1 = Watchdog is enabled. The watchdog starts after reception of the first life sign from control (PCube_serveWatchdogAll).
0x02000000L	CONFIGID_MOD_ZERO_MOVE_AFTER_HOK	1 = After Homing is finished the module automatically moves to ist zero position
0x04000000L	CONFIGID_MOD_DISABLE_ACK	1 = Messages are not acknowledged anymore. Get commands will still be answered. Valid only for CAN-Bus.
0x08000000L	CONFIGID_MOD_SYNC_MOTION	1 = Enables synchronized Motion commands. After sending the motion command the a special Start Motion broadcast is expected (PCube_startMotionAll). Valid only for CAN-Bus.

11.2 Setup word

The Setup word is a result of the function call PCube_getDefSetup.

Value	Define	Description
0x00000001L	SETUPID_MOD_ENCODER_FEEDBACK	not used
0x00000002L	SETUPID_MOD_RESOLVER_FEEDBACK	not used
0x00000004L	SETUPID_MOD_ABSOLUTE_FEEDBACK	not used
0x00000008L	SETUPID_MOD_4IN_4OUT	1 = The 15pole connector is configured for 4 I/O signals.
0x00000010L	SETUPID_MOD_3IN_ENCODER_IN	1 = The 15pole connector is configured for encoder input.
0x00000020L	SETUPID_MOD_3IN_ENCODER_OUT	1 = The 15pole connector is configured for encoder output.
0x00000040L	SETUPID_MOD_RS232	1 = The module is configured for RS232-communication.
0x00000200L	SETUPID_MOD_CAN	1 = The module is configured for CAN-communication.
0x00000400L	SETUPID_MOD_PROFIBUS	1 = The module is configured for Profibus-communication
0x00000800L	SETUPID_MOD_USE_M3ID	1 = CAN identifiers for MoRSE3 modules are activated.
0x00001000L	SETUPID_MOD_USE_M4ID	1 = CAN identifiers for MoRSE4 modules are activated.
0x00002000L	SETUPID_MOD_USE_CANOPEN	1 = The module is configured for CANopen.
0x00008000L	SETUPID_MOD_USE_SW2_AS_ENABLE	1 = The input for limit switch 2 is used as enable signal for the drive.
0x00010000L	SETUPID_MOD_USE_SW2_AS_BRAKE	1 = The input for limit switch 2 is used to release the brake.

Value	Define	Description
0x00020000L	SETUPID_MOD_ERROR_TO_OUT0	1 = An error is signalized on output 0.

12 System Configuration using the Ini file

The function call PCube_configFromFile enables the user to configure the complete system just by using one function call. This concerns in detail:

- Opening the communication interface
- Setting debug and logging options
- Configuration of module parameters:
  - Home Offset (to be specified in [m] resp. [rad])
  - Home Geschwindigkeit (to be specified in [m/s] resp. [rad/s])
  - PID loop coefficient C0
  - PID loop coefficient Damp
  - PID loop coefficient A0
  - minimum Position (to be specified in [m] resp. [rad])
  - maximum Position (to be specified in [m] resp. [rad])
  - maximum tow distance (to be specified in [m] resp. [rad])
  - maximum Speed (to be specified in [m/s] resp. [rad/s])
  - maximale Acceleration (to be specified in [m/s²] resp. [rad/s²])
  - maximum current (to be specified in [A])

This is an example of an Ini file:

Section	Entries	Description
[PROCESS]	DeviceNumber = 1	Number of the device (interfaces))) to be opened
	DeviceStart = 0	ID of the first device to be opened (Offset)
	ModuleNumber = 1	Number of PowerCube modules to configure
	ModuleStart = 0	ID of the first PowerCube module to be configured (Offset)
	Debug = 1	enables Error logging
	DebugLevel = 0	specifies the debug level
	DebugFile = 0	enables logging to a file
[DEVICE_00]	DeviceName = CAN0	A name identifiing the Device (CAN interface)
	InitString = ESD:0,250	InitString of the ESD CAN interface (Name:port,baudrate)
[MODULE_00]	DeviceName = CAN0	Name of the device to use
	ModuleId = 12	physical address of the PowerCube module to configure
	HomeOffset = 0.1	Home Offset of the PowerCube Module in [m] resp. [rad]
	HomeVel = 0.1	Home velocity of the PowerCube Module in [m/s] resp. [rad/s]
	C0 = 32	PID loop coefficient C0 of the PowerCube Module
	Damp = 3	PID loop coefficient Damp of the PowerCube Module
	A0 = 1	PID loop coefficient A0 of the PowerCube Module
	MinPos = -1.0	minimum Position of the PowerCube Module in [m] resp. [rad]
	MaxPos = 1.0	maximum Position of the PowerCube Module in [m] resp. [rad]
	MaxDeltaPos = 0.01	maximum tow distance of the PowerCube Module in [m] resp. [rad]
	MaxVel = 1	maximum speed of the PowerCube Module in [m/s] resp. [rad/s]
	MaxAcc = 1	maximum acceleration of the PowerCube Module in [m/s^2] resp. [rad/s^2]
	MaxCur = 5	maximum current of the PowerCube Module in [A]

If only one of the settings specified in the Ini file fails, the function will stop immediatly and return an error.

13 Function reference

This reference includes all function calls available in the M5APIW32 application programming interface.

Opening and closing the communication Interface	Function	Description

Opening and closing the communication Interface	Function	Description
	PCube_openDevice	Opens the interface by specifiing an InitString. Result is a valid deviceID. See document "First steps" for further information on the InitString.
	PCube_closeDevice	Closes the interface by specifiing the deviceID.

Administrative functions	Function	Description
	PCube_getModuleIdMap	Retrieves the number of PowerCube modules found on the bus. At the same time this function maps the physical addresses of the modules to logical IDs. The physical addresses are strored in an ascending order in the array specified.
	PCube_updateModuleIdMap	Redoes the mapping.
	PCube_getModuleCount	Retrieves the number of modules connected to the bus.
	PCube_getModuleType	Retrieves the module type by specifiing deviceID and a moduleID. Rotary drives: TYPEID_MOD_ROTARY = 0x0F Linear drives: TYPEID_MOD_LINEAR = 0xF0
	PCube_getModuleVersion	Retrieves the version of the operating system of the module by specifiing deviceID and a moduleID. The result has to be interpreted as a hexadecimal number.
	PCube_getModuleSerialNo	Retrieves the serial number of a module by specifiing deviceID and a moduleID.
	PCube_getDllVersion	Retrieves the version of the running DLL.
	PCube_configFromFile	Configures the complete system to the specifications in a Ini file. The file name is a parameter for the call. See section "System configuration using an Ini file".
	PCube_serveWatchdogAll	Refreshs the watchdog in all conected modules if these are enabled. Valid only for CAN bus.
	PCube_getDefSetup	Retrieves the default module setup by specifiing deviceID and a moduleID. Result is a setup word, separatly described in the section "Module configuration“.
	PCube_getDefBaudRate	Retrieves the default Baudrate of the module (useful only for CAN- and RS232 modules). Result is a value between 0..5. CAN: 0=50 1=250 2=500 4=1000 kbit/s RS232: 0=1200 1=2400 2=4800 3=9600 4=19200 5=38400 bit/s
	PCube_setBaudRateAll	Broadcast command to change the baudrate of all connected modules (only CAN bus). All modules conected immediately change their baudrate to the new value. Only a valid baudrate will be accepted. Valid only for CAN-Bus.
	PCube_getDefGearRatio	Retrieves the default gear ratio.
	PCube_getDefLinearRatio	Retrieves the default factor for conversion of rotary to linear motion.
	PCube_getDefCurRatio	Retrieves the default factor for conversion of current digits to A.
	PCube_getDefBrakeTimeOut	Retrieves the default delay between end of motion and release of the brake.
	PCube_getDefIncPerTurn	Retrieves the default value for the number of increments per motor rotation.

