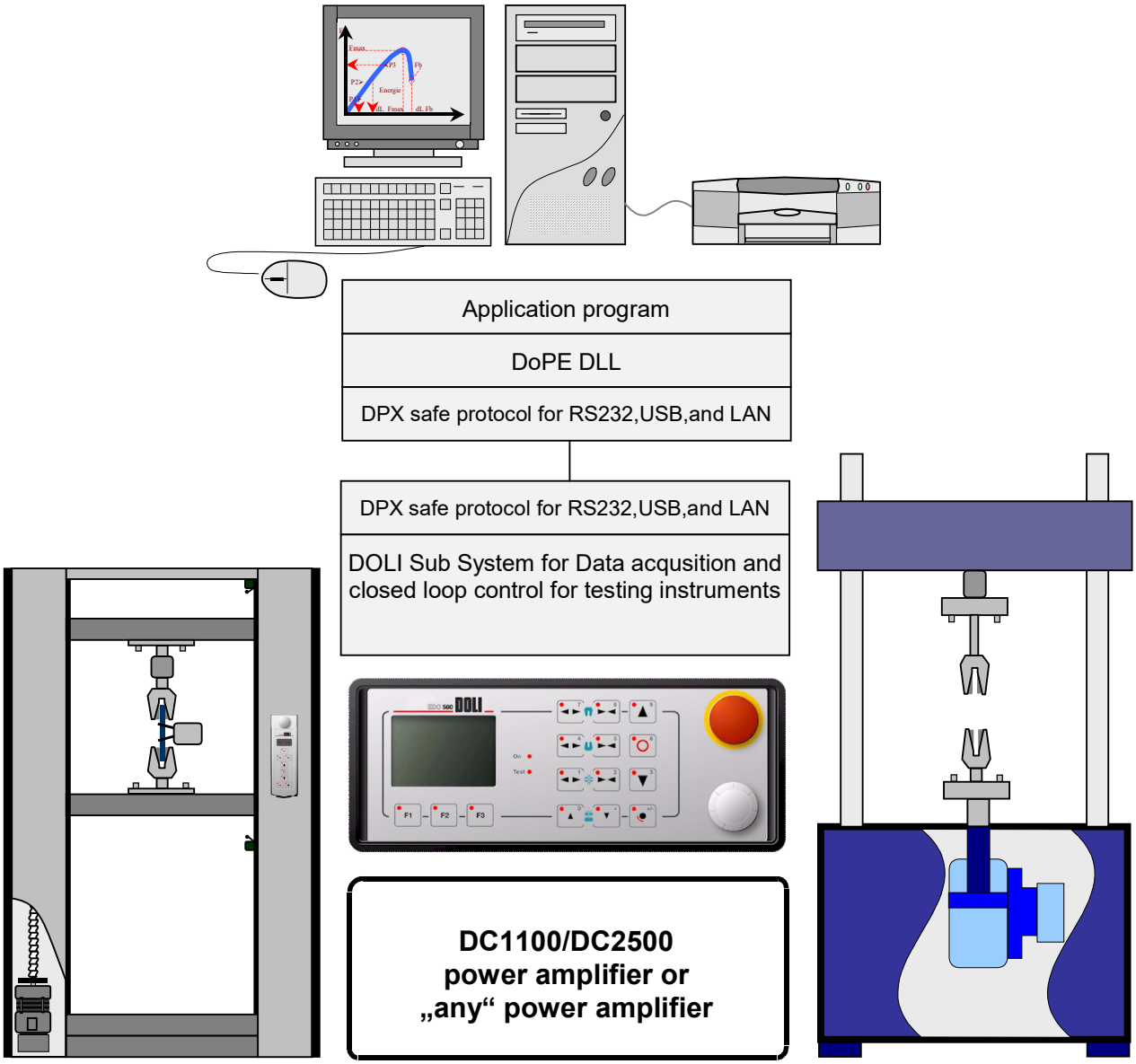
DoPE
Documentation



Headquarters:

DOLI Elektronik GmbH
Rudolf-Diesel-Str. 3
72525 Münsingen, Germany

☎ +49 89 20243 - 0
Fax +49 89 20243 - 232
Email info@doli.de
Web www.doli.de

Development & Service:

DOLI Elektronik GmbH
Adi-Maislinger-Str. 7
81373 Munich, Germany

☎ +49 89 20243 - 0
Fax +49 89 20243 - 243

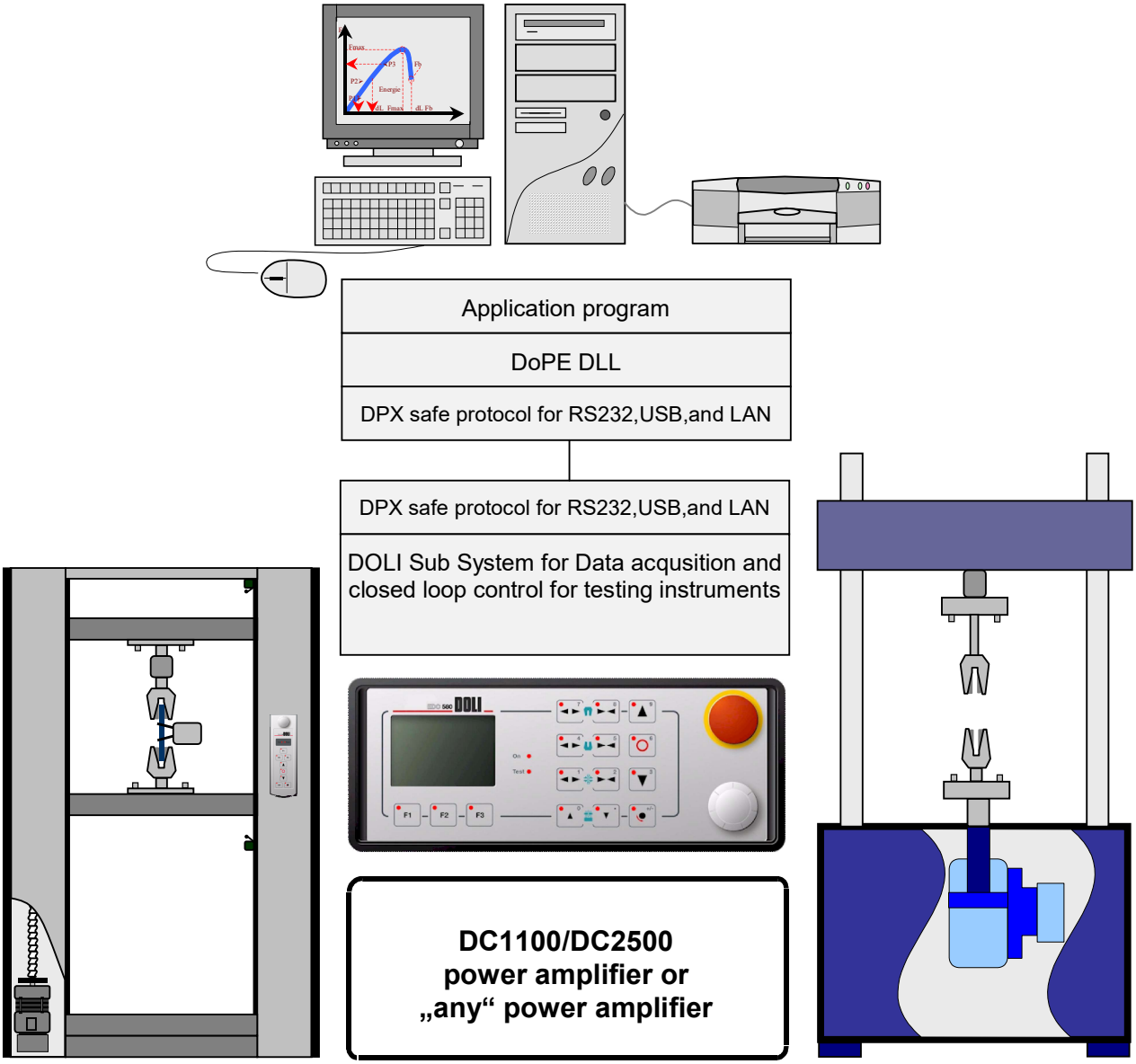
Contents

1 Overview

DOLI has 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, EDC60/120, EDC220/222, EDC580/580V and others. It is configurable within a wide range to satisfy all requirements at a testing instrument.

Communication between EDC and PC is handled by a communication layer, called DPX. DPX is a safe protocol and supports currently RS232, USB, and Ethernet (LAN) as transportation layer.

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



1.1 Tasks of DoPE

- DoPE will initialize the EDC according to predefined set-up data, stored in EDC memory.
- DoPE provides the application program with 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 /2000/XP/Vista/7.

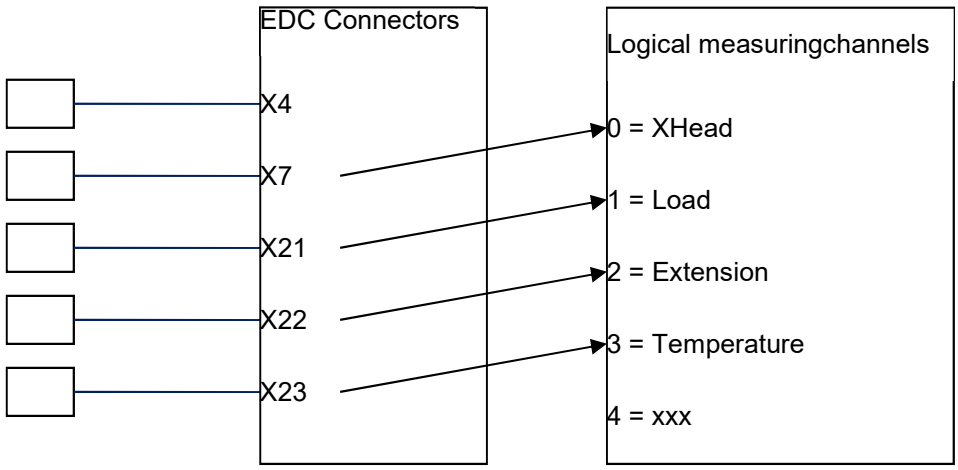
1.2 Initializing

The EDC firmware 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.

Use DOLI InstallationCenter to edit all necessary set-up data.



All Set-up data will be stored in 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", Visual Basic, and a .NET wrapper.

For these programming languages, simple sample programs are available.

Two DLL's are needed to use DoPE: DoPE.DLL and DoDPX.DLL. The .NET wrapper is placed in the DoPENet.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 "DoPEOpenFunctionID" is in Visual Basic "DoPEVBOpenFunctionID".
- Do not use the DoPE realtime handlers in your VB program. Microsoft's VB developing system doesn't support this feature.
- Use "DoPEVB.DLL" additional to the "DoPE.DLL".

2 Communication Principles

DoPE represents a library that uses functions inside the remote EDC-controller. Physically EDC and PC are connected via a standard communication port, like COM, USB or LAN. A safe protocol is used for communication. This protocol, named DPX, detects ON/OFF-line transitions, transmits messages and measured data packets.

The user of DoPE must always be aware of the remote controller and due to this an asynchronous behavior of DoPE.

2.1 Open a connection

The application program establishes a connection by calling one of the **DoPEOpen...** functions. 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 Using Eventhandler

Whenever necessary, EDC will send a message, possibly with parameter to PC. There are many different reasons for such messages. E.g. if a positioning command has reached the destination, EDC will send a message to PC, reporting the destination has been reached.

To make it easy for the application program to get these messages, DoPE offers an event handler interface.For each event, the application program wants to process, an event handler must be installed. DoPE will call these event handlers, whenever the event occurs.

2.3 Transaction Number (TAN)

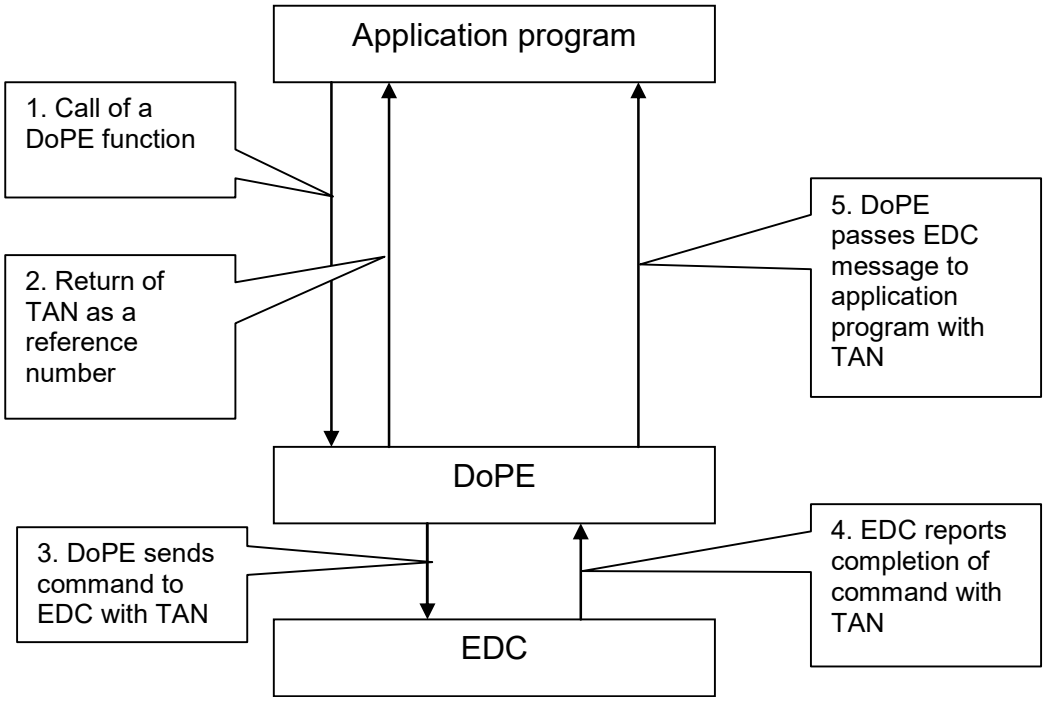
Some of the following commands have pointers to a transaction number (lpusTAN) as a parameter. This TAN may be used to identify messages, which are received asynchronously, to the command.

Example: A positioning command is sent 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, which may send a message later on.

Some DoPE commands will send more than one command to the EDC. In this case two pointers to transaction numbers are needed (lpusTANFirst, lpusTANLast). All messages with TAN’s between lpusTANFirst and lpusTANLast belong to this single DoPE command.

Of course to analyze the TAN is optional. If NULL-Pointers are passed, no TAN’s are returned.



2.4 Synchronous / Asynchronous DoPE Commands

Most of the DoPE commands will be passed from DoPE to the connected EDC.
EDC-firmware checks the command and its parameter, and returns an error code.

Synchronous commands wait until the EDC has checked the parameter and return the result.

Asynchronous commands are just transmitted to the EDC and return to the caller immediately. The result of parameter checks inside the EDC will activate anOnCommandError event.

We recommend using **Synchronous commands.**

In both cases positioning commands will report completion of the command after destination position is reached.

3 Establish communication to EDC

3.1 DoPEOpenDeviceID / DoPEOpenFunctionID

Open the given DoPE link with a matching device or function ID. All USB and LAN communication ports are scanned, starting with USB. All LAN ports of the PC are included to the scan. The link parameters are set and the link is established. If DoPEOpenDeviceID / DoPEOpenFunctionID returns DoPERR_TIMEOUT, connection to the EDC did not go online. You must connect the EDC, switch it on and try the open function again.
If the parameter DeviceID in DoPEOpenDeviceID is set to ZERO, DoPE will open communication to the first EDC that answers. This method is recommended for systems with only one EDC.
This function is available for USB and LAN connections only.

Minimum requirements: EDC580/220, DoPE 2.59

Function declaration		Description
extern unsigned DLLAPI	DoPEOpenDeviceID / DoPEOpenFunctionID (	Function returns Error constant (DoPERR_xxxx)
unsigned	DeviceID/ FunctionID	EDC Device ID / Function ID
unsigned	RcvBuffers	Number of requested receiver buffers for messages.This number of messages can be stored inside DoPE until they are read with DoPEGetMsg function.
unsigned	XmitBuffers	Number of requested transmitter buffers for messages.This number of messages can be stored inside DoPE.They will be transmitted by DoPE to EDC.
unsigned	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 will be overwritten!
unsigned	APIVersion	DoPE API version used by the DoPE user.
void	*Reserved	Reserved for future use
DoPE_HANDLE	DoPEHdl	Pointer to memory for DoPE link handle **This handle has to be used for all further DoPE commands as a reference for this link.**

3.2 DoPEOpenAll / DoPEwOpenAll

Open all available DoPE links and fill the open link info table. All USB and LAN communication ports are scanned, starting with DoPEPORT_USB. All LAN ports of the PC are included to the scan.
DoPEOpenAll will return strings in PortInfo and ModuleInfo as ASCII, while DoPE**w**OpenAll will return wide character strings.

Minimum requirements:
DoPEOpenAll EDC580/220, DoPE 2.59
DoPEwOpenAll EDC580/220, DoPE 2.65

Function declaration			Description
extern unsigned DLLAPI		DoPEOpenAll (	Function returns Error constant (DoPERR_xxxx)
	unsigned	RcvBuffers	Number of requested receiver buffers for messages.This number of messages can be stored inside DoPEuntilthey are read with DoPEGetMsg function.
	unsigned	XmitBuffers	Number of requested transmitter buffers for messages.This number of messages can be stored inside DoPE.They will be transmitted by DoPE to EDC.
	unsigned	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 will be overwritten!
	unsigned	APIVersion	DoPE API version used by the DoPE user.
	void	*Reserved	Reserved for future use
	unsigned	InfoTableMaxEntries	Number of entries that can be stored to the info table.
	unsigned	*InfoTableValidEntries	Pointer to storage for the number of valid entries in the info table.
	DoPEOpenLinkInfo	InfoTable[ ])	Infotable will be filled by with all detected EDC's

```
typedef struct
{
  DoPE_HANDLE     DoPEHdl;
  unsigned        PortType;              /* DoPEPORT_USB or DoPEPORT_LAN    */
  DoPE_PORTINFO   PortInfo;
  DoPEModuleInfo ModuleInfo;
} DoPEOpenLinkInfo;
```

Function declaration			Description
extern unsigned DLLAPI		DoPEwOpenAll (	Function returns Error constant (DoPERR_xxxx)
	unsigned	RcvBuffers	Number of requested receiver buffers for messages.This number of messages can be stored inside DoPEuntilthey are read with DoPEGetMsg function.
	unsigned	XmitBuffers	Number of requested transmitter buffers for messages.This number of messages can be stored inside DoPE.They will be transmitted by DoPE to EDC.
	unsigned	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 will be overwritten!
	unsigned	APIVersion	DoPE API version used by the DoPE user.
	void	*Reserved	Reserved for future use
	unsigned	InfoTableMaxEntries	Number of entries that can be stored to the info table.
	unsigned	*InfoTableValidEntries	Pointer to storage for the number of valid entries in the info table.
	DoPEwOpenLinkInfo	InfoTable[ ])	Infotable will be filled by with all detected EDC's

```
typedef struct
{
  DoPE_HANDLE     DoPEHdl;
  unsigned        PortType;              /* DoPEPORT_USB or DoPEPORT_LAN    */
  DoPE_wPORTINFO  PortInfo;
  DoPEwModuleInfo ModuleInfo;
} DoPEwOpenLinkInfo;
```

3.3 DoPECloseAll / DoPEwCloseAll

Close all DoPE links previously opened by DoPEOpenAll / DoPE**w**OpenAll.

Minimum requirements:
DoPEClopseAll EDC580/220, DoPE 2.59
DoPEwCloseAll EDC580/220, DoPE 2.65

Function declaration	Description
extern unsigned DLLAPI DoPECloseAll (unsigned InfoTableValidEntries DoPEOpenLinkInfo InfoTable[ ])	Function returns Error constant (DoPERR_xxxx) Number of valid entries in the info table. Info table containing valid DoPE handles for all links to close. All DoPE handles in the InfoTable are set to NULL.
extern unsigned DLLAPI DoPEwCloseAll (unsigned InfoTableValidEntries DoPEwOpenLinkInfo InfoTable[ ])	Function returns Error constant (DoPERR_xxxx) Number of valid entries in the info table. Info table containing valid DoPE handles for all links to close. All DoPE handles in the InfoTable are set to NULL.

3.4 DoPEPortInfo / DoPEwPortInfo / DoPECurrentPortInfo / DoPEwCurrentPortInfo

Get the port information for PC-communication ports (COM, USB, LAN)
DoPEPortInfo will return strings in ASCII, while DoPE**w**PortInfo will return wide character strings.

Minimum requirements:
DoPEPortInfo DoPE 2.50
DoPEwPortInfo DoPE 2.65
DoPE(w)CurrentPortInfo DoPE 2.70

Function declaration	Description
extern unsigned DLLAPI DoPEPortInfo (unsigned Port, unsigned First, DoPE_PORTINFO *pPortInfo)	Function returns Error constant (DoPERR_xxxx) (DPXERR_BADPORT if no more port info is available) Communication Port Type. (DoPEPORT_USB, DoPEPORT_LAN or DoPEPORT_COM) != 0: get the first port information == 0: get the next port information pointer to port info structure
extern unsigned DLLAPI DoPEwPortInfo (unsigned Port, unsigned First, DoPE_wPORTINFO *pPortInfo)	Function returns Error constant (DoPERR_xxxx) (DPXERR_BADPORT if no more port info is available) Communication Port Type. (DoPEPORT_USB, DoPEPORT_LAN or DoPEPORT_COM) != 0: get the first port information == 0: get the next port information pointer to port info structure

Get the current port informationof an established link
DoPEPortInfo will return strings in ASCII, while DoPE**w**PortInfo will return wide character strings.

Function declaration	Description
extern unsigned DLLAPI DoPECurrentPortInfo (DoPE_HANDLE DoPEHdl, DoPE_PORTINFO *pPortInfo)	Function returns Error constant (DoPERR_xxxx) DoPE link handle pointer to port info structure
extern unsigned DLLAPI DoPEwCurrentPortInfo (DoPE_HANDLE DoPEHdl, DoPE_wPORTINFO *pPortInfo)	Function returns Error constant (DoPERR_xxxx) DoPE link handle pointer to port info structure

```
typedef struct
{
  DWORD     Ix;                         /* Device index                    */
  char      Name[DoPE_PORTNAMELEN];     /* Device name                     */
DWORD     ComPort;                    /* COM port (0=COM1,...)           */
  MAC       NicMAC;                     /* NIC address                     */
} DoPE_PORTINFO;
```

```
typedef struct
{
  DWORD     Ix;                         /* Device index                    */
  wchar_t   Name[DoPE_PORTNAMELEN];     /* Device name                     */
  DWORD     ComPort;                    /* COM port                        */
  MAC       NicMAC;                     /* NIC address                     */
} DoPE_wPORTINFO;
```

3.5 DoPECloseLink

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.	
Minimum requirements: DoPE 2.21	
Function declaration	**Description**
extern unsigned DLLAPI DoPECloseLink (DoPE_HANDLE DoPEHdl)	Function returns Error constant (DoPERR_xxxx) DoPE link handle

4 Message and Data handling

4.1 DoPECurrentData

Get current samples from receiver buffer. DoPE receives measuring data in an adjustable time scale. The data record is stored inside DoPE in a circular buffer. You can read the latest data record with this function. For definition of DoPEData record refer to Page 145.	
Minimum requirements: DoPE 2.20	
Function declaration	**Description**
extern unsigned DLLAPI DoPECurrentData(DoPE_HANDLE DoPEHdl, DoPEData *Sample)	Function returns Error constant (DoPERR_xxxx) DoPE link handle Pointer to storage for data record.

4.2 DoPEClearReceiver

Discard all received messages. If you want to get rid of old messages, that are of no interest any more, use this command.	
Function declaration	**Description**
extern unsigned DLLAPI DoPEClearReceiver (DoPE_HANDLE DoPEHdl)	Function returns Error constant (DoPERR_xxxx) DoPE link handle

4.3 DoPEClearTransmitter

Discard all unsent transmitter buffers	
Function declaration	**Description**
extern unsigned DLLAPI DoPEClearTransmitter (DoPE_HANDLE DoPEHdl)	Function returns Error constant (DoPERR_xxxx) DoPE link handle

4.4 DoPEGetErrors

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

Function declaration		Description
extern unsigned DLLAPI DoPE_HANDLE DoPEError *	DoPEGetErrors(DoPEHdl, Error)	Function returns Error constant (DoPERR_xxxx) DoPE link handle Pointer to buffer for error counters
typedef struct { unsigned unsigned unsigned unsigned unsigned unsigned unsigned unsigned unsigned } DoPEError;	 Parity; Overrun; Frame; InvACK; NoBuffer; DLESeq; BufOFlow; BccErr; MwError;	Error counters DoPE link handle Parity errors Overrun errors Framing movement Invalid ACKs received No receiver buffer available Invalid DLE sequence Receiver buffer overflow Checksum error Invalid sample encoding

4.5 DoPEClearErrors

Clear current error counter values.

Minimum requirements: DoPE 2.51

Function declaration		Description
extern unsigned DLLAPI DoPE_HANDLE	DoPEClearErrors(DoPEHdl)	Function returns Error constant (DoPERR_xxxx) DoPE link handle

4.6 DoPEGetState

Get state information structure.

Function declaration		Description
extern unsigned DLLAPI DoPEState DoPE_HANDLE	DoPEGetState (*Status DoPEHdl)	Function returns Error constant (DoPERR_xxxx) Pointer to buffer for state information DoPE link handle
typedef struct { unsigned unsigned unsigned } DoPEState; **Constants for ComState** COM_STATE_OFF COM_STATE_OFFLINE COM_STATE_INITCYCLE COM_STATE_ONLINE	 ComState; RcvBuffer; XmitBuffer;	 Communication state Number messages in receive queue Number of messages to be transmitted Link is disabled Link is offline Link is initializing Link is established and OnLine

5 Movement commands

DoPE provides several commands for moving the cross head of a machine. The movement may be in different control modes, such as position, load or extension control. There are simple commands like move up, or down with a certain speed without explicit destination, or commands with a given destination. Furthermore DoPE supplies commands with a destination in another control channel such as go in position control to a load destination, and cyclic commands.

For most of the movement commands DoPEprovides a command that uses default acceleration e.g. DoPEPos and another where acceleration and deceleration are specified in the command e.g. DoPEPos_A.

Furthermore, all movement commands exist in an asynchronous version e.g. DoPEPos, and a synchronous version e.g. DoPEPosSynch. (See2.4)

After a movement has been finished, the event OnPosMsg will be activated.

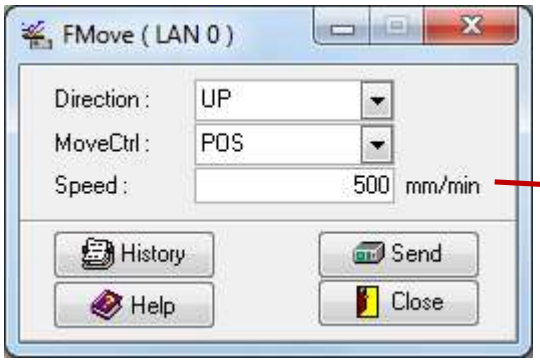
A new movement command terminates an active movement command.

