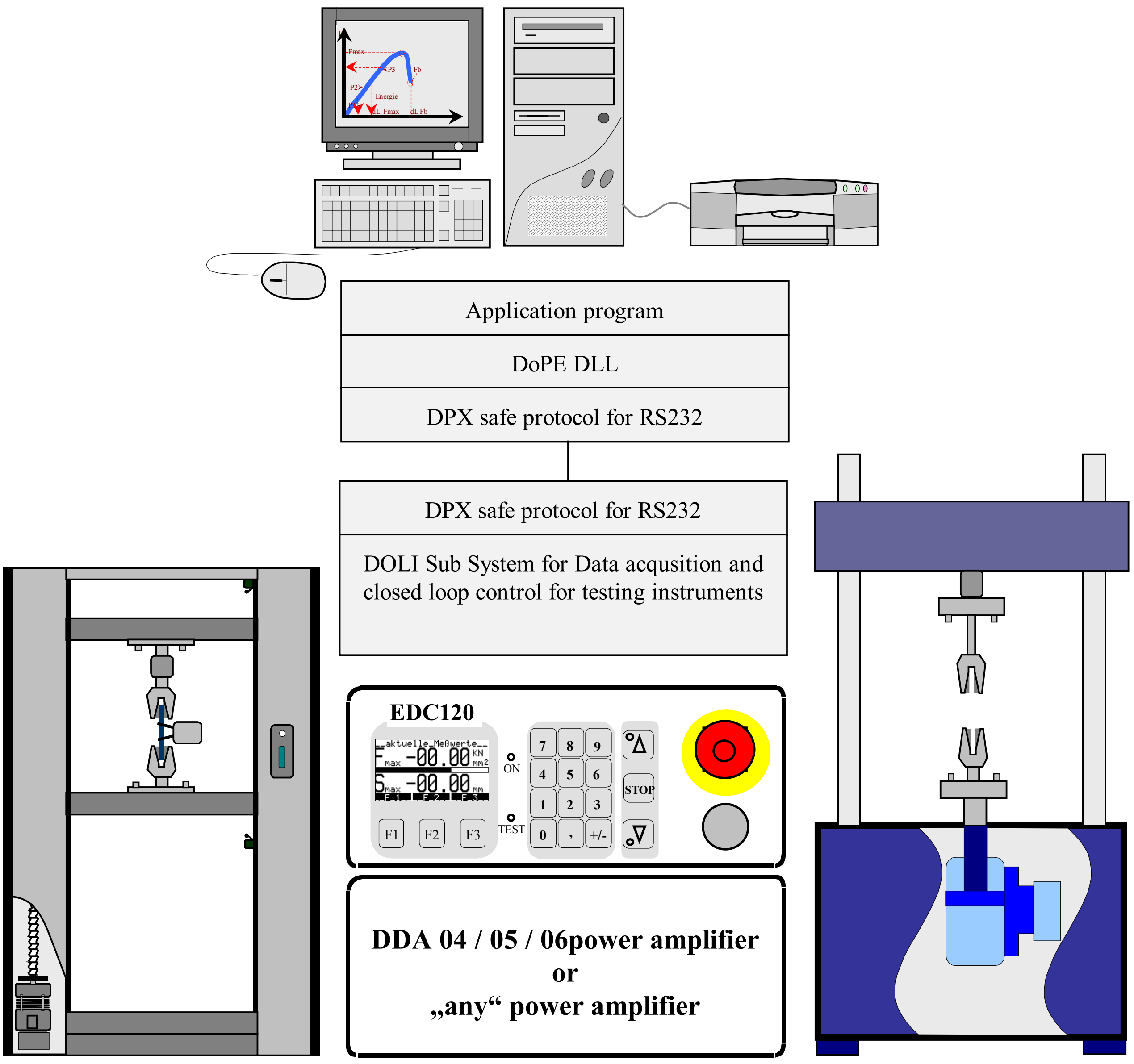
DoPE
Documentation


Application program
DoPE DLL
DPX safe protocol for RS232
DPX safe protocol for RS232
DOLI Sub System for Data acqusition and closed loop control for testing instruments
EDC120
ON
TEST
STOP
F1
F2
F3
DDA 04 / 05 / 06power amplifier
or
„any“ power amplifier

Measuring and Controlling with DOLI Components

DOLI Elektronik GmbH ▫ Fraunhoferstraße 23h ▫ D-80469 Munich/Germany ▫ http://www.doli.de
Tel:++49-89-20243-0 ▫ Fax:++49-89-20243-243 ▫ eMail:info@doli.de

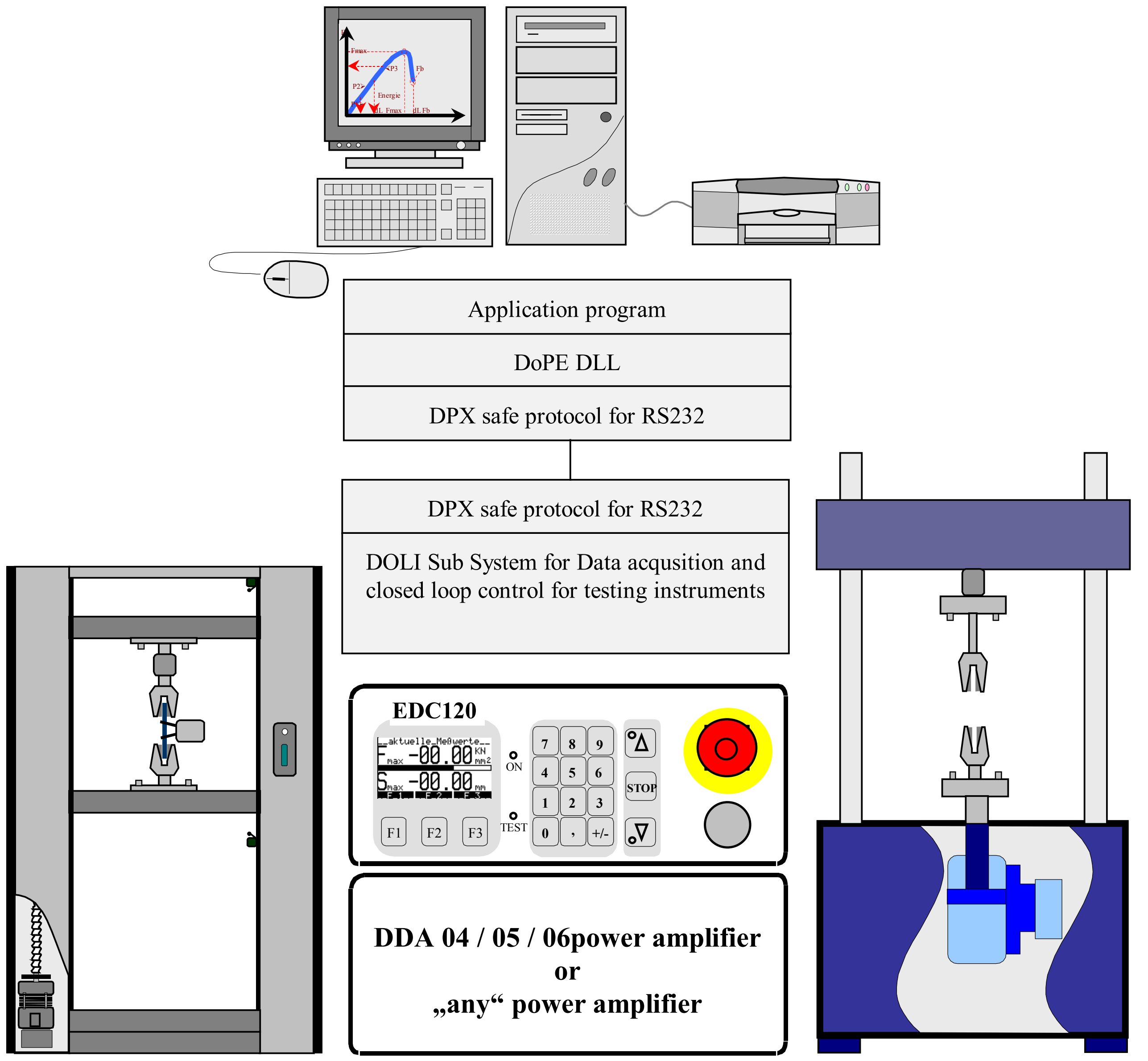
Contents

1 Overview

DOLI has a very powerful software for data acquisition and closed loop control for testing instruments like tensile testers. This software runs on a hardware platform from DOLI like EDC5/25, EDC100, EDC60, EDC120 and others. It is configurable within a wide range to satisfy all tasks at a testing instrument. Normally the system communicates with a PC via a serial RS232 connection. We use a safe protocol called DPX on this serial line to transmit commands and data in both directions.

DoPE represents the interface to access all EDC-functions from any standard PC under Windows ®.

DoPE enables a PC-Programmer to receive measured values from a testing instrument and control it after a few hours.


Application program
DoPE DLL
DPX safe protocol for RS232
DPX safe protocol for RS232
DOLI Sub System for Data acqusition and closed loop control for testing instruments
EDC120
F max -00.00 KN mm2
S max -00.00 mm
F1
F2
F3
ON
TEST
STOP
DDA 04 / 05 / 06power amplifier
or
„any“ power amplifier

1.1 Tasks of DoPE

- DoPE will initialise the Subsystem according to predefined set-up data.
- DoPE gives the application program measuring data in SI-units (N, m etc.)
- DoPE offers a wide range of machine control commands
  Simple commands like:
  „move Up in position control with 0.01 m/sec“
  Complex commands like:
  „move in position control with 0.02 m/sec to 500 N and keep 500 N in load control“
  Cycling commands like:
  „do 100 sinusoidal load cycles with an amplitude of 100N with 3 Hz“
- DoPE reports events like limit switch etc.

DoPE is available as a 32 Bit DLL for Windows 95/98/2000 and Windows NT.

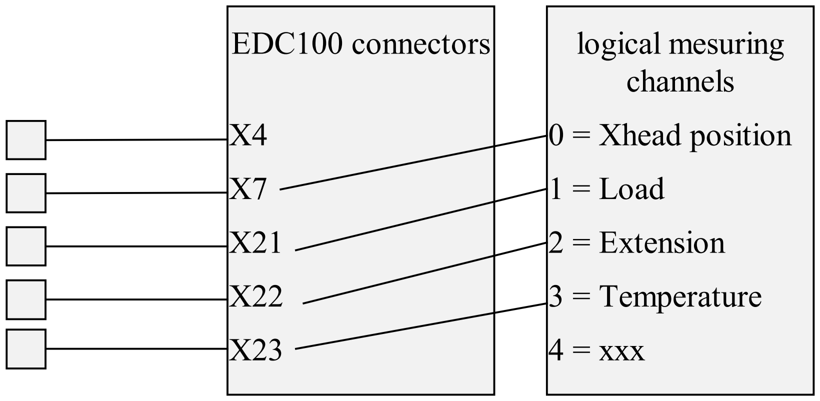
DoPE uses the „DOLI Subsystem for test machine control“. Please use the documentation of this software for more detailed information's

1.2 Initialising

The Subsystem software is able to handle up to:

- sixteen logical measuring channels
- sixteen logical analogue output channels
- ten logical Bit input devices
- ten logical Bit output devices

The above logical channels and devices must be assigned to physical interfaces. For assignment you just have to specify the connector number, the logical measuring channel and some transducer specific data. Our Windows Program “DoSE” (Doli Set-up Editor) helps to edit all necessary set-up data.



All Set-up data will be stored in an EEPROM inside the EDC.

1.3 Supported programming languages

DoPE is delivered as a DLL running under Windows ® operating system. Each programming language that supports the use of DLL's can be used for the application program.
However DOLI supplies only header files for "C", "Delphi/Pascal" and Visual Basic.
Two DLL's are needed to use DoPE: DoPE32.DLL and NTDPX.DLL.

1.3.1 Special considerations for VB-Programmer

- All DoPE function names use "DoPEVB" instead of "DoPE". For instance the name of the function "DoPEOpenLink" is in Visual Basic "DoPEVBOpenLink".
- Do not use the DoPE call-back mechanism while debugging VB program. Only use it for executable versions. Microsoft's VB- developing system has obviously problems debugging call-back mechanism.
- Use "DoPEVB32.DLL" additional to the "DoPE32.DLL".

1.4 Sample programs to initialise DoPE

1.4.1 „C" Program

```
#include "dope.h"
// First step open DoPE connection, define Port number, Baudrate, number of buffers for receive, transmit and data
DoPEErr = DoPEOpenLink ( Port, Baudrate, 10, 10, 10, DoPEAPIVERSION, NULL, &DoPEHdl);
    if(DoPEErr != DoPERR_NOERROR)
    {
    // do error handling
    }
// Here DoPEOpenLink was successful and communication to the EDC is online.
DoPEErr=DoPESetNotification (DoPEHdl, DoPEEVT_ALL, NULL, hWnd, WM_DoPEEvent );
    if(DoPEErr != DoPERR_NOERROR)
    {
    // do error handling
    }
// Select one of the up to four possibile Setups
DoPEErr=DoPESelSetup ( DoPEHdl, 1, Uscale, &lpusTANFirst, &lpusTANLast );
    if(DoPEErr != DoPERR_NOERROR)
    {
    // do error handling
    }
// Here the initialisation sequence for DoPE is done.
// Now you can move the machine e.g. Up with 0.01 m/sec
DoPEErr = DoPEFMove (DoPEHdl, MOVE_UP, 0.01);
// Or you can position Xhead to 0.1m with 0.02 m/sec
DoPEErr = DoPEPos (DoPEHdl, CTRL_POS, 0.02, 0.1);
```

1.4.2 „VB" Program

```
' open connection to EDC
e = DoPEVBOpenLink(ComPort, ComBaudRate, 10, 10, 50, DoPEAPIVERSION, ByVal 0&, DoPEHdl)
If e <> DoPERR_NOERROR Then
    do error handling
' set DoPE notification for DoPE callback
e = DoPEVBSetNotification(DoPEHdl, DoPEEventMask, AddressOf DoPECallBack, hMsgWnd, WM_DoPEEvent)
If e <> DoPERR_NOERROR Then
    do error handling
' select Setup No. 1 and initialize EDC
e = DoPEVBSelSetup(DoPEHdl, 1, UserScale(0), ByVal 0&, ByVal 0&)
```

1.4.3 „Delphi“ Program

```
uses
  DoPE;

const
 CM_CbMessage = WM_USER  // for DoPECallback

// Open Connection to the EDC
   DoPEErr := DoPEOpenLink(Port, Baud, 10, 10, 10, DoPEAPIVERSION, NIL, DoPEHdl);
   If (DoPEErr <> DoPEErr_NOERROR)
   begin
     do Error handling;
   end;
// OpenLink was successful; Communication to the EDC is online
// SetNotification for DoPECallBack
   DoPEErr := DoPESetNotification(DoPEHdl, DoPEEVT_ALL, DoPECallBack-Function, handle, CM_CbMessage);
   If (DoPEErr <> DoPEErr_NOERROR)
   begin
     do Error handling;
   end;
// Select one of the up to four possible Setups
   DoPEErr := DoPESelSetup(DoPEHdl, 1, UsScale, @TanFirst, @TanLast);
   If (DoPEErr <> DoPEErr_NOERROR)
   begin
     do Error handling;
   end;
// Now the initialisation sequence for DoPE is done and you can move the machine...
end;
```

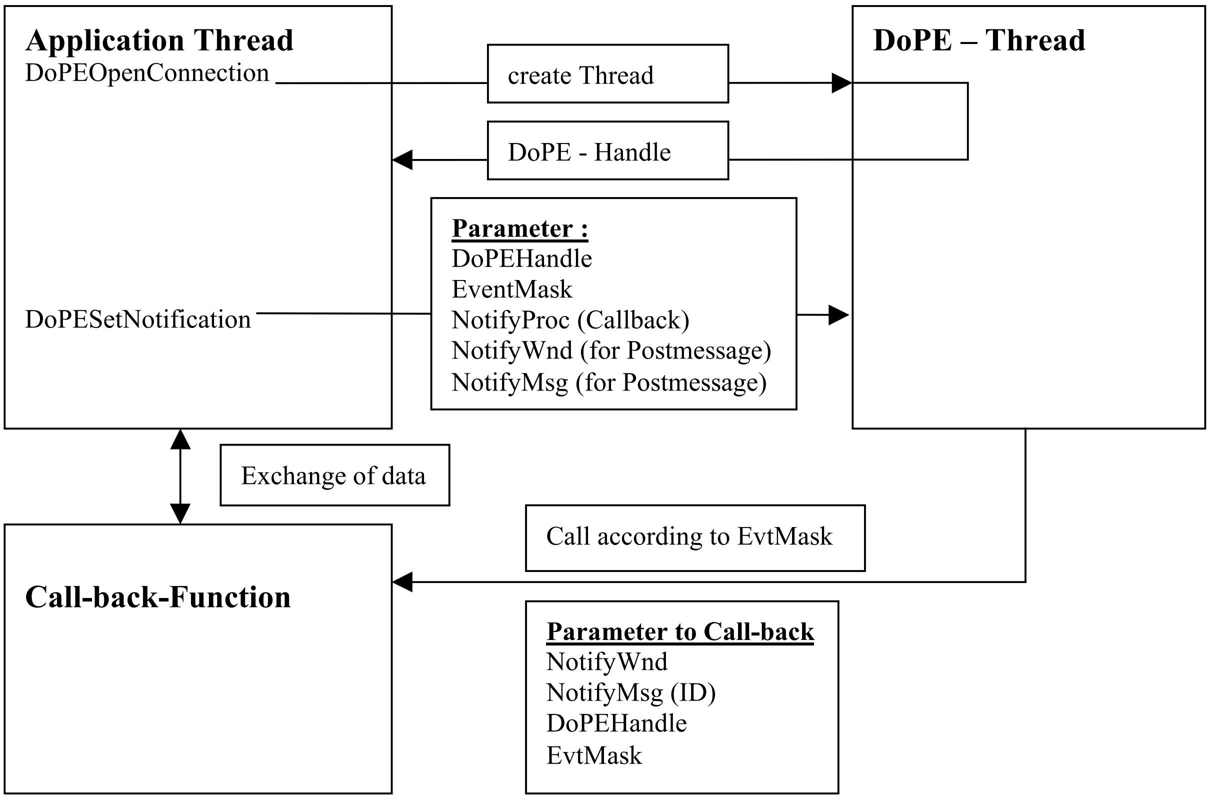
2 Communication Priciples

DoPE represents a library that addresses functions inside the remote EDC-controller. Physically the EDC is connected via a standard COM-Port with the PC. A safe serial protocol is used for communications. This protocol, named DPX, detects ON/OFF-line transitions, transmits messages and measured data packets with ACK/NAC mechanism.
The user of DoPE must always be aware of the remote controller and due to this a asynchronous behaviour of DoPE.

2.1 Open a connection

The application program establishes a connection by calling the **DoPEOpenLink** function. This function creates a thread with a priority depending on the priority of the application thread.
If the application thread has the priority THREAD_PRIORITY_NORMAL, DoPE thread will use the priority THREAD_PRIORITY_ABOVE_NORMAL.
If the application thread has the priority THREAD_PRIORITY_ABOVE_NORMAL, DoPE thread will use the priority THREAD_PRIORITY_TIME_CRITICAL.
The application thread should not start another thread with a higher priority than its own. Otherwise the communication with the EDC might be delayed. This will cause unpredictable problems.
By this mechanism it is safeguarded that DoPE and hence the communication with the EDC has always a higher priority than the application program.

2.2 Establishing a connection with call-back function



After the connection was successfully opened by the DoPEOpenLink command, the communication between DoPE and the application program has to be established by the **DoPESetNotification** command. Here are the parameters for DoPESetNotification if a call-back mechanism is used for communications:

• EventMask	specifies events after which the call-back is activated.

- NotifyProc Specifies the address of the call-back function.
- NotifyWnd This parameter is not used by DoPE, it may be used to pass information between the application program and call-back.
- NotifyMsg This parameter is not used by DoPE, it may be used to pass information between the application program and call-back.

This call-back mechanism enables the application program to do some "real time" work independent from Windows message passing. The application programmer must be aware of the fact that this call-back function can interrupt the application program at any point. Care must be taken of communication between application program and call-back function. The application programmer must use a Windows mechanism like MUTEX to synchronise communication between application program and call-back.
The application programme should not call lengthy functions like graphics, file handling etc. Due to the high priority of the call-back function, only absolutely necessary tasks should be done inside the call-back function.
Tasks like:

- Analysing data record and react on it by sending command to the EDC.
- Copy the data record to a circular buffer. (The application thread reads this data for further analysis, update the online graphic etc.)

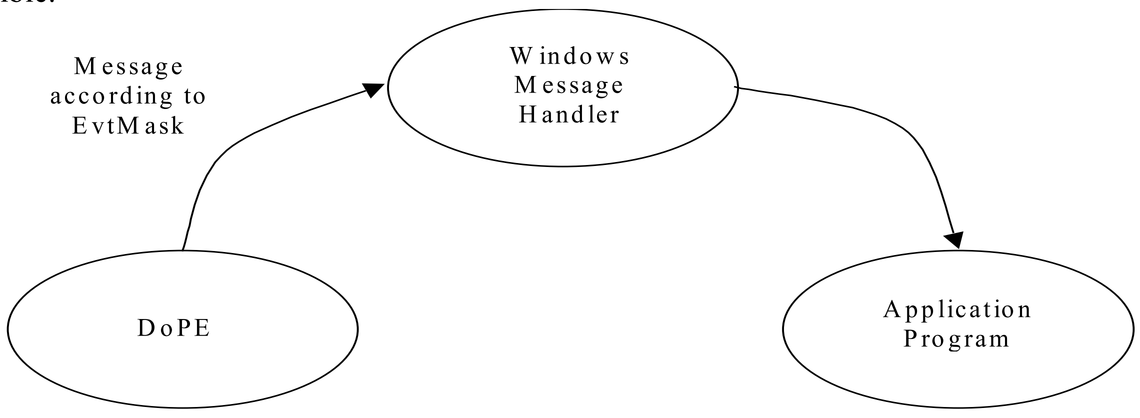
may be done inside the call-back function.

2.3 Establishing a connection using Windows message passing

Alternatively to the above call-back mechanism, a simple message passing may be used to communicate from DoPE with the application program. After the connection was successfully opened by the DoPEOpenLink command, the communication between DoPE and the application program has to be established by **DoPESetNotification** command. Here are the parameters for DoPESetNotification if a Windows message passing mechanism is used for communications:

- EventMask specifies events that causes a message to be send.
- NotifyProc NULL
- NotifyWnd the window handle of the application program window
- NotifyMsg the ID for the message passed by Windows post message mechanism.

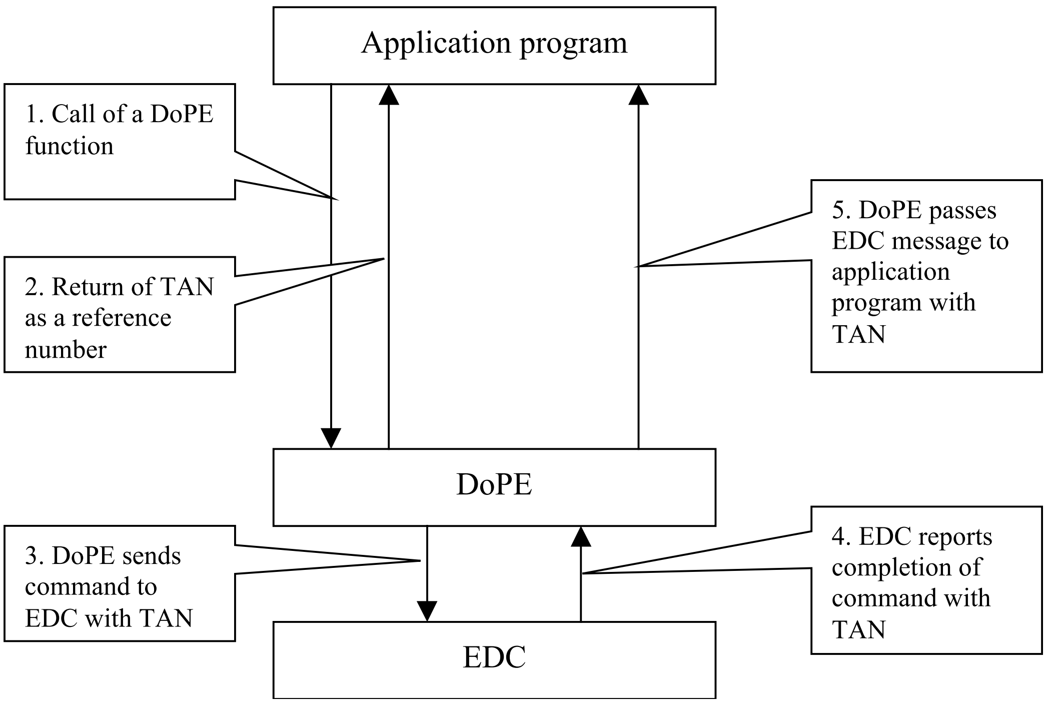
DoPE will use the standard Windows " PostMessage" function to pass messages to the application program. This method is quite easy to handle, but a deterministic time behaviour is not predictable. Messages from DoPE are passed to the Windows message handler and then they are delivered to the application program. The time needed for the Windows message handling depends on the system load. Under normal conditions (only one active application program), this method is also sufficient. But there is one big disadvantage. If a Window on screen is touched by a mouse click and e.g. moved on the screen, no messages can be passed during this period. This makes a "real time" behaviour impossible.



2.4 Transaction Number

Some of the following commands have pointers to a transaction number (lpusTAN) as a parameter. This TAN may be used to identify messages, which are received asynchronously, to the command.
Example: A positioning command is send to the EDC. After 30 seconds the desired position is reached and a message "position reached" is received. This message comes with the TAN of the positioning command.

The TAN is generated by DoPE for all commands, that may cause a message later on.
Some DoPE commands will send more than one command to the EDC. In this cases two pointers to transaction numbers are needed (lpusTANFirst, lpusTANLast). In this case all messages with TAN's between lpusTANFirst and lpusTANLast belong to this single DoPE command.
Of course to analyse the TAN is optional. If NULL-Pointers are passed, no TAN's are returned.


Application program
1. Call of a DoPE function
2. Return of TAN as a reference number
5. DoPE passes EDC message to application program with TAN
DoPE
3. DoPE sends command to EDC with TAN
4. EDC reports completion of command with TAN
EDC

2.5 Synchronous / Asynchronous DoPE Commands

Most of the DoPE command will be passed from DoPE to the connected EDC.
There are synchronous and asynchronous commands.
Synchronous commands wait until the EDC has checked the parameter and returned the result.
Asynchronous commands are just transmitted to the EDC and return to the caller immediately. The result of parameter check inside the EDC is transmitted later if enabled by DoPESetNotification command.
In both cases positioning commands will report completion of the command after destination position is reached.

2.6 Messages from DoPE to the application program

Besides measured values, messages are send from the EDC via DoPE to the application program.
If enabled by DoPESetNotification, a incoming message will create a DoPEEVT_RXAVAIL event which is send to the application program via call-back or PostMessage mechanism.
In both cases the real message must be read by DoPEGetMsg function.
There are three major types of messages possible:

1. **COMMAND_ERROR**
   This message is send after a DoPE command was called and the parameter check has been finished.

2. **RUNTIME_ERROR**
   Errors that occur during the system is running like limit switch, power off and so on.

3. **MOVE_CTRL_MSG**
   Move control messages report normally the completion of a positioning command.

3 Overview of all DoPE Commands

3.1 Initialisation Commands

Command DLL-Ordination No.	Page	Parameter	Name
DoPEOpenLink	**15**	**Establish Communication to a EDC**	
@8		Port	Communication Port No.
		BaudRate	Baudrate for communication
		RcvBuffers	Number of buffers for received messages
		XmitBuffers	Number of buffers for messages to transmit
		DataBuffers	Number of buffers for received data records
		APIVersion	Version of application interface
		Reserved	Reserved parameter
		DoPEHdl	DoPE Handle
DoPECloseLink	**35**	**Close Communication with a EDC**	
@9		DoPEHdl	DoPE Handle
DoPESetNotification	**36**	**Set Notification parameter**	
@12		DoPEHdl	DoPE Handle
		EventMask	Mask of events that cause a notification
		NotifyProc	Address of Call-back function or NULL
		NotifyWnd	Window Handle of application program
		NotifyMsg	ID for message passed from DoPE to application
DoPEInitialize	**37**	**Initialise EDC with active Set-up**	
@14		DoPE Handle	DoPE Handle
		lpusTANFirst	*Pointer to memory for first Transaction Number*
		lpusTANLast	*Pointer to memory for last Transaction Number*

3.2 Setup Commands

Command DLL-Ordination No.	Page	Parameter	Name
DoPEOpenSetup(Sync)	**38**	**Open Set-up inside EDC for Read / Write operations**	
@113 (@1113)		DoPEHdl	DoPE Handle
		SetupNo	Number of Set-up
not required for Sync version		*lpusTAN*	*Pointer to memory for Transaction Number*
DoPECloseSetup(Sync)	**38**	**Close previously opened Set-up**	
@114 (@1114)		DoPEHdl	DoPE Handle
not required for Sync version		*lpusTAN*	*Pointer to memory for Transaction Number*
DoPESetupScale	**38**	**Define User scale for Set-up sensor data**	
@148		DoPEHdl	DoPE Handle
		SetupNo	Number of Set-up
		US	User scale for all set-up sensor data (except DoPESenDef)
DoPERdSetupAll (Sync)	**39**	**Read total Set-up data of opened Set-up**	
@111 (@1111)		DoPEHdl	DoPE Handle
		SetupNo	Number of Set-up
		DoPESetup	Pointer to memory for Set-up data
not required for Sync version		*lpusTANFirst*	*Pointer to memory for first Transaction Number*
not required for Sync version		*lpusTANLast*	*Pointer to memory for last Transaction Number*

Command DLL-Ordination No.	Page	Parameter	Name
DoPEWrSetupAll (Sync)	**39**	**Write total set-up**	
@112 (@1112)		DoPEHdl	DoPE Handle
		SetupNo	Number of Set-up
		DoPESetup	Pointer to memory with Set-up data
not required for Sync version		*lpusTANFirst*	*Pointer to memory for first Transaction Number*
not required for Sync version		*lpusTANLast*	*Pointer to memory for last Transaction Number*
DoPERdSetupNumber	**40**	**Read active set-up number**	
@110		DoPEHdl	DoPE Handle
		SetupNo	Pointer to memory for Setup number
DoPERdSensorDef	**40**	**Read definition data of a sensor**	
@92		DoPEHdl	DoPE Handle
		SensorNo	Sensor Number
		SensorDef	Pointer to memory for Sensor definition data
DoPEWrSensorDef (Sync)	**40**	**Write definition data of a sensor**	
@101 (@1101)		DoPEHdl	DoPE Handle
		SensorNo	Sensor Number
		SensorDef	Pointer to memory with Sensor definition data
not required for Sync version		*lpusTAN*	*Pointer to memory for Transaction Number*
DoPERdCtrlSensorDef	**41**	**Read definition data for closed loop control of a sensor**	
@93		DoPEHdl	DoPE Handle
		SensorNo	Sensor Number
		CtrlSensorDef	Pointer to memory for closed loop control data for Sensor
DoPEWrCtrlSensorDef (Sync)	**41**	**Write definition data for closed loop control of a sensor**	
@102 (@1102)		DoPEHdl	DoPE Handle
		SensorNo	Sensor Number
		CtrlSensorDef	Pointer to memory with closed loop control data for Sensor
not required for Sync version		*lpusTAN*	*Pointer to memory for Transaction Number*
DoPERdOutChannelDef	**42**	**Read definition data of an analogue output channel**	
@94		DoPEHdl	DoPE Handle
		OutChNo	Number of analogue output channel
		OutChDef	Pointer to memory for output channel definition data
DoPEWrOutChannelDef (Sync)	**42**	**Write definition data of an analogue output channel**	
@103 (@1103)		DoPEHdl	DoPE Handle
		OutChNo	Number of analogue output channel
		OutChDef	Pointer to memory with output channel definition data
not required for Sync version		*lpusTAN*	*Pointer to memory for Transaction Number*
DoPERdBitInDef	**42**	**Read definition data of Bit input channel**	
@96		DoPEHdl	DoPE Handle
		BitInNo	Number of Bit input channel
		BitInDef	Pointer to memory for Bit input channel definition data
DoPEWrBitInDef (Sync)	**43**	**Write definition data of Bit input channel**	
@105 (@1105)		DoPEHdl	DoPE Handle
		BitInNo	Number of Bit input channel
		BitInDef	Pointer to memory with Bit input channel definition data
not required for Sync version		*lpusTAN*	*Pointer to memory for Transaction Number*

Command DLL-Ordination No.	Page	Parameter	Name
DoPERdBitOutDef	**43**	**Read definition data of Bit output channel**	
@95		DoPEHdl	DoPE Handle
		BitOutNo	Number of Bit input channel
		BitOutDef	Pointer to memory for Bit output channel definition data
DoPEWrBitOutDef (Sync)	**43**	**Write definition data of Bit output channel**	
@104 (@1104)		DoPEHdl	DoPE Handle
		BitOutNo	Number of Bit output channel
		BitOutDef	Pointer to memory with Bit output channel definition data
not required for Sync version		*lpusTAN*	*Pointer to memory for Transaction Number*
DoPERdMachineDef	**44**	**Read definition data of machine**	
@97		DoPEHdl	DoPE Handle
		MachineDef	Pointer to memory for machine definition data
DoPEWrMachineDef (Sync)	**44**	**Write definition data of machine**	
@106 (@1106)		DoPEHdl	DoPE Handle
		MachineDef	Pointer to memory with machine definition data
not required for Sync version		*lpusTAN*	*Pointer to memory for Transaction Number*
DoPERdLinTbl	**45**	**Read machine load linearisation table**	
@96		DoPEHdl	DoPE Handle
		LinTblFalse	Pointer to memory for FALSE values
		LinTblTrue	Pointer to memory for TRUE values
DoPEWrLinTbl (Sync)	**45**	**Write machine load linearisation table**	
@105 (@1105)		DoPEHdl	DoPE Handle
		LinTblFalse	Pointer to memory with FALSE values
		LinTblTrue	Pointer to memory with TRUE values
not required for Sync version		*lpusTANFirst*	*Pointer to memory for first Transaction Number*
not required for Sync version		*lpusTANLast*	*Pointer to memory for last Transaction Number*
DoPERdGeneralData	**46**	**Read general data**	
@100		DoPEHdl	DoPE Handle
		GeneralData	Pointer to memory for general data
DoPEWrGeneralData (Sync)	**46**	**Write general data**	
@109 (@1109)		DoPEHdl	DoPE Handle
		GeneralData	Pointer to memory with general data
not required for Sync version		*lpusTAN*	*Pointer to memory for Transaction Number*
DoPESelSetup	**47**	**Select machine Set-up**	
@13		DoPEHdl	DoPE Handle
		SetupNo	Machine Set-up number
		US	User scale for all measuring channels
		lpusTANFirst	Pointer to memory for first Transaction Number
		lpusTANLast	Pointer to memory for last Transaction Number
DoPERdSensorInfo	**47**	**Read Sensor information**	
@91		DoPEHdl	DoPE Handle
		SensorNo	Sensor Number
		Info	Pointer to memory for information data of this sensor
DoPERdSysUserData	**48**	**Read system user data**	
@99		DoPEHdl	DoPE Handle
		SysUsrData	Pointer to memory for system user data
		Length	Length of system user data in bytes

