DES RS232-Protocol Windows 32-Bit DLL

PROPRIETARY NOTICE
Documentation DES_WinDLL

CH-Sachseln, 19.08.2004

The latest edition of this DES_WinDLL documentation may also be found on the internet site http://www.maxonmotor.com
(see «Downloads» under «Service»).

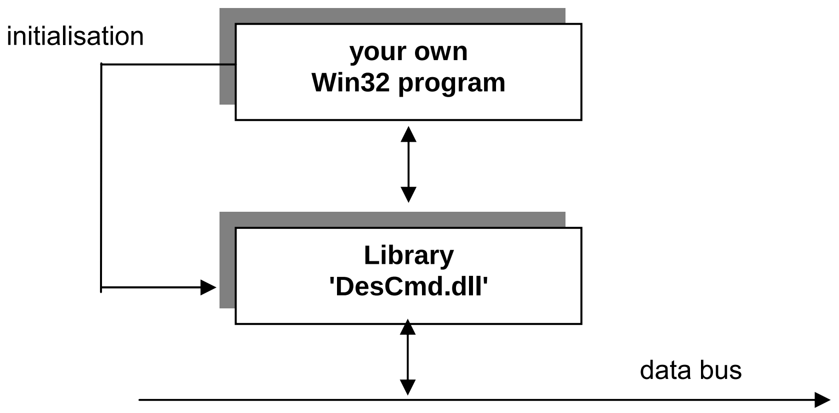
TABLE OF CONTENTS

Introduction

The library 'DesCmd.dll' is an implementation of the DES RS232 protocol for the communication between DES and a personal computer. Using this library is a simple way to develop your own application for velocity control applications on DES (**D**igital **E**C **S**ervoamplifier) based systems.

This library is running on each **Windows 32-Bit Operating System**. You can include it into different programming environments. All the DES RS232 commands are implemented and they can be called directly from your own program. You don't have to care about the protocol details. The only thing you have to ensure is a correct communication setting of the serial port (Portname, Baudrate, Timeout, Trials).

The library 'DesCmd.dll' offers the whole set of DES RS232 commands. Generating data frames, sending and receiving these frames is the business of this communication layer.



Library Hierarchy

For your own program you have to include this library.

This manual consists of three chapters and the appendix.

- **Chapter 1:** In this chapter all library functions are described. For further information have a look at the 'DES Communication Guide'.
- **Chapter 2:** How can you include these library functions into your own program? This question will be answered in this chapter. There's also a section about including the library into the 'LabView' environment.
- **Chapter 3:** The third chapter shows you some examples of windows applications written with different programming languages. At the beginning the initialisation of the serial port (COM1, ...) is described. Also all necessary files are here noted, so that the program is executable.
  The demo examples with the newest Windows DLL can be downloaded by the InterNet.
- **Appendix:** The appendix lists all constants, declarations, structure definitions and error codes concerning the library.

1. Library Functions

1.1. Configuration

DES_InitCommunication

BOOL __stdcall DES_InitCommunication(char portName[], __int32 baudrate, DWORD timeout, DWORD trials);

Description:	Opens and configures the serial port for sending DES RS232 commands.	
Parameter:	char portName[]:	Specified the serial port Possible values: COM1, COM2, ..., COM9
	__int32 baudrate:	Specifies the communication baudrate. Possible values: 9600, 19200, 38400, 57600, 115200 [baud]
	DWORD timeout:	Timeout for reading functions. If the time between two bytes exceeds the timeout limit the reading function aborts. The unit is [ms]
	DWORD trials:	Number of trials if the communication fails
Return Value:	BOOL:	Nonzero if successful; otherwise 0.

DES_OpenCommunication

BOOL __stdcall DES_OpenCommunication(char portName[]);

Description:	Open the serial port (COMx). Last configured setting or default setting is used.	
Parameter:	char portName:	Serial port (COM1, ...)
Return Value:	BOOL:	Nonzero if successful; otherwise 0.

DES_SearchCommunicationSetting

BOOL __stdcall DES_SearchCommunicationSetting(BOOL startAtBeginning, BOOL showMsg,
char foundPort[]);

Description:	Search the serial port (COM1, ...) and the adjustments.	
Parameter:	BOOL startAtBeginning:	TRUE: Start the search at beginning, FALSE: Start after the port which was found last
	BOOL showMsg:	TRUE: Show message box, FALSE: no message box
	char foundPort[]:	Found port
Return Value:	BOOL:	Nonzero if successful; otherwise 0.

DES_GetCommunicationSetting

BOOL __stdcall DES_GetCommunicationSetting(char portName[], __int32* baudrate,
DWORD* timeout, DWORD* trials);

Description:	Read the configuration setting of the serial port (COM1, ...).	
Parameter:	char portName[]:	Serial port (COM1, ...)
	__int32* baudrate:	Pointer to baudrate
	DWORD* timeout:	Pointer to timeout for reading functions. The unit is [ms]
	DWORD* trials:	Pointer to number of trials if the communication fails
Return Value:	BOOL:	Nonzero if successful; otherwise 0.

DES_CloseCommunication

BOOL __stdcall DES_CloseCommunication(char portName[]);

Description:	Close the serial port and release it for other applications.	
Parameter:	char portName[]:	Serial port (COM1, ...)
Return Value:	BOOL:	Nonzero if successful; otherwise 0.

DES_InitCommunicationDlg

BOOL __stdcall DES_InitCommunicationDlg(char portName[]);

Description:	Dialog for initialisation.	
Parameter:	char portName[]:	Serial port (COM1, ...)
Return Value:	BOOL:	Nonzero if successful; otherwise 0.

1.2. Command Layer

1.2.1. Status Functions

DES_ReadSysStatus

BOOL __stdcall DES_ReadSysStatus(char portName[], WORD* sysStatus);

Description:	Execute the DES RS232 command 'ReadSysStatus' (OpCode = 0x01). Read the system status of the DES. The system status is a 16-bit value containing different flags.	
Parameter:	char portName[]: WORD* sysStatus:	Serial port (COM1, ...) Pointer to 16-bit status variable. For exact information see in the appendix under System Operating Status.
Return Value:	BOOL:	Nonzero if successful; otherwise 0.

DES_ReadError

BOOL __stdcall DES_ReadError(char portName[], WORD* error);

Description:	Execute the DES RS232 command 'ReadError' (OpCode = 0x02). Read a 16-bit value of system errors.	
Parameter:	char portName[]: WORD* error:	Serial port (COM1, ...) Pointer to variable. For exact description see in the appendix under Standard Error Message.
Return Value:	BOOL:	Nonzero if successful; otherwise 0.

DES_ClearError

BOOL __stdcall DES_ClearError(char portName[]);

Description:	Execute the DES RS232 command 'ClearError' (Opcode = 0x03). Clear the system error.	
Parameter:	char portName[]:	Serial port (COM1, ...)
Return Value:	BOOL:	Nonzero if successful; otherwise 0.

DES_Reset

BOOL __stdcall DES_Reset(char portName[]);

Description:	Execute the DES RS232 command 'Reset' (OpCode 0x04). Reset the system by restarting the software.	
Parameter:	char portName[]:	Serial port (COM1, ...)
Return Value:	BOOL:	Nonzero if successful; otherwise 0.

DES_Enable

BOOL __stdcall DES_Enable(char portName[], WORD enable);

Description:	Execute the DES RS232 command 'Enable' (OpCode 0x05). Set the system to enabled or disabled state. The DES has to be configured for a software setting of enable. If the hardware enable is activated this command has no effect.	
Parameter:	char portName[]: WORD enable:	Serial port (COM1, ...) New state of the system Possible values: 0 = Disable or 1 = Enable
Return Value:	BOOL:	Nonzero if successful; otherwise 0.

1.2.2. Service Functions

DES_ReadAddrVariable

BOOL DES_ReadAddrVariabel(char portName[], WORD address, WORD parType, void* param)

Description:	Execute the DES RS232 command 'ReadAddrVariable' (OpCode = 0x12). Read a value at a given address in the memory. This function can only be used in the service mode.	
Parameter:	char portName[]: WORD address: WORD parType: void* param:	Serial port (COM1, ...) Memory address of the variable Data format of the variable Possible values: 0 = WORD or 1 = LWORD Pointer to the parameter.
Return Value:	BOOL:	Nonzero if successful; otherwise 0.

1.2.3. System Parameter Functions

DES_ReadTempParam

BOOL __stdcall DES_ReadTempParam(char portName[], WORD parNb, WORD parType, void* param);

Description:	Execute the DES RS232 command 'ReadTempParam' (OpCode = 0x14). Read the requested temporary system parameter from DES RAM.	
Parameter:	char portName[]: WORD parNb: WORD parType: void* param:	Serial port (COM1, ...) Number of the system parameter. For exact information see in the appendix under DES System Parameter. Data format of the variable Possible values: 0 = WORD or 1 = LWORD Pointer to temporary system parameter
Return Value:	BOOL:	Nonzero if successful; otherwise 0.

DES_SetTempParam

BOOL __stdcall DES_SetTempParam(char portName[], WORD parNb, WORD parType, void* param);

Description:	Execute the DES RS232 command 'SetTempParam' (OpCode = 0x15). Write a new value to a temporary system parameter.	
Parameter:	char portName[]: WORD parNb: WORD parType: void* param:	Serial port (COM1, ...) Number of the system parameter. For exact information see in the appendix under DES System Parameter. Date format of the variable Possible value: 0 = WORD or 1 = LWORD Pointer to temporary system parameter
Return Value:	BOOL:	Nonzero if successful; otherwise 0.

DES_ResetTempParam

BOOL __stdcall DES_ResetTempParam(char portName[]);

Description:	Execute the DES RS232 command 'ResetTempParam' (OpCode = 0x16). Copy the permanent system parameters contained in the EEPROM memory into the temporary parameter set.	
Parameter:	char portName[]:	Serial port (COM1, ...)
Return Value:	BOOL:	Nonzero if successful; otherwise 0.

DES_SaveTempParam

BOOL __stdcall DES_SaveTempParam(char portName[]);

Description:	Execute the DES RS232 command 'SaveTempParam' (OpCode = 0x17). Save the temporary parameters to the EEPROM.	
Parameter:	char portName[]:	Serial port (COM1, ...)
Return Value:	BOOL:	Nonzero if successful; otherwise 0.

DES_ReadAllTempParam

BOOL __stdcall DES_ReadAllTempParam(char portName[], DES_SysParam* sysParam);

Description:	Execute the DES RS232 command 'ReadAllTempParam' (OpCode = 0x18). Read all temporary system parameters. The system parameters structure 'DES_SysParam' is described in the appendix.	
Parameter:	char portName[]: DES_SysParam* sysParam:	Serial port (COM1, ...) Pointer to the temporary parameters
Return Value:	BOOL:	Nonzero if successful; otherwise 0.

DES_SetAllTempParam

BOOL __stdcall DES_SetAllTempParam(char portName[], DES_SysParam* sysParam);

Description:	Execute the DES RS232 command 'SetAllTempParam' (OpCode = 0x19). Write all temporary system parameters. The system parameter structure 'DES_SysParam' is discribed in the section 'Data Structures' in the appendix.	
Parameter:	char portName[]: DES_SysParam* sysParam:	Serial port (COM1, ...) Pointer to the new system parameter values
Return Value:	BOOL:	Nonzero if successful; otherwise 0.

DES_ReadVersion

BOOL __stdcall DES_ReadVersion(char portName[], WORD* softVersion, WORD* hardVersion, WORD* appNb, WORD* appVersion);

Description:	Execute the DES RS232 command 'ReadVersion' (OpCode = 0x1A). Read the actual (software/hardware/application) version of the connected DES. application number, application version, hardware version and software version of the connected DES.	
Parameter:	char portName[]: WORD* softVersion: WORD* hardVersion: WORD* appNb: WORD* appVersion:	Serial port (COM1, ...) Pointer to the software version of the DES Pointer to the hardware version of the DES Pointer to application number of the DES Pointer to application version of the DES
Return Value:	BOOL:	Nonzero if successful; otherwise 0.

DES_SetDefaultSysParam

BOOL __stdcall DES_SetDefaultSysParam(char portName[]);

Description:	Execute the DES RS232 command 'SysParSetDefault' (OpCode = 0x1B). Set all system parameters to default. The system parameter structure is described in the section DES System Parameter. Function only available since Software Version 0x1050 and higher!	
Parameter:	char portName[]:	Serial port (COM1, ...)
Return Value:	BOOL:	Nonzero if successful; otherwise 0.

1.2.4. Setting Functions

DES_SetVelocity

BOOL __stdcall DES_SetVelocity(char portName[], short velocity);

Description:	Execute the DES RS232 command 'SetVelocity' (OpCode = 0x21). Set the new velocity of the rotor. This function is only available in speed regulation mode.	
Parameter:	char portName[]: short velocity:	Serial port (COM1, ...) New velocity [rpm] of the rotor
Return Value:	BOOL:	Nonzero if successful; otherwise 0.

DES_SetCurrent

BOOL __stdcall DES_SetCurrent(char portName[], short current);

Description:	Execute the DES RS232 command 'SetCurrent' (OpCode = 0x22). Set a new current amplitude. This function is only available in current regulation mode.	
Parameter:	char portName[]: short current:	Serial port (COM1, ...). New current amplitude [mA]
Return Value:	BOOL:	Nonzero if successful; otherwise 0.

DES_StopMotion

BOOL __stdcall DES_StopMotion(char portName[]);

Description:	Execute the DES RS232 command 'SetMotion' (OpCode = 0x23). This command changes the stopping state. If the motor is already stopped it will be released. The digital input STOP has the same behaviour. This function is only available in speed regulation mode.	
Parameter:	char portName[]:	Serial port (COM1, ...)
Return Value:	BOOL:	Nonzero if successful; otherwise 0.