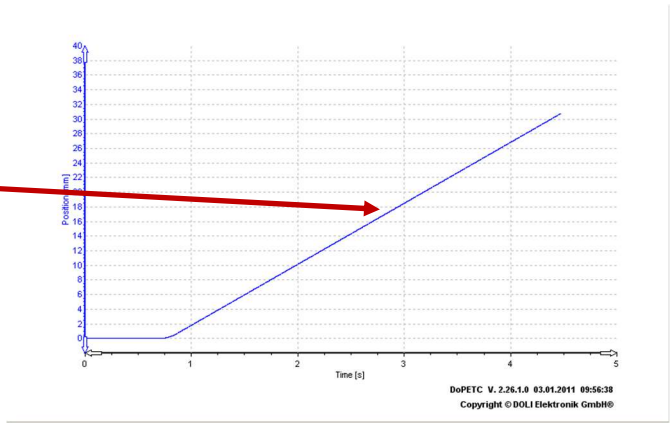
5.1 Simple movement commands

5.1.1 DoPEFMove(Sync), DoPEFMove_A(Sync)

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

Minimum requirements:
DoPEFMove_A(Sync) EDC120/60, EdcApp 9123.11 / 9133.007, DoPE 2.57

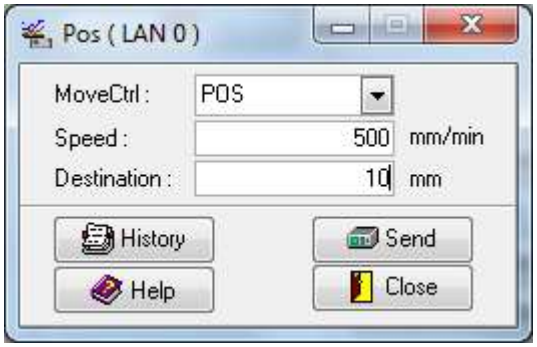


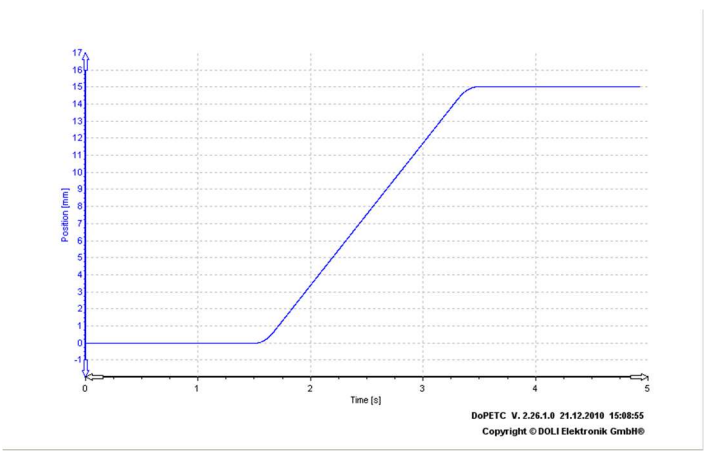


Function declaration		Description	Unit
extern unsigned DLLAPI DoPE_HANDLE unsigned short unsigned short double WORD *	DoPEFMove(DoPEHdl, Direction, MoveCtrl, Speed, lpusTAN);	Function returns Error constant (DoPERR_xxxx) DoPE link handle Direction of movement Control mode ofmovement (CTRL_xxx) Speed for movement Pointer to transaction number.	 Unit/s
extern unsigned DLLAPI DoPE_HANDLE unsigned short unsigned short double double WORD *	DoPEFMove_A(DoPEHdl, Direction, MoveCtrl, Acc Speed, lpusTAN);	Function returns Error constant (DoPERR_xxxx) DoPE link handle Direction of movement Control mode of movement(CTRL_xxx) Acceleration Speed for movement Pointer to transaction number.	 Unit/s² Unit/s

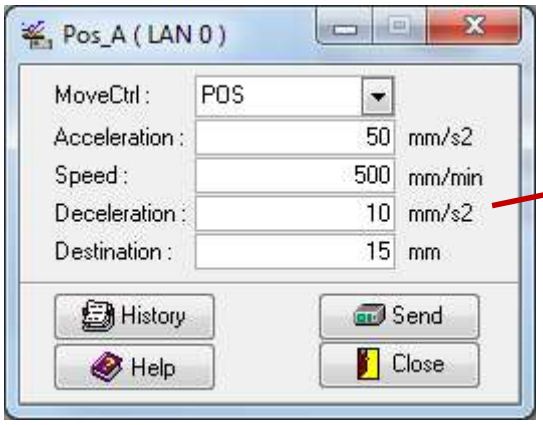
5.1.2 DoPEPos(Sync), DoPEPos_A(Sync)

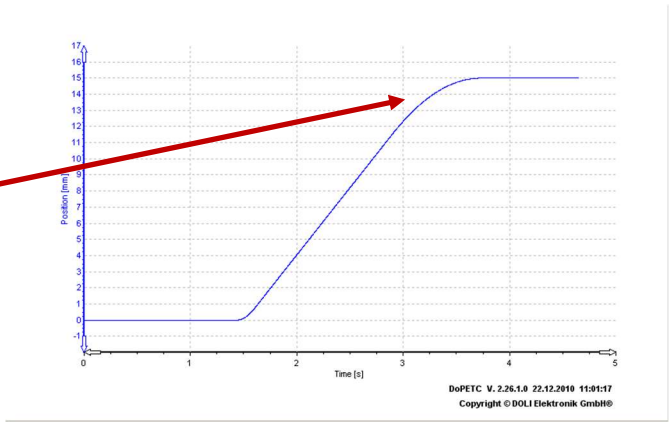
Move cross-head in the specified control mode and speed to the given destination.
DoPEPos will use the default acceleration while DoPEPos_A uses the given acceleration.





Function declaration		Description	Unit
extern unsigned DLLAPI	DoPEPos (	Function returns Error constant (DoPERR_xxxx)	
DoPE_HANDLE	DoPEHdl,	DoPE link handle	
unsigned short	MoveCtrl,	Control mode (CTRL_xxx)	
double	Speed,	Speed for movement	Unit/s
double	Destination	Final destination	Unit
WORD *	lpusTAN);	Pointer to transaction number.	



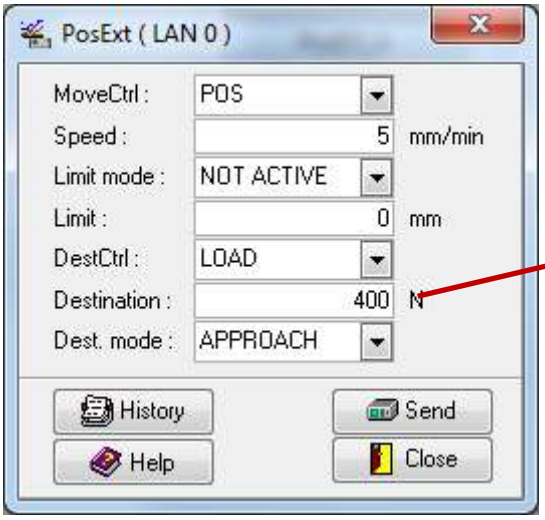


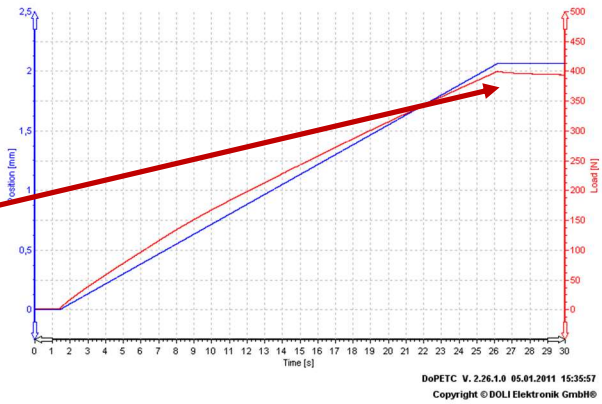
Function declaration		Description	Unit
extern unsigned DLLAPI	DoPEPos_A (	Function returns Error constant (DoPERR_xxxx)	
DoPE_HANDLE	DoPEHdl,	DoPE link handle	
unsigned short	MoveCtrl,	Control mode (CTRL_xxx)	
double	Acc	Acceleration	Unit/s²
double	Speed,	Speed for movement	Unit/s
double	Dec	Deceleration	Unit/s²
double	Destination	Final destination	Unit
WORD *	lpusTAN);	Pointer to transaction number.	

5.1.3 DoPEPosExt(Sync), DoPEPosExt_A(Sync)

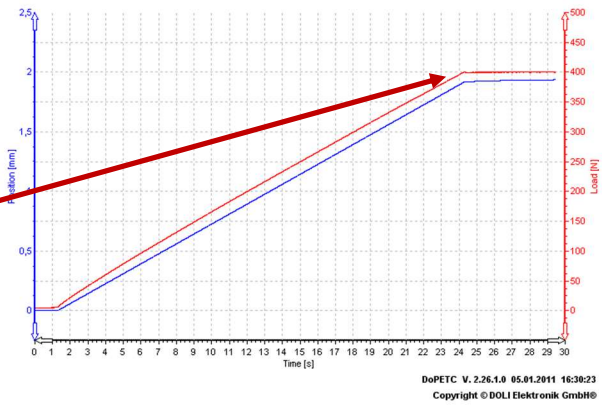
Move cross-head in the specified control mode ‘MoveCtrl’ and ‘Speed’ to the given ‘Destination’.
‘DestinationCtrl’ may be different to ‘MoveCtrl’. Default acceleration and deceleration will be used.
The Parameter ‘DestinationMode’ specifies how to reach destination position and the action after reaching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

Minimum requirements: EDC120/60, DoPE 2.21

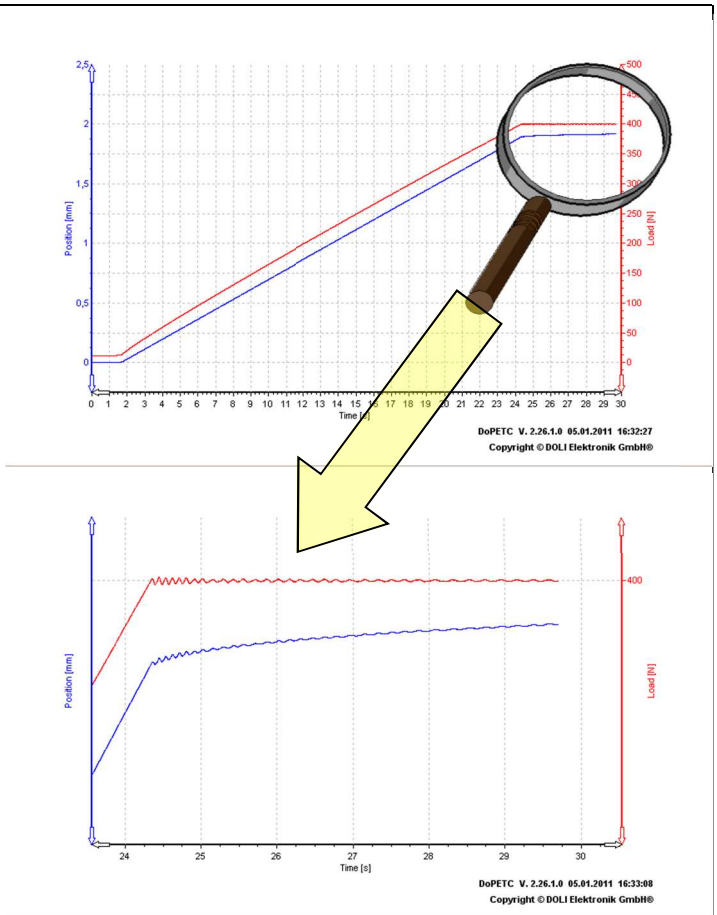




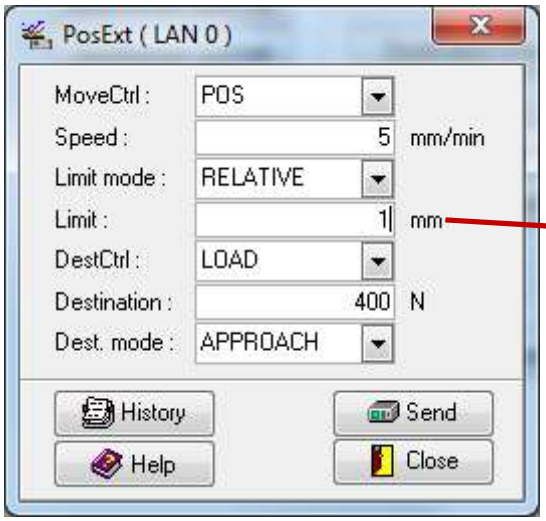
Halt in position control after load destination is reached.
Due to specimen relaxation, load will decrease!

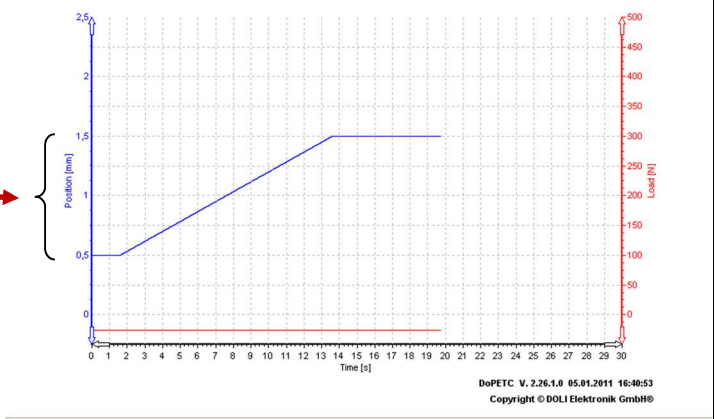


Switch to load control just before load destination is reached. Load is kept constant, position is increasing, due to specimen relaxation.

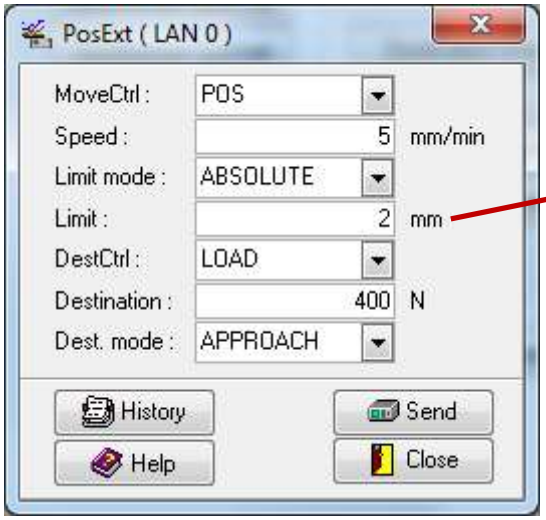


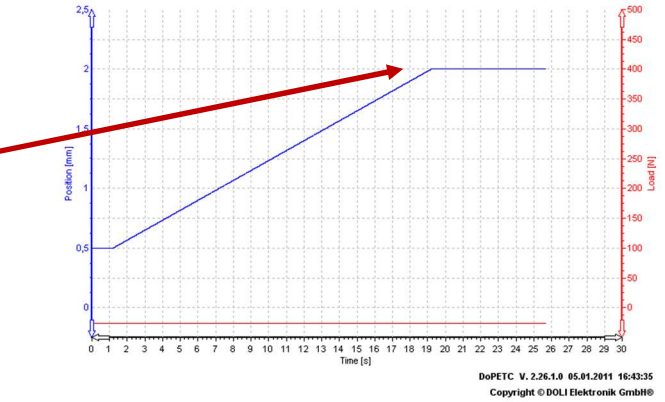
Halt in position control after load destination is reached and maintain this load destination in position control.





Load destination cannot be reached after 1 mm relative movement. Movement command is terminated.





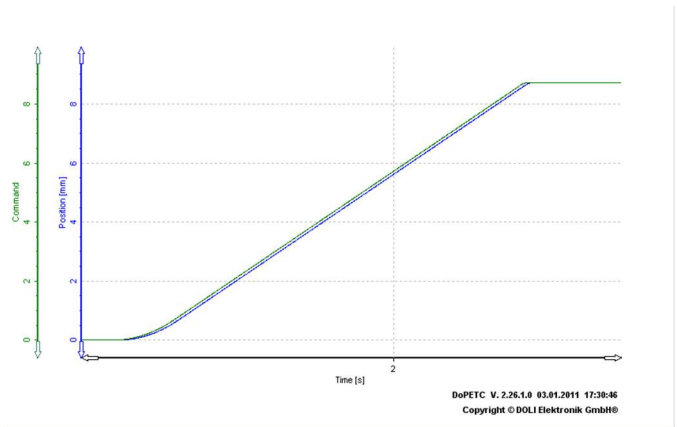
Load destination cannot be reached after the absolute position of 2mm is reached. Movement command is terminated.

Function declaration		Description	Unit
extern unsigned DLLAPI	DoPEPosExt (	Function returns Error constant (DoPERR_xxxx)	
DoPE_HANDLE	DoPEHdl,	DoPE link handle	
unsigned short	MoveCtrl,	Control mode (CTRL_xxx)	
double	Speed,	Speed for movement	Unit/s
unsigned short	LimitMode,	**LIMIT_ABSOLUTE**: Limit is a absolute position **LIMIT_RELATIVE**: Limit is a distance (relative) position **LIMIT_NOT_ACTIVE**: No Limit is active	
double	Limit,	Limit position in case Destination cannot be reached	Unit
unsigned short	DestinationCtrl,	Channel definition for destination (CTRL_xxx)	
double	Destination	Final destination	Unit
unsigned short	DestinationMode,	**DEST_APPROACH**: Halt after reaching destination, do not change control mode. **DEST_POSITION**: Switch to 'DestinationCtrl'just before 'Destination' is reached. Thenmove to 'Destination'. **DEST_MAINTAIN**: No change of control mode at destination but maintain destination in 'MoveCtrl'mode.	
WORD *	lpusTAN);	Pointer to transaction number.	
extern unsigned DLLAPI	DoPEPosExt_A (	Function returns Error constant (DoPERR_xxxx)	
DoPE_HANDLE	DoPEHdl,	DoPE link handle	
unsigned short	MoveCtrl,	Control mode (CTRL_xxx)	
double	Acc,	Acceleration	Unit/s²
double	Speed,	Speed for movement	Unit/s
double	DecLimit,	Deceleration at Limit	Unit/s²
unsigned short	LimitMode,	**LIMIT_ABSOLUTE**: Limit is a absolute position **LIMIT_RELATIVE**: Limit is a distance (relative) position **LIMIT_NOT_ACTIVE**: No Limit is active	
double	Limit,	Limit position in case Destination cannot be reached	Unit
unsigned short	DestinationCtrl,	Channel definition for destination (CTRL_xxx)	
double	DecDestination,	Deceleration at Limit	Unit/s²
double	Destination	Final destination	Unit
unsigned short	DestinationMode,	**DEST_APPROACH**: Halt after reaching destination, do not change control mode. **DEST_POSITION**: Switch to 'DestinationCtrl'just before 'Destination' is reached. Thenmove to 'Destination'. **DEST_MAINTAIN**: No change of control mode at destination but maintain destination in'MoveCtrl'mode.	
WORD *	lpusTAN);	Pointer to transaction number.	

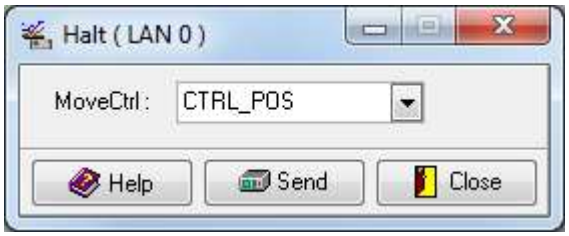
5.2 Halt commands

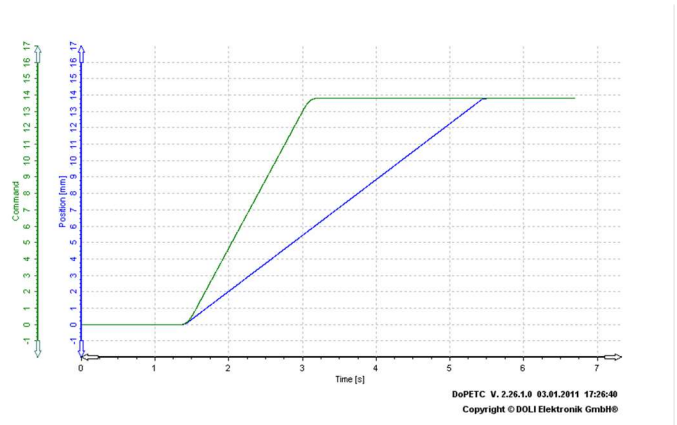
5.2.1 DoPEHalt(Sync), DoPEHalt_A(Sync)

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.



DoPEHalt will terminate a current moving command. The Command will be decelerated to ZERO speed.



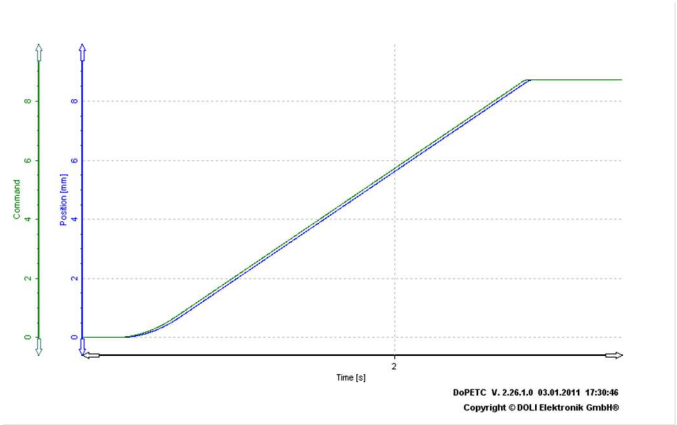


The machine cannot follow the desired speed.
(nominal speed in set-up is too high)
Command will be decelerated to ZERO speed.
After the command has decreased speed to zero, the machine is still moving, until the command position has been reached.

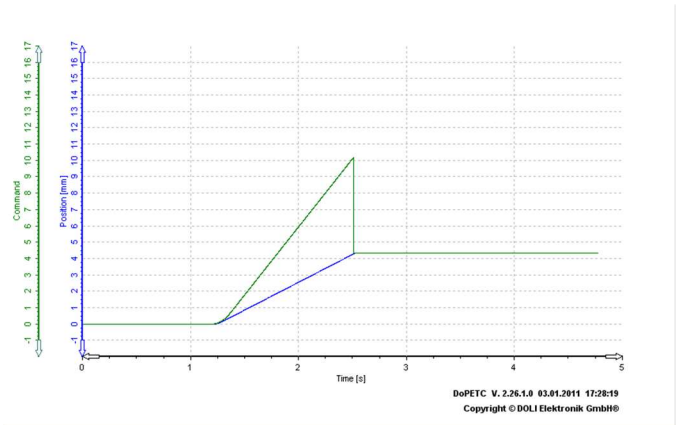
Function declaration		Description	Unit
extern unsigned DLLAPI DoPE_HANDLE unsigned short WORD *	DoPEHalt(DoPEHdl, MoveCtrl, lpusTAN);	Function returns Error constant (DoPERR_xxxx) DoPE link handle Control mode (CTRL_xxx) Pointer to transaction number.	
extern unsigned DLLAPI DoPE_HANDLE unsigned short double WORD *	DoPEHalt_A(DoPEHdl, MoveCtrl, Dec lpusTAN);	Function returns Error constant (DoPERR_xxxx) DoPE link handle Control mode of movement (CTRL_xxx) Deceleration Pointer to transaction number.	Unit/s²

5.2.2 DoPESHalt(Sync)

Instant Halt of cross-head in position control mode. If needed, the control mode is changed to position control. Before decelerating with default deceleration, the command is set to the current position.
This is the fastest method to halt the machine in position control.



DoPESHalt will terminate a current moving command.
If nominal speed is set correctly, there is no difference to DoPEHalt command.

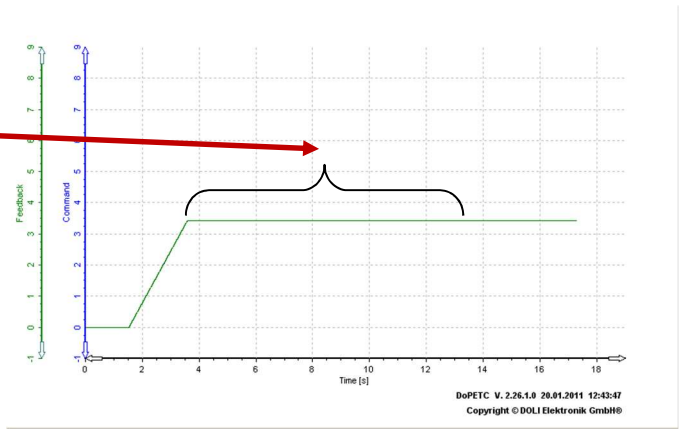


The machine cannot follow the desired speed.
(nominal speed in set-up is too high)
Command will be set to current position and then decelerated to ZERO speed.

Function declaration	Description	Unit
extern unsigned DLLAPI DoPESHalt(DoPE_HANDLE DoPEHdl, WORD * lpusTAN);	Function returns Error constant (DoPERR_xxxx) DoPE link handle Pointer to transaction number.	

5.2.3 DoPEHaltW(Sync), DoPEHaltW_A(Sync)

Decelerate from current speed to speed zero and wait delay time. After delay time has expired, the command has finished and a position reached message is generated.
This command is only useful inside a combined movement command.

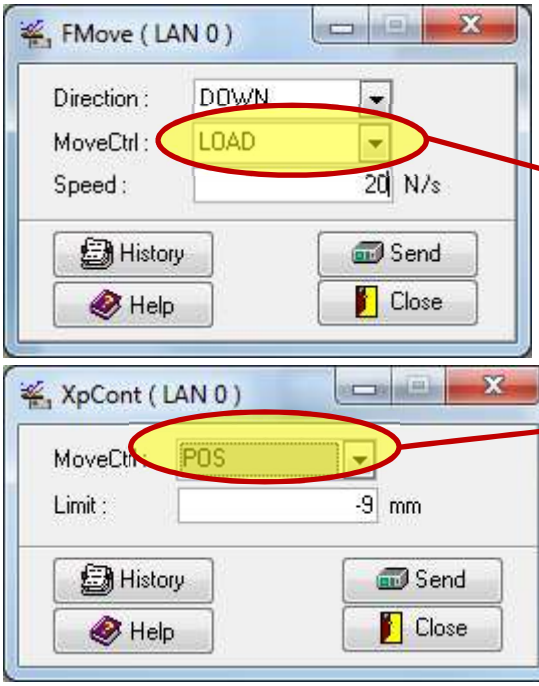


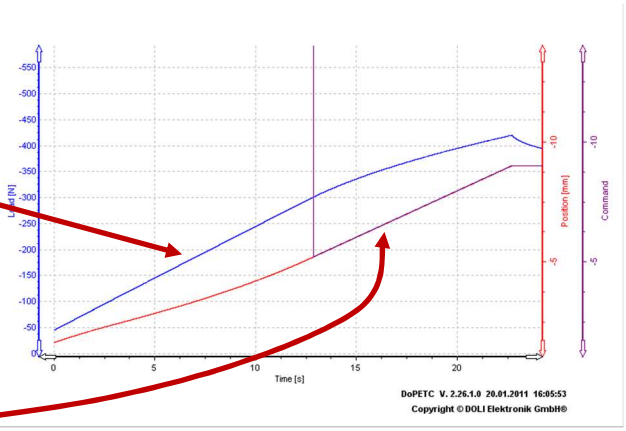
DoPEHalt will terminate a current moving command. The message, indicating the end of this command will be delayed. (here 10 seconds)

Function declaration		Description	Unit
extern unsigned DLLAPI DoPE_HANDLE unsigned short double WORD *	DoPEHaltW(DoPEHdl, MoveCtrl, Delay lpusTAN);	Function returns Error constant (DoPERR_xxxx) DoPE link handle Control mode (CTRL_xxx) Delay time Pointer to transaction number.	 s
extern unsigned DLLAPI DoPE_HANDLE unsigned short double double WORD *	DoPEHaltW_A(DoPEHdl, MoveCtrl, Dec Delay lpusTAN);	Function returns Error constant (DoPERR_xxxx) DoPE link handle Control mode of movement (CTRL_xxx) Deceleration Delay time Pointer to transaction number.	 Unit/s² s

5.2.4 DoPEXpCont(Sync),DoPEXpCont_A(Sync)

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



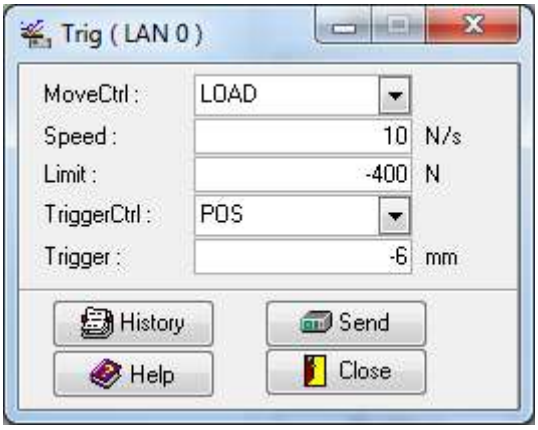


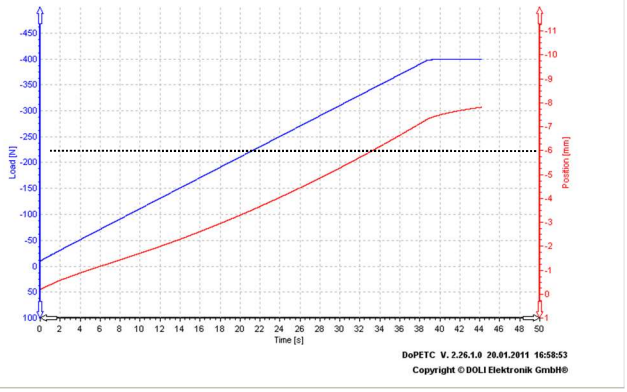
Machine is moving in load control with 20 N/s. At about 17 s, the DoPEXpCont command switches to position control, and continues with the current speed in position. After reaching the Limit position, the command is terminated.

Function declaration		Description	Unit
extern unsigned DLLAPI DoPE_HANDLE unsigned short double WORD *	DoPEXpCont(DoPEHdl, MoveCtrl, Limit lpusTAN);	Function returns Error constant (DoPERR_xxxx) DoPE link handle Control mode (CTRL_xxx) Limit position Pointer to transaction number.	 Unit
extern unsigned DLLAPI DoPE_HANDLE unsigned short double double WORD *	DoPEXpCont_A(DoPEHdl, MoveCtrl, Limit Dec lpusTAN);	Function returns Error constant (DoPERR_xxxx) DoPE link handle Control mode of movement (CTRL_xxx) Limit position Deceleration Pointer to transaction number.	 Unit Unit/s²

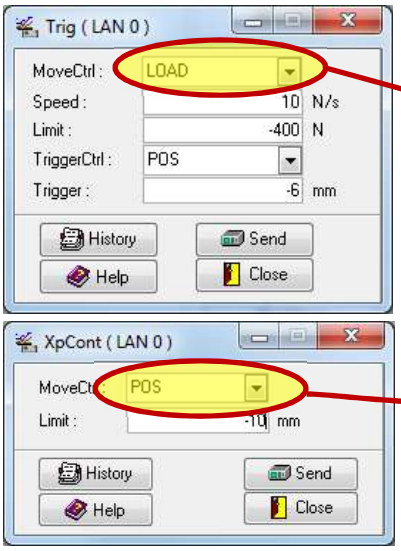
5.2.5 DoPETrig(Sync), DoPETrig_A(Sync)

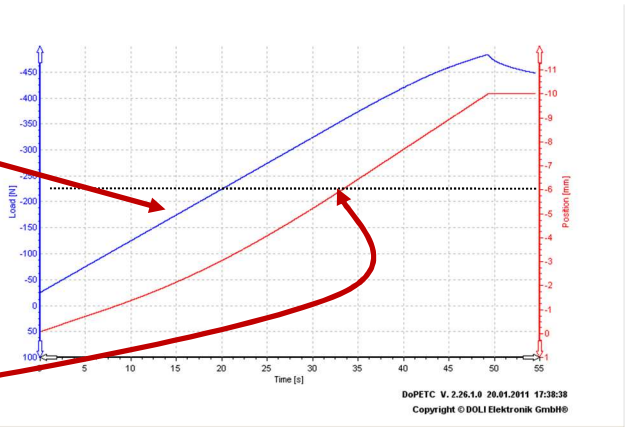
Move cross-head with the specified speed to the limit position. If the trigger position is reached a message will be transmitted and if used inside a combined moving sequence, the next command is activated.