Command DLL-Ordination No.	Page	Parameter	Name
DoPEWrSysUserData (Sync)	**48**	**Write system user data**	
@108 (@1108)		DoPEHdl	DoPE Handle
		SysUsrData	Pointer to memory for system user data
		Length	Length of system user data in bytes
not required for Sync version		*lpusTAN*	*Pointer to memory for Transaction Number*
DoPERdSenUserData	**49**	**Read sensor user data**	
@140		DoPEHdl	DoPE Handle
		SensorNo	Sensor number
		SenUsrData	Pointer to memory for sensor user data
		Length	Length of system user data in bytes
DoPEWrSenUserData (Sync)	**49**	**Write sensor user data**	
@141 (@1141)		DoPEHdl	DoPE Handle
		SensorNo	Sensor number
		SenUsrData	Pointer to memory with sensor user data
		Length	Length of sensor user data in bytes
not required for Sync version		*lpusTAN*	*Pointer to memory for Transaction Number*
DoPERdSensorUserData	**49**	**Read sensor user data (max 128 Byte).**	
@168		DoPEHdl	DoPE Handle
		Connector	Connector number of sensor
		*SenUsrData	Pointer for sensor user data
		Length	User data buffer length in BYTE's
DoPEWrSensorUserData	**50**	**Write sensor user data (max 128 Byte).**	
@169		DoPEHdl	DoPE Handle
		Connector	Connector number of sensor
		*SenUsrData	Pointer for sensor user data
		Length	User data buffer length in BYTE's

3.3 Message and Data handling

Command DLL-Ordination No.	Page	Parameter	Name
DoPESendMsg (Sync)	**51**	**Send a message to EDC-subsystem**	
@17 (@1017)		DoPEHdl	DoPE Handle
		Buffer	Pointer to memory with the message
		Length	Length of sensor user data in bytes
not required for Sync version		lpusTAN	Pointer to memory for Transaction Number
DoPEGetMsg	**51**	**Get message from receive buffer**	
@18		DoPEHdl	DoPE Handle
		Buffer	Pointer to memory for the message
		BufSize	Length of Buffer in bytes
		Length	Length of received message
DoPEGetData	**57**	**Get data record**	
@19		DoPEHdl	DoPE Handle
		Buffer	Pointer to memory for data record
DoPECurrentData	**58**	**Get current data record**	
@146		DoPEHdl	DoPE Handle
		Buffer	Pointer to memory for data record
DoPEClearReceiver	**57**	**Clear (discard) all received messages**	
@20		DoPEHdl	DoPE Handle

Command DLL-Ordination No.	Page	Parameter	Name
DoPEClearTransmitter @21	**58**	**Clear (discard) all messages to be sent** DoPEHdl	 DoPE Handle
DoPEGetState @16	**58**	**Read DoPE state (Communication state, Number of received messages, Number of messages to be transmitted)** DoPEHdl Status	 DoPE Handle Pointer to memory for state record
DoPEGetErrors @15	**59**	**Read current error counter values** DoPEHdl Error	 DoPE Handle Pointer to memory for error record

3.4 Configuration Commands

Command DLL-Ordination No.	Page	Parameter	Name
DoPERdVersion @80	**60**	**Read Version strings** DoPEHdl Version	 DoPE Handle Pointer to memory for Version strings
DoPERefr (Sync) @22 (@1022) *not required for Sync version*	**60**	**Set time base for data acquisition** DoPEHdl Refresh *lpusTAN*	 DoPE Handle Time base *Pointer to memory for Transaction Number*
DoPESetTime (Sync) @23 (@1023) *not required for Sync version*	**60**	**Set time** DoPEHdl Time *lpusTAN*	 DoPE Handle Time value *Pointer to memory for Transaction Number*
DoPETransmitData (Sync) @24 (@1024) *not required for Sync version*	**61**	**Enable / Disable transmission of data record** DoPEHdl Enable *lpusTAN*	 DoPE Handle TRUE = Enable transmission, FALSE = Disable transmission *Pointer to memory for Transaction Number*
DoPEIntgr (Sync) @30 (@1030) *not required for Sync version*	**61**	**Set Integration time for an analogue sensor** DoPEHdl SensorNo Intgr *lpusTAN*	 DoPE Handle Sensor number Integration time for the sensor *Pointer to memory for Transaction Number*

DoPECtrlTestValues
@180
	62	**Enable/Disable controller test values in the DoPEData record**
		DoPEHdl
		State
			
			DoPE Handle
			TRUE = Enable
			FALSE = Disable
			controller test values in the DoPEData record

Command DLL-Ordination No.	Page	Parameter	Name
DoPEShieldLimit (Sync)	65	**Activate safety shield (screen)**	
@127 (@1127)		DoPEHdl	DoPE Handle
		SensorNo	Sensor number (normally load)
		UprLock	Above this value shield must be closed
		UprUnLock	Below this value shield is unlocked
		LwrUnLock	Above this value shield is unlocked
		LwrLock	Below this value shield must be closed
		CtrlLimit	Control mode for SpeedLimit (normally position)
		SpeedLimit	Maximum allowed speed if shield is opened
		CtrlAction	Control mode for Action
		Action	Action to be activated if the limits are reached
not required for Sync version		*lpusTAN*	*Pointer to memory for Transaction Number*
DoPEShieldDisable (Sync)	65	**Deactivate safety shield (screen)**	
@128 (@1128)		DoPEHdl	DoPE Handle
not required for Sync version		*lpusTAN*	*Pointer to memory for Transaction Number*
DoPEShieldLock (Sync)	66	**Lock or unlock the shield (screen)**	
@129 (@1129)		DoPEHdl	DoPE Handle
		State	TRUE = Lock, FALSE = UNLOCK shield
not required for Sync version		*lpusTAN*	*Pointer to memory for Transaction Number*
DoPESetCheck (Sync) **DoPESetCheckX (Sync)**	66	**Activate measuring channel supervision**	
@40 (@1040)		DoPEHdl	DoPE Handle
@144 (@1144)		CheckId	ID of this check, use the CheckId constants
		SensorNo	Sensor to be supervised
		Limit	Limit to be supervised
only for DoPESetCheckX		*Tare*	*Tare value needed for* PercentMin and PercentMax
		Mode	Below, Above, PercentMin, PercentMax, AbsMax, AbsMin
		Action	Action to be activated if the check hits (parameter for action are listed below)
		Ctrl	Control mode for selected action
		Acc	Acceleration
		Speed	Maximum speed
		Dec	Deceleration
		Destination	Final destination
		lpusTAN	Pointer to memory for Transaction Number
DoPEClrCheck (Sync)	67	**Clear measuring channel supervision**	
@41 (@1041)		DoPEHdl	DoPE Handle
		CheckId	ID of the supervision to be cleared, use the CheckId constants
		lpusTAN	Pointer to memory for Transaction Number
DoPESetCheckLimit (Sync)	68	**Activate measuring channel supervision and set digital output**	
@125 (@1125)		DoPEHdl	DoPE Handle
		SensorNo	Sensor to be supervised
		UprLimitSet	Set digital output above this value
		UprLimitReset	Reset digital output below this value
		LwrLimitReset	Reset digital output above this value
		LwrLimitSet	Set digital output below this value
not required for Sync version		*lpusTAN*	*Pointer to memory for Transaction Number*
DoPEClrCheckLimit (Sync)	68	**Deactivate measuring channel supervision with digital output**	
@126 (@1126)		DoPEHdl	DoPE Handle
not required for Sync version		*lpusTAN*	*Pointer to memory for Transaction Number*

Command DLL-Ordination No.	Page	Parameter	Name
DoPEPosPID (Sync)	**69**	**Set parameter for closed loop position controller**	
@176 (@1176)		DoPEHdl	DoPE Handle
		MoveCtrl	Control mode (CTRL_POS, CTRL_LOAD, CTRL_EXTENSION)
		P	Proportional gain of the position controller
		I	Integration time constant (normally ZERO!!!)
		D	Derivative time constant (for future use)
not required for Sync version		*lpusTAN*	*Pointer to memory for Transaction Number*
DoPESpeedPID (Sync)	**69**	**Set parameter for closed loop speed controller**	
@177 (@1177)		DoPEHdl	DoPE Handle
		MoveCtrl	Control mode (CTRL_POS, CTRL_LOAD, CTRL_EXTENSION)
		P	Proportional gain of the position controller
		I	Integration time
		D	Derivative time constant
not required for Sync version		*lpusTAN*	*Pointer to memory for Transaction Number*
DoPEPosFeedForward (Sync)	**70**	**Set feed forward gain of closed loop controller.**	
@178 (@1178)		DoPEHdl	DoPE Handle
		MoveCtrl	Control mode (CTRL_POS, CTRL_LOAD, CTRL_EXTENSION)
		P	Gain for feed forward control
not required for Sync version		*lpusTAN*	*Pointer to memory for Transaction Number*
DoPEDestWnd (Sync)	**71**	**Definitions for destination window**	
@75 (@1075)		DoPEHdl	DoPE Handle
		MoveCtrl	Control mode (CTRL_POS, CTRL_LOAD, CTRL_EXTENSION)
		WndSize	Size of destination window
		WndTime	Time for destination window
not required for Sync version		*lpusTAN*	*Pointer to memory for Transaction Number*
DoPESft (Sync)	**72**	**Definitions of limits supervised by software (softend)**	
@76 (@1076)		DoPEHdl	DoPE Handle
		MoveCtrl	Control mode (CTRL_POS, CTRL_LOAD, CTRL_EXTENSION)
		UpperSft	Upper soft limit
		LowerSft	lower soft limit
		Reaction	Action to be activated after softend is reached
not required for Sync version		*lpusTAN*	*Pointer to memory for Transaction Number*
DoPECtrlError (Sync)	**73**	**Define maximum error signal for closed loop controller**	
@77 (@1077)		DoPEHdl	DoPE Handle
		MoveCtrl	Control mode (CTRL_POS, CTRL_LOAD, CTRL_EXTENSION)
		Error	Maximum difference between command and signal
		Reaction	Action to be activated after error is reached
not required for Sync version		*lpusTAN*	*Pointer to memory for Transaction Number*
DoPESetDither (Sync)	**74**	**Set Parameter for dither**	
@151 (@1151)		DoPEHdl	DoPE Handle
		Output	Number of analogue output channel
		Frequency	Dither frequency
		Amplitude	Dither aplitude
not required for Sync version		*lpusTAN*	*Pointer to memory for Transaction Number*
DoPECtrlSpeedTimeBase (Sync)	**74**	**Define Time Base for Speed detection (for calculated channels !)**	
@134 (@1134)		DoPEHdl	DoPE Handle
		Time	Time base for speed detection
not required for Sync version		*lpusTAN*	*Pointer to memory for Transaction Number*

Command **DLL-Ordination No.**	**Page**	**Parameter**	**Name**
DoPECal (Sync)	**74**	**Calibrate (offset and gain) analogue measuring channel(s)**	
@31 (@1031)		DoPEHdl	DoPE Handle
		SensorBits	Bit 0 = 1 Calibrate Sensor 0 and so on
not required for Sync version		*lpusTAN*	*Pointer to memory for Transaction Number*
DoPEZeroCal (Sync)	**74**	**Calibrate (only offset) analogue measuring channel(s)**	
@32 (@1032)		DoPEHdl	DoPE Handle
		SensorBits	Bit 0 = 1 Calibrate Sensor 0 and so on
not required for Sync version		*lpusTAN*	*Pointer to memory for Transaction Number*
DoPESetBasicTare (Sync)	**76**	**Set basic tare of the measuring channel**	
@25 (@1025)		DoPEHdl	DoPE Handle
		SensorNo	Sensor Number
		Mode	Mode = 0 -> BasicTare represents the value at this point
			Mode = 1 -> BasicTare will be subtracted
		BasicTare	Basic tare value
not required for Sync version		*lpusTANFirst*	*Pointer to memory for first Transaction Number*
not required for Sync version		*lpusTANFirst*	*Pointer to memory for last Transaction Number*
DoPEGetBasicTare	**77**	**Read basic tare of the measuring channel**	
@27		DoPEHdl	DoPE Handle
		SensorNo	Sensor Number
		BasicTare	Pointer to memory for basic tare value
DoPESetTare	**76**	**Set tare of the measuring channel**	
@26		DoPEHdl	DoPE Handle
		SensorNo	Sensor Number
		Tare	Tare value
DoPEGetTare	**77**	**Read tare of the measuring channel**	
@28		DoPEHdl	DoPE Handle
		SensorNo	Sensor Number
		Tare	Pointer to memory for tare value
DoPEConfPeakValue (Sync)	**82**	**Configure peak values to measuring data record.**	
@131 (@1131)		DoPEHdl	DoPE Handle
		PositionMin	Position of Minimum value of XHead Position
		PositionMax	Position of Maximum value of XHead Position
		LoadMin	Position of Minimum value of Load
		LoadMax	Position of Maximum value of Load
		ExtensionMin	Position of Minimum value of Extension
		ExtensionMax	Position of Maximum value of Extension
not required for Sync version		*lpusTANFirst*	*Pointer to memory for first Transaction Number*
not required for Sync version		*lpusTANLast*	*Pointer to memory for last Transaction Number*
DoPEPeakValueTime (Sync)	**82**	**Set reset time for peak value detection.**	
@132 (@1132)		DoPEHdl	DoPE Handle
		Time	Reset time for peak value detection
not required for Sync version		*lpusTAN*	*Pointer to memory for Transaction Number*
DoPEConfCMcSpeed (Sync)	**79**	**Configure calculated speed to measuring data record.**	
@135 (@1135)		DoPEHdl	DoPE Handle
		CalculatedSensorNo	Position in measuring data record for the calculated speed value
		SensorNo	SensorNo to calculate speed of
		IntegrationTime	Integration time for data acquisition (only for analogue channels)
		Timebase	Time base for speed calculation (maximum 2.56 sec.)
not required for Sync version		*lpusTANFirst*	*Pointer to memory for first Transaction Number*
not required for Sync version		*lpusTANLast*	*Pointer to memory for last Transaction Number*

Command DLL-Ordination No.	Page	Parameter	Name
DoPEConfCMcCommandSpeed (Sync)	**79**	**Configure calculated speed of command output to data record.**	
@136 (@1136)		DoPEHdl	DoPE Handle
		CalculatedSensorNo	Position in measuring data record for the calculated command speed
		Timebase	Time base for speed calculation (maximum 2.56 sec.)
not required for Sync version		*lpusTANFirst*	*Pointer to memory for first Transaction Number*
not required for Sync version		*lpusTANLast*	*Pointer to memory for last Transaction Number*
DoPEConfCMcGradient (Sync)	**80**	**Configure calculated gradient between two measured values to measuring data record.**	
@137 (@1137)		DoPEHdl	DoPE Handle
		CalculatedSensorNo	Position in measuring data record for the calculated gradient.
		DividentSensorNo	Sensor No. for Dividend
		DivisorSensorNo	Sensor No. for Divisor
		IntegrationTime	Integration time for data acquisition (only for analogue channels)
		Timebase	Time base for gradient calculation (maximum 2.56 sec.)
not required for Sync version		*lpusTANFirst*	*Pointer to memory for first Transaction Number*
not required for Sync version		*lpusTANLast*	*Pointer to memory for last Transaction Number*
DoPEClearCMc (Sync)	**80**	**Clear calculated measuring channel.**	
@138 (@1138)		DoPEHdl	DoPE Handle
		CalculatedSensorNo	Position in measuring data record for the calculated value
not required for Sync version		*lpusTANFirst*	*Pointer to memory for first Transaction Number*
not required for Sync version		*lpusTANLast*	*Pointer to memory for last Transaction Number*
DoPEMc2Output (Sync)	**81**	**Output of a measured value to a analogue output channel**	
@142 (@1142)		DoPEHdl	DoPE Handle
		Mode	Mode (OFF, HIGH_PRIORITY, LOW_PRIORITY)
		SensorNo	Sensor number
		OffsetSensor	Offset value to be subtracted from measured value
		Output	Output channel number
		OffsetOutput	Offset to be subtracted
		Scale	Scale
not required for Sync version		*lpusTAN*	*Pointer to memory for Transaction Number*
DoPESetRefSignalMode	**83**	**Set Reference signal mode for incremental sensors (only EDC60/120)**	
@154		DoPEHdl	DoPE Handle
		SensorNo	Sensor number
		Mode	REFSIG_NON: never, REFSIG_ON: always, REFSIG_ONCE: only once
		lpusTAN	Pointer to memory for Transaction Number
DoPESetRefSignalTare	**83**	**Set Reference signal mode for incremental sensors (only EDC60/120)**	
@155		DoPEHdl	DoPE Handle
		SensorNo	Sensor number
		Mode	1 = At the next reference signal the measuring channel, will be set to the tare value 0 = Reference signals don't affect the measuring channel
		Tare	Value for the measuring channel
		lpusTAN	Pointer to memory for Transaction Number
DoPEWrSensorMsg(Synch)	**84**	**This function can be used to send a message to a serial sensor. (only EDC60/120)**	
@159 (@1159)		DoPEHdl	DoPE Handle
		SensorNo	Sensor number
		*Buffer	Pointer to message to transmit
		Length	Message length in Byte's
		lpusTAN	Pointer to memory for Transaction Number

3.5 Input / Output-Commands

Command **DLL-Ordination No.**	**Page**	**Parameter**	**Name**
DoPESetOutput (Sync)	**85**	**Set an analogue output channel**	
@33 (@1033)		DoPEHdl	DoPE Handle
		Output	Output channel number
		Value	New value of output channel in % of max. value
not required for Sync version		*lpusTAN*	*Pointer to memory for Transaction Number*
DoPESetOutChannelOffset(Synch)	**85**	**Set an analogue output channel offset (only EDC60/120)**	
@156 (@1156)		DoPEHdl	DoPE Handle
		Output	Number of analogue output channel
		Offset	New offset value of output channel
		lpusTAN	Pointer to memory for Transaction Number
DoPEOfflineActionOutput(Synch)	**85**	**Definition of an action for an initialised analogue output channel after EDC has detected offline (only EDC60/120)**	
@ 180 (@1180)		DoPEHdl	DoPE Handle
		OutputNo	Number of analogue output channel
		Mode	DO_NOTHING (0): Don't modify this digital output USE_INIT_VALUE (1): Use Initial value from set-up after offline USE_VALUE (2): Use defined value after offline
		Value	Output value used in USE_VALUE mode in % of max. value
		lpusTAN	Pointer to memory for Transaction Number
DoPESetB (Sync)	**87**	**Set an bit output channel**	
@29 (@1029)		DoPEHdl	DoPE Handle
		BitOutputNo	Digital output channel number
		SetB	These bits will be set
		ResB	These bits will be reset
		FlashB	These bits will flash
not required for Sync version		*lpusTAN*	*Pointer to memory for Transaction Number*
DoPECalOut (Sync)	**87**	**Activate (Deactivate) digital calibration contact**	
@82 (@1082)		DoPEHdl	DoPE Handle
		Cal	TRUE = Activate, FALSE = Deactivate
not required for Sync version		*lpusTAN*	*Pointer to memory for Transaction Number*
DoPEBeep (Sync)	**88**	**Activate (Deactivate) beeper**	
@83 (@1083)		DoPEHdl	DoPE Handle
		Beep	TRUE = Activate, FALSE = Deactivate
not required for Sync version		*lpusTAN*	*Pointer to memory for Transaction Number*
DoPELed (Sync)	**88**	**Switch On/Off/Flash LED's at EDC frontpanel**	
@84 (@1084)		DoPEHdl	DoPE Handle
		LedOn	These LED's will be set
		LedOff	These LED's will be reset
		LedFlash	These LED's 'flash'
not required for Sync version		*lpusTAN*	*Pointer to memory for Transaction Number*
DoPEUniOut (Sync)	**88**	**Activate/Deactivate universal digital output bits at EDC100, EDC60, ECC120**	
@85 (@1085)		DoPEHdl	DoPE Handle
		Output	Bit 0 .. 3 represent the four digital outputs
not required for Sync version		*lpusTAN*	*Pointer to memory for Transaction Number*
DoPEBypass (Sync)	**89**	**Activate/Deactivate bypass output**	
@33 (@1033)		DoPEHdl	DoPE Handle
		Bypass	TRUE = Activate, FALSE = Deactivate
not required for Sync version		*lpusTAN*	*Pointer to memory for Transaction Number*

Command **DLL-Ordination No.**	**Page**	**Parameter**	**Name**
DoPERdBitInput	**89**	**Directly read digital input device**	
@115		DoPEHdl	DoPE Handle
		Connector	Connector Number
		Value	Pointer to memory for digital input value
DoPEWrBitOutput (Sync)	**89**	**Directly write digital output device**	
@116 (@1116)		DoPEHdl	DoPE Handle
		Connector	Connector Number
		Value	Digital output value
not required for Sync version		*lpusTAN*	*Pointer to memory for Transaction Number*
DoPEOfflineActionBitOutput (Synch)	**90**	**Definition of an action for an initialised digital output after EDC has detected offline (only EDC60/120)**	
@ 179 (@1179)		DoPEHdl	DoPE Handle
		BitOutputNo	Number of bit output device
		Mode	DO_NOTHING (0): Don't modify this digital output USE_INIT_VALUE (1): Use Initial value from set-up after offline USE_VALUE (2): Use defined value after offline
		Value	Output value used in USE_VALUE mode
		lpusTAN	Pointer to memory for Transaction Number
DoPEDspClear (Sync)	**91**	**Clear LCD-display at EDC frontpanel**	
@87 (@1087)		DoPEHdl	DoPE Handle
not required for Sync version		*lpusTAN*	*Pointer to memory for Transaction Number*
DoPEDspHeadLine (Sync)	**91**	**Display headline on LCD-display at EDC frontpanel**	
@88 (@1088)		DoPEHdl	DoPE Handle
		HeadLine	Pointer to headline text
not required for Sync version		*lpusTAN*	*Pointer to memory for Transaction Number*
DoPEDspFKeys (Sync)	**92**	**Display function keys on LCD-display at EDC frontpanel**	
@89 (@1089)		DoPEHdl	DoPE Handle
		FKeys	Pointer to text for function keys
not required for Sync version		*lpusTAN*	*Pointer to memory for Transaction Number*
DoPEDspMValue (Sync)	**92**	**Display data and dimensions on LCD-display at EDC frontpanel**	
@90 (@1090)		DoPEHdl	DoPE Handle
		Value1	Pointer to character string for first value
		Value2	Pointer to character string for second value
		Dim1	Pointer to character string for first dimension
		Dim2	Pointer to character string for second dimension
not required for Sync version		*lpusTAN*	*Pointer to memory for Transaction Number*
DoPEDspBeamScreen (Sync)	**92**	**Display frame & beam on LCD-display at EDC frontpanel**	
@122 (@1122)		DoPEHdl	DoPE Handle
		Value	Value of the beam in %
not required for Sync version		*lpusTAN*	*Pointer to memory for Transaction Number*
DoPEDspBeamValue (Sync)	**93**	**Display beam on LCD-display at EDC frontpanel**	
@122 (@1122)		DoPEHdl	DoPE Handle
		Value	Value of the beam in %
not required for Sync version		*lpusTAN*	*Pointer to memory for Transaction Number*

3.6 Movement commands

Command DLL-Ordination No.	Page	Parameter	Name
DoPEHalt (Sync) @46 (@1046)	**94**	**Halt movement of cross-head in the specified control mode**	
		DoPEHdl	DoPE Handle
		MoveCtrl	Control mode (CTRL_POS, CTRL_LOAD, CTRL_EXTENSION)
		lpusTAN	Pointer to memory for Transaction Number
DoPESHalt (Sync) @139 (@1139)	**96**	**Instantly halt movement of cross-head in position control mode**	
		DoPEHdl	DoPE Handle
		lpusTAN	Pointer to memory for Transaction Number
DoPEHalt_A (Sync) @51 (@1051)	**95**	**Halt movement of cross-head in the specified control mode and deceleration**	
		DoPEHdl	DoPE Handle
		MoveCtrl	Control mode (CTRL_POS, CTRL_LOAD, CTRL_EXTENSION)
		Dec	Deceleration
		lpusTAN	Pointer to memory for Transaction Number
DoPEHaltW (Sync) @67 (@1067)	**95**	**Halt movement of cross-head in the specified control mode with delay time**	
		DoPEHdl	DoPE Handle
		MoveCtrl	Control mode (CTRL_POS, CTRL_LOAD, CTRL_EXTENSION)
		Delay	Delay time (used in combined commands)
		lpusTAN	Pointer to memory for Transaction Number
DoPEHaltW_A (Sync) @68 (@1068)	**96**	**Halt movement of cross-head in the specified control mode and deceleration with delay time**	
		DoPEHdl	DoPE Handle
		MoveCtrl	Control mode (CTRL_POS, CTRL_LOAD, CTRL_EXTENSION)
		Dec	Deceleration
		Delay	Delay time (used in combined commands)
		lpusTAN	Pointer to memory for Transaction Number
DoPEPos (Sync) @42 (@1042)	**97**	**Move cross-head in the specified control mode and speed to the given destination**	
		DoPEHdl	DoPE Handle
		MoveCtrl	Control mode (CTRL_POS, CTRL_LOAD, CTRL_EXTENSION)
		Speed	Maximum speed for positioning
		Destination	Destination position
		lpusTAN	Pointer to memory for Transaction Number
DoPEPos_A (Sync) @50 (@1050)	**98**	**Move cross-head in the specified control mode and speed to the given destination. Use specified acceleration and deceleration**	
		DoPEHdl	DoPE Handle
		MoveCtrl	Control mode (CTRL_POS, CTRL_LOAD, CTRL_EXTENSION)
		Acc	Acceleration
		Speed	Maximum speed for positioning
		Dec	Deceleration
		Destination	Destination position
		lpusTAN	Pointer to memory for Transaction Number

Command DLL-Ordination No.	Page	Parameter	Name
DoPEPosG1 (Sync)	**99**	**Move cross-head in the specified control mode and speed to the given destination. Do not change control mode. Destination must be different to move control mode!**	
@57 (@1057)		DoPEHdl	DoPE Handle
		MoveCtrl	Control mode (CTRL_POS, CTRL_LOAD, CTRL_EXTENSION)
		Speed	Maximum speed for positioning
		Limit	**Absolute** Limit position if Destination is not reached.
		DestCtrl	Control mode for Destination
		Destination	Destination position
		lpusTAN	Pointer to memory for Transaction Number
DoPEPosG1_A (Sync)	**100**	**Move cross-head in the specified control mode and speed to the given destination. Use specified acceleration and deceleration. Do not change control mode. Destination must be different to move control mode!**	
@58 (@1058)		DoPEHdl	DoPE Handle
		MoveCtrl	Control mode (CTRL_POS, CTRL_LOAD, CTRL_EXTENSION)
		Acc	Acceleration
		Speed	Maximum speed for positioning
		DecLimit	Deceleration for limit position
		Limit	**Absolute** Limit position if Destination is not reached.
		DecDest	Deceleration for destination
		DestCtrl	Control mode for Destination
		Destination	Destination position
		lpusTAN	Pointer to memory for Transaction Number
DoPEPosD1 (Sync)	**101**	**Move cross-head in the specified control mode and speed to the given destination. Do not change control mode. Destination must be different to move control mode!**	
@59 (@1059)		DoPEHdl	DoPE Handle
		MoveCtrl	Control mode (CTRL_POS, CTRL_LOAD, CTRL_EXTENSION)
		Speed	Maximum speed for positioning
		Limit	**Relative** Limit position if Destination is not reached.
		DestCtrl	Control mode for Destination
		Destination	Destination position
		lpusTAN	Pointer to memory for Transaction Number
DoPEPosD1_A (Sync)	**102**	**Move cross-head in the specified control mode and speed to the given destination. Use specified acceleration and deceleration. Do not change control mode. Destination must be different to move control mode!**	
@60 (@1060)		DoPEHdl	DoPE Handle
		MoveCtrl	Control mode (CTRL_POS, CTRL_LOAD, CTRL_EXTENSION)
		Acc	Acceleration
		Speed	Maximum speed for positioning
		DecLimit	Deceleration for limit position
		Limit	**Relative** Limit position if Destination is not reached.
		DecDest	Deceleration for destination
		DestCtrl	Control mode for Destination
		Destination	Destination position
		lpusTAN	Pointer to memory for Transaction Number

Command DLL-Ordination No.	Page	Parameter	Name
DoPEPosG2 (Sync)	**103**	**Move cross-head in the specified control mode and speed to the given destination. Change control mode. Destination must be different to move control mode!**	
@61 (@1061)		DoPEHdl	DoPE Handle
		MoveCtrl	Control mode (CTRL_POS, CTRL_LOAD, CTRL_EXTENSION)
		Speed	Maximum speed for positioning
		Limit	**Absolute** Limit position if Destination is not reached.
		DestCtrl	Control mode for Destination
		Destination	Destination position
		lpusTAN	Pointer to memory for Transaction Number
DoPEPosG2_A (Sync)	**104**	**Move cross-head in the specified control mode and speed to the given destination. Use specified acceleration and deceleration. Change control mode. Destination must be different to move control mode!**	
@62 (@1062)		DoPEHdl	DoPE Handle
		MoveCtrl	Control mode (CTRL_POS, CTRL_LOAD, CTRL_EXTENSION)
		Acc	Acceleration
		Speed	Maximum speed for positioning
		DecLimit	Deceleration for limit position
		Limit	**Absolute** Limit position if Destination is not reached.
		DecDest	Deceleration for destination
		DestCtrl	Control mode for Destination
		Destination	Destination position
		lpusTAN	Pointer to memory for Transaction Number
DoPEPosD2 (Sync)	**105**	**Move cross-head in the specified control mode and speed to the given destination. Change control mode. Destination must be different to move control mode!**	
@63 (@1063)		DoPEHdl	DoPE Handle
		MoveCtrl	Control mode (CTRL_POS, CTRL_LOAD, CTRL_EXTENSION)
		Speed	Maximum speed for positioning
		Limit	**Relative** Limit position if Destination is not reached.
		DestCtrl	Control mode for Destination
		Destination	Destination position
		lpusTAN	Pointer to memory for Transaction Number
DoPEPosD2_A (Sync)	**106**	**Move cross-head in the specified control mode and speed to the given destination. Use specified acceleration and deceleration. Change control mode. Destination must be different to move control mode!**	
@64 (@1064)		DoPEHdl	DoPE Handle
		MoveCtrl	Control mode (CTRL_POS, CTRL_LOAD, CTRL_EXTENSION)
		Acc	Acceleration
		Speed	Maximum speed for positioning
		DecLimit	Deceleration for limit position
		Limit	**Relative** Limit position if Destination is not reached.
		DecDest	Deceleration for destination
		DestCtrl	Control mode for Destination
		Destination	Destination position
		lpusTAN	Pointer to memory for Transaction Number

Command DLL-Ordination No.	Page	Parameter	Name
DoPEPosExt (Sync)	**108**	**General positioning command to a external destination. Use this command instead of DoPEPosG1, DoPEPosD1, DoPEPosG2 and DoPEPosD2. This command is not available for EDC5/20 and EDC100!!!**	
@149 (@1149)		DoPEHdl	DoPE Handle
		MoveCtrl	Control mode (CTRL_POS, CTRL_LOAD, CTRL_EXTENSION)
		Speed	Maximum speed for positioning
		LimitMode	Limit position is Absolute or relative
		Limit	Limit position if Destination is not reached.
		DestCtrl	Control mode for Destination
		Destination	Destination position
		DestinationMode	Control mode at destination
		lpusTAN	Pointer to memory for Transaction Number
DoPEPosExt_A (Sync)	**109**	**General positioning command to a external destination. Acceleration and deceleration are specified. Use this command instead of DoPEPosG1_A, DoPEPosD1_A, DoPEPosG2_A and DoPEPosD2_A. This command is not available for EDC5/20 and EDC100!!!**	
@150 (@1150)		DoPEHdl	DoPE Handle
		MoveCtrl	Control mode (CTRL_POS, CTRL_LOAD, CTRL_EXTENSION)
		Acc	Acceleration
		Speed	Maximum speed for positioning
		DecLimit	Deceleration for limit position
		LimitMode	Limit position is Absolute or relative
		Limit	Limit position if Destination is not reached.
		DecDest	Deceleration for destination
		DestCtrl	Control mode for Destination
		Destination	Destination position
		DestinationMode	Control mode at destination
		lpusTAN	Pointer to memory for Transaction Number

Command DLL-Ordination No.	Page	Parameter	Name
DoPEFMove (Sync)	**107**	**Move cross-head in the specified control mode and speed UP or DOWN**	
@45 (@1045)		DoPEHdl	DoPE Handle
		Direction	MOVE_HALT, MOVE_UP or MOVE_DOWN
		MoveCtrl	Control mode (CTRL_POS, CTRL_LOAD, CTRL_EXTENSION)
		Speed	Maximum speed for movement
		lpusTAN	Pointer to memory for Transaction Number
DoPEXpCont (Sync)	**110**	**Change control mode and continue movement in the new control mode with the actual speed**	
@69 (@1069)		DoPEHdl	DoPE Handle
		MoveCtrl	Control mode (CTRL_POS, CTRL_LOAD, CTRL_EXTENSION)
		Limit	Limit position
		lpusTAN	Pointer to memory for Transaction Number
DoPETrig (Sync)	**111**	**Move cross-head with the specified speed to the limit position. If the trigger position is reached a message will be transmitted.**	
@65 (@1065)		DoPEHdl	DoPE Handle
		Speed	Maximum speed for movement
		Limit	Limit position
		TriggerCtrl	Control mode for trigger channel
		Trigger	Trigger position
		lpusTAN	Pointer to memory for Transaction Number
DoPETrig_A (Sync)	**112**	**Move cross-head with the specified speed to the limit position. Use specified acceleration and deceleration. If the trigger position is reached a message will be transmitted.**	
@66 (@1066)		DoPEHdl	DoPE Handle
		Acc	Acceleration
		Speed	Maximum speed for movement
		Dec	Deceleration
		Limit	Limit position
		TriggerCtrl	Control mode for trigger channel
		Trigger	Trigger position
		lpusTAN	Pointer to memory for Transaction Number
DoPEStartCMD (Sync)	**113**	**Start definition of a combined movement command**	
@70 (@1070)		DoPEHdl	DoPE Handle
		Cycles	Repeat combined moving command this number of cycles
		ModeFlags	Flags definition
not required for Sync version		*lpusTAN*	*Pointer to memory for Transaction Number*
DoPEEndCMD (Sync)	**113**	**End of combined moving command and start it.**	
@70 (@1070)		DoPEHdl	DoPE Handle
		Operation	CMD_DISCARD or CMD_START
not required for Sync version		*lpusTAN*	*Pointer to memory for Transaction Number*

