maxon motor

maxon motor control 4-Q-EC Servoamplifier DES 70/10

Order number 228597

Communication Guide Edition April 2007

PROPRIETARY NOTICE
Communication Guide

All rights reserved by maxon motor ag.
All instructions, information and specifications contained in this manual are for reference only and remain subject to change without announcement.

CH-Sachseln, 23.01.2007

The latest edition of this Communication Guide may also be found on the internet site http://www.maxonmotor.com (see category «Service & Downloads»).

TABLE OF CONTENTS

1 INTRODUCTION 5

2 FIRMWARE UPDATE IN GRAPHICAL USER INTERFACE GUI 6

3 COMMAND OPERATION 7

3.1 Basic Instruction Format 7
3.2 DES Command Reference 8
3.2.1 Status Functions 8
3.2.2 Service Functions (reserved to advanced users) 10
3.2.3 System Parameter Functions 11
3.2.4 Setting Functions 13
3.2.5 Monitor Functions 13
3.2.6 Data Recording Functions 14
3.2.7 CAN Bus Configuration Functions 15
3.3 DES System Parameters 19
3.4 DES Status Variables 21
3.5 Data Type Definitions 21
3.6 Data Structures 22
3.7 Standard Error Messages 25
3.8 CAN Error Messages 28

4 SERIAL EIA-RS232 COMMUNICATION 32

4.1 Physical Layer 32
4.1.1 Electrical Standard 32
4.1.2 Medium 32
4.2 Data Link Layer 32
4.2.1 Data Format 32
4.2.2 Frame Structure 33
4.2.3 Transmission Byte Order 34
4.2.4 Command Instruction Example 34
4.2.5 Protocol and Flow Control 35

5 CAN BUS COMMUNICATION 38

5.1 Physical Layer 38
5.1.1 Bit Timing 38
5.1.2 Physical Medium Attachment 40
5.1.3 Medium Dependant Interface 40
5.2 Data Link Layer 41
5.2.1 Standard CAN Data Frame 41
5.2.2 CAN Data Frame 41
5.2.3 CAN Remote Transmission Request Frame 41
5.3 Application Layer 42
5.3.1 Communication Channels 42
5.3.2 SDO Communication (Service Data Objects) 43
5.3.3 PDO Communication (Process Data Objects) 45
5.3.4 RTR Communication (Remote Transmission Request) 47
5.4 Using a DES in a CAN Network via RS232 49
5.4.1 Configuration 49
5.4.2 Control of CAN using the Serial Interface 49
5.4.3 Command Instruction Example 50

FIRMWARE VERSION HISTORY

Date	Version	Documentation	Description
14.05.2001	SW: 1030h HW: 4101h	Software Guide Edition May 2001	• First Firmware Version
15.09.2001	SW: 1040h HW: 4101h	Communication Guide Edition September 2001	• New CAN Communication. (see chapter CAN Bus Communication) ⇨ SW: 1040h • New PDO and RTR communication. • Change of speed regulation structure. The regulation gains have to be readjusted. • New error display mode with LED. The red LED shines continually in the event of error condition. The number of flashing pulses of the green LED shows the number of the error state. (see manual) • New commands: ConfigPDO (0x45), SetRTRID (0x46), ConfigRTR (0x47), AddRTRParameter (0x4A), GetRTRParameter (0x4B), ResetCANError (0x06), ResetCAN (0x07). • Changed commands: ReadVersion (0x1A), ReadVelocityIsMust (0x28), ReadCurrentIsMust (0x29), RecordData (0x31) and SetCANBCR (0x3F). • New system parameters: RxSDO ID (No.37), TxSDO ID (No.38), RTR0 ID (No.39), RTR1 ID (No.40), CAN Config (No. 41) • Changed system parameters: ServiceID (No.31), Factor qc/ms to rpm (No.22). • New status parameters: Absolute rotor position (No.136), Standard Error (No.137), CAN Error (No.138), Actual current value (not averaged) (No.139). Actual speed value (not averaged) (No.140) • New CAN Errors: CAN Error 09: PDO Accessing frequency too high CAN Error 10: PDO Overflow CAN Error 11: TxPDO No Ack CAN Error 12: TxSDO No Ack CAN Error 13: RxPDO Message Lost CAN Error 14: RxSDO Message Lost
17.06.2002	SW: 1041h HW: 4101h	Communication Guide Edition June 2002	• CAN Communication Bug corrected: The reset command during the enable state made the whole CAN communication crashing. • Improved current offset measurement. Correct current offset measurement and calculation for phase V
21.03.2003	SW: 1050h HW: 4101h	Communication Guide Edition March 2003	• New commands SysParSetDefault (0x1B) SetCANBitRate (0x40) • New system parameters No.43 Error Proc: Error reaction procedure No.44 MaxSpeedCur: Max. speed in current mode • New status parameters No.141 Error History 1 No.142 Error History 2

			No.143 Encoder Counter No.144 Encoder Counter at last index No.145 Hall Sensor Pattern: Actual state of hall sensors (HS3, HS2, HS1) • Changed status parameters No.128 System operating status: New definition of bits 5 and 9 No.137 Standard Error: New definition of bit 8, bit 14 unused • Extended Error Management - Error 8 new: Angle Detection Error - Error 11 new: Over temperature Error - Error 14 removed: On 'Flash code error' both LEDs are on now • GUI Diagnostic Wizard compatibility • Extended current offset measurement solves torque asymmetry • Bug fixes - Update of current limits (Imax) corrected - Range check of paramNb at commands 'ReadTempParam', 'SetTempParam' corrected - Status parameter No. 136 'Absolute rotor position' over-/underflows now (instead of bounds) - CAN parameters set to default corrected - RTR values update within 0.1ms now **Note:** The DES_UserInterface 1.10 or newer should be used for parameter setting and firmware downloading!
13.05.2004	SW: 1050h HW: 4102h	Communication Guide Edition May 2004	New Hardware HW 4102h: • New low voltage DSP • New 'SetValue' circuit • Extended monitor output
23.01.2007	SW: 1051h HW: 4102h	Communication Guide Edition January/April 2007	New Software SW 1051h: • Features - support of new hardware revision HW4004 - Overspeed-Error in current mode replaced by speed limitation functionality - Hall angle supervision algorithm improved - CAN bus error handling improved • Bug fixes - Time for reset-sequence reduced - Peak current limitation corrected - Space Vector PWM improved for small currents - Range Check within Command SetAllTempParam **Note:** The DES_UserInterface **1.15** or newer should be used for parameter setting and firmware downloading!
Note: For further information have a look at the DES_UserInterface (menu 'Help', menu item Firmware 'ReadMe').			

1 Introduction

The DES servoamplifier is equipped with a serial EIA-RS232 interface and a CAN ISO/DIS 11898 compliant interface. Both physical interfaces are internally supported by a fast software interface implemented in the firmware, which offers services for RS232 and CAN. The internal services enable configuration and monitoring of a DES by means of command instructions.

The serial interface communication features are mainly intended to enable firmware upgrade, system configuration and device monitoring. The serial protocol is based on a single ended and unbalanced data transmission, i.e. only point-to-point connections are allowed. To enhance data reliability in the unbalanced line the protocol makes use of CRC-checking.

The CAN interface is suitable for the system configuration of single or multiple modules in a network and is able to control the operation of DES servoamplifiers over the protocol CAN 2.0B. This can be accomplished with the help of CAN controllers, I/O modules, PC CAN-interface cards or simply using a DES connected to the CAN network and the serial port of an host system. The CAN interface also enhances the DES hardware functionality by extending the number of possible physical inputs and outputs. The CAN protocol enables the connection of multiple devices on a bus with a high degree of data reliability even in a noisy and harsh environment.

Chapter 2 introduces first the general structure of the command instructions used to exchange data with a DES over RS232 or CAN bus. Each instruction and the related parameters are then listed and described.

Chapter 3 describes the implemented RS232 serial communication protocol. This allows the user to build applications of its own for the control of a DES or to monitor the system operation.

The CAN bus interface and some important protocol features are presented in Chapter 4. The use and configuration of a DES in a CAN bus environment are also explained.

This communication guide refers to the communication features implemented by the DES firmware. Further changes to protocols and instructions are not excluded.

2 Firmware Update in Graphical User Interface GUI

A new firmware can be downloaded from the internet site http://www.maxonmotor.com (see category «Service & Downloads»).

For the firmware update on the DES use the graphical user interface (available on the internet). Start the download wizard in the wizard view and follow the instructions.

Step 1: Download WARNING!
Read the download warning page and confirm that you've read the text. You're going to be informed that this firmware download is a critical procedure. In case of download interruption the code in the flash memory can get lost.

Step 2: Communication Settings!
This page appears only if a problem with the communication is detected. If the communication settings are correct you will step directly to the next page.
If you don't know which serial port is connected to the DES, use the function 'Search Communication'. Otherwise select the port and the baudrate manually.

Step 3: Firmware File!
Select the firmware file (*.bin) you want to download to the DES. In the middle of this page you see some version information. The old versions (on the DES) are listed on the left side. The new versions (selected firmware file) you see on the right side.
Additionally you have the possibility to read some information about the new firmware ('Header', 'Default Value' and 'ReadMe').

Step 4: Firmware Download!
Start the download clicking on the button 'Start'. Do not interrupt this download, otherwise the flash code can get lost.
If there's a problem with the download read the 'History'.

Step 5: System Parameter!
Download the default system parameter for the downloaded firmware. If you don't use this function a correct behaviour can't be guaranteed. You have also the possibility to edit the downloaded system parameters.

Step 6: Summary!
The downloaded versions are listed. If an error occurred you will be informed how serious this error is.

3 Command Operation

The DES can be controlled by the hardware I/Os by means of digital inputs, analogue setting values and potentiometers. The software control of a DES is accomplished by the command operation. Commands are transmitted using a serial EIA-RS232 protocol or a standard CAN 2.0B protocol.

3.1 Basic Instruction Format

The communication over serial interface (RS232) and CAN bus is based on the exchange of information following a defined command structure, which will be here referred as **B**asic **I**nstruction (BI).

BI messages are used to transmit and receive commands to and from DES modules through host systems or other DES. Depending on the interface used (CAN or RS232), other fields, as for example data length and identifier, are added to a Basic Instruction to form the complete message packet as described in the related sections about RS232 and CAN bus communication.

The BI format consists of two main blocks: **OpCode** (Operation Code) and **Data**. The *OpCode* specifies the operation to be executed by a DES servoamplifier or by modules connected in the CAN network. The *Data* block is used for data or parameters (also referred here as *Param1...n*) transmission as specified by the OpCode.

The length of the OpCode is 1 byte. The length of the Data block is determined by the command to be transmitted or received. The following figure illustrates the structure of a Basic Instruction. The Data block is here represented by the fields *Data 1...n*.