Move in load control with 10 N/s to -400N. A trigger message is generated at -6mm position. Upon this, the PC program may send another e.g. moving command.





The DoPETrig command can be used inside a combined moving command. Upon reaching -6mm, the next command will be automatically started. Here go with the current speed in position to -10mm.

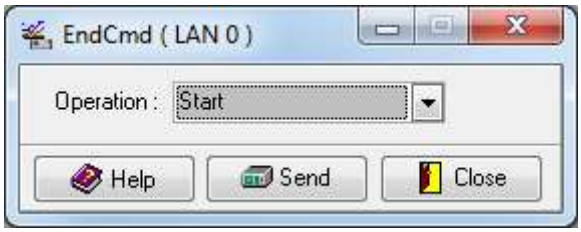
Function declaration		Description	Unit
extern unsigned DLLAPI	DoPETrig(	Function returns Error constant (DoPERR_xxxx)	
DoPE_HANDLE	DoPEHdl,	DoPE link handle	
unsigned short	MoveCtrl,	Control mode (CTRL_xxx)	
double	Speed,	Speed for movement	Unit/s
double	Limit,	Limit position in case Destination cannot be reached	
unsigned short	TriggerCtrl,	Control mode for trigger channel(CTRL_xxx)	
double	Trigger,	Trigger level	
WORD *	lpusTAN);	Pointer to transaction number.	
extern unsigned DLLAPI	DoPETrig_A(	Function returns Error constant (DoPERR_xxxx)	
DoPE_HANDLE	DoPEHdl,	DoPE link handle	
unsigned short	MoveCtrl,	Control mode (CTRL_xxx)	
double	Acc,	Acceleration	Units/s²
double	Speed,	Speed for movement	Unit/s
double	Dec,	Deceleration	Units/s²
double	Limit,	Limit position in case Destination cannot be reached	
unsigned short	TriggerCtrl,	Control mode for trigger channel(CTRL_xxx)	
double	Trigger,	Trigger level	
WORD *	lpusTAN);	Pointer to transaction number.	

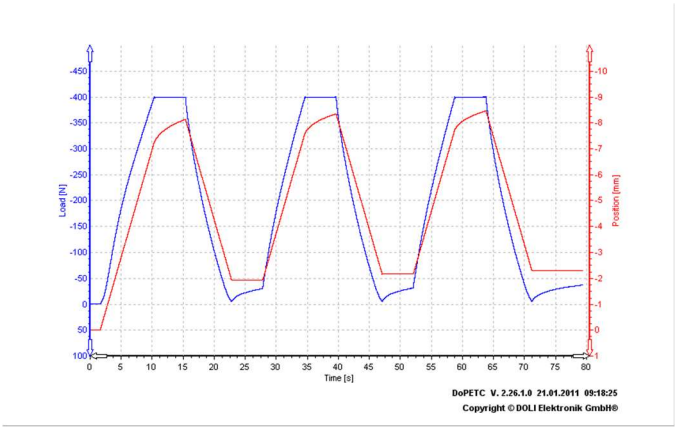
5.3 Combined movement commands

5.3.1 DoPEStartCMD(Sync),DoPEEndCMD(Sync)

All simple moving commands can be used inside a combined movement command. The combined movement command starts with DoPEStartCMD and ends with DoPEEndCMD.


```
DoPEPosExt(CTRL_POS, 0.833, LIMIT_RELATIVE, 10,
           CTRL_LOAD, -400, DEST_POSITION);
DoPEHaltW(CTRL_LOAD, 5);
DoPEPosExt(CTRL_POS, 0.833, LIMIT_RELATIVE, 10,
           CTRL_LOAD, -5, DEST_APPROACH)
DoPEHaltW(CTRL_POS, 5);
```





Repeat following sequence 3 times:

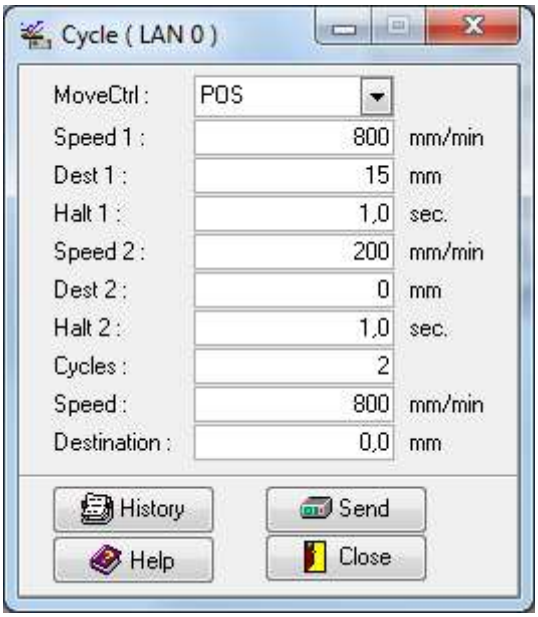
1. Go in position control with 8.33 mm/s (50 mm/min) to -400N and switch to load control.
2. Keep load for 5 seconds.
3. Go in position control with 8.33 mm/s (50 mm/min) to -5N and halt in position control.
4. Keep position for 5 seconds.

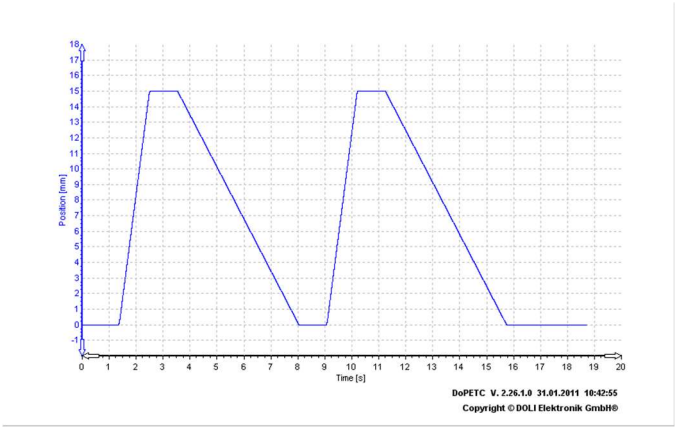
Function declaration		Description	Unit
extern unsigned DLLAPI DoPE_HANDLE unsigned long unsigned short WORD *	DoPEStartCMD (DoPEHdl, Cycles, ModeFlags lpusTAN);	Function returns Error constant (DoPERR_xxxx) DoPE link handle Number of cycles to be done Flags. Both flags may be 'ored'. **CMD_DWND**: Supervise destination window. Next command is started after the current reaches its destination window. Otherwise the next command is started, after the command value reaches its destination **CMD_MESSAGE**: Report intermediate destinations. For each single command a message (OnTPos) is sent after it was finished. After the last command in the sequence is finished a message (OnPos) is always sent. Pointer to transaction number (not for Sync. version).	
extern unsignedDLLAPI DoPE_HANDLE unsigned short WORD *	DoPEEndCMD(DoPEHdl, Operation, lpusTAN);	Function returns Error constant (DoPERR_xxxx) DoPE link handle CMD_START: Start the command sequence. CMD_DISCARD: Discard the command sequence. Pointer to transaction number (not for Sync. version).	

5.4 Complex moving commands

5.4.1 DoPECycle(Sync)

Cycle movement using ramps.





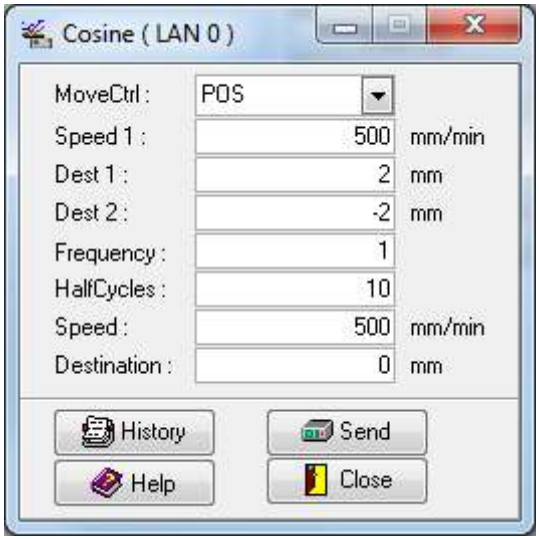
Go with 800mm/min to 15mm, Halt 15mm for 1 second, go with 200mm/min to 0mm and halt 0mm for 1 second.

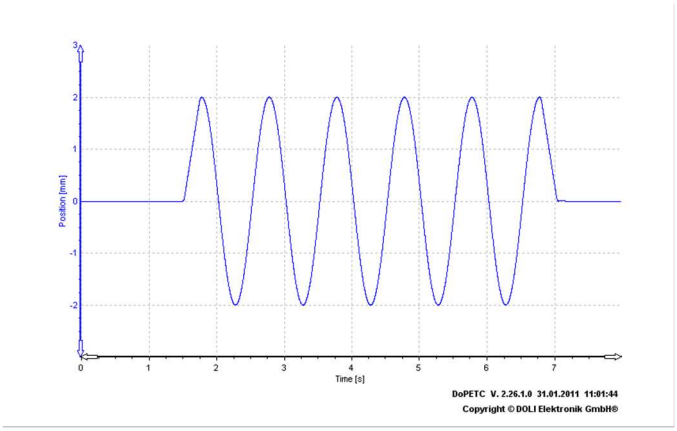
Repeat this sequence two times.

Function declaration		Description	Unit
extern unsigned DLLAPI	DoPECycle(	Function returns Error constant (DoPERR_xxxx)	
DoPE_HANDLE	DoPEHdl	DoPE link handle	
unsigned short	MoveCtrl	Control mode (CTRL_xxx)	
double	Speed1	Speed to reach destination 1	Unit/s
double	Dest1	Destination 1	Unit
double	Halt1	Halt time at destination 1	s
double	Speed2	Speed to reach destination 2	Unit/s
double	Dest2	Destination 2	Unit
double	Halt2	Halt time at destination 2	s
unsigned long	Cycles	Number of cycles	
double	Speed	Speed to final destination	Unit/s
double	Destination	Final destination	Unit
WORD	*lpusTAN	Pointer to transaction number.	

5.4.2 DoPECosine(Sync)

Cosine movement.





Go with 500mm/min to 2mm.
Start Cosine to -2mm. Repeat cosine half cycle 10 times
Go with 500 mm/min to 0mm.

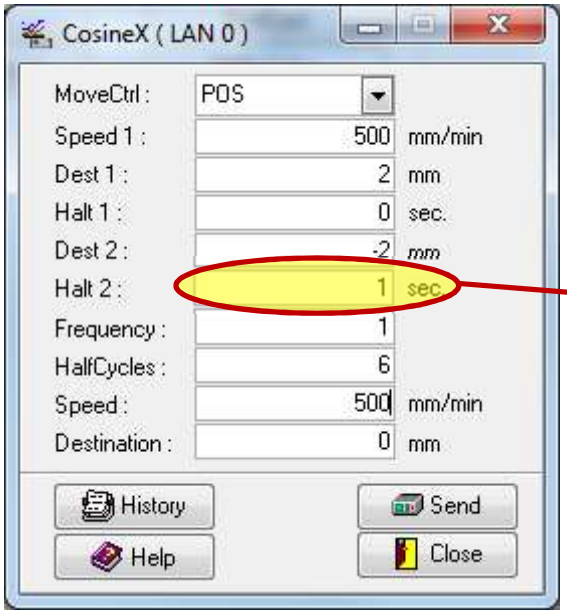
Function declaration		Description	Unit
extern unsigned DLLAPI	DoPECosine(	Function returns Error constant (DoPERR_xxxx)	
DoPE_HANDLE	DoPEHdl	DoPE link handle	
unsigned short	MoveCtrl	Control mode (CTRL_xxx)	
double	Speed1	Speed to reach destination 1	Unit/s
double	Dest1	Destination 1	Unit
double	Dest2	Destination 2	Unit
double	Frequency	Cosine frequency	1/s
unsigned long	HalfCycles	Number of half cycles	
double	Speed	Speed to final destination	Unit/s
double	Destination	Final destination	Unit
WORD	*lpusTAN	Pointer to transaction number.	

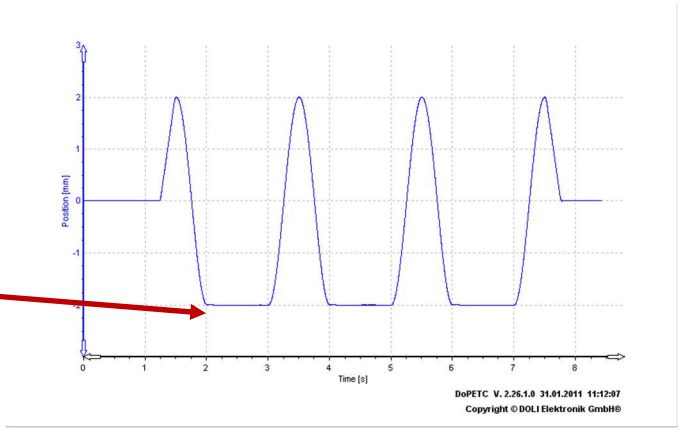
For modification of Dest1, Dest2, or Frequency of an active DoPECosine, resend the command with HalfCycles = ZERO and modified other parameters.

If Speed1 or Speed parameters are ZERO, the ramp to reach Dest1 or Destination will be automatically calculated.

5.4.3 DoPECosineX(Sync)

Cosine movement with halt times at peak and valley positions.





Go with 500mm/min to 2mm.
Start Cosine to -2mm, keep -2mm for 1 second.
Repeat cosine half cycle 6 times
Go with 500 mm/min to 0mm.

Function declaration		Description	Unit
extern unsigned DLLAPI	DoPECosineX (	Function returns Error constant (DoPERR_xxxx)	
DoPE_HANDLE	DoPEHdl	DoPE link handle	
unsigned short	MoveCtrl	Control mode (CTRL_xxx)	
double	Speed1	Speed to reach destination 1	Unit/s
double	Dest1	Destination 1	Unit
double	Halt1	Halt time at Destination 1	s
double	Dest2	Destination 2	Unit
double	Halt2	Halt time at Destination 2	s
double	Frequency	Cosine frequency	1/s
unsigned long	HalfCycles	Number of half cycles	
double	Speed	Speed to final destination	Unit/s
double	Destination	Final destination	Unit
WORD	*lpusTAN	Pointer to transaction number.	

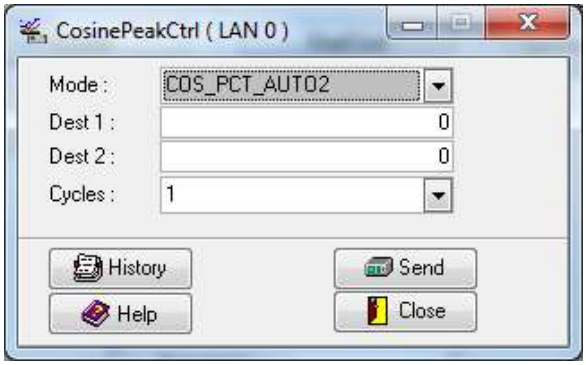
For modification of Dest1, Halt1, Dest2, Halt1, or Frequency of an active DoPECosineX, resend the command with HalfCycles = ZEROand modified other parameters.

If Speed1 or Speed parameters are ZERO, the ramp to reach Dest1 or Destination will be automatically calculated.

5.4.4 DoPECosinePeakCtrl (Sync)

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

Minimum requirements: DoPE 2.51



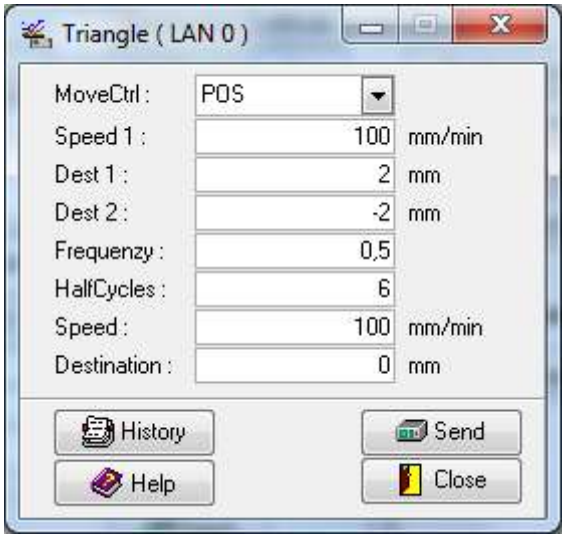
Function declaration		Description	Unit
extern unsigned DLLAPI	DoPECosinePeakCtrl (	Function returns Error constant (DoPERR_xxxx)	
DoPE_HANDLE	DoPEHdl	DoPE link handle	
unsigned short	Mode	Mode, see below	
double	Dest1	Corrected Destination 1 (only used for COS_PCT_AUTO1 andCOS_PCT_MANUAL modes)	
double	Dest2	Corrected Destination 1 (only used for COS_PCT_AUTO1 andCOS_PCT_MANUAL modes)	
unsigned short	Cycles	Peak control is active every xx cycles (0,1,2,4,8,16)	
WORD	*lpusTAN	Pointer to transaction number (not for Sync. version).	

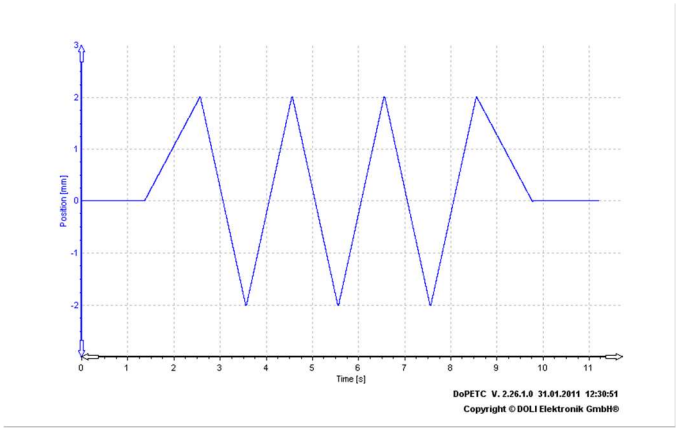
Modes for DoPECosinePeakCtrl:

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

5.4.5 DoPETriangle(Sync)

Triangular cyclic movement.





Go with 100mm/min to 2mm.

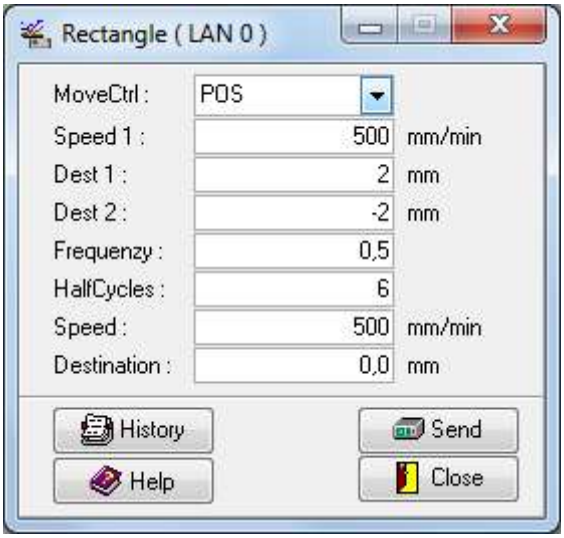
Start Triangle to -2mm. Repeat triangle half cycle 6 times.

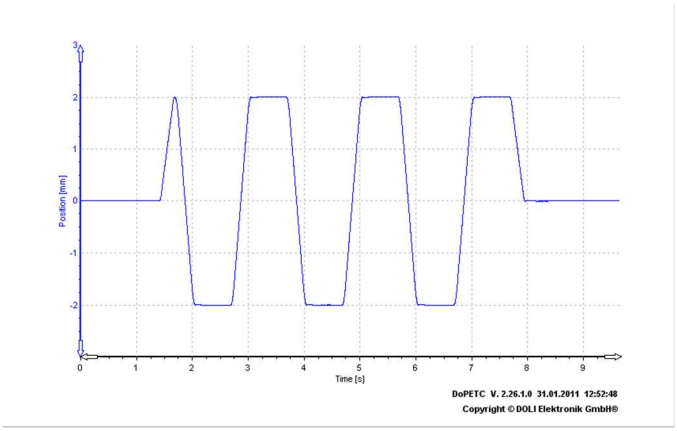
Go with 100 mm/min to 0mm.

Function declaration		Description	Unit
extern unsigned DLLAPI	DoPETriangle (	Function returns Error constant (DoPERR_xxxx)	
DoPE_HANDLE	DoPEHdl	DoPE link handle	
unsigned short	MoveCtrl	Control mode (CTRL_xxx)	
double	Speed1	Speed to reach destination 1	Unit/s
double	Dest1	Destination 1	Unit
double	Dest2	Destination 2	Unit
double	Frequency	Triangle frequency	1/s
unsigned long	HalfCycles	Number of half cycles	
double	Speed	Speed to final destination	Unit/s
double	Destination	Final destination	Unit
WORD	*lpusTAN	Pointer to transaction number.	

5.4.6 DoPERectangle(Sync)

Rectangular cyclic movement.





Go with 500mm/min to 2mm.

Start Rectangle to -2mm. Repeat rectangle half cycle 6 times.

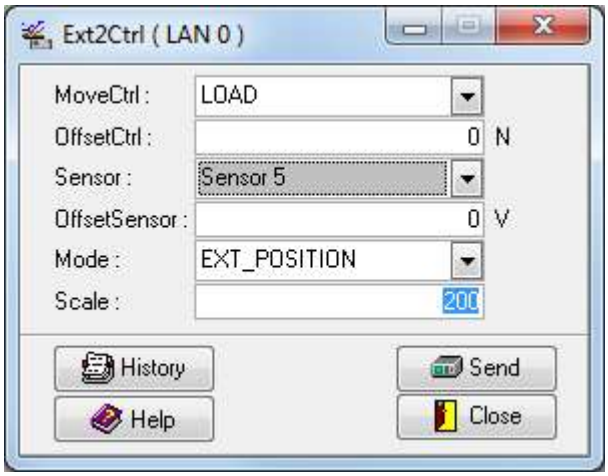
Go with 500 mm/min to 0mm.

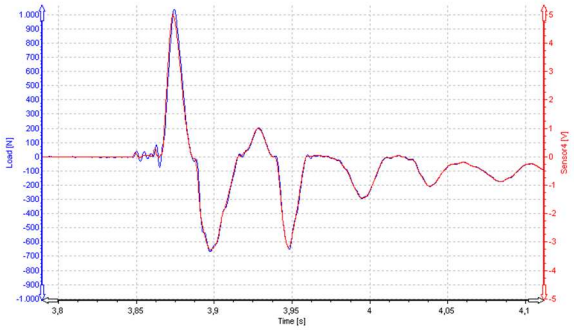
Function declaration		Description	Unit
extern unsigned DLLAPI	DoPERectangle(	Function returns Error constant (DoPERR_xxxx)	
DoPE_HANDLE	DoPEHdl	DoPE link handle	
unsigned short	MoveCtrl	Control mode (CTRL_xxx)	
double	Speed1	Speed to reach destination 1	Unit/s
double	Dest1	Destination 1	Unit
double	Dest2	Destination 2	Unit
double	Frequency	Rectangle frequency	1/s
unsigned long	HalfCycles	Number of half cycles	
double	Speed	Speed to final destination	Unit/s
double	Destination	Final destination	Unit
WORD	*lpusTAN	Pointer to transaction number.	

5.4.7 DoPEExt2Ctrl(Sync)

Move cross-head according to an external command signal. You may supply a random function to an 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:

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



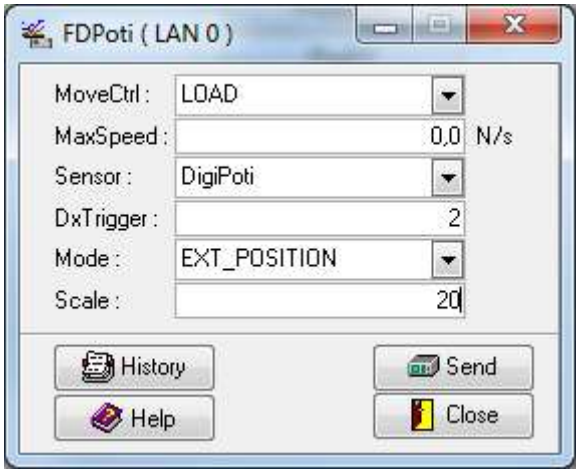


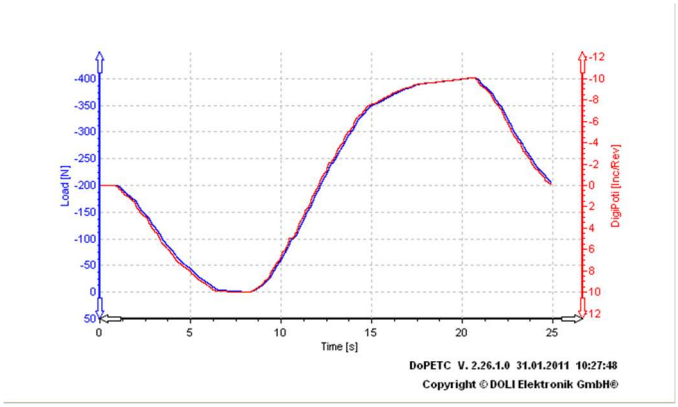
Follow external signal at Sensor4 in load control.
0V at Sensor4 will be 0N,
10V at Sensor4 will be 2000N.

Function declaration		Description	Unit
extern unsigned DLLAPI	DoPEExt2Ctrl (	Function returns Error constant (DoPERR_xxxx)	
DoPE_HANDLE	DoPEHdl	DoPE link handle	
unsigned short	MoveCtrl	Control mode (CTRL_xxx)	
double	OffsetCtrl	Offset for position, load or extension	Unit
unsigned short	SensorNo	Sensor number for the external command signal	
double	OffsetSensor	Offset for external command signal	
unsigned short	Mode	various position or speed control modes (see next page)	
double	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, 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	
WORD	*lpusTAN)	Pointer to transaction number.	

5.4.8 DoPEFDPoti (Sync)

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.





Follow DigiPoti signal in load control.
One revolution of the DigiPoti will cause 20N in load.

Function declaration		Description	Unit
extern unsigned DLLAPI	DoPEFDPoti (	Function returns Error constant (DoPERR_xxxx)	
DoPE_HANDLE	DoPEHdl	DoPE link handle	
unsigned short	MoveCtrl	Control mode (CTRL_xxx)	
double	MaxSpeed	Start speed for speed controlled modes	Unit/s
unsigned short	SensorNo	Sensor number for the external command input. Use sensor SENSOR_DP for Digital Encoder (DigiPoti) on EDC front panel.	
unsigned short	DxTrigger	Dead area of encoder. The Encoder has to change the specified number of digits before the command is active. For EDC front panel DigiPoti 2 or 3 is a good value.	
unsigned short	Mode	various position or speed control modes (see below)	
double	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.	
WORD	*lpusTAN	Pointer to transaction number.	

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

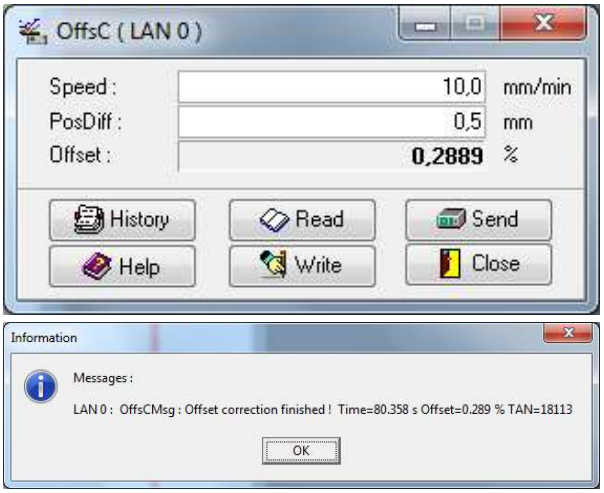
The next four modes refer to movement UP/DOWN. E.g. EXT_SPEED_UP moves Cross-head UP. The relation between measured values and movement Up/Down is defined in the set-up data.

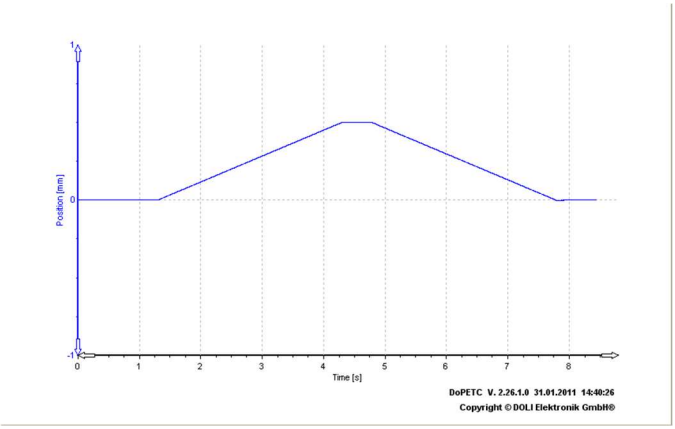
EXT_POS_UP_DOWN	4	Position Up / Down
EXT_SPEED_UP_DOWN	5	Speed bipolar (Up and Down)
EXT_SPEED_UP	6	Speed UP
EXT_SPEED_DOWN	7	Speed DOWN
EXT_POS_OPENLOOP	8	Position mode for openloop configurations.