3.7 Complex moving commands

Command **DLL-Ordination No.**	**Page**	**Parameter**	**Name**
DoPEExt2Ctrl (Sync)	**114**	**Move cross-head according to an external command signal.**	
@43 (@1043)		DoPEHdl	DoPE Handle
		MoveCtrl	Control mode (CTRL_POS, CTRL_LOAD, CTRL_EXTENSION)
		OffsetCtrl	Offset for position, load or extension
		SensorNo	Sensor number for the external command signal
		OffsetSensor	Offset for external command signal
		Mode	Various position or speed control modes
		Scale	Scaling factor for external command signal
		lpusTAN	Pointer to memory for Transaction Number
DoPEFDPoti (Sync)	**115**	**Move cross-head according to an external command signal generated by a digital encoder (DigiPoti).**	
@44 (@1044)		DoPEHdl	DoPE Handle
		MoveCtrl	Control mode (CTRL_POS, CTRL_LOAD, CTRL_EXTENSION)
		MaxSpeed	Starting speed for speed controlled modes
		SensorNo	Sensor number for the external command signal
		DxTrigger	Dead area of encoder
		Mode	Various position or speed control modes
		Scale	Scaling factor for external command signal
		lpusTAN	Pointer to memory for Transaction Number
DoPECycle (Sync)	**116**	**Cycle movement command**	
@52 (@1052)		DoPEHdl	DoPE Handle
		MoveCtrl	Control mode (CTRL_POS, CTRL_LOAD, CTRL_EXTENSION)
		Speed1	Maximum speed to reach destination 1
		Dest1	Destination 1
		Halt1	Halt time at destination 1
		Speed2	Maximum speed to reach destination 2
		Dest2	Destination 2
		Halt2	Halt time at destination 2
		Cycles	Number of cycles
		Speed	Speed to final destination
		Destination	Final destination
		lpusTAN	Pointer to memory for Transaction Number
DoPECosine (Sync)	**117**	**Cosine movement command**	
@54 (@1054)		DoPEHdl	DoPE Handle
		MoveCtrl	Control mode (CTRL_POS, CTRL_LOAD, CTRL_EXTENSION)
		Speed1	Maximum speed to reach destination 1
		Dest1	Destination 1
		Dest2	Destination 2
		Frequency	Frequency for cosine
		HalfCycles	Number of half cycles
		Speed	Speed to final destination
		Destination	Final destination
		lpusTAN	Pointer to memory for Transaction Number

Command DLL-Ordination No.	Page	Parameter	Name
DoPECosineX (Sync)	**118**	**Cosine movement with halt time at Destination 1 and 2**	
@117 (@1117)		DoPEHdl	DoPE Handle
		MoveCtrl	Control mode (CTRL_POS, CTRL_LOAD, CTRL_EXTENSION)
		Speed1	Maximum speed to reach destination 1
		Dest1	Destination 1
		Halt1	Halt time at destination 1
		Dest2	Destination 2
		Halt2	Halt time at destination 2
		Frequency	Frequency for cosine
		HalfCycles	Number of half cycles
		Speed	Speed to final destination
		Destination	Final destination
		lpusTAN	Pointer to memory for Transaction Number
DoPECosineV (Sync)	**118**	**Activate peak value control for Cosine Command**	
@118 (@1118)		DoPEHdl	DoPE Handle
		Mode	Various operational mode (see definition)
		Dest1	Destination 1
		Dest2	Destination 2
		Cycles	Peak value control is active every xx Cycles
not required for Sync version		*lpusTAN*	*Pointer to memory for Transaction Number*
DoPETriangle (Sync)	**120**	**Triangular movement command**	
@55 (@1055)		DoPEHdl	DoPE Handle
		MoveCtrl	Control mode (CTRL_POS, CTRL_LOAD, CTRL_EXTENSION)
		Speed1	Maximum speed to reach destination 1
		Dest1	Destination 1
		Dest2	Destination 2
		Frequency	Frequency for triangle
		HalfCycles	Number of half cycles
		Speed	Speed to final destination
		Destination	Final destination
		lpusTAN	Pointer to memory for Transaction Number
DoPERectangle (Sync)	**121**	**Rectangular movement command**	
@56 (@1056)		DoPEHdl	DoPE Handle
		MoveCtrl	Control mode (CTRL_POS, CTRL_LOAD, CTRL_EXTENSION)
		Speed1	Maximum speed to reach destination 1
		Dest1	Destination 1
		Dest2	Destination 2
		Frequency	Frequency for rectangle
		HalfCycles	Number of half cycles
		Speed	Speed to final destination
		Destination	Final destination
		lpusTAN	Pointer to memory for Transaction Number
DoPEOffsC (Sync)	**122**	**Special moving command to measure the offset of an external, analogue speed controller**	
@72 (@1072)		DoPEHdl	DoPE Handle
		Speed	Maximum speed
		PosDiff	Distance to move cross-head
		lpusTAN	Pointer to memory for Transaction Number

3.8 Miscellaneous Control Commands

Command **DLL-Ordination No.**	**Page**	**Parameter**	**Name**
DoPEOn (Sync)	**125**	**Activate drive (only for EDC5/25 and new EDC60/120)**	
@81 (@1081)		DoPEHdl	DoPE Handle
not required for Sync version		*lpusTAN*	*Pointer to memory for Transaction Number*
DoPEDefaultAcc (Sync)	**125**	**Set default acceleration (and deceleration) for all moving commands.**	
@47 (@1047)		DoPEHdl	DoPE Handle
		MoveCtrl	Control mode (CTRL_POS, CTRL_LOAD, CTRL_EXTENSION)
		Acc	Default acceleration
not required for Sync version		*lpusTAN*	*Pointer to memory for Transaction Number*
DoPESpeedLimit (Sync)	**125**	**Set speed limit for all moving commands.**	
@130 (@1130)		DoPEHdl	DoPE Handle
		MoveCtrl	Control mode (CTRL_POS, CTRL_LOAD, CTRL_EXTENSION)
		Speed	Maximum speed
not required for Sync version		*lpusTAN*	*Pointer to memory for Transaction Number*
DoPEStop (Sync)	**126**	**Enable / disable drive**	
@48 (@1048)		DoPEHdl	DoPE Handle
		State	TRUE: Stop state -> drive disabled FALSE: drive enabled
		lpusTAN	Pointer to memory for Transaction Number
DoPEEmergencyMove (Sync)	**126**	**Activate / deactivate emergency movement.**	
@49 (@1049)		DoPEHdl	DoPE Handle
		State	TRUE: On FALSE: OFF
not required for Sync version		*lpusTAN*	*Pointer to memory for Transaction Number*
DoPEEmergencyOff (Sync)	**126**	**Activate / deactivate EmergencyOff state.**	
@53 (@1053)		DoPEHdl	DoPE Handle
		State	TRUE: On FALSE: OFF
not required for Sync version		*lpusTAN*	*Pointer to memory for Transaction Number*
DoPESetOpenLoopCommand (Sync)	**127**	**Set command output in open loop structure**	
@133 (@1133)		DoPEHdl	DoPE Handle
		Command	Command value in % of nominal output value
not required for Sync version		*lpusTAN*	*Pointer to memory for Transaction Number*
DoPESynchronizeSystemMode (Sync)	**123**	**Set mode for synchronized movements. (only EDC60/120)**	
@157 (@1157)		DoPEHdl	DoPE Handle
		Mode	SSM_SYNCMOVE SSM_ SYSTEMTIME SSM_ DISCARD
		Time	Delay or system time to set with the next DoPESynchronizeSystemMode commands
not required for Sync version		*lpusTAN*	*Pointer to memory for Transaction Number*
DoPESynchronizeSystemStart (Sync)	**124**	**Activate syncronisation. (only EDC60/120)**	
@158 (@1158)		DoPEHdl	DoPE Handle
not required for Sync version		*lpusTAN*	*Pointer to memory for Transaction Number*

3.9 Sensor EEPROM Handling

Command DLL-Ordination No.	Page	Parameter	Name
DoPERdSensorConKey @170	**128**	**Read sensor plug connected and key state.**	
		DoPEHdl	DoPE Handle
		Connector	Connector number of sensor
		*Connected	Pointer to the sensor plug connected state (0=not connected, 1=connected)
		*KeyPressed	Pointer to the sensor plug key state (0=not pressed, 1=pressed)
DoPERdSensorHeaderData @160	**129**	**Read sensor EEPROM data header.**	
		DoPEHdl	DoPE Handle
		Connector	Connector number of sensor
		*SenHdrData	Pointer to sensor data header structure
DoPERdSensorAnalogueData @162	**130**	**Read analogue sensor data.**	
		DoPEHdl	DoPE Handle
		Connector	Connector number of sensor
		*SenAnalogueData	Pointer to analogue sensor data structure
DoPEWrSensorAnalogueData @163	**130**	**Write analogue sensor data.**	
		DoPEHdl	DoPE Handle
		Connector	Connector number of sensor
		*SenAnalogueData	Pointer to analogue sensor data structure
DoPERdSensorIncData @164	**131**	**Read incremental sensor data.**	
		DoPEHdl	DoPE Handle
		Connector	Connector number of sensor
		*SenIncData	Pointer to incremental sensor data structure
DoPEWrSensorIncData @165	**131**	**Write incremental sensor data.**	
		DoPEHdl	DoPE Handle
		Connector	Connector number of sensor
		*SenIncData	Pointer to incremental sensor data structure
DoPERdSensorAbsData @166	**132**	**Read absolute sensor data.**	
		DoPEHdl	DoPE Handle
		Connector	Connector number of sensor
		*SenAbsData	Pointer to absolute value sensor data structure
DoPEWrSensorAbsData @167	**132**	**Write absolute sensor data.**	
		DoPEHdl	DoPE Handle
		Connector	Connector number of sensor
		*SenAbsData	Pointer to absolute value sensor data structure

3.10 DoPEOpenLink

unsigned	Port
unsigned	BaudRate
unsigned	RcvBuffers
unsigned	XmitBuffers
unsigned	DataBuffers
unsigned	APIVersion
void FAR	*Reserved
DoPE_HANDLE FAR	*DoPEHdl

This must be the first call to start communications with DoPE.
All needed memory is allocated, link parameters are set and the link is established.

In:	Port	Port number 0 = COM1, 1 = COM2 etc.
	BaudRate	Baud rate for serial lines, as supported by Windows
	RcvBuffers	Number of requested receiver buffers for messages. This number of received messages can be stored inside DoPE until they are read with DoPEGetMsg function.
	XmitBuffers	Number of requested transmitter buffers for messages. This number of transmit messages can be stored inside DoPE. They will be transmitted by DoPE to the EDC.
	DataBuffers	Number of requested data buffers. The measuring data record will be stored inside DoPE in a circular buffer. If data are not read with DoPEGetData the oldest record be overwritten!
	APIVersion	Version number of application program interface (API). Use DoPEAPIVERSION defined in dope.h.
	Reserved	Reserved parameter.
	DoPEHdl	Pointer to memory for DoPE link handle
Out:	*DoPEHdl	DoPE link handle. This handle has to be used for all further DoPE commands as a reference for this link.
Returns:		Error constant (DoPERR_xxxx)

If DoPEOpenLink returns DoPERR_TIMEOUT, connection to the EDC did not go online. EDC must be connected to the defined communication port. It must be switched on, PC-Control must be active and either automatic baud rate detection or the correct baud rate must be selected.

3.11 DoPECloseLink

DoPE_HANDLE	DoPEHdl

Close the link and free all allocated memory. After this call DoPEHdl is invalid and all further calls with this DoPEHdl will return with an error. After DoPECloseLink was called.

In:	DoPEHdl	DoPE link handle
Returns:		Error constant (DoPERR_xxxx)

3.12 DoPESetNotification

DoPE_HANDLE	DoPEHdl
unsigned	EventMask
NPROC	*NotifyProc
HWND	NotifyWnd
UINT	NotifyMsg

Select the events you are interested in.

There are two mechanism to notify your program if events occur:

1. Call-back mechanism
   The parameter NotifyProc contains the address of the user call-back-function. This function is called from DoPE after one or more active event(s) occurred. The call-back function is running at a higher priority than the application.
   Be aware the call-back can interrupt your application program at any time!
   The application programmer must use a Windows mechanism like MUTEX to synchronize communication between application program and call-back.
   Only time critical, short actions should be programmed in call-back function!
   The call-back function is called with the parameter:

HWND	NotifyWnd
UINT	NotifyMsg
WPARAM	DoPEHdl
LPARAM	Event

   Return Event mask. All processed events must be reset. The next call-back has all unprocessed events set. Thus unprocessed events are not lost, but stored inside DoPE.

2. PostMessage mechanism
   DoPE sends the active events using the windows function PostMessage to the application program (NotifyWnd). LPARAM contains the active events. WPARAM is not used.

Both, the application program or call-back must process the events by calling DoPE functions like DoPEGetMsg (see below)

In:	DoPEHdl	DoPE link handle
	EventMask	Events for Notification. (DoPEEVT_xx see below)
	NotifyProc	Notification call-back.
	NotifyWnd	Window handle of the application program. This handle is used in PostMessage function as an address (LPARAM) and it is to pass to call-back as a parameter.
	NotifyMsg	Message-Number used in PostMessage function as an identifier (WPARAM) and it is to pass to call-back as a parameter.
Returns:		Error constant (DoPERR_xxxx)

Possible events for EventMask:

Event	Description	Action
DoPEEVT_RXAVAIL	New message received.	Read message by DoPEGetMsg
DoPEEVT_DATAVAIL	New measured data record available.	Read data by DoPEGetData
DoPEEVT_DATAOVERFLOW	Overflow of data record. The oldest record was overwritten.	Read all old data by repetitively calling DoPEGetData until no more data are available.
DoPEEVT_ACK	For all commands sent to the EDC this event is generated after the EDC has checked the command and no error was found.	Just information, no DoPE function has to be called.
DoPEEVT_NAK	If the parameter check inside EDC found an error, this event will be generated.	Just information, no DoPE function has to be called.
DoPEEVT_OVERFLOW	Receiver queue overflow, the latest message was lost!	Read all old messages by repetitively calling DoPEGetMsg until no more data are available.
DoPEEVT_OFFLINE	State transition to offline. (EDC was switched off, cable disconnected ...)	Just information, no DoPE function has to be called.
DoPEEVT_ONLINE	State transition to online. Communication is OK.	Just information, no DoPE function has to be called.
DoPEEVT_ALL	All valid event bits.	

All needed events may be combined with an OR operation.

3.13 DoPEInitialize

DoPE_HANDLE	DP
WORD FAR	*lpusTANFirst
WORD FAR	*lpusTANLast

Initialise System with selected set-up data. This command must be called after a change of set-up data was made without selecting a new set-up.

In:	DoPEHdl	DoPE link handle
	lpusTANFirst	Pointer to first transaction number.
	lpusTANLast	Pointer to last transaction number.

Returns:	Error constant (DoPERR_xxxx)

4 Setup Commands

You can read or write machine set-up data. There are maximal four plus one machine set-up's possible. Set-up data from number one to four are stored inside EDC in a EEPROM. Set-up number zero is the working set-up. All data written to set-up zero are not stored, but can be used to initialise EDC. Thus handling more than four different set-up's by writing set-up data to set-up No. 0.
Before any set-up operation (read or write) can be done, the set-up must be opened. After the operations on the opened set-up are finished, you have to close the opened set-up.

4.1 DoPEOpenSetup(Sync)

DoPE_HANDLE	DoPEHdl	
unsigned short	SetupNo	
WORD FAR	**lpusTAN*	*(not for Sync. version)*

Open set-up 'SetupNo' for read/write operations.

In:	DoPEHdl	DoPE link handle
	SetupNo	Set-up Number
	lpusTAN	Pointer to transaction number.
Returns:		Error constant (DoPERR_xxxx)

4.2 DoPECloseSetup(Sync)

DoPE_HANDLE	DoPEHdl	
WORD FAR	**lpusTAN*	*(not for Sync. version)*

Closes previously opened set-up.

In:	DoPEHdl	DoPE link handle
	lpusTAN	Pointer to transaction number.
Returns:		Error constant (DoPERR_xxxx)

4.3 DoPESetupScale

DoPE_HANDLE	DoPEHdl
unsigned short	SetupNo
UserScale	US

Sets the set-up user scale.

In:	DoPEHdl	DoPE link handle
	SetupNo	Set-up Number
	US	User scale for all set-up sensor data (except DoPESenDef). The sensor data will be multiplied with the value in US. e.g. use this to convert the SI unit meter into mm by setting the UserScale to 1000 for the position sensor Default values 1.0 will be used if US is NULL.
Returns:		Error constant (DoPERR_xxxx)

4.4 DoPERdSetupAll(Sync)

DoPERdSetupAll(Sync)		
DoPE_HANDLE	DoPEHdl	
unsigned short	SetupNo	
DoPESetup FAR	*Setup	
WORD FAR	** lpusTANFirst*	*(not for Sync. version)*
WORD FAR	** lpusTANLast*	*(not for Sync. version)*

Read the total set-up structure.

In:	DoPEHdl	DoPE link handle
	SetupNo	Set-up Number
	DoPESetup FAR	Pointer to set-up data
	lpusTANFirst	Pointer to first transaction number.
	lpusTANLast	Pointer to last transaction number.
Returns:		Error constant (DoPERR_xxxx)

4.5 DoPEWrSetupAll(Sync)

DoPE_HANDLE	DoPEHdl	
unsigned short	SetupNo	
DoPESetup FAR	*Set-up	
WORD FAR	** lpusTANFirst*	*(not for Sync. version)*
WORD FAR	** lpusTANLast*	*(not for Sync. version)*

Write the set-up structure.

In:	DoPEHdl	DoPE link handle
	SetupNo	Set-up Number
	DoPESetup FAR	Pointer to set-up data
	lpusTANFirst	Pointer to first transaction number.
	lpusTANLast	Pointer to last transaction number.
Returns:		Error constant (DoPERR_xxxx)

DoPESetup structure:

```
typedef struct                       /* EDC Setup Data                    */
  {                                  /* --------------------------------- */
  DoPESenDef       SDef[MAX_MC];     /* Sensor definition data            */
  DoPECtrlSenDef   CSDef[MAX_CTRL];  /* Control-Sensor definition data    */
  DoPEOutChaDef    ODef[MAX_OC];     /* Analogue output definition data   */
  DoPEBitOutDef    BODef[MAX_BOUT];  /* Digital bit output definition data */
  DoPEBitInDef     BIDef[MAX_BIN];   /* Digital bit input definition data  */
  DoPEMachineDef   MDef;             /* Machine definition data            */
  DoPELinTblFalse  LinTblFalse;      /* Linearisation table FALSE values   */
  DoPELinTblTrue   LinTblTrue;       /* Linearisation table TRUE values    */
  } DoPESetup;
```

4.6 DoPERdSetupNumber

DoPE_HANDLE	DoPEHdl
unsigned short	*SetupNo

Read currently selected set-up number.

In:	DoPEHdl	DoPE link handle
	SetupNo	Pointer to store set-up number
Returns:		Error constant (DoPERR_xxxx)

4.7 DoPERdSensorDef

DoPE_HANDLE	DP
unsigned short	SensorNo
DoPESenDef FAR	*SensorDef

Read sensor definitions of opened set-up.

In:	DoPEHdl	DoPE link handle
	SensorNo	Sensor Number
	*SensorDef	Pointer for DoPESenDef structure
Returns:		Error constant (DoPERR_xxxx)

4.8 DoPEWrSensorDef(Sync)

DoPE_HANDLE	DP	
Unsigned short	SensorNo	
DoPESenDef FAR	*SensorDef	
WORD FAR	**lpusTAN*	*(not for Sync. version)*

Write Sensor definitions to opened set-up.

In:	DoPEHdl	DoPE link handle
	SensorNo	Sensor Number
	*SensorDef	Pointer for Sensor Definition structure
	lpusTAN	Pointer to Transaction-Number
Returns:		Error constant (DoPERR_xxxx)

DoPESenDef structure:

```
typedef struct                       /* Sensor definition data             */
  {                                  /* ---------------------------------- */
  WORD   Connector;                  /* Connector number of sensor     [No]*/
  WORD   Sign;                       /* Invert sign of sensor         [1/0]*/
  WORD   CtrlChannel;                /* Activate control channel      [1/0]*/
  WORD   LimitCtrl;                  /* Stop if limit exceeded        [1/0]*/
  WORD   ConnectedCtrl;              /* Stop if disconnected          [1/0]*/
  WORD   UseEeprom;                  /* Use sensor EEPROM data        [1/0]*/
                                     /* Only for analogue sensors:         */
  double Integr;                     /* Time of integration             [s]*/
                                     /* Only for sensors without EEPROM:   */
  WORD   Init;                       /* Sensor init                    [No]*/
  double NominalValue;               /* Nominal value of sensor      [Unit]*/
  WORD   Unit;                       /* Unit of sensor UNIT_xxx        [No]*/
  double Offset;                     /* Offset of sensor             [Unit]*/
  double UpperLimit;                 /* Upper range limit of sensor     [%]*/
  double LowerLimit;                 /* Lower range limit of sensor     [%]*/
```

```
                                            /* Only for incremental sensors:      */
  double Scale;                             /* Scale of sensor    [inc/revolution]*/
                                            /*               or [Unit/revolution]*/
  double Correction;                        /* Correction value of sensor     [No]*/
  } DoPESenDef;
```

4.9 DoPERdCtrlSensorDef

DoPE_HANDLE	DP
unsigned short	SensorNo
DoPECtrlSenDef FAR	*CtrlSensorDef

Read definitions of control sensor of opened set-up.

In:	DP	DoPE link handle
	SensorNo	Sensor Number
	*CtrlSensorDef	Pointer for DoPECtrlSenDef structure
Returns:		Error constant (DoPERR_xxxx)

4.10 DoPEWrCtrlSensorDef(Sync)

DoPE_HANDLE	DP	
Unsigned short	SensorNo	
DoPECtrlSenDef FAR	*CtrlSensorDef	
WORD FAR	**lpusTAN*	*(not for Sync. version)*

Write control sensor definitions to opened set-up.

In:	DoPEHdl	DoPE link handle
	SensorNo	Sensor Number
	*CtrlSensorDef	Pointer for Control-Sensor Definition structure
	lpusTAN	Pointer to Transaction-Number
Returns:		Error constant (DoPERR_xxxx)

DoPECtrlSenDef structure:

```
typedef struct                              /* Control Sensor definition data     */
  {                                         /* ---------------------------------- */
  double Acceleration;                      /* Nominal acceleration      [Unit/s²]*/
  double Speed;                             /* Nominal speed              [Unit/s]*/
  double WndSize;                           /* Target window size           [Unit]*/
  double WndTime;                           /* Time for target window          [s]*/
  double Deviation;                         /* Max. deviation of controller [Unit]*/
  WORD   DevReaction;                       /* Reaction if deviation exceeded [No]*/
  DWORD  PosK;                              /* Pos.  contr. P: gain           [No]*/
  WORD   PosTi;                             /* Pos.  contr. I: time constant  [No]*/
  WORD   PosTd;                             /* Pos.  contr. D: time constant  [No]*/
  DWORD  PosGenTd;                          /* Pos.  generator D              [No]*/
  DWORD  SpeedK;                            /* Speed contr. P: gain           [No]*/
  WORD   SpeedTi;                           /* Speed contr. I: time constant  [No]*/
  WORD   SpeedTd;                           /* Speed contr. D: time constant  [No]*/
  DWORD  SpeedGenTd;                        /* Speed generator D              [No]*/
  DWORD  AccK;                              /* Acceleration contrl. P: gain   [No]*/
                                            /* Only for analogue sensors:         */
  double Integr;                            /* Time of integration for control.[s]*/
  } DoPECtrlSenDef;
```

4.11 DoPERdOutChannelDef

DoPE_HANDLE	DP
unsigned short	OutChNo
DoPEOutChaDef FAR	*OutChDef

Read analogue output channel definitions of opened set-up.

In:	DoPEHdl	DoPE link handle
	OutChNo	Output channel no.
	*OutChDef	Pointer for DoPEOutChaDef structure
Returns:		Error constant (DoPERR_xxxx)

4.12 DoPEWrOutChannelDef(Sync)

DoPE_HANDLE	DP	
Unsigned short	OutChNo	
DoPEOutChaDef FAR	*CtrlSensorDef	
WORD FAR	**lpusTAN*	*(not for Sync. version)*

Write analogue output channel definitions to opened set-up.

In:	DoPEHdl	DoPE link handle
	OutChNo	Analogue Output channel no
	*DoPEOutChaDef	Pointer for DoPEOutChaDef structure
	lpusTAN	Pointer to Transaction-Number
Returns:		Error constant (DoPERR_xxxx)

DoPEOutChaDef structure:

```
typedef struct                       /* Definition of output channel       */
  {                                  /* ---------------------------------- */
  WORD   Connector;                  /* Connector number of channel    [No]*/
  WORD   Sign;                       /* Invert sign of channel        [1/0]*/
  double MaxValue;                   /* Maximum output value            [%]*/
  double MinValue;                   /* Minimum output value            [%]*/
  double InitValue;                  /* Initial output value            [%]*/
                                     /* Only if adjustable (DDAxx):        */
  double PaVoltage;                  /* Max. voltage of power amplifier [V]*/
  double PaCurrent;                  /* Max. current of power amplifier [A]*/
  double MaxCurrTime;                /* Max. time for max. current (I²t)[s]*/
  double DitherFrequency;            /* Dither frequency               [Hz]*/
  double DitherAmplitude;            /* Dither amplitude                [%]*/
  WORD   CurrentControllerGain;      /* Current controller gain set    [No]*/
  WORD   Signal;                     /* Digital command output signal  [No]*/
  WORD   SignalFrequency;            /* Digital command output signal  [No]*/
  WORD   ChangeDirection;            /* Dir.Outp.for synchronous motor[1/0]*/
  WORD   ChangeDirectionLevel;       /* Diriction Output level        [1/0]*/
  double Offset;                     /* Offset of channel               [%]*/
} DoPEOutChaDef;
```

4.13 DoPERdBitInDef

DoPE_HANDLE	DP
unsigned short	BitInNo
DoPEBitInDef FAR	*BitInDef

Read Bit input definitions of opened set-up.

In:	DP	DoPE link handle
	BitInNo	Output channel no.
	*BitInDef	Pointer for DoPEBitInDef structure
Returns:		Error constant (DoPERR_xxxx)

4.14 DoPEWrBitInDef(Sync)

DoPE_HANDLE	DP	
Unsigned short	BitInNo	
DoPEBitInDef FAR	* BitInDef	
WORD FAR	**lpusTAN*	*(not for Sync. version)*

Write Bit input definitions to opened set-up.

In:	DoPEHdl	DoPE link handle
	BitInNo	Bit Input channel no
	*DoPEBitInDef	Pointer for DoPEBitInDef structure
	lpusTAN	Pointer to Transaction-Number
Returns:		Error constant (DoPERR_xxxx)

DoPEBitInDef structure:

```
typedef struct                               /* Definition of bit output          */
  {                                          /* ---------------------------------- */
  WORD Connector;                            /* Connector number of device     [No]*/
  WORD InitValue;                            /* Initial value of device        [No]*/
  } DoPEBitOutDef;
```

4.15 DoPERdBitOutDef

DoPE_HANDLE	DP
unsigned short	BitOutNo
DoPEBitOutDef FAR	*BitOutDef

Read Bit output definitions of opened set-up.

In:	DoPEHdl	DoPE link handle
	BitOutNo	Output channel no.
	*BitOutDef	Pointer for DoPEBitOutDef structure
Returns:		Error constant (DoPERR_xxxx)

4.16 DoPEWrBitOutDef(Sync)

DoPE_HANDLE	DP	
Unsigned short	BitOutNo	
DoPEBitOutDef FAR	*BitOutDef	
WORD FAR	**lpusTAN*	*(not for Sync. version)*

Write Bit output definitions to opened set-up.

In:	DoPEHdl	DoPE link handle
	BitOutNo	Bit Output channel no
	*DoPEBitOutDef	Pointer for DoPEBitOutDef structure
	lpusTAN	Pointer to Transaction-Number

Returns: Error constant (DoPERR_xxxx)

DoPEBitOutDef structure:

```
typedef struct                          /* Definition of bit input            */
  {                                     /* ---------------------------------- */
  WORD Connector;                       /* Connector number of device     [No]*/
  WORD StopMask;                        /* Set bits stop the machine      [No]*/
  WORD StopLevel;                       /* Active level mask of StopMask  [No]*/
  } DoPEBitInDef;
```

4.17 DoPERdMachineDef

DoPE_HANDLE	DP
DoPEMachineDef FAR	*MachineDef

Read definitions of active machine of opened set-up.

In:	DoPEHdl	DoPE link handle
	*MachineDef	Pointer for DoPEMachineDef structure

Returns: Error constant (DoPERR_xxxx)

4.18 DoPEWrMachineDef(Sync)

DoPE_HANDLE	DP	
DoPEMachineDef FAR	*MachineDef	
WORD FAR	**lpusTAN*	*(not for Sync. version)*

Write definitions of active machine to opened set-up.

In:	DoPEHdl	DoPE link handle
	*MachineDef	Pointer for DoPEMachineDef structure
	lpusTAN	Pointer to Transaction-Number

Returns: Error constant (DoPERR_xxxx)

DoPEMachineDef structure:

```
typedef struct                          /* Definition of machine data         */
  {                                     /* ---------------------------------- */
  double SpeedCtrlTime;                 /* Speed controller cycle time     [s]*/
                                        /*                   [0.5ms .. 2.5ms]*/
  double PosCtrlTime;                   /* Position controller cycle time  [s]*/
  WORD   CtrlStructure;                 /* Closed loop control structure  [No]*/
  double DataAqTime;                    /* Data acquisition cycle time     [s]*/
  WORD   Mode;                          /* Data acquisition or control   [1/0]*/
  WORD   Bypass;                        /* Bypass for hydraulic          [1/0]*/
  WORD   XheadDir;                      /* Machine moves up/down         [1/0]*/
                                        /* with positive output signal        */
  double MotorEncRatio;                 /* Transmis. ratio motor-encoder [No] */
  double EncXheadRatio;                 /* Ratio encoder-Xhead     [Rev/Unit] */
  double BrakeOpen;                     /* Delay time to open brake after  [s]*/
                                        /* closed loop control is active      */
  double BrakeClose;                    /* Delay time to close brake before[s]*/
                                        /* closed loop control is deactivated */
  double PistonArea;                    /* Area of piston for hydraulic   [m²]*/
  double LoadMax;                       /* Max. load capacity of machine   [N]*/
  double Load100;                       /* Nominal load of machine         [N]*/
  double Stiffness;                     /* Over all stiffness of machine [N/m]*/
  short  UnUsed1;                       /* Unused                             */
  short  UnUsed2;                       /* Unused                             */
  WORD   ClampConnector;                /* Clamp: connector number        [No]*/
  WORD   ClampChannel;                  /* Clamp: channel                 [No]*/
  WORD   ClampActive;                   /* Clamp: active Low/High         [No]*/
  WORD   ShieldConnector;                /* Shield: connector number       [No]*/
  WORD   ShieldType;                     /* Shield: type simple/locked     [No]*/
  double ShieldTimeout;                  /* Shield: timeout                 [s]*/
  } DoPEMachineDef;
```

4.19 DoPERdLinTbl

DoPE_HANDLE	DP
DoPELinTblFalse FAR	*LinTblFalse
DoPELinTblTrue FAR	*LinTblTrue

Read linearisation table of opened set-up.

In:	DoPEHdl	DoPE link handle
	*LinTblFalse	Pointer to measured values structure
	*LinTblTrue	Pointer to reference values structure
Returns:		Error constant (DoPERR_xxxx)

4.20 DoPEWrLinTbl(Sync)

DoPE_HANDLE	DP	
DoPELinTblFalse FAR	*LinTblFalse	
DoPELinTblTrue FAR	*LinTblTrue	
WORD FAR	** lpusTANFirst*	*(not for Sync. version)*
WORD FAR	** lpusTANLast*	*(not for Sync. version)*

Write linearisation table to opened set-up.

In:	DoPEHdl	DoPE link handle
	*LinTblFalse	Pointer to measured values structure
	*LinTblTrue	Pointer to reference values structure
	*lpusTANFirst	Pointer to first Transaction-Number
	*lpusTANLast	Pointer to last Transaction-Number
Returns:		Error constant (DoPERR_xxxx)

DoPELinTblFalse and DoPELinTblTrue structure:

```
typedef struct                         /* Definition of linearisation table  */
  {                                    /* ---------------------------------- */
  WORD     LinNo;                      /* Number of points for mode lin. [No]*/
  double   FalseValue[LIN_DATA_MAX];   /* Measured value by the EDC       [N]*/
  } DoPELinTblFalse;

typedef struct                         /* Definition of linearisation table  */
  {                                    /* ---------------------------------- */
  WORD     LinNo;                      /* Number of points for mode lin. [No]*/
  double   TrueValue[LIN_DATA_MAX];    /* True value measured by the      [N]*/
                                       /* reference system                   */
  } DoPELinTblTrue;
```

4.21 DoPERdGeneralData

DoPE_HANDLE	DP
DoPEGeneralData FAR	*GeneralData

Read general data of opened set-up.

In:	DoPEHdl	DoPE link handle
	*GeneralData	Pointer for DoPEGeneralData structure
Returns:		Error constant (DoPERR_xxxx)

4.22 DoPEWrGeneralData(Sync)

DoPE_HANDLE	DP	
DoPEGeneralData FAR	*GeneralData	
WORD FAR	**lpusTAN*	*(not for Sync. version)*

Write general data to opened set-up.

In:	DoPEHdl	DoPE link handle
	*GeneralData	Pointer for DoPEGeneralData structure
	*lpusTAN	Pointer to Transaction-Number
Returns:		Error constant (DoPERR_xxxx)

DoPEGeneralData structure:

```
typedef struct                         /* General data                         */
  {                                    /* ------------------------------------ */
  WORD   MachineNo;                    /* Number of active machines        [No]*/
  WORD   MachineNoIo;                  /* Use IO's to select machine      [1/0]*/
                                       /* number (only EDC100, EDC60 and EDC120*/
  WORD   Supervisor;                   /* Supervisor mode                  [No]*/
  WORD   SuperPassword;                /* Supervisor Password(0=inactive)[No]*/
  WORD   UserPassword;                 /* User       Password(0=inactive)[No]*/
  WORD   Logo;                         /* DOLI Logo (0 = inactive)         [No]*/
  } DoPEGeneralData;
```

4.23 DoPESelSetup

DoPE_HANDLE	DP
unsigned short	SetupNo
UserScale FAR	US
WORD FAR	*lpusTANFirst
WORD FAR	*lpusTANLast

Select a machine set-up and initialise.

Set-up number 1 to 4 are stored inside EDC in a EEPROM. Set-up number 0 is the working set-up. If set-up 1 to 4 is selected, set-up data and basic tare values are copied from the EEPROM to the working set-up 0.

If set-up 0 is selected, set-up data in the working set-up 0 are not altered. Basic tare values cannot be stored permanently inside EDC. Thus if set-up 0 was written completely by PC, this data are used to initialise the system.

The DoPESelSetup function calls DoPEInitialize to do system initialisation.

In:	DoPEHdl	DoPE link handle
	SetupNo	Set-up No.
	US	User scale for all measuring channels. The measured data will be multiplied with the value in US. e.g. use this to convert the SI unit meter into mm by setting the UserScale to 1000 for the position channel.
	lpusTANFirst	Pointer to first transaction number.
	lpusTANLast	Pointer to last transaction number.
Returns:		Error constant (DoPERR_xxxx)

4.24 DoPERdSensorInfo

DoPE_HANDLE	DP
unsigned short	SensorNo
DoPESumSenInfo FAR	*Info

Read summary sensor information's.

In:	DoPEHdl	DoPE link handle
	SensorNo	Sensor Number
	*Info	Pointer for SumSenInfo structure
Returns:		Error constant (DoPERR_xxxx)

DoPESumSenInfo structure:

```
typedef struct                      /* Summary Sensor Information        */
  {                                 /* --------------------------------- */
  WORD   Connector;                 /* Connector number of sensor     [No]*/
  double NominalValue;              /* Nominal value of sensor      [Unit]*/
  WORD   Unit;                      /* Unit of sensor UNIT_xxx        [No]*/
  double Offset;                    /* Offset of sensor             [Unit]*/
  double UpperLimit;                /* Upper range limit of sensor  [Unit]*/
  double LowerLimit;                /* Lower range limit of sensor  [Unit]*/
  WORD   SensorState;               /* Sensor state SEN_STATE_xxx     [No]*/
  WORD   McType;                    /* Measuring channel type         [No]*/
  double UpperSoftLimit;            /* Upper soft limit             [Unit]*/
  double LowerSoftLimit;            /* Lower soft limit             [Unit]*/
  WORD   SoftLimitReaction;            /* reaction if soft limit         [No]*/
  double BasicTare;                    /* Basic   tare                 [Unit]*/
  } DoPESumSenInfo;
```

4.25 DoPERdSysUserData

DoPE_HANDLE	DP
BYTE FAR	*SysUsrData
unsigned	Length

Read system user data (16 Byte). The application program may use this 16 bytes EEPROM to store information permanently outside PC. It may be used for software protection or similar.