1.2.5. Monitor Functions

DES_ReadVelocityIsMust

BOOL __stdcall DES_ReadVelocityIsMust(char portName[], WORD type, short* isVelocity, short* mustVelocity);

Description:	Execute the DES RS232 command 'ReadVelocityIsMust' (OpCode = 0x28). Read the effective and requested velocity of the motor (Mean or Realtime values). Different types only available since Software Version 0x1040 and higher!		
Parameter:	char portName[]:	Serial port (COM1, ...)	
	WORD type:	0 = Mean value; 1 = Realtime value	
	short* isVelocity:	type = 0:	Mean velocity [rpm],
		type = 1:	Effective velocity [rpm]
	short* mustVelocity:	type = 0:	Requested velocity [rpm],
		type = 1:	Requested velocity [rpm]
Return Value:	BOOL:	Nonzero if successful; otherwise 0.	

DES_ReadCurrentIsMust

BOOL __stdcall DES_ReadCurrentIsMust(char portName[], WORD type, short* isQCurrent, short* isDCurrent, short* mustCurAMp, short* currentAngle);

Description:	Execute the DES RS232 command 'ReadCurrentIsMust' (OpCode = 0x29). Read the effective and requested current components of the motor (Mean or Realtime values). Different types only available from Software Version 0x1040 and higher!		
Parameter:	char portName[]:	Serial port (COM1, ...)	
	WORD type:	0 = Mean value, 1 = Realtime value	
	short* isQCurrent:	type = 0:	Mean value of Q-axis component of actual current [mA] ($\Rightarrow$ Torque)
		type = 1:	Q-axis component of actual current (Torque)
	short* isDCurrent:	type = 0:	Mean value of D-axis component of actual current [mA] (≈ 0)
		type = 1:	D-axis component of actual current (≈ 0)
	short* mustCurAmp:	type = 0 or 1:	Amplitude of requested current [mA]
	short* currentAngle:	type = 0 or 1:	Phase of the rotor [qc]
Return Value:	BOOL:	Nonzero if successful; otherwise 0.	

1.2.6. Recording Functions

DES_SetupRecorder

BOOL __stdcall DES_SetupRecorder(char portName[], WORD samplePeriod, WORD paramNbAddress);

Description:	Execute the DES RS232 command 'SetupRecorder' (OpCode = 0x30). Set up the recorder in the current regulator (10kHz).	
Parameter:	char portName[]:	Serial port (COM1, ...)
	WORD samplePeriod:	Sampling period as a factor of 0.1ms (e.g. 124 = 12.4ms)
	WORD paramNbAddress:	Number of a system parameter or a numbered status variable. If the number is greater than 0x0300 it represents a memory address.
Return Value:	BOOL:	Nonzero if successful; otherwise 0.

DES_RecordData

BOOL __stdcall DES_RecordData(char portName[], WORD* data, DWORD bufferLength,
DWORD* read, DWORD timeout, WORD jump);

Description:	Execute the DES RS232 command 'RecordData' (OpCode = 0x31). Start recording data. The sampling starts immediately. The recording is stopped after 256 samples. The current jump (SW Version 0x1040 and higher) is only executed if the DES configured for a digital setting value.	
Parameter:	char portName[]:	Serial port (COM1, ...)
	WORD* data:	Pointer to the recordet samples
	DWORD bufferLength:	Length of the buffer
	DWORD* read:	Pointer to the number of data read
	DWORD timeout:	Timeout for reading functions. If the time between two bytes exceeds the timeout limit, the reading function aborts. The unit is 'ms'.
	WORD jump:	Setting value jumping amplitude; 0 = No jump
Return Value:	BOOL:	Nonzero if successful; otherwise 0.

DES_ReadNVariables

BOOL __stdcall DES_ReadNVariables(char portName[], WORD nbOfVariables,
WORD* parNumbersAddresses, void* dataVector);

Description:	Execute the DES RS232 command 'ReadNVariables' (OpCode = 0x32). Read the values of a number of different variables synchronously.	
Parameter:	char portName[]:	Serial port (COM1, ...)
	WORD nbOfVariables:	Number of variables to read
	WORD* parNumbersAddresses:	Pointer to the numbers of system parameters or a numbered status variable. If the number is greater than 0x0300 it represents a memory address.
	void* dataVector:	Pointer to the variables
Return Value:	BOOL:	Nonzero if successful; otherwise 0.

1.2.7. CAN Functions

DES_ResetCANError

BOOL __stdcall DES_ResetCANError(char portName[]);

Beschreibung:	Execute the DES RS232 command 'ResetCANError' (OpCode = 0x06). Reset the CAN errors. Only available since Software Version 0x1040 and higher!	
Parameter:	char portName[]:	Angabe der seriellen Schnittstelle (COM1, ...)
Rückgabewert:	BOOL:	Nonzero if successful; otherwise 0.

DES_ResetCAN

BOOL __stdcall DES_ResetCAN(char portName[]);

Beschreibung:	Execute the DES RS232 command 'ResetCAN' (OpCode = 0x07). Reset the CAN communication. Only available since Software Version 0x1040 and higher!	
Parameter:	char portName[]:	Angabe der seriellen Schnittstelle (COM1, ...)
Rückgabewert:	BOOL:	Nonzero if successful; otherwise 0.

DES_SetModuleID

BOOL __stdcall DES_SetModuleID(char portName[], WORD moduleID);

Description:	Execute the DES RS232 command 'SetModuleID' (OpCode = 0x39). Set CAN Modul-ID (max. 11bit). The module ID is set by DIP switches at the system power up. During operation it's possible to overwright temporary the module ID with the command 'SetModuleID'. The module ID determines the ID's for SDO communication (TxSDO ID = 1408 + ModuleID; RxSDO = 1536 + ModuleID).	
Parameter:	char portName[]: WORD moduleID:	Serial port (COM1, ...) Modul ID
Return Value:	BOOL:	Nonzero if successful; otherwise 0.

DES_SetTPDOID

BOOL __stdcall DES_SetTPDOID(char portName[], WORD transPDOID);

Description:	Execute the DES RS232 command 'SetTPOID' (OpCode 0x3B). Set CAN Transmit PDO ID (max. 11bit).	
Parameter:	char portName[]: WORD transPDOID:	Serial port (COM1, ...) Transmit ID
Return Value:	BOOL:	Nonzero if successful; otherwise 0.

DES_SetRPDOID

BOOL __stdcall DES_SetRPDOID(char portName[], WORD receivePDOID);

Description:	Execute the DES RS232 command 'SetRPDOID' (OpCode = 0x3C). Set CAN Receive-PDO ID (max. 11bit).	
Parameter:	char portName[]: WORD receivePDOID:	Serial port (COM1, ...) Receive ID
Return Value:	BOOL:	Nonzero if successful; otherwise 0.

DES_SendCANmsg

BOOL __stdcall DES_SendCANmsg(char portName[], WORD id, WORD dataA, WORD dataB, WORD dataC, WORD dataD);

Description:	Execute the DES RS232 command 'SendCANmsg' (OpCode = 0x3D). Send standard frame command.	
Parameter:	char portName[]: WORD id: WORD dataA: WORD dataB: WORD dataC: WORD dataD:	Serial port (COM1, ...) Module ID CAN DataBytes 2-1 CAN DataBytes 4-3 CAN DataBytes 6-5 CAN DataBytes 8-7
Return Value:	BOOL:	Nonzero if successful; otherwise 0.

DES_ReadModuleID

BOOL __stdcall DES_ReadModuleID(char portName[], WORD* moduleID);

Description:	Execute the DES RS232 command 'ReadModuleID' (OpCode = 0x3E). Read DES CAN Module ID.	
Parameter:	char portName[]: WORD* moduleID:	Serial port (COM1, ...) Pointer to die Modul ID
Return Value:	BOOL:	Nonzero if successful; otherwise 0.

DES_SetCAN_BCR1_BCR2

BOOL __stdcall DES_SetCAN_BCR1_BCR2(char portName[], WORD bcr1, WORD bcr2);

Description:	Execute the DES RS232 command 'SetCANBCR' (OpCode = 0x3F). Set CAN bit timing configuration register 1 and 2 (only available since Software Version 0x1040 and higher). The command is for internal use only. For exact information see in the appendix (CAN Bit Timing).	
Parameter:	char portName[]:	Serial port (COM1, ...)
	WORD bcr1:	CAN bit timing configuration register 1
	WORD bcr2:	CAN bit timing configuration register 2
Return Value:	BOOL:	Nonzero if successful; otherwise 0.

DES_SetCAN_Bitrate

BOOL __stdcall DES_SetCAN_Bitrate(char portName[], WORD index);

Description:	Execute the DES RS232 command 'SetCANBCR1' (OpCode = 0x40). Set CAN transfer rate to calculated values. See the section Bit Timing for more information about this configuration register. Only available since Software Version 0x1050 and higher!	
Parameter:	char portName[]:	Serial port (COM1, ...)
	WORD index:	Index for transfer rate
Return Value:	BOOL:	Nonzero if successful; otherwise 0.

DES_ReadCANError

BOOL __stdcall DES_ReadCANError(char portName[], WORD* error);

Description:	Execute the DES RS232 command 'SetCANBCR1' (OpCode = 0x43). Read a 16 bit value of the CAN error register (CAN Error Message).	
Parameter:	char portName[]:	Serial port (COM1, ...)
	WORD* error:	Frame containing the 16-bit error variable.
Return Value:	BOOL:	Nonzero if successful; otherwise 0.

DES_GetRemoteData

BOOL __stdcall DES_GetRemoteData(char portName[], WORD id,BYTE opCode, WORD* param, BYTE nbOfParam, WORD* returnParam,BYTE nbOfReturnParam);

Description:	Execute the DES RS232 command 'GetRemoteData' (OpCode = 0x44). Read data from other DES connected to the CAN bus.	
Parameter:	char portName[]:	Serial port (COM1, ...)
	WORD id:	Module ID of addressed DES
	BYTE opCode:	Operation code of the requested command
	WORD* param:	Parameters of the command
	BYTE nbOfParam:	Numbers of parameters
	WORD* returnParam:	Returned parameters
	BYTE nbOfReturnParam:	Numbers of returned parameters
Return Value:	BOOL:	Nonzero if successful; otherwise 0.

DES_ConfigPDO

BOOL __stdcall DES_ConfigPDO(char portName[], WORD action);

Description:	Execute the DES RS232 command 'ConfigPDO' (OpCode = 0x45). Switch on and off the PDO communication. The state of the PDO communication can be read with the system parameter 'CAN Config' (SysParam 41, Bit14). Only available since Software Version 0x1040 and higher!	
Parameter:	char portName[]: WORD action:	Serial port (COM1, ...) 0 = Switch OFF, 1 = Switch ON
Return Value:	BOOL:	Nonzero if successful; otherwise 0.

DES_SetRTRID

BOOL __stdcall DES_SetRTRID(char portName[], WORD rtrChannel, WORD rtrID);

Description:	Execute the DES RS232 command 'SetRTRID' (OpCode = 0x46). Set CAN Remote Request Frame ID (11 bit). There are two possible channels for remote request frames. Only available since Software Version 0x1040 and higher!	
Parameter:	char portName[]: WORD rtrChannel: WORD rtrID:	Serial port (COM1, ...) Channel: 0 for RTR0, 1 for RTR1 Remote Request ID
Return Value:	BOOL:	Nonzero if successful; otherwise 0.

DES_ConfigRTR

BOOL __stdcall DES_ConfigRTR(char portName[], WORD rtrChannel, WORD action);

Description:	Execute the DES RS232 command 'Config RTR' (OpCode = 0x47). Switch on and off the RTR communication. The state of the RTR communication can be read with the system parameter "CAN Config" (SysParam 41, Bit 13 = RTR0, Bit 14 = RTR1). Only available since Software Version 0x1040 and higher!	
Parameter:	char portName[]: WORD rtrChannel: WORD action:	Serial port (COM1, ...) 0 for RTR channel 0, 1 for RTR channel 1 0 = Switch OFF 1 = Switch ON, 2 = Reset Parameters
Return Value:	BOOL:	Nonzero if successful; otherwise 0.

DES_AddRTRParameter

BOOL __stdcall DES_AddRTRParameter(char portName[], WORD paramSel, WORD param, WORD *ack);

Description:	Execute the DES RS232 command 'AddRTRParameter' (OpCode = 4A). Register a new RTR parameter for the remote request frame. Reset the RTR parameter for this channel before adding new parameters. The maximum number of parameters are 4 words (4 x 16-bit). If the data buffer is full, the command sends a negative acknowledge ('F' = 0x0046). Only available since Software Version 0x1040 and higher!	
Parameter:	char portName[]: WORD paramSel: WORD param: WORD *ack:	Serial port (COM1, ...) Bit0: select channel (0 for channel 0 or 1 for channel 1) Bit4: 0 to select paramMode, 1 to select addressMode. Bit8: 0 to select Word (16bit), 1 to select LWord (32bit) Number (paramMode) or adress (addrMode) of the system parameter. See the section system parameters 'O(0x004F) = parameter added 'F'(0x0046) = parameter not added. The buffer is full.
Return Value:	BOOL:	Nonzero if successful; otherwise 0.