For use with DoPEFDPoti command we recommend only the following modes:
EXT_POS_UP_DOWN, EXT_SPEED_UP, EXT_SPEED_DOWN, EXT_POS_OPENLOOP

5.4.9 DoPEOffsC(Sync)

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.





Move 0.5 mm up and down and calculate the offset.
The offset is transmitted to PC (see DoPEOnOffsCMsgHdlr)

Function declaration		Description	Unit
extern unsigned DLLAPI	DoPEOffsC (	Function returns Error constant (DoPERR_xxxx)	
DoPE_HANDLE	DoPEHdl	DoPE link handle	
double	Speed	Speed	Unit/s
double	PosDiff	Distance to move cross-head	Unit
WORD	*lpusTAN	Pointer to transaction number.	

5.5 PC Command

With the PC Command a digital command signal can be used to move the cross-head.

5.5.1 DoPEPcCmd

Move cross-head according to a digital command signal.
Scaling of the command- and sensor-system is done by offsets and scale. Following formula is used:

$$Output = (Value - Offset) \bullet Scale + OffsetCtrl$$

Minimum requirements: EDC580V/220V, EdcApp 9140.008, DoPE 2.79

Function declaration		Description	Unit
extern unsigned DLLAPI DoPEPcCmd (		Function returns Error constant (DoPERR_xxxx)	
DoPE_HANDLE	DoPEHdl,	DoPE link handle	
unsigned short	ExecutionMode,	(see description below)	
unsigned short	MoveCtrl,	Control mode for movement (Position, Load, Extension)	
unsigned short	CmdMode,	Interpolation mode between the values **DoPE_PC_CMD_MODE_POS:** Command data are position values for each control loop cycle. **DoPE_PC_CMD_MODE_POS_LINEAR:** Command data are time and position. A linear function is interpolated between the position values **DoPE_PC_CMD_MODE_POS_COSINUS:** Command data are time and position. A cosine function is interpolated between the position values	
double	SpeedToStart,	Speed to StartPosition	Unit/s
double	Offset,	Offset for stimulation signal	stimUnit
double	Scale,	Scaling factor for stimulation signal	
double	OffsetCtrl,	Offset for move control channel	Unit
double	FadeInTime,	Fade in time	s
double	FadeOutTime,	Fade out time	s
unsigned	Cycles,	Number of cycles (ignored in append mode)	
unsigned	Count,	Number of PcCmdData items in Data array	
DoPEPcCmdData	*Data,	Pointer to PcCmdData array In DoPE_PC_CMD_MODE_POS mode the time values will be ignored	
unsigned short	*lpusTAN);	Pointer to transaction number.	
typedef struct { double Time; double StimulationSignal; } DoPEPcCmdData;			s stimUnit

DoPEPcCmd can be used in two different ways. A single call can include all information to perform the movement command. In this case ExecutionMode DoPE_PC_CMD_EXECUTE_LAST must be used and all parameters are considered. Appending data to the running movement command isn't possible.

If the Data can't be stored on the EDC the command will be discarded and the return code is set to DoPERR_PARAMETER.

In the second case ExecutionMode is set to DoPE_PC_CMD_EXECUTE_APPEND in the first call. The movement command starts emedeatly after the transmission of the data. Now any number of calls can follow to append data to the running movement command (DoPE_PC_CMD_EXECUTE_APPEND). With the last call ExecutionMode must be set to DoPE_PC_CMD_EXECUTE_LAST. No more data can be appended.

In the append mode buffer overruns and underruns can occur. At a buffer overrun PC Command data is providet to fast and can't be buffered in the EDC. In this case the data block will be discarded and the return code is set to DoPERR_PARAMETER.

At a buffer underrun PC Command data isn't provided fast enough. In this case the last point of the latest PC Command data block is used as stimulation signal until a new data block is available.

The number of buffer underruns can be retrieved with DoPERdPcCmdInfo.

Parameter	First call	Following calls	Last call
ExecutionMode	**EXECUTE_APPEND**	**EXECUTE_APPEND**	**EXECUTE_LAST**
MoveCtrl	**valid**	ignored	ignored
SpeedToStart	**valid**	ignored	ignored
CmdMode, Offset, Scale, OffsetCtrl, Count, Data	**valid**	**valid**	**valid**
FadeInTime	**valid**	ignored	ignored
FadeOutTime	ignored	ignored	**valid**
Cycles	ignored	ignored	**valid**
IpusTAN	**valid**	ignored	ignored

To interrupt, continue or terminate a running movement command the ExecutionModes DoPE_PC_CMD_EXECUTE_PAUSE, DoPE_PC_CMD_EXECUTE_CONTINUE and DoPE_PC_CMD_EXECUTE_TERMINATE can be used.

ExecutionMode	EXECUTE_PAUSE	EXECUTE_CONTINUE	EXECUTE_TERMINATE
MoveCtrl	ignored	ignored	ignored
SpeedToStart	ignored	ignored	ignored
CmdMode, Offset, Scale, OffsetCtrl, Count, Data	ignored	ignored	ignored
FadeInTime	ignored	**valid**	ignored
FadeOutTime	**valid**	ignored	**valid**
Cycles	ignored	ignored	**ignored**
IpusTAN	ignored	ignored	ignored

OffsetCtrl will be used as start/end point of the movement command if fade in/out will be used. Otherwise the first/last Data point will be used.

The Cycles count of the measuring data record is set to zero at the PcCmd call. Cycles count will be incremented with the execution of the first point of each PC Command data block.

For a detailed description of the SpeedToStart, Scale and Offset parameters and the fade times, please refer to the DoPEPcCmdFromFile dokumentation.

5.5.2 DoPEPcCmdFromFile

Move cross-head according to a digital command signal stored in a file.
DoPEPcCmdFromFile is similar to DoPEPcCmd except the stimulation data is passed thru a file and no data can be appended. So, all parameters will be considered.
Scaling of the command- and sensor-system is done by offsets and scale. Following formula is used:

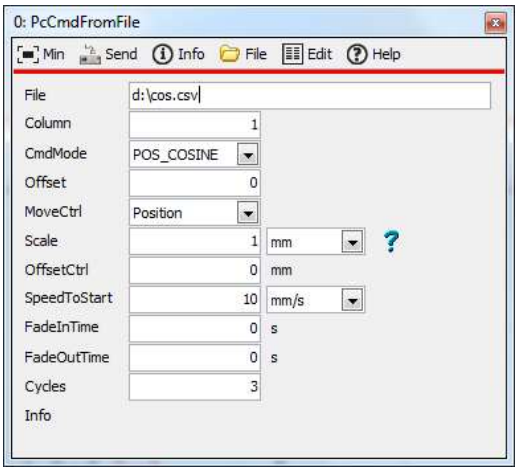
$$Output = (Value - Offset) \bullet Scale + OffsetCtrl$$

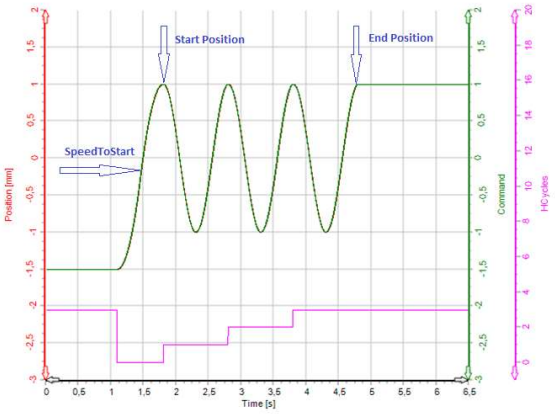
Minimum requirements: EDC580V/220V, EdcApp 9140.008, DoPE 2.79

Function declaration			Description	Unit
extern unsigned DLLAPI	DoPEPcCmdFromFile (		Function returns Error constant (DoPERR_xxxx)	
	DoPE_HANDLE	DoPEHdl,	DoPE link handle	
	unsigned short	MoveCtrl,	Control mode for movement (Position, Load, Extension)	
	unsigned short	CmdMode,	Interpolation mode between the values **DoPE_PC_CMD_MODE_POS:** Command data are position values for each control loop cycle. **DoPE_PC_CMD_MODE_POS_LINEAR:** Command data are time and position. A linear function is interpolated between the position values **DoPE_PC_CMD_MODE_POS_COSINUS:** Command data are time and position. A cosine function is interpolated between the position values	
	double	SpeedToStart,	Speed to StartPosition	Unit/s
	double	Offset,	Offset for stimulation signal	stimUnit
	double	Scale,	Scaling factor for stimulation signal	
	double	OffsetCtrl,	Offset for move control channel	Unit
	double	FadeInTime,	Fade in time	s
	double	FadeOutTime,	Fade out time	s
	unsigned	Cycles,	Number of cycles	
	unsigned	Column,	Column number of Stimulation Signal In interpolation modes Time has always column number 0	
	char	*FileName,	Pointer to file name and path. Supported file types: csv "Comma Separated Values" text file: Value separator must be ';' Decimal point can be '.' or ',' Lines NOT starting with a number or a separator will be skipped bmv "DOLI Binary Measured Values" file: generated by DoPETestCenter version 2.27.1 (or above) In DoPE_PC_CMD_MODE_POS the time values will be ignored File access errors (e.g. wrong file extension) leads to an DoPERR_OPEN error	
	unsigned short	*lpusTAN);	Pointer to transaction number.	

To illustrate the various parameters we use a simple cos.csv file with cosinus interpolation and three cycles.

Time;	Position
Sec;	mm
0;	1
0.5;	-1
1.0;	1

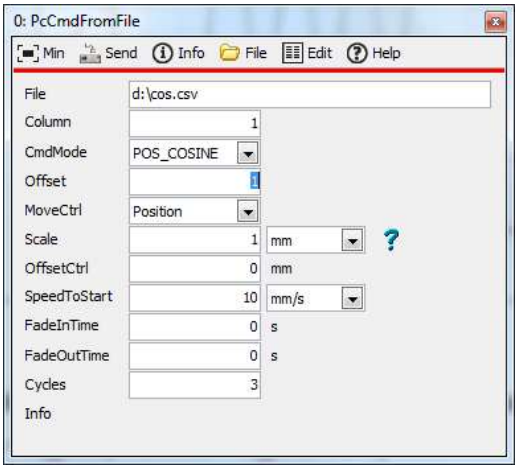


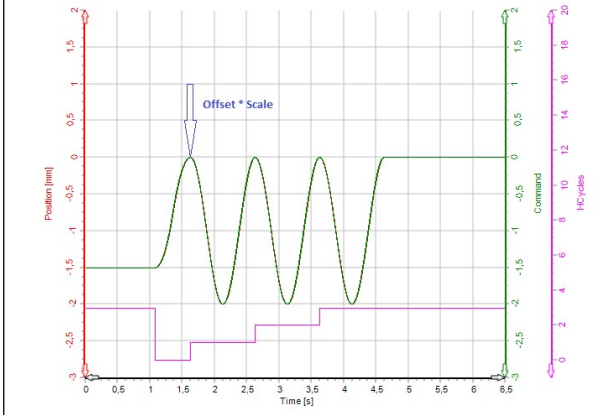


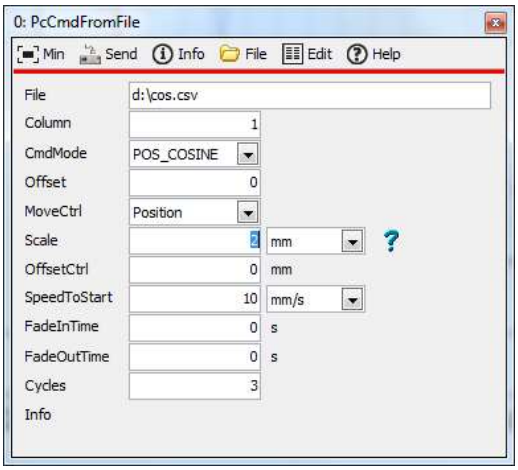
5.5.2.1 Scale and Offsets

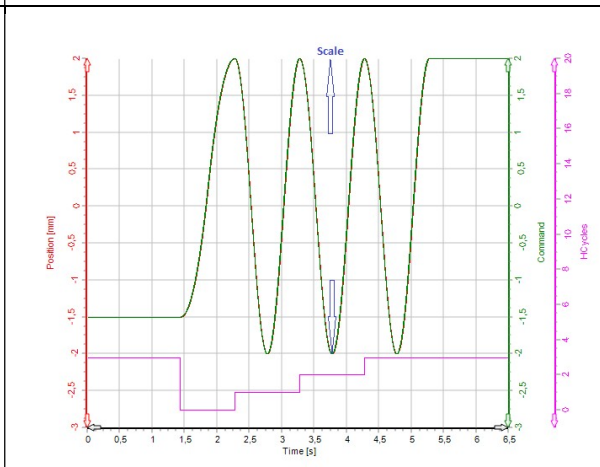
Scaling of the command- and sensor-system is done by offsets and scale. Following formula is used:

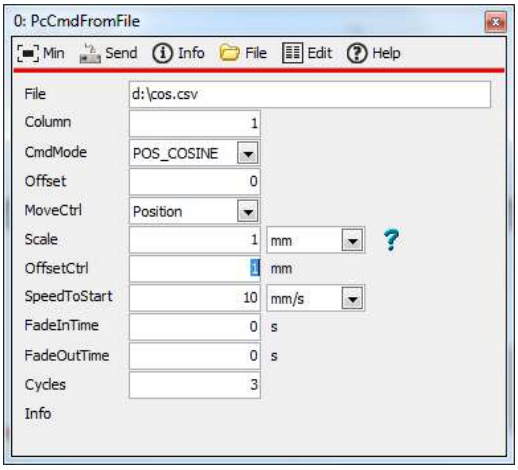
$$Output = (Value - Offset) \bullet Scale + OffsetCtrl$$

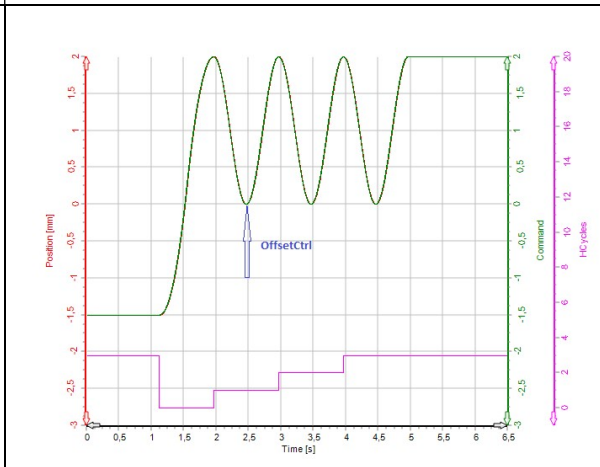












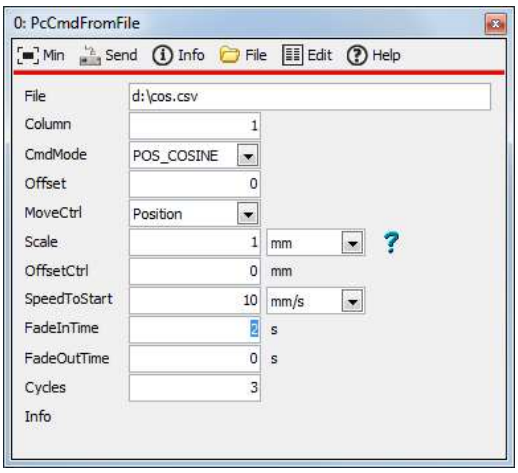
To convert e.g. temperature from Fahrenheit to Celsius according the formula °C = (°F-30)/2 Offset should be 30 and Scale 0.5.

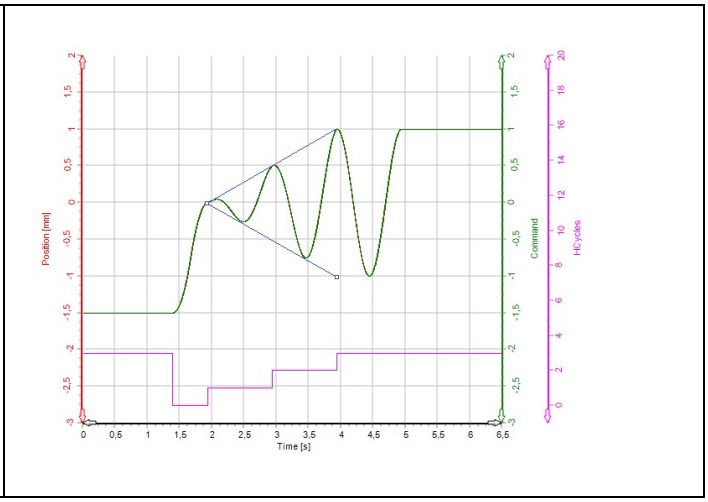
OffsetCtrl specifies the start/end position for fade in/out.

5.5.2.2 "Start/End Position", SpeedToStart and Fade In/Out

If fade in/out time are zero the first point of the PcCmd data is used as the start/end position.
Otherwise OffsetCtrl is used as the start/end position.
During the Fade In time the following formula is used:

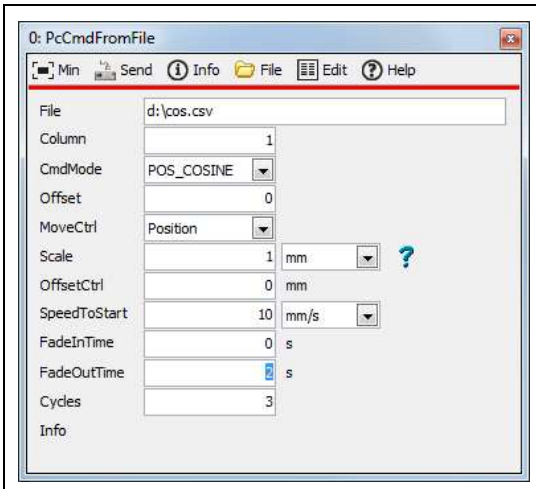
$$Output(t) = ((Value(t) - Offset) \bullet Scale + OffsetCtrl) \bullet (t - t0) \div FadeInTime$$

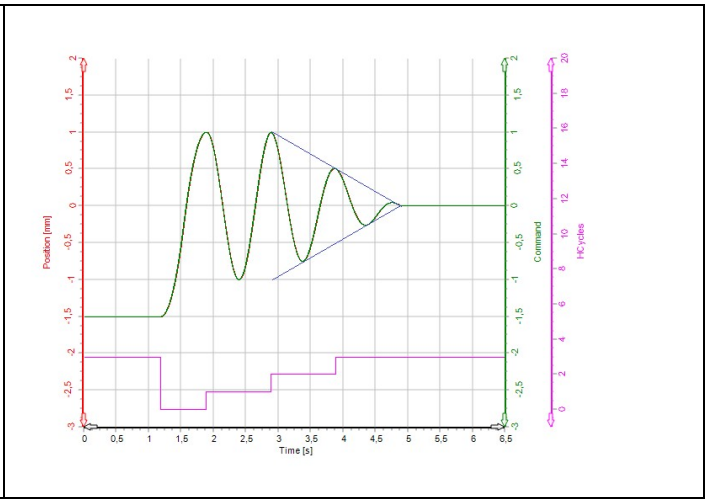




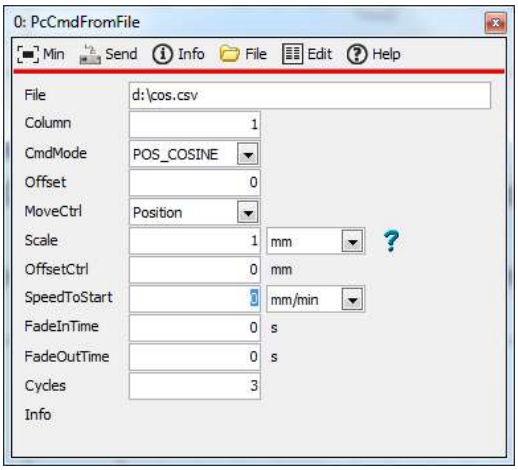
During the Fade Out time the following formula is used:

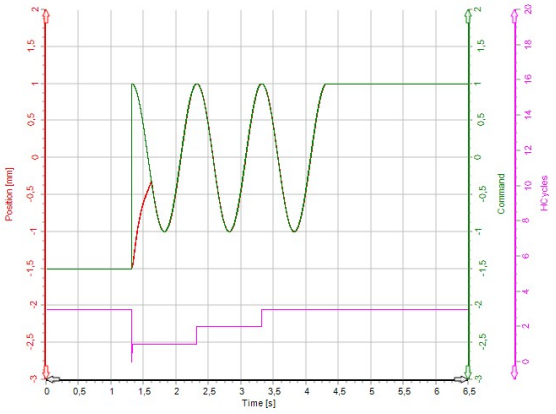
$$Output(t) = ((Value(t) - Offset) \bullet Scale + OffsetCtrl) \bullet (tend - t) \div FadeOutTime$$



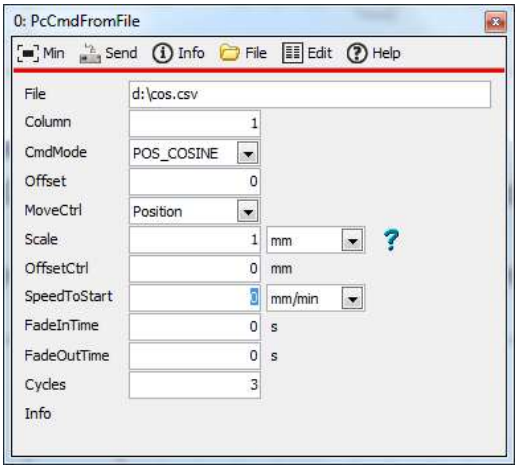


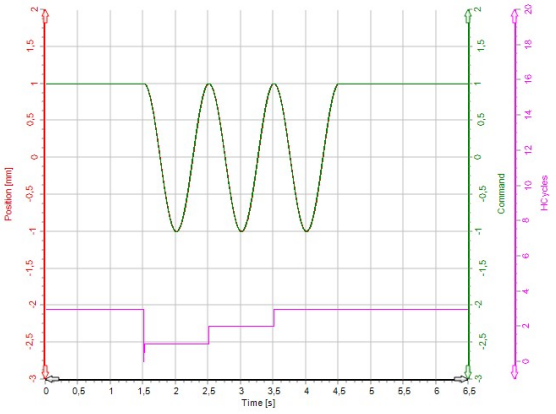
If SpeedToStart is set to zero the command of the closed loop controller 'jumps' to the start position within one closed loop controller cycle. This is a feature used for synchronized EDCs.



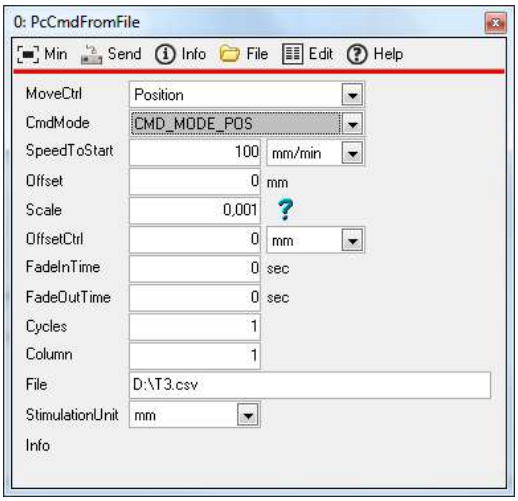


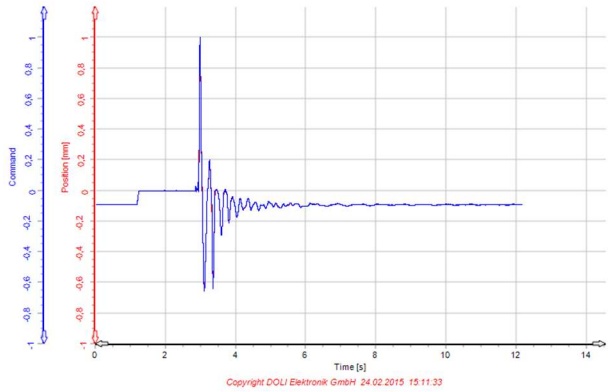
Obviously the start position should be set by a separate movement command.





5.5.2.3 Random Values



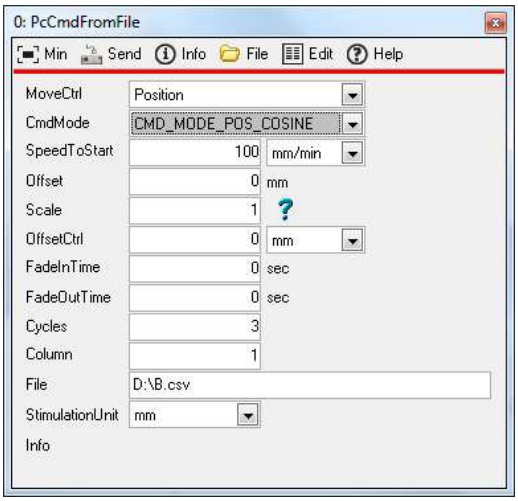


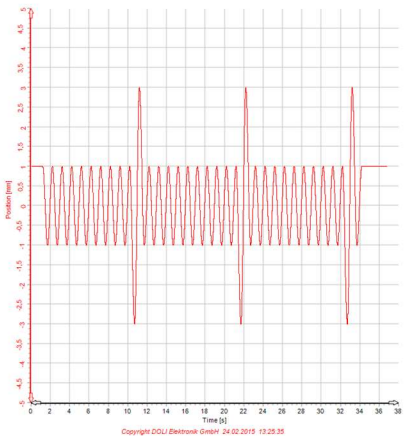
Follow digital command from PC in position control. Don't use inperpolation between the points in the file (ignore time column).
Use scale 0.001 to convert µm to mm.

Sample of a csv file:

Time;	Position
Sec;	µm
3.819;	-0,8092685
3.8192;	-0,6743904
3.8194;	-0,6069514
3.8196;	-0,5395123
3.8198;	-0,6069514
3.820;	-0,6743904
3.8202;	-0,6743904
3.8204;	-0,7418295
3.8206;	-0,8767075
3.8208;	-1,011586
3.821;	-1,146464
3.8212;	-1,281342
3.8214;	-1,146464
3.8216;	-1,011586
3.8218;	-0,8767075
3.822;	-0,8092685
…	…

5.5.2.4 Cosine with one Cycle big Amplitude

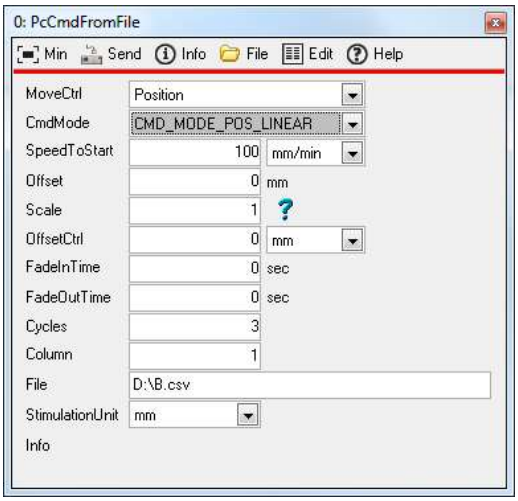


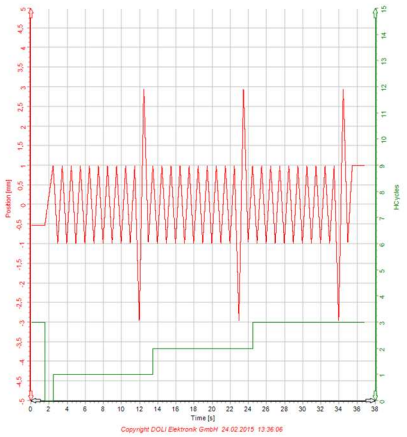


Follow digital command from PC in position control. Use a **cosine** interpolation between the points in the file.
Sample of a csv file:

Time;	Position
Sec;	mm
0;	1.0
0.5;	-1.0
1.0;	1.0
1.5;	-1.0
2.0;	1.0
2.5;	-1.0
3.0;	1.0
3.5;	-1.0
4.0;	1.0
4.5;	-1.0
5.0;	1.0
5.5;	-1.0
6.0;	1.0
6.5;	-1.0
7.0;	1.0
7.5;	-1.0
8.0;	1.0
8.5;	-1.0
9.0;	1.0
9.5;	**-3.0**
10.0;	**3.0**
10.5;	-1.0
11.0;	1.0

5.5.2.5 Triangle with one Cycle big Amplitude





Follow digital command from PC in position control. Use a **linear** interpolation between the points in the file.

At the beginning of the command, the HCycle counter is set to zero. At the beginning of each cycle, the counter is incremented.