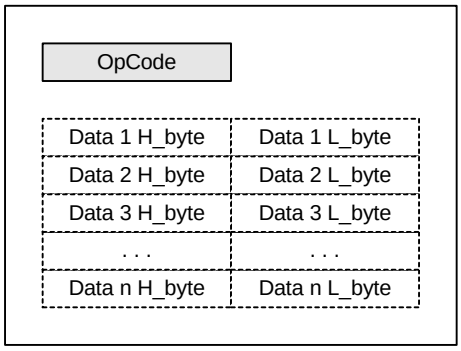


Figure 2.1: Basic Instruction format.

The unit of data in the DES memory is a 16-bit word. Words are always transmitted and received starting with the less significant byte (LSB). The OpCode, which is given by an 8-bit field, is always transmitted before the Data block. However when using one of the protocols implemented it is important to consider the transmission byte order of OpCode and of additional fields.

3.2 DES Command Reference

The following table lists all the user commands and the related parameters supported by the DES firmware. It is generally possible to divide the commands into two categories: one category contains commands which do not require the transmission of an answer from the receiver; the other consists of commands which require answers. The first ones are mainly used to configure the system or to change settings and parameters. The commands requiring an answer have the main purpose of monitoring the system operation and read the actual configuration values and parameters. The transmission and reception is in all cases performed by means of a message using the BI format.

The DES commands can be further classified in relation to the functionality type as follows:

- Status functions
- Service functions
- System parameter functions
- Setting functions
- Monitor functions
- Data recording functions
- CAN bus configuration functions

3.2.1 Status Functions

Command name	**Answer**
Description	The command is issued only by the DES after being polled by other commands and represents therefore an answer message. The DES response data are contained in the block Data (returned parameters).
Protocol support	RS232, CAN
Operation code	OpCode = 0x00
Data length	len = 1 ... n
Returned Parameter(s)	WORD param1 ... n; Returned parameters

Command name	**ReadSysStatus**
Description	Read the system status of the DES. The system status is a 16-bit value containing different flags.
Protocol support	RS232, CAN (SDO & PDO channels)
Operation code	OpCode = 0x01
Data length	len = 1
Parameter	WORD dummy = 0x0000; Variable without meaning
Response data	Frame containing the 16-bit status variable. b0: 0 = Encoder index not found yet 1 = Encoder index found b1: 0 = Hall sensor signal not found yet 1 = Hall sensor signal found b2: 0 = Rotor position not found yet 1 = Rotor position found b3: 0 = Not saving the system parameters in EEPROM 1 = Saving the system parameters in EEPROM b4: not used b5 + b6: reserved b7: 0 = Max. current set to peak current 1 = Max. current reduced to continuous current b8: 0 = In the large current region 1 = In the small current region b9: 0 = no error 1 = errors b10: 0 = Software disabled 1 = Software enabled b11: 0 = Not debouncing the enable input 1 = Debouncing the enable input

	b12:	0 = No offset in current circuit detected 1 = Offset in current circuit detected
	b13:	0 = Not braking 1 = Braking with the maximum setting current
	b14 + b15:	0 + 0 = Power stage is disabled 0 + 1 = Refresh the power stage 1 + 0 = Power stage is enabled 1 + 1 = Power stage is enabled

Command name	**ReadError**
Description	Read a 16-bit value of system errors.
Protocol support	RS232, CAN (SDO & PDO channels)
Operation code	OpCode = 0x02
Data length	len = 1
Parameter	WORD dummy = 0x0000; Variable without meaning
Response data	Frame containing the 16-bit error variable: b0: 1 = Hall sensor error b1: 1 = Index processing error b2: 1 = Wrong setting of encoder resolution b3: 1 = Hall sensor 3 not found b4: 1 = Over current error b5: 1 = Over voltage error b6: 1 = Over speed error b7: 1 = Supply voltage too low for operation b8: 1 = Angle detection error b13: 1 = Parameter out of range b15: 0 = No errors; 1 = There are errors

Command name	**ClearError**
Description	Clear the system error.
Protocol support	RS232, CAN (SDO & PDO channels)
Operation code	OpCode = 0x03
Data length	len = 1
Parameter	WORD dummy = 0x0000; Variable without meaning
Response data	No answer

Command name	**Reset**
Description	Reset the system by restarting the software.
Protocol support	RS232, CAN (SDO channel only)
Operation code	OpCode = 0x04
Data length	len = 1
Parameter	WORD dummy = 0x0000; Variable without meaning
Response data	No answer

Command name	Enable
Description	Set the system into the 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.
Protocol support	RS232, CAN (SDO & PDO channels)
Operation code	OpCode = 0x05
Data length	len = 1
Parameter	WORD newState; New state of the system Possible values: 0 = Disable 1 = Enable
Response data	No answer

3.2.2 Service Functions (reserved to advanced users)

Command name	Service
Description	Set the system into the service mode. This mode is enabled if a correct password consisting of 4 chars is received. The service mode allows a direct reading and writing of the DES memory. The command is reserved to advanced users.
Protocol support	RS232, CAN (SDO channel only)
Operation code	OpCode = 0x10
Data length	len = 2
Parameter	char password[4]; Password containing 4 characters
Response data	Frame containing a password acknowledge. WORD passwordAck; Possible values: 'O' (0x004F) = password is okay 'F' (0x0046) = password not correct

Command name	SetAddrVariable
Description	Write a value to a given memory address. This function can only be used in the service mode.
Protocol support	RS232, CAN (for len = 3) (SDO channel only)
Operation code	OpCode = 0x11
Data length	len = 3 or 4
Parameter	WORD addr; Memory address of the variable WORD dataFormat; Data Format of the variable Possible values: 0 = WORD;1 = LWORD WORD value; Value to be written at the memory address or LWORD value; Value to be written at the memory address (RS232 only)
Response data	Frame containing an acknowledge. WORD passwordAck; Possible values: 'O' (0x004F) = command executed 'F' (0x0046) = command not executed The service bit is not set.

Command name	ReadAddrVariable
Description	Read a value at a given address in the memory.
Protocol support	RS232, CAN (SDO channel only)
Operation code	OpCode = 0x12
Data length	len = 2
Parameter	WORD addr; Memory address of the variable WORD dataFormat; Data Format of the variable Possible values: 0 = WORD 1 = LWORD
Response data	Frame containing the value of the memory variable. WORD value; Value read from the memory address or LWORD value; Value read from the memory address

3.2.3 System Parameter Functions

Command name	**ReadTempParam**
Description	Read the requested temporary system parameter from DES-RAM.
Protocol support	RS232, CAN (SDO channel only)
Operation code	OpCode = 0x14
Data length	len = 2
Parameter	WORD paramNb; Number of the system parameter. See section on system parameters. WORD dataFormat; Data Format of the variable Possible values: 0 = WORD 1 = LWORD
Response data	Frame containing the temporary system parameter. WORD value; Value read from the system parameters or LWORD value; Value read from the system parameters

Command name	**SetTempParam**
Description	Write a new value to a temporary system parameter. Refer to the section about system parameters to find the desired system parameter numbers.
Protocol support	RS232, CAN (for len = 3) (SDO channel only)
Operation code	OpCode = 0x15
Data length	len = 3 or 4
Parameter	WORD paramNb; Number of the system parameter. See the section system parameters. WORD dataFormat; Data Format of the variable Possible values: 0 = WORD 1 = LWORD WORD value; New value of the system parameter or LWORD value; New value of the system parameter (RS232 only)
Response data	No answer

Command name	**ResetTempParam**
Description	Copy the permanent system parameter contained in the EEPROM memory into the temporary parameter set.
Protocol support	RS232, CAN (SDO channel only)
Operation code	OpCode = 0x16
Data length	len = 1
Parameter	WORD dummy = 0x0000; Variable without meaning
Response data	No answer

Command name	**SaveTempParam**
Description	Save the temporary parameters to the EEPROM (non volatile memory).
Protocol support	RS232, CAN (SDO channel only)
Operation code	OpCode = 0x17
Data length	len = 1
Parameter	WORD dummy = 0x0000; Variable without meaning
Response data	No answer

Command name	**ReadAllTempParam**
Description	Read all temporary system parameters. The system parameter structure is described in the section 'Data Structures'.
Protocol support	RS232
Operation code	OpCode = 0x18
Data length	len = 1
Parameter	WORD dummy = 0x0000; Variable without meaning
Response data	Frame containing the system parameter structure. DES_SysParam sysParam; Data structure for all system parameters

Command name	**SetAllTempParam**
Description	Write all temporary system parameters. The system parameter structure 'DES_SysParam' is described in the section 'Data Structures'.
Protocol support	RS232
Operation code	OpCode = 0x19
Data length	len = Number of words in the structure DES_SysParam
Parameter	DES_SysParam sysParam; Data structure containing the new system parameter values.
Response data	No answer

Command name	**ReadVersion**
Description	Read the versions of the firmware loaded on the DES.
Protocol support	RS232, CAN (SDO channel only)
Operation code	OpCode = 0x1A
Data length	len = 1
Parameter	WORD versionGroup RS232 : 0 = softVer, hardVer, appNb, appVer CAN : 0 = softVer, hardVer 1 = appNb, appVer
Response data	Frame containing the version information. RS232: WORD softVersion; Software version of firmware WORD hardVersion; Hardware version of firmware WORD appNumber; Application number of firmware WORD appVersion; Application version of firmware CAN (versionGroup = 0): WORD softVersion; Software version of firmware WORD hardVersion; Hardware version of firmware CAN (versionGroup = 1) WORD appNumber; Application number of firmware WORD appVersion; Application version of firmware

Command name	**SysParSetDefault** (Software Version 0x1050 and higher)
Description	Set all system parameters to default. The system parameter structure is described in the section 'Data Structures'.
Protocol support	RS232, CAN(SDO channel only)
Operation code	OpCode = 0x1B
Data length	len = 1
Parameter	WORD dummy = 0x0000; Variable without meaning
Response data	No answer

3.2.4 Setting Functions

Command name	**SetVelocity**
Description	Set a new velocity of the rotor. This function is only available in speed regulation mode.
Protocol support	RS232, CAN (SDO & PDO channels)
Operation code	OpCode = 0x21
Data length	len = 1
Parameter	short newVelocity; New velocity [rpm] of the rotor Positive value: counter clockwise Negative value: clockwise
Response data	No answer

Command name	**SetCurrent**
Description	Set new current amplitude. This function is only available in current regulation mode.
Protocol support	RS232, CAN (SDO & PDO channels)
Operation code	OpCode = 0x22
Data length	len = 1
Parameter	short newCurrent; New current amplitude [mA] Positive value: counter clockwise Negative value: clockwise
Response data	No answer

Command name	**StopMotion**
Description	This command changes the stopping state. If the motor is already stopped it will be released. The digital input STOP has the same behaviour. Only for speed regulation mode!
Protocol support	RS232, CAN (SDO & PDO channels)
Operation code	OpCode = 0x23
Data length	len = 1
Parameter	WORD dummy = 0x0000; Variable without meaning
Response data	No answer