In:	DoPEHdl	DoPE link handle
	*SysUsrData	Pointer for SYSEEPROM user data
	Length*	User data buffer length in BYTE's
Returns:		Error constant (DoPERR_xxxx)

4.26 DoPEWrSysUserData(Sync)

DoPE_HANDLE	DP	
BYTE FAR	*SysUsrData	
unsigned	Length	
WORD FAR	**lpusTAN*	*(not for Sync. version)*

Write system user data (16 Byte).

In:	DoPEHdl	DoPE link handle
	*SysUsrData	Pointer for SYSEEPROM user data
	Length*	User data buffer length in BYTE's
	*lpusTAN	Pointer to Transaction-Number

Returns: Error constant (DoPERR_xxxx)

4.27 DoPERdSenUserData

DoPE_HANDLE	DP
BYTE FAR	*SenUsrData
unsigned	Length

Read sensor user data (128 Byte). The application program may use this 128 bytes EEPROM data to store information about the sensor. The EEPROM is located inside the sensor plug!

In:	DoPEHdl	DoPE link handle
	*SenUsrData	Pointer for Sensor-EEPROM user data
	Length*	User data buffer length in BYTE's
Returns:		Error constant (DoPERR_xxxx)

4.28 DoPEWrSenUserData(Sync)

DoPE_HANDLE	DP	
BYTE FAR	*SenUsrData	
unsigned	Length	
WORD FAR	**lpusTAN*	*(not for Sync. version)*

Write sensor user data (128 Byte).

In:	DoPEHdl	DoPE link handle
	*SenUsrData	Pointer for Sensor-EEPROM user data
	Length*	User data buffer length in BYTE's
	*lpusTAN	Pointer to Transaction-Number

Returns: Error constant (DoPERR_xxxx)

4.29 DoPERdSensorUserData

DoPE_HANDLE	DoPEHdl
WORD	Connector
BYTE	*SenUsrData
unsigned	Length

This function has the same effect as DoPERdSenUserData. But use here a connector number instead of a sensor number. It may also be used for not initialized sensors.

Read sensor user data (128 Byte). The application program may use this 128 bytes EEPROM data to store information about the sensor. The EEPROM is located inside the sensor plug!

In:	DoPEHdl	DoPE link handle
	Connector	Connector number of sensor
	*SenUsrData	Pointer for sensor user data
	Length	User data buffer length in BYTE’s
Returns:		Error constant (DoPERR_xxxx)

4.30 DoPEWrSensorUserData

DoPE_HANDLE	DoPEHdl
WORD	Connector
BYTE	*SenUsrData
unsigned	Length

This function has the same effect as DoPEWrSenUserData. But use here a connector number instead of a sensor number. It may also be used for not initialized sensors.

Write sensor user data (128 Byte).

In:	DoPEHdl	DoPE link handle
	Connector	Connector number of sensor
	*SenUsrData	Pointer for sensor user data
	Length	User data buffer length in BYTE’s
Returns:		Error constant (DoPERR_xxxx)

5 Message and Data handling

5.1 DoPESendMsg(Sync)

DoPE_HANDLE	DoPEHdl
void FAR	*Buffer
unsigned	Length
WORD FAR	*lpusTAN

Send a message to EDC. This function is only needed if you have to communicate directly to EDC's Subsystem. The message must be a command to the subsystem (see struct below).
Refer Documentation Subsystem for Test Machine Control.

Attention: This function is not available for Visual Basic program's!

In:	DoPEHdl	DoPE link handle
	Buffer	Pointer to message to transmit
	Length	Message length in bytes
	lpusTAN	Pointer to Transaction-Number
Returns:		Error constant (DoPERR_xxxx)

```
struct
  {
    unsigned short    Typ;     /* command number see pmx.h subsystem header */
    unsigned short    usTAN;   /* space for TAN                             */
    aPSxxxx           Usr;     /* command specific data see pmx.h           */
  } Buf;
```

5.2 DoPEGetMsg

DoPE_HANDLE	DoPEHdl
void FAR	*Buffer
unsigned	BufSize
unsigned FAR	*Length

Get a message from receive buffer. Incoming messages (from EDC) are stored inside DoPE. If enabled by DoPESetNotification, DoPE will signal the incoming message by "DoPEEVT_RXAVAIL" event. You can read a stored message with this command.
Messages may be error messages, end of positioning message, run time errors or sensor messages (see below).

In:	DoPEHdl	DoPE link handle
	Buffer	Pointer to message buffer
	BufSize	Buffer size in bytes
	Length	Pointer to store received message length
Out:	*Buffer	Message
	*Length	Message length in bytes
Returns:		Error constant (DoPERR_xxxx)

5.2.1 Command error messages

The parameter of all DoPE commands are checked and if not correct a error message is generated. Due to the remote state of EDC this error message is received from the PC some few milliseconds after the command has been transmitted. Synchronous DoPE command wait util the result of the parameter check is received and pas the result as the function return value.
Asynchronous DoPE commands do not wait. The error message must be read by DoPEGetMsg function.
Structure of command error messages:

```
typedef  struct  DoPECommandError   /* Command error                      */
  {                                 /* ---------------------------------- */
  unsigned short     CommandNumber; /* Number of command                  */
  unsigned short     ErrorNumber;   /* Number of error                    */
  } DoPECE;
```

List of command errors:

CMD_ERROR_NOERROR (0)
No error with this command, the command was executed.

CMD_ERROR_PARCORR (1001)
Parameter error:
One or more of the parameters exceeds the allowable limits. These parameters were corrected to the associated limiting value. The command was executed.

CMD_ERROR_PAR (1003)
Parameter error:
One or more of the parameters exceeds the allowable limits. Therefore the command was not clearly interpretable and not executed.

CMD_ERROR_XMOVE (1004)
Cross-head is not halted. Command not executed.

CMD_ERROR_INITSEQ (1005)
Sequence during the initialisation of the controller not observed. Command not executed.

CMD_ERROR_NOTINIT (1006)
Controller part not initialised. Command not executed.

CMD_ERROR_DIR (1007)
Required direction of movement is not permissible at the moment, because a corresponding error exists. Command not executed.

CMD_ERROR_TMP (1008)
Required resource is used at the moment. Command not executed.

CMD_ERROR_RUNTIME (1009)
Active runtime error. Command not executed.

CMD_ERROR_INTERN (1010)
Internal controller fault. Command not executed.

CMD_ERROR_MEM (1011)
Insufficient memory capacity for the execution of the order. Command not executed.

CMD_ERROR_CST (1012)
Wrong controller Structure for this command.

CMD_ERROR_NIM (1013)
Command not implemented.

CMD_ERROR_MSGNO Unknown message ID.	(2001)
CMD_ERROR_VERSION Wrong PE interface version.	(2003)
CMD_ERROR_OPEN Set-up not opened.	(2004)
CMD_ERROR_MEMORY Not enough memory.	(2005)

5.2.2 Run-time error messages

Errors or events that happen during the system is running, are reported to the application program as run-time errors.
Structure of run-time error message:

```
typedef  struct  DoPERunTimeError   /* Run time error (old style)              */
  {                                 /* ----------------------------------------*/
  unsigned short     ErrorNumber;   /* Number of run time error                */
  double             Time;          /* System time the error occurred          */
  unsigned short     Device;        /* Device Number responsible for the error */
  unsigned short     Bits;          /* Responsible bits if digital input device */
  } DoPERTE;
```

Currently a DoPE user cannot (or only with deep inside knowledge) interpret the parameters device and bits. But in most cases it is not necessary to use this two parameter. The error number is sufficient to display helpful error messages on screen.
In later DoPE versions there will be more detailed error messages.

List of run-time errors:

RTE_ERROR_S X-Head position controller deviation error. The difference between command and measured value is too big. Increase “deviation” parameter in “DoPECtrlSenDef” set-up structure or optimise control loop.	(101)
RTE_ERROR_F Load controller deviation error.	(102)
RTE_ERROR_E Extension controller deviation error.	(103)
RTE_EMOVE_END Emergency movement is finished but run-time error is still active.	(104)
RTE_DRIVE_OFF Drive was switched off.	(201)
RTE_E_MOVE_RQ Drive was switched off. A emergency drive is possible to clear this situation.	(202)
RTE_UPPER_LIMIT_SWITCH Drive was switched off due to an active upper limit switch.	(203)
RTE_LOWER_LIMIT_SWITCH Drive was switched off due to an active lower limit switch.	(204)

RTE_STOP	(205)
Drive not ready.	
RTE_DF_KEY	(206)
"Drive enable" output signal was withdrawn due to an inactive input signal "Drive enable".	
RTE_SHALT	(207)
Machine was halted in position control due to a active input signal. This normally happens after "STOP" key at EDC front panel was pressed.	
RTE_UPPER_LIMIT	(301)
Upper range limit exceeded.	
RTE_LOWER_LIMIT	(302)
Lower range limit exceeded.	
RTE_ERROR_UNHANDLED	(999)
Unknown runtime error.	

5.2.3 Movement control messages

Positioning, or in general movement commands, report their regular or irregular completion with movement control messages. There are four different structures to access different parameter. Please refer to the list of movement control messages below.

```
  typedef  struct  DoPEMoveCtrlMsg    /* Messages of movement control         */
    {                                 /* ------------------------------------ */
    unsigned short     MsgId;         /* ID of message                        */
    double             Time;          /* System time for the message          */
    unsigned short     Control;       /* Control mode of position             */
    double             Position;      /* Position                             */
    unsigned short     DControl;      /* Control mode of destination position */
    double             Destination;   /* Destination position                 */
    } DoPEMCM;

  typedef  struct  DoPESftMsg         /* 'Softend' Message                    */
    {                                 /* ------------------------------------ */
    unsigned short     MsgId;         /* ID of message                        */
    double             Time;          /* System time for the message          */
    unsigned short     Control;       /* Control mode of position             */
    double             Position;      /* Position                             */
    } DoPESftM;

  typedef  struct  DoPEOffsCMsg       /* 'Offset-Correction' Message          */
    {                                 /* ------------------------------------ */
    unsigned short     MsgId;         /* ID of message                        */
    double             Time;          /* System time for the message          */
    double             Offset;        /* Power Amplifier Offset               */
    } DoPEOffsCM;

  typedef  struct  DoPECheckMsg       /* 'Measuring Channel Supervision' Msg. */
    {                                 /* ------------------------------------ */
    unsigned short     MsgId;         /* ID of message                        */
    double             Time;          /* System time for the message          */
    unsigned short     CheckId;       /* ID of Measuring Channel Check        */
    double             Position;      /* Position                             */
    } DoPECheckM;

typedef  struct  DoPERefSignalMsg     /* 'Reference Signal' Message           */
  {                                   /* ------------------------------------ */
    unsigned short     MsgId;         /* ID of message                        */
    double             Time;          /* System time for the message          */
    unsigned short     SensorNo;      /* Control mode of position             */
    double             Position;      /* Position                             */
  } DoPERefSignalM;
```

List of movement control messages:

CMSG_POS (1) use DoPEMCM

A positioning command has been successfully competed and the machine reached the destination window within the specified time. The parameter “Control” and “Position” represent the position generator values after the command has been completed. The Parameter “Dcontrol” and “Destination” represent the destination position of the positioning command. For positioning commands without an explicit destination, these values are identical with “Control” and “Position”.

CMSG_UPPER_SFT (2) use DoPESftM

CMSG_ LOWER _SFT (3) use DoPESftM

During the positioning process, a configured softend for the supervision of movement sequences was overrun. The parameters “Control” and “Position” provide information concerning the control type of the overrun softend, as well as the actual position in this control channel at the time of recognition of the softend fault.

CMSG_POS_ERR (4) use DoPEMCM

A positioning command has been competed and the machine **did not reach** the destination window within the specified time. The parameters “Control” and “Position” represent the actually reached machine position after . the command has been completed. The Parameter “Dcontrol” and “Destination” represent the destination position of the movement command. For positioning commands without an explicit destination, these values are identical with “Control” and “Position”.

CMSG_TPOS (6) use DoPEMCM

A triggering position of the movement sequences has been reached. The parameters “Control” and “Position” supply the position in the triggering channel actual at time of triggering. At intermediate destinations of combined movement sequences with destination window supervision, this is the actual command value, otherwise it is the actual value of the triggering channel. The parameters “Dcontrol” and “Destination” supply the destination-respectively triggering position, given in the movement command.

CMSG_TPOS_ERR (7) use DoPEMCM

The destination window around the destination position of the movement command, was not reached within the specified time. The parameters “Control” and “Position” specify the position actually reached at the time of recognition of this positioning fault. The parameters “Dcontrol” and “Destination” represent the destination position of the movement command. For positioning commands without an explicit destination, these values are current values of the position generator.

CMSG_LPOS (8) use DoPEMCM

The destination window around the limit position of the movement command, was reached within the specified time. The parameters “Control” and “Position” supply the position, reached by the position generator. The parameters “Dcontrol” and “Destination” supply the specified destination position of the movement command. For positioning commands without an explicit destination, these values are identical with “Control” and “Position”.

CMSG_LPOS_ERR (9) use DoPEMCM

The destination window around the limit position of the movement command, was not reached within the specified time. The parameters “Control” and “Position” specify the position actually reached at the time of recognition of this positioning fault. The parameters “Dcontrol” and “Destination” represent the destination respectively triggering position of the movement command. For positioning commands without an explicit destination, these values are current values of the position generator.

CMSG_OFFSET (49) use DoPEOffsCM

The offset correction run has been terminated successfully. The parameter “Offset” contains the determined offset of the power amplifier.

CMSG_CHECK (51) use DoPECheckM

One of the active measurement channel supervisions has triggered. The action defined for this supervision was started. The parameter “CheckId” contains the identification of the supervision, and the parameter “Position” the position of the supervised measurement channel, which has led to the triggering.

CMSG_CHECK_ERR (52) use DoPECheckM

One of the active measurement channel supervisions has triggered. The action defined for this supervision was not started, because the direction of movement defined through the action was not possible due to an active softend. The parameter “CheckId” contains the identification of the supervision, and the parameter “Position” the position of the supervised measurement channel, which has led to the triggering.

CMSG_SHIELD (53) use DoPECheckM

The protective screen supervision has triggered. The action defined was started. The parameter “CheckId” contains the identification of the supervision, and the parameter “Position” the position of the supervised measurement channel, which has led to the triggering.

CMSG_SHIELD_ERR (54) use DoPECheckM

The protective screen supervision has triggered. The action defined was not started. The parameter “CheckId” contains the identification of the supervision, and the parameter “Position” the position of the supervised measurement channel, which has led to the triggering.

CMSG_ REFSIGNAL (55) use DoPERefSignalM

Reference Signal of a incremental sensor occurred.

5.2.4 Sensor message (only serial sensors)

A serial “sensor” may be connected via RS232 (X62A – X62D) to EDC60/120.
If it is specified with “NoProtocol”, it’s data are not interpreted by EDC or Dope, but send to the application program. Following message will be given to the application program:

```
typedef  struct  DoPESensorMsg        /* Sensor Message                      */
  {                                   /* ----------------------------------- */
    double         Time;              /* System time for the message         */
    unsigned short SensorNo;          /* Sensor Number 0 .. 15               */
    unsigned short Length;            /* Number of bytes in the message      */
    BYTE           Data[SENSOR_MSG_LEN]; /* Message                          */
  } DoPESensorM;
```

5.3 DoPEGetData

DoPE_HANDLE	DoPEHdl
void FAR	*Buffer

Get samples from receiver buffer. DoPE receives measuring data in a adjustable time scale. The data record is stored inside DoPE in a circular buffer. You can read one data record with this function. This data record will be deleted in the circular buffer. **For definition of DoPEData record refer to Page 128.**

In:	DoPEHdl	DoPE link handle
	Buffer	Pointer to buffer for data record. Buffer must be big enough to store the DoPEData structure
Out:	*Buffer	DoPEData record
Returns:		Error constant (DoPERR_xxxx)

5.4 DoPECurrentData

DoPE_HANDLE	DoPEHdl
void FAR	*Buffer

Get current samples from receiver buffer. This function reads the latest data record. This data record will not be deleted in the circular buffer. **For definition of DoPEData record refer to Page 128.**

In:	DoPEHdl	DoPE link handle
	Buffer	Pointer to buffer for data record. Buffer must be big enough to store the DoPEData structure
Out:	*Buffer	DoPEData record
Returns:		Error constant (DoPERR_xxxx)

5.5 DoPEClearReceiver

DoPE_HANDLE	DoPEHdl

Discard all received messages. If you want to get rid of old messages, that are of no interest any more, use this command.

In:	DoPEHdl	DoPE link handle
Returns:		Error constant (DoPERR_xxxx)

5.6 DoPEClearTransmitter

DoPE_HANDLE	DoPEHdl

Discard all unsent transmitter buffers.

In:	DoPEHdl	DoPE link handle
Returns:		Error constant (DoPERR_xxxx)

5.7 DoPEGetState

DoPE_HANDLE	DoPEHdl
DoPEState FAR	*Status

Get state information structure.

In:	DoPEHdl	DoPE link handle
	Status	Pointer to buffer for state information
Out:	*Status	State information (see below)
Returns:		Error constant (DoPERR_xxxx)

State information structure:

```
typedef struct
  {
  unsigned  ComState;            /* Communication state                */
  unsigned  RcvBuffer;           /* Number messages in receive queue   */
  unsigned  XmitBuffer;          /* Number of messages to be transmitted */
  } DoPEState;
```

Constants for ComState:

```
#define COM_STATE_OFF        0          /* Link is disabled                  */
```

```
#define COM_STATE_OFFLINE    1         /* Link is offline                   */
#define COM_STATE_INITCYCLE  2         /* Link is initialising              */
#define COM_STATE_ONLINE     3         /* Link is established and OnLine    */
```

5.8 DoPEGetErrors

DoPE_HANDLE	DoPEHdl
DoPEError FAR	*Error

Get current error counter values. DoPE counts all sort of communication errors. You may read this errors by using this command.

In:	DoPEHdl	DoPE link handle
	Error	Pointer to buffer for error counters
Out:	*Error	Current error counter values (see below)
Returns:		Error constant (DoPERR_xxxx)

The DoPEError-structure contains several communication error counters. Whenever an error is detected, the appropriate counter is increased by one. This error counters are useful to annualise communication problems.

```
typedef  struct                      /* Error counters                 */
    {
    unsigned   Parity;               /* Parity errors                  */
    unsigned   Overrun;              /* Overrun errors                 */
    unsigned   Frame;                /* Framing errors                 */
    unsigned   InvACK;               /* Invalid ACKs received          */
    unsigned   NoBuffer;             /* No receiver buffer available   */
    unsigned   DLESeq;               /* Invalid DLE sequence           */
    unsigned   BufOFlow;             /* Receiver buffer overflow       */
    unsigned   BccErr;               /* Checksum error                 */
    unsigned   MwError;              /* Invalid sample encoding        */
    } DoPEError;
```

6 Configuration

6.1 DoPERdVersion

DoPE_HANDLE	DP
DoPEVersion FAR	*Version

Read Version strings.

In:	DoPEHdl	DoPE link handle
	Version	Version strings
Returns:		Error constant (DoPERR_xxxx)

6.2 Data Acquisition commands

6.2.1 DoPERefr(Sync)

DoPE_HANDLE	DP	
double	Refresh	
WORD FAR	**lpusTAN*	*(not for Sync. version)*

Set time base for data acquisition. The default refresh time is defined in the set-up data and is normally 0.02 sec. You may change it according to your needs.

In:	DoPEHdl	DoPE link handle
	Refresh	Time in s
	lpusTAN	Pointer to Transaction-Number
Returns:		Error constant (DoPERR_xxxx)

6.2.2 DoPESetTime(Sync)

DoPE_HANDLE	DP	
double	Time	
WORD FAR	**lpusTAN*	*(not for Sync. version)*

Set time counter to a desired value.

In:	DoPEHdl	DoPE link handle
	Time	New value for Time in s
	lpusTAN	Pointer to Transaction-Number
Returns:		Error constant (DoPERR_xxxx)

6.2.3 DoPETransmitData(Sync)

DoPE_HANDLE	DP	
unsigned short	Enable	
WORD FAR	**lpusTAN*	*(not for Sync. version)*

Activate / Deactivate transmission of data. If deactivated no measuring data will be transmitted to the PC!

In:	DoPEHdl	DoPE link handle
	Enable	TRUE = Activate transmission
		FALSE = Deactivate transmission
	lpusTAN	Pointer to Transaction-Number
Returns:		Error constant (DoPERR_xxxx)

6.2.4 DoPEIntgr(Sync)

DoPE_HANDLE	DP	
unsigned short	SensorNo	
double	Intgr	
WORD FAR	**lpusTAN*	*(not for Sync. version)*

Set time of integration for an analogue measuring channel. Normally integration time is set to 20 ms. If this time is increased to a higher value e.g. 250 ms, resolution is increased and noise is decreased. For example you have a 100 kN load cell and you display 1 N as the smallest value. You reach this resolution at about 20 ms. Now you increase the time to 250 ms and you still display only 1 N steps, your signal noise will be reduced. You may use this effect for a stable load display while no test is running. Set integration time always to 250 ms while no test is running and just before starting a test reduce it to e.g. 20 ms.
Note: The integration time is independent from the refresh time (DoPERefr). It is possible to set the refresh time to 20 ms and the integration time to 250 ms.

In:	DoPEHdl	DoPE link handle
	SensorNo	Sensor Number 0 .. 9
	Intgr	Integration time for analogue measuring channels in sec. The limits of integration time depend on the selected time base for the speed control loop cycle time. (see machine set-up data) The minimum time is 1 x (time base for the speed control) The maximum time is 100 x (time base for the speed control)
	lpusTAN	Pointer to Transaction-Number
Returns:		Error constant (DoPERR_xxxx)

6.2.5 DoPECtrlTestValues

DoPE_HANDLE	DP
unsigned short	State

Enable or disable the controller Test values in the DoPEData record.
The three Test values are configured to:
Test1: Command
Test2: Feedback
Test3: Output

In:	DoPEHdl	DoPE link handle
	State	TRUE: enables FALSE: disables the controller test values in the DoPEData record.
Returns:		Error constant (DoPERR_xxxx)

6.3 Safety Shield

In order to protect the user from injures, the EDC contains a shield control. Two different shield types will be supported, simple and lockable shields. The allocations of the digital in- and outputs depend on the device selected in EDC-Set-up:

EDC100 Pin X2: (only simple shield)

Bit 10	Pin 10	Shield closed
Bit 11	Pin 3	Shield exists

4INC/4IO Pin B: (simple and locked shield)

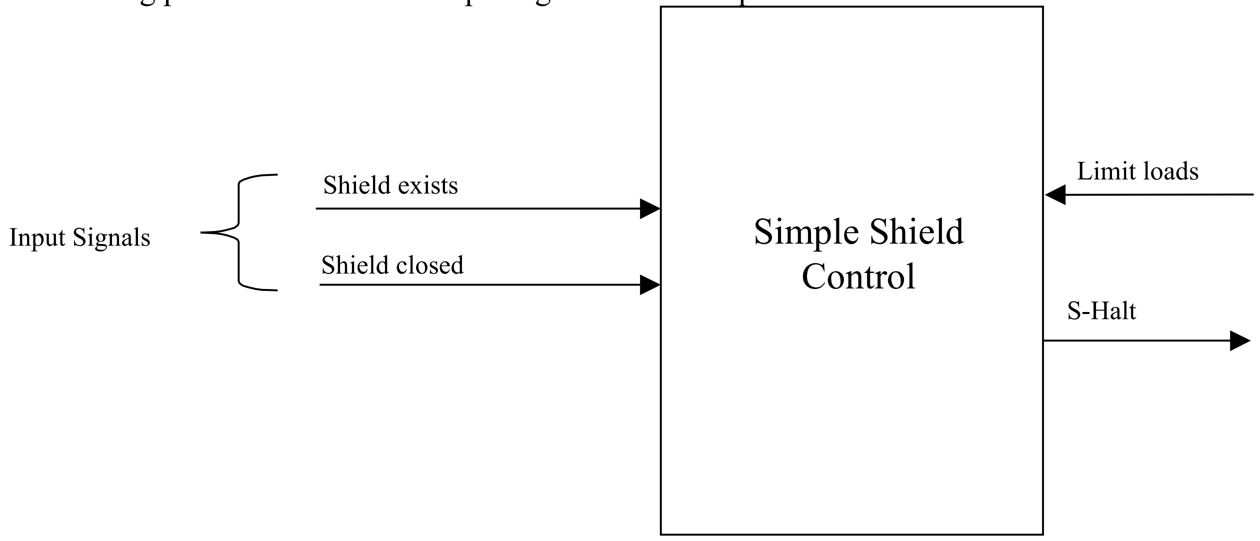
Bit 0	Pin 27	Shield exists
Bit 1	Pin 8	Shield closed
Bit 2	Pin 26	Shield is locked
Bit 3	Pin 7	Shield Test (key-switch)
Bit 6		Lock Shield
	Pin 29	Relay common contact
	Pin 10	Relay open contact
	Pin 28	Relay close contact

EDC60/120 shield option connector X10:

Bit 8	Pin 4	Shield exists
Bit 9	Pin 5	Shield closed
Bit 10	Pin 9	Shield is locked
Bit 11	Pin 10	Shield SETUP (key switch)
Bit 14		Lock shield
	Pin 3	Relays Common
	Pin 8	Relays Open
	Pin 2	Relays Close

Simple Shield operation:

The following picture illustrates the input signals of the simple shield control.



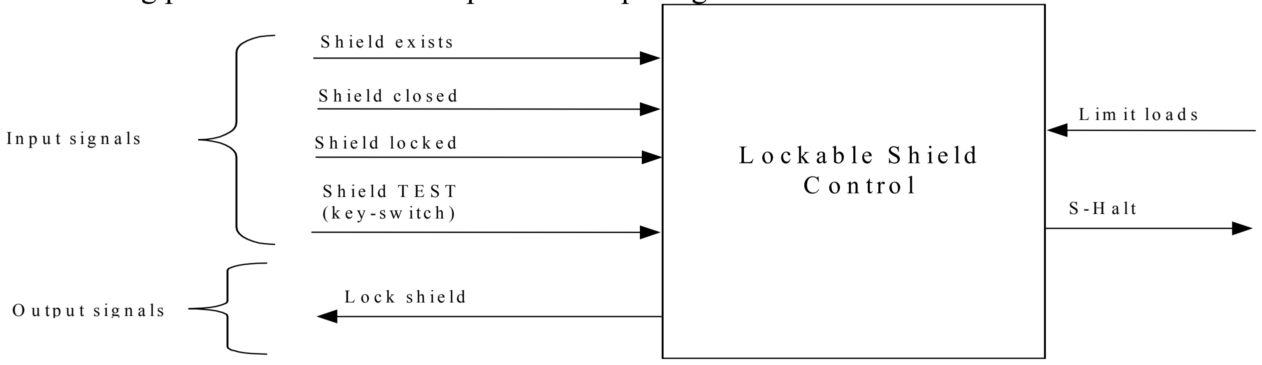
Meaning of the Signals:

1. Shield exists
   indicates the existence of the necessary hardware for the shield control.
2. Shield closed
   indicates that the shield is closed (has been closed by the user).

Simple Shield Control:

As long as the shield does not report 'closed', the loads limit LwrLock and UprLock will be checked permanently and the speed of X-head will be limited to SpeedLimit. If one load limit is exceeded, the X-head will be stopped position controlled. In this case, the user has to close the shield. When the shield reports 'closed', the X-head can be driven within the given nominal speed. The shield can be opened when the load remain under the limit loads. But if the shield gets opened despite the limit loads being exceeded, the X-head will stop position controlled.

Lockable Shield:
The following picture illustrates the input- and output signals of the lockable shield control.



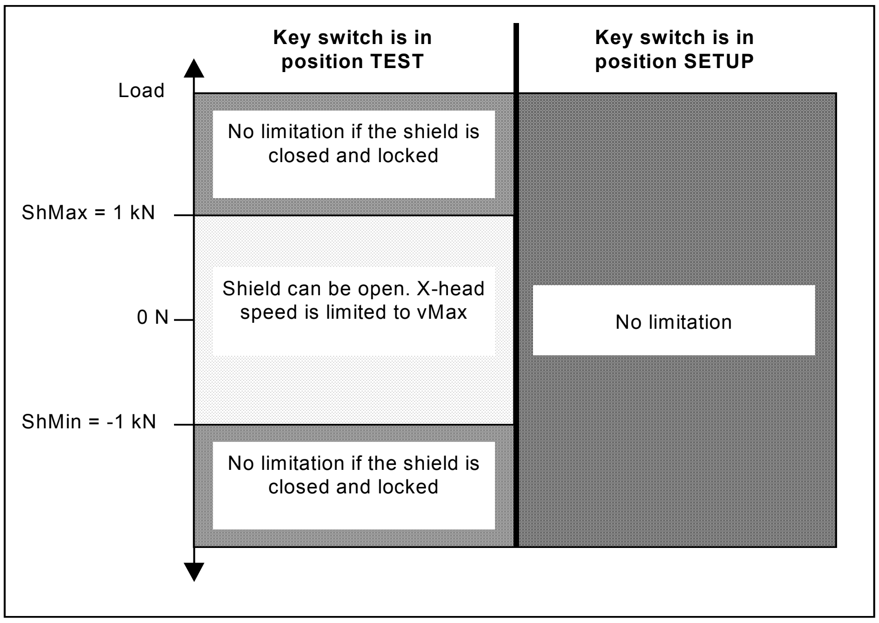
<u>Meaning of the Signals:</u>

1. Shield exists
   indicates the existence of the necessary hardware for the shield control.
2. Shield closed
   indicates that the shield is closed (has been closed by the user).
3. Shield locked
   The shield has been locked via the fitting output.
4. Shield TEST
   Key-switch at the shield enables the authorised user to drive the instrument even if the shield is not locked.
5. Lock shield
   Activates the locking mechanism at the shield.

<u>Lockable Shield Control:</u>

1. Key-switch is not in position TEST:
   As long as the shield does not report 'closed', the load limits LwrLock and UprLock will be checked permanently and the X-head speed will be limited to SpeedLimit. If one of the load limits is exceeded, the X-head will be stopped position controlled. In this case, the user has to close the shield. The shield will be locked automatically. When the shield reports 'closed', the X-head can be driven within the given nominal speed. The shield will be unlocked automatically and can be opened when the loads remain under the load limits.
2. Key-switch is in position TEST:
   All shield detection's are off. The X-head can be driven within the given nominal speed.

This is an example in order to illustrate the shield control:


Key switch is in position TEST
Key switch is in position SETUP
Load
No limitation if the shield is closed and locked
ShMax = 1 kN
Shield can be open. X-head speed is limited to vMax
0 N
No limitation
ShMin = -1 kN
No limitation if the shield is closed and locked

6.3.1 DoPEShieldLimit(Sync)

DoPE_HANDLE	DP	
unsigned short	SensorNo	
double	UprLock	
double	UprUnLock	
double	LwrUnLock	
double	LwrLock	
unsigned short	CtrlLimit	
double	SpeedLimit	
unsigned short	CtrlAction	
unsigned short	Action	
WORD FAR	**lpusTAN*	*(not for Sync. version)*

Activate the shield supervision.

Attention: The values for UprLock, UprUnLock, LwrUnLock and LwrLock are calculated with the current Tare and BasicTare.
If BasicTare is changed, while shield supervision is still active, these values are recalculated with the new BasicTare.
A new Tare does not effect these vales!

In:	DoPEHdl	DoPE link handle
	SensorNo	Sensor Number 0 .. 15
	UprLock	For all load values higher than this the shield must be closed and will be locked.
	UprUnLock	Upper limit for unlock the shield. (e.g. 10 % lower than UprLock)
	LwrUnLock	Lower limit for unlock the shield (e.g. 10 % higher than LwrLock).
	LwrLock	For all load values lower than this the shield must be closed and will be locked.
	CtrlLimit	Control mode for speed limit (normally X-head)
	SpeedLimit	Maximum allowed speed if shield is open.
	CtrlAction	Control mode for selected action
	Action	This action will be activated if the limits are reached FB_HALT: Halt machine in ‘CtrlAction’ control mode. FB_STOP: Switch off close loop control. (Use this action for machines without position sensor)
	lpusTAN	Pointer to Transaction-Number
Returns:		Error constant (DoPERR_xxxx)

6.3.2 DoPEShieldDisable(Sync)

DoPE_HANDLE	DP	
WORD FAR	**lpusTAN*	*(not for Sync. version)*

Deactivate the shield supervision.

In:	DoPEHdl	DoPE link handle
	lpusTAN	Pointer to Transaction-Number
Returns:		Error constant (DoPERR_xxxx)

6.3.3 DoPEShieldLock(Sync)

DoPE_HANDLE	DP	
WORD	State	
WORD FAR	**lpusTAN*	*(not for Sync. version)*

Lock or unlock the shield.

In:	DoPEHdl	DoPE link handle
	State	State of Shield lock. (TRUE = Lock, FALSE = UNLOCK)
	lpusTAN	Pointer to Transaction-Number
Returns:		Error constant (DoPERR_xxxx)

6.4 Channel supervision

6.4.1 DoPESetCheck(Sync)

6.4.2 DoPESetCheckX(Sync)

DoPE_HANDLE	DP	
unsigned short	CheckId	
unsigned short	SensorNo	
double	Limit	
double	*Tare*	*(only for DoPESetCheckX)*
unsigned short	Mode	
unsigned short	Action	
unsigned short	Ctrl	
double	Acc	
double	Speed	
double	Dec	
double	Destination	
WORD FAR	*lpusTAN	

Activate measuring channel supervision.
As support for special condition movement sequences, up to ten measurement channel supervisions are provided, all may be simultaneously active. Each supervision consists of the condition to supervise and the associated action. Supervisions are applicable to all configured measuring channels also for calculated channels if configured. All possible conditions and actions are listed below.
The condition of all active supervisions will be checked every 20 ms.
If a supervision condition hits, the associated action will be executed, an appropriate message transmitted and all supervisions which are not marked with the CHK_NOCLEAR bit are deactivated.