Sample of a csv file:

Time;	Position
Sec;	mm
0;	1.0
0.5;	-1.0
1.0;	1.0
1.5;	-1.0
2.0;	1.0
2.5;	-1.0
3.0;	1.0
3.5;	-1.0
4.0;	1.0
4.5;	-1.0
5.0;	1.0
5.5;	-1.0
6.0;	1.0
6.5;	-1.0
7.0;	1.0
7.5;	-1.0
8.0;	1.0
8.5;	-1.0
9.0;	1.0
9.5;	**-3.0**
10.0;	**3.0**
10.5;	-1.0
11.0;	1.0

5.5.3 DoPERdPcCmdInfo

Read PC Command buffer and status information.		
Minimum requirements: EDC580V/220V, EdcApp 9140.008, DoPE 2.79		
Function declaration	**Description**	**Unit**
extern unsigned DLLAPI DoPERdPcCmdInfo (DoPE_HANDLE DoPEHdl, DoPEPcCmdInfo *Info);	Function returns Error constant (DoPERR_xxxx) DoPE link handle Pointer for DoPEPcCmdInfo structure.	
typedef struct { unsigned BufItemsFree unsigned BufItemsUsed; unsigned BufferUnderrun; double PosCtrlTime; unsigned Paused; } DoPEPcCmdInfo;	 number of available PcCmdData items number of used PcCmdData items number of buffer underruns Position controller cycle time PC Command execution paused	 s

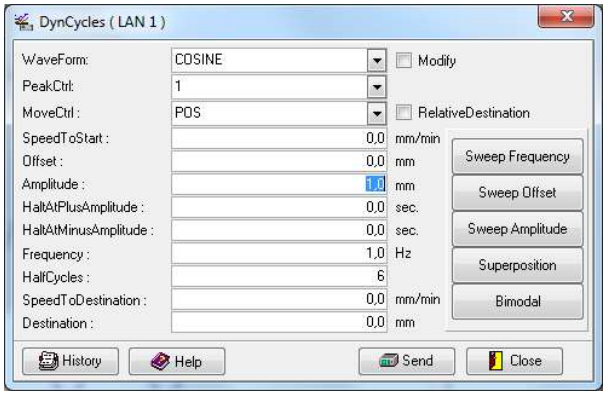
5.6 DoPEDynCycles

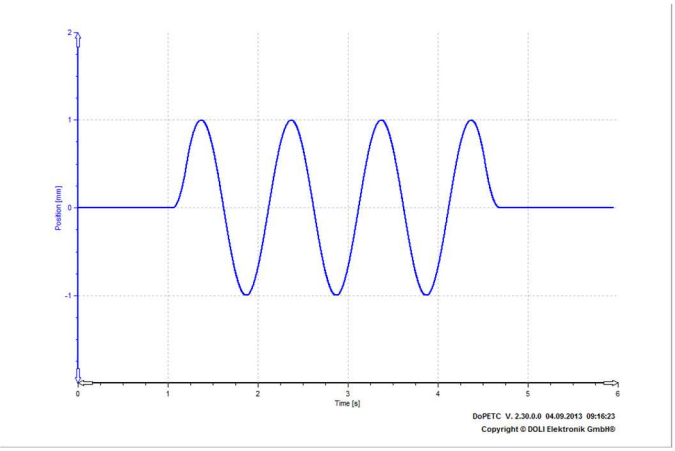
The DoPEDynCycles is a powerful command for mostly needed dynamic cycles:
Possible waveforms: **Cosine**, **Triangle**, **Rectangle**, **Saw tooth**, **inverted Saw tooth**, and **Pulse**.
Peak/Valley control may be activated for all of these waveforms.
Linear or logarithmic sweeps for frequency, offset, or amplitude may be activated, even any combination e.g. frequency and amplitude sweep.
All waveforms may be superimposed with a cosine wave.
Different Bi-Modal modes like perform a test in e.g. position control, and keep load peak/valley constant are also available.

Minimum requirements: EDC580/220, EdcApp 9133.013, DoPE 2.63
The wave forms Triangle, Rectangle, Saw tooth, inverted Saw tooth, and Pulse are only implemented for EDC580V/220V in EdcApp 9140.005 and later.

5.6.1 Basic Waveforms

5.6.1.1 Cosine

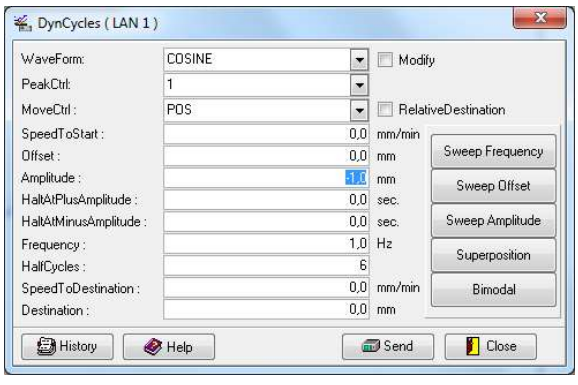


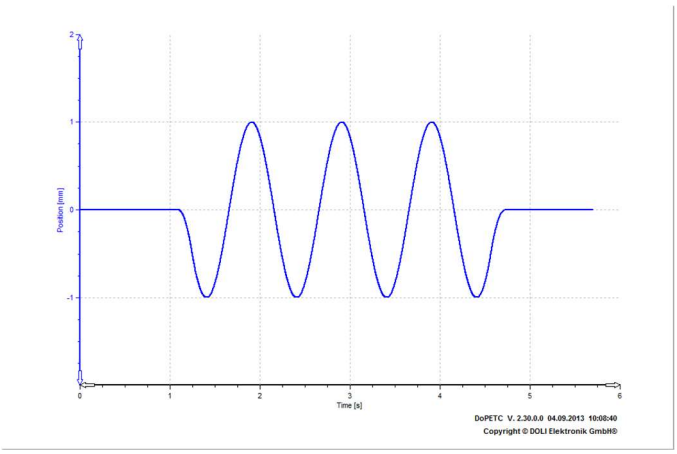


Cycle with a Cosine waveform at an offset of 0mm with amplitude of **+1mm** with 1Hz.
Do this for 6 half-cycles (3 cosine periods). SpeedToStart and SpeedToDestination are specified with 0mm/min.
In this case; the ramp to the starting point (1mm) and destination (0mm) is automatically calculated.

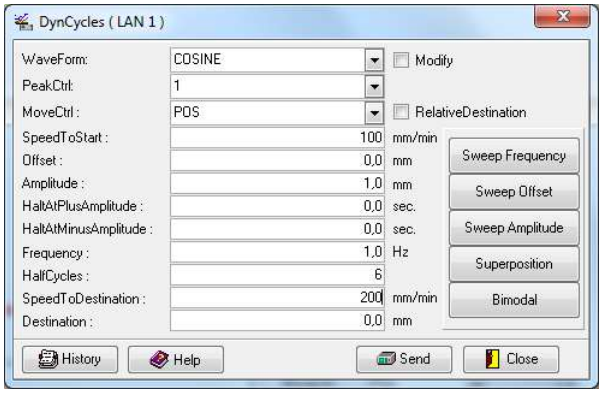
The automatic calculation of the ramp doesn't work If the DynCyle command interrupts another running movement command!

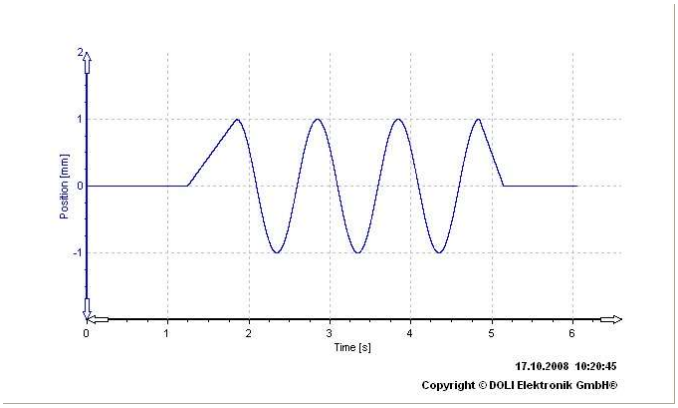
For very low frequencies the minum acceleration is used for the automatic ramp calculation. This leads to a ramp with constant speed between the acceleration and deceleration phase. The lowest possible frequency without a constant speed ramp depends on the requeseted frequency, the amplitude and the resolution of the measured signal.





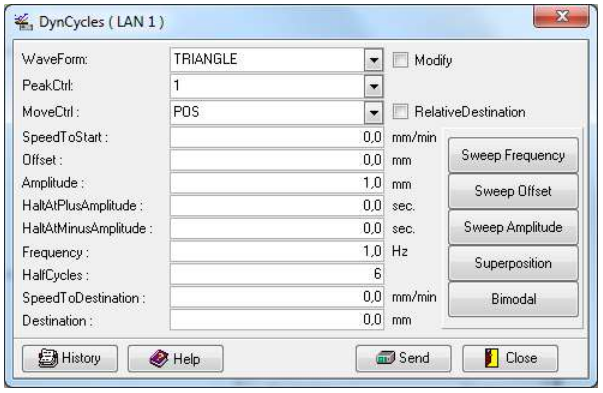
Same as above but with amplitude of **-1mm**.

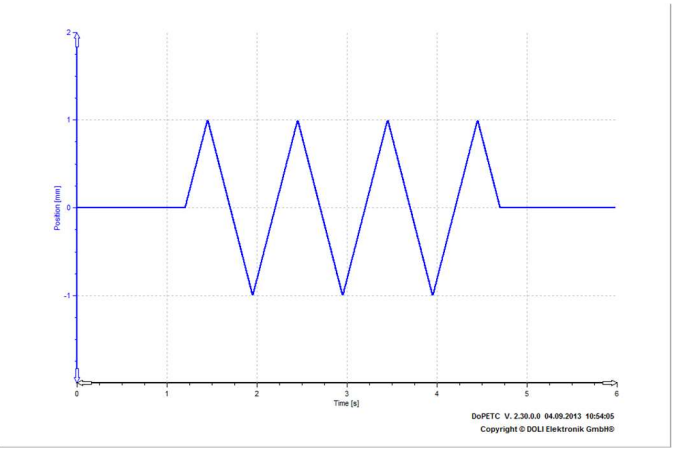




Same as above but SpeedToStart and SpeedToDestination specified.

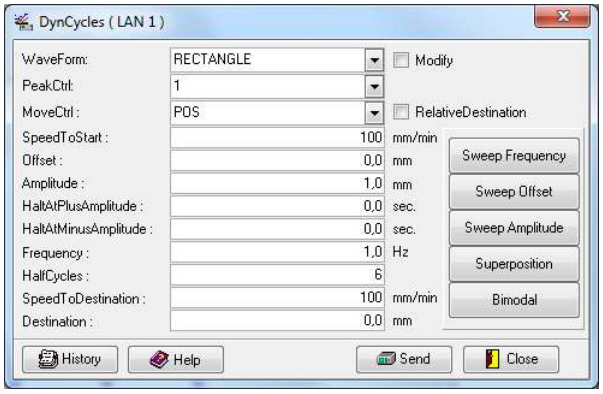
5.6.1.2 Triangle

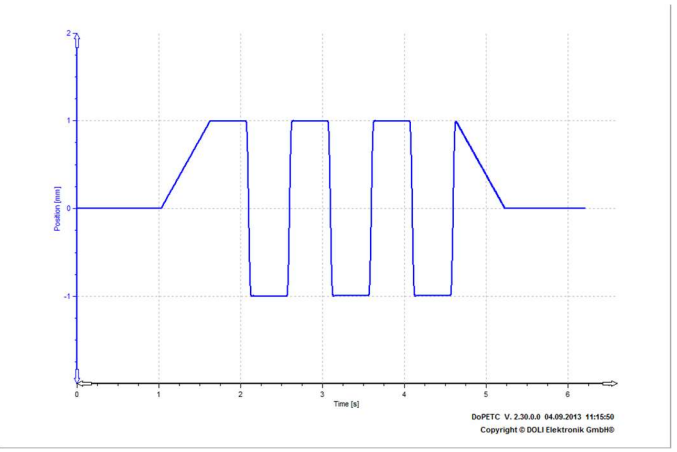




Same as above but with Triangle wafevorm.

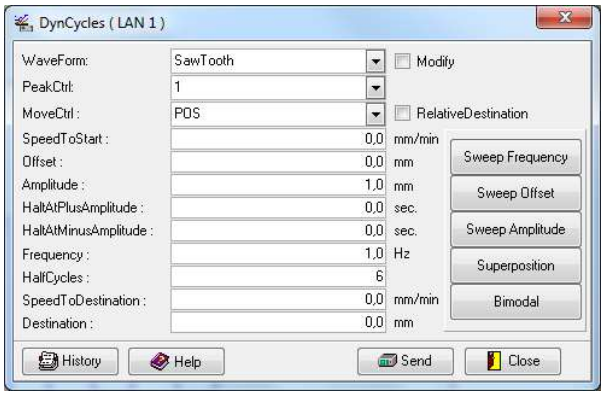
5.6.1.3 Rectangle

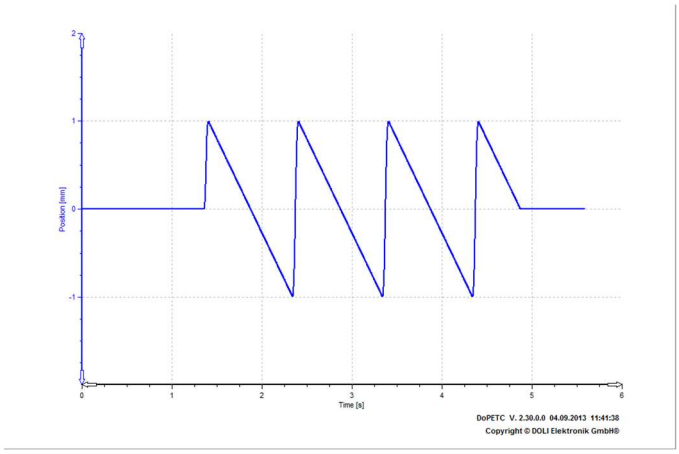




Same as above but with Rectangle wafevorm.

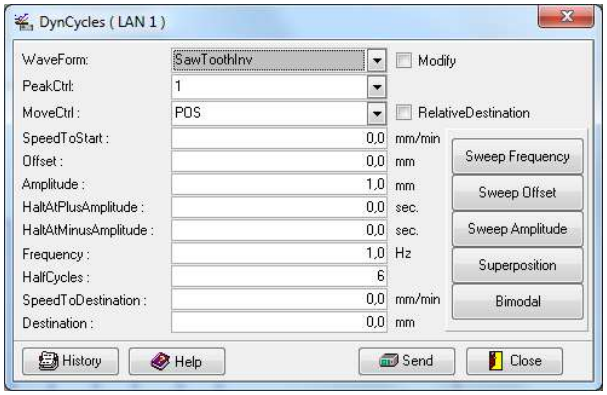
5.6.1.4 Sawtooth

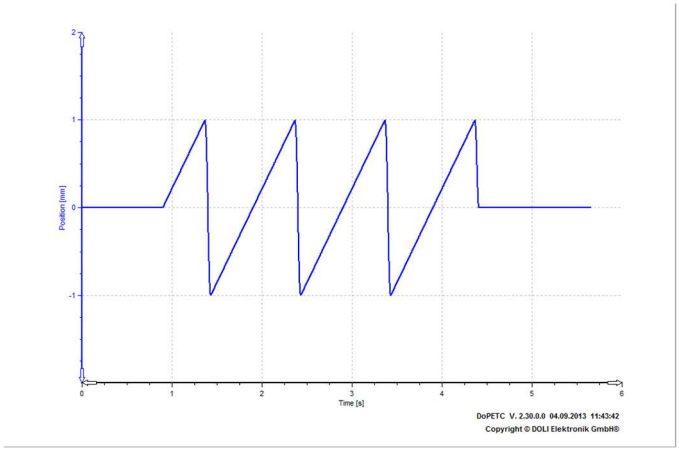




Same as above but with Sawtooth wafevorm.

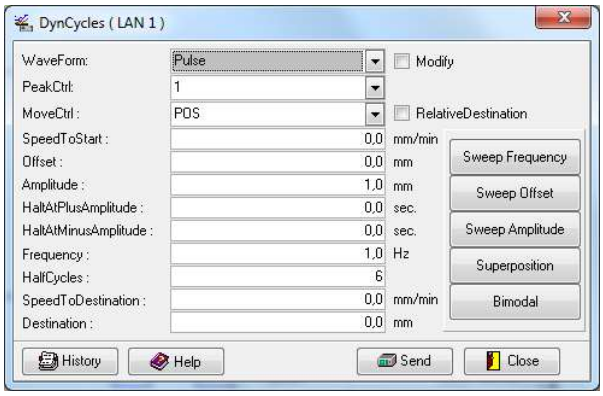
5.6.1.5 Sawtooth inverse

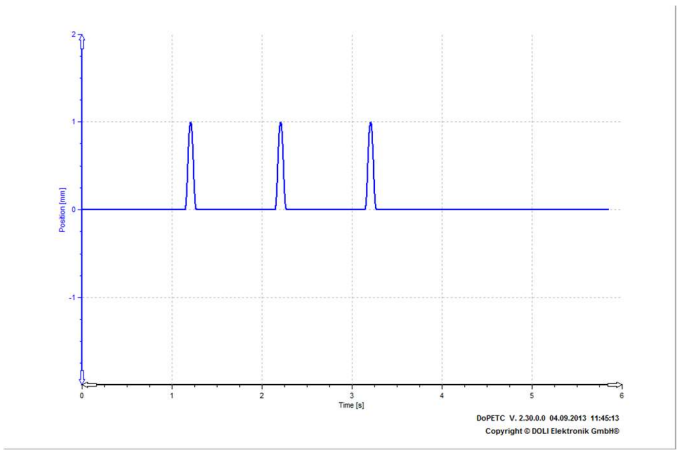




Same as above but with an inverted Sawtooth wafevorm.

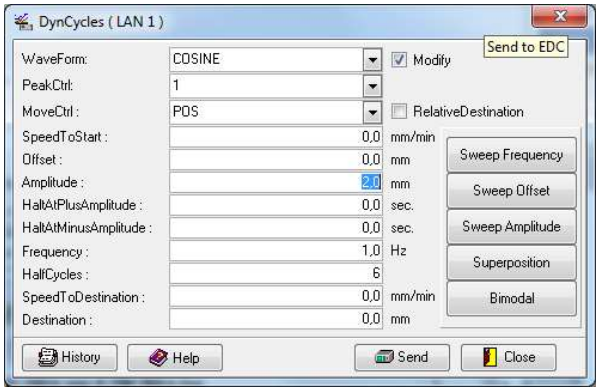
5.6.1.6 Pulse

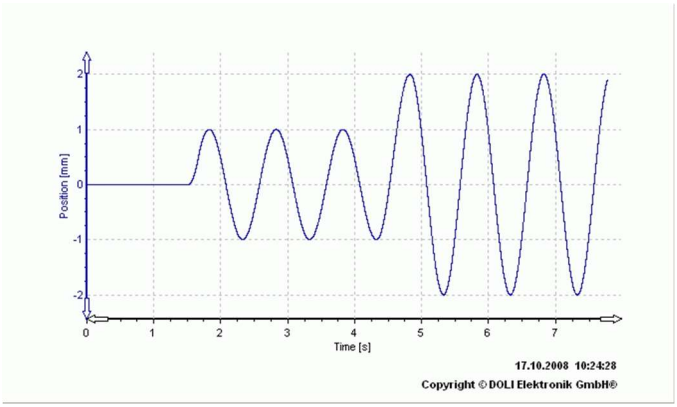




Same as above but with Pulse wafevorm.

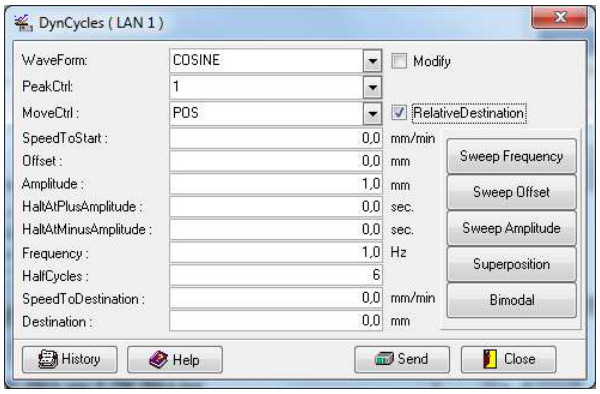
5.6.2 Modify Parameter of an active test

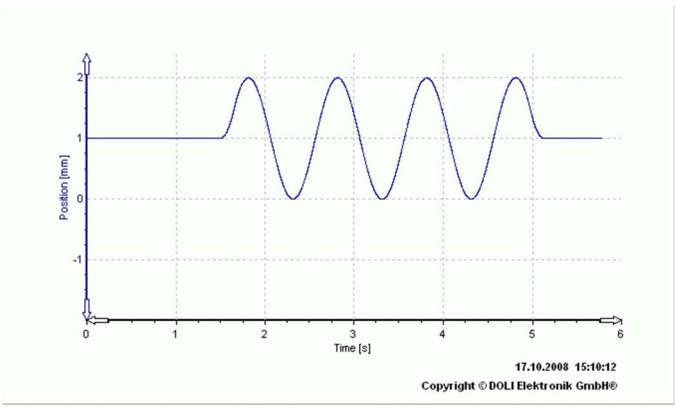




Modify amplitude of an active Cosine.
Cosine was started with 1mm amplitude. After some cycles, DynCycle command was sent again, with the Modify flag set, and amplitude of 2mm.
Besides amplitude, offset, and frequency may also be modified.

5.6.3 Relative Destinations

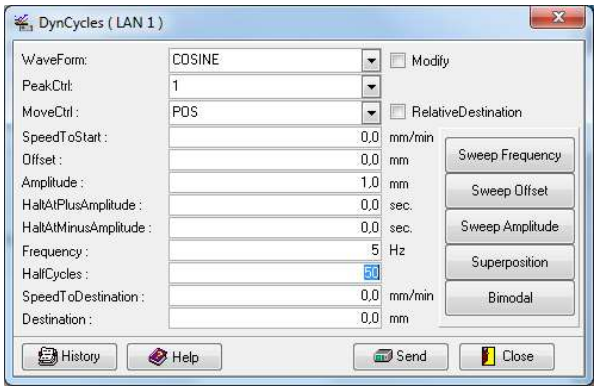


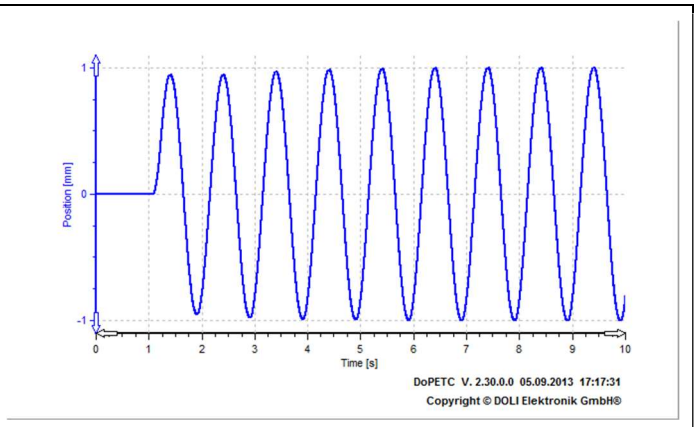


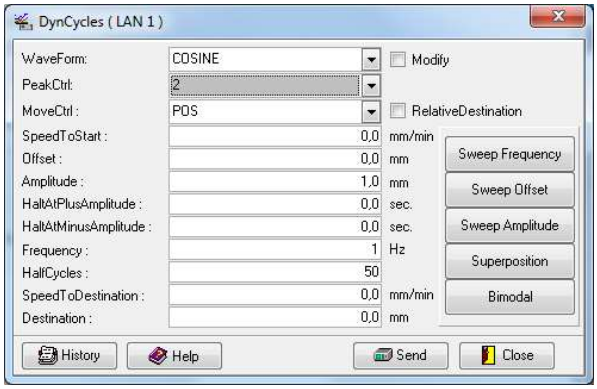
Cosine command with relative Destinations.
Cosine was started at a position of 1mm. This position will be added to the offset, and Destination.

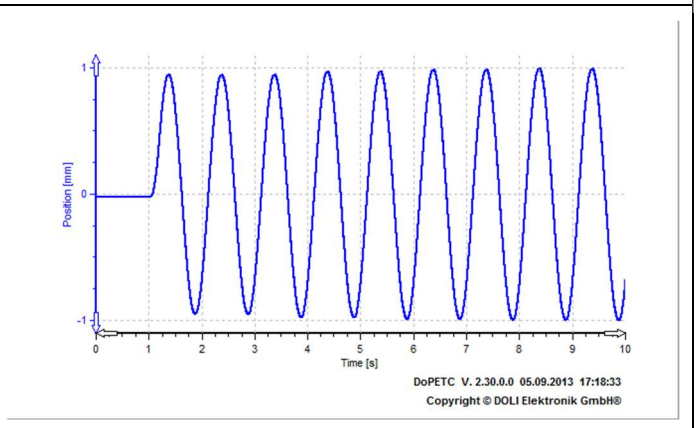
5.6.4 Peak and Valley control

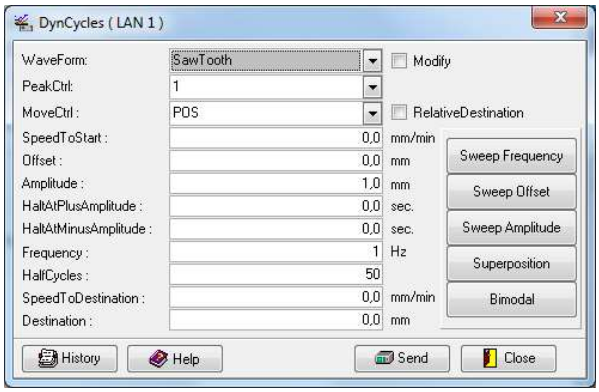
Peak (and Valley) Control can be activated for all Waveform by choosing any value for PeakCtrl (1, 2, 4, 8, 16).
PeakCtrl = 0 will deactivate peak conctol.
PeakCtrl = 1 will control the error at peak and valley points using the error of each cycle.
PeakCtrl = 2 will control the error every second peak and valley points, using the average error of two cycles.
PeakCtrl = 16 will control the error every 16 peak and valley points, using the average error of 16 cycles.
Below different samples of peak control:

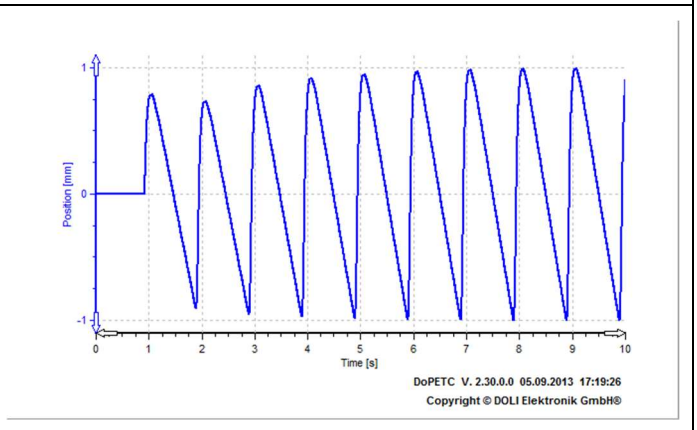


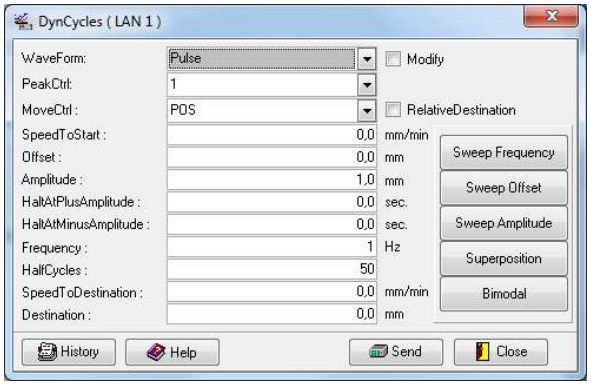


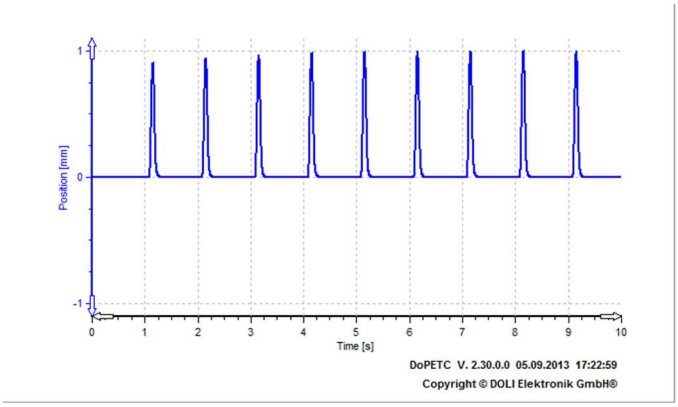






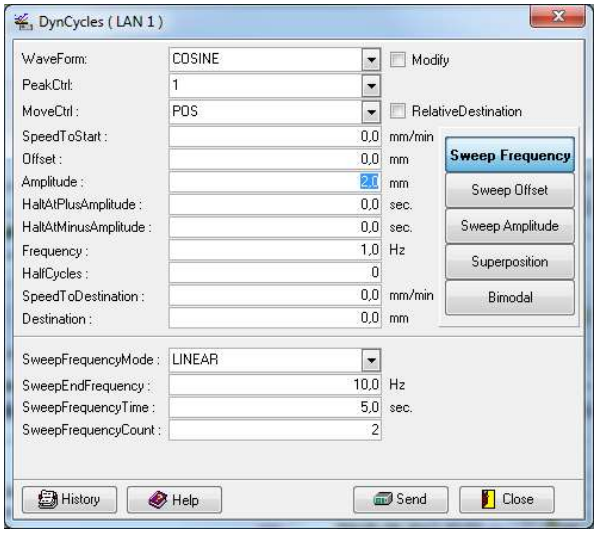


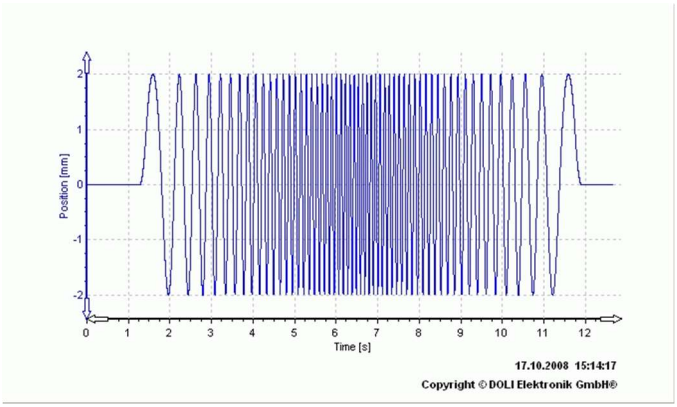

DynCycles (LAN 1)
WaveForm:
Pulse
Modify
PeakCtrl:
1
MoveCtrl :
POS
RelativeDestination
SpeedToStart :
0,0
mm/min
Offset :
0,0
mm
Amplitude :
1,0
mm
HaltAtPlusAmplitude :
0,0
sec.
HaltAtMinusAmplitude :
0,0
sec.
Frequency :
1
Hz
HalfCycles :
50
SpeedToDestination :
0,0
mm/min
Destination :
0,0
mm
Sweep Frequency
Sweep Offset
Sweep Amplitude
Superposition
Bimodal
History
Help
Send
Close


Position [mm]
Time [s]
DoPETC V. 2.30.0.0 05.09.2013 17:22:59
Copyright © DOLI Elektronik GmbH®

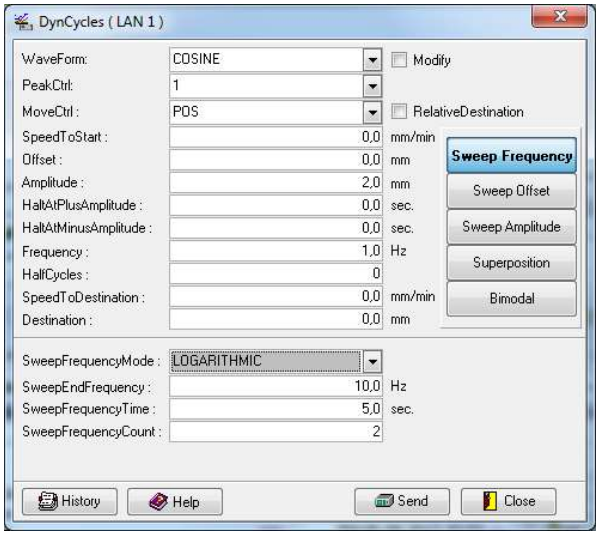
.