3.2.5 Monitor Functions

Command name	**ReadVelocityIsMust**
Description	Read the effective and requested velocity of the motor.
Protocol support	RS232, CAN (SDO & PDO channels)
Operation code	OpCode = 0x28
Data length	len = 1
Parameter	WORD type = 0x0000; Mean values = 0x0001; Real time values
Response data	Frame containing the velocity information. type = 0 short isMeanVelocity; Mean velocity [rpm] short mustVelocity; Requested velocity [rpm] type = 1 short isVelocity; Effective velocity [rpm] short mustVelocity; Requested velocity [rpm]

Command name	ReadCurrentIsMust
Description	Read the effective and requested current components of the motor.
Protocol support	RS232
Operation code	OpCode = 0x29
Data length	len = 1
Parameter	WORD type 0 = Mean values; 1 = Real time values
Response data	Frame containing the current information. type = 0; (Mean values) short isQCurrent; Mean value of q-axis component of actual current (Torque) short isDCurrent; Mean value of d-axis component of actual current (≈ 0) short mustCurAmp; Amplitude of requested current [mA] type = 1; (Real time values) short isQCurrent; q-axis component of actual current [mA] (=> Torque) short isDCurrent; d-axis component of actual current [mA] (≈ 0) short mustCurAmp; Amplitude of requested current [mA]

3.2.6 Data Recording Functions

Command name	SetupRecorder
Description	Set up the recorder in the current regulator (10kHz)
Protocol support	RS232
Operation code	OpCode = 0x30
Data length	len = 2
Parameter	WORD samplePeriod; Sampling period as a factor of 0.1ms. (e.g. 124 = 12.4ms) WORD varNb; Number of a system parameter or a numbered status variable. If the number is greater than 0x0300 it represents a memory address.
Response data	No answer

Command name	RecordData
Description	Start recording data. The sampling starts immediately after a jump of the setting value. The recording is stopped after 256 samples. The current jump is only executed if the DES configured for a digital setting value.
Protocol support	RS232
Operation code	OpCode = 0x31
Data length	len = 1
Parameter	WORD jumpAmp; Setting value jump amplitude; 0 = No jump
Response data	Frame containing the sampled data. WORD data[256]; 256 recorded samples

Command name	ReadNVariables
Description	Read the values of a number of different variables synchronously.
Protocol support	RS232
Operation code	OpCode = 0x32
Data length	len = 1 to (n+1)...
Parameter	WORD nbOfVariables; Number of variables to read WORD varNb[n]; Numbers of system parameters or a numbered status variables. If the number is greater than 0x0300 it represents a memory address.
Response data	Frame containing the data vector. WORD data[n]; A vector of variables

3.2.7 CAN Bus Configuration Functions

Command name	**SetModuleID**
Description	Set CAN Module-ID (max. 11bit). The module ID is set by DIP switches at the system power up. During operation it's possible to overwrite temporary the moduleID with the command 'SetModuleID'. The moduleID determines the IDs for SDO communication .(TxSDO ID = 1408 + Module-ID; RxSDO ID = 1536 + Module-ID)
Protocol support	RS232, CAN (SDO channel only)
Operation code	OpCode = 0x39
Data length	len = 1
Parameter	WORD moduleID; new module ID
Response data	No answer

Command name	**SetServiceID**
Description	Set CAN Service-ID (max. 11bit). ☝This command has no meaning from SW version 0x1040 upwards. The serviceID is equal to the moduleID ☝
Protocol support	RS232, CAN (SDO channel only)
Operation code	OpCode = 0x3A
Data length	len = 1
Parameter	WORD serviceID; New service ID (Attention: serviceID = moduleID)
Response data	No answer

Command name	**SetTPDOID**
Description	Set CAN Transmit-PDO ID (11 bit). This is the message ID sent by the DES.
Protocol support	RS232, CAN (SDO channel only)
Operation code	OpCode = 0x3B
Data length	len = 1
Parameter	WORD tpdoID; Transmit–PDO ID
Response data	No answer

Command name	**SetRPDOID**
Description	Set CAN Receive-PDO ID (11 bit). This is the message ID received by the DES.
Protocol support	RS232, CAN (SDO channel only)
Operation code	OpCode = 0x3C
Data length	len = 1
Parameter	WORD rpdoID; Receive–PDO ID
Response data	No answer

Command name	**SendCANmsg**
Description	Send CAN standard frame message command.
Protocol support	RS232, CAN (SDO channel only)
Operation code	OpCode = 0x3D
Data length	len = 5
Parameter	WORD id; CAN 11-bit ID WORD dataA; CAN data A (CAN data bytes 2-1) WORD dataB; CAN data B (CAN data bytes 4-3) WORD dataC; CAN data C (CAN data bytes 6-5) WORD dataD; CAN data D (CAN data bytes 8-7)
Response data	No answer

Command name	**ReadModuleID**
Description	Read DES CAN Module-ID. The module ID is set by DIP switches at the system power up. It's possible that the system parameter 'ModuleID' has another value than the DIP switches. It's possible to overwrite this value temporary.
Protocol support	RS232, CAN (SDO channel only)
Operation code	OpCode = 0x3E
Data length	len = 1
Parameter	WORD dummy = 0x0000; Variable without meaning
Response data	Frame containing the module ID. WORD moduleID; CAN module ID

Command name	**SetCANBCR**
Description	Set CAN bit timing configuration register 1 and 2. See the section Bit Timing for more information about this configuration register.
Protocol support	RS232, CAN (SDO channel only)
Operation code	OpCode = 0x3F
Data length	len = 2
Parameter	WORD bcr1; CAN bit timing configuration register 1 WORD bcr2; CAN bit timing configuration register 2
Response data	No answer

Command name	**SetCANBitrate**
Description	Set CAN transfer rate to calculated values. See the section Bit Timing for more information about this configuration register.
Protocol support	RS232, CAN (SDO channel only)
Operation code	OpCode = 0x40
Data length	len = 1
Parameter	WORD bitrate; Index for transfer rate 0: 1 MBit/s 1: 800 kBit/s 2: 500 kBit/s 3: 250 kBit/s 4: 125 kBit/s 5: 50 kBit/s 6: 20 kBit/s 7: 10 kBit/s
Response data	No answer

Command name	**ReadCANError** (Version 0x1010 and higher)
Description	Read a 16 bit value of the CAN error register.
Protocol support	RS232, CAN (SDO channel only)
Operation code	OpCode = 0x43
Data length	len = 1
Parameter	WORD dummy = 0x0000; Variable without meaning
Response data	Frame containing the 16-bit error variable. b0: 1 = EW. Warning Status b1: 1 = EP. Error Passive Status b2: 1 = BO. Bus Off Status b3: 1 = ACKE. Acknowledge Error b4: 1 = SER. Stuff Error b5: 1 = CRCE. CRC Error b6: 1 = SA1. Stuck at dominant Error b7: 1 = BEF. Bit Error Flag b8: 1 = FER. Form Error Flag

	b9:	1 = PDO accessing frequency is too high
	b10:	1 = PDO overflow to lose sending message
	b11:	1 = TxPDO abort acknowledge in sending a message
	b12:	1 = TxSDO abort acknowledge in sending a message
	b13:	1 = RxPDO receive message lost
	b14:	1 = RxSDO receive message lost
	b15:	0 = Transmission successful; 1 = Transmission failed

Command name	**GetRemoteData** (Version 0x1010 and higher)	
Description	Read data from other DES connected to the CAN bus.	
Protocol support	RS232	
Operation code	OpCode = 0x44	
Data length	len = 2 ... 4	
Parameter	WORD destination; BYTE dummy = 0x00; BYTE opCode; WORD param1; WORD param2;	Module ID of addressed DES Dummy byte without meaning Operation code of the request command First parameter of the command (optional) Second parameter of the command (optional)
Response data	Frame consisting of three 16-bit values containing the answer to the command. WORD returnParam1; WORD returnParam2; WORD returnParam3;	 First returned parameter Second returned parameter Third returned parameter

Command name	**ConfigPDO**	
Description	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).	
Protocol support	RS232, CAN (SDO channel only)	
Operation code	OpCode = 0x45	
Data length	len = 1	
Parameter	WORD action;	0 = Switch Off; 1 = Switch On
Response data	No answer	

Command name	**SetRTRID**	
Description	Set CAN Remote Request Frame ID (11 bit). There are two possible channels for remote request frames. Read the actual RTR IDs from the system parameters 39 & 40 (RTR0 ID & RTR1 ID).	
Protocol support	RS232, CAN (SDO channel only)	
Operation code	OpCode = 0x46	
Data length	len = 2	
Parameter	WORD rtrChannel; WORD rtrID;	0 = RTR0; 1 = RTR1 Remote Request ID
Response data	No answer	

Command name	**ConfigRTR**	
Description	Switch on and off the RTR communication. The state of the RTR communication can be read with the system parameter 'CAN Config' (SysParam 41, Bit13 = RTR0; Bit12 = RTR1). Reset the RTR communication parameters before adding new parameters. (action = 2)	
Protocol support	RS232, CAN (SDO channel only)	
Operation code	OpCode = 0x47	
Data length	len = 2	
Parameter	WORD rtrChannel; WORD action;	0 = RTR0; 1 = RTR1 0 = Switch Off; 1 = Switch On; 2 = Reset Parameters
Response data	No answer	

Command name	**AddRTRParameter**
Description	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).
Protocol support	RS232, CAN (SDO channel only)
Operation code	OpCode = 0x4A
Data length	len = 2
Parameter	WORD paramSel: Bit0: 0 to select channel 0; 1 to select channel 1 Bit4: 0 to select paramMode; 1 to select addrMode Bit8: 0 to select Word (16-bit); 1 to select LWord (32-bit) WORD param Number (paramMode) or address (addrMode) of the system parameter. See the section system parameters.
Response data	Frame containing an acknowledge showing whether the parameter buffer is full or not. WORD ack; Possible values: 'O' (0x004F) = parameter added 'F' (0x0046) = parameter not added. The buffer is full.

Command name	**GetRTRParameter**
Description	Read the registered RTR parameters. The maximum number of parameters is 4 words (4 x 16-bit). If there are less than 4 parameters registered, the command sends a negative acknowledge ('F' = 0x0046) for the parameters which are not available.
Protocol support	RS232, CAN (SDO channel only)
Operation code	OpCode = 0x4B
Data length	len = 2
Parameter	WORD rtrChannel; 0 = RTR0; 1 = RTR1 WORD index; 0,1,2,3 (if parameter is available)
Response data	Frame containing RTR parameters. WORD ack: 'O' (0x004F) = parameter available 'F' (0x0046) = parameter not available WORD format: Bit4: 0 = paramMode, 1 = addrMode Bit8: 0 = Word (16-bit), 1 = LWord (32-bit) WORD param: Number (paramMode) or address (addrMode) of the registered system parameter. See the section system parameters.