DES_GetRTRParameter

BOOL __stdcall DES_GetRTRParameter(char portName[], WORD rtrChannel, WORD index,
WORD *ack, WORD *format, WORD *param);

Description:	Execute the DES RS232 command 'GetRTRParameter' (OpCode = 4B). Read the registred RTR parameters. The maximum number of parameters is 4 words (4 x 16-bit). If there are less than 4 parameters registred, the command sends a negative acknowledge ('F' = 0x0046) for the parameters which are not available. Only available since Software Version 0x1040 and higher!	
Parameter:	char portName[]:	Serial port (COM1, ...)
	WORD rtrChannel:	Channel: 0 for RTR0, 1 for RTR1
	WORD index:	0,1,2,3 (if parameter is available)
	WORD *ack:	'O' (0x004F) = parameter available 'F' (0x0046) = parameter not available
	WORD *format:	Bit4: 0 = paramMode, 1 = addressMode Bit8: 0 = Word (16bit), 1 = LWord (32bit)
	WORD *param:	Number (paramMode) or address (addressMode) of the registered system parameter. See section system parameters.
Return Value:	BOOL:	Nonzero if successful; otherwise 0.

1.3. Dialog Layer

1.3.1. Status Dialogs

DES_ReadSysStatusDlg

BOOL __stdcall DES_ReadSysStatusDlg(char portName[]);

Description:	Open dialog to read system status of the DES.	
Parameter:	char portName[]:	Serial port (COM1, ...)
Return Value:	BOOL:	Nonzero if successful; otherwise 0.

DES_ReadErrorDlg

BOOL __stdcall DES_ReadErrorDlg(char portName[]);

Description:	Open dialog to read system errors.	
Parameter:	char portName[]:	Serial port (COM1, ...)
Return Value:	BOOL:	Nonzero if successful; otherwise 0.

DES_EnableDlg

BOOL __stdcall DES_EnableDlg(char portName[]);

Description:	Open dialog to enable or disable the system.	
Parameter:	char portName[]:	Serial port (COM1, ...)
Return Value:	BOOL:	Nonzero if successful; otherwise 0.

1.3.2. System Parameter Dialogs

DES_EditTempParamDlg

BOOL __stdcall DES_EditTempParamDlg(char portName[]);

Description:	Dialog to read and write the requested system parameter of the DES.	
Parameter:	char portName[]:	Serial port (COM1, ...)
Return Value:	BOOL:	Nonzero if successful; otherwise 0.

DES_EditAllTempParamDlg

BOOL __stdcall DES_EditAllTempParamDlg(char portName[]);

Description:	Dialog to read and write all temporary system parameters.	
Parameter:	char portName[]:	Serial port (COM1, ...)
Return Value:	BOOL:	Nonzero if successful; otherwise 0.

DES_ReadVersionDlg

BOOL __stdcall DES_ReadVersionDlg(char portName[]);

Description:	Dialog to read the application and the hardware version from the DES.	
Parameter:	char portName[]:	Serial port (COM1, ...)
Return Value:	BOOL:	Nonzero if successful; otherwise 0.

1.3.3. Setting Function Dialogs

DES_SetVelocityDlg

BOOL __stdcall DES_SetVelocityDlg(char portName[]);

Description:	Dialog to set a new velocity of the rotor.	
Parameter:	char portName[]:	Serial port (COM1, ...)
Return Value:	BOOL:	Nonzero if successful; otherwise 0.

DES_SetCurrentDlg

BOOL __stdcall DES_SetCurrentDlg(char portName []);

Description:	Dialog to set a new current amplitude.	
Parameter:	char portName[]:	Serial port (COM1, ...)
Return Value:	BOOL:	Nonzero if successful; otherwise 0.

1.3.4. Monitor Function Dialogs

DES_ReadVelocityIsMustDlg

BOOL __stdcall DES_ReadVelocityIsMustDlg(char portName[]);

Description:	Dialog to read the effective and requested velocity of the motor.	
Parameter:	char portName[]:	Serial port (COM1, ...)
Return Value:	BOOL:	Nonzero if successful; otherwise 0.

DES_ReadCurrentIsMustDlg

BOOL __stdcall DES_ReadCurrentIsMustDlg(char portName[]);

Description:	Dialog to read the effective and requested current components of the motor.	
Parameter:	char portName[]:	Serial port (COM1, ...)
Return Value:	BOOL:	Nonzero if successful; otherwise 0.

1.3.5. Recording Dialogs

DES_SetupRecorderDlg

BOOL __stdcall DES_SetupRecorderDlg(char portName[]);

Description:	Dialog to setup the recorder in the current regulator.	
Parameter:	char portName[]:	Serial port (COM1, ...)
Return Value:	BOOL:	Nonzero if successful; otherwise 0.

DES_RecordDataDlg

BOOL __stdcall DES_RecordDataDlg(char portName[]);

Description:	Dialog to start the recording.	
Parameter:	char portName[]:	Serial port (COM1, ...)
Return Value:	BOOL:	Nonzero if successful; otherwise 0.

DES_ReadNVariablesDlg

BOOL __stdcall DES_ReadNVariablesDlg(char portName[]);

Description:	Dialog to read the values of a number of different variables synchronously.	
Parameter:	char portName[]:	Serial port (COM1, ...)
Return Value:	BOOL:	Nonzero if successful; otherwise 0.

1.3.6. CAN Function Dialogs

DES_ResetCANErrorDlg

BOOL __stdcall DES_ResetCANErrorDlg(char portName[]);

Beschreibung:	Dialog to reset CAN errors Function available since Software Version 0x1040 and higher!	
Parameter:	char portName[]:	Serial port (COM1, ...)
Rückgabewert:	BOOL:	Nonzero if successful; otherwise 0.

DES_ResetCANDlg

BOOL __stdcall DES_ResetCANDlg(char portName[]);

Beschreibung:	Dialog to reset CAN communication. Function available since Software Version 0x1040 and higher!	
Parameter:	char portName[]:	Serial port (COM1, ...)
Rückgabewert:	BOOL:	Nonzero if successful; otherwise 0.

DES_SetModuleIDDlg

BOOL __stdcall DES_SetModuleIDDlg(char portName[]);

Description:	Dialog to set CAN Module ID.	
Parameter:	char portName[]:	Serial port (COM1, ...)
Return Value:	BOOL:	Nonzero if successful; otherwise 0.

DES_SetTPDOIDDlg

BOOL __stdcall DES_SetTPDOIDDlg(char portName[]);

Description:	Dialog to set Transmit PDO ID.	
Parameter:	char portName[]:	Serial port (COM1, ...)
Return Value:	BOOL:	Nonzero if successful; otherwise 0.

DES_SetRPDOIDDlg

BOOL __stdcall DES_SetRPDOIDDlg(char portName[]);

Description:	Dialog to set Receive PDO ID.	
Parameter:	char portName[]:	Serial port (COM1, ...)
Return Value:	BOOL:	Nonzero if successful; otherwise 0.

DES_SendCANmsgDlg

BOOL __stdcall DES_SendCANmsgDlg(char portName[]);

Description:	Dialog to send CAN standard frame message command.	
Parameter:	char portName[]:	Serial port (COM1, ...)
Return Value:	BOOL:	Nonzero if successful; otherwise 0.

DES_ReadModuleIDDlg

BOOL __stdcall DES_ReadModuleIDDlg(char portName[]);

Description:	Dialog to read CAN Module ID.	
Parameter:	char portName[]:	Serial port (COM1, ...)
Return Value:	BOOL:	Nonzero if successful; otherwise 0.

DES_SetCAN_BCR1_BCR2Dlg

BOOL __stdcall DES_SetCAN_BCR1_BCR2Dlg(char portName[]);

Description:	Dialog to set CAN bit timing configuration register BCR1 and BCR2. Function available since Software Version 0x1040 and higher!	
Parameter:	char portName[]:	Serial port (COM1, ...)
Return Value:	BOOL:	Nonzero if successful; otherwise 0.

DES_SetCAN_BitrateDlg

BOOL __stdcall DES_SetCAN_BitrateDlg(char portName[]);

Description:	Dialog to set CAN bit timing configuration registers BCR1 and BCR2. Function not available since Software Version 0x1050 and higher!	
Parameter:	char portName[]:	Serial port (COM1, ...)
Return Value:	BOOL:	Nonzero if successful; otherwise 0.

DES_ReadCANErrorDlg

BOOL __stdcall DES_ReadCANErrorDlg(char portName[]);

Description:	Dialog to read CAN errors.	
Parameter:	char portName[]:	Serial port (COM1, ...)
Return Value:	BOOL:	Nonzero if successful; otherwise 0.

DES_GetRemoteDataDlg

BOOL __stdcall DES_GetRemoteDataDlg(char portName[]);

Description:	Dialog to read data from other DES connected to the CAN bus.	
Parameter:	char portName[]:	Serial port (COM1, ...)
Return Value:	BOOL:	Nonzero if successful; otherwise 0.

DES_ConfigPDODlg

BOOL __stdcall DES_ConfigPDODlg(char portName[]);

Description:	Dialog to switch on and off the PDO communication. Function available since Software Version 0x1040 and higher!	
Parameter:	char portName[]:	Serial port (COM1, ...)
Return Value:	BOOL:	Nonzero if successful; otherwise 0.

DES_SetRTRIDDlg

BOOL __stdcall DES_SetRTRIDDlg(char portName[]);

Description:	Dialog to set Remote Request Frame ID of channel 0 or 1. Function available since Software Version 0x1040 and higher!	
Parameter:	char portName[]:	Serial port (COM1, ...)
Return Value:	BOOL:	Nonzero if successful; otherwise 0.

DES_SetRTRID0Dlg

BOOL __stdcall DES_SetRTRID0Dlg(char portName[]);

Description:	Dialog to set Remote Request Frame ID of channel 0. Only available since Software Version 0x1040 and higher!	
Parameter:	char portName[]:	Serial port (COM1, ...)
Return Value:	BOOL:	Nonzero if successful; otherwise 0.

DES_SetRTRID1Dlg

BOOL __stdcall DES_SetRTRID1Dlg(char portName[]);

Description:	Dialog to set Remote Request Frame ID of channel 1. Only available since Software Version 0x1040 and higher!	
Parameter:	char portName[]:	Serial port (COM1, ...)
Return Value:	BOOL:	Nonzero if successful; otherwise 0.

DES_ConfigRTRDlg

BOOL __stdcall DES_ConfigRTRDlg(char portName[]);

Description:	Dialog to switch on and off the RTR communication. Only available since Software Version 0x1040 and higher!	
Parameter:	char portName[]:	Serial port (COM1, ...)
Return Value:	BOOL:	Nonzero if successful; otherwise 0.

DES_AddRTRParameterDlg

BOOL __stdcall DES_AddRTRParameterDlg(char portName[]);

Description:	Dialog to register a new RTR parameter for the remote request frame. Only available since Software Version 0x1040 and higher!	
Parameter:	char portName[]:	Serial port (COM1, ...)
Return Value:	BOOL:	Nonzero if successful; otherwise 0.

DES_GetRTRParameterDlg

BOOL __stdcall DES_GetRTRParameterDlg(char portName[]);

Description:	Dialog to read the registered RTR parameters. Only available since Software Version 0x1040 and higher!	
Parameter:	char portName[]:	Serial port (COM1, ...)
Return Value:	BOOL:	Nonzero if successful; otherwise 0.

1.4. CAN Command Layer

1.4.1. CAN Status Functions

DES_CAN_ReadSysStatus

BOOL __stdcall DES_CAN_ReadSysStatus(char portName[], WORD dest, WORD* sysStatus);

Description:	Execute the DES RS232 command ‘ReadSysStatus’ (OpCode = 0x01). Read the system status of the DES. The system status is a 16-bit value containing different flags.	
Parameter:	char portName[]: WORD dest: WORD* sysStatus:	Serial port (COM1, ...) Module ID of addressed DES Pointer to 16-bit status variable. For exact information see in the appendix (System Operating Status)
Return Value:	BOOL:	Nonzero if successful; otherwise 0.

DES_CAN_ReadError

BOOL __stdcall DES_CAN_ReadError(char portName[], WORD dest, WORD* error);

Description:	Execute the DES RS232 command ‘ReadError’ (OpCode = 0x02) Read a 16-bit value of system errors.	
Parameter:	char portName[]: WORD dest: WORD* error	Serial port (COM1, ...) Module ID of addressed DES Pointer to 16-bit error variable. For exact information see in the appendix (Standard Error Message)
Return Value:	BOOL:	Nonzero if successful; otherwise 0.

DES_CAN_ClearError

BOOL __stdcall DES_CAN_ClearError(char portName[], WORD dest);

Description:	Execute the DES RS232 command 'ClearError' (OpCode = 0x03). Clear the system error.	
Parameter:	char portName[]: WORD dest:	Serial port (COM1, ...) Module ID of addressed DES
Return Value:	BOOL:	Nonzero if successful; otherwise 0.

DES_CAN_Enable

BOOL __stdcall DES_CAN_Enable(char portName[], WORD dest, WORD enable);

Description:	Execute the DES RS232 command 'Enable' (OpCode = 0x05). Set the system to enabled or disabled state. The DES has to be configured for a software setting of enable. If the hardware enable is activated this command has no effect.	
Parameter:	char portName[]: WORD dest: WORD enable:	Serial port (COM1, ...) Module ID of addressed DES New state of the system Possible values: 0 = Disable; 1 = Enable
Return Value:	BOOL:	Nonzero if successful; otherwise 0.

DES_CAN_Reset

BOOL __stdcall DES_CAN_Reset(char portName[], WORD dest);

Description:	Execute the DES RS232 command 'Reset' (OpCode = 0x04). Reset the system by restarting the software.	
Parameter:	char portName[]: WORD dest:	Serial port (COM1, ...) Module ID of addressed DES
Return Value:	BOOL:	Nonzero if successful; otherwise 0.