Retrieve position	Function	Description
	PCube_getPos	Retrieves the actual module position by specifiing deviceID and a moduleID. Result is the position in rad (rotary) or m (linear modules).
	PCube_getPosInc	Retrieves the actual module position by specifiing deviceID and a moduleID. Result is the position in increments.
	PCube_getPosCountInc	Retrieves the current counter value by specifiing deviceID and a moduleID. Result is the current counter value in increments (position without any offsets).

Retrieve speed	Function	Description
	PCube_getVel	Retrieves the current speed by specifiing deviceID and moduleID. Result is the real speed in rad/s (rotary) or m/s (linear modules).
	PCube_getVelInc	Retrieves the current speed by specifiing deviceID and moduleID. Result is the real speed in increments/s.
	PCube_getIPolVel	Retrieves the current interpolated speed by specifiing deviceID and moduleID. Result is the interpolated speed in rad/s (rotary) or m/s (linear modules).

Retrieve current	Function	Description
	PCube_getCur	Retrieves the actual current information by specifiing deviceID an a moduleID. Result is the actual current in A.
	PCube_getCurInc	Retrieves the actual current information by specifiing deviceID an a moduleID. Result is the actual current in Digits.

Retrieve tow distance	Function	Description
	PCube_getDeltaPos	Retrieves the actual tow distance by specifiing deviceID and a moduleID. Result is the actual tow distance in rad (rotary) or m (linear modules).
	PCube_getDeltaPosInc	Retrieves the actual tow distance by specifiing deviceID and a moduleID. Result is the actual tow distance in increments.

Retrieve module state	Function	Description
	PCube_getModuleState	Retrieves the actual module state by specifiing deviceID and a moduleID. Result is the module status word, separately described in section „Module state“.
	PCube_getStateDioPos	Retrieves a combined information on module state, position and digital IO state. Result is the module state (section Module state), the actual position in rad resp. m and the state of the digital I/Os.

Retrieve position synchronously	Function	Description
	PCube_savePosAll	This broadcast command forces all connected modules to save their current position at the same time. The deviceID is a necessary parameter. For CAN-Bus only.
	PCube_getSavePos	Retrieves the position value saved during the call PCube_savePosAll by specifiing deviceID and a moduleID. Result is the saved with PCube_savePosAll position in rad resp. m. For CAN-Bus only.

Configuration	Function	Description
	PCube_getDefConfig	Retrieves the default module configuration by specifiing deviceID and a moduleID. Result is the configuration word, separately described in section „Module configuration“.
	PCube_getConfig	Retrieves the actual module configuration by specifiing deviceID and a moduleID. Result is the configuration word as saved to the module with the last call of PCube_setConfig. After Power on this value is identical to the default value.
	PCube_setConfig	Sets the actual module configuration by specifiing deviceID, a moduleID and a new configuration word.

Digitale I/O	Function	Description
	PCube_getDefDioData	Retrieves the default state of the digital IOs by specifiing deviceID and a moduleID. Result is the IO state described in section "Module state".
	PCube_getDioData	Retrieves the actual state of the digital IOs by specifiing deviceID and a moduleID. Result is the actual IO state
	PCube_setDioData	Sets the actual IO state (only outputs) by specifiing deviceID, a moduleID and a valid IO state word.

PID loop coefficients	Function	Description
	PCube_getDefA0	Retrieves the default value of the PID loop coefficient A0.
	PCube_getA0	Retrieves the actual value of the PID loop coefficient A0. After Power on this value is identical to the default.
	PCube_setA0	Sets the actual value of the PID loop coefficient A0 (range 1..12)
	PCube_getDefC0	Retrieves the default value of the PID loop coefficient C0.
	PCube_getC0	Retrieves the actual value of the PID loop coefficient C0. After Power on this value is identical to the default.
	PCube_setC0	Sets the actual value of the PID loop coefficient C0 (range 12..64, even values only)
	PCube_getDefDamp	Retrieves the default value of the PID loop coefficient "Damping".
	PCube_getDamp	Retrieves the actual value of the PID loop coefficient "Damping". After Power on this value is identisch to the default.
	PCube_setDamp	Sets the actual value of the PID loop coefficient "Damping" (range 1..4)

PID loop coefficients	Function	Description
	PCube_getDefA0	Retrieves the default value of the PID loop coefficient A0.
	PCube_recalcPIDParams	Call to update the PID loop and to make the new coefficients valid. This function must be called after A0, C0 or Damp have been altered.