In:	DP	DoPE link handle
	CheckId	ID of this check, use the CheckId constants (CHK_ID0 ... CHK_ID9) If the check should no be cleared after an other check hits, or the CheckId with CHK_NOCLEAR.
	SensorNo	Sensor to be supervised
	Limit	Limit to be supervised
	Tare	Tare to be subtracted for CHK_PERCENT_MAX/MIN mode

Mode	Mode, how the limit is detected. (see below)
Action	This action will be activated if the check hits. (see below)
Ctrl	Control mode for selected action (CTRL_POS, CTRL_LOAD or CTRL_EXTENSION)
Acc	Acceleration
Speed	maximum speed
Dec	Deceleration
Destination	Final destination
lpusTAN	Pointer to Transaction-Number

Returns: Error constant (DoPERR_xxxx)

Possible modes:

CHK_BELOW	Action if $value < Limit$
CHK_ABOVE	Action if $value > Limit$

The following mode are only valid for DoPESetCheckX command:

CHK_PERCENT_MAX	Action if $\max.value - actualvalue > \frac{\max.value * Limit}{100}$
CHK_PERCENT_MIN	Action if $actualvalue - \min.value > \frac{\min.value * Limit}{100}$
CHK_ABS_MAX	Action if $\max.value - actualvalue > Limit$
CHK_ABS_MIN	Action if $actualvalue - \min.value > Limit$

Possible actions:

ACTION_HALT	HALT with default deceleration
ACTION_HALT_A	HALT with specified deceleration
ACTION_POS	Position with default deceleration
ACTION_POS_A	Position with specified deceleration
ACTION_XPCONT	Change control mode, go on with current speed
ACTION_STOP	STOP (switch off drive)
ACTION_NOACTION	No action, only message to host
ACTION_SETOL	Set command output (only in open loop structure)
ACTION_SHAL	Immediate Halt in X-head position control
ACTION_UP	Move Up with specified speed
ACTION_DOWN	Move Down with specified speed

6.4.3 DoPEClrCheck(Sync)

DoPE_HANDLE	DP	
unsigned short	CheckId	
WORD FAR	**lpusTAN*	*(not for Sync. version)*

Deactivate a measuring channel supervision

In:	DoPEHdl	DoPE link handle
	CheckId	ID of this check, (CHK_ID0 ... CHK_ID9 or CHK_ID_ALL)
	lpusTAN	Pointer to Transaction-Number

Returns: Error constant (DoPERR_xxxx)

6.4.4 DoPESetCheckLimit(Sync)

DoPE_HANDLE	DP	
unsigned short	SensorNo	
double	UprLimitSet	
double	UprLimitReset	
double	LwrLimitReset	
double	LwrLimitSet	
WORD FAR	**lpusTAN*	*(not for Sync. version)*

Activate a limit supervision for a measuring channel. If the measured value is outside UprLimitSet or LwrLimitSet a digital output will be set. This function may be used to prevent opening of clamps under load. The allocation of the digital output depend on the device selected in EDC-Set-up.

EDC100	**X2**	**Bit 14, Pin 6**
EDC60/120	**X2**	**Bit 3, Pin 8**
4INC/4IO	**Xxx**	**Bit 0, Pin 32**

Attention: The values for UprLimitSet, UprLimitReset, LwrLimitReset and LwrLimitSet are calculated with the current Tare and BasicTare.
If BasicTare is changed, while limit check is still active, these values are recalculated with the new BasicTare.
A new Tare does not effect these vales!

In:	DoPEHdl	DoPE link handle
	SensorNo	Defines the sensor to be supervised.
	UprLimitSet	For all values above, the digital output will be set.
	UprLimitReset	For all values below, the digital output will be reset
	LwrLimitReset	For all values above, the digital output will be reset
	LwrLimitSet	For all values below, the digital output will be set.
	lpusTAN	Pointer to Transaction-Number
Returns:		Error constant (DoPERR_xxxx)

6.4.5 DoPEClrCheckLimit(Sync)

DoPE_HANDLE	DP	
WORD FAR	**lpusTAN*	*(not for Sync. version)*

Deactivate check limit function. The digital output will be set inactive.

In:	DoPEHdl	DoPE link handle
	lpusTAN	Pointer to Transaction-Number
Returns:		Error constant (DoPERR_xxxx)

6.5 Controller Parameter

6.5.1 DoPEPosPID (Sync)

DoPE_HANDLE	DoPEHdl	
unsigned short	MoveCtrl	
unsigned long	P	
unsigned short	I	
unsigned short	D	
WORD FAR	**lpusTAN*	*(not for Sync. version)*

This function replaces the older function “DoPECtrlP”. The current DoPE version supports both functions, but in future versions the old function may be removed.

Set parameter for closed loop position controller.

In:	DoPEHdl	DoPE link handle
	MoveCtrl	Control mode (CTRL_POS, CTRL_LOAD or CTRL_EXTENSION)
	P	Proportional gain of the position controller
	I	Integration time constant **Attention: Normally there is a speed control loop with an integration part. In such cases set I here to ZERO. Otherwise we have a control loop with two integration parts and this will never be stable!!**
	D	Derivative time constant **Attention: For future use only**
	lpusTAN	Pointer to Transaction-Number
Returns:		Error constant (DoPERR_xxxx)

6.5.2 DoPESpeedPID(Sync)

DoPE_HANDLE	DoPEHdl	
unsigned short	MoveCtrl	
unsigned long	P	
unsigned short	I	
unsigned short	D	
WORD FAR	**lpusTAN*	*(not for Sync. version)*

This function replaces the older function “ DoPECtrlP_Xp ”. The current DoPE version supports both functions, but in future versions the old function may be removed.

Set parameter for closed loop speed controller.

In:	DoPEHdl	DoPE link handle
	MoveCtrl	Control mode (CTRL_POS, CTRL_LOAD or CTRL_EXTENSION)
	P	Proportional gain of the speed controller

I	Integration time constant
D	Derivative time constant
lpusTAN	Pointer to Transaction-Number
Returns:	Error constant (DoPERR_xxxx)

6.5.3 DoPEPosFeedForward(Sync)

DoPE_HANDLE	DoPEHdl	
unsigned short	MoveCtrl	
unsigned long	P	
WORD FAR	**lpusTAN*	*(not for Sync. version)*

This function replaces the older function “ DoPECtrlPGKTd ”. The current DoPE version supports both functions, but in future versions the old function may be removed.

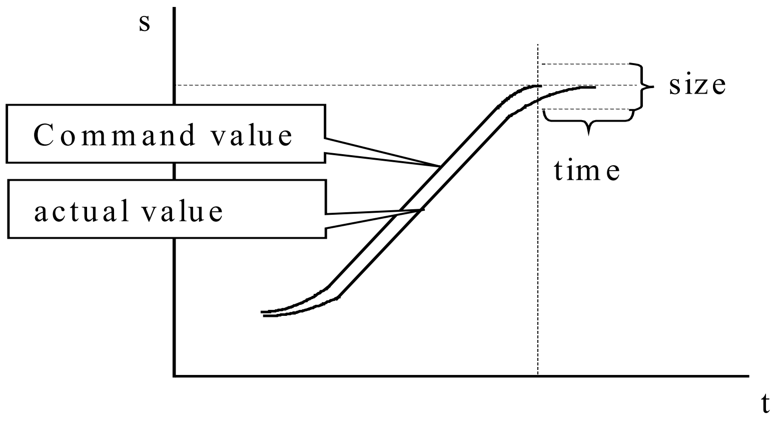
Set feed forward gain of closed loop controller.

In:	DoPEHdl	DoPE link handle
	MoveCtrl	Control mode (CTRL_POS, CTRL_LOAD or CTRL_EXTENSION)
	P	Gain for feed forward control.
	lpusTAN	Pointer to Transaction-Number
Returns:		Error constant (DoPERR_xxxx)

6.5.4 DoPEDestWnd(Sync)

DoPE_HANDLE	DP	
unsigned short	MoveCtrl	
double	WndSize	
double	WndTime	
WORD FAR	**lpusTAN*	*(not for Sync. version)*

Definitions for destination window.
The closed loop controller supervises the controlled channel. After the command value reaches the final destination, the controlled channel has to reach this position within in the specified destination window. If the actual value reaches the destination window within the specified time the positioning sequence if successfully finished. The message "CMSG_POS" is send to the application program. If it does not reach it, the message "CMSG_POS_ERR" is send.



In:	DoPEHdl	DoPE link handle
	MoveCtrl	Control mode (CTRL_POS, CTRL_LOAD or CTRL_EXTENSION)
	WndSize	Size of destination window
	WndTime	Time for destination window
	lpusTAN	Pointer to Transaction-Number
Returns:		Error constant (DoPERR_xxxx)

6.5.5 DoPESft(Sync)

DoPE_HANDLE	DP	
unsigned short	MoveCtrl	
double	UpperSft	
double	LowerSft	
unsigned short	Reaction	
WORD FAR	**lpusTAN*	*(not for Sync. version)*

Definitions of limits supervised by software (softend). All moving commands will limit destinations to this softend's.

The software limit switches, also called softend's, fulfil several functions. They are used as limits, the movement sequence control should not exceed under normal circumstances. However, in contrast to the range limits of the measurement channels, they are working limits that can be changed during the operation of the machine in some defined conditions. Additionally, the softend's can be used for the supervision of movement sequences.

The softend's provide the limits for the destination positions for movement instructions and are implicitly used as destination positions for movement instructions. The state of the softend's is maintained in CtrlState2 (see default measuring dada record).

Attention: The values for UpperSft and LowerSft are calculated with the current Tare and BasicTare.
If BasicTare is changed, these values are recalculated with the new BasicTare.
A new Tare does not effect these vales!

In:	DoPEHdl	DoPE link handle
	MoveCtrl	Control mode (CTRL_POS, CTRL_LOAD or CTRL_EXTENSION)
	UpperSft	Upper soft limit
	LowerSft	Lower soft limit
	Reaction	Action to be activated after softend is reached Note: for the active control mode, the reaction is always REACT_ACTION
	lpusTAN	Pointer to Transaction-Number
Returns:		Error constant (DoPERR_xxxx)

REACT_STATUS	only status bits are set
REACT_ACTION	Halt in X-head position control

6.5.6 DoPECtrlError(Sync)

DoPE_HANDLE	DP	
unsigned short	MoveCtrl	
double	Error	
unsigned short	Reaction	
WORD FAR	**lpusTAN*	*(not for Sync. version)*

Define maximum error signal for closed loop controller. The closed loop controller supervises the error signal (command - measured value). If the error exceeds the specified range, the desired action will be activated.

In:	DoPEHdl	DoPE link handle
	MoveCtrl	Control mode (CTRL_POS, CTRL_LOAD or CTRL_EXTENSION)
	Error	Maximum error signal
	Reaction	Action to be activated after error is reached REACT_ACTION is for: CTRL_POS Emergency off CTRL_LOAD Halt in Xhead control CTRL_EXTENSION Halt in Xhead control
	lpusTAN	Pointer to Transaction-Number
Returns:		Error constant (DoPERR_xxxx)

6.5.7 DoPESetDither (Sync)

DoPE_HANDLE	DP	
unsigned short	Output	
double	Frequency	
double	Amplitude	
WORD FAR	**lpusTAN*	*(not for Sync. version)*

Set Parameter for dither. Some DOLI output amplifier like D03I, D16I, D32I have a digital dither generator function. The dither amplitute and frequency can be set by this command.

In:	DoPEHdl	DoPE link handle
	Output	Number of analogue output channel
	Frequency	Dither frequency in Hz
	Amplitude	Dither aplitude in % of nominal output current.
	lpusTAN	Pointer to Transaction-Number
Returns:		Error constant (DoPERR_xxxx)

6.5.8 DoPECtrlSpeedTimeBase(Sync)

DoPE_HANDLE	DP	
double	Time	
WORD FAR	**lpusTAN*	*(not for Sync. version)*

Define maximum time base for speed calculation.

When changing control mode, the current speed of the new control channel is used as the start speed after control mode was changed. Normally this speed is determined by differentiation by two following position values. In case of a small resolution of the transducer for the new control mode, the speed values are very small and thus also inaccurate. Using this command the time base can be increased and thus a higher accuracy for speed calculation can be archived. Use this command only if speed is not changing too fast otherwise with a big time base the calculated speed is incorrect. . <u>This time base exclusively works for the determination of the start speed when control mode is changed. It has no effect for speed calculation for the digital speed controller.</u>

In:	DoPEHdl	DoPE link handle
	Time	Time base in seconds. internally only 1, 2, 4, 8, 16, 32, 64, 128 x position controller time base will be used!
	lpusTAN	Pointer to Transaction-Number
Returns:		Error constant (DoPERR_xxxx)

6.6 Calibration

6.6.1 DoPEZeroCal(Sync)

DoPE_HANDLE	DP	
unsigned short	SensorBits	
WORD FAR	**lpusTAN*	*(not for Sync. version)*

Compensate only zero offset drift. It takes about 0.2 s to compensate zero offset drift. During compensation the sensor is not measured!!!

In:	DoPEHdl	DoPE link handle
	SensorBits	Bit 0 .. 15 define the sensor Number to be calibrated. Bit 0 = 1 Calibrate zero offset of Sensor 0 Bit 1 = 1 Calibrate zero offset of Sensor 1 ...
	lpusTAN	Pointer to Transaction-Number
Returns:		Error constant (DoPERR_xxxx)

6.6.2 DoPECal(Sync)

DoPE_HANDLE	DP	
unsigned short	SensorBits	
WORD FAR	**lpusTAN*	*(not for Sync. version)*

Compensate drifts (zero and amplification) of the measuring channel. It takes about 0.5 s to compensate drifts. During compensation the sensor is not measured!!!

In:	DoPEHdl	DoPE link handle
	SensorBits	Bit 0 .. 15 define the sensor Number to be calibrated. Bit 0 = 1 Calibrate Sensor 0 Bit 1 = 1 Calibrate Sensor 1 ...
	lpusTAN	Pointer to Transaction-Number
Returns:		Error constant (DoPERR_xxxx)

6.7 Tare functions

DoPE offers two different tare functions, Tare and BasicTare.
The **DoPESetBasicTare** function should be used for set-up the machine configuration like changing clamps. The Basic-Tare value is stored in the active machine set-up inside EDC. After switching mains power off / on, the Basic-Tare value is still valid.
The simple **DoPESetTare** function can be used at any time. This tare value will be lost after reset.

6.7.1 DoPESetBasicTare(Sync)

DoPE_HANDLE	DP	
unsigned short	SensorNo	
unsigned short	Mode	
double	BasicTare	
WORD FAR	** lpusTANFirst*	*(not for Sync. version)*
WORD FAR	** lpusTANLast*	*(not for Sync. version)*

Set basic tare of the measuring channel. BasicTare value will be stored in EDC's non volatile memory. This function clears the ordinary tare value

In:	DP	DoPE link handle
	SensorNo	Sensor Number
	Mode = 0	BasicTare represents the desired measuring value. This is useful to set cross-head position for systems with encoder.
	Mode = 1	BasicTare will be subtracted. This is useful to compensate the weight of clamps.
	BasicTare	Value for BasicTare
	lpusTANFirst	Pointer to first transaction number.
	lpusTANLast	Pointer to last transaction number.
Returns:		Error constant (DoPERR_xxxx)

6.7.2 DoPESetTare

DoPE_HANDLE	DP
unsigned short	SensorNo
double	Tare

Set tare of the measuring channel. This tare function may be used as a working tare.

In:	DP	DoPE link handle
	SensorNo	Sensor Number Mode
	Tare	Value to be subtracted.
Returns:		Error constant (DoPERR_xxxx)

6.7.3 DoPEGetBasicTare

DoPE_HANDLE	DP
unsigned short	SensorNo
double FAR	*BasicTare

Read tare value of the measuring channel.

In:	DP	DoPE link handle
	SensorNo	Sensor Number
	*BasicTare	Pointer for BasicTare
Out:	BasicTare	Active value for BasicTare
Returns:		Error constant (DoPERR_xxxx)

6.7.4 DoPEGetTare

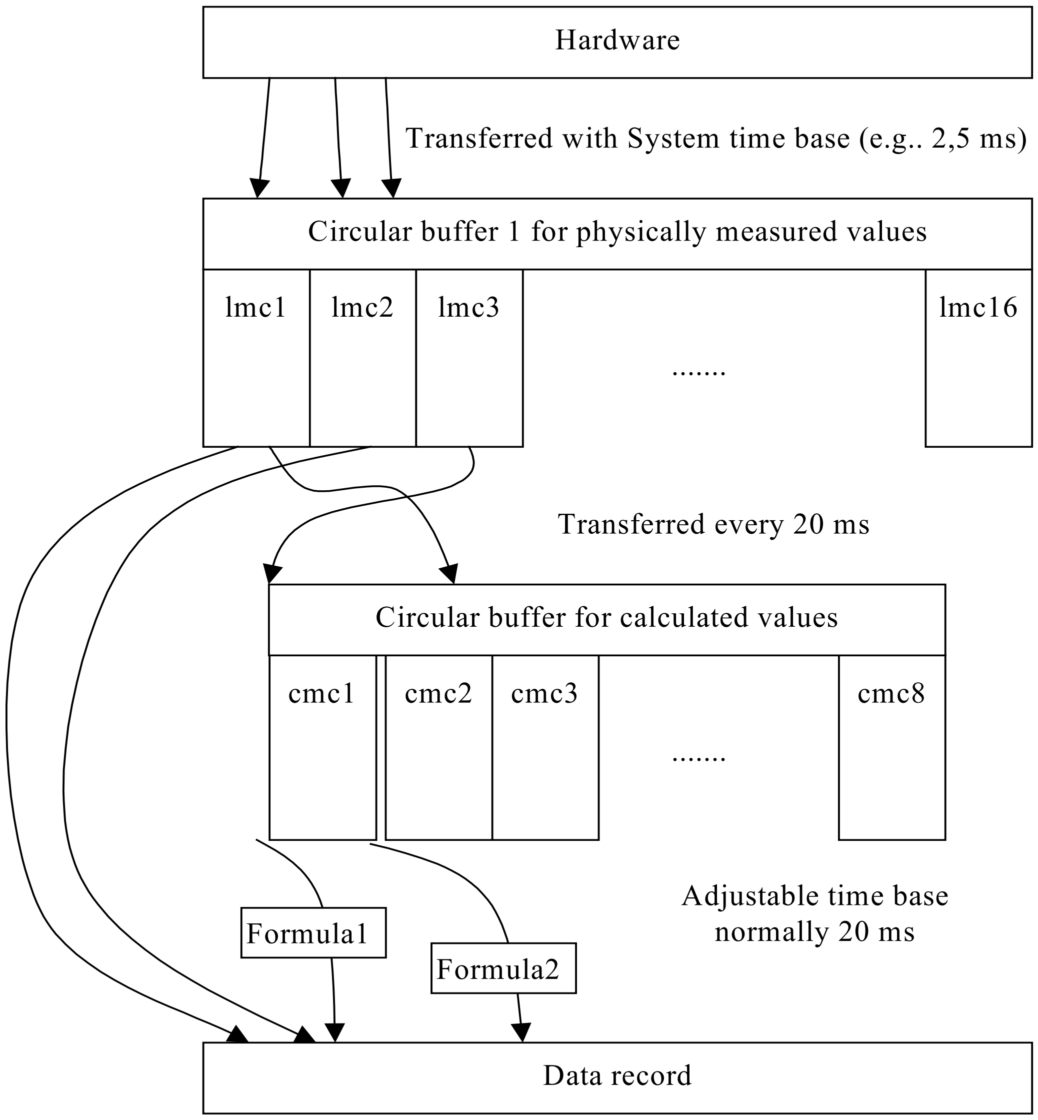
DoPE_HANDLE	DP
unsigned short	SensorNo
double FAR	*Tare

Read tare value of the measuring channel

In:	DP	DoPE link handle
	SensorNo	Sensor Number
	*Tare	Pointer for tare
Out:	Tare	Active value for tare.
Returns:		Error constant (DoPERR_xxxx)

6.8 Calculated measuring channels

Additional to the measured values determined by sensors, the system provides some formulas for calculated measuring channels. The calculated values are basically calculated from physical measured values. All physical measured values are transferred into a circular buffer at a fixed time base of 20 ms. This circular buffer has 128 entries. For analogue channels, the integration time may be defined.


Hardware
Transferred with System time base (e.g.. 2,5 ms)
Circular buffer 1 for physically measured values
lmc1
lmc2
lmc3
....
lmc16
Transferred every 20 ms
Circular buffer for calculated values
cmc1
cmc2
cmc3
....
cmc8
Formula1
Formula2
Adjustable time base normally 20 ms
Data record

The input values for the calculation are read from this circular buffer, calculated by the desired formula and transferred to the measuring data record. This calculation is done every data refresh time, normally 20 ms. Maximum eight calculated measuring channels are possible. They must be assigned to one of the sixteen logical measuring channels. The allocation can take place also after the initialisation at run-time. A calculated measuring channel can be deleted at any time again.
Calculated measuring channels may be especially used for supervisions.
At present three formulas are implemented:

1. Calculated Speed
2. Calculated speed of command signal
3. Gradient between two measure values

6.8.1 DoPEConfCMcSpeed(Sync)

DoPE_HANDLE	DP	
unsigned short	CalculatedSensorNo	
unsigned short	SensorNo	
double	IntegrationTime	
double	Timebase	
WORD FAR	** lpusTANFirst*	*(not for Sync. version)*
WORD FAR	** lpusTANLast*	*(not for Sync. version)*

Configure calculated speed to measuring data record.
From any measuring channel speed (the first differentiation) may be calculated and configured to the data record. After it is configured, it may be used as the supervised channel in the command "DoPESetCheck".
This calculated channel is not allowed as the command channel in DoPEExts2 or DoPEFDPoti command!

In:	DP	DoPE link handle
	CalculatedSensorNo	Sensor number for the calculated speed value (SENSOR_4 to SENSOR_15)
	SensorNo	Sensor Number of the physical channel for speed calculation
	IntegrationTime	Integration time (only for analogue channels)
	Timebase	Time base for speed calculation (maximum 2.56 sec)
	lpusTANFirst	Pointer to first transaction number.
	lpusTANLast	Pointer to last transaction number.
Returns:		Error constant (DoPERR_xxxx)

6.8.2 DoPEConfCMcCommandSpeed(Sync)

DoPE_HANDLE	DP	
unsigned short	CalculatedSensorNo	
double	Timebase	
WORD FAR	** lpusTANFirst*	*(not for Sync. version)*
WORD FAR	** lpusTANLast*	*(not for Sync. version)*

Configure calculated speed of command output to data record.
Speed of command output (the first differentiation) may be calculated and configured to the data record. After it is configured, it may be used as the supervised channel in the command "DoPESetCheck".
This calculated channel is not allowed as the command channel in DoPEExts2 or DoPEFDPoti command!

In:	DP	DoPE link handle
	CalculatedSensorNo	Sensor number for the calculated command speed (SENSOR_4 to SENSOR_15)
	Timebase	Time base for command speed calculation (maximum 2.56 sec)
	lpusTANFirst	Pointer to first transaction number.
	lpusTANLast	Pointer to last transaction number.
Returns:		Error constant (DoPERR_xxxx)

6.8.3 DoPEConfCMcGradient(Sync)

DoPE_HANDLE	DP	
unsigned short	CalculatedSensorNo	
unsigned short	DividentSensorNo	
unsigned short	DivisorSensorNo	
double	IntegrationTime	
double	Timebase	
WORD FAR	** lpusTANFirst*	*(not for Sync. version)*
WORD FAR	** lpusTANLast*	*(not for Sync. version)*

Configure calculated gradient between two measured values to measuring data record.
A gradient between two measured values may be calculated and configured to the data record. After it is configured, it may be used as the supervised channel in the command "DoPESetCheck".
This calculated channel is not allowed as the command channel in DoPEExts2 or DoPEFDPoti command!

In:	DP	DoPE link handle
	CalculatedSensorNo	Sensor number for the calculated gradient (SENSOR_4 to SENSOR_15)
	DividentSensorNo	Sensor Number of the dividend
	DivisorSensorNo	Sensor Number of the divisor
	IntegrationTime	
	Timebase	Integration time for both dividend and divisor (only for analogue channels)
	lpusTANFirst	Pointer to first transaction number.
	lpusTANLast	Pointer to last transaction number.
Returns:		Error constant (DoPERR_xxxx)

6.8.4 DoPEClearCMc(Sync)

DoPE_HANDLE	DP	
unsigned short	CalculatedSensorNo	
WORD FAR	** lpusTANFirst*	*(not for Sync. version)*
WORD FAR	** lpusTANLast*	*(not for Sync. version)*

Clear calculated measuring channel.

In:	DP	DoPE link handle
	CalculatedSensorNo	Clear this Sensor number (SENSOR_4 to SENSOR_15)
	lpusTANFirst	Pointer to first transaction number.
	lpusTANLast	Pointer to last transaction number.
Returns:		Error constant (DoPERR_xxxx)

6.8.5 DoPEMc2Output(Sync)

DoPE_HANDLE	DP	
unsigned short	Mode	
unsigned short	SensorNo	
double	OffsetSensor	
unsigned short	Output	
double	OffsetOutput	
double	Scale	
WORD FAR	**lpusTAN*	*(not for Sync. version)*

Output of a measured value to a analogue output channel. Any measured channel may be scaled and via a DAC converted to a analogue signal. If active, the analogue output is calculated every 20 ms. With the Mode parameter the calculation of the output is controlled:

MC2OUT_OFF	Stop output to this analogue output channel.
MC2OUT_HIGH_PRIORITY	Output of this channel every 20 ms.
MC2OUT_LOW_PRIORITY	Every 20 ms only one of the channels with this priority is calculated and given output.

$$AnalougeOutput \;=\; ((MeasuredValue - OffsetSensor) \;*\; Scale) + OffsetOutput$$

In:	DP	DoPE link handle
	Mode	Mode flag (see above)
	SensorNo	Sensor number
	OffsetSensor	Offset of Sensor
	Output	Number of analogue output channel
	OffsetOutput	Offset of output channel
	Scale	Scale
	lpusTAN	Pointer to transaction number.

Returns: Error constant (DoPERR_xxxx)

6.8.6 DoPEConfPeakValue(Sync)

DoPE_HANDLE	DP	
unsigned short	PositionMin	
unsigned short	PositionMax	
unsigned short	LoadMin	
unsigned short	LoadMax	
unsigned short	ExtensionMin	
unsigned short	ExtensionMax	
WORD FAR	** lpusTANFirst*	*(not for Sync. version)*
WORD FAR	** lpusTANLast*	*(not for Sync. version)*

Configure peak values to measuring data record.
The peak values are detected by the EDC. They may be transmitted to PC instead of an unused measuring channel. With this command the peak values for X-head position, load and extension may be configured to measuring channels in the data record. SENSOR4 to SENSOR15 are allowed to be configured as peak values. E.g. use the constant SENSOR4 for PositionMin to configure the minimum value of X-head position to SENSOR4 within the measuring data record. Any number outside SENSOR4 to SENSOR15 will reset the measuring channels to the original values. Use this feature to cancel Peak Values configuration. This calculated channels are not allowed as the command channel in DoPEExts2 or DoPEFDPoti command!

In:	DP	DoPE link handle
	PositionMin	Position of Minimum value of XHead Position
	PositionMax	Position of Maximum value of XHead Position
	LoadMin	Position of Minimum value of Load
	LoadMax	Position of Maximum value of Load
	ExtensionMin	Position of Minimum value of Extension
	ExtensionMax	Position of Maximum value of Extension
	lpusTANFirst	Pointer to first transaction number.
	lpusTANLast	Pointer to last transaction number.
Returns:		Error constant (DoPERR_xxxx)

6.8.7 DoPEPeakValueTime(Sync)

DoPE_HANDLE	DP	
double	Time	
WORD FAR	**lpusTAN*	*(not for Sync. version)*

Set reset time for peak value detection.
The maximum and minimum values within this time are considered as peak values.

In:	DP	DoPE link handle
	Time	Reset time for peak value detection
	lpusTAN	Pointer to transaction number.
Returns:		Error constant (DoPERR_xxxx)

6.9 Reference signal handling of incremental sensors

Incremental sensors like encoder or linear gauges may use the build in reference to set the counter-value. Encoder have one reference pulse per revolution. Linear gauges have either one pulse at a certain position, or so called distance coded references.
Currently, DoPE supports the reference pulse to set the counter to a certain value.
ATTENTION: The hardware interfaces must support this function as well!
The two channels of a 4INC-interface support this function, the X-head channel (X7) of EDC not.
ATTENTION: This functions are only supported by EDC60/120 systems!

6.9.1 DoPESetRefSignalMode

DoPE_HANDLE	DoPEHdl
unsigned short	SensorNo
unsigned short	Mode
WORD FAR	*lpusTAN

After the reference pulse occurred, the counter value is stored in a Hardware register.
This command defines what to do with this value:

1. Ignore it, don't send any message. (REFSIG_NON)
2. Send a message after every occurrence of the reference pulse. (REFSIG_ON)
3. Send a message after the next occurrence of the reference pulse. (REFSIG_ONCE)

With the message (DoPERefSignalMsg) you get the exact position of the reference pulse.

In:	DoPEHdl	DoPE link handle
	SensorNo	Sensor Number
	Mode	reference signal messages will be reported: REFSIG_NON: never REFSIG_ON: always REFSIG_ONCE: only once
	lpusTAN	Pointer to Transaction-Number
Returns:		Error constant (DoPERR_xxxx)

```
typedef  struct  DoPERefSignalMsg      /* 'Reference Signal' Message           */
  {                                    /* ----------------------------------- */
    unsigned short     MsgId;          /* ID of message                       */
    double             Time;           /* System time for the message         */
    unsigned short     SensorNo;       /* Control mode of position            */
    double             Position;       /* Position                            */
  } DoPERefSignalM;
```

6.9.2 DoPESetRefSignalTare

DoPE_HANDLE	DoPEHdl
unsigned short	SensorNo
unsigned short	Mode
double	Tare
WORD FAR	*lpusTAN

With this function, the processing of the **next** reference pulse is done inside DoPE. The reference pulse is used to set the BasicTare-value to a certain value. The function DoPESetRefSignalMode is not necessary.

In:	DoPEHdl	DoPE link handle
	SensorNo	Sensor Number
	Mode	1 = At the next reference signal the measuring channel will be set to the tare value. Tare value will be stored in EDC's non volatile BasicTare memory. One DoPERefSignalMsg will be send. This function clears the current basic tare and ordinary tare value at the next occurrence of the reference signal. 0 = Reference signals don't affect the measuring channel. At the next reference signal, one DoPERefSignalMsg will be send.
	Tare	Value for the measuring channel
	lpusTAN	Pointer to Transaction-Number
Returns:		Error constant (DoPERR_xxxx)

6.10 Serial Sensors

6.10.1 DoPEWrSensorMsg (Sync)

DoPE_HANDLE	DoPEHdl	
unsigned short	SensorNo	
void	*Buffer	
unsigned	Length	
WORD FAR	**lpusTAN*	*(not for Sync. version)*

This function can be used to send a message to a **serial sensor**. **(only EDC60/120)**
The message is not interpreted by DoPE!
Maximum message length is 80 Bytes.

In:	DoPEHdl	DoPE link handle
	SensorNo	Sensor Number 0 .. 15
	*Buffer	Pointer to message to transmit
	Length	Message length in Byte’s
	lpusTAN	Pointer to Transaction-Number
Returns:		Error constant (DoPERR_xxxx)

7 Input / Output-Commands

7.1 Analogue output

7.1.1 DoPESetOutput(Sync)

DoPE_HANDLE	DP	
unsigned short	Output	
double	Value	
WORD FAR	**lpusTAN*	*(not for Sync. version)*

Set an analogue output channel. The output channel must be assigned in EDC set-up.

In:	DoPEHdl	DoPE link handle
	Output	Number of analogue output channel
	Value	New value of output channel
	lpusTAN	Pointer to Transaction-Number
Returns:		Error constant (DoPERR_xxxx)

7.1.2 DoPESetOutChannelOffset (Sync)

DoPE_HANDLE	DoPEHdl	
unsigned short	Output	
double	Offset	
WORD FAR	**lpusTAN*	*(not for Sync. version)*

Set an analogue output channel offset. **(only EDC60/120)**

In:	DoPEHdl	DoPE link handle
	Output	Number of analogue output channel
	Offset	New offset value of output channel
	lpusTAN	Pointer to Transaction-Number
Returns:		Error constant (DoPERR_xxxx)

7.1.3 DoPEOfflineActionOutput (Sync)

DoPE_HANDLE	DP	
unsigned short	OutputNo	
unsigned short	Mode	
double	Value	
WORD FAR	**lpusTAN*	*(not for Sync. version)*

Definition of an action for an initialised analogue output channel after EDC has detected offline.

With this command, the PC-Software can specify the state of any analogue output channel, after the communication between PC and EDC was disturbed or interrupted. (OFFLINE)

EDC-Software checks automatically communication with PC, and if PC does not answer for a period of 2 Seconds, EDC regards PC to be offline. This may happen if the line between PC and EDC was disconnected, or PC-Program has crashed.

If PC-Program terminates regularly by using DoPECloseLink command, EDC will reinitialise, and DoPEOfflineActionBitOutput command has no effect.

In:	DoPEHdl	DoPE link handle	
	OutputNo	Number of analogue output channel If closed loop control, Output channel 0 cannot be selected!	
	Mode	DO_NOTHING (0):	Don't modify this digital output
		USE_INIT_VALUE (1):	Use Initial value from set-up after offline
		USE_VALUE (2):	Use defined value after offline
	Value	Output value used in USE_VALUE mode in % of max. value	
	lpusTAN	Pointer to Transaction-Number	
Returns:		Error constant (DoPERR_xxxx)	

7.2 Bit I/O-Commands

These commands can be uses to read or write digital I/O’s.
Reading digital inputs is normally not necessary, since all configured inputs are automatically read and transferred in the measuring data record.
General digital bit outputs are set with the command DoPESetB. You have to assign the outputs you use in EDC set-up. Please refer to the document “Drive of Hardware-Modules of the EDC-Family” for definition of output devices and bits.
For dedicated output bits, DoPE supplies dedicated functions like DoPEBeep to activate/deactivate the beep at a EDC.
For Digital input/output bits that are not often used, DoPE provides functions to read or write this devises without having initialised it (not assigned in EDC set-up).

7.2.1 DoPESetB(Sync)

DoPE_HANDLE	DP	
unsigned short	BitOutputNo	
unsigned short	SetB	
unsigned short	ResB	
unsigned short	FlashB	
WORD FAR	**lpusTAN*	*(not for Sync. version)*

Set, Reset, Flash Bits. This command is used to set any output bit on a digital output device. The devices are specified in the set-up data.

In:	DoPEHdl	DoPE link handle
	BitOutputNo	Number of bit output device
	SetB	These bits will be set
	ResB	These bits will be reset
	FlashB	These bits 'flash'
	lpusTAN	Pointer to Transaction-Number
Returns:		Error constant (DoPERR_xxxx)

The three data words will be processed in the following sequence (important with conflicting data):

1.) Flashing bits.

2.) Resetting of the bits.

3.) Setting of the bits.

7.2.2 DoPECalOut(Sync)

DoPE_HANDLE	DP	
unsigned short	Cal	
WORD FAR	**lpusTAN*	*(not for Sync. version)*

Activate / deactivate calibration output on EDC. The calibration contact may be used to trigger a reference measuring system during load calibration.

In:	DoPEHdl	DoPE link handle
	Cal	TRUE activate Calibration output
		FALSE deactivate Calibration output

lpusTAN	Pointer to Transaction-Number
Returns:	Error constant (DoPERR_xxxx)

7.2.3 DoPEBeep(Sync)

DoPE_HANDLE	DP	
unsigned short	Beep	
WORD FAR	**lpusTAN*	*(not for Sync. version)*

Activate / deactivate beep on EDC.

In:	DoPEHdl	DoPE link handle
	Beep	TRUE activate Beep
		FALSE deactivate Beep
	lpusTAN	Pointer to Transaction-Number
Returns:		Error constant (DoPERR_xxxx)

7.2.4 DoPELed(Sync)

DoPE_HANDLE	DP	
unsigned short	LedOn	
unsigned short	LedOff	
unsigned short	LedFlash	
WORD FAR	**lpusTAN*	*(not for Sync. version)*

Switch On/Off LED's at EDC frontpanel. There are LED’s for Up, Down and Test at the frontpanel.