1.4.2. CAN Service Functions

DES_CAN_ReadAddrVariable

BOOL __stdcall DES_CAN_ReadAddrVariable(char portName[], WORD dest, WORD address,
WORD parType, void* param);

Description:	Execute the DES RS232 command 'ReadAddrVariable' (OpCode = 0x12). Read a value at a given address in the memory. This function can only be used in the service mode.	
Parameter:	char portName[]: WORD dest: WORD address: WORD parType: void* param:	Serial port (COM1, ...) Module ID of addressed DES Memory address of the variable Data format of the variable Possible values: 0 = WORD; 1 = LWORD Pointer to the parameter
Return Value:	BOOL:	Nonzero if successful; otherwise 0.

1.4.3. CAN System Parameter Functions

DES_CAN_ResetTempParam

BOOL __stdcall DES_CAN_ResetTempParam(char portName[], WORD dest);

Description:	Execute the DES RS232 command 'ResetTempParam' (OpCode = 0x16). Copy the permanent system parameters contained in the EEPROM memory into the temporary parameter set.	
Parameter:	char portName[]: WORD:	Serial port (COM1, ...) Module ID of addressed DES
Return Value:	BOOL:	Nonzero if successful; otherwise 0.

DES_CAN_ReadTempParam

BOOL __stdcall DES_CAN_ReadTempParam(char portName[], WORD dest, WORD parNb, WORD parType, void* param);

Description:	Execute the DES RS232 command 'ReadTempParam' (OpCode = 0x14). Read the requested temporary system parameter from DES RAM.	
Parameter:	char portName[]: WORD dest: WORD parNb: WORD parType: void* param:	Serial port (COM1, ...) Module ID of addressed DES Number of the system parameter. For exact information see in the appendix under DES System Parameter. Data format of the variable: 0 = WORD; 1 = LWORD Pointer to temporary system parameter
Return Value:	BOOL:	Nonzero if successful; otherwise 0.

DES_CAN_SetTempParam

BOOL __stdcall DES_CAN_SetTempParam(char portName[], WORD dest, WORD parNb, WORD parType, void* param);

Description:	Execute the DES RS232 command 'SetTempParam' (OpCode = 0x15). Write a new value to a temporary system parameter.	
Parameter:	char portName[]: WORD dest: WORD parNb: WORD parType: void* param:	Serial port (COM1, ...) Module ID of addressed DES Number of the system parameter, For exact information see in the appendix under DES System Parameter. Date format of the variable Possible value: 0 = WORD, 1 = LWORD Pointer to temporary system parameter
Return Value:	BOOL:	Nonzero if successful; otherwise 0.

DES_CAN_SaveTempParam

BOOL __stdcall DES_CAN_SaveTempParam(char portName[], WORD dest);

Description:	Execute the DES RS232 command 'SaveTempParam' (OpCode = 0x17). Save the temporary parameters to the EEPROM.	
Parameter:	char portName[]: WORD dest:	Serial port (COM1, ...) Module ID of addressed DES
Return Value:	BOOL:	Nonzero if successful; otherwise 0.

DES_CAN_ReadAllTempParam

BOOL __stdcall DES_CAN_ReadAllTempParam(char portName[], WORD dest, DES_SysParam* sysParam);

Description:	Execute the DES RS232 command 'ReadAllTempParam' (OpCode = 0x18). Read all temporary system parameters. The system parameters structure 'DES_SysParam' is described in the appendix.	
Parameter:	char portName[]: WORD dest: DES_SysParam* sysParam:	Serial port (COM1, ...) Module ID of addressed DES Pointer to the temporary parameters
Return Value:	BOOL:	Nonzero if successful; otherwise 0.

DES_CAN_SetAllTempParam

BOOL __stdcall DES_CAN_SetAllTempParam(char portName[], WORD dest, DES_SysParam* sysParam);

Description:	Execute the DES RS232 command 'SetAllTempParam' (OpCode = 0x19). Write all temporary system parameters. The system parameter structure 'DES_SysParam' is discribed in the section 'Data Structures' in the appendix.	
Parameter:	char portName[]: WORD dest: DES_SysParam* sysParam:	Serial port (COM1, ...) Module ID of addressed DES Pointer to the new system parameter values
Return Value:	BOOL:	Nonzero if successful; otherwise 0.

DES_CAN_ReadVersion

BOOL __stdcall DES_CAN_ReadVersion(char portName[], WORD dest, WORD* param1, WORD* param2, WORD versionGroup);

Description:	Execute the DES RS232 command 'ReadVersion' (OpCode = 0x1A). Read the hardware version and software version or the application number and application version of the DES. New command since Software version 0x1040 and higher.	
Parameter:	char portName[]: WORD dest: CAN (versionGroup = 0): WORD* param1: WORD* param2: CAN (versionGroup = 1): WORD* param1: WORD* param2: WORD versionGroup:	Serial port (COM1, ...) Module ID of addressed DES Pointer to the software version of the DES Pointer to the hardware version of the DES Pointer to application number of the DES Pointer to application version of the DES 0 = soft- and hardware version 1 = application number and version
Return Value:	BOOL:	Nonzero if successful; otherwise 0.

DES_CAN_SetDefaultSysParam

BOOL __stdcall DES_CAN_SetDefaultSysParam(char portName[], WORD dest);

Description:	Execute the DES RS232 command 'SysParSetDefault' (OpCode = 0x1B). Set all system parameters to default. The system parameter structure is described in the section DES System Parameter. Only available since Software Version 0x1050 and higher!	
Parameter:	char portName[]: WORD dest:	Serial port (COM1, ...) Module ID of addressed DES
Return Value:	BOOL:	Nonzero if successful; otherwise 0.

1.4.4. CAN Setting Functions

DES_CAN_SetVelocity

BOOL __stdcall DES_CAN_SetVelocity(char portName[], WORD dest, short velocity);

Description:	Execute the DES RS232 command ‘SetVelocity’ (OpCode = 0x21). Set a new velocity of the rotor. This function is only available in speed regulation mode.	
Parameter:	char portName[]: short velocity:	Serial port (COM1, ...) New velocity of the rotor [rpm]
Return Value:	BOOL:	Nonzero if successful; otherwise 0.

DES_CAN_SetCurrent

BOOL __stdcall DES_CAN_SetCurrent(char portName[], WORD dest, short current);

Description:	Execute the DES RS232 command ‘SetCurrent’ (OpCode = 0x22). Set a new current amplitude. This function is only available in current regulation mode.	
Parameter:	char portName[]: short current:	Serial port (COM1, ...) New current amplitude [mA]
Return Value:	BOOL:	Nonzero if successful; otherwise 0.

DES_CAN_StopMotion

BOOL __stdcall DES_CAN_StopMotion(char portName[], WORD dest);

Description:	Execute the DES RS232 command ‘SetMotion’ (OpCode = 0x23). This command changes the stopping state. If the motor is already stopped it will be released. The digital input STOP has the same behaviour. This function is only available in speed regulation mode.	
Parameter:	char portName[]: WORD dest:	Serial port (COM1, ...) Module ID of addressed DES
Return Value:	BOOL:	Nonzero if successful; otherwise 0.

1.4.5. CAN Monitor Functions

DES_CAN_ReadVelocityIsMust

BOOL __stdcall DES_CAN_ReadVelocityIsMust(char portName[], WORD dest, short* isVelocity, short* mustVelocity);

Description:	Execute the DES RS232 command ‘ReadVelocityIsMust’ (OpCode = 0x28). Read the effective and requested velocity of the motor (Mean or Realtime values). Different types only available from Software Version 0x1040 and higher!	
Parameter:	char portName[]: WORD dest: WORD type: short* isVelocity: short* mustVelocity:	Serial port (COM1, ...) Module ID of addressed DES 0 = Mean values; 1 = Realtime values type = 0: Mean velocity; type = 1: Effective velocity type = 0 = 1: Requested velocity
Return Value:	BOOL:	Nonzero if successful; otherwise 0.

DES_CAN_ReadCurrentIsMust

BOOL __stdcall DES_CAN_ReadCurrentIsMust(char portName[], WORD dest,
short* isCurrentQAxis, short* isCurrentDAxis, short* mustCurrentAmp, short* rotorAngle);

Description:	Execute the DES RS232 command 'ReadCurrentIsMust' (OpCode = 0x29). Read the effective and requested current components of the motor (Mean or Realtime values). Different types only available from Software Version 0x1040 and higher!		
Parameter:	char portName[]:	Serial port (COM1, ...)	
	WORD dest:	Module ID of addressed DES	
	WORD type:	0 = Mean values, 1 = Realtime values	
	short* isQCurrent:	type = 0:	Mean value of Q-axis component of actual current [mA] (⇒ Torque)
		type = 1:	Q-axis component of actual current (⇒ Torque)
	short* isDCurrent:	type = 0:	Mean value of D-axis component of actual current [mA] (≈ 0)
		type = 1:	D-axis component of actual current (≈ 0)
	short* mustCurAmp:	type = 0 or 1:	Amplitude of requested current [mA]
	short* currentAngle:	type = 0 or 1:	Phase of the rotor [qc]
Return Value:	BOOL:	Nonzero if successful; otherwise 0.	

1.4.6. CAN Functions

DES_CAN_ResetCANError

BOOL __stdcall DES_CAN_ResetCANError(char portName[], WORD dest);

Beschreibung:	Execute the DES RS232 command 'ResetCANError' (OpCode = 0x06). Reset the CAN errors. Only available since Software Version 0x1040 and higher!	
Parameter:	char portName[]:	Angabe der seriellen Schnittstelle (COM1, ...)
	WORD dest:	Module ID of addressed DES
Rückgabewert:	BOOL:	Nonzero if successful; otherwise 0.

DES_CAN_ResetCAN

BOOL __stdcall DES_CAN_ResetCAN(char portName[], WORD dest);

Description:	Execute the DES RS232 command 'ResetCAN' (OpCode = 0x07). Reset the CAN communication. Only available since Software Version 0x1040 and higher!	
Parameter:	char portName[]:	Serial port (COM1, ...)
	WORD dest:	Module ID of addressed DES
Return Value:	BOOL:	Nonzero if successful; otherwise 0.

DES_CAN_SetModuleID

BOOL __stdcall DES_CAN_SetModuleID(char portName[], WORD dest, WORD moduleID);

Description:	Execute the DES RS232 command ‘SetModuleID’ (OpCode = 0x39). Set CAN Modul-ID (max. 11bit). The module ID is set by DIP switches at the system power up. During operation it’s possible to overwright temporary the module ID with the command ‘SetModuleID’. The module ID determines the ID’s for SDO communication (TxSDO ID = 1408 + ModuleID; RxSDO = 1536 + ModuleID).	
Parameter:	char portName[]: WORD dest: WORD moduleID:	Serial port (COM1, ...) Module ID of addressed DES New Module ID
Return Value:	BOOL:	Nonzero if successful; otherwise 0.

DES_CAN_SetTPDOID

BOOL __stdcall DES_CAN_SetTPDOID(char portName[], WORD dest, WORD transPDOID);

Description:	Execute the DES RS232 command ‘SetTPDOID’ (OpCode = 0x3B). Set CAN Transmit-PDO ID des CAN (max. 11bit).	
Parameter:	char portName[]: WORD dest: WORD transPDOID:	Serial port (COM1, ...) Module ID of addressed DES Transmit-PDO ID
Return Value:	BOOL:	Nonzero if successful; otherwise 0.

DES_CAN_SetRPDOID

BOOL __stdcall DES_CAN_SetRPDOID(char portName[], WORD dest, WORD receivePDOID);

Description:	Execute the DES RS232 command ‘SetRPDOID’ (OpCode = 0x3C). Set CAN Receive-PDO ID (max. 11bit).	
Parameter:	char portName[]: WORD dest: WORD receivePDOID:	Serial port (COM1, ...) Module ID of addressed DES Receive-PDO ID
Return Value:	BOOL:	Nonzero if successful; otherwise 0.

DES_CAN_ReadModuleID

BOOL __stdcall DES_CAN_ReadModuleID(char portName[], WORD dest, WORD* id);

Description:	Execute the DES RS232 command ‘ReadModuleID’ (OpCode = 0x3E). Read the ID from the DES.	
Parameter:	char portName[]: WORD dest: WORD* id:	Serial port (COM1, ...) Module ID of addressed DES Pointer to the ID of DES
Return Value:	BOOL:	Nonzero if successful; otherwise 0.

DES_CAN_SetCAN_BCR1_BCR2

BOOL __stdcall DES_CAN_SetCAN_BCR1_BCR2(char portName[], WORD dest, WORD bcr1, WORD bcr2);

Description:	Execute the DES RS232 command 'SetCAN_BCR1' (OpCode = 0x3F). Set CAN bit timing configuration register 1 and 2. For exact information see in the appendix (CAN Bit Timing). Only available since Software Version 0x1040 and higher!	
Parameter:	char portName[]: WORD dest: WORD bcr1: WORD bcr2:	Serial port (COM1, ...) Module ID of addressed DES CAN bit timing configuration register 1 CAN bit timing configuration register 2
Return Value:	BOOL:	Nonzero if successful; otherwise 0.

DES_CAN_SetCAN_Bitrate

BOOL __stdcall DES_CAN_SetCAN_Bitrate(char portName[], WORD dest, WORD index);

Description:	Execute the DES RS232 command 'SetCANBCR1' (OpCode = 0x40). Set CAN transfer rate to calculated values. See the section Bit Timing for more information about this configuration register. Only available since Software Version 0x1050 and higher!	
Parameter:	char portName[]: WORD dest: WORD index:	Serial port (COM1, ...) Module ID of addressed DES Index for transfer rate
Return Value:	BOOL:	Nonzero if successful; otherwise 0.