Position offset	Function	Description
	PCube_getDefHomeOffset	Retrieves the default Home offset by specifiing deviceID and a moduleID. Result is the default home offset in rad resp. m. The home offset is the position value in home position.
	PCube_getHomeOffset	Retrieves the actual Home offset by specifiing deviceID and a moduleID. Result is the actual home offset in rad resp. m. After Power on this value is identical to the default.
	PCube_getHomeOffsetInc	Retrieves the actual Home offset by specifiing deviceID and a moduleID. Result is the actual home offset in Increments.
	PCube_setHomeOffset	Sets the actual Home offset by specifiing deviceID, a moduleID and the new value in rad resp. m.
	PCube_setHomeOffsetInc	Sets the actual Home offset by specifiing deviceID, a moduleID and the new value in Increments.

Homing speed	Function	Description
	PCube_getDefHomeVel	Retrieves the default Homing speed by specifiing deviceID and a moduleID. Result is the default Homing speed in rad/s resp. m/s. This speed is used during the homing procedure.
	PCube_getHomeVel	Retrieves the actual Homing speed by specifiing deviceID and a moduleID. Result is the Homing speed in rad/s resp. m/s. After Power on this value is identical to the default.
	PCube_getHomeVelInc	Retrieves the actual Homing speed by specifiing deviceID and a moduleID. Result is the Homing speed in Increments/s.
	PCube_setHomeVel	Sets the actual Homing speed by specifiing deviceID, a moduleID and the new value in rad/s resp. m/s.
	PCube_setHomeVelInc	Sets the actual Homing speed by specifiing deviceID, a moduleID and the new value in Increments/s.

Operation range: Minimum position	Function	Description
	PCube_getDefMinPos	Retrieves the default minimum position by specifiing deviceID and a moduleID. Result is the minimum position in rad resp. m. This parameter is used as a limit for the operation range. Values less than this will be limited to the given minimum.
	PCube_getMinPos	Retrieves the actual minimum position by specifiing deviceID and a moduleID. Result is the minimum position in rad resp. m. After Power on this value is identical to the default.
	PCube_getMinPosInc	Retrieves the actual minimum position by specifiing deviceID and a moduleID. Result is the minimum position in increments.
	PCube_setMinPos	Sets the actual minimum position by specifiing deviceID, a moduleID and the new value in rad resp. m.
	PCube_setMinPosInc	Sets the actual minimum position by specifiing deviceID, a moduleID and the new value in Increments.

Operation range: Maximum position	Function	Description
	PCube_getDefMaxPos	Retrieves the default maximum position by specifiing deviceID and a moduleID. Result is the maximum position in rad resp. m. This parameter is used as a limit for the operation range. Values greater than this will be limited to the given maximum.
	PCube_getMaxPos	Retrieves the actual maximum position by specifiing deviceID and a moduleID. Result is the maximum position in rad resp. m. After Power on this value is identical to the default.
	PCube_getMaxPosInc	Retrieves the actual maximum position by specifiing deviceID and a moduleID. Result is the maximum position in increments.
	PCube_setMaxPos	Sets the actual maximum position by specifiing deviceID, a moduleID and the new value in rad resp. m.
	PCube_setMaxPosInc	Sets the actual maximum position by specifiing deviceID, a moduleID and the new value in Increments.

Maximum speed	Function	Description
	PCube_getDefMaxVel	Retrieves the default maximum speed by speciffiing deviceID and a moduleID. Result is the maximum speed in rad/s resp. m/s. This parameter is used as a limit. Values greater than this will be limited to the maximum.
	PCube_getMaxVel	Retrieves the actual maximum speed by specifiing deviceID and a moduleID. Result is the maximum speed in rad/s resp. m/s. After Power on this value is identical to the default.
	PCube_getMaxVelInc	Retrieves the actual maximum speed by specifiing deviceID and a moduleID. Result is the maximum speed in Increments/s.
	PCube_setMaxVel	Sets the maximum speed by specifiing deviceID, a moduleID and the new value in rad/s resp. m/s.
	PCube_setMaxVelInc	Sets the maximum speed by specifiing deviceID, a moduleID and the new value in Increments/s.