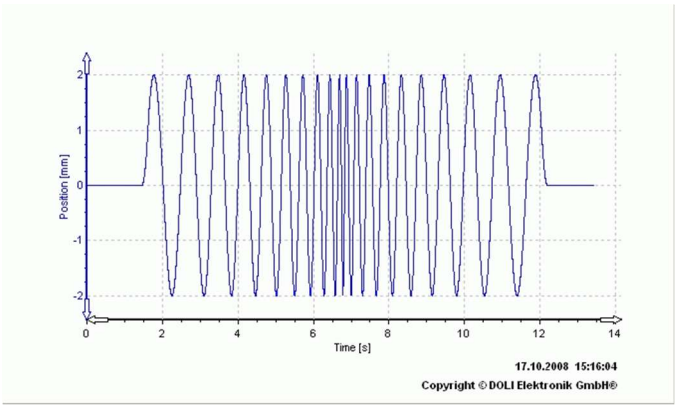
5.6.5 Sweeps



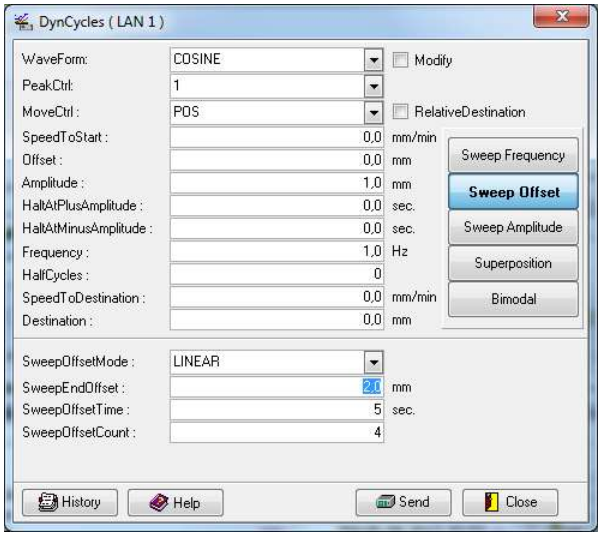


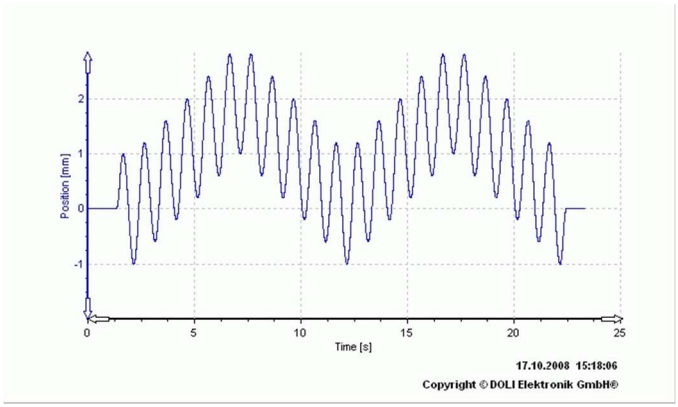
Cosine command with a linear frequency sweep.
Cosine starts with a frequency of 1Hz. The frequency will be continuously increased to 10Hz within 5 seconds. The parameter SweepFrequencyCount specifies how many sweeps should be done. In this case go up to 10Hz, and back to 1Hz.



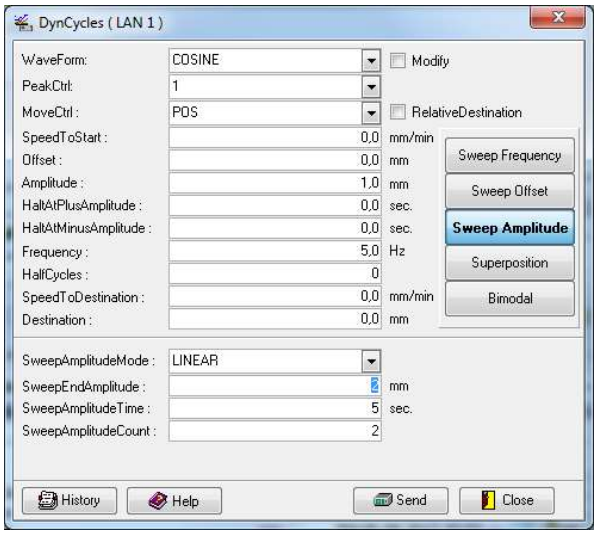


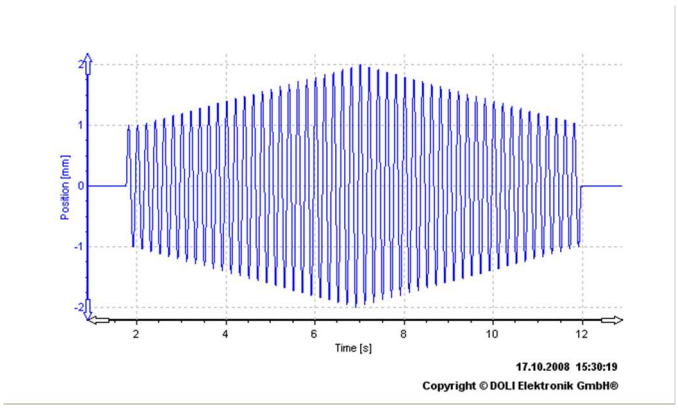
Cosine command with a logarithmic frequency sweep.
Cosine starts with a frequency of 1Hz. The frequency will be exponentially increased to 10Hz within 5 seconds. The parameter SweepFrequencyCount specifies how many sweeps should be done. In this case go up to 10Hz, and back to 1Hz.



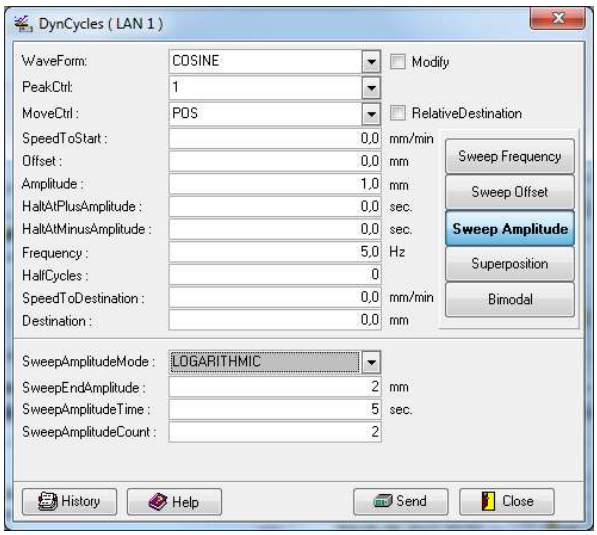


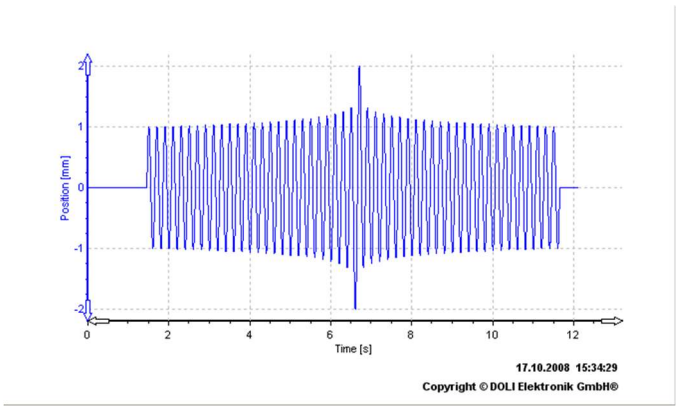
Cosine command with a linear offset sweep.
Cosine starts with an offset of 0mm. The offset is linearly increased to 2mm within 5 seconds. The parameter SweepOffsetCount specifies how many sweeps should be done.



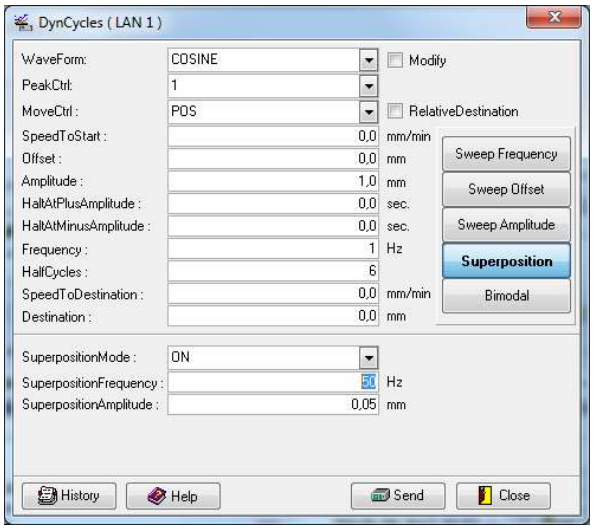


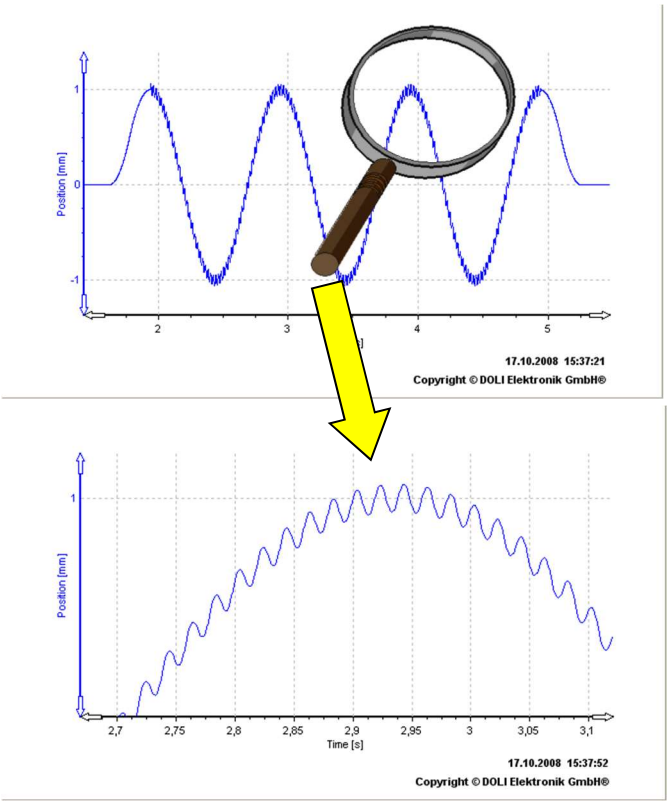
Cosine command with a linear amplitude sweep.
Cosine starts with amplitude of 1mm. The amplitude is linearly increased to 2mm within 5 seconds. The parameter SweepAmplitudeCount specifies how many sweeps should be done.





Cosine command with a logarithmic amplitude sweep.
Cosine starts with amplitude of 1mm. The amplitude is exponentially increased to 2mm within 5 seconds. The parameter SweepAmplitudeCount specifies how many sweeps should be done.





Cosine command superposition cosine.
The basic frequency of the Cosine is 1Hz. A cosine with a frequency of 50Hz and amplitude of 0.05mm is superimposed.

Please note; it is possible to have different sweeps at the same time. E.g. Frequency and amplitude sweep.

Rule for detecting end of cycling:

First priority has the parameter HalfCycles. If the Parameter HalfCycles is not zero, the command is finished after the number of half-cycles are finished, no matter if the sweep count has been reached.

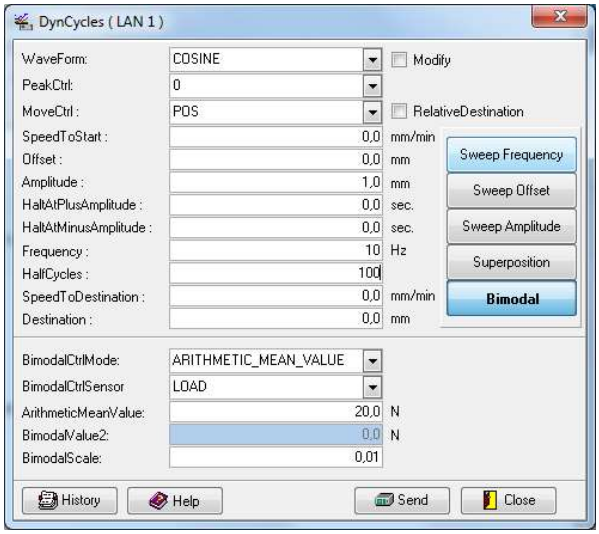
If HalfCycles is set to zero, the command is finished after **all** sweeps are finished.

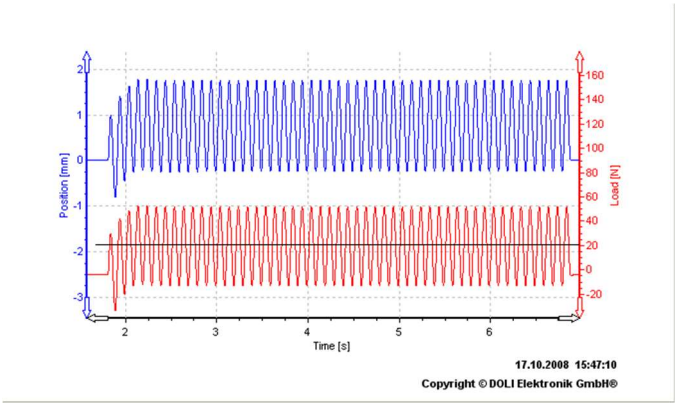
5.6.6 Bimodal commands

A bimodal command runs the test in e.g. position control, but keep a certain e.g. load. The load is controlled by changing the offset of e.g. position. The parameter BimodalScale defines the scaling between the two sensors. You may calculate this parameter by determining the ratio between move control sensor / bimodal control sensor.

(For move control = Position, and bimodal control = Load, BimodalScale represents 1/stiffness of your specimen in mm/N.)

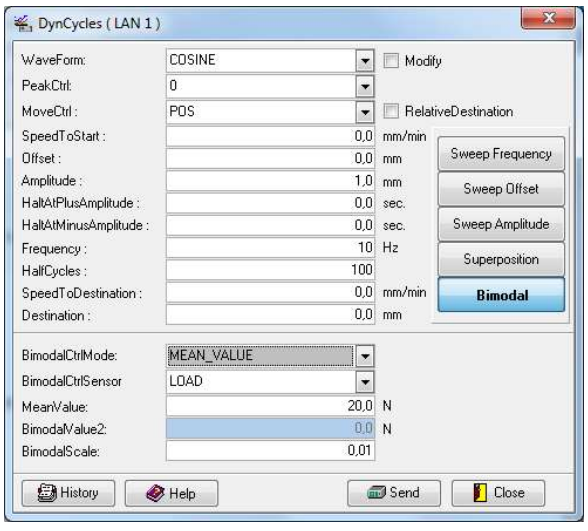
5.6.6.1 Control arithmetic mean value

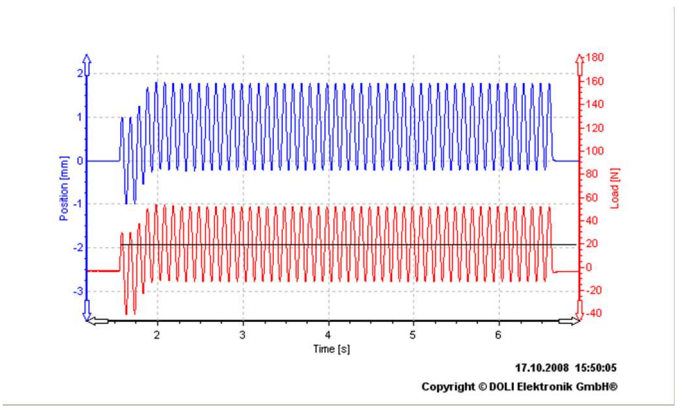




Do cosine cycles in position control with amplitude of 1mm, and starting offset of 0mm.
Control the arithmetic mean value of load at 20N, by modifying the offset of position.

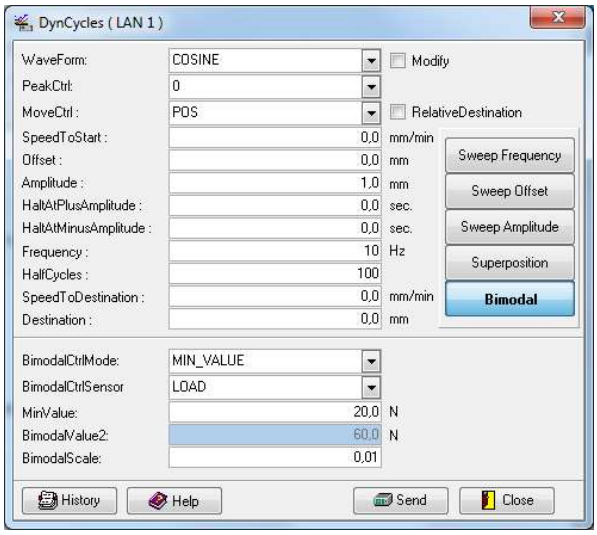
5.6.6.2 Control mean value

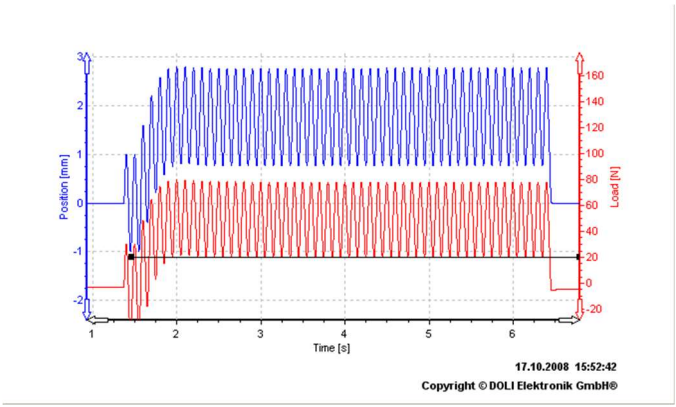




Do cosine cycles in position control with an amplitude of 1mm, and starting offset of 0mm.
Control the mean value of load at 20N, by modifying the offset of position.

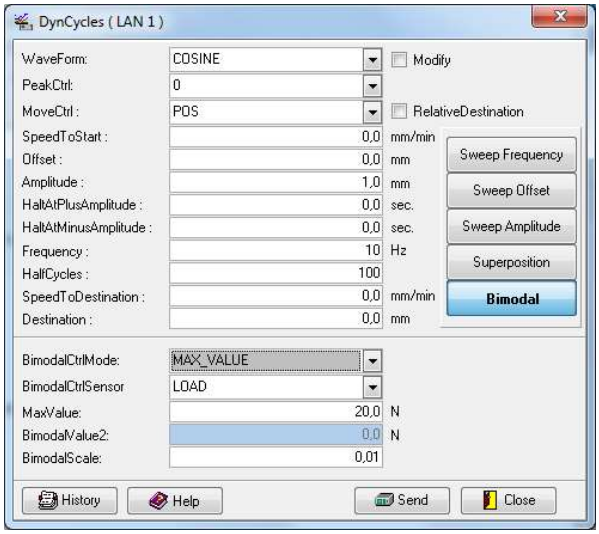
5.6.6.3 Control minimum value

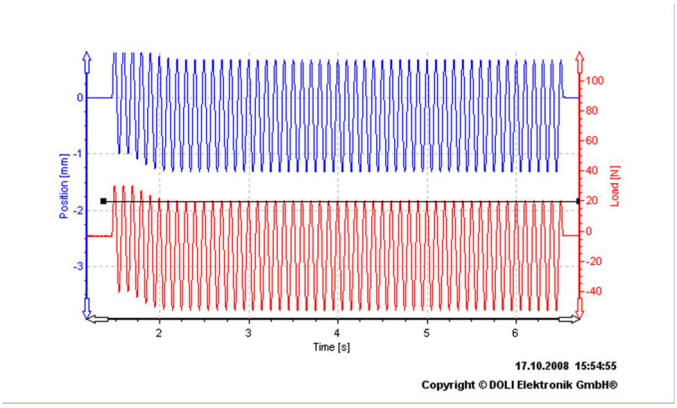




Do cosine cycles in position control with amplitude of 1mm, and starting offset of 0mm.
Control the minimum value of load at 20N, by modifying the offset of position.

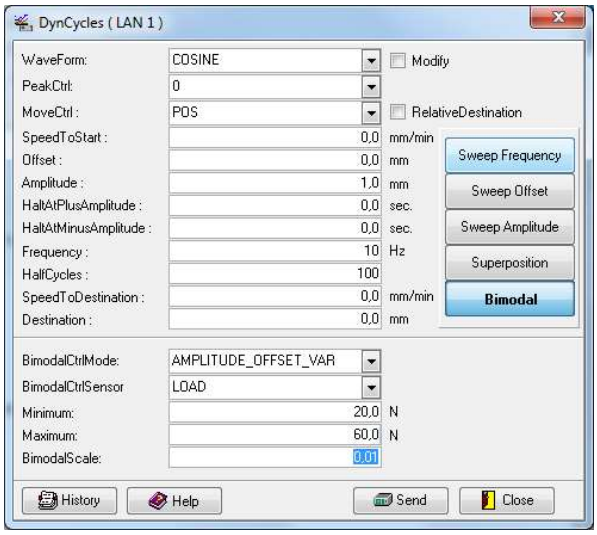
5.6.6.4 Control maximum value

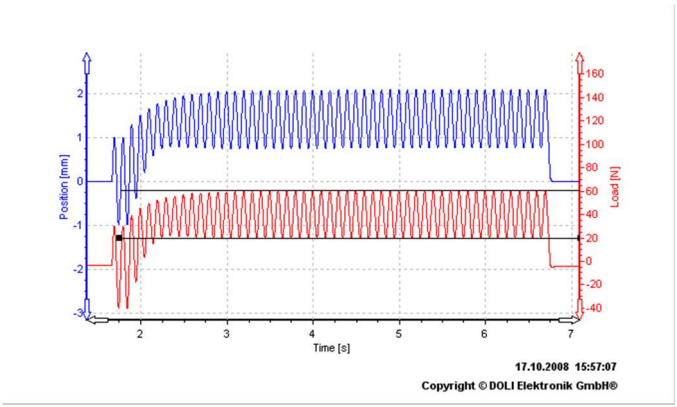




Do cosine cycles in position control with amplitude of 1mm, and starting offset of 0mm.
Control the maximum value of load at 20N, by modifying the offset of position.

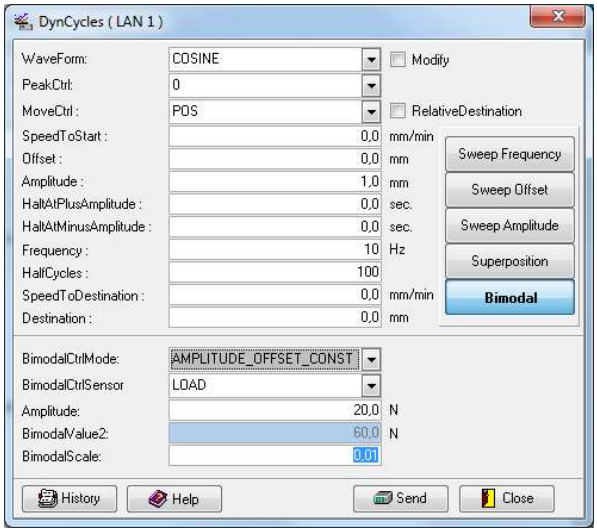
5.6.6.5 Control minimum AND maximum values,

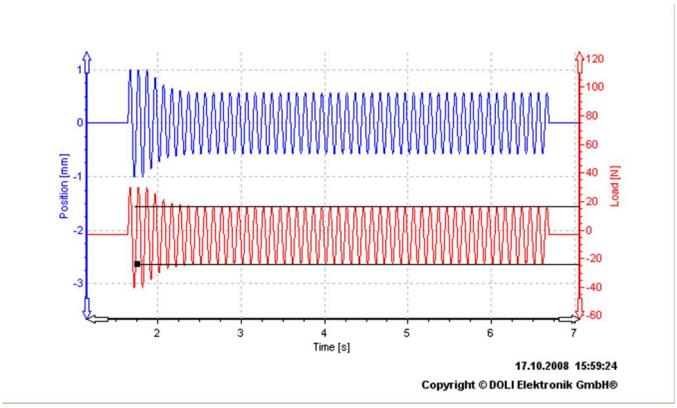




Do cosine cycles in position control with amplitude of 1mm, and starting offset of 0mm.
Control the minimum and maximum value of the load at 20N, and 60N, by modifying the offset, and amplitude of position.

5.6.6.6 Control minimum AND maximum values





Do cosine cycles in position control with amplitude of 1mm, and starting offset of 0mm.
Control the amplitude of load at 20N, by modifying the amplitude of position.

5.6.7 Restrictions:

- Frequency Sweeps are only possible for Cosine waveforms!
- Amplitude or Offset sweep should not change faster than 4 * 1/frequency!
- Amplitude AND SweepEndAmplitude MUST have the same sign!
- Modify not permitted with active Amplitude or Offset Sweep
- Modify not permitted for different waveforms
- Modify not permitted for different control modes
- Modify not permitted for different signs of the Amplitude
- Modify not permitted with RelativeDestination
- Bimodal control sensor must different to move control sensor
- BimodalValue1 MUST be smaller than BimodalValue2
- BimodalValue1 (Amplitude) MUST be positiv
- BimodalScale MUST be smaller than 32767 (EDC point scale)
- Amplitude and Offset Sweeps are not permitted for bi-modal modes: "DYN_BIMODAL_MIN_VALUE" or "DYN_BIMODAL_MAX_VALUE".
- Amplitude and Offset Sweeps are not permitted for bi-modal modes: "DYN_BIMODAL_AMPLITUDE_OFFSET_VAR" or "DYN_BIMODAL_AMPLITUDE_OFFSET_CONST"

5.6.8 DoPEDynCycle function declaration:

Function declaration		Description	Unit
extern unsigned DLLAPI	DoPEDynCycles (	Function returns Error constant (DoPERR_xxxx)	
DoPE_HANDLE	DoPEHdl	DoPE link handle	
unsigned short	WaveForm	DYN_WAVEFORM_xxx (Cosine, Triangular,…)	
unsigned short	Modify	Modify Parameter of active cycles	
unsigned short	PeakCtrl	Peak control cycles 0, 1,2,4,8,16	
unsigned short	MoveCtrl	Control mode (CTRL_xxx)	
unsigned short	RelativeDestination	false, Offset, and Destination absolute true, Offset, and Destination relative	
double	SpeedToStart	Speed to Offset + Amplitude	Unit/s
double	Offset	Offset	Unit
double	Amplitude	Amplitude	Unit
double	HaltAtPlusAmplitude	Halt Time at Offset + Amplitude	s
double	HaltAtMinusAmplitude	Halt Time at Offset – Amplitude	s
double	Frequency	Frequency	1/s
unsigned long	HalfCycles	Number of half cycles	
double	SpeedToDestination	Speed to final destination (0=automatic speed calculation)	Unit/s
double	Destination	Final destination	Unit
unsigned short	SweepFrequencyMode	DYN_SWEEP_xxx (Off, Linear, Logarithmic,…)	
double	SweepEndFrequency	End frequency for frequency sweep	1/s
double	SweepFrequencyTime	Time for frequency sweep	s
unsigned long	SweepFrequencyCount	Frequency sweep counter (0=infinitive)	
unsigned short	SweepOffsetMode	DYN_SWEEP_xxx (Off, Linear, Logarithmic,…)	
double	SweepEndOffset	Second offset for offset sweep	Unit
double	SweepOffsetTime	Time for offset sweep	s
unsigned long	SweepOffsetCount	Offset sweep counter (0=infinitive)	
unsigned short	SweepAmplitudeMode	DYN_SWEEP_xxx (Off, Linear, Logarithmic,…)	
double	SweepEndAmplitude	Second offset for offset sweep	Unit
double	SweepAmplitudeTime	Time for amplitude sweep	s
unsigned long	SweepAmplitudeCount	Amplitude sweep counter (0=infinitive)	Unit
unsigned short	SuperpositionMode	Superposition Mode (DYN_SUPERPOS_ON / _OFF)	
double	SuperpositionFrequency	Superposition Frequency	1/s
double	SuperpositionAmplitude	Superposition Amplitude	Unit
unsigned short	BimodalCtrlMode	Mode for bimodal control	
unsigned short	BimodalCtrlSensor	Sensor for bimodal control	
double	BimodalValue1	Value1 to keep constant	UnitBi
double	BimodalValue2	Value2 only for DYN_BIMODAL_AMPLITUDE_OFFSET_VAR	UnitBi
double	BimodalScale	Scale	Unit/UnitBi

WORD	*lpusTAN	Pointer to transaction number.	