DES_CAN_ConfigPDO

BOOL __stdcall DES_CAN_ConfigPDO(char portName[], WORD dest, WORD action);

Description:	Execute the DES RS232 command 'ConfigPDO' (OpCode = 0x45). Switch on and off the PDO communication. The state of the PDO communication can be read with the system parameter 'CAN Config' (SysParam 41, Bit14). Only available since Software Version 0x1040 and higher!	
Parameter:	char portName[]: WORD dest: WORD action:	Serial port (COM1, ...) Module ID of addressed DES 0 = Switch Off, 1 = Switch On
Return Value:	BOOL:	Nonzero if successful; otherwise 0.

DES_CAN_SetRTRID

BOOL __stdcall DES_CAN_SetRTRID(char portName[], WORD dest, WORD rtrChannel, WORD rtrID);

Description:	Execute the DES RS232 command 'SetRTRID' (OpCode = 0x46). Set CAN Remote Request Frame ID (11 bit). There are two possible channels for remote request frames. Only available since Software Version 0x1040 and higher!	
Parameter:	char portName[]: WORD dest: WORD rtrChannel: WORD rtrID:	Serial port (COM1, ...) Module ID of addressed DES 0 for RTR channel 0, 1 for RTR channel 1 Remote Request ID
Return Value:	BOOL:	Nonzero if successful; otherwise 0.

DES_CAN_ConfigRTR

BOOL __stdcall DES_CAN_ConfigRTR(char portName[], WORD dest, WORD rtrChannel, WORD action);

Description:	Execute the DES RS232 command 'Config RTR' (OpCode = 0x47). Switch on and off the RTR communication. The state of the RTR communication can be read with the system parameter "CAN Config" (SysParam 41, Bit 13 = RTR0, Bit 14 = RTR1). Only available since Software Version 0x1040 and higher!	
Parameter:	char portName[]: WORD dest: WORD rtrChannel: WORD action:	Serial port (COM1, ...) Module ID of addressed DES Channel: 0 for RTR0, 1 for RTR1 0 = Switch Off; 1 = Switch On; 2 = Reset Parameters
Return Value:	BOOL:	Nonzero if successful; otherwise 0.

DES_CAN_AddRTRParameter

BOOL __stdcall DES_CAN_AddRTRParameter(char portName[], WORD dest, WORD paramSel, WORD param, WORD *ack);

Description:	Execute the DES RS232 command 'AddRTRParameter' (OpCode = 4A). Register a new RTR parameter for the remote request frame. Reset the RTR parameter for this channel before adding new parameters. The maximum number of parameters are 4 words (4 x 16-bit). If the data buffer is full, the command sends a negative acknowledge ('F' = 0x0046). Only available since Software Version 0x1040 and higher!	
Parameter:	char portName[]: WORD dest: WORD paramSel: WORD param: WORD *ack:	Serial port (COM1, ...) Module ID of addressed DES Bit0: select channel (0 for channel 0 or 1 for channel 1) Bit4: 0 to select paramMode, 1 to select addressMode. Bit8: 0 to select Word (16bit), 1 to select LWord (32bit) Number (paramMode) or adress (addrMode) of the system parameter. See the section system parameters 'O' (0x004F) = parameter available 'F' (0x0046) = parameter not available, Buffer is full
Return Value:	BOOL:	Nonzero if successful; otherwise 0.

DES_CAN_GetRTRParameter

BOOL __stdcall DES_CAN_GetRTRParameter(char portName[], WORD dest, WORD rtrChannel, WORD index, WORD *ack, WORD *format, WORD *param);

Description:	Execute the DES RS232 command 'GetRTRParameter' (OpCode = 4B). Read the registred RTR parameters. The maximum number of parameters is 4 words (4 x 16-bit). If there are less than 4 parameters registred, the command sends a negative acknowledge ('F' = 0x0046) for the parameters which are not available. Only available since Software Version 0x1040 and higher!	
Parameter:	char portName[]: WORD dest: WORD rtrChannel: WORD index: WORD *ack: WORD *format: WORD *param:	Serial port (COM1, ...) Module ID of addressed DES Channel: 0 for RTR0, 1 for RTR1 0,1,2,3 (if parameter is available) 'O' (0x004F) = parameter available 'F' (0x0046) = parameter not available Bit4: 0 = paramMode, 1 = addressMode Bit8: 0 = Word (16bit), 1 = LWord (32bit) Number or address (addressMode) of the registered system parameter. See section system parameter
Return Value:	BOOL:	Nonzero if successful; otherwise 0.

1.5. CAN Dialog Layer

1.5.1. CAN Status Dialogs

DES_CAN_ReadSysStatusDlg

BOOL __stdcall DES_CAN_ReadSysStatusDlg(char portName[], WORD dest);

Description:	Dialog to read the system status of the DES.	
Parameter:	char portName[]: WORD dest:	Serial port (COM1, ...) Module ID of addressed DES
Return Value:	BOOL:	Nonzero if successful; otherwise 0.

DES_CAN_ReadErrorDlg

BOOL __stdcall DES_CAN_ReadErrorDlg(char portName[], WORD dest);

Description:	Dialog to read a 16-bit value of system errors.	
Parameter:	char portName[]: WORD dest:	Serial port (COM1, ...) Module ID of addressed DES
Return Value:	BOOL:	Nonzero if successful; otherwise 0.

DES_CAN_EnableDlg

BOOL __stdcall DES_CAN_EnableDlg(char portName[], WORD dest);

Description:	Dialog to set the system into the enabled or disabled state.	
Parameter:	char portName[]: WORD dest:	Serial port (COM1, ...) Module ID of addressed DES
Return Value:	BOOL:	Nonzero if successful; otherwise 0.

1.5.2. CAN System Parameter Dialogs

DES_CAN_EditTempParamDlg

BOOL __stdcall DES_CAN_EditTempParamDlg(char portName[], WORD dest);

Description:	Dialog to read and write a new value to a temporary system parameter.	
Parameter:	char portName[]: WORD dest:	Serial port (COM1, ...) Module ID of addressed DES
Return Value:	BOOL:	Nonzero if successful; otherwise 0.

DES_CAN_EditAllTempParamDlg

BOOL __stdcall DES_CAN_EditAllTempParamDlg(char portName[], WORD dest);

Description:	Dialog to edit all temporary system parameters.	
Parameter:	char portName[]: WORD dest:	Serial port (COM1, ...) Module ID of addressed DES
Return Value:	BOOL:	Nonzero if successful; otherwise 0.

DES_CAN_ReadVersionDlg

BOOL __stdcall DES_CAN_ReadVersionDlg(char portName[], WORD dest);

Description:	Dialog to read the application and the hardware version from the DES.	
Parameter:	char portName[]: WORD dest:	Serial port (COM1, ...) Module ID of addressed DES
Return Value:	BOOL:	Nonzero if successful; otherwise 0.

1.5.3. CAN Setting Dialogs

DES_CAN_SetVelocityDlg

BOOL __stdcall DES_CAN_SetVelocityDlg(char portName[], WORD dest);

Description:	Dialog to set a new velocity of the rotor.	
Parameter:	char portName[]: WORD dest:	Serial port (COM1, ...) Module ID of addressed DES
Return Value:	BOOL:	Nonzero if successful; otherwise 0.

DES_CAN_SetCurrentDlg

BOOL __stdcall DES_CAN_SetCurrentDlg(char portName[], WORD dest);

Description:	Dialog to set a new current amplitude.	
Parameter:	char portName[]: WORD dest:	Serial port (COM1, ...) Module ID of addressed DES
Return Value:	BOOL:	Nonzero if successful; otherwise 0.

1.5.4. CAN Monitor Dialogs

DES_CAN_ReadVelocityIsMustDlg

BOOL __stdcall DES_CAN_ReadVelocityIsMustDlg(char portName[], WORD dest);

Description:	Dialog to show the effective and requested velocity of the motor.	
Parameter:	char portName[]: WORD dest:	Serial port (COM1, ...) Module ID of addressed DES
Return Value:	BOOL:	Nonzero if successful; otherwise 0.

DES_CAN_ReadCurrentIsMustDlg

BOOL __stdcall DES_CAN_ReadCurrentIsMustDlg(char portName[], WORD dest);

Description:	Dialog to show the effective and requested current components of the motor.	
Parameter:	char portName[]: WORD dest:	Serial port (COM1, ...) Module ID of addressed DES
Return Value:	BOOL:	Nonzero if successful; otherwise 0.

1.5.5. CAN Function Dialogs

DES_CAN_ResetCANErrorDlg

BOOL __stdcall DES_CAN_ResetCANErrorDlg(char portName[], WORD dest);

Beschreibung:	Dialog to reset CAN errors Function available since Software Version 0x1040 and higher!	
Parameter:	char portName[]: WORD dest:	Serial port (COM1, ...) Module ID of addressed DES
Rückgabewert:	BOOL:	Nonzero if successful; otherwise 0.

DES_CAN_ResetCANDlg

BOOL __stdcall DES_CAN_ResetCANDlg(char portName[], WORD dest);

Beschreibung:	Dialog to reset CAN communication. Function available since Software Version 0x1040 and higher!	
Parameter:	char portName[]: WORD dest:	Serial port (COM1, ...) Module ID of addressed DES
Rückgabewert:	BOOL:	Nonzero if successful; otherwise 0.

DES_CAN_SetModuleIDDlg

BOOL __stdcall DES_CAN_SetModuleIDDlg(char portName[], WORD dest);

Description:	Dialog to set CAN Module ID.	
Parameter:	char portName[]: WORD dest:	Serial port (COM1, ...) Module ID of addressed DES
Return Value:	BOOL:	Nonzero if successful; otherwise 0.

DES_CAN_SetTPDOIDDlg

BOOL __stdcall DES_CAN_SetTPDOIDDlg(char portName[], WORD dest);

Description:	Dialog to set Transmit PDO ID.	
Parameter:	char portName[]: WORD dest:	Serial port (COM1, ...) Module ID of addressed DES
Return Value:	BOOL:	Nonzero if successful; otherwise 0.

DES_CAN_SetRPDOIDDlg

BOOL __stdcall DES_CAN_SetRPDOIDDlg(char portName[], WORD dest);

Description:	Dialog to set Receive PDO ID.	
Parameter:	char portName[]: WORD dest:	Serial port (COM1, ...) Module ID of addressed DES
Return Value:	BOOL:	Nonzero if successful; otherwise 0.

DES_CAN_ReadModuleIDDlg

BOOL __stdcall DES_CAN_ReadModuleIDDlg(char portName[], WORD dest);

Description:	Dialog to read the Module ID.	
Parameter:	char portName[]: WORD dest:	Serial port (COM1, ...) Module ID of addressed DES
Return Value:	BOOL:	Nonzero if successful; otherwise 0.

DES_CAN_SetCAN_BCR1_BCR2Dlg

BOOL __stdcall DES_CAN_SetCAN_BCR1_BCR2Dlg(char portName[], WORD dest);

Description:	Dialog to set CAN bit timing configuration register 1 and 2. For exact information see in the appendix (CAN Bit Timing). Only available since Software Version 0x1040 and higher!	
Parameter:	char portName[]: WORD dest:	Serial port (COM1, ...) Module ID of addressed DES
Return Value:	BOOL:	Nonzero if successful; otherwise 0.

DES_CAN_SetCAN_BitrateDlg

BOOL __stdcall DES_CAN_SetCAN_BitrateDlg(char portName[], WORD dest);

Description:	Dialog to set CAN bit timing configuration registers BCR1 and BCR2. Function not available since Software Version 0x1050 and higher!	
Parameter:	char portName[]: WORD dest:	Serial port (COM1, ...) Module ID of addressed DES
Return Value:	BOOL:	Nonzero if successful; otherwise 0.

DES_CAN_ConfigPDODlg

BOOL __stdcall DES_CAN_ConfigPDODlg(char portName[], WORD dest);

Description:	Dialog to switch on or off PDO communication. Only available since Software Version 0x1040 and higher!	
Parameter:	char portName[]: WORD dest:	Serial port (COM1, ...) Module ID of addressed DES
Return Value:	BOOL:	Nonzero if successful; otherwise 0.

DES_CAN_SetRTRIDDlg

BOOL __stdcall DES_CAN_SetRTRIDDlg(char portName[], WORD dest);

Description:	Dialog to set Remote Request Frame ID of channel 0 or 1. Only available since Software Version 0x1040 and higher!	
Parameter:	char portName[]: WORD dest:	Serial port (COM1, ...) Module ID of addressed DES
Return Value:	BOOL:	Nonzero if successful; otherwise 0.

DES_CAN_ConfigRTRDlg

BOOL __stdcall DES_CAN_ConfigRTRDlg(char portName[], WORD dest);

Description:	Dialog to switch on and off the RTR communication. Only available since Software Version 0x1040 and higher!	
Parameter:	char portName[]: WORD dest:	Serial port (COM1, ...) Module ID of addressed DES
Return Value:	BOOL:	Nonzero if successful; otherwise 0.

DES_CAN_SetRTRID0Dlg

BOOL __stdcall DES_CAN_SetRTRID0Dlg(char portName[]);

Description:	Dialog to set Remote Request Frame ID of channel 0. Only available since Software Version 0x1040 and higher!	
Parameter:	char portName[]: WORD dest:	Serial port (COM1, ...) Module ID of addressed DES
Return Value:	BOOL:	Nonzero if successful; otherwise 0.