Maximum acceleration	Function	Description
	PCube_getDefMaxAcc	Retrieves the default maximum acceleration by speciffiing deviceID and a moduleID. Result is the maximum acceleration in rad/s² resp. m/s². This parameter is used as a limit. Values greater than this will be limited to the maximum.
	PCube_getMaxAcc	Retrieves the actual maximum acceleration by specifiing deviceID and a moduleID. Result is the maximum acceleration in rad/s² resp. m/s². After Power on this value is identical to the default.
	PCube_getMaxAccInc	Retrieves the actual maximum acceleration by specifiing deviceID and a moduleID. Result is the maximum acceleration in Increments/s².
	PCube_setMaxAcc	Sets the maximum speed by specifiing deviceID, a moduleID and the new value in rad/s² resp. m/s².
	PCube_setMaxAccInc	Sets the maximum speed by specifiing deviceID, a moduleID and the new value in Increments/s².

Maximum current	Function	Description
	PCube_getDefMaxCur	Retrieves the default maximum current by specifiing deviceID and a moduleID. Result is the maximum current in A. This value is a limit for the maximum motor current used during operation.
	PCube_getMaxCur	Retrieves the actual maximum current by specifiing deviceID and a moduleID. Result is the maximum current in A. After Power on this value is identical to the default.
	PCube_setMaxCur	Sets the actual maximum current by specifiing deviceID, a moduleID the new maximum in A.

Maximum tow distance	Function	Description
	PCube_getDefMaxDeltaPos	Retrieves the default maximum tow distance by specifiing deviceID and a moduleID. Result is the default maximum tow distance in rad resp. m. This is a limiting value for the maximum tow distance allowed. If the drive overshoots this value during operation an error will be generated and the motor stops.
	PCube_getMaxDeltaPos	Retrieves the actual maximum tow distance by specifiing deviceID and a moduleID. Result is the actual maximum tow distance in rad resp. m. After Power on this value is identical to the default.
	PCube_getMaxDeltaPosInc	Retrieves the actual maximum tow distance by specifiing deviceID and a moduleID. Result is the actual maximum tow distance in increments.
	PCube_setMaxDeltaPos	Sets the actual maximum tow distance by specifiing deviceID, a moduleID and the new maximum in rad resp. m.
	PCube_setMaxDeltaPosInc	Sets the actual maximum tow distance by specifiing deviceID, a moduleID and the new maximum in Increments.

Target ramp motion speed	Function	Description
	PCube_setRampVel	Sets the target speed for a ramp motion profile by specifiing deviceID, a moduleID and the new value in rad/s resp. m/s. This value will be used for all ramp motion commands started with PCube_movePos, PCube_movePosInc or PCube_movePosExtended. In order to do so a target acceleration greater than zero must have been set using PCube_setRampAcc or PCube_setRampAccInc.

Target ramp motion speed	Function	Description
	PCube_setRampVel	Sets the target speed for a ramp motion profile by specifiing deviceID, a moduleID and the new value in rad/s resp. m/s. This value will be used for all ramp motion commands started with PCube_movePos, PCube_movePosInc or PCube_movePosExtended. In order to do so a target acceleration greater than zero must have been set using PCube_setRampAcc or PCube_setRampAccInc.
	PCube_setRampVelInc	Sets the target speed for ramp motion profiles by specifiing deviceID, a moduleID and the new value in Increments/s.

Target ramp motion acceleration	Function	Description
	PCube_setRampAcc	Sets the target acceleration for a ramp motion profile by specifiing deviceID, a moduleID and the new value in rad/s² resp. m/s². This value will be used for all ramp motion commands started with PCube_movePos, PCube_movePosInc or PCube_movePosExtended. In order to do so a target speed greater than zero must have been set using PCube_setRampVel oder PCube_setRampVelInc.
	PCube_setRampAccInc	Sets the target acceleration for ramp motion profiles by specifiing deviceID, a moduleID and the new value in Increments/s².