Command name	**ResetCANError** (Software Version 0x1040 and higher)
Description	Reset the CAN Error.
Protocol support	RS232, CAN (SDO channel only)
Operation code	OpCode = 0x06
Data length	len = 1
Parameter	WORD dummy = 0x0000; Variable without meaning
Response data	No answer

Command name	**ResetCAN** (Software Version 0x1040 and higher)
Description	Reset the CAN communication.
Protocol support	RS232, CAN (SDO channel only)
Operation code	OpCode = 0x07
Data length	len = 1
Parameter	WORD dummy = 0x0000; Variable without meaning
Response data	No answer

3.3 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'	

Nb.	Parameter	Length	Access	Default	Range	Unit
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	

Nb.	Parameter	Length	Access	Default	Range	Unit
5	Speed regulator P-gain	16-bit	Read/Write	682	0 ... 32767	
6	Speed regulator I-gain	16-bit	Read/Write	220	0 ... 32767	

Nb.	Parameter	Length	Access	Default	Range	Unit
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	

Nb.	Parameter	Length	Access	Default	Range	Unit
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	

Nb.	Parameter	Length	Access	Default	Range	Unit
14	Peak Current	16-bit	Read/Write	30000	1 ... 30000	mA
15	Max continuous current	16-bit	Read/Write	10000	1 ... 10000	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	2189	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 ... 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	

Nb.	Parameter	Length	Access	Default	Range	Unit
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	378	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 Structures'	

Nb.	Parameter	Length	Access	Default	Range	Unit
42	InternalParam5	16-bit	Read	0	do not change	
43	ErrorProc	16-bit	Read/Write	0	0: Disable 1: Stop 0 ... 1	
44	MaxSpeed in Current Regulation Mode	16-bit	Read/Write	30000	0 ... 32767	rpm
45	HallAngleOffs	16-bit	Read/ServiceWrite	0	-32768 ... 32767	qc
46	MaxAngleMpy1	16-bit	Read/ServiceWrite	0x3FFF	0...0xFFFF	
47	MaxAngleMpyN	16-bit	Read/ServiceWrite	0x7FFF	0...0xFFFF	

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*)

3.4 DES Status Variables

Nb	Variable	Length	Unit
128	System operating status	16-bit	See section 'Data Structures'
129	Actual mean current value in d-axis (≈ 0)	16-bit	mA
130	Actual mean current value in q-axis (=> Torque)	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'
146	Adc Value of Set In	16-bit	
147	Adc Value of Ntc Sensor 1	16-bit	
148	Adc Value of Ntc Sensor 2	16-bit	
149	Adc Value of Gain Poti	16-bit	
150	Adc Value of Imax Poti	16-bit	
151	Adc Value of Offset Poti	16-bit	
152	Adc Value of Nmax Poti	16-bit	
153	Digital Inputs (Bit0:Enable, Bit1:Stop, Bit2:Digital Input 1, Bit3: Digital Input 2, Bit4: DipSwitch 9)	16-bit	

3.5 Data Type Definitions

Name	Data type	Size bits	Size bytes	Range
char	unsigned integer	8	1	0 ... 256
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