DES_CAN_SetRTRID1Dlg

BOOL __stdcall DES_CAN_SetRTRID1Dlg(char portName[]);

Description:	Dialog to set Remote Request Frame ID of channel 1. Only available since Software Version 0x1040 and higher!	
Parameter:	char portName[]: WORD dest:	Serial port (COM1, ...) Module ID of addressed DES
Return Value:	BOOL:	Nonzero if successful; otherwise 0.

DES_CAN_AddRTRParameterDlg

BOOL __stdcall DES_CAN_AddRTRParameterDlg(char portName[], WORD dest);

Description:	Dialog to register a new RTR parameter for the remote request frame. Only available since Software Version 0x1040 and higher!	
Parameter:	char portName[]: WORD dest:	Serial port (COM1, ...) Module ID of addressed DES
Return Value:	BOOL:	Nonzero if successful; otherwise 0.

DES_CAN_GetRTRParameterDlg

BOOL __stdcall DES_CAN_GetRTRParameterDlg(char portName[], WORD dest);

Description:	Dialog to read the registered RTR parameters. Only available since Software Version 0x1040 and higher!	
Parameter:	char portName[]: WORD dest:	Serial port (COM1, ...) Module ID of addressed DES
Return Value:	BOOL:	Nonzero if successful; otherwise 0.

2. Including Library Functions

The following chapter describes how to include all DES RS232 functions in your own windows program. The way how you do that, depends on the compiler and on the programming language you use. Thus we are going to explain the procedure to include the library based on some examples of the most popular programming languages.

2.1. General Information

In order to have a correctly working communication, you have to include the library to your

DesCmd.dll

programming environment. You have to copy this file to the working directory of your system.

To configure the library ‚DesCmd.dll' you have to select the serial port, the baudrate, the timeout and the trials for the communication. You have to do this before you can execute any DES RS232 command.
At the end you have to close the serial port.

For more detailed information about the initialisation procedure, have a look at the following chapter ‚Programming'.
Use the calling convention __**stdcall** for this library. This convention is managing how the parameters are put on the stack and who is responsible to clean the stack after the function execution.

2.2. Microsoft Visual C++

You need the following files to include the library to the programming environment of ‚Microsoft Visual C++‘.

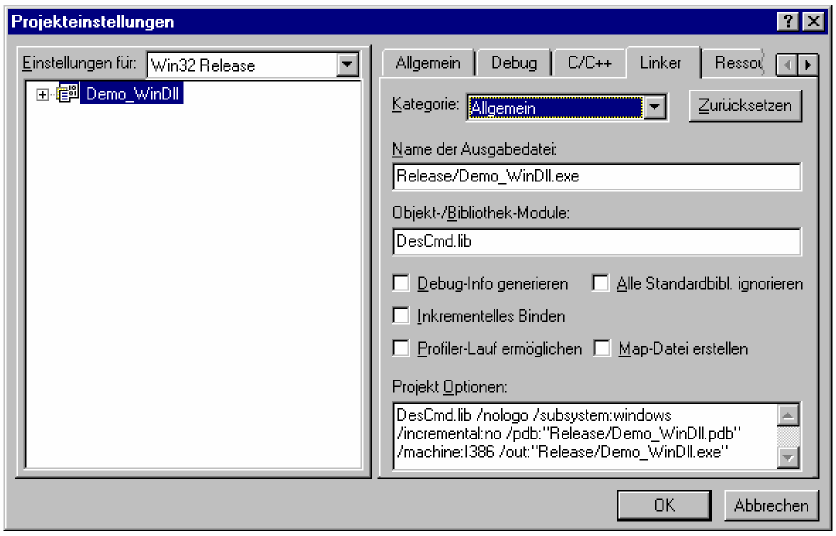
- **Def.h**: Constant definitions and declarations of the library functions
- **DesCmd.dll**: Dynamic link library
- **DesCmd.lib**: Import library (COFF Format)

The following steps are necessary to include the library correctly:

First Step: You have to copy all files to the working directory of the project.

Second Step: The header file ‘Def.h’ has to be included into the programm code. Use the instruction **‘ #include “Def.h” ‘**.

Third Step: The file ‘DesCmd.lib’ has to be added to the project. For that purpose you have to open the menu point ‘Settings’ of the menu ‘Project’. Select the tab ‘Linker’ and add the file ‘DesCmd.lib’ in the edit field Object-/Library modules. The figure 2.1 shows the german version of this dialog.


Projekteinstellungen

Einstellungen für: Win32 Release

Demo_WinDll

Kategorie: Allgemein

Name der Ausgabedatei: Release/Demo_WinDll.exe

Objekt-/Bibliothek-Module: DesCmd.lib

☐ Debug-Info generieren ☐ Alle Standardbibl. ignorieren

☐ Inkrementelles Binden

☐ Profiler-Lauf ermöglichen ☐ Map-Datei erstellen

Projekt Optionen:

```
DesCmd.lib /nologo /subsystem:windows
/incremental:no /pdb:"Release/Demo_WinDll.pdb"
/machine:I386 /out:"Release/Demo_WinDll.exe"
```

Figure 2.1: Project Settings Visual C++

After this three steps you can execute directly all library functions in your own code.

2.3. Microsoft Visual Basic

The programming environment of ‘Microsoft Visual Basic’ needs the following files to include the library functions to your program:

- **Def.bas**: Constant definitions and declarations of the library functions
- **DesCmd.dll**: Dynamic link library

The following steps are necessary to include the library correctly:

First Step: You have to copy all files to the working directory of the project.

Second Step: Include the program module ‘Def.bas’ to your project. For this purpose select the menu point ‘Add module’ of the menu ‘Project’. Change the view to the tab ‘Existing’ and select the file ‘Def.bas’. The figure 2.2 shows the german version of this dialog.

Third Step: Additionaly you have to copy the library ‘DesCmd.dll’ to the windows system directory or any other visible directory.

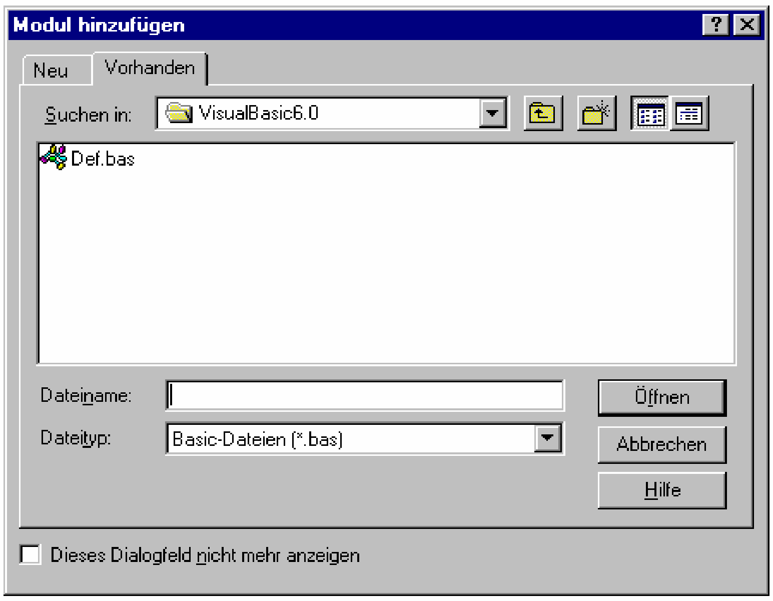

Modul hinzufügen
Neu
Vorhanden
Suchen in: VisualBasic6.0
Def.bas
Dateiname:
Dateityp: Basic-Dateien (*.bas)
Öffnen
Abbrechen
Hilfe
Dieses Dialogfeld nicht mehr anzeigen

Figure 2.2: Adding modules in Visual Basic

Now you can use all library functions directly in your own program code.

Attention!

Be careful managing the data types of the parameters. Visual Basic has no unsigned data types. There are different DES RS232 functions using unsigned variables. Be careful to map the signed data types correctly to the unsigned data types. Otherwise there will be a confusion of data. In the file ‘Def.bas’ there are comments hinting at this problem.

2.4. Borland C++ Builder

The following files are necessary to include the library functions to the programming environment of Borland C++ Builder:

• **Def.h**	Constant definitions and declarations of the library functions
• **DesCmd.dll**	Dynamic link library
• **DesCmd.lib**	Import library (OMF format)

The file ‚DesCmd.lib‘ has to be in the OMF format of Borland. This format differs from the COFF format of Microsoft Visual C++.

To include the listed files you have to do the following steps:

First Step:	The files have to be copied to a working directory of the project.
Second Step:	Write the instruction **‚ #include “Def.h“ ‘** to your program to include the constant definitions and the declarations of the library functions.
Third Step:	Add the file ‚DesCmd.dll‘ to the project.

After this three steps you can execute all the DES RS232 commands directly in your own program.

2.5. Borland Delphi

The following files are necessary to include the library functions to the programming environment of Borland Delphi:

• **Def.pas**:	Constant definitions and declarations of the library functions
• **DesCmd.dll**:	Dynamic link library

To include the listed files you have to do the following steps:

First Step:	The files have to be copied to a working directory of the project.
Second Step:	Write the instruction **‘Uses Def’** to your program to include the constant definitions and the declarations of the library functions.

After this two steps you can execute all the DES RS232 commands directly in your own program.

2.6. National Instruments LabView

Including the library into the programming environment of ‚LabView' is a little bit more complicated. LabView offers a function block to include extern library functions. You have to configure for each library function of the DES RS232 library a new function block. In these function blocks you configure the name, the parameter and the location of the library function.
For an easy start with LabView programming, most of the function blocks are already configured for you. For each DES RS232 command we have a SubVI block. These blocks can easily be copied to your own VIs. The provided library **(DesCmd.lib)** includes all function groups. The function group contains a number of SubVIs. Every SubVI is an DES RS232 command and can be connected directly.

The following files and directories have to be available to include the function blocks in your program:

Files:

• **DesCmd.dll**:	Dynamic link library	
• **DesCmd**.**llb**:	LabView library:	Initialisation.vi (Contains SubVIs)
		MonitorFunctions.vi (Contains SubVIs)
		Settings.vi (Contains SubVIs)
		SystemParameter.vi (Contains SubVIs)
		Status.vi (Contains SubVIs)
		Recording.vi (Contains SubVIs)
		CANFunctions.vi (Contains SubVIs)
		CANCommands.vi (Contains SubVIs)

Directories:

• **\DesCmd\ Initialisation\*.vi**	SubVIs of DesCmd.llb; Initialisation.vi
• **\DesCmd\ MonitorFunctions\*.vi**	SubVIs of DesCmd.llb; MonitorFunctions.vi
• **\DesCmd\ Settings\*.vi**	SubVIs of DesCmd.llb; Settings.vi
• **\DesCmd\ SystemParameter\*.vi**	SubVIs of DesCmd.llb; SystemParameter.vi
• **\DesCmd\ Status\*.vi**	SubVIs of DesCmd.llb; Status.vi
• **\DesCmd\ Recording\*.vi**	SubVIs of DesCmd.llb; Recording.vi
• **\DesCmd\ CANFunctions\*.vi**	SubVIs of DesCmd.llb; CANFunctions.vi
• **\DesCmd\ CANCommands\*.vi**	SubVIs of DesCmd.llb; CANCommands.vi

To include a SubVI into your own LabView program you have to do the following steps:

First Step: The listed files and directories have to be copied to a working directory of your program.

Second Step: Open the library ‘DesCmd.lib’. Choose the function group for your DES RS232 command you want include. The figure bellow shows a version of this dialog. Afterwards a front panel windows appears. This window is empty.

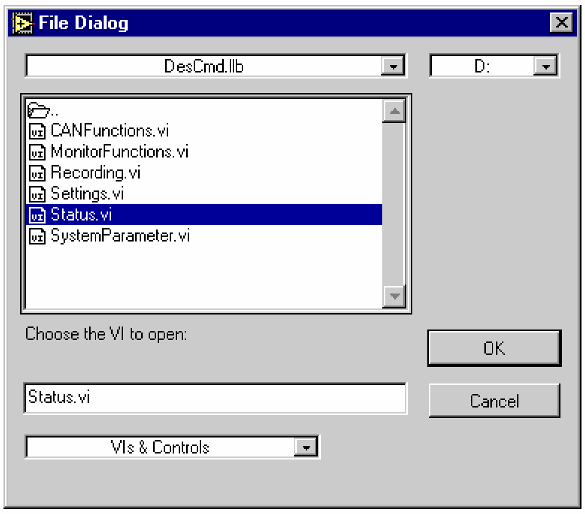


Figure 2.3: Open VI Libraries

Third Step: Select the menu point ‚Diagram‘ in the menu ‚Window‘ to open the diagram window. In this window you see all the predefined SubVIs containing the DES RS232 commands. Copy one of this blocks to include a new command to your program.

Fourth Step: Choose the tool ‘Connector’ to connect all the inputs and outputs of your new DES RS232 command block. If you don’t connect the inputs, the default values stored within the SubVI block are used.

3. Programming

The following chapter explains how the fundamental programming looks like. For several programming languages we deliver an example program. These programs are also explained in this chapter.

3.1. Fundamental Program Flow

To configure the communication with the DES correctly, you have to execute an initialisation function before the first communication command. The exact program flow looks like this:

Initialisation procedure
These function has to be executed at the beginning of the program.

Functions	Description
DES_InitCommunication(...,...,...,...)	Initialisation of the serial port; with the user data.

Communication with DES
Choose any of the DES RS232 commands.

Functions	Description
DES_SetVelocity(...,...)	Set the new velocity.
DES_ReadTempParam(...,...,...,...)	Read the temporary system parameter from DES.
etc.	

Closing procedure
Before closing the program you have to release the serial port.