Homing	Function	Description
	PCube_homeModule	Starts a Homing procedure of the module specified by deviceID and moduleID.
	PCube_homeAll	Starts a Homing procedure of all modules connected to the bus. For CAN-Bus only.

Quick stop	Function	Description
	PCube_haltModule	Issues a Quick stop of the module specified by deviceID and moduleID.
	PCube_haltAll	Issues a Quick stop of all modules connected to the bus. For CAN-Bus only.

Softstop	Funktion	Bemerkung
	PCube_softStopModule	Issues a Soft stop of the module specified by deviceID and moduleID.
	PCube_softStopAll	Issues a Soft stop of all modules connected to the bus. For CAN-Bus only.

Reset of module state	Function	Description
	PCube_resetModule	Issues a Reset of the module specified by deviceID and moduleID. A Reset can clear error flags in the module state. If an error is permanent, Reset is ignored.
	PCube_resetAll	Issues a Reset of all modules connected to the bus. For CAN-Bus only.

Ramp motion with specification of Target position	Function	Description
	PCube_movePos	Starts a ramp motion profile of the module specified by deviceID and moduleID. The target position is given in rad resp. m. Prior to this call target speed and acceleration must be set using Funktionen PCube_setRampVel and PCube_setRampAcc resp. PCube_setRampVelInc and PCube_setRampAccInc.
	PCube_movePosInc	Starts a ramp motion profile of the module specified by deviceID and moduleID. The target position is given in increments.
	PCube_movePosExtended	Starts a ramp motion profile of the module specified by deviceID and moduleID. The target position is given in rad resp. m. Results of this call are State, actual position and Digital IO state (like PCube_getStatePosDio).

Ramp motion with specification of position, speed and acceleration	Function	Description
	PCube_moveRamp	Starts a ramp motion profile of the module specified by deviceID and moduleID. The target position is given in rad resp. m, target speed in rad/s resp. m/s and target acceleration in rad/s² resp. m/s².

Ramp motion with specification of position, speed and acceleration	Function	Description
	PCube_moveRamp	Starts a ramp motion profile of the module specified by deviceID and moduleID. The target position is given in rad resp. m, target speed in rad/s resp. m/s and target acceleration in rad/s² resp. m/s².
	PCube_moveRampInc	Starts a ramp motion profile of the module specified by deviceID and moduleID. The target position is given in Increments, target speed in Increments/s and target acceleration in Increments/s².
	PCube_moveRampExtended	Starts a ramp motion profile of the module specified by deviceID and moduleID. The target position is given in rad resp. m, target speed in rad/s resp. m/s and target acceleration in rad/s² resp. m/s². Results of this call are State, actual position and Digital IO state (like PCube_getStatePosDio).

Constant speed motion	Function	Description
	PCube_moveVel	Starts a constant speed motion. Target speed is specified in rad/s resp. m/s.
	PCube_moveVelInc	Starts a constant speed motion. Target speed is specified in Increments/s.
	PCube_moveVelExtended	Starts a constant speed motion. Target speed is specified in rad/s resp. m/s. Results of this call are State, actual position and Digital IO state (like PCube_getStatePosDio).

Constant current motion	Function	Description
	PCube_moveCur	Starts a constant current motion. The target currrent is specified in A.
	PCube_moveCurInc	Starts a constant current motion. The target currrent is specified in Digits.
	PCube_moveCurExtended	Starts a constant current motion. The target currrent is specified in A. Results of this call are State, actual position and Digital IO state (like PCube_getStatePosDio).

Motion with specification of target position and time	Function	Description
	PCube_moveStep	Starts motion to the target position specified in rad resp. m. Target time for the ride is specified in ms.
	PCube_moveStepInc	Starts motion to the target position specified in increments. Target time for the ride is specified in ms.
	PCube_moveStepExtended	Starts motion to the target position specified in rad resp. m. Target time for the ride is specified in ms. Results of this call are State, actual position and Digital IO state (like PCube_getStatePosDio).

Synchronized Start of all drives with new targets	Function	Description
	PCube_startMotionAll	If the configuration of all connected modules has been altered (see module configuration) a synchronous motion command can be issued. By sending PCube_startMotionAll all connected modules start their motion command at exactly the same time. For CAN-Bus only.