In:	DoPEHdl	DoPE link handle
	LedOn	These LED's will be set
	LedOff	These LED's will be reset
	LedFlash	These LED's 'flash'
	lpusTAN	Pointer to Transaction-Number
Returns:		Error constant (DoPERR_xxxx)

Constants for LED’s at frontpanel:

PE_LED_UP	Bit mask for LED 'UP'
PE_LED_DOWN	Bit mask for LED 'DOWN'
PE_LED_TEST	Bit mask for LED 'TEST'

7.2.5 DoPEUniOut(Sync)

DoPE_HANDLE	DP	
unsigned short	Output	
WORD FAR	**lpusTAN*	*(not for Sync. version)*

Activate/Deactivate universal digital output bits at EDC100/EDC120. The universal digital output bits are located at X2.
EDC100 offers **four** output bits, EDC120 has **eight**!
This digital outputs may be used for general purposes.
Attention: If the parameter “MaIo” in EDC set-up (general data -> MaIo) is set to “YES”, bit 0 and 1 are not accessible by this command. In this case the EDC uses this two output bits to give out the active machine set-up number.

In:	DoPEHdl	DoPE link handle
	Output	Bit 0 .. 15 represent the digital outputs. 0x0000 will reset all bits, 0x000F will set bit 0 to 3, 0x000C will reset bit 0 and 1and set bit 2 and 3)
	lpusTAN	Pointer to Transaction-Number
Returns:		Error constant (DoPERR_xxxx)

7.2.6 DoPEBypass(Sync)

DoPE_HANDLE	DP	
unsigned short	Bypass	
WORD FAR	**lpusTAN*	*(not for Sync. version)*

Activate/Deactivate bypass output.

In:	DoPEHdl	DoPE link handle
	Bypass	TRUE activates bypass output FALSE deactivates bypass output
	lpusTAN	Pointer to Transaction-Number
Returns:		Error constant (DoPERR_xxxx)

7.2.7 DoPERdBitInput

DoPE_HANDLE	DP
WORD	Connector
WORD FAR	*Value

Read an digital input device. This command will also work on not initialised I/O-devices.

In:	DoPEHdl	DoPE link handle
	Connector	Connector number, for the I/O device. Use connector constants from header file like "`CON_X1`"
	Value	Data pointer to store result.
Returns:		Error constant (DoPERR_xxxx)

7.2.8 DoPEWrBitOutput(Sync)

DoPE_HANDLE	DP	
WORD	Connector	
WORD	Value	
WORD FAR	**lpusTAN*	*(not for Sync. version)*

Write to a digital output device. This command will also work on not initialised I/O-devices.

In:	DoPEHdl	DoPE link handle
	Connector	Connector number, for the I/O device. Use connector constants from header file like "`CON_X1`"
	Value	Data to be written to the output device.
Returns:		Error constant (DoPERR_xxxx)

7.2.9 DoPEOfflineActionBitOutput (Sync)

DoPE_HANDLE	DP	
unsigned short	BitOutputNo	
unsigned short	Mode	
unsigned short	Value	
WORD FAR	**lpusTAN*	*(not for Sync. version)*

Definition of an action for an initialised digital output device after EDC has detected offline.

With this command, the PC-Software can specify the state of any digital output, after the communication between PC and EDC was disturbed or interrupted. (OFFLINE)

EDC-Software checks automatically communication with PC, and if PC does not answer for a period of 2 Seconds, EDC regards PC to be offline. This may happen if the line between PC and EDC was disconnected, or PC-Program has crashed.

If PC-Program terminates regularly by using DoPECloseLink command, EDC will reinitialise, and DoPEOfflineActionBitOutput command has no effect.

In:	DoPEHdl	DoPE link handle	
	BitOutputNo	Number of bit output device	
	Mode	DO_NOTHING (0):	Don't modify this digital output
		USE_INIT_VALUE (1):	Use Initial value from set-up after offline
		USE_VALUE (2):	Use defined value after offline
	Value	Output value used in USE_VALUE mode	
	lpusTAN	Pointer to Transaction-Number	
Returns:		Error constant (DoPERR_xxxx)	

7.3 Display Commands

7.3.1 DoPEDspClear(Sync)

DoPE_HANDLE	DP	
WORD FAR	**lpusTAN*	*(not for Sync. version)*

Clear LCD-display at EDC front panel

In:	DoPEHdl	DoPE link handle
	lpusTAN	Pointer to Transaction-Number
Returns:		Error constant (DoPERR_xxxx)

7.3.2 DoPEDspHeadLine(Sync)

DoPE_HANDLE	DP	
char FAR	*HeadLine	
WORD FAR	**lpusTAN*	*(not for Sync. version)*

Display HeadLine on LCD-display at EDC front panel

In:	DoPEHdl	DoPE link handle
	HeadLine	Character string
	lpusTAN	Pointer to Transaction-Number
Returns:		Error constant (DoPERR_xxxx)

7.3.3 DoPEDspFKeys(Sync)

DoPE_HANDLE	DP	
char FAR	*FKeys	
WORD FAR	**lpusTAN*	*(not for Sync. version)*

Display function keys on LCD-display at EDC front panel

In:	DoPEHdl	DoPE link handle
	FKeys	Character string
	lpusTAN	Pointer to Transaction-Number
Returns:		Error constant (DoPERR_xxxx)

7.3.4 DoPEDspMValue(Sync)

DoPE_HANDLE	DP	
char FAR	*Value1	
char FAR	*Value2	
char FAR	*Dim1	
char FAR	*Dim2	
WORD FAR	**lpusTAN*	*(not for Sync. version)*

Display data and dimensions on LCD-display at EDC front panel

In:	DoPEHdl	DoPE link handle
	Value1	Character string for first value
	Value2	Character string for second value
	Dim1	Character string for first dimension
	Dim2	Character string for second dimension
	lpusTAN	Pointer to Transaction-Number
Returns:		Error constant (DoPERR_xxxx)

7.3.5 DoPEDspBeamScreen(Sync)

DoPE_HANDLE	DP	
short	Value	
WORD FAR	**lpusTAN*	*(not for Sync. version)*

Display frame and beam on LCD-display at EDC front panel.

In:	DoPEHdl	DoPE link handle
	Value	Value of the beam in %
	lpusTAN	Pointer to Transaction-Number
Returns:		Error constant (DoPERR_xxxx)

7.3.6 DoPEDspBeam Value(Sync)

DoPE_HANDLE	DP	
short	Value	
WORD FAR	**lpusTAN*	*(not for Sync. version)*

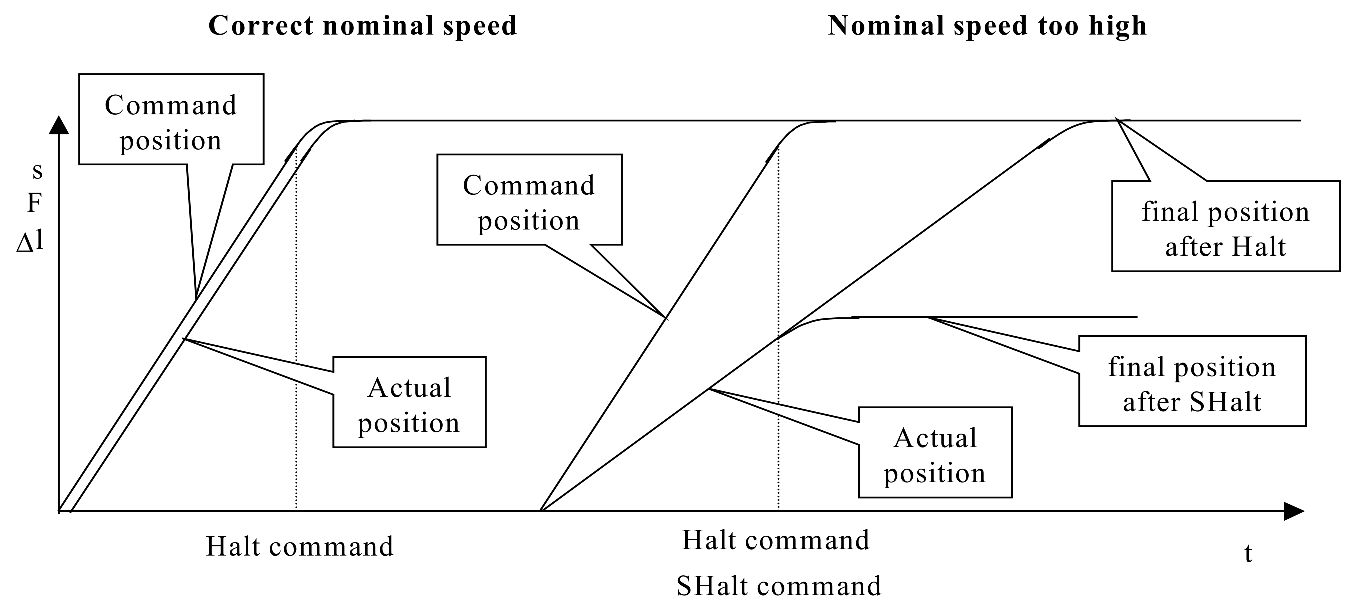
Display beam on LCD-display at EDC front panel.

In:	DoPEHdl	DoPE link handle
	Value	Value of the beam in %
	lpusTAN	Pointer to Transaction-Number
Returns:		Error constant (DoPERR_xxxx)

8 Movement commands

8.1 Halt commands

HALT is understood as a deceleration from current speed to speed zero. This is done by the build in position generator. If not specified by the halt command default deceleration will be used. The final destination position cannot be specified, it depends on current speed and deceleration.
In case nominal speed is not correctly defined and the controller error is set to high values or its action is set to state, there might be a remarkable difference between command value and actual position. After a halt command the machine will still move until the actual value reaches the command value.



8.1.1 DoPEHalt(Sync)

DoPE_HANDLE	DP
unsigned short	MoveCtrl

```
            WORD  FAR                    *lpusTAN
```

Halt movement of cross-head in the specified control mode. Default deceleration will be used. After cross-head is halted, message will be transmitted.

In:	DoPEHdl	DoPE link handle
	MoveCtrl	Control mode for halt (CTRL_POS, CTRL_LOAD or CTRL_EXTENSION)
	lpusTAN	Pointer to Transaction-Number
Returns:		Error constant (DoPERR_xxxx)

8.1.2 DoPEHalt_A(Sync)

DoPE_HANDLE	DP
unsigned short	MoveCtrl
double	Dec
WORD FAR	*lpusTAN

Halt movement of cross-head in the specified control mode. Deceleration is a parameter of the command. After cross-head is halted, message will be transmitted.

In:	DoPEHdl	DoPE link handle
	MoveCtrl	Control mode for halt (CTRL_POS, CTRL_LOAD or CTRL_EXTENSION)
	Dec	Deceleration
	lpusTAN	Pointer to Transaction-Number
Returns:		Error constant (DoPERR_xxxx)

8.1.3 DoPEHaltW(Sync)

DoPE_HANDLE	DP
unsigned short	MoveCtrl
double	Delay
WORD FAR	*lpusTAN

Halt movement of cross-head in the specified control mode. Default deceleration will be used. After cross-head is halted and the specified delay time is over a message will be transmitted.

In:	DoPEHdl	DoPE link handle
	MoveCtrl	Control mode for halt (CTRL_POS, CTRL_LOAD or CTRL_EXTENSION)
	Delay	Delay time
	lpusTAN	Pointer to Transaction-Number
Returns:		Error constant (DoPERR_xxxx)

8.1.4 DoPEHaltW_A(Sync)

DoPE_HANDLE	DP
unsigned short	MoveCtrl
double	Dec
double	Delay
WORD FAR	*lpusTAN

Halt movement of cross-head in the specified control mode. Deceleration is a parameter of the command. After cross-head is halted and the specified delay time is over a message will be transmitted.

In:	DoPEHdl	DoPE link handle
	MoveCtrl	Control mode for halt (CTRL_POS, CTRL_LOAD or CTRL_EXTENSION)
	Dec	Deceleration
	Delay	Delay time
	lpusTAN	Pointer to Transaction-Number
Returns:		Error constant (DoPERR_xxxx)

8.1.5 DoPESHalt(Sync)

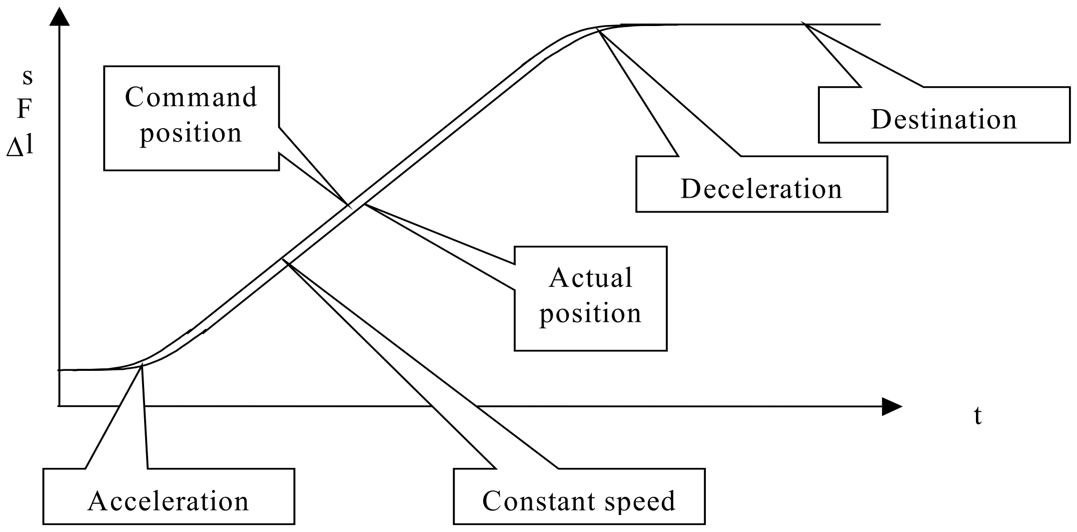
DoPE_HANDLE	DP
WORD FAR	*lpusTAN

Halt movement of cross-head in position control mode. Instant start of deceleration (command value = measured value). Default deceleration will be used. After cross-head is halted, message will be transmitted.

In:	DoPEHdl	DoPE link handle
	lpusTAN	Pointer to Transaction-Number
Returns:		Error constant (DoPERR_xxxx)

8.2 Simple position commands

Simple positioning commands move e.g. in cross head position control to a new cross head position. The Movement Control Type is equal to the destination type.



8.2.1 DoPEPos(Sync)

DoPE_HANDLE	DP
unsigned short	MoveCtrl
double	Speed
double	Destination
WORD FAR	*lpusTAN

Move cross-head in the specified control mode and speed to the given destination. Default acceleration and deceleration will be used. After destination has been reached, a message will be transmitted.

In:	DoPEHdl	DoPE link handle
	MoveCtrl	Control mode for positioning (CTRL_POS, CTRL_LOAD or CTRL_EXTENSION)
	Speed	Speed for positioning
	Destination Final destination	
	lpusTAN	Pointer to Transaction-Number
Returns:		Error constant (DoPERR_xxxx)

8.2.2 DoPEPos_A(Sync)

DoPE_HANDLE	DP
unsigned short	MoveCtrl
double	Acc
double	Speed
double	Dec
double	Destination
WORD FAR	*lpusTAN

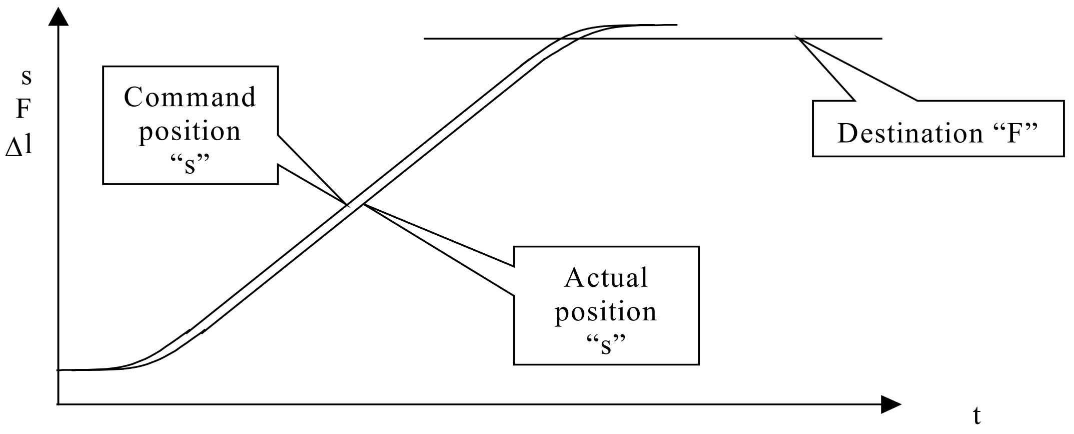
Move cross-head in the specified control mode and speed to the given destination. Acceleration and deceleration are parameters of the command. After destination has been reached, a message will be transmitted.

In:	DoPEHdl	DoPE link handle
	MoveCtrl	Control mode for positioning (CTRL_POS, CTRL_LOAD or CTRL_EXTENSION)
	Acc	Acceleration
	Speed	Speed for positioning
	Dec	Deceleration
	Destination	Final destination
	lpusTAN	Pointer to Transaction-Number

Returns: Error constant (DoPERR_xxxx)

8.3 Approach of an external destination position

This commands move e.g. in cross-head position control to a certain load or extension (destination). Control mode in not changed after reaching the destination. That is why it is not possible to position exactly to the destination position.



8.3.1 DoPEPosG1(Sync)

DoPE_HANDLE	DP
unsigned short	MoveCtrl
double	Speed
double	Limit
unsigned short	DestCtrl
double	Destination
WORD FAR	*lpusTAN

Move cross-head in the specified control mode and speed to the given destination. Destination must be different to move control. Default acceleration and deceleration will be used. After destination or the absolute limit position has been reached a message will be transmitted. This command will not change control mode after the destination is reached.

In:	DoPEHdl	DoPE link handle
	MoveCtrl	Control mode for positioning (CTRL_POS, CTRL_LOAD or CTRL_EXTENSION)
	Speed	Speed for positioning
	Limit	absolute limit position
	DestCtrl	Channel definition for destination (CTRL_POS, CTRL_LOAD or CTRL_EXTENSION)
	Destination	Final destination
	lpusTAN	Pointer to Transaction-Number
Returns:		Error constant (DoPERR_xxxx)

8.3.2 DoPEPosG1_A(Sync)

DoPE_HANDLE	DP
unsigned short	MoveCtrl
double	Acc
double	Speed
double	DecLimit
double	Limit
double	DecDest
unsigned short	DestCtrl
double	Destination
WORD FAR	*lpusTAN

Move cross-head in the specified control mode and speed to the given destination. Destination may be different to move control. Acceleration and deceleration are parameters of the command. After destination or the absolute limit position has been reached a message will be transmitted. This command will not change control mode after the destination is reached.

In:	DoPEHdl	DoPE link handle
	MoveCtrl	Control mode for positioning (position, load or extension)
	Acc	Acceleration
	Speed	Speed for positioning (CTRL_POS, CTRL_LOAD or CTRL_EXTENSION)
	DecLimit	Deceleration for limit position
	Limit	absolute limit position
	DecDest	Deceleration for final destination
	DestCtrl	Channel definition for destination (CTRL_POS, CTRL_LOAD or CTRL_EXTENSION)
	Destination	Final destination
	lpusTAN	Pointer to Transaction-Number
Returns:		Error constant (DoPERR_xxxx)

8.3.3 DoPEPosD1(Sync)

DoPE_HANDLE	DP
unsigned short	MoveCtrl
double	Speed
double	Limit
unsigned short	DestCtrl,
double	Destination
WORD FAR	*lpusTAN

Move cross-head in the specified control mode and speed to the given destination. Destination may be different to move control. Default acceleration and deceleration will be used. After destination or the relative limit position has been reached a message will be transmitted. This command will not change control mode after the destination is reached.

In:	DoPEHdl	DoPE link handle
	MoveCtrl	Control mode for positioning (CTRL_POS, CTRL_LOAD or CTRL_EXTENSION)
	Speed	Speed for positioning
	Limit	relative limit position (e.g. current position + 10)
	DestCtrl	Channel definition for destination (CTRL_POS, CTRL_LOAD or CTRL_EXTENSION)
	Destination	Final destination
	lpusTAN	Pointer to Transaction-Number
Returns:		Error constant (DoPERR_xxxx)

8.3.4 DoPEPosD1_A(Sync)

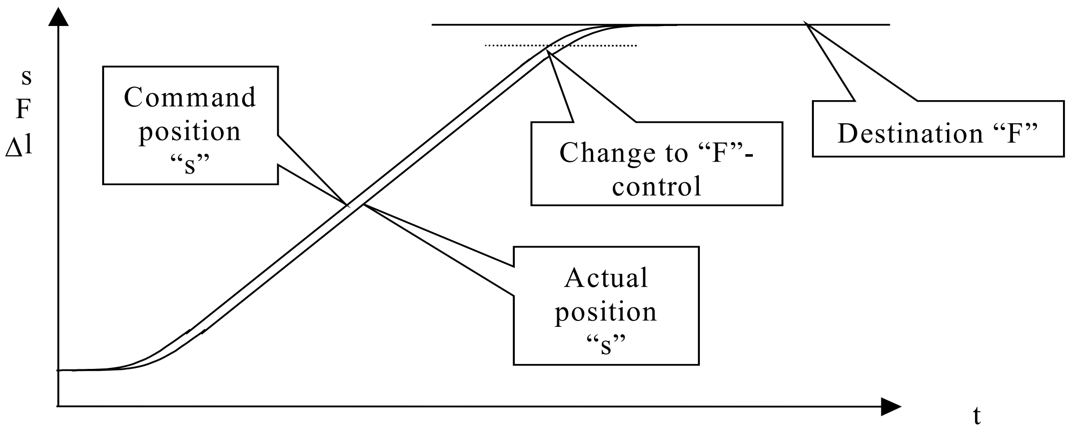
DoPE_HANDLE	DP
unsigned short	MoveCtrl
double	Acc
double	Speed
double	DecLimit
double	Limit
double	DecDest
unsigned short	DestCtrl
double	Destination
WORD FAR	*lpusTAN

Move cross-head in the specified control mode and speed to the given destination. Destination may be different to move control. Acceleration and deceleration are parameters of the command. After destination or the relative limit position has been reached a message will be transmitted. This command will not change control mode after the destination is reached.

In:	DoPEHdl	DoPE link handle
	MoveCtrl	Control mode for positioning (CTRL_POS, CTRL_LOAD or CTRL_EXTENSION)
	Acc	Acceleration
	Speed	Speed for positioning
	DecLimit	Deceleration for limit position
	Limit	relative limit position (e.g. current position + 10)
	DecDest	Deceleration for final destination
	DestCtrl	Channel definition for destination (CTRL_POS, CTRL_LOAD or CTRL_EXTENSION)
	Destination	Final destination
	lpusTAN	Pointer to Transaction-Number
Returns:		Error constant (DoPERR_xxxx)

8.4 Positioning to an external destination position

This commands move e.g. in cross-head position control to a certain load or extension (destination). Control mode in changed in time to be able to decelerate exactly to the destination position.



8.4.1 DoPEPosG2(Sync)

DoPE_HANDLE	DP
unsigned short	MoveCtrl
double	Speed
double	Limit
unsigned short	DestCtrl
double	Destination
WORD FAR	*lpusTAN

Move cross-head in the specified control mode and speed to the given destination. Destination may be different to move control. Default acceleration and deceleration will be used. After destination or the absolute limit position has been reached a message will be transmitted. This command will change control mode before the destination is reached and positions exactly.

In:	DoPEHdl	DoPE link handle
	MoveCtrl	Control mode for positioning (position, load or extension) (CTRL_POS, CTRL_LOAD or CTRL_EXTENSION)
	Speed	Speed for positioning
	Limit	absolute limit position
	DestCtrl	Channel definition for destination (CTRL_POS, CTRL_LOAD or CTRL_EXTENSION)
	Destination	Final destination
	lpusTAN	Pointer to Transaction-Number
Returns:		Error constant (DoPERR_xxxx)

8.4.2 DoPEPosG2_A(Sync)

DoPE_HANDLE	DP
unsigned short	MoveCtrl
double	Acc
double	Speed
double	DecLimit
double	Limit
unsigned short	DestCtrl
double	DecDest
double	Destination
WORD FAR	*lpusTAN

Move cross-head in the specified control mode and speed to the given destination. Destination may be different to move control. Acceleration and deceleration are parameters of the command. After destination or the absolute limit position has been reached a message will be transmitted. This command will change control mode before the destination is reached and positions exactly.

In:	DoPEHdl	DoPE link handle
	MoveCtrl	Control mode for positioning (CTRL_POS, CTRL_LOAD or CTRL_EXTENSION)
	Acc	Acceleration
	Speed	Speed for positioning
	DecLimit	Deceleration for limit position
	Limit	absolute limit position
	DecDest	Deceleration for final destination
	DestCtrl	Channel definition for destination (CTRL_POS, CTRL_LOAD or CTRL_EXTENSION)
	Destination	Final destination
	lpusTAN	Pointer to Transaction-Number

Returns: Error constant (DoPERR_xxxx)

8.4.3 DoPEPosD2(Sync)

DoPE_HANDLE	DP
unsigned short	MoveCtrl
double	Speed
double	Limit
unsigned short	DestCtrl
double	Destination
WORD FAR	*lpusTAN

Move cross-head in the specified control mode and speed to the given destination. Destination may be different to move control. Default acceleration and deceleration will be used. After destination or the relative limit position has been reached a message will be transmitted. This command will change control mode before the destination is reached and positions exactly.

In:	DoPEHdl	DoPE link handle
	MoveCtrl	Control mode for positioning (CTRL_POS, CTRL_LOAD or CTRL_EXTENSION)
	Speed	Speed for positioning
	Limit	relative limit position (e.g. current position + 10)
	DestCtrl	Channel definition for destination (CTRL_POS, CTRL_LOAD or CTRL_EXTENSION)
	Destination	Final destination
	lpusTAN	Pointer to Transaction-Number
Returns:		Error constant (DoPERR_xxxx)

8.4.4 DoPEPosD2_A(Sync)

DoPE_HANDLE	DP
unsigned short	MoveCtrl
double	Acc
double	Speed
double	DecLimit
double	Limit
unsigned short	DestCtrl
double	DecDest
double	Destination
WORD FAR	*lpusTAN

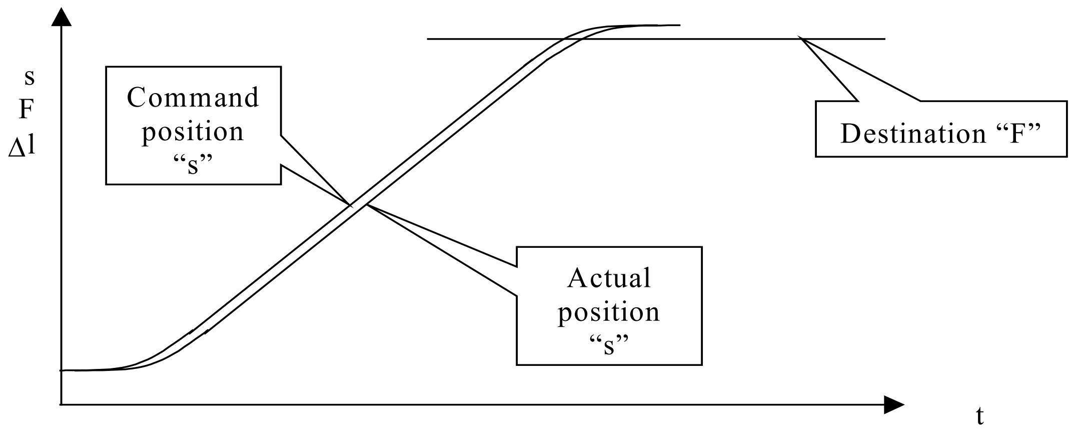
Move cross-head in the specified control mode and speed to the given destination. Destination may be different to move control. Acceleration and deceleration are parameters of the command. After destination or the relative limit position has been reached a message will be transmitted. This command will change control mode before the destination is reached and positions exactly.

In:	DoPEHdl	DoPE link handle
	MoveCtrl	Control mode for positioning (position, load or extension) (CTRL_POS, CTRL_LOAD or CTRL_EXTENSION)
	Acc	Acceleration
	Speed	Speed for positioning
	DecLimit	Deceleration for limit position
	Limit	relative limit position (e.g. current position + 10)
	DecDest	Deceleration for final destination
	DestCtrl	Channel definition for destination (CTRL_POS, CTRL_LOAD or CTRL_EXTENSION)
	Destination	Final destination
	lpusTAN	Pointer to Transaction-Number
Returns:		Error constant (DoPERR_xxxx)

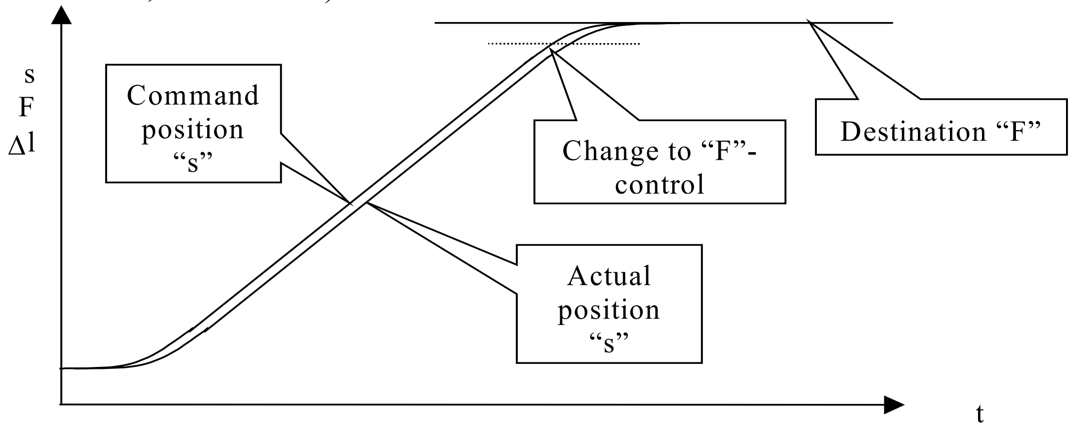
8.5 General position command to a external destination position

This commands move e.g. in cross-head position control to a certain load or extension (destination). At destination, three different modes are possible:

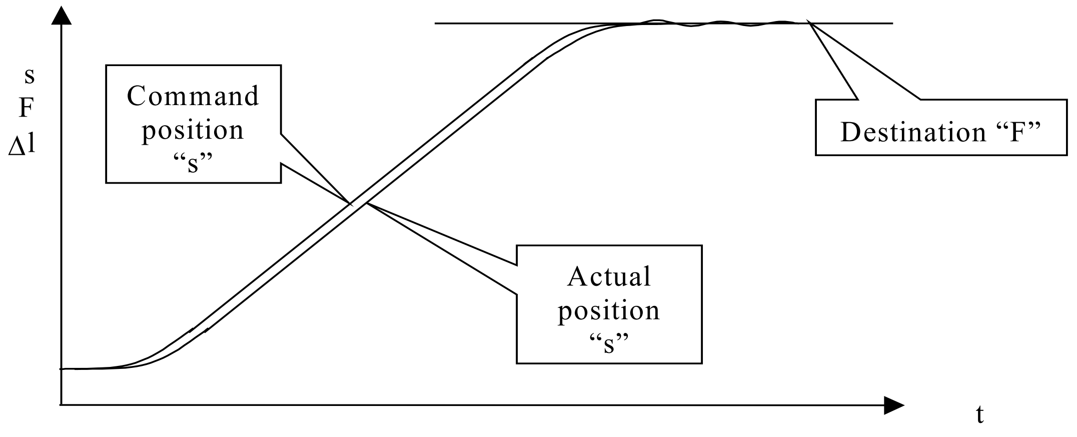
1. DEST_APPROACH: Halt in ‘MoveCtrl’ after destination is reached. (like DoPEPosG1 and DoPEPosD1)



2. DEST_POSITION: Change control mode to ‘DestCtrl’ and position to destination (like DoPEPosG2, DoPEPosD2)



3. DEST_MAINTAIN: Maintain destination in ‘MoveCtrl’ e.g. keep a load constant in position control.



8.5.1 DoPEPosExt(Sync)

DoPE_HANDLE	DP
unsigned short	MoveCtrl
double	Speed
unsigned short	LimitMode
double	Limit
unsigned short	DestinationCtrl
double	Destination
unsigned short	DestinationMode
WORD FAR	*lpusTAN

Move cross-head in the specified control mode ‘MoveCtrl’ and speed to the given destination. ‘DestinationCtrl’ may be different to ‘MoveCtrl’. Default acceleration and deceleration will be used.
The Parameter ‘DestinationMode’ specifies how to reach destination position and the action after reaching it. In case the limit position is reached before destination position, cross-head will be halted in ‘MoveCtrl’ at limit position.
After destination or the limit position has been reached a message will be transmitted.

This command is not available for EDC5/25 and EDC100 !!!

In:	DoPEHdl	DoPE link handle	
	MoveCtrl	Control mode for positioning (CTRL_POS, CTRL_LOAD or CTRL_EXTENSION)	
	Speed	Speed for positioning	
	LimitMode	LIMIT_ABSOLUTE:	Limit is absolute.
		LIMIT_RELATIVE:	Limit is a distance from the actual position.
	Limit	Limit position	
	DestinationCtrl	Channel definition for destination (CTRL_POS, CTRL_LOAD or CTRL_EXTENSION)	
	Destination	Final destination	
	DestinationMode	DEST_APPROACH:	Halt in ‘MoveCtrl’ after destination is reached. (like DoPEPosG1, DoPEPosD1)
		DEST_POSITION:	Change control mode to ‘DestCtrl’ and position to destination (like DoPEPosG2, DoPEPosD2)
		DEST_MAINTAIN:	Maintain destination in ‘MoveCtrl’ e.g. keep a load constant in position control.
	lpusTAN	Pointer to Transaction-Number	
Returns:		Error constant (DoPERR_xxxx)	

8.5.2 DoPEPosExt_A(Sync)

DoPE_HANDLE	DP
unsigned short	MoveCtrl
double	Acc
double	Speed
double	DecLimit
unsigned short	LimitMode
double	Limit
unsigned short	DestinationCtrl
double	DecDestination
double	Destination
unsigned short	DestinationMode
WORD FAR	*lpusTAN

Move cross-head in the specified control mode 'MoveCtrl' and speed to the given destination. 'DestinationCtrl' may be different to 'MoveCtrl'. The specified acceleration and deceleration will be used.
The Parameter 'DestinationMode' specifies how to reach destination position and the action after reaching it. In case the limit position is reached before destination position, cross-head will be halted in 'MoveCtrl' at limit position.
After destination or the limit position has been reached a message will be transmitted.

This command is not available for EDC5/25 and EDC100 !!!

In:	DoPEHdl	DoPE link handle	
	MoveCtrl	Control mode for positioning (position, load or extension)	
	Acc	Acceleration	
	Speed	Speed for positioning (CTRL_POS, CTRL_LOAD or CTRL_EXTENSION)	
	DecLimit	Deceleration for limit position	
	LimitMode	LIMIT_ABSOLUTE:	Limit is absolute.
		LIMIT_RELATIVE:	Limit is a distance from the actual position.
	Limit	Limit position	
	DestinationCtrl	Channel definition for destination	
	DecDestination	Deceleration for final destination (CTRL_POS, CTRL_LOAD or CTRL_EXTENSION)	
	Destination	Final destination	
	DestinationMode	DEST_APPROACH:	Halt in 'MoveCtrl' after destination is reached. (like DoPEPosG1_A, DoPEPosD1_A)
		DEST_POSITION:	Change control mode to 'DestCtrl' and position to destination (like DoPEPosG2_A, DoPEPosD2_A)
		DEST_MAINTAIN:	Maintain destination in 'MoveCtrl' e.g. keep a load constant in position control.
	lpusTAN	Pointer to Transaction-Number	
Returns:		Error constant (DoPERR_xxxx)	

8.5.3 DoPEFMove(Sync)

DoPE_HANDLE	DP
unsigned short	Direction
unsigned short	MoveCtrl
double	Speed
WORD FAR	*lpusTAN

Move cross-head in the specified control mode and speed UP or DOWN. As an implicit limit of this command, softend's are used.