Functions	Description
DES_CloseCommunication(...)	Release the serial port.

3.2. Microsoft Visual C++ 6.0 Example

The example ‘Demo_WinDll‘ in Visual C++ is a dialog based application. The application shows you, how the communication with the DES has to be configured. The communication is configured with the parameters COM1 and 38400 Baud. If this communication settings are changed you have to edit the source code.

The whole initialisation is programmed in the member function “OnInitDialog()“ of the class ‘Demo_WinDllDlg‘. The serial port is released at the end in the function “DestroyWindow()“.

Clicking on the buttons you can execute the DES RS232 commands “DES_Enable“, “DES_InitCommunicationDlg“ and “DES_SetVelocity“.

A timer is controlling a periodical update of the status. Every 200ms the function “UpdateStatus()“ is executed. If an error occurs during the update of the status the timer is stopped and it appears an error report.

If the example is used with an old Visual C++ version, you have to open a new project and include the files and resources to this new project.

3.3. Visual Basic 6.0 Example

The example in Visual Basic 6.0 has the same layout and behavior like the other examples. Starting the application the communication is set by default. The settings COM1 and 38400 Baud are configured.
During the cycle the status are written periodically and shows on the dialog. Clicking on a button you can execute one of the DES RS2323 commands “DES_InitCommunicationDlg“, “DES_Enable“ and “DES_SetVelocity“. A timer is polling the status of the DES. Is there an error during the communication the timer is stopped.

The library 'DesCmd.dll' is searching in order of working directory, windows directory and search path. If the library wasn’t found, the library has to copy in one of this directories.

If there are any problems with different versions of the programming environment, you have to open a new project with your version of the programming environment. Use the example program files as a help.

3.4. Borland C++ Builder 5.0 Example

The example program in the programming environment of Borland C++ Builder has the same layout and behavior like the other examples. Starting the application the communication is set by default. The settings ‚COM1‘ and ‚38400 Baud‘ are configured. If you want to change this settings you have edit the source code and recompile the application.
To push the button the DES is enabled or disabled. After this the motor can be activated or stopped.

The library “DesCmd.lib“ is not the same which is constructed from Microsoft Visual C++. The format has to be changed.

If there are any problems with different versions of the programming environment you can use this project as a help to program your own project.

3.5. Borland Delphi 4.0 Example

The example program of Borland Delphi environment has the same layout and behavior like the other examples. Starting the application the communication is set by default. The settings ‚COM1‘ and ‚38400 Baud‘ are configured. If you want to change these settings you have edit the source code and recompile the application.
Clicking on the buttons you can execute the DES RS232 commands “DES_Enable“, “DES_InitCommunicationDlg“ or “DES_SetVelocity“. A timer is polling the status of the axis. Is there an error during the communication the timer is stopped.

If there any problems with different versions of the programming environment you can use this project as a help to program your own project.

3.6. National Instruments LabView 6.0 Example

The demo program of LabView shows you how you can include the various function blocks to your own program
Using a sequence structure you can assure that the initialisation commands are executed at the beginning of the program and the closing function is executed at the end of the program.
During the program there is a loop which is running until the user switches off the program using the switch “On / Off“.
In this loop the DES can be enabled or disabled. The motor is running with a fixed velocity or the motor can be stopped.
The following files belong to the example program:

- \LabView6.0\DesDemoDll.vi
- \LabView6.0\DesCmd.dll
- \LabView6.0\InitCommunication.vi
- \LabView6.0\Enable.vi
- \LabView6.0\SetTempParam.vi
- \LabView6.0\ReadTempParam.vi
- \LabView6.0\SetVelocity.vi
- \LabView6.0\ReadSysStatus.vi
- \LabView6.0\CloseCommunication.vi

For a correct communication you have to connect the inputs of the function “DES_InitCommunication“ or you have to set the values correctly within the function block.

4. Appendix

4.1. DES System Parameters

Nb	Parameter	Length	Access	Default	Range	Unit
0	Baudrate	16-bit	Read/Write	3	9600, 14400, 19200, 38400, 57600,115200 range: 0 ... 5	
1	SysConfig	16-bit	Read/Write	1	See section data structures	
2	Current regulation P-gain	16-bit	Read/Write	3057	0 ... 32767	
3	Current regulation I-gain	16-bit	Read/Write	994	0 ... 32767	
4	Max. output of current regulator	16-bit	Read/ServiceWrite	32512	0 ... 32767	
5	Speed regulator P-gain	16-bit	Read/Write	682	0 ... 32767	
6	Speed regulator I-gain	16-bit	Read/Write	220	0 ... 32767	
7	InternalParam1 (do not change)	16-bit	Read/Write	2200	do not change	
8	InternalParam2 (do not change)	16-bit	Read/Write	729	do not change	
9	InternalParam3 (do not change)	16-bit	Read/Write	13640	do not change	
10	Limitation of speed error for the input of speed regulator	16-bit	Read/ServiceWrite	32767	0 ... 32767	
11	Gain of setting unit	16-bit	Read	24576	0 ... 32767	
12	Offset of setting unit	16-bit	Read/Write	0	-100 ... +100	
13	Delay of setting unit (not used)	16-bit	Read/ServiceWrite	0	0 ... 32767	
14	Peak current	16-bit	Read/Write	15000	1 ... 15000	mA
15	Max continuous current	16-bit	Read/Write	5000	1 ... 5000	mA
16	Thermal constant	16-bit	Read/ServiceWrite	30400	0 ... 32767	
17	Max. speed	16-bit	Read/Write	25000	0 ... 25000	rpm
18	Acceleration	16-bit	Read/Write	32000	0 ... 32767	(rpm/128ms)
19	Speed constant (not used)	16-bit	Read/ServiceWrite	0	0 ... 32767	rpm/V
20	Encoder resolution	16-bit	Read/Write	500	0 ... 32767	pulse/turn
21	Pole-pair number	16-bit	Read/Write	1	1 ... 64 Standard 1 pole pair Flat motors x pole pair	

Nb	Parameter	Length	Access	Default	Range	Unit
22	InternalParam4 (do not change)	16-bit	Read/ServiceWrite	960	do not change	
23	Factor of conversion: rpm to qc/ms	16-bit	Read/ServiceWrite	2185	0 ... 32767	qc/(65535*rpm*ms)
24	Angular offset of index pulse	16-bit	Read/ServiceWrite	0	-32768 ... 32767	qc
25	PWM period	16-bit	Read	400	0-32768 ... 32767	clock
26	Max. duty cycle	16-bit	Read/ServiceWrite	7373	0 ... 32767	
27	Offset of phase u current detection	16-bit	Read/ServiceWrite	32512	0 ... 65535	
28	Offset of phase v current detection	16-bit	Read/ServiceWrite	32512	0 ... 65535	
29	Offset of general AD converter	16-bit	Read/ServiceWrite	32768	0 ... 65535	
30	CAN module ID	16-bit	Read	1	1 ... 127	
31	CAN service ID (= CAN module ID)	16-bit	Read	1	1 ... 127	
32	CAN RxPDO ID	16-bit	Read	513	385 ... 1407	
33	CAN TxPDO ID	16-bit	Read	385	385 ... 1407	
34	CAN BCR1	16-bit	Read	1578	0 ... 65535	
35	CAN BCR2	16-bit	Read	1	0 ... 65535	
36	CAN operation mode (not used)	16-bit	Read/ServiceWrite	0	0 ... 65535	
37	CAN RxSDO ID	16-bit	Read	1537	1537 ... 1663 ID = 1536 + moduleID	
38	CAN TxSDO ID	16-bit	Read	1409	1409 ... 1535 ID = 1408 + moduleID	
39	CAN RTR0 ID	16-bit	Read	386	385 ... 1407	
40	CAN RTR1 ID	16-bit	Read	387	385 ... 1407	
41	CAN Config	16-bit	Read	0	See section data struct	
42	InternalParam5	16-bit	Read	0	do not change	
43	ErrorProc	16-bit	Read/Write	0	0: Disable 1: Stop	
44	MaxSpeed in Current Reg Mode	16-bit	Read/Write	30000	0 ... 32767	rpm
45	HallAngleOffs	16-bit	Read/ServiceWrite	0	-32768 ... 32767	qc
46	MaxAngleMpy1	16-bit	Lesen/Service Schreiben	0xFFFF	0 ... 0xFFFF	qc
47	MaxAngleMpyN	16-bit	Lesen/Service Schreiben	0xFFFF	0 ... 0xFFFF	qc

Note:

Read	= the parameter value can be read
Write	= the user has write access to the parameter
ServiceWrite	= the user has write access only if the service mode was set (see command *Service*)

4.2. DES Status Variables

Nb	Variable	Length	Unit
128	System operating status	16-bit	See section 'Data structures'
129	Effective current detected value in d-axis	16-bit	mA
130	Effective current detected value in q-axis	16-bit	mA
131	Current setting value	16-bit	mA
132	Relative rotor position in a revolution	16-bit	qc
133	Speed setting value	16-bit	rpm
134	Actual mean speed value	16-bit	rpm
135	reserved		
136	Absolute rotor position	32-bit	qc
137	Standard Error	16-bit	See section 'Standard Error Messages'
138	CAN Error	16-bit	See section 'CAN Error Messages'
139	Actual current value in q-axis (=> Torque) (not averaged)	16-bit	mA
140	Actual speed value (not averaged)	16-bit	rpm
141	Error History 1	16-bit	See section 'Standard Error Messages'
142	Error History 2	16-bit	See section 'Standard Error Messages'
143	Encoder Counter	16-bit	qc
144	Encoder Counter at last index	16-bit	qc
145	Hall sensor pattern	16-bit	See section 'Data structures'

4.3. Data Type Definitions

Name	Datatype	Bits	Byte	Area
char	unsigned Integer	8	1	-128 ... 127
BYTE	unsigned Integer	8	1	0 ... 256
short	signed Integer	16	2	-32'768 ... 32'767
WORD	unsigned Integer	16	2	0 ... 65'535
long	signed Integer	32	4	-2'147'483'648 ... 2'147'483'647
DWORD	unsigned Integer	32	4	0 ... 4'294'967'295

4.4. Data Structures

The most important data structures are explained in the following section. The library is written with the programming language „C++“. Thus the following definitions of the data structures are also written with the syntax of „C++“.

T_ErrHandler

- typedef void (*T_ErrHandler)(BYTE, BYTE, __int16, BOOL);

- BOOL InitDesCommander(T_ErrHandler yourHandler);

4.4.1. Definition of 'DES_SysParam'

```
typedef struct DES_SysParam
{
        short Baudrate;             //ParNb 0:   R/W; 0=9600; 1=14400; 2=19200; 3=38400;
                                                      4=57600; 5=115200 baud
        short SysConfig;            //ParNb 1;   R/W; System Configuration (see bit definition)
        short CurRegGainP;          //ParNb 2;   R/W; Current regulation P-gain
        short CurRegGainI;          //ParNb 3;   R/W; Current regulation I-gain
        short MaxCurOutput;         //ParNb 4;   R/W; Max output of current regulator
        short SpeedRegGainP;        //ParNb 5;   R/W; Speed regulator P-gain
        short SpeedRegGainI;        //ParNb 6;   R/W; Speed regulator I-gain
        short InternalParam1;       //ParNb 7;   R/W; Internally used, do not change
        short InternalParam2;       //ParNb 8;   R/W; Internally used, do not change
        short InternalParam3;       //ParNb 9;   R/W; Internally used, do not change
        short MaxSpeedError;        //ParNb 10;  R/W; Limitation of speed error for the input of
                                                      the speed regulator
        short SettingUnitGain;      //ParNb 11;  R/W; Gain of setting unit
        short SettingUnitOffset;    //ParNb 12;  R/W; Offset of setting unit
        short SettingUnitDelay;     //ParNb 13;  R/W; Delay of setting unit
        short PeakCurrent;          //ParNb 14;  R/W; Peak current in mA
        short MaxContCurrent;       //ParNb 15;  R/W; Maximum continuous current
        short ThermConst;           //ParNb 16;  R/W; Thermal constant
        short MaxSpeed;             //ParNb 17;  R/W; Maximum speed
        short Acceleration;         //ParNb 18;  R/W; Acceleration in rpm/ms2
        short SpeedConstant;        //ParNb 19;  R/W; Speed constant of motor
        short EncResolution;        //ParNb 20;  R/W; Encoder resolution in counts/turn
        short PolePairNumber;       //ParNb 21;  R/W; Number of pole pair
        short Qc2RpmFactor;         //ParNb 22;  R/W; Conversion factor from qc to rpm
        short Rpm2QcFactor;         //ParNb 23;  R/W; Conversion factor form rpm to qc
        short IndexOffset;          //ParNb 24;  R/W; Angular offset of index pulse
        short PWM_Period;           //ParNb 25;  R;   PWM period in clock
        short MaxDutyCycle;         //ParNb 26;  R/W; Max duty cycle
        short CurDetPhUOffset;      //ParNb 27;  R/W; Offset of (phase U) current detection
        short CurDetPhVOffset;      //ParNb 28;  R/W; Offset of (phase V) current detection
        short ADConvOffset;         //ParNb 29;  R/W; Offset of general AD converter
        short CAN_ModuleID;         //ParNb 30;  R;   CAN module ID
        short CAN_ServiceID;        //ParNb 31;  R;   CAN service ID
        short CAN_RPDO_ID;          //ParNb 32;  R/W; CAN RPDO ID
        short CAN_TPDO_ID;          //ParNb 33;  R;   CAN TPDO ID
        short CAN_BCR1;             //ParNb 34;  R;   CAN BCR1
        short CAN_BCR2;             //ParNb 35;  R;   CAN BCR2
        short CAN_OpMode;           //ParNb 36;  R/W; CAN operation mode
        short CAN_RxSDO_ID;         //ParNb 37;  R;   CAN Receive SDO ID = 1536 + moduleID
        short CAN_TxSDO_ID;         //ParNb 38;  R;   CAN Transmit SDO ID = 1408 + moduleID
        short CAN_RTR0_ID;          //ParNb 39;  R;   RTR ID (channel 0)
        short CAN_RTR1_ID;          //ParNb 40;  R;   RTR ID (channel 1)
        short CAN_Config;           //ParNb 41;  R;   CAN communication configuration register
        short InternalParam;        //ParNb 42;  R;   Internally used, do not change
        short ErrorProc             //ParNb 43;  RW;  Error Reaction Procedure
        short MaxSpeedCurr          //ParNb 44;  RW;  Maximal speed in current regulation mode
        short HallAngleOffs         //ParNb 45;  R;   Angular offset of hall sensor signals
        short MaxAngleMpy1          //ParNb 46;  RW;  Max Angle Mpy1
        short MaxAngleMpy1          //ParNb 47;  RW;  Max Angle MpyN
}DES_SysParam;
```