3.6 Data Structures

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/128ms
        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 InternalParam4;       //ParNb 22; R/W; Internally used, do not change
        short Rpm2QcFactor;         //ParNb 23; R/W; Conversion factor from 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 = CAN module ID
        short CAN_RxPDO_ID;         //ParNb 32; R; CAN Receive PDO ID
        short CAN_TxPDO_ID;         //ParNb 33; R; CAN Transmit PDO ID
        short CAN_BCR1;             //ParNb 34; R; CAN BCR1
        short CAN_BCR2;             //ParNb 35; R; CAN BCR2
        short CAN_OpMode;           //ParNb 36; R; 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; Remote Transmission Request ID (channel 0)
        short CAN_RTR1_ID;          //ParNb 40; R; Remote Transmission Request ID (channel 1)
        short CAN_Config;           //ParNb 41; R; CAN communication configuration register
        short InternalParam5        //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; R; Internally used, do not change
        short MaxAngleMpyN          //ParNb 47; R; Internally used, do not change
}DES_SysParam;
```

Definition of SysConfig

BIT 0:	0: speed/current setting by software
	1: speed/current setting by analogue 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 maximum speed by potentiometer 'P1'
	1: set maximum 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

Configuration of Regulation Mode

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

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

Definition of the 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:	not used
BIT 5:	reserved
BIT 6:	reserved
BIT 7:	0: Maximum current set to peak current
	1: Maximum 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
	1: debouncing the enable input
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

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

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

3.7 Standard Error Messages

Error 0	Hall Sensor Error

Meaning: Hall Sensor Error

Caused by:
* Wrong wiring of the hall sensors or the hall sensor supply voltage.
* Damaged hall sensors of the motor.

Effect: The red LED is on. The green LED flashes 1 interval.

Remark:
* The motor hall sensors report an impossible signal combination. This error can only occur during the initialisation procedure after power on.
* This error requires a hardware reset! Switch off and on the power supply!

Cross Reference:

Error 1	Index Processing Error

Meaning: Index Processing Error

Caused by:
* Encoder without or none working index channel.
* Too low setting of system parameter 'Encoder Resolution'.
* To high input frequency of encoder signals.

Effect: The red LED is on. The green LED flashes 2 intervals.

Remark:
* The index pulse of the encoder was not found within two turns.
* This error requires a hardware reset! Switch off and on the power supply!

Cross Reference: System Parameter No.20 'Encoder Resolution'

Error 2	Wrong setting of encoder resolution

Meaning: Wrong setting of encoder resolution

Caused by: * The setting of the system parameter 'Encoder Resolution' is wrong.

Effect: The red LED is on. The green LED flashes 3 intervals.

Remark:
* The encoder pulses counted between two index pulses do not correspond with the system parameter 'Encoder Resolution'. This error can only occur during the initialisation procedure after power on.
* This error requires a hardware reset! Switch off and on the power supply!

Cross Reference: System Parameter No.20 'Encoder Resolution'

Error 3	**Hall Sensor 3 not found**

Meaning: Hall Sensor 3 not found

Caused by:
* Wrong wiring of the hall sensor 3.
* Damaged hall sensor 3 of the motor.
* Too low setting of system parameter 'Encoder Resolution'.

Effect: The red LED is on. The green LED flashes 4 intervals.

Remark:
* No edge of hall sensor 3 found within one turn. This error can only occur during the initialisation procedure after power on.
* This error requires a hardware reset! Switch off and on the power supply!

Cross Reference: System Parameter No.20 'Encoder Resolution'

Error 4	**Over Current Error**

Meaning: Over Current Error

Caused by:
* Short circuit at motor windings.
* Power supply can not supply acceleration current.
* Gain regulation loop is too high. Reduce speed regulation gains.
* System parameter 'Acceleration' too high.
* Damaged power stage.
* Over temperature of power stage.

Effect: The red LED is on. The green LED flashes 5 intervals.

Remark:

Cross Reference: System Parameter No.18 'Acceleration' & No.1 'SysConfig' (Bit1)

Error 5	**Over Voltage Error**

Meaning: Over Voltage Error

Caused by:
* The power supply voltage is too high.
* Too high voltage in generation mode.

Effect: The red LED is on. The green LED flashes 6 intervals.

Remark:

Cross Reference:

Error 6	**Over Speed Error**

Meaning: Over Speed Error

Caused by: * The speed is too high (over 30'000 rpm).

Effect: The red LED is on. The green LED flashes 7 intervals.

Remark:

Cross Reference:

Error 7	Supply voltage too low for operation

Meaning: The supply voltage is too low for operation

Caused by: * The voltage is too low (under 8V).

Effect: The red LED is on. The green LED flashes 8 intervals.

Remark:

Cross Reference:

Error 8	Angle detection error

Meaning: Angle detection error

Caused by: * The angle difference measured between encoder and hall sensors is too high

Effect: The red LED shines continually. The green LED flashes at an interval of 9 pulses.

Remark:

Cross Reference:

Error 13	Parameter out of range

Meaning: Parameter out of range

Caused by: * The system parameter 'Encoder resolution' is out of range.

Effect: The red LED is on. The green LED flashes 14 intervals.

Remark:

Cross Reference: System Parameter No.20 'Encoder Resolution'

3.8 CAN Error Messages

CAN Error 0	Warning Status

Meaning: Warning Status

Caused by: * At least 96 transmit or receive errors occurred

Effect: No effect. The DES is still able to communicate. The DES is sending active error flags.

Remark:

Cross Reference:

CAN Error 1	Error Passive Status

Meaning: Error Passive Status

Caused by: * At least 127 transmit or receive errors occurred

Effect: The DES is still able to communicate. The DES is only sending passive error flags.

Remark:

Cross Reference:

CAN Error 2	Bus Off Status

Meaning: Bus Off Status

Caused by: * At least 255 transmit or receive errors occurred

Effect: The DES is automatically disconnected from the bus.

Remark:

Cross Reference:

CAN Error 3	Acknowledge Error

Meaning: Acknowledge Error

Caused by: * If the transmitting DES receives no ACK from one of the receivers. (No dominant level in ACK-Slot)

Effect: see CAN Error 0, CAN Error 1, CAN Error 2

Remark:

Cross Reference:

CAN Error 4 **Stuff Error**

Meaning: Stuff Error

Caused by: * After five consecutive equal bits, the sender is supposed to insert a stuff bit. This stuff bit is missing.

Effect: see CAN Error 0, CAN Error 1, CAN Error 2

Remark:

Cross Reference:

CAN Error 5 **CRC Error**

Meaning: CRC Error

Caused by: * If the received CRC code does not match the transmitted CRC code.

Effect: see CAN Error 0, CAN Error 1, CAN Error 2

Remark:

Cross Reference:

CAN Error 6 **Stuck at dominant Error**

Meaning: Stuck at dominant Error

Caused by: * More than 12 dominant error bits are detected

Effect: see CAN Error 0, CAN Error 1, CAN Error 2

Remark:

Cross Reference:

CAN Error 7 **Bit Error Flag**

Meaning: Bit Error Flag

Caused by: * If the transmitted bit and the received bit are different

Effect: see CAN Error 0, CAN Error 1, CAN Error 2

Remark:

Cross Reference:

CAN Error 8 **Form Error Flag**

Meaning: Form Error Flag

Caused by: * If a violation of frame format occurs

Effect: see CAN Error 0, CAN Error 1, CAN Error 2

Remark:

Cross Reference:

CAN Error 9 **PDO Accessing frequency is too high**

Meaning: PDO Accessing frequency is too high

Caused by: * To high frequency of data frames sent to the PDO input channel. (RxPDO ID)

Effect: The CAN communication is still running. This is only an information.

Remark:

Cross Reference:

CAN Error 10 **PDO Overflow**

Meaning: PDO Overflow. (message sent is lost)

Caused by: * Last PDO input message is not handled yet.

Effect: Last message is lost. The CAN communication is not in error state.

Remark:

Cross Reference:

CAN Error 11 **TxPDO No Acknowledge**

Meaning: TxPDO No Acknowledge received.

Caused by: * There's no CAN node setting the acknowledge slot to the dominant level. This means, no Can node received the last message sent via the PDO channel.

Effect: Last message via PDO channel is lost.

Remark: * The CAN communication is still working correctly.

Cross Reference:

CAN Error 12 **TxSDO No Acknowledge**

Meaning: TxSDO No Acknowledge received.

Caused by: * There's no CAN node setting the acknowledge slot to the dominant level. This means, no Can node received the last message sent via the SDO channel.

Effect: Last message via SDO channel is lost.

Remark: * The CAN communication is still working correctly.

Cross Reference:

CAN Error 13 **RxPDO Message Lost**

Meaning: RxPDO message is lost.

Caused by: * There's an error in the CAN frame.

Effect: Last received message via PDO channel is lost.

Remark: * The CAN communication is still working correctly.

Cross Reference:

CAN Error 14 **RxSDO Message Lost**

Meaning: RxSDO message is lost.

Caused by: * There's an error in the CAN frame.

Effect: Last received message via SDO channel is lost.

Remark: * The CAN communication is still working correctly.

Cross Reference:

4 Serial EIA-RS232 Communication

The serial RS232 communication protocol was developed for transmitting and receiving data over the RS232 serial port of a DES. Its principal task is to transmit data from a master (Personal Computer or any other central processing unit) to a single slave. The protocol is defined for a point-to-point communication based on the EIA-RS232 standard.
The protocol can be used to implement the command set defined for the DES. For a high degree of reliability in an electrically noisy environment it is designed with a checksum.

4.1 Physical Layer

4.1.1 Electrical Standard

The DES communication protocol uses the RS232 standard for transmitting data over a three wires cable, for the signals TxD, RxD and GND.
The RS232 standard can be used only for a point-to-point communication between a master and a single DES 50/5 slave. The standard uses negative, bipolar logic in which a negative voltage signal represents a logic '1', and positive voltage represents a logic '0'. Voltages of –3V to –25V with respect to signal ground (GND) are considered logic '1', whereas voltages of +3V to 25V are considered logic '0'.

4.1.2 Medium

For the physical connection a 3 wire cable is required. It is recommended to install a shielded cable in order to have a good performance even in an electrically noisy environment. Depending on the bit rate used the cable length can range from 3 meters up to 15 meters. However we do not recommend RS232 cables longer than 5 meters.