6 Synchronized data and movement

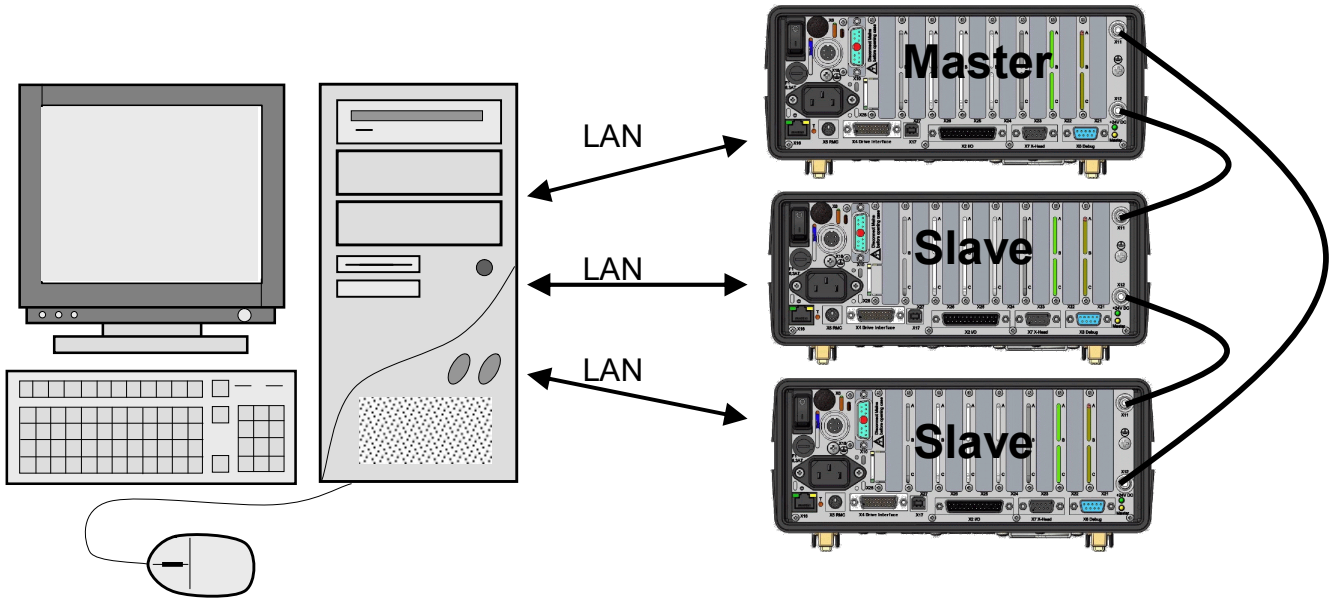
There are two commands (DoPESynchronizeSystemMode, DoPESynchronizeSystemStart) to control synchronized movement of two or more EDC systems.These commands are only useful if the synchronization option is installed, and the two or more EDC are correctly connected.

Important: **Make sure that all synchronized EDC's use the same**
Speed Controller Time (SpeedT)
Position Controller Time (PosT)
Data Transmission Rate (DaRate).

With DoPESynchronizeSystemMode you can synchronize time and movement, with DoPESynchronizeSystemStart you can start the synchronization.

There is also an event handler for synchronized measuring data samples available. The event is activated after DoPE receives samples with the same time from all synchronized EDCs.

The parameter SyncOption of DoPEGeneralData must be set correctly:
SYNC_OPTION_MASTER: for one Master EDC
SYNC_OPTION_SLAVE: for all other Slave EDCs



6.1 DoPESynchronizeSystemMode (Sync)

DoPESynchronizeSystemMode command is used for synchronized movement
(see sample and state diagram below).

Minimum requirements: EDC580/220, DoPE 2.51

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.

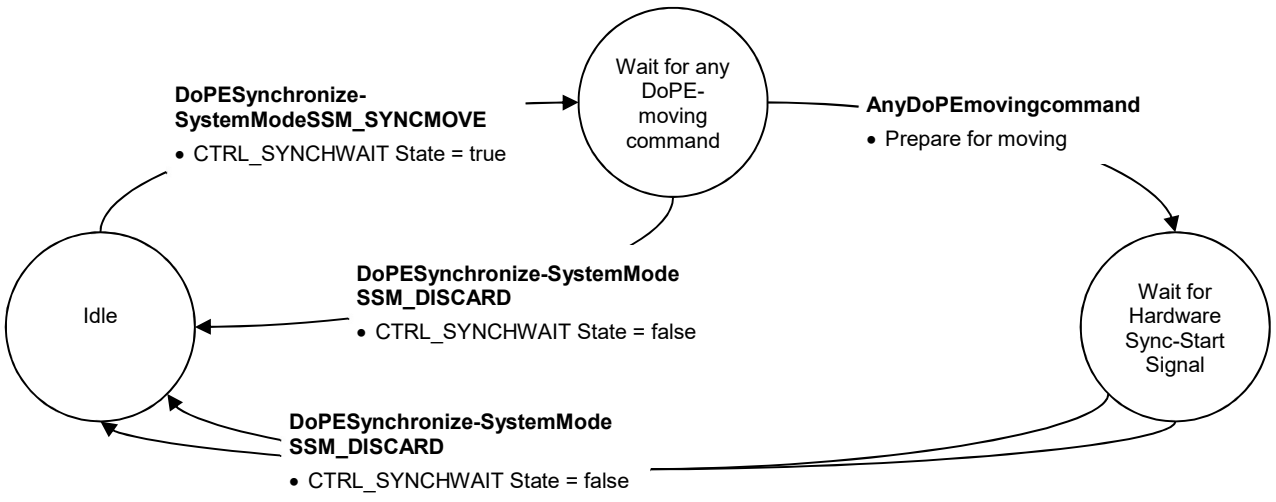
Function declaration		Description	Unit
extern unsigned DLLAPI	DoPESynchronizeSystemMode(	Function returns Error constant (DoPERR_xxxx)	
DoPE_HANDLE	DoPEHdl	DoPE link handle	
unsigned short	Mode	SSM_SYNCMOVE: If Time is ZERO, start movement after synchronize signal is active. If Time is not ZERO, wait Time after synchronize signal is active before startingthe movement. SSM_SYSTEMTIME: Set EDC-system time to Time, aftersynchronize signal is active. SSM_DISCARD: Discard previous DoPESynchronizeSystemMode command.	
double	Time	Delay or system time to set with the nextDoPESynchronizeSystemStartcommand.	s
WORD	*lpusTAN	Pointer to transaction number (not for Sync. version).	

6.2 DoPESynchronizeSystemStart (Sync)

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

Minimum requirements: EDC580/220, DoPE 2.51

Function declaration		Description	Unit
extern unsigned DLLAPI	DoPESynchronizeSystemStart (	Function returns Error constant (DoPERR_xxxx)	
DoPE_HANDLE	DoPEHdl	DoPE link handleof Master EDC	
WORD	*lpusTAN	Pointer to transaction number (not for Sync. version).	



Example1: Two axes should do synchronized sinus EDC 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 = DoPECosineSync ( 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 = DoPECosineSync ( DoPEHdl_2, CTRL_POS, Speed1, Dest1, Dest2, Frequency, HalfCycles, Speed
                     Destination, &lpusTAN_2);

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

Example2: Two axes should do synchronized sinusoidal cycles with 90° phase shift.
EDC 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 = DoPECosineSync ( 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 = DoPECosineSync ( DoPEHdl_2, CTRL_POS, Speed1, Dest1, Dest2, Frequency, HalfCycles, Speed
                      Destination, &lpusTAN_2);

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

6.3 DoPESynchronizeData

Activate sample synchronization for a list of DoPE handles.
Passing an empty linklist (first item NULL) stops sample synchronization. Initializing, selecting a setup or closing one or more links stops sample synchronization, too.

Function declaration	Description
extern unsigned DLLAPIDoPESynchronizeData(DoPE_HANDLE DoPEHdl[] DoPE_HANDLE *pMaster)	Function returns Error constant (DoPERR_xxxx) DoPE link handle array (terminated by NULL). Pointer to storage for the master DoPE link handle.

6.4 DoPESetOnSynchronizeDataHdlr

Set the OnSynchronizeData handler.

Minimum requirements: EDC580/220, DoPE 2.51

Function declaration	Description
extern unsigned DLLAPIDoPESetOnSynchronizeDataHdlr(DoPEOnSynchronizeDataHdlr Hdlr LPVOID lpParameter)	Function returns Error constant (DoPERR_xxxx) User function to be called with everyreceived synchronized sample. User specific pointer.
User function for the OnSynchronizeDataevent:	
unsigned CALLBACK OnSynchronizeData DoPE_HANDLE DoPEHdl DoPEOnSynchronizeData *Data LPVOID lpParameter	Function should return 0 (reserved for future versions) DoPE link handle Pointer to received data User specific pointer set with DoPESetOnSynchronizeDataHdlr
typedef struct { unsigned DoPError unsigned nData DoPELinkData *Sample double Occupied } DoPEOnSynchronizeData; typedef struct { DoPE_HANDLE DP DoPEData Data } DoPELinkData	 DoPERR_NOERROR event is ok. Number of measuring data records Array of measuring data records. Utilization [% of data buffer size] DoPE link handle Measuring data record

6.5 DoPESetOnSynchronizeDataOverflowHdlr

Set the OnSynchronizeDataOverflow handler.

Minimum requirements: EDC580/220, DoPE 2.67

Function declaration	Description
extern unsigned DLLAPI DoPESetOnSynchronizeDataOverflowHdlr(DoPEOnSynchronizeDataOverflowHdlr Hdlr LPVOID lpParameter	Function returns Error constant (DoPERR_xxxx) User function to be called with every overflow of synchronized sample. User specific pointer.
User function for the OnSynchronizeDataOverflow event:	
unsigned CALLBACK OnSynchronizeDataOverflow LPVOID lpParameter	Function should return 0 (reserved for future versions) Userspecific pointer set with DoPESetOnSynchronizeDataOverflowHdlr

7 Miscellaneous Control Commands

7.1 DoPEOn(Sync)

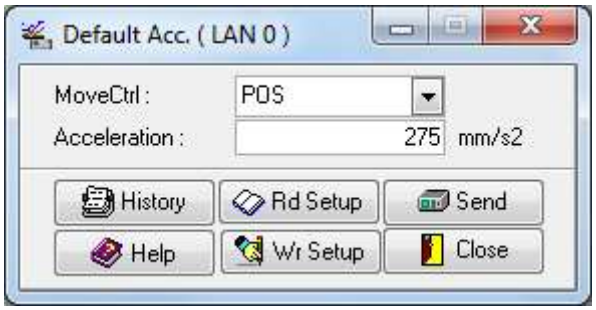
Activate drive (not available for EDC100.The EDC100 has a hardware "ON" push button).			
Function declaration		**Description**	**Unit**
extern unsigned DLLAPI DoPE_HANDLE WORD	DoPEOn(DoPEHdl *lpusTAN	Function returns Error constant (DoPERR_xxxx) DoPE link handle Pointer to transaction number (not for Sync. version).	

7.2 DoPEOff (Sync)

Deactivate drive.			
Function declaration		**Description**	**Unit**
extern unsigned DLLAPI DoPE_HANDLE WORD WORD	DoPEOff(DoPEHdl *lpusTANFirst *lpusTANLast	Function returns Error constant (DoPERR_xxxx) DoPE link handle Pointer to transaction number (not for Sync. version)	

7.3 DoPEDefaultAcc(Sync)

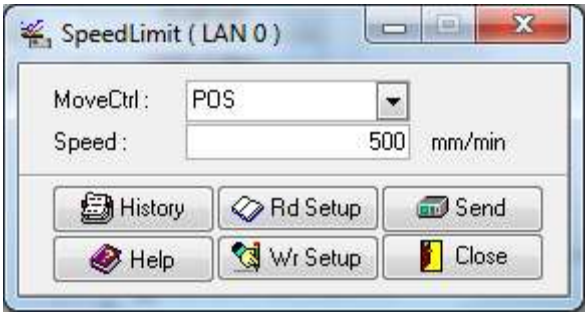
Set default acceleration (and deceleration) for all moving commands. After initialization default and nominal acceleration are identical.



Function declaration		**Description**	**Unit**
extern unsigned DLLAPI DoPE_HANDLE unsigned short double WORD	DoPEDefaultAcc (DoPEHdl MoveCtrl Acc *lpusTAN	Function returns Error constant (DoPERR_xxxx) DoPE link handle Control mode (CTRL_xxx) Default Acceleration Pointer to transaction number (not for Sync. version)	

7.4 DoPESpeedLimit(Sync)

Set speed limit. The maximum speed for all moving commands will be limited to this SpeedLimit.



Function declaration		Description	Unit
extern unsigned DLLAPI	DoPESpeedLimit (	Function returns Error constant (DoPERR_xxxx)	
DoPE_HANDLE	DoPEHdl	DoPE link handle	
unsigned short	MoveCtrl	Control mode (CTRL_xxx)	
double	Speed	Maximum allowed speed (must be below nominal speed)	Unit/s
WORD	*lpusTAN	Pointer to transaction number (not for Sync. version).	

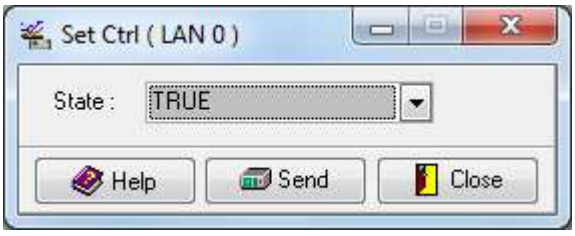
7.5 DoPESetCtrl (Sync)

Enable / disable closed loop control.

Minimum requirements: DoPE 2.51

Use this command to switch off EDC-Closed Loop Control.

After switching closed loop control off, PC must wait unlil all bits in ActiveCtrl are ZERO.



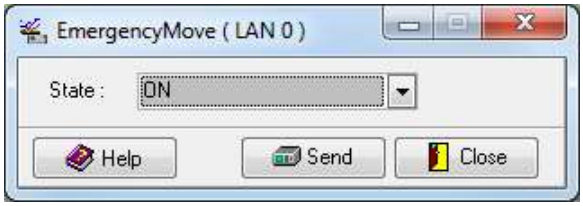
This command activates closed loop control of EDC.

Wait until the CTRL_READY bit in controller status word 1 is active before any moving command is used!

Function declaration		Description	Unit
extern unsigned DLLAPI	DoPESetCtrl (	Function returns Error constant (DoPERR_xxxx)	
DoPE_HANDLE	DoPEHdl	DoPE link handle	
unsigned	Enable	!=0 enables =0 disables closed loop control	
WORD	*lpusTAN	Pointer to transaction number (not for Sync. version).	

7.6 DoPEEmergencyMove(Sync)

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



Function declaration		Description	Unit
extern unsigned DLLAPI	DoPEEmergencyMove (	Function returns Error constant (DoPERR_xxxx)	
DoPE_HANDLE	DoPEHdl	DoPE link handle	
Unsigned short	State	!=0 on =0 off	
WORD	*lpusTAN	Pointer to transaction number (not for Sync. version).	

7.7 DoPEEmergencyOff(Sync)

Activate / deactivate EmergencyOff state.

The emergency off state may be activated or deactivated by software.

Function declaration		Description	Unit
extern unsigned DLLAPI	DoPEEmergencyOff (	Function returns Error constant (DoPERR_xxxx)	
DoPE_HANDLE	DoPEHdl	DoPE link handle	
Unsigned short	State	!=0 Activate emergency off =0 Deactivate emergency off	
WORD	*lpusTAN	Pointer to transaction number (not for Sync. version).	

7.8 DoPESetOpenLoopCommand(Sync)

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.

Function declaration		Description	Unit
extern unsigned DLLAPI	DoPESetOpenLoopCommand (	Function returns Error constant (DoPERR_xxxx)	
DoPE_HANDLE	DoPEHdl	DoPE link handle	
double	Command	Output value to valve in %.	%
WORD	*lpusTAN	Pointer to transaction number (not for Sync. version).	

8 Event Handler

8.1 Using DoPE Event handler

Whenever necessary, EDC will send a message, possibly with parameters to PC. There are many different reasons for such messages. E.g. if a positioning command has reached the destination, EDC will send a message to PC, reporting the destination has been reached.

To make it easy for the application program to get these messages, DoPE offers an event handler interface.For each event, the application program wants to process, an event handler must be installed. DoPE will call these event handlers, whenever the event occurs.

8.2 Overview of Events:

Event Name	Description	Related DoPE Command
OnLine	EDC goes On/Offline	-
OnData	New data record with measures values	-
OnCommandError	Error event upon a command	Any command
OnPosMsg	A positioning command has reached the destination	Any positioning command like DoPEPos, DoPEEndCMD
OnTPosMsg	Trigger position hit	DoPETrig, Positioning command with DoPEStartCMD/EndCMD
OnLPosMsg	Limit position reached	DoPEPosExt, DoPEFDPoti when reaching softend
OnSftMsg	Softend reached	DoPESft, Any positioning command
OnOffsCMsg	Offset correction finished	DoPEOffsC
OnCheckMsg	A measuring channel supervision hit.	DoPESetCheck
OnShieldMsg	Safety shield message	DoPEShieldLimit
OnRefSignalMsg	Reference signal message of a encoder	DoPESetRefSignalMode
OnSensorMsg	Message from a serial sensor	-
OnOverflow	Overflow of data records	-
OnRuntimeError	Any error condition, like emergency off active.	-
OnIoSHaltMsg	Upper / Lower S-Halt input triggered	-
OnKeyMsg	RMC-key state	-
OnSystemMsg	System message, like initialization errors	-
OnDebugMsg	Debug message (for remote support)	DoPEDebugMsgEnable DoPESendDebugCommand

8.3 DoPESetOnLineHdlr

The OnLine Handler supplies any change of the communication state between EDC, and PC.	
Minimum requirements: DoPE 2.23	
Function declaration	**Description**
extern unsigned DLLAPI DoPESetOnLineHdlr DoPE_HANDLE DoPEHdl DoPEOnLineHdlr Hdlr LPVOID lpParameter	Function returns Error constant (DoPERR_xxxx) DoPE link handle User function to be called after every line state change. User specific pointer.
User function for the OnLine event:	
unsigned CALLBACK OnLine DoPE_HANDLE DoPEHdl int LineState LPVOID lpParameter	Function should return 0 (reserved for future versions) DoPE link handle DoPEOFFLINE (0) link is offline DoPEONLINE (1) link is online DoPERESTART (2) asynchronous restart of the link User specific pointer set with DoPESetOnLineHdlr

8.4 DoPESetOnDataHdlr

The OnData event handler is called with each received measuring data record from EDC.	
Minimum requirements: DoPE 2.23	
Function declaration	**Description**
extern unsigned DLLAPI DoPESetOnDataHdlr DoPE_HANDLE DoPEHdl DoPEOnDataHdlr Hdlr LPVOID lpParameter	Function returns Error constant (DoPERR_xxxx) DoPE link handle User function to be called after a new data record is available. User specific pointer
User function for the OnData event:	
unsigned CALLBACK OnData DoPE_HANDLE DoPEHdl DoPEOnData *pData LPVOID lpParameter	Function should return 0 (reserved for future versions) DoPE link handle Pointer to received measuring data record User specific pointer set with DoPESetOnDataHdlr
typedef struct { unsigned DoPError; DoPEData Data; double Occupied; } DoPEOnData;	 DoPERR_NOERROR event is ok, else error occurred during event generation, e.g. data conversion error. Measuring data record. Utilization in percent of the measuring data circular buffer inside DoPE.

8.5 DoPESetOnDataBlockHdlr

The OnDataBlock event handler is called with each received measuring data record block from EDC. Per default DoPE collects 50 measuring data records in a block. This number can be changed with the DoPESetOnDataBlockSize function.	
Minimum requirements: DoPE 2.80	
Function declaration	**Description**
extern unsigned DLLAPI DoPESetOnDataBlockHdlr DoPE_HANDLE DoPEHdl DoPEOnDataBlockHdlr Hdlr LPVOID lpParameter	Function returns Error constant (DoPERR_xxxx) DoPE link handle User function to be called after a new data record block is available. User specific pointer
User function for the OnDataBlock event:	
unsigned CALLBACK OnDataBlock DoPE_HANDLE DoPEHdl DoPEOnDataBlock *pData LPVOID lpParameter	Function should return 0 (reserved for future versions) DoPE link handle Pointer to measuring data record block User specific pointer set with DoPESetOnDataBlockHdlr
typedef struct { unsigned DoPError; unsigned nData DoPEOnData *pData double Occupied; } DoPEOnDataBlock;	 DoPERR_NOERROR event is ok, else at least one measuring data record reports an error. Number of measuring data records in the block Pointer to measuring data records Utilization in percent of the measuring data circular buffer inside DoPE.

8.6 DoPESetOnDataBlockSize

Set the number of measuring data records to collect in an OnDataBlock block.	
Minimum requirements: DoPE 2.80	
Function declaration	**Description**
extern unsigned DLLAPI DoPESetOnDataBlockSize DoPE_HANDLE DoPEHdl unsigned nData	Function returns Error constant (DoPERR_xxxx) DoPE link handle Number of samples in a block

8.7 DoPESetOnDataBlockSize

Get the number of measuring data record to collect in an OnDataBlock block.	
Minimum requirements: DoPE 2.80	
Function declaration	**Description**
extern unsigned DLLAPI DoPEGetOnDataBlockSize DoPE_HANDLE DoPEHdl unsigned *nData	Function returns Error constant (DoPERR_xxxx) DoPE link handle Pointer to storage for number of samples in a block

8.8 DoPE SetOnCommandErrorHdlr

The OnCommandError event handler is called after parameter check of any DoPE command.
DoPE commands like DoPEPos (not DoPEPosSync) report errors after being checked by EDC-firmware.
After receiving the command error message from EDC, DoPE sends the OnCommandError event.
(Only for non synchronized commands)

Minimum requirements: DoPE 2.23

Function declaration	**Description**
extern unsigned DLLAPISetOnCommandErrorHdlr DoPE_HANDLE DoPEHdl DoPEOnCommandErrorHdlr Hdlr LPVOID lpParameter	Function returns Error constant (DoPERR_xxxx) DoPE link handle User function to be called. User specific pointer.
User function for the OnCommandErrorevent:	
unsigned CALLBACK OnCommandError DoPE_HANDLE DoPEHdl DoPEOnCommandError *pError LPVOID lpParameter	Function should return 0 (reserved for future versions) DoPE link handle Pointer to command error structure User specific pointer set with DoPESetOnDataHdlr
typedef struct { WORD CommandNumber WORD ErrorNumber WORD usTAN } DoPEOnCommandError;	 EDC internal command number (for internal use). ErrorNumber specifies the error type (see DoPERR_CMD_xxx). Transaction number of the corresponding DoPE command. **Note: Application programs using the Sync commands do not need to set the OnCommandError event handler.**

8.9 DoPESetOnPosMsgHdlr

Whenever a moving command, like DoPEPos, is finished, EDC will send a message to PC. The moving command is finished after the command generator reached the final destination. DoPE will call the OnPosMsg event handler with a pointer to the DoPEOnPosMsg structure:

Minimum requirements: DoPE 2.23

Function declaration	**Description**
extern unsigned DLLAPI DoPESetOnPosMsgHdlr DoPE_HANDLE DoPEHdl DoPEOnPosMsgHdlr Hdlr LPVOID lpParameter	Function returns Error constant (DoPERR_xxxx) DoPE link handle User function to be called. User specific pointer.
User function for the OnPosMsg event:	
unsigned CALLBACK OnPosMsg DoPE_HANDLE DoPEHdl DoPEOnPosMsg * pPosMsg LPVOID lpParameter	Function should return 0 (reserved for future versions) DoPE link handle Pointer to positioning message structure User specific pointer set with DoPESetOnPosMsgHdlr
typedef struct { Unsigned DoPError WORD Reached double Time WORD Control double Position WORD DControl double Destination WORD usTAN } DoPEOnPosMsg;	 DoPERR_NOERROR event is ok. Else error occurred during event generation, e.g. data conversion error. 1 if current position is within the destination window, 0 if current position is outside destination window. EDC-System time when reaching destination. Control mode of the next parameter (Position) If Reached = 1 Destination of the corresponding moving command. If Reached = 0 Current position. Control mode of the next parameter (Destination) Here: DControl has always the same value as Control. Destination of the corresponding moving command. Transaction number of the corresponding moving command.

8.10 DoPESetOnTPosMsgHdlr

This event is sent if the trigger position of a DoPETrig command hits. DoPE will call the OnTPosMsg event handler with a pointer to the DoPEOnPosMsg structure:

Minimum requirements: DoPE 2.23

Function declaration	**Description**
extern unsigned DLLAPI DoPESetOnTPosMsgHdlr DoPE_HANDLE DoPEHdl DoPEOnTPosMsgHdlr Hdlr LPVOID lpParameter	Function returns Error constant (DoPERR_xxxx) DoPE link handle User function to be called after the trigger position of DoPETrig command hits. User specific pointer.
User function for the OnTPosMsg event:	
unsigned CALLBACK OnTPosMsg DoPE_HANDLE DoPEHdl DoPEOnTPosMsg * pTPosMsg LPVOID lpParameter	Function should return 0 (reserved for future versions) DoPE link handle Pointer to positioning message structure User specific pointer set with DoPESetOnTPosMsgHdlr
typedef struct { Unsigned DoPError WORD Reached double Time WORD Control double Position WORD DControl double Destination WORD usTAN } DoPEOnPosMsg;	 DoPERR_NOERROR event is ok. Else error occurred during event generation, e.g. data conversion error. Always 1. EDC-System time when trigger hits. Control mode of the next parameter (Position) Limit position of DoPETrig com. Control mode of the next parameter (Destination). Trigger position of DoPETrig command. Transaction number of the corresponding DoPETrig command.

8.11 DoPESetOnLPosMsgHdlr

This event is sent if the limit position of a DoPEPosExt command hits, or a softend position exceeds during DoPEFDPoti command. DoPE will call the OnLPosMsg event handler with a pointer to the DoPEOnPosMsg structure:

Minimum requirements: DoPE 2.23

Function declaration		**Description**
extern unsigned DLLAPI	DoPESetOnLPosMsgHdlr	Function returns Error constant (DoPERR_xxxx)
DoPE_HANDLE	DoPEHdl	DoPE link handle
DoPEOnLPosMsgHdlr	Hdlr	User function to be called after the limit position of DoPEPosExt command hits.
LPVOID	lpParameter	User specific pointer.
User function for the OnLPosMsg event:		
unsigned CALLBACK OnLPosMsg		Function should return 0 (reserved for future versions)
DoPE_HANDLE	DoPEHdl	DoPE link handle
DoPEOnLPosMsg	* pLPosMsg	Pointer to positioning message structure
LPVOID	lpParameter	User specific pointer set with DoPESetOnLPosMsgHdlr
typedef struct {		
Unsigned	DoPError	DoPERR_NOERROR event is ok. Else error occurred during event generation, e.g. data conversion error.
WORD	Reached	Always 1.
double	Time	EDC-System time when limit hits.
WORD	Control	Control mode of the next parameter (Position)
double	Position	Limit position of DoPEPosExt com.
WORD	DControl	Control mode of the next parameter (Destination).
double	Destination	Destination position of DoPEPosExt command.
WORD	usTAN	Transaction number of the corresponding DoPEPosExt command.
} DoPEOnPosMsg;		

8.12 DoPESetOnSftMsgHdlr

This event is sent if a softend position was exceeded.