4.4.2. Definition of 'DES SysConfig'

BIT 0:	0: speed/current setting by software
	1: speed/current setting by analog input 'Set value'
BIT 1:	0: acceleration enabled
	1: acceleration disabled
BIT 2:	0: depending on BIT3
	1: current regulator
BIT 3:	0: speed regulator
	1: reserved
BIT 4:	0: speed monitor signal
	1: torque setting monitor signal
BIT 5:	do not change
BIT 6:	not used
BIT 7:	0: stop motor by digital input 'STOP'
	1: stop motor by software (command 'StopMotion')
BIT 8:	0: set max. speed by potentiometer 'P1'
	1: set max. speed by software (system parameter no.17)
BIT 9:	0: set offset by potentiometer 'P2'
	1: set offset by software (system parameter no. 12)
BIT 10:	0: set maximum current by potentiometer 'P3'
	1: set maximum current by software (system parameter no. 14 & 15)
BIT 11:	0: set the regulation gains (speed regulator) by potentiometer 'P4'
	1: set the regulation gains by software (system parameter no. 5 & 6)
BIT 12:	0: enable system by digital input 'Enable'
	1: enable system by software (command 'Enable')
BIT 13:	0: select monitor signal by digital input 'Digital 1'
	1: select monitor signal by BIT4
BIT 14:	0: the addressed parameters and variables are not allowed to be written (no service)
	1: the addressed parameters and variables are allowed to be written (service mode)
BIT 15:	0: select regulation mode by digital input 'Digital 2'
	1: select regulation mode by BIT2, BIT3

4.4.3. Configuration of Regulation Mode

Current Regulation Mode:	SysConfig.Bit2 = 1;	SysConfig.Bit3 = 0;
Speed Regulation Mode:	SysConfig.Bit2 = 0;	SysConfig.Bit3 = 0;

4.4.4. Definition of Hall sensor pattern

HallSensorPattern.bit0 :	State of Hall Sensor 1
HallSensorPattern.bit1 :	State of Hall Sensor 2
HallSensorPattern.bit2 :	State of Hall Sensor 3

4.5. Status Flags

4.5.1. Definition of 'CAN Error Message'

b0:	CAN Error 0	Warning Status
b1:	CAN Error 1	Error Passive Status
b2:	CAN Error 2	Bus Off Status
b3:	CAN Error 3	Acknowledge Error
b4:	CAN Error 4	Stuff Error
b5:	CAN Error 5	CRC Error
b6:	CAN Error 6	Stuck at dominant Error
b7:	CAN Error 7	Bit Error Flag
b8:	CAN Error 8	Form Error Flag
b9:	CAN Error 9	PDO Accessing frequency is too high
b10:	CAN Error 10	PDO Overflow
b11:	CAN Error 11	TxPDO No Acknowledge
b12:	CAN Error 12	TxSDO No Acknowledge
b13:	CAN Error 13	RxPDO Message Lost
b14:	CAN Error 14	RxSDO Message Lost
b15:	CAN Error 15	0 = Transmission successful; 1 = Transmission failed

4.5.2. Definition of 'System Operating Status'

BIT 0:	0: encoder index not found yet
	1: encoder index found
BIT 1:	0: hall sensor signal not found yet
	1: hall sensor signal found
BIT 2:	0: rotor position not found yet
	1: rotor position found
BIT 3:	0: not saving the system parameters in EEPROM
	1: saving the system parameters in EEPROM
BIT 4:	0: Second hall sensor signal not found yet
	1: Second hall sensor signal found
BIT 5:	0: measure Vmax/Offset
	1: measure Temperature
BIT 6:	0: ±10V SetValue
	1: 0 ... 5V SetValue
BIT 7:	0: Max current set to peak current
	1: Max. current reduced to continuous current
BIT 8:	0: in the small current region
	1: in the large current region
BIT 9:	0: no error
	1: error
BIT 10:	0: software disabled
	1: software enabled
BIT 11:	0: not debouncing the enable input, the input can be changed
	1: debouncing the enable input, the input has to be stable
BIT 12:	0: no offset in current circuit detected
	1: offsets in current circuit detected
BIT 13:	0: not braking
	1: braking with the maximum setting current
BIT 14 + 15:	0 + 0: power stage is disabled
	0 + 1: refresh the power stage
	1 + 0: power stage is enabled
	1 + 1: power stage is enabled

(BIT 5 and BIT 6:) Only Hardware Version 4003h!

4.5.3. Definition of 'Standard Error Message'

b0:	Error 0	Hall Sensor Error
b1:	Error 1	Index Processing Error
b2:	Error 2	Wrong setting of encoder resolution
b3:	Error 3	Hall Sensor 3 not found
b4:	Error 4	Over Current Error
b5:	Error 5	Over Voltage Error
b6:	Error 6	Over Speed Error
b7:	Error 7	Supply voltage too low for operation
b8:	Error 8	Angle detection error
b9:		
b10:		
b11:	Error 11	Over temperature error
b12:		
b13:	Error 13	Parameter out of range
b14:		
b15:	Error 15	0 = no errors; 1 = there are errors

4.5.4. Definition of 'CAN Config'

BIT 14:	0: PDO channel disabled
	1: PDO channel enabled
BIT 13:	0: Remote Transmission Request Channel 1 disabled
	1: Remote Transmission Request Channel 1 enabled
BIT 12:	0: Remote Transmission Request Channel 0 disabled
	1: Remote Transmission Request Channel 0 enabled

4.5.5. Definition of 'ErrorProc'

Definition of the error reaction. Only the specified errors can be configured. All other errors disable the drive.

ErrorProc = 0:	Disable DES on error
ErrorProc = 1:	Stop DES on error

Configurable errors:	Error 7: Supply voltage too low for operation
	Error 8: Angle Detection Error

4.6. CAN Bit Timing (BCR1 & BCR2)

The DES is configured to work optimally at the maximal bit rate of 1Mbit/s. The bit timing parameters, like nominal sampling point and time quanta, are chosen to be very close to the CiA recommendations for CAN or CANopen devices.

If you want to change the bit timing you have to adjust the registers 'BCR1' and 'BCR2'. Use the function 'SetCANBCR' or the function 'SetCANBitrate'.

Here is some information for calculating these two register values.

f_{Osc}	= 4 * Quartz Frequence
TQ	= (BRP + 1) / f_{Osc}
$t_{SYNCSEG}$	= SYNCSEG * TQ
T_{TSEG1}	= (TSEG1 + 1) * TQ
T_{TSEG2}	= (TSEG2 + 1) * TQ
Bit Time	= $t_{SYNCSEG} + T_{TSEG1} + T_{TSEG2}$

SYNCSEG	= 1
SJW	= 0 - 3
TSEG1	= 2 - 15
TSEG2	= 1 - 7

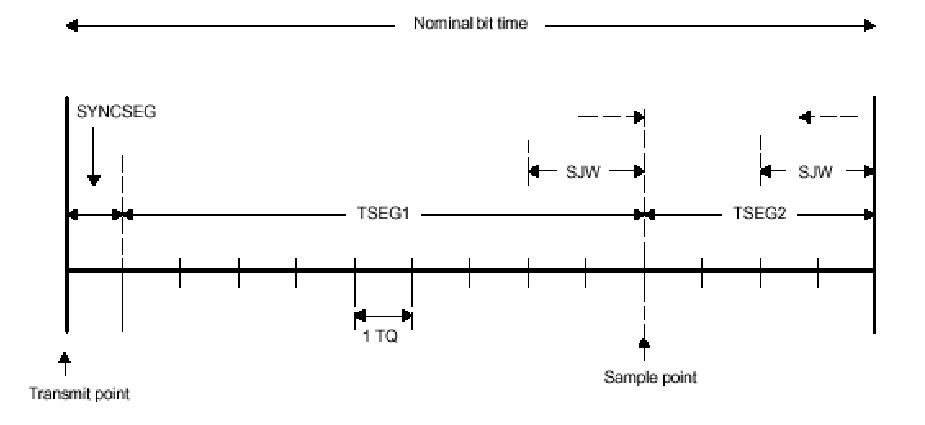


Figure 4.1: Bit Timing Calculation

Definitions:

$T_{SEG1} \geq T_{SEG2}$

$T_{SEG2min} = 1 + SJW$

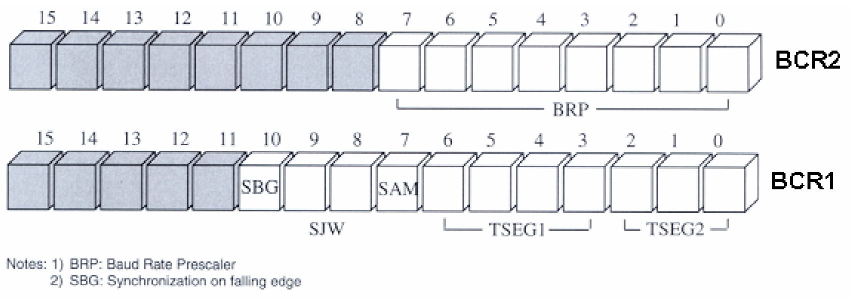


Figure 4.2: Bit Timing Register

BRP	= Baudrate Prescaler
SBG	= 0 (Synchronisation on falling edge); 1(Synchronisation on rising edge)
SJW	= Synchronisation jump width
SAM	= 0 (CAN module samples only once); 1(CAN module samples three times and makes a majority decision
TSEG1	= Time segment 1
TSEG2	= Time segment 2

Calculation Example for 500kBit/s:

Quartz frequence	= 5 MHz (Hardware Version 0x4001, 0x4002 and 0x4101)
f_{Osc}	= 4 * Quartz frequence = 20MHz
Bitrate	= 500 kBit/s
Nominal Bit Time	= 1/Bitrate = 2µs
Number of time quanta	= 20
Nominal TQ	= 100ns

BRP (= BCR2)	= (Nominal TQ * f_{Osc}) – 1 = 1
TQ	= (BRP + 1) / f_{Osc} = 100ns
$t_{SYNCSEG}$	= 1 * TQ = 100ns
TSEG1	= 15
TSEG2	= 2
T_{TSEG1}	= (TSEG1 + 1) * TQ = 1.6µs
T_{TSEG2}	= (TSEG2 + 1) * TQ = 300ns
Bit Time	= $t_{SYNCSEG}$ + T_{TSEG1} + T_{TSEG2} = 2µs
SJW	= 1
SAM	= 0
SBG	= 0

BCR1 = 017Ah; BCR2 = 0001h

BCR1 and BCR2 recommendations

There are some bit timing values already calculated. Take these values only as a reference. These values have to be adjusted for your own CAN network.

Table for Quartz with 10MHz Quartz frequency (HW 0x4003):

Bitrate	1MBit/s	800kBit/s	500kBit/s	250kBit/s	125kBit/s	50kBit/s	20kbit/s	10kbit/s
Max Line Length [m]	25	50	100	250	500	1000	2500	5000
BCR1 (hexadecimal)	0h017A	0h0031	0h017A	0h017A	0h017A	0h0173	0h0173	0h0173
BCR2 (hexadecimal)	0h0001	0h0004	0h0003	0h0007	0h000F	0h0027	0h0063	0h00C7

Table for Quartz with 5MHz Quartz frequency (HW 0x4001, 0x4002 and 0x4101):

Bitrate	1MBit/s	800kBit/s	500kBit/s	250kBit/s	125kBit/s	50kBit/s	20kbit/s	10kbit/s
Max Line Length [m]	25	50	100	250	500	1000	2500	5000
BCR1 (hexadecimal)	0h017A	0h017F	0h017A	0h017A	0h017A	0h0173	0h0173	0h0173
BCR2 (hexadecimal)	0h0000	0h0000	0h0001	0h0003	0h0007	0h0013	0h0031	0h0063

Table for Quartz with 4.9152MHz Quartz frequency (HW 0x0001):

Bitrate	1MBit/s	800kBit/s	500kBit/s	250kBit/s	125kBit/s	50kBit/s	20kbit/s	10kbit/s
Max Line Length [m]	25	50	100	250	500	1000	2500	5000
BCR1 (hexadecimal)	0h062A	0h0633	0h063B	0h063B	0h063B	0h0643	0h063C	0h0652
BCR2 (hexadecimal)	0h0001	0h0001	0h0002	0h0005	0h000B	0h001B	0h0045	0h0082
Effective Bitrate [Bit/s]	983040	819200	504123	252062	126030	50155	20062	10005