Function	Description	Description
	PCube_initEMS_IO	Initializes an EMS module on the bus specified by deviceID, Type and Serial number. Result of this call is a valid moduleID for the chosen EMS module. For CAN-Bus only.
	PCube_getDataEMS_DIO	Retrieves data from the EMS module specified by deviceID and moduleID. Result is the state of the channel specified by channelID. The moduleID must have been requested prior to this using PCube_initEMS_IO. For CAN-Bus only. For Digital EMS IO-Moduls only.
	PCube_getDataEMS_AIO	Retrieves data from the EMS module specified by deviceID and moduleID. Result is the value of the channel specified by channelID. The moduleID must have been requested prior to this using PCube_initEMS_IO. For CAN-Bus only. For Analog EMS IO-Moduls only.
	PCube_setDataEMS_DIO	Sets data on the EMS module specified by deviceID and moduleID. The state given is transferred to the channel chosen with channelID. For CAN-Bus only. For Digital EMS IO-Moduls only.

Function	Description	Description
	PCube_initEMS_IO	Initializes an EMS module on the bus specified by deviceID, Type and Serial number. Result of this call is a valid moduleID for the chosen EMS module. For CAN-Bus only.
	PCube_setDataEMS_AIO	Sets data on the EMS module specified by deviceID and moduleID. The value given is transferred to the channel chosen with channelID. For CAN-Bus only. For Analog EMS IO-Moduls only.

Retrieve data from DLR Force torque sensor	Function	Description
	PCube_initDLR_FTS	Initializes the Force Ttorque sensor on the bus by specifiing the deviceID. For CAN-Bus only. Only one Sensor per device (bus) is allowed.
	PCube_getDataDLR_FTS	Retrieves data from the Force Torque sensor. Results are 3 force values (X,Y,Z) and 3 torque values (X,Y,Z) as well as the sensor state. For CAN-Bus only.

Retrieve data from SCHUNK Force torque sensor	Function	Description
	PCube_getDataSCHUNK_FTC	Retrieves data from the Force Torque sensor. Results are 3 force values (X,Y,Z) and 3 torque values (X,Y,Z) or 3 translation values (X,Y,Z) and 3 rotation values (X,Y,Z) as well as the sensor state. For CAN-Bus only.
	PCube_setNullSCHUNK_FTC	Nulls the Force Torque sensor. Results is the sensor state. For CAN-Bus only.

Retrieve data from MP55, produced by HBM	Function	Description
	PCube_getDataMP55_IO	Retrieves the actual measurement data from a MP55 specified by deviceID and moduleID. For CAN-Bus only.
	PCube_setTaraMP55_IO	Tares the MP55 specified by deviceID and moduleID. For CAN-Bus only.

Time functions	Function	Description
	PCube_getPosTime	Retrieves the actual position with a time stamp. Time is measured in Milliseconds and counts from 0 to 65536 ms (ca. 65s).
	PCube_resetTime	Resets the internal clock to zero.
	PCube_resetTimeAll	Resets the clock of all bus connected drives to zero. For CAN-Bus only.

Set Position	Function	Description
	PCube_setPos	Sets a new position value in rad rsp. m. Works only when drive is not in motion!
	PCube_setPosInc	Sets a new position value in encoder ticks. Works only when drive is not in motion!

Functions for Scanners	Function	Description
	PCube_getScannerPosFallEdge	Retrieves the position saved upon trigger occurence „Falling edge on input SW3“. Returns the last position saved at trigger time in rad rsp. m.
	PCube_getScannerPosRisgEdge	Retrieves the position saved upon trigger occurence „Rising edge on input SW3“. Returns the last position saved at trigger time in rad rsp. m.
	PCube_moveCosLoop	Starts an automatic continous Loop between actual and target position. Spee dis a result of the given period time for one complete sweep. The speed follows a sine profile.
	PCube_moveRampLoop	Starts an automatic continous Loop between actual and target position. Speed and acceleration need to be configured up front. The speed follows a trapezoidal profile.