In:	DoPEHdl	DoPE link handle
	Direction	Direction of movement (MOVE_UP, MOVE_DOWN, MOVE_HALT)
	MoveCtrl	Control mode of movement
	Speed	Speed for movement
	lpusTAN	Pointer to Transaction-Number
Returns:		Error constant (DoPERR_xxxx)

8.5.4 DoPEXpCont(Sync)

DoPE_HANDLE	DP
unsigned short	MoveCtrl
double	Limit
WORD FAR	*lpusTAN

Change control mode and continue movement in the new control mode with the actual speed.

In:	DoPEHdl	DoPE link handle
	MoveCtrl	Control mode for movement (CTRL_POS, CTRL_LOAD or CTRL_EXTENSION)
	Limit	Limit position for movement
	lpusTAN	Pointer to Transaction-Number
Returns:		Error constant (DoPERR_xxxx)

8.6 Trigger

8.6.1 DoPETrig(Sync)

DoPE_HANDLE	DP
unsigned short	MoveCtrl
double	Speed
double	Limit
unsigned short	TriggerCtrl
double	Trigger
WORD FAR	*lpusTAN

Move cross-head with the specified speed to the limit position. Default acceleration and deceleration will be used. If the trigger position is reached a message will be transmitted and if used inside a combined moving sequence, the next command is activated.

In:	DoPEHdl	DoPE link handle
	MoveCtrl	Control mode for moving (CTRL_POS, CTRL_LOAD or CTRL_EXTENSION)
	Speed	Speed for moving
	Limit	Absolute limit position for movement TriggerCtrl Sensor number of trigger channel (CTRL_POS, CTRL_LOAD or CTRL_EXTENSION)
	Trigger	Position
	lpusTAN	Pointer to Transaction-Number
Returns:		Error constant (DoPERR_xxxx)

8.6.2 DoPETrig_A(Sync)

DoPE_HANDLE	DP
unsigned short	MoveCtrl
double	Acc
double	Speed
double	Dec
double	Limit
unsigned short	TriggerCtrl
double	Trigger
WORD FAR	*lpusTAN

Move cross-head with the specified speed to the limit position. Acceleration and deceleration are parameters of the command. If the trigger position is reached a message will be transmitted and if used inside a combined moving sequence, the next command is activated.

In:	DoPEHdl	DoPE link handle
	MoveCtrl	Control mode for moving (CTRL_POS, CTRL_LOAD or CTRL_EXTENSION)
	Acc	Acceleration
	Speed	Speed for moving
	Dec	Deceleration
	Limit	Absolute limit position for movement TriggerCtrl Sensor number of trigger channel (CTRL_POS, CTRL_LOAD or CTRL_EXTENSION)
	Trigger	Position
	lpusTAN	Pointer to Transaction-Number
Returns:		Error constant (DoPERR_xxxx)

8.7 Combined movement commands

8.7.1 DoPEStartCMD(Sync)

DoPE_HANDLE	DP
unsigned long	Cycles
unsigned short	ModeFlags
WORD FAR	*lpusTAN

Start of a combined movement command. Up to 10 simple moving commands may be send after this command to specify a complex moving sequence.

In:	DoPEHdl	DoPE link handle
	Cycles	Repeat combined moving command this number of cycles
	ModeFlags	Flags (see below). Both flags may be ored.
	lpusTAN	Pointer to Transaction-Number
Returns:		Error constant (DoPERR_xxxx)

Definition of ModeFlags:

CMD_DWND	Supervise destination window. Next command is started after the actual value reaches destination window. Otherwise the next command is started, after the command value reaches it's destination.
CMD_MESSAGE	Report intermediate destinations. For each single command a message is send after it was finished. Otherwise a message is send after the last command in the sequence is finished.

8.7.2 DoPEEndCMD(Sync)

DoPE_HANDLE	DP
unsigned short	Operation
WORD FAR	*lpusTAN

End of combined moving command. With this command the first moving command inside this sequence will be started.

In:	DoPEHdl	DoPE link handle
	Operation	Start (CMD_START) or discard (CMD_DISCARD) the sequence
	lpusTAN	Pointer to Transaction-Number
Returns:		Error constant (DoPERR_xxxx)

9 Complex moving commands

9.1 DoPEExt2Ctrl(Sync)

DoPE_HANDLE	DP
unsigned short	MoveCtrl
double	OffsetCtrl
unsigned short	SensorNo
double	OffsetSensor
unsigned short	Mode
double	Scale
WORD FAR	*lpusTAN

Move cross-head according to an external command signal. You may supply a random function to a A/D converter and specify movement according to this random function. Scaling of the command- and sensor-system is done by offsets and scale. Following formula is used:

$$Output = (externalCommandValue - OffsetSensor) \bullet Scale + OffsetCtrl$$

In:	DP	DoPE link handle
	MoveCtrl	Control mode for positioning (CTRL_POS, CTRL_LOAD or CTRL_EXTENSION)
	OffsetCtrl	Offset for position, load or extension
	SensorNo	Sensor number for the external command signal
	OffsetSensor	Offset for external command signal
	Mode	various position or speed control modes (see next page)
	Scale	Scaling factor for external command signal: Scale = 1 for POSITION mode: Nominal value of external command (e.g. 10V, or 1 round of a encoder) represents one unit of the control channel (e.g. 1mm, or 1N, or 1kN) Scale = 1 for SPEED mode: Nominal value of external command (e.g. 10V) (e.g. 10V, or 1 round of a encoder) represents nominal speed of the control channel (e.g. 20mm/s, or 100kN/s) Scale = 1 for OPENLOOP mode: Nominal value of external command (e.g. 10V, or 1 round of a encoder) represents 100% output
	lpusTAN	Pointer to Transaction-Number
Returns:		Error constant (DoPERR_xxxx)

9.2 DoPEFDPoti(Sync)

DoPE_HANDLE	DP
unsigned short	MoveCtrl
double	MaxSpeed
unsigned short	SensorNo
unsigned short	DxTrigger
unsigned short	Mode
double	Scale
WORD FAR	*lpusTAN

Move cross-head according to an external command signal generated by a digital encoder (DigiPoti). This is a special version of the DoPEExt2Ctrl command. Offsets and limits are handled inside this function.

In:	DoPEHdl	DoPE link handle
	MoveCtrl	Control mode for movement (CTRL_POS, CTRL_LOAD or CTRL_EXTENSION)
	MaxSpeed	Speed offset for speed controlled modes
	SensorNo	Sensor Number for the external command input. Use Sensor number 9 for Digital Encoder (DigiPoti) on EDC
	frontpanel.	
	DxTrigger	Dead area of encoder. The Encoder has to change the specified number of digits before the command is active. For EDC frontpanel DigiPoti 2 or 3 is a good value.
	Mode	various position or speed control modes (see below)
	Scale	For EXT_POSITION Number of Units per revolution e.g. Scale = 1 -> 1 mm per revolution Scale = 10 -> 10 N per revolution. For EXT_SPEED_xx Number of revolutions to nominal speed e.g. Scale = 2 -> after 2 revolutions to nominal speed.
	lpusTAN	Pointer to Transaction-Number
Returns:		Error constant (DoPERR_xxxx)

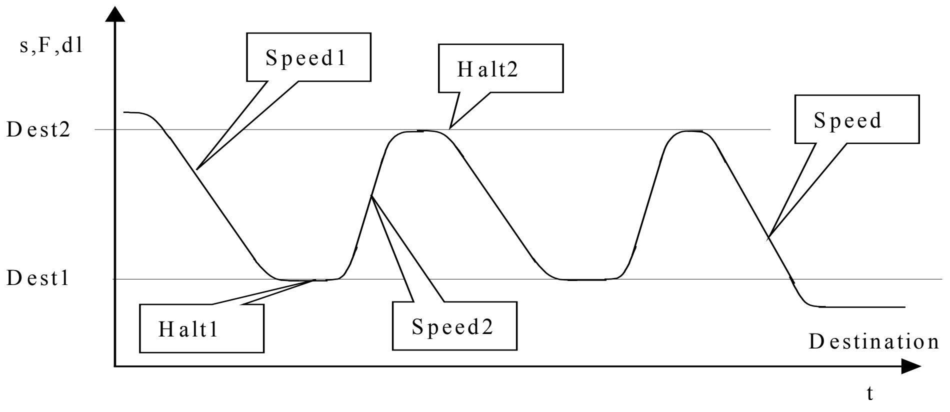
Various position or speed control modes:
The first four modes refer to measuring values. E.g. EXT_SPEED_POSITIVE moves to increasing measuring values.

EXT_POSITION	0	Position, moves to increasing and decreasing positions
EXT_SPEED_BIPOLAR	1	Speed bipolar (positive and negative speed)
EXT_SPEED_POSITIVE	2	Speed positive direction
EXT_SPEED_NEGATIVE	3	Speed negative direction

The next four modes refer to movement UP/DOWN. E.g. EXT_SPEED_UP moves Cross-head UP. The relation between measured values and movement Up/Down is defined in the set-up data.

EXT_POS_UP_DOWN	4	Position Up / Down
EXT_SPEED_UP_DOWN	5	Speed bipolar (Up and Down)
EXT_SPEED_UP	6	Speed UP
EXT_SPEED_DOWN	7	Speed DOWN
EXT_POS_OPENLOOP	8	Position mode for openloop configurations.

9.3 DoPECycle(Sync)

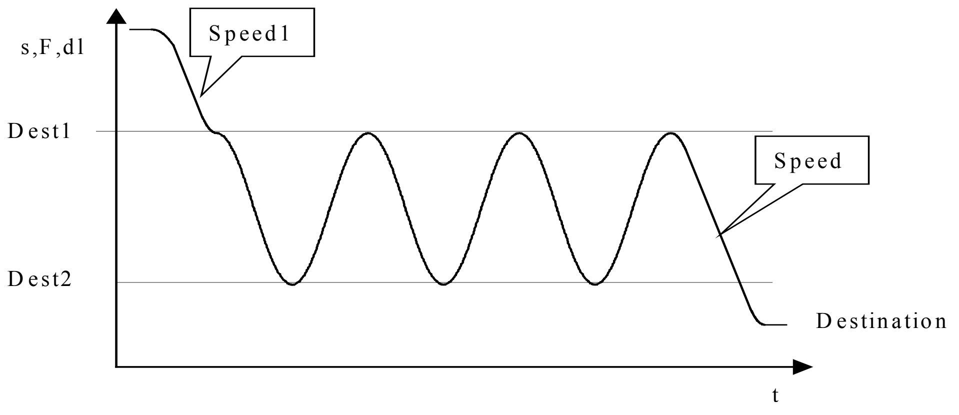


DoPE_HANDLE	DP
unsigned short	MoveCtrl
double	Speed1
double	Dest1
double	Halt1
double	Speed2
double	Dest2
double	Halt2
unsigned long	Cycles
double	Speed
double	Destination
WORD FAR	*lpusTAN

General cycle movement.

In:	DoPEHdl	DoPE link handle
	MoveCtrl	Control mode for movement (CTRL_POS, CTRL_LOAD or CTRL_EXTENSION)
	Speed1	Maximum speed to reach destination 1
	Dest1	Destination 1
	Halt1	Halt time at destination 1
	Speed2	Maximum speed to reach destination 2
	Dest2	Destination 2
	Halt2	Halt time at destination 2
	Cycles	Number of cycles
	Speed	Speed to final destination
	Destination	Final destination
	lpusTAN	Pointer to Transaction-Number
Returns:		Error constant (DoPERR_xxxx)

9.4 DoPECosine(Sync)



DoPE_HANDLE	DP
unsigned short	MoveCtrl
double	Speed1
double	Dest1
double	Dest2
double	Frequency
unsigned long	HalfCycles
double	Speed
double	Destination
WORD FAR	*lpusTAN

Cosine movement.

In:	DoPEHdl	DoPE link handle
	MoveCtrl	Control mode for movement (CTRL_POS, CTRL_LOAD or CTRL_EXTENSION)
	Speed1	Maximum speed to reach destination 1
	Dest1	Destination 1
	Dest2	Destination 2
	Frequency	Frequency
	HalfCycles	Number of half cycles **If Zero, and a Cosine cycle is active, only Frequency, Dest1 and Dest2 are used to modify actve Parameter!**
	Speed	Speed to final destination
	Destination	Final destination
	lpusTAN	Pointer to Transaction-Number
Returns:		Error constant (DoPERR_xxxx)

9.5 DoPECosineX(Sync)

DoPE_HANDLE	DP
unsigned short	MoveCtrl
double	Speed1
double	Dest1
double	Halt1
double	Dest2
double	Halt2
double	Frequency
unsigned long	HalfCycles
double	Speed
double	Destination
WORD FAR	*lpusTAN

Extended version of Cosine movement.
With this command it is possible to halt at destination 1 or 2.

In:	DoPEHdl	DoPE link handle
	MoveCtrl	Control mode for movement (CTRL_POS, CTRL_LOAD or CTRL_EXTENSION)
	Speed1	Maximum speed to reach destination 1
	Dest1	Destination 1
	Halt1	Halt time at Destination 1
	Dest2	Destination 2
	Halt2	Halt time at Destination 2
	Frequency	Frequency
	HalfCycles	Number of half cycles **If Zero, and a Cosine cycle is active, only Frequency, Dest1 and Dest2 are used to modify actve Parameter!**
	Speed	Speed to final destination
	Destination	Final destination
	lpusTAN	Pointer to Transaction-Number
Returns:		Error constant (DoPERR_xxxx)

9.6 DoPECosineV(Sync)

DoPE_HANDLE	DP
unsigned short	Model
double	Dest1
double	Dest2
unsigned short	Cycles
WORD FAR	*lpusTAN

Activate peak value control for currently active Cosine Command. The cosine command value for the control loop will we varied (pilot control) to keep the measured peaks close to the desired (peak value control).

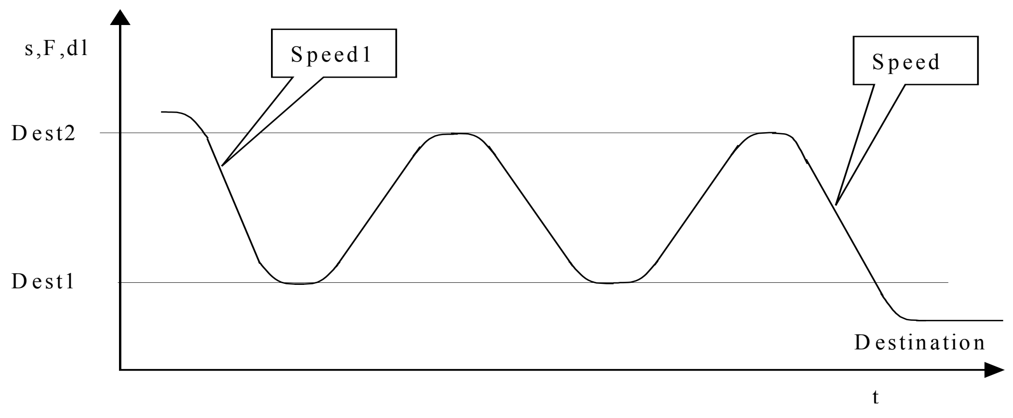
Modes for DoPECosineV:

COS_PCT_AUTO1	Automatic peak value control **with** Start values. Pilot control is activated with the peak values (Destination 1 and 2). After xx cycles peak values will be automatically controlled.
COS_PCT_AUTO2	Automatic peak value control **without** Start offset. Every xx cycles peak values will be automatically controlled.
COS_PCT_MANUAL	Manual pilot control. Destination 1 and 2 will be used for the cosine function generator.
COS_PCT_KEEP	Stop pilot control, keep the found values.
COS_PCT_CONTINUE	Continue pilot control, use the previously values.
COS_PCT_RESET	Reset pilot control, zero values .

In:	DoPEHdl	DoPE link handle
	Mode	Mode for pilot control (see above)
	Dest1	Corrected Destination 1 (only used for COS_PCT_AUTO1 and COS_PCT_MANUAL modes)
	Dest2	Corrected Destination 2 (only used for COS_PCT_AUTO1 and COS_PCT_MANUAL modes)
	Cycles	pilot control is active every xx Cycles
	lpusTAN	Pointer to Transaction-Number

Returns: Error constant (DoPERR_xxxx)

9.7 DoPETriangle(Sync)

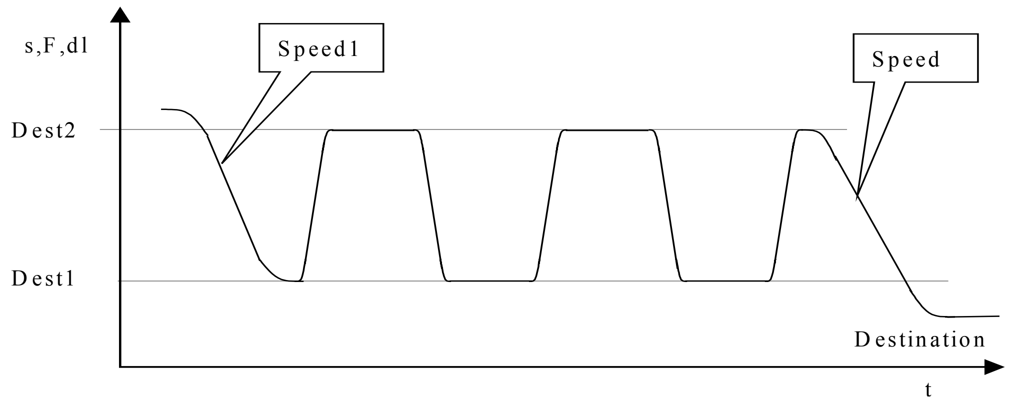


DoPE_HANDLE	DP
unsigned short	MoveCtrl
double	Speed1
double	Dest1
double	Dest2
double	Frequency
unsigned long	HalfCycles
double	Speed
double	Destination
WORD FAR	*lpusTAN

Triangular cyclic movement.

In:	DoPEHdl	DoPE link handle
	MoveCtrl	Control mode for movement (CTRL_POS, CTRL_LOAD or CTRL_EXTENSION)
	Speed1	Maximum speed to reach destination 1
	Dest1	Destination 1
	Dest2	Destination 2
	Frequency	Frequency
	HalfCycles	Number of half cycles Speed
	Speed	to final destination
	Destination	Final destination
	lpusTAN	Pointer to Transaction-Number
Returns:		Error constant (DoPERR_xxxx)

9.8 DoPERectangle(Sync)



DoPE_HANDLE	DP
unsigned short	MoveCtrl
double	Speed1
double	Dest1
double	Dest2
double	Frequency
unsigned long	HalfCycles
double	Speed
double	Destination
WORD FAR	*lpusTAN

Rectangular cyclic movement.

In:	DoPEHdl	DoPE link handle
	MoveCtrl	Control mode for movement (CTRL_POS, CTRL_LOAD or CTRL_EXTENSION)
	Speed1	Maximum speed to reach destination 1
	Dest1	Destination 1
	Dest2	Destination 2
	Frequency	Frequency
	HalfCycles	Number of half cycles Speed
	Speed	to final destination
	Destination	Final destination
	lpusTAN	Pointer to Transaction-Number
Returns:		Error constant (DoPERR_xxxx)

9.9 DoPEOffsC(Sync)

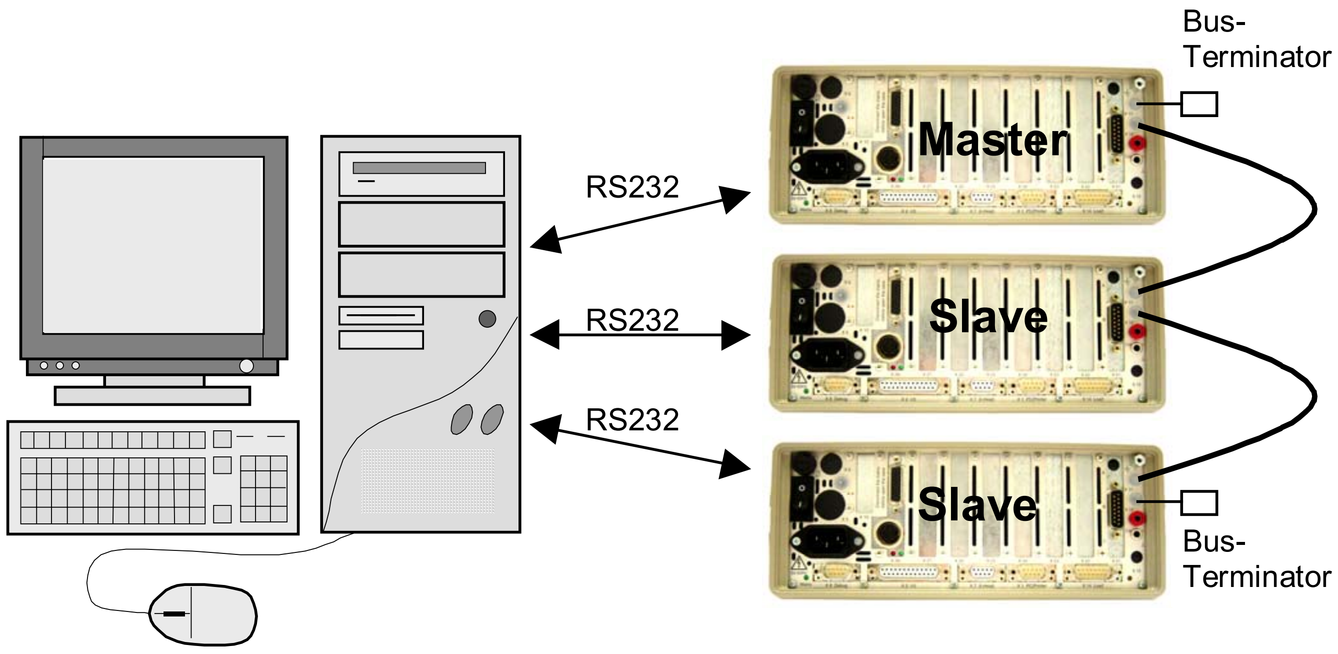
DoPE_HANDLE	DP
double	Speed
double	PosDiff
WORD FAR	*lpusTAN

Special moving command to measure the offset of an external, analogue speed controller. This offset will be used for the speed output signal and compensates the offset of the external speed controller.

In:	DoPEHdl	DoPE link handle
	Speed	Maximum speed
	PosDiff	Distance to move cross-head
	lpusTAN	Pointer to Transaction-Number
Returns:		Error constant (DoPERR_xxxx)

10 Synchronized movement

There are two commands to control synchronized movement of two or more EDC120 systems.
This commands are only useful if the synchronization option is installed, and the two or more EDC120 are correctly connected.
With the first command, you can synchronize time and movement, with the second command you can start the synchronization.
ATTENTION: This function is only supported in only EDC60/120- systems



10.1 DoPESynchronizeSystemMode (Sync)

DoPE_HANDLE	DoPEHdl
unsigned short	Mode
double	Time

WORD FAR **lpusTAN* *(not for Sync. version)*

Set or discard the delay time for synchronized movement commands, or the system time for synchronized systems. **(only EDC60/120)**
If Mode is SSM_SYNCMOVE, the next moving command like DoPEPos, DoPECosine or any other moving command will be delayed until synchronization condition is true. The systems will be synchronized by the DoPESynchronizeSystemStart command.

In:	DoPEHdl	DoPE link handle	
	Mode	SSM_SYNCMOVE:	If Time is ZERO, start movement after synchonize signal is active. If Time is not ZERO, wait Time after synchonize signal is active before starting the movement.
		SSM_SYSTEMTIME:	Set EDC-system time to Time, after synchonize signal is active.
		SSM_DISCARD:	Discard previous DoPESynchronizeSystemMode commands.
	Time	Delay or system time to set with the next DoPESynchronizeSystemStart command.	
	lpusTAN	Pointer to Transaction-Number	
Returns:		Error constant (DoPERR_xxxx)	

Example1: Two axis should do synchronized sinusoidal cycles with no phase shift.
EDC120 with DoPEHdl_1 is the master EDC.

```
/* Attention: The return code in this example is not checked!
   A real program must check the return code !!                   */

/* Set system time for both EDC to zero after synchronization started */
Error = DoPESynchronizeSystemMode (DoPEHdl_1, SSM_SYSTEMTIME, 0, &lpusTAN_1);
Error = DoPESynchronizeSystemMode (DoPEHdl_2, SSM_SYSTEMTIME, 0, &lpusTAN_2);

/* Start cosine after synchronization without delay */
Error = DoPESynchronizeSystemMode (DoPEHdl_1, SSM_SYNCMOVE, 0, &lpusTAN_1);
Error = DoPECosine  ( DoPEHdl_1, CTRL_POS, Speed1, Dest1, Dest2, Frequency, HalfCycles, Speed,
                      Destination, &lpusTAN_1);

/* Start cosine after synchronization without delay */
Error = DoPESynchronizeSystemMode (DoPEHdl_2, SSM_SYNCMOVE, 0, &lpusTAN_2);
Error = DoPECosine  ( DoPEHdl_2, CTRL_POS, Speed1, Dest1, Dest2, Frequency, HalfCycles, Speed,
                      Destination, &lpusTAN_2);

/* Start synchronization at Master EDC120 */
Error = DoPESynchronizeSystemStart (DoPEHdl_1, &lpusTAN_1);
```

Example2: Two axis should do synchronized sinusoidal cycles with 90° phase shift.
EDC120 with DoPEHdl_1 is the master EDC.

```
/* Attention: The return code in this example is not checked!
   A real program must check the return code !!                   */

/* Set system time for both EDC to zero after synchronization started */
Error = DoPESynchronizeSystemMode (DoPEHdl_1, SSM_SYSTEMTIME, 0, &lpusTAN_1);
Error = DoPESynchronizeSystemMode (DoPEHdl_2, SSM_SYSTEMTIME, 0, &lpusTAN_2);

/* Start cosine after synchronization without delay */
Error = DoPESynchronizeSystemMode (DoPEHdl_1, SSM_SYNCMOVE, 0, &lpusTAN_1);
Error = DoPECosine  ( DoPEHdl_1, CTRL_POS, Speed1, Dest1, Dest2, Frequency, HalfCycles, Speed,
                      Destination, &lpusTAN_1);

/* Start cosine after synchronization 90° phase shift */
Error = DoPESynchronizeSystemMode (DoPEHdl_2, SSM_SYNCMOVE, 1 / (Frequency * 4), &lpusTAN_2);
Error = DoPECosine  ( DoPEHdl_2, CTRL_POS, Speed1, Dest1, Dest2, Frequency, HalfCycles, Speed,
                      Destination, &lpusTAN_2);

/* Start synchronization at Master EDC120 */
Error = DoPESynchronizeSystemStart (DoPEHdl_1, &lpusTAN_1);
```

10.2 DoPESynchronizeSystemStart (Sync)

DoPE_HANDLE	DoPEHdl	
WORD FAR	**lpusTAN*	*(not for Sync. version)*

This function can only be processed on the Master-EDC120.
The digital synchronization signal will be activated, and with the next system clock, all involved EDC120 will start the previously defined actions, set by DoPESynchronizeSystemMode.

In:	DoPEHdl	DoPE link handle
	lpusTAN	Pointer to Transaction-Number
Returns:		Error constant (DoPERR_xxxx)

11 Miscellaneous Control Commands

11.1 DoPEOn(Sync)

DoPE_HANDLE	DP	
WORD FAR	**lpusTAN*	*(not for Sync. version)*

Activate drive (only for EDC5/25 and EDC60/120). On EDC100 systems the drive is activated by the "ON" push button at the EDC100.

In:	DoPEHdl	DoPE link handle
	lpusTAN	Pointer to Transaction-Number
Returns:		Error constant (DoPERR_xxxx)

11.2 DoPEDefaultAcc(Sync)

DoPE_HANDLE	DP	
unsigned short	MoveCtrl	
double	Acc	
WORD FAR	**lpusTAN*	*(not for Sync. version)*

Set default acceleration (and deceleration) for all moving commands. After initialisation default and nominal acceleration are identical.

In:	DoPEHdl	DoPE link handle
	MoveCtrl	Control mode (Position, Load, Extension)
	Acc	Acceleration
	lpusTAN	Pointer to Transaction-Number
Returns:		Error constant (DoPERR_xxxx)

11.3 DoPESpeedLimit(Sync)

DoPE_HANDLE	DP	
unsigned short	MoveCtrl	
double	Speed	
WORD FAR	**lpusTAN*	*(not for Sync. version)*

Set speed limit.

In:	DoPEHdl	DoPE link handle
	MoveCtrl	Control mode (Position, Load, Extension)
	Speed	Maximum allowed speed (must be below nominal speed)
	lpusTAN	Pointer to Transaction-Number
Returns:		Error constant (DoPERR_xxxx)

11.4 DoPEStop(Sync)

DoPE_HANDLE	DP
unsigned short	State
WORD FAR	*lpusTAN

Activate / deactivate stop state.

In:	DoPEHdl	DoPE link handle
	Status	TRUE: stop state (reset drive enable) FALSE: free state (set drive enable)
	lpusTAN	Pointer to Transaction-Number
Returns:		Error constant (DoPERR_xxxx)

11.5 DoPEEmergencyMove(Sync)

DoPE_HANDLE	DP	
unsigned short	State	
WORD FAR	**lpusTAN*	*(not for Sync. version)*

Activate / deactivate emergency movement. Emergency movement is used to move cross-head if the hardware limit switch is active. (Not supported on EDC5/25)

In:	DoPEHdl	DoPE link handle
	State	TRUE: on, FALSE: off
	lpusTAN	Pointer to Transaction-Number
Returns:		Error constant (DoPERR_xxxx)

11.6 DoPEEmergencyOff(Sync)

DoPE_HANDLE	DP	
unsigned short	Status	
WORD FAR	**lpusTAN*	*(not for Sync. version)*

Activate / deactivate EmergencyOff state.

In:	DoPEHdl	DoPE link handle
	Status	TRUE: Activate emergency off FALSE: Deactivate emergency off
	lpusTAN	Pointer to Transaction-Number
Returns:		Error constant (DoPERR_xxxx)

11.7 DoPESetOpenLoopCommand(Sync)

DoPE_HANDLE	DP	
double	Command	
WORD FAR	**lpusTAN*	*(not for Sync. version)*

Set output value to valve. This command is only allowed in OpenLoop controller structure. It is used for hydraulic machines, that operate without a position sensor. (typically concrete testing machines) With this command it is possible to move the piston up or down. The command value where the piston is floating (not moving up or down) must be determined during installation.

In:	DoPEHdl	DoPE link handle
	Command	Output value to valve in %.
	lpusTAN	Pointer to Transaction-Number
Returns:		Error constant (DoPERR_xxxx)

12 Sensor EEPROM Handling

Following functions support read and write of sensor-EEPROM data.

ATTENTION:	**Use this functions very carefully.**
	DoPE cannot check the total integrity of the data!
	UserScale has no effect on the Sensor EEPROM data!

These function work on initialised and not initialised sensors. The sensors are selected by the connector number.

The data inside the sensor EEPROM are divided into two sections.
The first section (DoPESensorHeaderData), is identical for all sensors.
The second section depends of the sensor class.
If you want to change data inside the sensor EEPROM, please follow the procedure described below:

1. Use DoPERdSensorConKey function and check if a sensor is connected.
2. Use DoPERdSensorHeaderData function to read header data.
   Analyse the sensor class!
3. For DoPESensorHeaderData.Class =SEN_ANALOGUE use DoPERdSensorAnalogueData,
   for DoPESensorHeaderData.Class =SEN_INC use DoPERdSensorIncData,
   for DoPESensorHeaderData.Class =SEN_ABS use DoPERdSensorAbsData
   to read the sensor class specific data.
4. Modify data.
   Only the **bold type** parameter in the sensor data structures are allowed to be changed!
5. Use DoPERdSensorConKey in a loop (not faster than 100 ms) and wait until the KeyPressed parameter is 1.
6. Use the appropriate write function to store the data into the sensor EEPROM.

It is not absolutely necessary to use the function DoPERdSensorConKey before writing data into the sensor EEPROM.
If you don't use it, be aware that calibration data may be changed, without breaking the seal at the sensor plug!

12.1 DoPERdSensorConKey

```
DoPE_HANDLE        DoPEHdl
WORD               Connector
WORD               *Connected
WORD               *KeyPressed
```

Read sensor plug connected and key state.

In:	DoPEHdl	DoPE link handle
	Connector	Connector number of sensor
	*Connected	Pointer to the sensor plug connected state (0=not connected, 1=connected)
	*KeyPressed	Pointer to the sensor plug key state (0=not pressed, 1=pressed)
Returns:		Error constant (DoPERR_xxxx)

12.2 DoPERdSensorHeaderData

DoPE_HANDLE	DoPEHdl
WORD	Connector
DoPESensorHeaderData	*SenHdrData

Read sensor EEPROM data header.

In:	DoPEHdl	DoPE link handle
	Connector	Connector number of sensor
	*SenHdrData	Pointer to sensor data header structure

Returns:	Error constant (DoPERR_xxxx)

DoPESensorHeaderData structure:

```
typedef struct                          /* Sensor EEPROM header data         */
{                                       /* --------------------------------- */
  WORD   PartNo;                        /* Part indent number            [No]*/
  BYTE   Version;                       /* Part revision                 [No]*/
  DWORD  SerNo;                         /* Part serial number            [No]*/
  WORD   Class;                         /* Sensor class                  [No]*/
  BYTE   DatVersion;                    /* Version of data               [No]*/
} DoPESensorHeaderData;

/* Sensor classes                      */
                                        /* --------------------------------- */
#define SEN_UNDEF           0           /* unknown sensor class              */
#define SEN_ANALOGUE        1           /* analogue sensor                   */
#define SEN_INC             2           /* incremental sensor                */
#define SEN_ABS             3           /* absolute value sensor             */

                                        /* Analogue sensor types             */
                                        /* --------------------------------- */
#define SIG_STRAINGAUGE     0           /* Strain gauge                      */
#define SIG_LVDT            1           /* LVDT                              */
#define SIG_DC              2           /* DC                                */

                                        /* Incremental sensor types          */
                                        /* --------------------------------- */
#define SIG_TTL             0           /* TTL Signal                        */
#define SIG_LINE            1           /* RS422 (line driver)               */
#define SIG_SINE11uA        2           /* Sine 11µA                         */
#define SIG_SINE1V          3           /* Sine 1V                           */

                                        /* Absolute value sensor types       */
                                        /* --------------------------------- */
#define SIG_UNDEF           0           /* undefined                         */
#define SIG_TR_LT_S         1           /* TR LT-S Sensor                    */

                                        /* Transducer types                  */
                                        /* --------------------------------- */
#define TRANSDUCER_LINEAR  0            /* Linear transducer                 */
#define TRANSDUCER_ROTARY  1            /* Rotary transducer                 */

                                        /* Reference mark types              */
                                        /* --------------------------------- */
#define REFMARK_NON        0            /* Transducer has no reference mark  */
#define REFMARK_ONE        1            /* Transducer has one reference mark */
#define REFMARK_DISTCODE   2            /* Transducer has distance coded     */
```

12.3 DoPERdSensorAnalogueData

DoPE_HANDLE	DoPEHdl
WORD	Connector
DoPESensorAnalogueData	*SenAnalogueData

Read analogue sensor data.