4.2 Data Link Layer

4.2.1 Data Format

Data is transmitted in an asynchronous way, that means each byte of data is transmitted individually with its own start and stop bit.

The format is:

1 Start bit, 8 Data bits, No parity, 1 Stop bit

Most serial communication chips (SCI, UART) can generate such data format.

4.2.2 Frame Structure

The data bytes are transmitted sequentially in frames. A frame is made of a header, a variably long data field and a 16-bit long cyclic redundancy check (CRC) for data integrity checking.

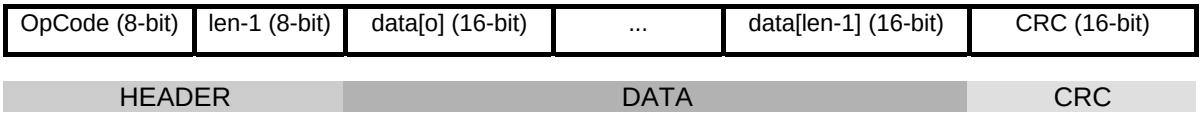


Figure 4.1: Frame structure

HEADER: The header consists of 2 bytes. The first field determines the type of data frame to be sent or received. The second field contains the length of the data fields.

OpCode: Operation command to be send to the slave. See the documentation of the command set.

len-1: 'Len' represents the number of words (16-bit value) in the data fields. The field 'Len-1' contains the number of words minus one. The smallest value in this field is zero, which represents a data length of one word. The data block must contain at least 1 word.

Examples:	1 word	⇨ len-1 = 0
	2 words	⇨ len-1 = 1
	...	
	256 words	⇨ len-1 = 255

DATA: The data fields contain the parameters of the message. It is important that this data block contains at least one word. The low byte of the word is transmitted first.

data[i]: Parameter word of the command. The low byte is transmitted first.

CRC: The 16-bit CRC checksum. The algorithm used is CRC-CCITT. The CRC calculation includes all bytes of the frame. The data bytes have to be calculated as a word. At first you have to shift in the high byte of the data word. This is the opposite way you transmit the data word. The 16-bit generator polynomial 'x^16+x^12+x^5+1' is used for the calculation.
Order of CRC calculation: 'OpCode', 'len-1', 'data[0]' high byte, 'data[0]' low byte, ...

CRC: Checksum of the frame. The low byte is transmitted first.

Warning: The data block must contain at least one word!

4.2.3 Transmission Byte Order

The unit of data memory in the DES is a word (16-bit value). To send and receive a word (16-bit) over the serial port of the DES, the low byte will be transmitted first.
Multiple word data are transmitted starting with the most significant word first.

A word will be transmitted in this order: LowByte (LSB), HighByte (MSB)
A long word will be transmitted in this order: HighWord, LowWord

HighWord (LowByte, HighByte)
LowWord (LowByte, HighByte)

4.2.4 Command Instruction Example

We give here an example of a command frame for the serial RS232 communication to show the composition and structure of DES messages during transmission and reception.

The command sent to the DES is ***ReadVersion***. The command can be used to read the versions loaded on the DES. The frame containing the BI is:

OpCode	len-1	data[0]	CRC
0x1A	0x00	0x0000	0x730C

OpCode: *ReadVersion* = 0x1A
data[0]: dummy word = 0x00

The DES answers to the command *ReadVersion* with a BI consisting of the command ***Answer*** and the returned parameters in the Data block as follows:

OpCode	len-1	data[0]	data[1]	data[2]	data[3]	CRC
0x00	0x03	0x1050	0x4101	0x00A1	0x0001	0x7902

OpCode: *Answer* = 0x00
data[0]: software version = 0x1050
data[1]: hardware version = 0x4101
data[2]: application number = 0x00A1
data[3]: application version = 0x0001

4.2.5 Protocol and Flow Control

Sequence for sending DES commands

The DES is always communicating as a slave. A frame is only sent as an answer to a request. Some of the DES commands send an answer, other commands do not. Have a look at the description of the commands to know which command sends an answer packet. The master always has to start the communication sending a packet structure.
The next two sections describe the data flow of transmission and reception frames.

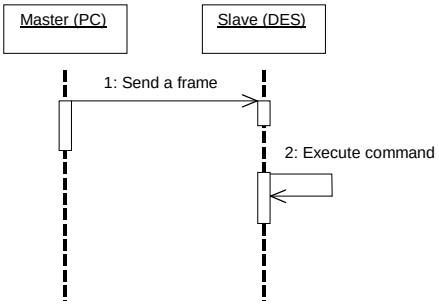


Figure 4.2: DES command without answer

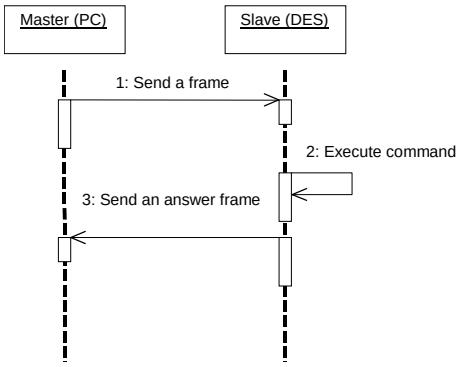


Figure 4.3: DES command with answer

Sending a data frame

When sending a frame you have to wait for different acknowledges. The first is a ‘Ready Acknowledge’. After sending the first byte of the frame (OpCode) you have to wait for an acknowledge of the DES. If the char ‘O’ (okay) is received, then the slave is ready to receive other data. If any other char is received the communication has to be stopped. If everything is okay you can send the rest of the data frame.

After sending the checksum you have to wait for the ‘End Acknowledge’. The slave sends either the char ‘O’ (okay) or the char ‘F’ (failed).

The following figure shows the interaction diagram of sending a packet structure.

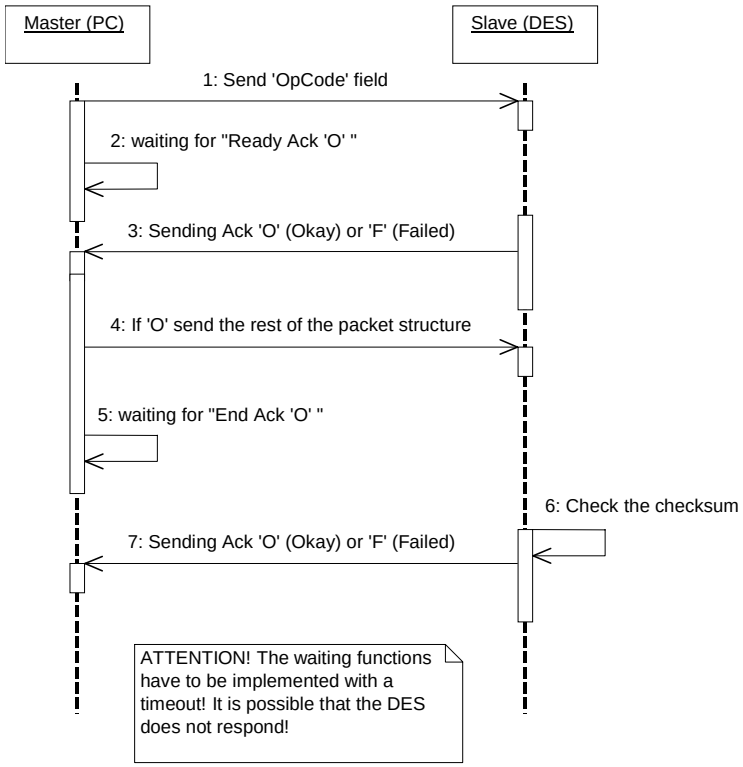


Figure 4.4: Sending a data frame to the DES

Receiving a data frame

In response to some of the command frames, the DES sends a data frame back to the master. The sequence of data flow is the same as for sending a data packet. Only the direction is changed. The master has also to send the two acknowledges to the slave. After receiving the first byte the master has to send the ‘Ready Acknowledge’. Send the char ’O’ if you have received the correct OpCode. The value of the field must always be 0x00. This is the operation code which describes an answer packet. If the received OpCode is not zero you have to send the acknowledge ‘F’.
After sending the ‘Ready Acknowledge’ the rest of the data frame can be read. Then the checksum must be calculated and compared with the one received. If the checksum is correct send the acknowledge ‘O’ (okay), otherwise send the acknowledge ‘F’ to the DES.

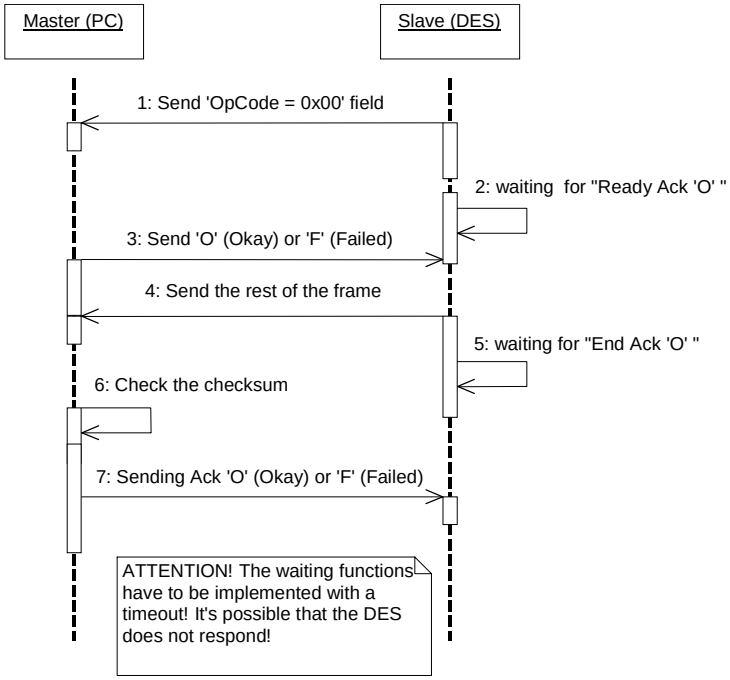


Figure 4.5: Receiving a data frame from the DES

5 CAN Bus Communication

The DES implements a CAN protocol fully compatible with the standard CAN 2.0B. The description and specifications of the whole protocol at the lowest two layers of the ISO/OSI model can be easily found in literature, and are therefore not covered by this guide. We will here only explain or recall some important general configuration and network setup aspects, as well as the usage of the protocol for the DES functionality.

The application layer (Layer 7) was developed to enable a simple exchange of data in a network containing DES and other CAN or CANopen modules. The user can send and receive most of the DES commands (see Command Reference). It is also possible to build own control programs with the commands (BI) and the CAN bytes mapping described in the following sections. Additionally the implementation at Layer 7 allows the user to send every CAN message to the network by means of the serial RS232 protocol. Therefore no CAN interface cards are needed to work with an host system like a PC and the CAN network containing one or more DES.

5.1 Physical Layer

5.1.1 Bit Timing

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 Frequency
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

Definitions: $T_{SEG1} \geq T_{SEG2}$
$T_{SEG2min}$ = 1 + SJW

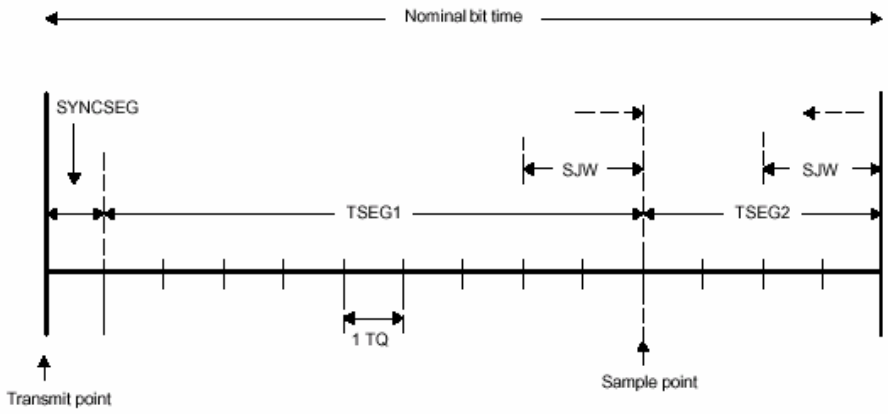


Figure 5.1: Bit Timing Calculation

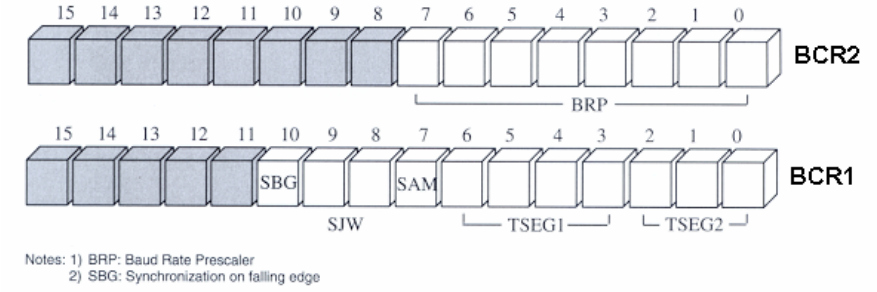


Figure 5.2: Bit Timing Register

BRP = Baudrate Prescaler
SBG = 0 (Synchronization on falling edge); 1(Synchronization on rising edge)
SJW = Synchronization jump width
SAM = 0 (CAN module samples only once); 1(CAN module samples three times and makes a majority decision
TSEG1 = T_{SEG1} (Time segment 1)
TSEG2 = T_{SEG2} (Time segment 2)

Calculation Example for 500kBit/s:

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

BRP (= BCR2)	= (Nominal TQ * fOsc) – 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 0x4102)

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 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

5.1.2 Physical Medium Attachment

The DES CAN physical medium attachment (PMA) is defined by a transceiver compatible up to 1Mbit/s to the ISO 11898 specifications (CAN High-Speed). The servoamplifier can be integrated in a CAN network by connecting “CAN high” with *CAN_H*, and respectively “CAN low” with the *CAN_L* cable. The ground signal “GND” of different CAN nodes must be interconnected.

5.1.3 Medium Dependant Interface

The bus physical medium is defined by a two-wire bus line terminated at both ends by a resistor of about 124Ohm. The two-wires may be twisted and/or shielded depending on EMC requirements. The nodes should be connected to the line with very short cable stubs, especially when operating at high bit-rates. At 1Mbit/s the length of cable stubs, i.e. the distance between node and bus line, should not exceed 0.3m.

The following figure illustrates the most common used termination concept. Each node, representing for example a DES module or any other CAN device, is connected to the line with short cable stubs. In this concept the network topology is close to a single line structure to reduce reflections.

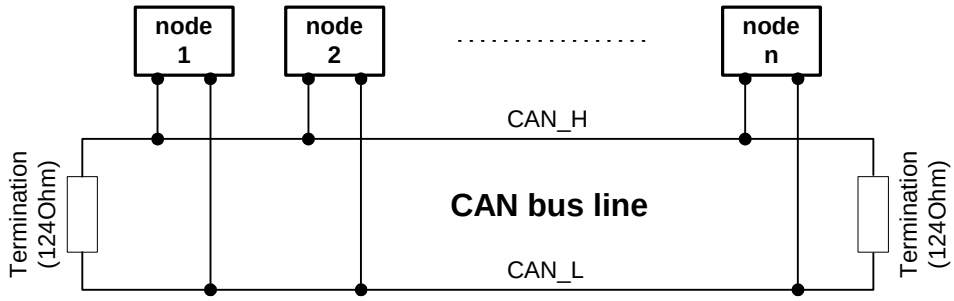


Figure 5.3: CAN bus line

The number of possible CAN nodes depends on several factors, like transceiver characteristics, network topology, line delays, etc.

The following table lists some important DC characteristics to observe when building the physical connection with a DES:

Bus length	Length related resistance	Bus-Line Cross-Section	Termination resistance	Max. baudrate
0 ... 40 m	70 mOhm/m	0.25 mm2 ... 0.34 mm2 AWG23,AWG22	124 Ohm (1%,200 mW)	1Mbit/s at 40 m

5.2 Data Link Layer

5.2.1 Standard CAN Data Frame

The Data Link Layer is conforming to the Robert Bosch GmbH specification 2.0B. The implementation makes use of standard 11-bit identifiers (passive implementation). Messages with extended identifiers are ignored.

A standard CAN Data Frame consists of several fields, which can be grouped in Start Of Frame (SOF), Arbitration Field (contains the Identifier or ID and the remote transmission request bit), Control Field, Data Field (contains from 0 to 8 *CANbytes*), CRC Field, ACK Field and EOF. The part of a standard data frame which is relevant for the understanding of the CAN protocol implementation in a DES is evidenced in the following figure.

Figure 5.4: ID, Remote Transmission Request Bit and Data Field in a Standard Data Frame

Remember that in CAN networks there is no addressing of nodes in the conventional sense, but instead messages with a given priority are transmitted. The Identifier defines the priority of messages and bus arbitration.

The user needs to specify ID (Identifier) and message content (Data Field) to send a valid DES Basic Instruction. The BI is mapped into the CAN Data Field, starting from CANbyte1 up to CANbyte8 (see Basic Instruction Mapping).

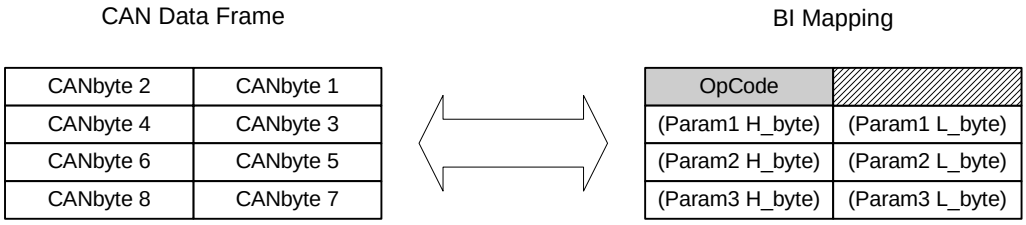
There's also the possibility to use two remote transmission request channels. The bit RTR distinguishes a CAN data frame (RTR-Bit = 0) from a CAN remote transmission request frame (RTR-Bit = 1).

5.2.2 CAN Data Frame

The ID of the CAN Data frame depends on the communication channel and the corresponding system parameters in the DES. See sections SDO, PDO or RTR communication.