In:	DoPEHdl	DoPE link handle
	Connector	Connector number of sensor
	*SenAnalogueData	Pointer to analogue sensor data structure
Returns:		Error constant (DoPERR_xxxx)

12.4 DoPEWrSensorAnalogueData

DoPE_HANDLE	DoPEHdl
WORD	Connector
DoPESensorAnalogueData	*SenAnalogueData

Write analogue sensor data.

In:	DoPEHdl	DoPE link handle
	Connector	Connector number of sensor
	*SenAnalogueData	Pointer to analogue sensor data structure
Returns:		Error constant (DoPERR_xxxx)

DoPESensorAnalogueData structure:

```
typedef struct                         /* Analogue sensor EEPROM data        */
{                                      /* ---------------------------------- */
  float  MaxExcitation;                /* Maximum excitation voltage      [V]*/
  WORD   MinImpedance;                 /* Impedance                     [Ohm]*/
  float  NominalValue;                 /* Nominal value of the sensor  [Unit]*/
  WORD   Unit;                         /* Unit of sensor UNIT_xxx        [No]*/
  float  Offset;                       /* Sensor offset                [Unit]*/
  WORD   NegLimit;                     /* Range limit - min.              [%]*/
  WORD   PosLimit;                     /* Range limit - max.              [%]*/
  float  Reference;                    /* Nominal value of the reference  [*]*/
  double CorrReference;                /* Corr. value of the reference   [No]*/
  WORD   Sensortype;                   /* Sensor type                        */
  double NominalSensitive;             /* Sensitivity at Nominal value    [*]*/
  WORD   Sign;                         /* Invert sign of channel        [1/0]*/
  int    Day;                          /* Date of last change            [No]*/
  int    Month;                        /* Date of last change            [No]*/
  int    Year;                         /* Date of last change            [No]*/
  WORD   LinPoint;                     /* Number of linearization steps  [No]*/
  struct LinVal
  {
    double MeasValue;                  /* Measured value               [Unit]*/
    double RefValue;                   /* Reference                    [Unit]*/
  } LinV[SEN_LIN_DATA_MAX];            /* Linearization table                */
} DoPESensorAnalogueData;
```

12.5 DoPERdSensorIncData

DoPE_HANDLE	DoPEHdl
WORD	Connector
DoPESensorIncData	* SenIncData

Read incremental sensor data.

In:	DoPEHdl	DoPE link handle
	Connector	Connector number of sensor
	*SenIncData	Pointer to incremental sensor data structure
Returns:		Error constant (DoPERR_xxxx)

12.6 DoPEWrSensorIncData

DoPE_HANDLE	DoPEHdl
WORD	Connector
DoPESensorIncData	*SenIncData

Write incremental sensor data.

In:	DoPEHdl	DoPE link handle
	Connector	Connector number of sensor
	*SenIncData	Pointer to incremental sensor data structure
Returns:		Error constant (DoPERR_xxxx)

DoPESensorIncData structure:

```
typedef struct                       /* Incremental sensor EEPROM data     */
{                                    /* ---------------------------------- */
  float  Voltage1;                   /* Supply voltage 1               [V]*/
  float  Voltage2;                   /* Supply voltage 2               [V]*/
  float  Voltage3;                   /* Supply voltage 3               [V]*/
  float  Current1;                   /* Current for supply voltage 1   [A]*/
  float  Current2;                   /* Current for supply voltage 2   [A]*/
  float  Current3;                   /* Current for supply voltage 3   [A]*/
  WORD   InputSignal;                /* Signal type at input  SIG_xxx  [No]*/
  WORD   OutputSignal;               /* Signal type at output SIG_xxx  [No]*/
  WORD   InterpolationFactor;        /* Factor for interpolation       [No]*/
  float  MaxInputFreq;               /* Maximum input frequency        [Hz]*/
  float  MaxOutputFreq;              /* Maximum output frequency       [Hz]*/
  WORD   TransducerType;             /* Trancducer type TRANSDUCER_xxx [No]*/
  WORD   Unit;                       /* Unit of sensor UNIT_xxx        [No]*/
  double SignalPeriod;               /* Signal period                [Unit]*/
  double CorrFactor;                 /* Correction factor              [No]*/
  double MeasuringRange;             /* Measuring range              [Unit]*/
  WORD   SignalType;                 /* Tancducer signal type  SIG_xxx  [No]*/
  WORD   ReferenceMark;              /* Reference mark type REFMARK_xxx[No]*/
  double FirstDistance;              /* First distance of the reference[Unit]*/
  double NominalDistance;            /* Nominal distance of the reference[Unit]*/
  double Delta;                      /* Dislocation of the mean reference[Unit]*/
  float  LimitFrequency;             /* Limit frequency of the trancducer[Hz]*/
  WORD   Sign;                       /* Invert sign of channel        [1/0]*/
  BYTE   NegLimit;                   /* Range limit - min.              [%]*/
  BYTE   PosLimit;                   /* Range limit - max.              [%]*/
} DoPESensorIncData;
```

12.7 DoPERdSensorAbsData

DoPE_HANDLE	DoPEHdl
WORD	Connector
DoPESensorAbsData	*SenAbsData

Read absolute sensor data.

In:	DoPEHdl	DoPE link handle
	Connector	Connector number of sensor
	*SenAbsData	Pointer to absolute value sensor data structure
Returns:		Error constant (DoPERR_xxxx)

12.8 DoPEWrSensorAbsData

DoPE_HANDLE	DoPEHdl
WORD	Connector
DoPESensorAbsData	*SenAbsData

Write absolute sensor data.

In:	DoPEHdl	DoPE link handle
	Connector	Connector number of sensor
	*SenAbsData	Pointer to absolute value sensor data structure
Returns:		Error constant (DoPERR_xxxx)

DoPESensorAbsData structure:

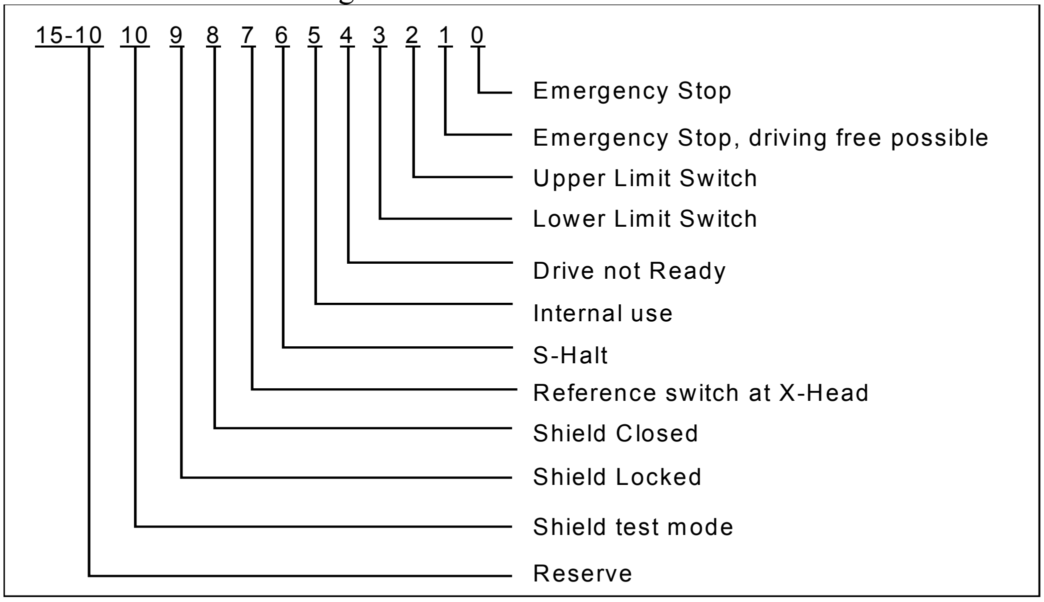
```
typedef struct                       /* Absolute value EEPROM header data  */
{                                    /* ---------------------------------- */
  float  Voltage1;                   /* Supply voltage 1                [V]*/
  float  Voltage2;                   /* Supply voltage 2                [V]*/
  float  Voltage3;                   /* Supply voltage 3                [V]*/
  float  Current1;                   /* Current for supply voltage 1    [A]*/
  float  Current2;                   /* Current for supply voltage 2    [A]*/
  float  Current3;                   /* Current for supply voltage 3    [A]*/
  WORD   InputSignal;                /* Signal type at input  SIG_xxx  [No]*/
  WORD   OutputSignal;               /* Signal type at output SIG_xxx  [No]*/
  float  MaxInputFreq;               /* Maximum input frequency        [Hz]*/
  float  MaxOutputFreq;              /* Maximum output frequency       [Hz]*/
  BYTE   DelayTime;                  /* Sensors signal delay time       [s]*/
  WORD   Unit;                       /* Unit of sensor UNIT_xxx        [No]*/
  double SignalPeriod;               /* Signal period                [Unit]*/
  float  Offset;                     /* Sensor offset                [Unit]*/
  double CorrFactor;                 /* Correction factor              [No]*/
  double NominalValue;               /* Nominal value of the sensor  [Unit]*/
  WORD   SignalType;                 /* Tancducer sgnal type  SIG_xxx  [No]*/
  float  LimitFrequency;             /* Limit frequency of the trancducer[Hz]*/
  WORD   Sign;                       /* Invert sign of channel        [1/0]*/
  BYTE   NegLimit;                   /* Range limit - min.              [%]*/
  BYTE   PosLimit;                   /* Range limit - max.              [%]*/
  int    Day;                        /* Date of last change            [No]*/
  int    Month;                      /* Date of last change            [No]*/
  int    Year;                       /* Date of last change            [No]*/
} DoPESensorAbsData;
```

13 Default measuring dada record

unsigned long	Cycles	Cycle counter
double	Time	Time from subsystem
double	Position	X-Head position
double	Load	Load
double	Extension	Extension
double	SensorD	DigiPoti
double	Sensor4	Sensor 4 measuring channel
double	Sensor5	Sensor 5 measuring channel
double	Sensor6	Sensor 6 measuring channel
double	Sensor7	Sensor 7 measuring channel
double	Sensor8	Sensor 8 measuring channel
double	Sensor9	Sensor 9 measuring channel
double	Sensor10	Sensor 10 measuring channel
double	Sensor11	Sensor 11 measuring channel
double	Sensor12	Sensor 12 measuring channel
double	Sensor13	Sensor 13 measuring channel
double	Sensor14	Sensor 14 measuring channel
double	Sensor15	Sensor 15 measuring channel
unsigned short	BitIn0	Digital input device 0
unsigned short	BitIn1	Digital input device 1
unsigned short	BitIn2	Digital input device 2
unsigned short	BitIn3	Digital input device 3
unsigned short	BitIn4	Digital input device 4
unsigned short	BitIn5	Digital input device 5
unsigned short	BitIn6	Digital input device 6
unsigned short	BitIn7	Digital input device 7
unsigned short	BitIn8	Digital input device 8
unsigned short	BitIn9	Digital input device 9
unsigned short	BitOut0	Digital output device 0
unsigned short	BitOut1	Digital output device 1
unsigned short	BitOut2	Digital output device 2
unsigned short	BitOut3	Digital output device 3
unsigned short	BitOut4	Digital output device 4
unsigned short	BitOut5	Digital output device 5
unsigned short	BitOut6	Digital output device 6
unsigned short	BitOut7	Digital output device 7
unsigned short	BitOut8	Digital output device 8
unsigned short	BitOut9	Digital output device 9
unsigned short	InSignals	Logical input signals definition see 13.1
unsigned short	OutSignals	Logical output signals definition see 13.2
unsigned short	CtrlState1	Controller status WORD 1 definition see 13.3
unsigned short	CtrlState2	Controller status WORD 2 definition see 13.4
unsigned short	UpperLimits	Upper limits exceeded
unsigned short	LowerLimits	Lower limits exceeded
unsigned short	SysState0	System status WORD 0
unsigned short	SysState1	System status WORD 1
unsigned short	SysState2	System status WORD 2
unsigned short	SysState3	System status WORD 3
unsigned short	SysState4	System status WORD 4
unsigned short	SysState5	System status WORD 5
double	Test1	Configured test value 1
double	Test2	Configured test value 2
double	Test3	Configured test value 3
unsigned short	Keys	Actual State of EDC frontpanel keys
unsigned short	NewKeys	New keys
unsigned short	GoneKeys	Gone keys

13.1 Logical input signals

The logical input signals are safety relevant digital inputs. They are configured during initialisation and one logical input signal may map to more than one digital input. The state of the logical input signals is maintained in the measuring dada record.



Emergency stop (IN_SIG_EMERGENCY_STOP):
If this signal is active, the machine control remains in the emergency stop state. Possible sources are Limit switches, Range limit active, emergency stop button pressed or similar safety equipment.

Emergency stop, driving free possible (IN_SIG_EMERGENCY_STOP_FREE_POSSIBILE):
This signal also forces the machine control into the emergency stop state, but can be masked during an driving free movement.

Limit switches, (IN_SIG_UPPER_LIMIT_SWITCH IN_SIG_LOWER_LIMIT_SWITCH):
These signals also force the machine control into the emergency stop state and can be masked during an driving free movement.

Drive not ready (IN_SIG_DRIVE_NOT_READY):
This signal reports the state of the connected drive unit. If set, the drive is not ready.

S-Halt (IN_SIG_SHALT):
The signal is especially suitable for the connection of a HALT key, the cross-head can be halted with independently of the actual operating state.

Drive not ready (IN_SIG_REFERENCE):
This signal reports the state a reference switch at the X-Head.

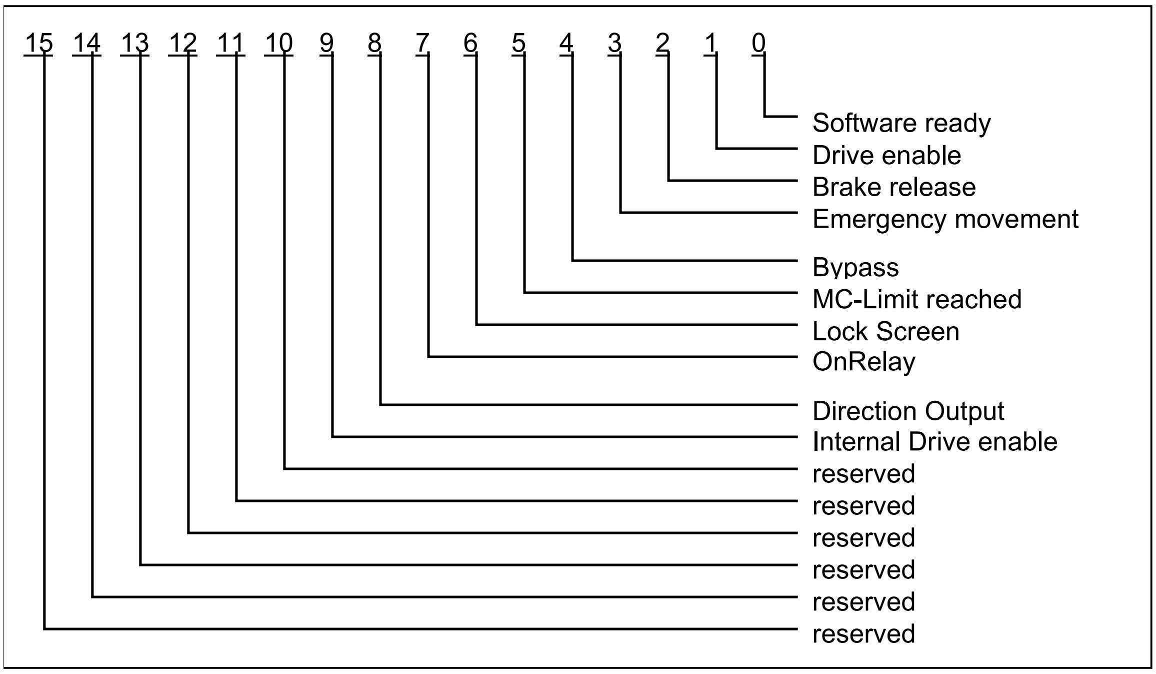
Drive not ready (IN_SIG_SHIELD_CLOSED):
This signal is set when a connected protective shield is closed.

Drive not ready (IN_SIG_SHIELD_LOCKED):
This signal is set when a connected protective shield is locked.

Drive not ready (IN_SIG_SHIELD_TEST):
This signal is set when the connected protective shield is in test mode. All protective shield functions are disabled.

13.2 Logical output signals

The logical output signals are digital outputs to control the drive and associated devices. They are configured during initialisation. The state of the logical output signals is maintained in the measuring dada record.



Drive ON (O_SIG_DRIVE_ON):

Being in the **inactive** state, this output signal should force the drive into its defined emergency stop state. If the signal is **active**, the drive should change after an arbitrary period of Time into the 'Operational' state. 'Operational' means, that the drive can react upon the output signal 'Drive Enable' (see below) without delay.

Drive enable (O_SIG_DRIVE_ENABLE):

The signal 'Drive enable' locks in the inactive state, or enables in the active state, the reaction of the drive to changes in the control variable. The change of state in the drive must be without delay, that is, it must be executed within 2.5 ms.

Brake release (O_SIG_BRAKE):

The signal 'Brake Release' is linked with the signal 'Drive Free' because of their fault levels, but will be operated with a time shift due to their corresponding change of states. The time shift should enable systems with mechanical brakes to achieve a controlled transition between braking and position control.

Emergency movement (O_SIG_E_MOVE):

The signal 'Emergency Movement' is only active, as long as the Subsystem executes an emergency movement. It can be used to deactivate the function of the machine limit switches on the drive system.

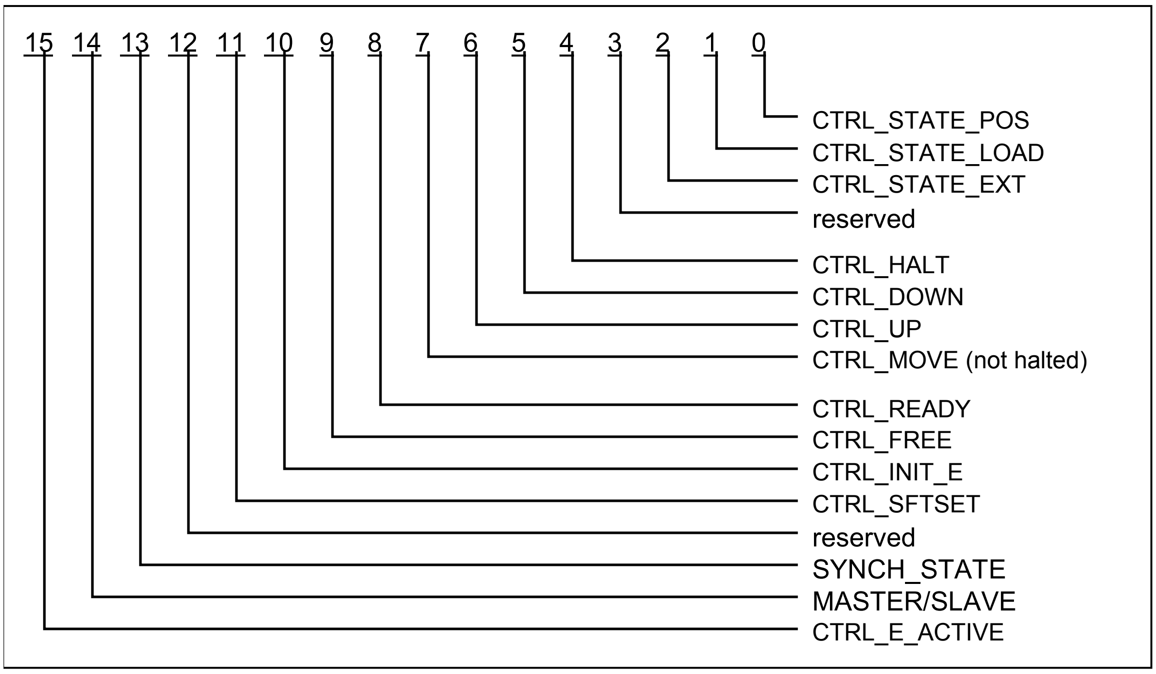
MC-Limit reached (O_SIG_LIMIT):

The signal ‘MC-Limit’ is used by the measuring channel supervision activated by DoPESetCheckLimit command.

Lock Screen (O_SIG_SH_LOCK):

The signal ‘Lock Screen’ is used by the screen (shield) supervision functions to lock the shield.

13.3 Controller status WORD 1



CTRL_STATE_POS:
X-Head closed loop control is active.

CTRL_STATE_LOAD:
Load closed loop control is active.

CTRL_STATE_EXT:
Extension closed loop control is active.

CTRL_HALT:
Command value generator halts (controlled halt in S/F/E). If the set speed 0 is temporary unstable, as for example by moving with the external command value of 0, then the status bits CTRL_DOWN and CTRL_UP are both set instead of this bit.

CTRL_DOWN:
Command value generator moves down (in S/ F/ E).

CTRL_UP:
Command value generator moves up (in S/F/E).

CTRL_MOVE:
This bit is always set when the cross-head is not securely halted. The bit is not set with a switched off machine and with S controlled holding of a position. In all other cases the bit is set.

CTRL_READY:
The bit is set, when the EDC accepts movement instructions from the DoPE. It is also particularly set, when the EDC is in the drive ready state, waiting for the start of the drive through the first movement instruction.

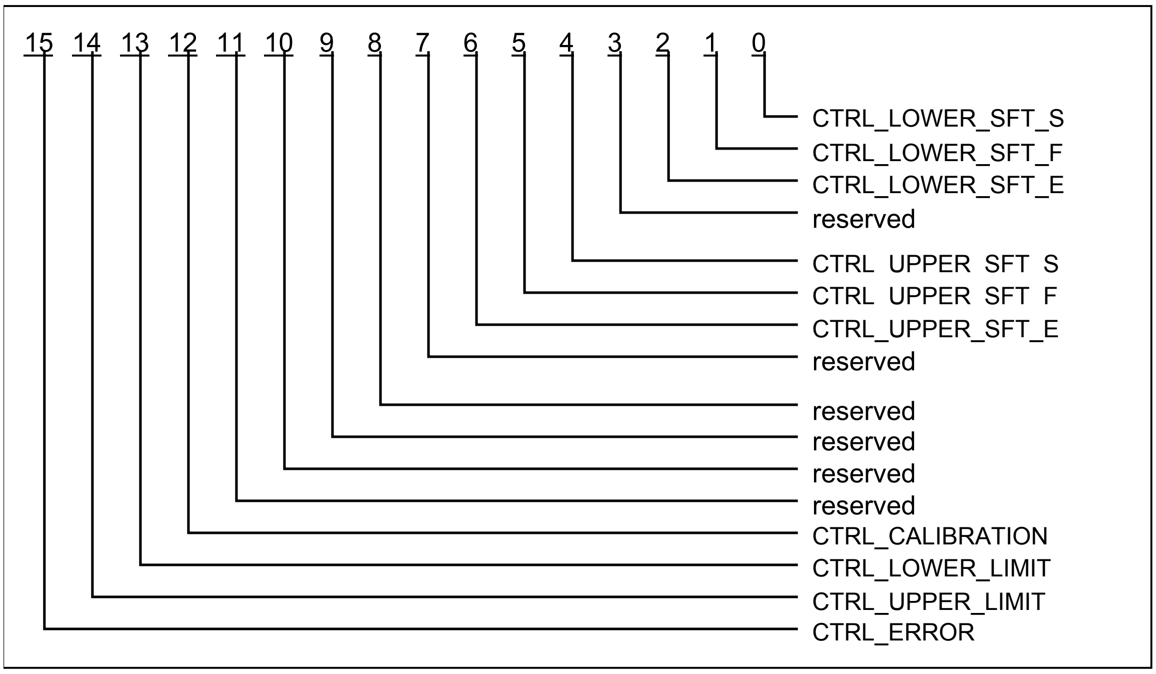
CTRL_FREE:
The EDC waits for the drive enable through the Host software or from the user.

CTRL_INIT_E:
The EDC awaits the initiation of an emergency movement through the Host.

CTRL_SFTSET:
Change of softends allowed.

CTRL_E_ACTIVE:
Emergency movement is active.

13.4 Controller status WORD 2



CTRL_LOWER_SFT_S:
Lower X-Head position softend is active.

CTRL_LOWER_SFT_F:
Lower load position softend is active.

CTRL_LOWER_SFT_E:
Lower extension position softend is active.

CTRL_UPPER_SFT_S:
Upper X-Head position softend is active.

CTRL_UPPER_SFT_F:
Upper load position softend is active.

CTRL_UPPER_SFT_E:
Upper extension position softend is active.

CTRL_LOWER_LIMIT, CTRL_UPPER_LIMIT:
A lower or upper range limit in X-head position, load or extension is active.

CTRL_CALIBRATION:
Calibration of analogue channels is currently active.

13.5 Definitions of EDC-front panel keys

The EDC-front panel keys are coded and transmitted in DoPE data record within three WORD's.
One WORD represents the active state of the keys, one WORD all new keys and one WORD all gone keys. The "0"-"9", "F1" - "F3", "." and "±" keys are coded in the low BYTE in ASCII.
The "Up", "Down", "Halt" and "DigiPoti" keys are represented as single bits in the upper BYTE.

```
#define PE_KEY_HALT           0x0100    /* Bit mask for Key 'HALT'          */
#define PE_KEY_UP             0x0200    /* Bit mask for Key 'UP'            */
#define PE_KEY_DOWN           0x0400    /* Bit mask for Key 'DOWN'          */
#define PE_KEY_DPOTI          0x0800    /* Bit mask for Key 'DigiPoti'      */

#define PE_KEY_0              0x0030    /* Code for Key '0'                 */
#define PE_KEY_1              0x0031    /* Code for Key '1'                 */
#define PE_KEY_2              0x0032    /* Code for Key '2'                 */
#define PE_KEY_3              0x0033    /* Code for Key '3'                 */
#define PE_KEY_4              0x0034    /* Code for Key '4'                 */
#define PE_KEY_5              0x0035    /* Code for Key '5'                 */
#define PE_KEY_6              0x0036    /* Code for Key '6'                 */
#define PE_KEY_7              0x0037    /* Code for Key '7'                 */
#define PE_KEY_8              0x0038    /* Code for Key '8'                 */
#define PE_KEY_9              0x0039    /* Code for Key '9'                 */
#define PE_KEY_DP             0x002E    /* Code for Key '.'                 */
#define PE_KEY_SIGN           0x00F1    /* Code for Key '±'                 */
#define PE_KEY_F1             0x0041    /* Code for Key 'F1'                */
#define PE_KEY_F2             0x0042    /* Code for Key 'F2'                */
#define PE_KEY_F3             0x0043    /* Code for Key 'F3'                */
```

14 DoPE Error constants

DoPERR_NOERROR	(0)	No error
DoPERR_NOFLOAT	(1)	No float in WIN16 callback
DoPERR_SYNC	(2)	Synchronization to callback failed
DoPERR_TIMEOUT	(3)	Timeout at await answer
DoPERR_NOFNC	(4)	Function not implemented
DoPERR_VERSION	(5)	No compatible Version EDC-DoPE
DoPERR_INIT	(6)	Initialisation Error Subsystem
DoPERR_PARAMETER	(7)	Invalid parameter
DoPERR_SETUPOPEN	(8)	Set-up open error
DoPERR_RTE_UNHANDLED	(9)	Unhandled runtime error
Command errors		
DoPERR_CMD_PARCORR	(1001)	rror in parameter (corrected)
DoPERR_CMD_PAR	(1003)	Error in parm. not correctable
DoPERR_CMD_XMOVE	(1004)	X-Head is not halted
DoPERR_CMD_INITSEQ	(1005)	Sequence in init. not observed
DoPERR_CMD_NOTINIT	(1006)	Controller part not initialised
DoPERR_CMD_DIR	(1007)	Movement direction not possible
DoPERR_CMD_TMP	(1008)	Required resource not available
DoPERR_CMD_RUNTIME	(1009)	Run time error active
DoPERR_CMD_INTERN	(1010)	Internal error in subsystem
DoPERR_CMD_MEM	(1011)	Insufficient memory
DoPERR_CMD_CST	(1012)	Wrong controller Structure
DoPERR_CMD_MSGNO	(2001)	Unknown message number
DoPERR_CMD_VERSION	(2003)	Wrong PE interface version
DoPERR_CMD_OPEN	(2004)	Set-up not opened
DoPERR_CMD_MEMORY	(2005)	Not enough memory

Machine normalisation errors		
DoPERR_PARMS	(0x4001)	Parameter Error
DoPERR_ZERODIV	(0x4002)	Division by ZERO
DoPERR_OVFLOW	(0x4003)	Overflow
DoPERR_NIN	(0x4004)	Not Initialised
Low level communication errors		
DoPERR_NODATA	(0x8001)	No receiver data available
DoPERR_NOBUFFER	(0x8002)	No transmitter buffer available
DoPERR_OFFLINE	(0x8003)	Connection is offline
DoPERR_HANDLE	(0x8004)	Invalid DoPE handle
DoPERR_MSGSIZE	(0x8005)	Message to long
DoPERR_NOMEM	(0x8007)	Not enough heap memory
DoPERR_BADPORT	(0x8008)	Invalid device ID
DoPERR_BAUDRATE	(0x8009)	Invalid baudrate
DoPERR_OPEN	(0x800A)	Device already in use
DoPERR_HARDWARE	(0x800B)	Device not present
DoPERR_NOTOPEN	(0x800C)	Connection not open
DoPERR_PORTLIMIT	(0x800D	Unused
DoPERR_NOTIMER	(0x800E)	No timer for timeout check
DoPERR_NODRIVER	(0x800F)	No driver available
DoPERR_NOTHREAD	(0x8010)	Win32: Thread creation failed
DoPERR_BADOS	(0x8011)	Not supported operating system
DoPERR_THUNK	(0x8012)	Win32: Thread creation failed
DoPERR_INTERNAL	(-1)	Internal driver error

15 Changes Version 1 → 2

- New sensor unit UNIT_INC_REV [Increments/Revolution].
- 16 measuring channels. New 'SENSOR_xx', 'MCBIT_xx' constants.
- 16 analogue output channels. New 'OUT_xx' constants.
- 10 channel supervisions.
- New 'BIN_xx' 'BOUT_xx' constants.
- Changed 'MAX_MC', 'MAX_OC' constants.
- New constant 'SENSOR_D' for digipoti (old digipoti was 'SENSOR_O7').
- New parameters in struct DoPESenDef: 'CtrlChannel', 'UseEeprom', Correction'.
- New parameters in struct DoPECtrlSenDef: 'PosTd', 'PosGenTd', 'SpeedTd', 'SpeedGenTd' and 'AccK'.
- 'UpperSoftLimit', 'LowerSoftLimit' and 'SoftLimitReaction' removed.
- New type double in struct DoPEOutChaDef for 'MaxValue','MinValue','InitValue'
- PaType removed
- Parameters removed from struct DoPEBitOutDef: 'FlashMask', 'SetMask'.
- New parameters in struct DoPEBitInDef: 'StopMask', 'StopLevel'. 'SetMask' removed.
- New parameters in struct DoPEMachineDef: 'Clampxx', 'Shieldxx'. MachineType' removed
- New parameters in struct DoPEVersion: 'PeInterfacePC', 'DpxVer'.
- Struct DoPEDPotiDef and function DoPERdDPotiDef() deleted. The digipoti has to be handled as normal sensor. Valid parameters are: 'Connector', 'Sign', Offset', 'Scale'.
- New struct DoPEGeneralData.
- New struct DoPESetup.
- New constants for 'Mode' in DoPECosineV.
- New constants for 'CtrlState2': CTRL_UPPER_LIMIT, CTRL_LOWER_LIMIT,
- CTRL_CALIBRATION'
- Constants for 'McState' removed.
- New interface function DoPEWrSensorDef, DoPEWrCtrlSensorDef, DoPEWrOutChannelDef, DoPEWrBitOutDef, DoPEWrBitInDef and DoPEWrMachineDef.
- New interface function DoPESetupOpen, DoPESetupClose, DoPERdSetupNumber
- New interface function DoPERdSetupAll, DopeWrSetupAll
- New interface function DoPERdLinTbl, DopeWrLinTbl
- New interface function DoPERdSysUserData, DoPEWrSysUserData
- New interface function DoPERdGeneralData, DoPEWrGeneralData
- New interface function DoPERdBitInput, DoPEWrBitOutput
- New interface function DoPECosineX, DoPECosineV
- New interface function DoPECtrlP_Xpp
- New interface function DoPEMTSpecial
- New interface function DoPECtrlPGKTd
- New interface function DoPEDspBeamScreen, DoPEDspBeamValue
- New interface function DoPESetCheckLimit, DoPEClrCheckLimit
- New interface function DoPEShieldLimit, DoPEShieldDisable, DoPEShieldLock
- New interface function DoPESpeedLimit

- New error code DoPERR_SETUPOPEN
- New command error codes DoPERR_CMD_...
- New command ErrorNumber definitions CMD_ERROR_...
- New connector constants.
- DoPECtrl... K/Ti parameters changed from double to unsigned long/short
- New default measuring dada record:
- New SysStateN and additional measuring channels and bit input devices.
- OutChanN, BitOutN, McState, McCal, FatalError, ErrorLevel and CtrlState removed.
- McOption1..11 renamed to SensorD, Sensor4..15
- New Events DoPEEVT_DATAOVERFLOW, DoPEEVT_ACK, DoPEEVT_NAK and DoPEEVT_ALL
- New struct DoPECommandError and DoPEEPError definitions
- Application version string in DoPEVersion struct changed to 13 char length.
- DoPEOffsK renamed to DoPEOffsC
- Definitions CHK_BELOW and CHK_ABOVE corrected
- New typedefinitions for messages not fitting the DoPEMCM sctructure: DoPESftM for 'Softend' Message
- DoPEOffsCM for 'Offset-Correction' Message
- DoPECheckM for 'Measuring Channel Supervision' Message

16Important Changes to Version 2.21, 2.22

New function DoPEOpenLink replaces the old function DoPEOpenConnection.
New function DoPECloseLink replaces the old function DoPECloseConnection.
It is necessary to recompile the application program before using this DoPE version.

New moving command DoPEPosExt(Sync) and DoPEPosExt_A(Sync)

17 Important Changes to Version 2.23, 2.24

- WIN16 support removed
- New constant definitions for 'MsgId' in MOVE_CTRL_MSG: CMSG_SHIELD_ERR
- **Support of serial Sensors**
  - New connector definitions CON_X62A, CON_X62B, CON_X62C, CON_X62D
  - DoPEWrSensorMsg(Sync)
  - New message type definition SENSOR_MSG: DoPESensorM
- **Functions to read and write Sensor EEPROM's**
  - DoPERdSensorHeaderData
  - DoPERdSensorAnalogueData
  - DoPEWrSensorAnalogueData
  - DoPERdSensorIncData
  - DoPEWrSensorIncData
  - DoPERdSensorAbsData
  - DoPEWrSensorAbsData
  - DoPERdSensorUserData
  - DoPEWrSensorUserData
  - DoPERdSensorConKey
- **Support of reference signal of a incremental sensor**
  - DoPESetRefSignalMode
  - DoPESetRefSignalTare
  - New MOVE_CTRL_MSG message definition CMSG_REFSIGNAL
  - New MOVE_CTRL_MSG message type definition DoPERefSignalM
- **Support of synchronized data acquisition and control**
  - DoPESynchronizeSystemMode(Sync)
  - DoPESynchronizeSystemStart(Sync)
- **New functions for controller parameter**
  - DoPEPosPID(Sync)
  - DoPESpeedPID(Sync)
  - DoPEPosFeedForward(Sync)

New DoPE functions:

- DoPECtrlTestValues
- DoPESetOutChannelOffset(Sync)
- New set-up Offset parameter in DoPEOutChaDef