For the transmission of a CAN Data Frame the Bit RTR has to be set to zero (RTR = 0).

The data bytes of the CAN Data Frame depends on the command you want to send. The DES instructions have to be mapped to the CAN Bytes 1 – 8. The mapping of a BI is illustrated in the following figure. The values of 'OpCode' and the meaning of the data 'Param1' to 'Param3' is



described in the section 'DES Command Reference'.

Figure 5.5: Basic Instruction mapping

In the mapped BI the transmission order correspond to the CANbytes order. The first byte transmitted is therefore a reserved dummy byte, which can be set as 0x00. The second byte is the OpCode. The Data fields are transmitted starting from the less significant byte (LSB) of the first command parameter (*Param1*); the second parameter and the third are transmitted in the same way. The Data block (*Param1..3*) is optionally, i.e. parameters are only required by certain commands. Note also that in a DES the CAN Data bytes are always transmitted and received in the order: CANbyte1, CANbyte2, CANbyte3, CANbyte4, ..., CANbyte8.

5.2.3 CAN Remote Transmission Request Frame

The DES protocol is able to handle remote transmission requests. There are two channels reserved for RTR communication (see section RTR communication). The Remote Transmission Request Frame (RTR Frame) contains no data. With this type of frame the user can make a request for a data frame. The user sends a CAN RTR frame with a defined ID (RTR Bit = 1, without data) and the DES responds with a CAN Data frame (RTR Bit set to zero, with data corresponding to the frame ID). For more information about RTR communication have a look at the section 'Application Layer'.

5.3 Application Layer

5.3.1 Communication Channels

The firmware of a DES provides a communication mechanism based on configurable CAN connections for the internal device access. The CAN input connections of a DES are composed of 3 types of communication channels.

SDO communication channels		
1 Receive SDO channel	ID = RxSDO ID (SysParam 37)	= 1536 + moduleID
1 Transmit SDO channel	ID = TxSDO ID (SysParam 38)	= 1408 + moduleID
PDO communication channels		
1 Receive PDO channel	ID = RxPDO ID (SysParam 32)	= 385 up to 1407
1 Transmit PDO channel	ID = TxPDO ID (SysParam 33)	= 385 up to 1407
RTR communication channels		
2 Transmit Request channels	ID = RTR0 ID (SysParam 39)	= 385 up to 1407
	ID = RTR1 ID (SysParam 40)	= 385 up to 1407

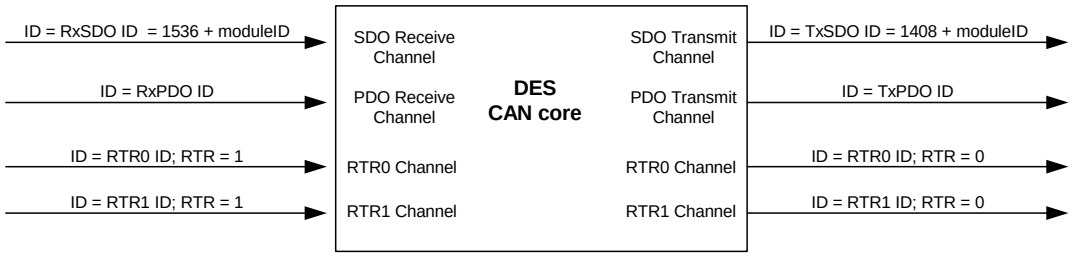


Figure 5.6: CAN communication channels

The **SDO IDs** are a result of the moduleID (SysParam30). The moduleID is set by the hardware DIP switch at system start-up. During operation the moduleID can be set temporary with the command 'SetModuleID'. This kind of ID setting guarantees a secure communication at system start-up.

The **PDO IDs** and the **RTR IDs** can be configured via the SDO communication channels. The IDs can be free adjusted in a certain range.

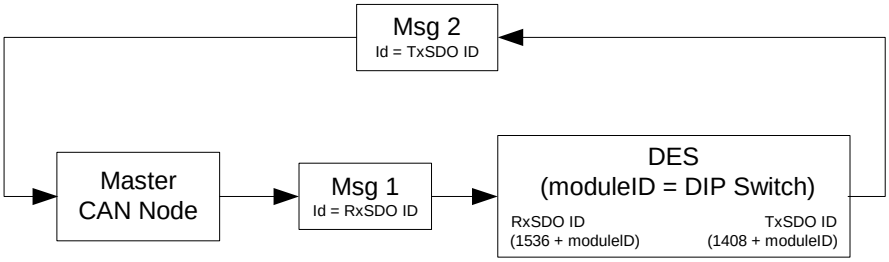
The identifier of each message sent on a CAN bus must match a valid input identifier in order to be accepted by a DES.

5.3.2 SDO Communication (Service Data Objects)

The SDO communication channels can be used to configure the system parameters of the DES. This type of communication is always active. The message IDs are related to the moduleID which is set by the DIP switch. (Older hardware versions don't have this DIP switch, use the command 'SetModuleID' to adjust the module ID). The moduleID is always read from the DIP at system start-up. After system start-up the moduleID can be changed temporarily.

The communication handling does not make use of interrupts. So this type of communication is not recommendable for a high frequency of process data.

SDO Communication Structure



System Parameters

SysParam No.30	moduleID	= DIP Switch	(Read/Write)
SysParam No.37	RxSDO ID	= 1536 + moduleID	(ReadOnly)
SysParam No.38	TxSDO ID	= 1408 + moduleID	(ReadOnly)

SDO Configuration Procedure

1. Select the moduleID with the DIP switch on the hardware
2. Read the system parameters No.37 & No.38 (RxSDO ID & TxSDO) to check SDO message IDs.

SDO Message Mapping

Msg1:			Msg2:		
	ID	= RxSDO ID		ID	= TxSDO ID
	RTR	= 0		RTR	= 0
	CanByte1	= 0x00 (Dummy)		CanByte1	= 0x00 (Dummy)
	CanByte2	= command OpCode		CanByte2	= 0x00 (OpCode = answer)
	CanByte3	= Param1 LowByte		CanByte3	= ReturnParam1 LowByte
	CanByte4	= Param1 HighByte		CanByte4	= ReturnParam1 HighByte
	CanByte5	= Param2 LowByte		CanByte5	= ReturnParam2 LowByte
	CanByte6	= Param2 HighByte		CanByte6	= ReturnParam2 HighByte
	CanByte7	= Param3 LowByte		CanByte7	= ReturnParam3 LowByte
	CanByte8	= Param3 HighByte		CanByte8	= ReturnParam3 HighByte

SDO Communication Examples

DIP Switch = moduleID = 1 -> RxSDO ID = 1536 + 1 = 1537
-> TxSDO ID = 1408 + 1 = 1409

Example 1: Read the system parameter No.14 'PeakCurrent'

ReadTempParam:		OpCode	= 0x14	= 20d	
		paramNb	= 0x000E	= 14d	
		dataFormat	= 0x0000	= 0d	
		responseValue	= 0x3A98	= 15000d	
Msg1:	ID	= 1537	Msg2:	ID	= 1409
	RTR	= 0		RTR	= 0
	CanByte1	= 0x00 (dummy)		CanByte1	= 0x00 (dummy)
	CanByte2	= 0x14		CanByte2	= 0x00 (OpCode = answer)
	CanByte3	= 0x0E		CanByte3	= 0x98
	CanByte4	= 0x00		CanByte4	= 0x3A
	CanByte5	= 0x00		CanByte5	= 0x00 (dummy)
	CanByte6	= 0x00		CanByte6	= 0x00 (dummy)
	CanByte7	= 0x00 (dummy)		CanByte7	= 0x00 (dummy)
	CanByte8	= 0x00 (dummy)		CanByte8	= 0x00 (dummy)

Example 2: Set the system parameter No.14 'PeakCurrent'

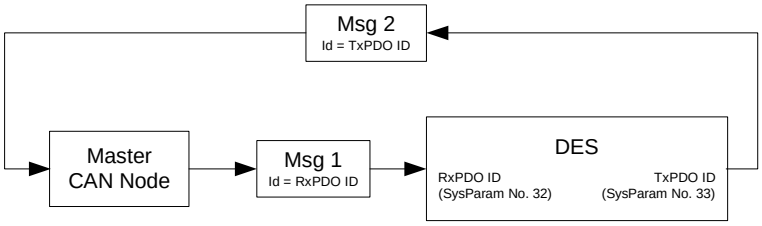
SetTempParam:		OpCode	= 0x15	= 21d
		paramNb	= 0x000E	= 14d
		dataFormat	= 0x0000	= 0d
		value	= 0x2710	= 10000d
Msg1:	ID	= 1537	Msg2:	No answer
	RTR	= 0		
	CanByte1	= 0x00 (dummy)		
	CanByte2	= 0x15		
	CanByte3	= 0x0E		
	CanByte4	= 0x00		
	CanByte5	= 0x00		
	CanByte6	= 0x00		
	CanByte7	= 0x10		
	CanByte8	= 0x27		

5.3.3 PDO Communication (Process Data Objects)

The DES Can communication implementation contains one receive and one transmit channel for PDO objects. To use more than one command with this single point to point connection, the PDO communication is implemented as multiplexed PDOs. All commands can be executed except the system parameter setting functions. Normally in a CAN network there's only one master which controls a DES and its motor axis. Use this type of communication for commands like 'SetVelocity', 'SetCurrent', 'Enable', 'StopMotion'.
The PDO communication has to be activated with the command 'ConfigPDO'. Normally the PDO communication is disabled. The message IDs have to be configured via SDO communication.

The PDO communication is internally handled by interrupts. This means this type of communication can guarantee a constant reaction time.

PDO Communication Structure



System Parameters

SysParam No.32	RxPDO ID	= 385 - 1407	(Read/Write)
SysParam No.33	TxPDO ID	= 385 - 1407	(Read/Write)
SysParam No.41	CANConfig.Bit14	= 0 (Disabled)	(ReadOnly)
		= 1 (Enabled)	(ReadOnly)

PDO Configuration Procedure

1. Write the system parameter No.32 (RxPDO ID) via SDO communication using the command 'SetRPDOID'.
2. Write the system parameter No.33 (TxPDO ID) via SDO communication using the command 'SetTPDOID' (Only necessary for commands with an answer message).
3. Enable the PDO communication via SDO communication using the command 'Config PDO' (With the same command the PDO communication can also be switched off).

PDO Message Mapping

Msg1:			Msg2:		
	ID	= RxPDO ID		ID	= TxPDO ID
	RTR	= 0		RTR	= 0
	CanByte1	= 0x00 (dummy)		CanByte1	= 0x00 (dummy)
	CanByte2	= command OpCode		CanByte2	= 0x00 (OpCode = answer)
	CanByte3	= Param1 LowByte		CanByte3	= ReturnParam1 LowByte
	CanByte4	= Param1 HighByte		CanByte4	= ReturnParam1 HighByte
	CanByte5	= Param2 LowByte		CanByte5	= ReturnParam2 LowByte
	CanByte6	= Param2 HighByte		CanByte6	= ReturnParam2 HighByte
	CanByte7	= Param3 LowByte		CanByte7	= ReturnParam3 LowByte
	CanByte8	= Param3 HighByte		CanByte8	= ReturnParam3 HighByte

PDO Communication Examples

SysParam No. 32 = RxPDO ID = 513
SysParam No. 33 = TxPDO ID = 385

Example 1: Enable the DES by a software command

Enable:		OpCode	= 0x05	= 5d
		newState	= 0x0001	= 1d
Msg1:	ID	= 513	Msg2:	No answer
	RTR	= 0		
	CanByte1	= 0x00 (dummy)		
	CanByte2	= 0x05		
	CanByte3	= 0x01		
	CanByte4	= 0x00		
	CanByte5	= 0x00 (dummy)		
	CanByte6	= 0x00 (dummy)		
	CanByte7	= 0x00 (dummy)		
	CanByte8	= 0x00 (dummy)		

Example 2: Set a new speed setting value

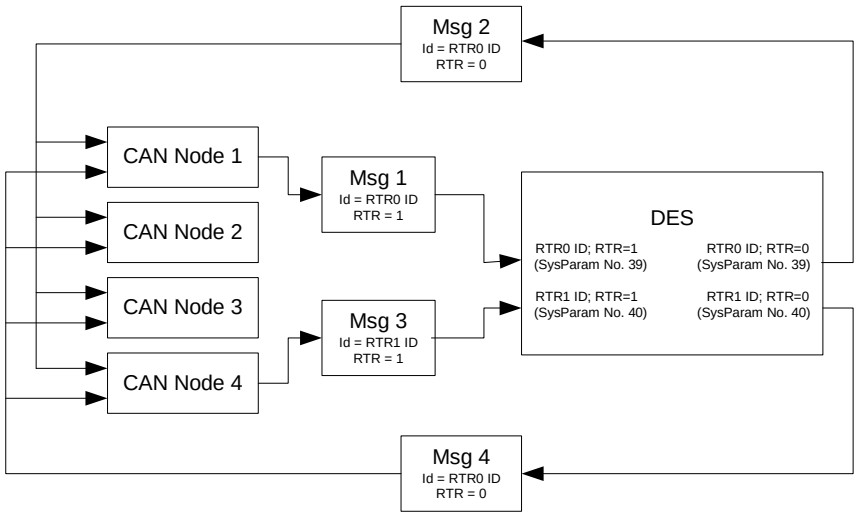
SetVelocity:		OpCode	= 0x21	= 33d
		newVelocity	= 0x09C4	= 2500d
Msg1:	ID	= 513	Msg2:	No answer
	RTR	= 0		
	CanByte1	= 0x00 (dummy)		
	CanByte2	= 0x21		
	CanByte3	= 0xC4		
	CanByte4	= 0x09		
	CanByte5	= 0x00 (dummy)		
	CanByte6	= 0x00 (dummy)		
	CanByte7	= 0x00 (dummy)		
	CanByte8	= 0x00 (dummy)		

5.3.4 RTR Communication (Remote Transmission Request)

For remote transmission requests there are two channels available (RTR0 & RTR1). The IDs of these two channels can be configured via SDO communication. These two channels guarantee a very fast information update. The data registers are updated with the frequency of the current regulation loop. Any CAN node in a network can request the latest values. The requested data frame contains up to 4 system- or status variables. The content of the data fields can be configured via SDO communication using the commands 'AddRTRParameter' and 'ConfigRTR'. Using two RTR channels it's possible to monitor maximally 8 parameters (8 x 16bit values, 4 x 32bit values).

The RTR communication has to be enabled with the command 'ConfigRTR'. Normally this communication is disabled.

RTR Communication Structure



System Parameters

SysParam No.39	RTR0 ID	= 385 - 1407	(Read/Write)
SysParam No.40	RTR1 ID	= 385 - 1407	(Read/Write)
SysParam No.41	CANConfig.Bit13	= 0 (RTR0 Disabled)	(ReadOnly)
		= 1 (RTR0 Enabled)	(ReadOnly)
SysParam No.41	CANConfig.Bit12	= 0 (RTR1 Disabled)	(ReadOnly)
		= 1 (RTR1 Enabled)	(ReadOnly)

RTR Configuration Procedure

1. Write the system parameter No.39 or No.40 (RTR0 ID or RTR1 ID) via SDO communication using the command 'SetRTRID'.
2. Reset the parameter configuration for the requested data frame. Send the command 'ConfigRTR' with the parameter 'action' = 2 (Reset) via SDO communication.
3. Register new parameters for the requested data frame. Execute the command 'AddRTRParameter'. The command answers with a negative acknowledge ('F' = 0x0046) if the buffer is full.
4. Enable the RTR communication channel using the command 'ConfigRTR' with the parameter 'action' = 1 (Switch On).

RTR Message Mapping

Msg1/3:	ID	= RTR0/RTR1 ID	Msg2/4:	ID	= RTR0/RTR1 ID
	RTR	= 1		RTR	= 0
No data			CanByte1	= Registered Param1 LowByte	
			CanByte2	= Registered Param1 HighByte	
			CanByte3	= Registered Param2 LowByte	
			CanByte4	= Registered Param2 HighByte	
			CanByte5	= Registered Param3 LowByte	
			CanByte6	= Registered Param3 HighByte	
			CanByte7	= Registered Param4 LowByte	
			CanByte8	= Registered Param4 HighByte	

RTR Communication Example

SysParam No. 39 = RTR0 ID = 386
SysParam No. 40 = RTR1 ID = 387

Example: Read the status parameters: opStatus, error, canError, velocityIs

Configuration via SDO communication:

```
ConfigRTR(action = 2)                                   //Reset parameter configuration
AddRTRParameter(paramSel = 0,paramNbAddr = 128) // Bit0 = 0 -> RTR0;
                                                        // Bit4 = 0 -> paramMode;
                                                        // Bit8 = 0 -> WORD (16-bit)
                                                        // paramNbAddr = 0x0080 = 128d (opStatus)

AddRTRParameter(paramSel = 0,paramNbAddr = 137) // Bit0 = 0 -> RTR0;
                                                        // Bit4 = 0 -> paramMode;
                                                        // Bit8 = 0 -> WORD (16-bit)
                                                        // paramNbAddr = 0x0089 = 137d (error)

AddRTRParameter(paramSel = 0,paramNbAddr = 138) // Bit0 = 0 -> RTR0;
                                                        // Bit4 = 0 -> paramMode;
                                                        // Bit8 = 0 -> WORD (16-bit)
                                                        // paramNbAddr = 0x008A = 138d (canError)

AddRTRParameter(paramSel = 0,paramNbAddr = 134) // Bit0 = 0 -> RTR0;
                                                        // Bit4 = 0 -> paramMode;
                                                        // Bit8 = 0 -> WORD (16-bit)
                                                        // paramNbAddr = 0x0086 = 134d (velocityIs)

ConfigRTR(action = 1)                                   // Switch on RTR communication
```

Results:	opStatus	= 0xC407 = 50183d
	error	= 0x8002 = 32770d
	canError	= 0x8001 = 32769d
	velocityIs	= 0x050C = 1500d

Msg1:	ID	= 386	Msg2:	ID	= 386
	RTR	= 1		RTR	= 0
	No data			CanByte1	= 0x07
				CanByte2	= 0xC4
				CanByte3	= 0x02
				CanByte4	= 0x80
				CanByte5	= 0x01
				CanByte6	= 0x80
				CanByte7	= 0x0C
				CanByte8	= 0x05

5.4 Using a DES in a CAN Network via RS232

5.4.1 Configuration

If a DES has to be used in CAN network then it is generally necessary to configure first each Module-ID. This parameter should be configured if another device is already using the same ID. To avoid confusion it is therefore recommended to configure separately each DES-ID by means of the delivered GUI and serial interface (RS232) before building the connection to the CAN bus. A configuration over CAN bus is of course possible, but then the user must be sure that the operation is not interfering with other DES modules.

5.4.2 Control of CAN using the Serial Interface

A remote DES in a CAN network can be controlled and configured by using a Serial Communication Interface (SCI, UART) and a DES connected to a host system with the serial protocol described for the standard EIA-RS232. The following figure illustrates a possible network scenario with a DES connected to a PC and other CAN modules.

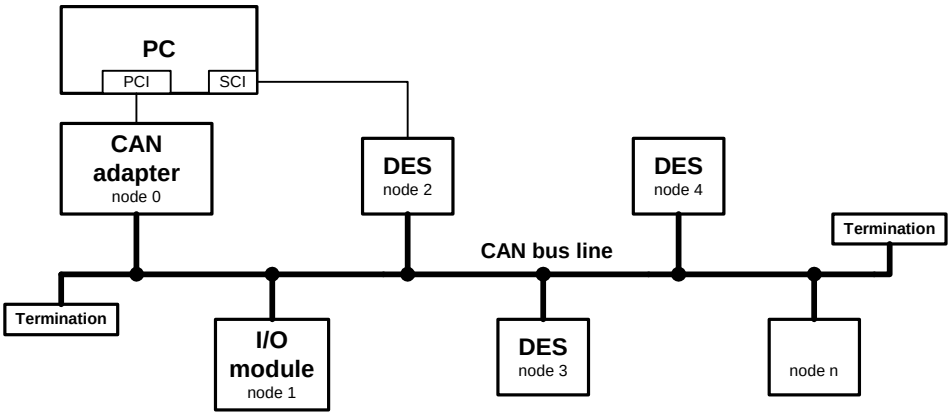


Figure 5.8: Connections on a CAN network with a PC host system over SCI and CAN adapter card

The communication from an host system like a PC and a DES in a CAN network can be of course established by means of a CAN interface for the intern system bus (for example PCI). However for simple applications the user can send every CAN message and use most of the DES commands with the help of the serial protocol. The DES connected over RS232 to the PC will act as an interface converting the message from SCI to CAN. It is therefore possible to operate on remote DES without the need of extra CAN adapter cards.

5.4.3 Command Instruction Example

As already mentioned DES commands are exchanged over CAN using the BI communication features. We present here an example of instruction requiring an answer from the receiver. This is the case of the command ***ReadVersion*** (see also the equivalent example for the RS232 protocol).

Supposing that we desire to read the version of a DES with *Module-ID* = 3 (*Module-Ch*), then we can send a CAN standard message to the bus with the ID = 3 and the CAN bytes containing the BI mapped as described.

The frame to send to the CAN bus for the command *ReadVersion* is composed as follows:

ID	CANbyte1	CANbyte2	CANbyte3	CANbyte4
		OpCode	Param1 L_byte	Param1 H_byte
0x0003	0x00	0x1A	0x00	0x00

ID:	*Module-ID*	= 3
OpCode:	*ReadVersion*	= 0x1A
Param1 (L-byte):		= 0x00 (Versions Group)
Param1 (H-byte):		= 0x00

The DES will then answer with a CAN standard message with the ID = *RPDO-ID* and an BI containing the required information.

ID	CANbyte1	CANbyte2	CANbyte3	CANbyte4	CANbyte5	CANbyte6
	(len-1)	OpCode	Param1 L_byte	Param1 H_byte	Param2 L_byte	Param2 H_byte
0x0201	0x01	0x00	0x50	0x10	0x01	0x41

ID:	*RPDO-ID*	= 0x0201
OpCode:	*Answer*	= 0x00
Param1:	Software version	= 0x1050
Param2:	Hardware version	= 0x4101