Minimum requirements: DoPE 2.23

Function declaration		**Description**
extern unsigned DLLAPI	DoPESetOnSftMsgHdlr	Function returns Error constant (DoPERR_xxxx)
DoPE_HANDLE	DoPEHdl	DoPE link handle
DoPEOnSftMsgHdlr	Hdlr	User function to be called after a positioning command wasfinished.
LPVOID	lpParameter	User specific pointer.
User function for the OnSftMsg event:		
unsigned CALLBACK OnSftMsg		Function should return 0 (reserved for future versions)
DoPE_HANDLE	DoPEHdl	DoPE link handle
DoPEOnSftMsg	*pSftMsg	Pointer to positioning message structure
LPVOID	lpParameter	User specific pointer set with DoPESetOnSftMsgHdlr
typedef struct {		
Unsigned	DoPError	DoPERR_NOERROR event is ok. Else error occurred during event generation, e.g. data conversion error.
WORD	Upper	1 = upper softend. 0 = lower softend.
double	Time	EDC-System time when softend was exceeded.
WORD	Control	Control mode of the softend that was exceeded.
double	Position	Position, where the softend was detected.
WORD	usTAN	Transaction number of the last positioning command.
} DoPEOnSftMsg;		

8.13 DoPESetOnOffsCMsgHdlr

Handler for offset correction messages	
Minimum requirements: DoPE 2.23	
Function declaration	**Description**
extern unsigned DLLAPI DoPESetOnOffsCMsgHdlr DoPE_HANDLE DoPEHdl DoPEOnOffsCMsgHdlr Hdlr LPVOID lpParameter	Function returns Error constant (DoPERR_xxxx) DoPE link handle User function to be called after an offset correction, activated withDoPEOffsC command was finished. User specific pointer.
User function for the OnOffsCMsg event:	
unsigned CALLBACK OnOffsCMsg DoPE_HANDLE DoPEHdl DoPEOnOffsCMsg *pOffsCMsg LPVOID lpParameter	Function should return 0 (reserved for future versions) DoPE link handle Pointer to positioning message structure User specific pointer set with DoPESetOnOffsCMsgHdlr
typedef struct { Unsigned DoPError double Time WORD Offset WORD usTAN } DoPEOnOffsCMsg;	 DoPERR_NOERROR event is ok. Else error occurred during event generation, e.g. data conversion error. EDC-System time when offset correction was finished. Measured offset of the output channel. Transaction number of the corresponding DoPEOffsC command.

8.14 DoPESetOnCheckMsgHdlr

Handler for channel supervision messages	
Minimum requirements: DoPE 2.23	
Function declaration	**Description**
extern unsigned DLLAPI DoPESetOnCheckMsgHdlr DoPE_HANDLE DoPEHdl DoPEOnCheckMsgHdlr Hdlr LPVOID lpParameter	Function returns Error constant (DoPERR_xxxx) DoPE link handle User function to be called after a channel supervision condition hits. User specific pointer.
User function for the OnCheckMsg event:	
unsigned CALLBACK OnCheckMsg DoPE_HANDLE DoPEHdl DoPEOnCheckMsg *pCheckMsg LPVOID lpParameter	Function should return 0 (reserved for future versions) DoPE link handle Pointer to check message structure User specific pointer set with DoPESetOnCheckMsgHdlr
typedef struct { Unsigned DoPError WORD Action double Time WORD CheckId Double Position WORD SensorNo WORD usTAN } DoPEOnCheckMsg;	 DoPERR_NOERROR event is ok. Else error occurred during event generation, e.g. data conversion error. 1 = the action of DoPESetCheck command was started. 0 = the action of DoPESetCheck command was NOT started. (due to an error condition like active softend) EDC-System time. ID of measuring channel check. Position of the supervised sensor. Supervised sensor. Transaction number of the corresponding DoPESetCheck command.

8.15 DoPESetOnShieldMsgHdlr

Handler for channel shield messages	
Minimum requirements: DoPE 2.23	
Function declaration	**Description**
extern unsigned DLLAPIDoPESetOnShieldMsgHdlr DoPE_HANDLE DoPEHdl DoPESetOnShieldMsgHdlr Hdlr LPVOID lpParameter	Function returns Error constant (DoPERR_xxxx) DoPE link handle User function to be called after a shield limit has triggered. User specific pointer.
User function for the OnShieldMsg event:	
unsigned CALLBACK OnShieldMsg DoPE_HANDLE DoPEHdl DoPEOnShieldMsg *pOnShieldMsg LPVOID lpParameter	Function should return 0 (reserved for future versions) DoPE link handle Pointer to shield message structure. User specific pointer set with DoPESetOnShieldMsgHdlr
typedef struct { Unsigned DoPError WORD Action double Time WORD SensorNo Double Position WORD usTAN } DoPEOnShieldMsg;	 DoPERR_NOERROR event is ok. Else error occurred during event generation, e.g. data conversion error. 1 = the action of DoPEShieldLimit command was started. 0 = the action of DoPEShieldLimit command was NOT started. (due to an error condition like active softend) EDC-System time. Supervised sensor. Position of the supervised sensor. Transaction number of the corresponding DoPEShieldLimit command.

8.16 DoPESetOnRefSignalMsgHdlr

Handler for reference signal messages	
Minimum requirements: DoPE 2.23	
Function declaration	**Description**
extern unsigned DLLAPIDoPESetOnRefSignalMsgHdlr DoPE_HANDLE DoPEHdl DoPESetOnRefSignalMsgHdlr Hdlr LPVOID lpParameter	Function returns Error constant (DoPERR_xxxx) DoPE link handle User function to be called after a reference signal occurred. User specific pointer.
User function for the OnRefSignalMsg event:	
unsigned CALLBACK OnRefSignalMsg DoPE_HANDLE DoPEHdl DoPEOnRefSignalMsg * pRefSignalMsg LPVOID lpParameter	Function should return 0 (reserved for future versions) DoPE link handle Pointer to reference signal message structure. User specific pointer set with DoPESetOnRefSignalMsgHdlr
typedef struct { Unsigned DoPError double Time WORD SensorNo Double Position WORD usTAN } DoPEOnRefSignalMsg;	 DoPERR_NOERROR event is ok. Else error occurred during event generation, e.g. data conversion error. EDC-System time. Sensor where the reference signal occurred. Position at reference signal. Transaction number of the corresponding DoPESetRefSignalMode command.

8.17 DoPESetOnSensorMsgHdlr

Handler for serial sensormessages	
Minimum requirements: DoPE 2.23	
Function declaration	**Description**
extern unsigned DLLAPIDoPESetOnSensorMsgHdlr DoPE_HANDLE DoPEHdl DoPESetOnSensorMsgHdlr Hdlr LPVOID lpParameter	Function returns Error constant (DoPERR_xxxx) DoPE link handle User function to be called after a message from a serial sensor wasreceived. User specific pointer.
User function for the OnSensorMsg event:	
unsigned CALLBACK OnSensorMsg DoPE_HANDLE DoPEHdl DoPEOnSensorMsg * pOnSensorMsg LPVOID lpParameter	Function should return 0 (reserved for future versions) DoPE link handle Pointer to serial sensor message structure. User specific pointer set with DoPESetOnSensorMsgHdlr
typedef struct { Unsigned DoPError double Time WORD SensorNo WORD Length BYTE Data[SENSOR_MSG_LEN+1] WORD usTAN } DoPEOnSensorMsg;	 DoPERR_NOERROR event is ok. Else error occurred during event generation, e.g. data conversion error. EDC-System time. Sensor number. Number of bytes in Data. Message from the sensor. Transaction number is unused, always ZERO.

8.18 DoPESetOnIoSHaltMsgHdlr

Handler for IO-SHalt messages.	
Minimum requirements: EDC580/220, EdcApp 9133.022, DoPE 2.71	
Function declaration	**Description**
extern unsigned DLLAPI DoPESetOnIoSHaltMsgHdlr DoPE_HANDLE DoPEHdl DoPESetOnIoSHaltMsgHdlr Hdlr LPVOID lpParameter	Function returns Error constant (DoPERR_xxxx) DoPE link handle User function to be called after a SHalt input has triggered. User specific pointer.
User function for the OnIoSHaltMsg event:	
unsigned CALLBACK OnIoSHaltMsg DoPE_HANDLE DoPEHdl DoPEOnIoSHaltMsg * pOnIoSHaltMsg LPVOID lpParameter	Function should return 0 (reserved for future versions) DoPE link handle Pointer to IO-SHalt message structure. User specific pointer set with DoPESetOnIoSHaltMsgHdlr
typedef struct { Unsigned DoPError WORD Upper double Time WORD Control double Position WORD usTAN } DoPEOnIoSHaltMsg;	 DoPERR_NOERROR event is ok. Else error occurred during event generation, e.g. data conversion error. 1: Upper SHalt input triggered. 0: Lower SHalt input triggered. EDC-System time. Control mode of Position. Position at SHalt input trigger. Transaction number.

8.19 DoPESetOnKeyMsgHdlr

Handler for keymessages. For each activated or deactivated key this message is sent.	
Minimum requirements: DoPE 2.51	
Function declaration	**Description**
extern unsigned DLLAPI DoPESetOnKeyMsgHdlr DoPE_HANDLE DoPEHdl DoPESetOnKeyMsgHdlr Hdlr LPVOID lpParameter	Function returns Error constant (DoPERR_xxxx) DoPE link handle User function to be called after a change of any (RMC) key was detected. User specific pointer.
User function for the OnKeyMsg event:	
unsigned CALLBACK OnKeyMsg DoPE_HANDLE DoPEHdl DoPEOnKeyMsg * pOnKeyMsg LPVOID lpParameter	Function should return 0 (reserved for future versions) DoPE link handle Pointer to key message structure. User specific pointer set with DoPESetOnKeyMsgHdlr
typedef struct { Unsigned DoPError double Time unsigned __int64 Keys unsigned __int64 NewKeys unsigned __int64 GoneKeys WORD OemKeys WORD NewOemKeys WORD GoneOemKeys WORD usTAN } DoPEOnKeyMsg;	 DoPERR_NOERROR event is ok. Else error occurred during event generation, e.g. data conversion error. EDC-System time. Current key state. All new keys. All gone keys. Current OEM key state (for future use). New OEM keys (for future use). Gone OEM keys (for future use). Transaction number.

8.20 DoPESetOnRuntimeErrorHdlr

Handler for runtime errormessages.	
Minimum requirements: DoPE 2.23	
Function declaration	**Description**
extern unsigned DLLAPIDoPESetOnRuntimeErrorHdlr DoPE_HANDLE DoPEHdl DoPEOnRuntimeErrorHdlr Hdlr LPVOID lpParameter	Function returns Error constant (DoPERR_xxxx) DoPE link handle User function to be called after a runtime error occurred. User specific pointer.
User function for the OnOnRuntimeError event:	
unsigned CALLBACK OnRuntimeError DoPE_HANDLE DoPEHdl pOnRuntimeError * pOnRuntimeError LPVOID lpParameter	Function should return 0 (reserved for future versions) DoPE link handle Pointer to runtime error structure. User specific pointer set with DoPESetOnRuntimeErrorHdlr
typedef struct { unsigned DoPError unsigned short ErrorNumber double Time unsigned short Device unsigned short Bits WORD usTAN } DoPEOnRuntimeError;	 DoPERR_NOERROR event is ok. Else error occurred during event generation, e.g. data conversion error. Number of run time error. System time the error occurred. Device Number responsible for the error. Responsible bits of the device. Transaction number. **Note: Device and Bits are only information for DOLI service personal.**

8.20.1 List of Runtime errors

Error number	Error constant	Description
104	RTE_EMOVE_END	Error at end of emergency movement still active
105	RTE_CTRL_DEVIATION	Controller deviation error (Control channel see device)
201	RTE_DRIVE_OFF	Drive off or emergency off
202	RTE_E_MOVE_RQ	emergency off, emergency drive required
203	RTE_UPPER_LIMIT_SWITCH	Upper hard-limit switch active
204	RTE_LOWER_LIMIT_SWITCH	Lower hard-limit switch active
205	RTE_STOP	Drive not ready
206	RTE_DF_KEY	Drive free withdrawn by key
207	RTE_SHALT	Signal S-HALT activated
301	RTE_UPPER_LIMIT	Upper range limit exceeded
302	RTE_LOWER_LIMIT	Lower range limit exceeded

8.21 DoPESetOnOverflowHdlr

Handler for overflow messages.
An overflow may occur, if data records or messages from EDC are not processed by the application program for a longer time. The application program can adjust the number of data records, and messages when opening the link to EDC with DoPE open functions.

Minimum requirements: DoPE 2.23

Function declaration	**Description**
extern unsigned DLLAPIDoPESetOnOverflowHdlr	Function returns Error constant (DoPERR_xxxx)
DoPE_HANDLE DoPEHdl	DoPE link handle
DoPEOnOverflowHdlr Hdlr	User function to be called after overflow detection.
LPVOID lpParameter	User specific pointer.
User function for the OnOverflow event:	
unsigned CALLBACK OnOverflow	Function should return 0 (reserved for future versions)
DoPE_HANDLE DoPEHdl	DoPE link handle
int Overflowtype	0=measuring data records lost else =message lost.
LPVOID lpParameter	User specific pointer set with DoPESetOnOverflowHdlr

8.22 DoPESetOnSystemMsgHdlr

Handler for system messages.

Minimum requirements: EDC580/220, EdcApp 9133.017, DoPE 2.65

Function declaration	**Description**
extern unsigned DLLAPI DoPESetOnSystemMsgHdlr	Function returns Error constant (DoPERR_xxxx)
DoPE_HANDLE DoPEHdl	DoPE link handle
DoPEOnSystemMsgHdlr Hdlr	User function to be called for system messages.
LPVOID lpParameter	User specific pointer.
User function for the OnSystemMsg event:	
unsigned CALLBACK OnSystemMsg	Function should return 0 (reserved for future versions)
DoPE_HANDLE DoPEHdl	DoPE link handle
DoPEOnSystemMsg *SystemMsg	Pointer to received system message
LPVOID lpParameter	User specific pointer set with DoPESetOnSystemMsgHdlr
typedef struct {	
unsigned DoPError	DoPERR_NOERROR event is ok.
unsigned short MsgNumber	Number of system message
double Time	System time the message was sent.
wchar_t Text[SYSTEM_MSG_TEXT_LEN] } DoPEOnSystemMsg;	Message text

8.23 DoPESetOnDebugMsgHdlr

Handler for debug messages.

Minimum requirements: EDC580V/220V, DoPE 2.77

Function declaration	**Description**
extern unsigned DLLAPI DoPESetOnDebugMsgHdlr DoPE_HANDLE DoPEHdl DoPEOnDebugMsgHdlr Hdlr LPVOID lpParameter	Function returns Error constant (DoPERR_xxxx) DoPE link handle User function to be called for debug messages. User specific pointer.
User function for the OnDebugMsg event:	
unsigned CALLBACK OnDebugMsg DoPE_HANDLE DoPEHdl DoPEOnDebugMsg *DebugMsg LPVOID lpParameter	Function should return 0 (reserved for future versions) DoPE link handle Pointer to received debug message User specific pointer set with DoPESetOnDebugMsgHdlr
typedef struct { unsigned DoPError unsigned short MsgType double Time wchar_t Text[DEBUG_MSG_TEXT_LEN] } DoPEOnDebugMsg;	 DoPERR_NOERROR event is ok. Type of debug message System time the message was sent. Message text
Constants for MsgType DEBUG_MSG DEBUG_MSG_DATA	 General debug message Measured values debug message (typically sent every 500ms)

8.24 DoPEThreadPollHdlr

In case, the application program uses an extra thread for communication with DoPE, the DoPEThreadPollHdlr must be called periodically.

1. The application program must install a tread message queue for the thread, using the windows API-function MsgWaitForMultipleObjectsEx.
2. Open communication with a DoPE open function.
3. Install all event handlers you want to process.
4. Use MsgWaitForMultipleObjectsEx function, and wait for tread messages.
5. Call DoPEThreadPollHdlr function if MsgWaitForMultipleObjectsEx stops waiting due to the QS_POSTMESSAGE wait mask. (See Microsoft MSDN for details)

Here a “C” sample program:

```
UINT Thread( LPVOID pParam)
{
  DoPE_HANDLE DoPEHdl = NULL;
  unsigned DoPEErr;
// create a message queue for the thread
  MsgWaitForMultipleObjectsEx(
        0,              //DWORD nCount,          // number of handles in the handle array
        NULL,           //LPHANDLE pHandles,     // pointer to the object-handle array
        0,              //DWORD dwMilliseconds,  // time-out interval in milliseconds
        QS_POSTMESSAGE,//DWORD dwWakeMask       // type of input events to wait for
        MWMO_INPUTAVAILABLE //DWORD dwFlags     // wait flags
        );
DoPEErr = DoPEErr = DoPEOpenFunctionID( ... &DoPEHdl ); // open the communication link
if( DoPEErr != DoPERR_NOERROR)
//add errorhandling here
    return 0;
DoPEErr = DoPESelSetup ( DoPEHdl, 1, UserScale, NULL, NULL );
if(DoPEErr != DoPERR_NOERROR)
//add errorhandling here
// install the on data handler
  DoPEErr = DoPESetOnDataHdlr ( DoPEHdl, OnData, NULL);
  if(DoPEErr != DoPERR_NOERROR)
    //add errorhandling here
  for ( BOOL Terminate = FALSE; !Terminate; ) // thread loop
  {
    DWORD n;
    #define nCOUNT 0
    switch(n=MsgWaitForMultipleObjectsEx(
        nCOUNT,         //DWORD nCount,          // number of handles in the handle array
        NULL,           //LPHANDLE pHandles,     // pointer to the object-handle array
        1000,           //DWORD dwMilliseconds,  // time-out interval in milliseconds
        QS_POSTMESSAGE,//DWORD dwWakeMask       // type of input events to wait for
        MWMO_INPUTAVAILABLE //DWORD dwFlags     // wait flags
        ))
    {
    case WAIT_TIMEOUT:
      break;
    case WAIT_OBJECT_0 + nCOUNT:
//New input of the type specified in the dwWakeMask parameter
      // is available in the thread's input queue.
      DoPEErr=DoPEThreadPollHdlr ( DoPEHdl );
      if(DoPEErr!=DoPERR_NOERROR)
      {
//add errorhandling here
        Terminate = TRUE;
      }
      // process thread messages with PeekMessage ( &Msg, NULL, WM_USER, WM_USER, PM_REMOVE ))
      break;
    }
  }
  return 0;
}
```

8.25 Realtime version of DoPE Event Handler

For each event, there are two versions of event handlers:

- The “normal” event handler (e.g. DoPESetOnDataHdlr),
- And a real time version (e.g. DoPESetOnData**Rt**Hdlr).

The difference of these two versions is the time priority; the event handler has to be called.

For the normal priority, DoPE uses the Windows message queue, while the real time handler is called by DoPE at a high priority level.

A realtime handler must not call any sync DoPE function like DoPEPosSync, and also functions that read data from EDC like DoPERdSetupAll.

Note: There are no realtime handlers available for VB6 and VB.NET

Application programmer using the real time event handler must be aware of the restrictions for functions at a high priority level.

In most cases, the normal version of event handler is sufficient.

9 Configuration

9.1 DoPERdVersion / DoPEwRdVersion

Read version strings. There are two versions of this function, one return ASCII-Strings, another one returns wide character strings.

Minimum requirements: DoPEwRdVersion, DoPE 2.65

Function declaration	Description
extern unsigned DLLAPIDoPERdVersion DoPE_HANDLE DoPEHdl DoPEVersion *Version	Function returns Error constant (DoPERR_xxxx) DoPE link handle Version strings.
extern unsigned DLLAPIDoPEwRdVersion DoPE_HANDLE DoPEHdl DoPEwVersion *Version	Function returns Error constant (DoPERR_xxxx) DoPE link handle Version strings.
typedef struct { char PeInterface[6]; char Application [13]; char Subsy[6]; char SubsyCustVer[6]; char SubsyCustName[9]; char Bios[6]; char HwCtrl[7]; char PeInterfacePC[6]; char DpxVer[6]; char SerialNumber[17]; } DoPEVersion;	 PE interface EDC VERSION "xx.xx" EDC appl. version "xxxxxxxx.xxx" Subsystem version "xx.xx" Subsystem customer version "xx.xx" Subsystem customer name "xxxxxxxx" EDC BIOS version "xx.xx" EDC controller hardware no "xxxx.x" PE interface PC Version "xx.xx" DPX version "xx.xx" EDC serial number"xxxxxxxxxxxxxxx" Strings are zero terminated '\0'
typedef struct { wchar_t PeInterface[6]; wchar_t Application [13]; wchar_t Subsy[6]; wchar_t SubsyCustVer[6]; wchar_t SubsyCustName[9]; wchar_t Bios[6]; wchar_t HwCtrl[7]; wchar_t PeInterfacePC[6]; wchar_t DpxVer[6]; wchar_t SerialNumber[17]; } DoPEwVersion;	 PE interface EDC VERSION "xx.xx" EDC appl. version "xxxxxxxx.xxx" Subsystem version "xx.xx" Subsystem customer version "xx.xx" Subsystem customer name "xxxxxxxx" EDC BIOS version "xx.xx" EDC controller hardware no "xxxx.x" PE interface PC Version "xx.xx" DPX version "xx.xx" EDC serial number"xxxxxxxxxxxxxxx" Strings are zero terminated '\0'

9.2 DoPEwRdLanguageInfo

Read language info.

Minimum requirements: EDC580/220, EdcApp 9133.017, DoPE 2.65

Function declaration	Description
extern unsigned DLLAPIDoPEwRdLanguageInfo DoPE_HANDLE DoPEHdl DoPEwLanguageInfo *Info	Function returns Error constant (DoPERR_xxxx) DoPE link handle Pointer to storage for the language info.
typedef struct { wchar_t Name0[LANGINFO_NAME_LEN]; wchar_t Name1[LANGINFO_NAME_LEN]; wchar_t Name2[LANGINFO_NAME_LEN]; wchar_t Name3[LANGINFO_NAME_LEN]; }DoPEwLanguageInfo;	 Language name 0 (e.g. "English") Language name 1 (e.g. "German") Language name 2 (e.g. "Spanish") Language name 3 (e.g. "French")

9.3 Data Acquisition commands

9.3.1 DoPESetDataTransmissionRate (Sync)

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

Minimum requirements: DoPE 2.51

Function declaration		Description	
extern unsigned DLLAPI	DoPESetDataTransmissionRate (	Function returns Error constant (DoPERR_xxxx)	
DoPE_HANDLE	DoPEHdl	DoPE link handle	
double	Refresh	Refresh time	s
WORD	*lpusTAN	Pointer to transaction number (not for Sync. version).	

9.3.2 DoPESetSensorDataTransmissionRate (Sync)

Set time base for data acquisition for an individual measuring channel.
The default refresh time is defined in the setup data.

Minimum requirements: EDC580/220, EdcApp 9133.021, DoPE 2.66

Function declaration		Description	
extern unsigned DLLAPI	DoPESetSensorDataTransmissionRate(	Function returns Error constant (DoPERR_xxxx)	
DoPE_HANDLE	DoPEHdl	DoPE link handle	
WORD	SensorNo	Sensor Number	
Double	Refresh	Refresh time	s
WORD	*lpusTAN	Pointer to transaction number (not for Sync. version).	

9.3.3 DoPESetTime(Sync)

Set time counter to a desired value.

Minimum requirements for Mode Parameter: EDC580V/220V, EdcApp 9140.011, DoPE 2.80

Function declaration		Description	
extern unsigned DLLAPI	DoPESetTime (	Function returns Error constant (DoPERR_xxxx)	
DoPE_HANDLE	DoPEHdl	DoPE link handle	
WORD	Mode	**SETTIME_MODE_IMMEDIATE**: set time without a delay **SETTIME_MODE_MOVE**: set time at the beginning of the next movement command **SETTIME_MODE_CYCLE**: set time at the beginning of the first cycle of the next cyclic command	
double	Time	New value for Time in	s
WORD	*lpusTAN	Pointer to transaction number (not for Sync. version).	

9.3.4 DoPETransmitData(Sync)

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

Function declaration		Description	
extern unsigned DLLAPI	DoPETransmitData (	Function returns Error constant (DoPERR_xxxx)	
DoPE_HANDLE	DoPEHdl	DoPE link handle	
unsigned short	Enable	TRUE = Activate transmission FALSE = Deactivate transmission	
double	Time		
WORD	*lpusTAN	Pointer to transaction number (not for Sync. version).	

9.3.5 DoPEIntgr(Sync)

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. 100 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 100 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 (DoPESetDataTransmissionRate). It is possible to set the refresh time to 20 ms and the integration time to 100ms.

Function declaration		Description	
extern unsigned DLLAPI	DoPEIntgr (	Function returns Error constant (DoPERR_xxxx)	
DoPE_HANDLE	DoPEHdl	DoPE link handle	
unsigned short	SensorNo	Sensor Number	
double	Intgr	Integration time for analogue measuring channels in sec. The limits of integration time depend on the selected time basefor the speed control loop cycle time. (see machine set-up data) The minimum time is1 x (time base for the speed control) The maximum time is 100 x (time base for the speed control)	s
WORD	*lpusTAN	Pointer to transaction number (not for Sync. version).	

9.3.6 DoPECtrlTestValues

Enable or disable the controller Test values in the DoPEData record.
The three Test values are configured to:

Test1: Command
Test2: Feedback
Test3: Output

The test values are disabled by default!

Minimum requirements: DoPE 2.23

Function declaration		Description	
extern unsigned DLLAPI	DoPECtrlTestValues (	Function returns Error constant (DoPERR_xxxx)	
DoPE_HANDLE	DoPEHdl	DoPE link handle	
unsigned short	State	TRUE: enables FALSE: disables the controller test values in the DoPEData record.	

9.3.7 DoPESetCycleMode

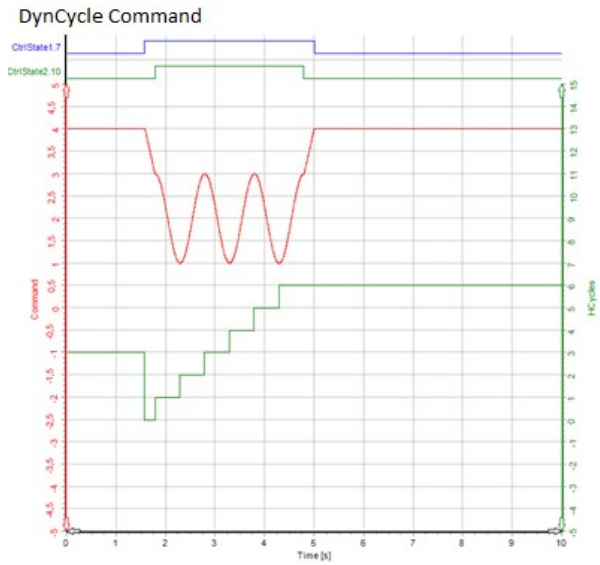
Enable/disable unified cycle handling for cyclic movement commands.

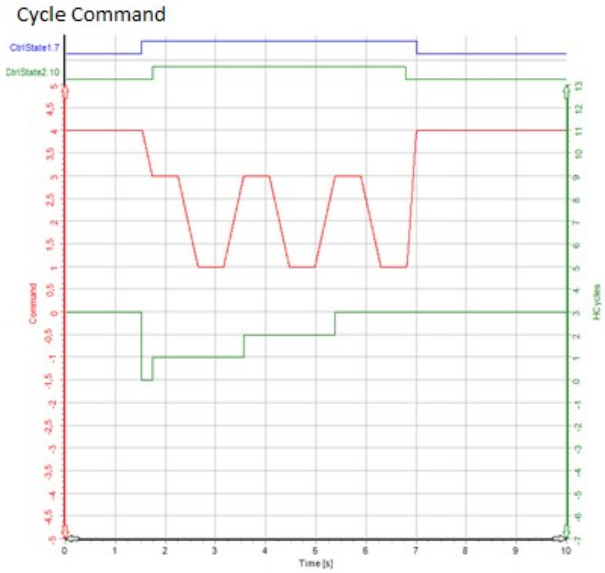
Minimum requirements: EDC580V/220V, EdcApp 9140.011, DoPE 2.80

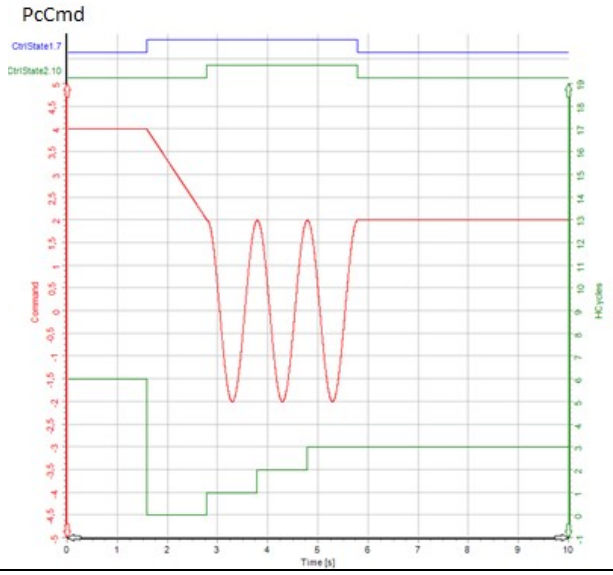
Function declaration		Description	
extern unsigned DLLAPI	DoPESetCycle (	Function returns Error constant (DoPERR_xxxx)	
DoPE_HANDLE	DoPEHdl	DoPE link handle	
unsigned	Enable	!= 0 enables ==0 disables unified cycle handling for cyclic (**default**)	

Unified cycle handling

- Bit7 of CtrlState1 signals ‘movement command is active’
- Bit10 of CtrlState2 signals ‘cycles active’
- Cycles count resets to zero at the beginning of the movement command (rising edge of CtrlState1.7)
- Cycles count increases at the **start of each (half)cycle**
- ‘cycles active’ resets to low at the end of the last cycle



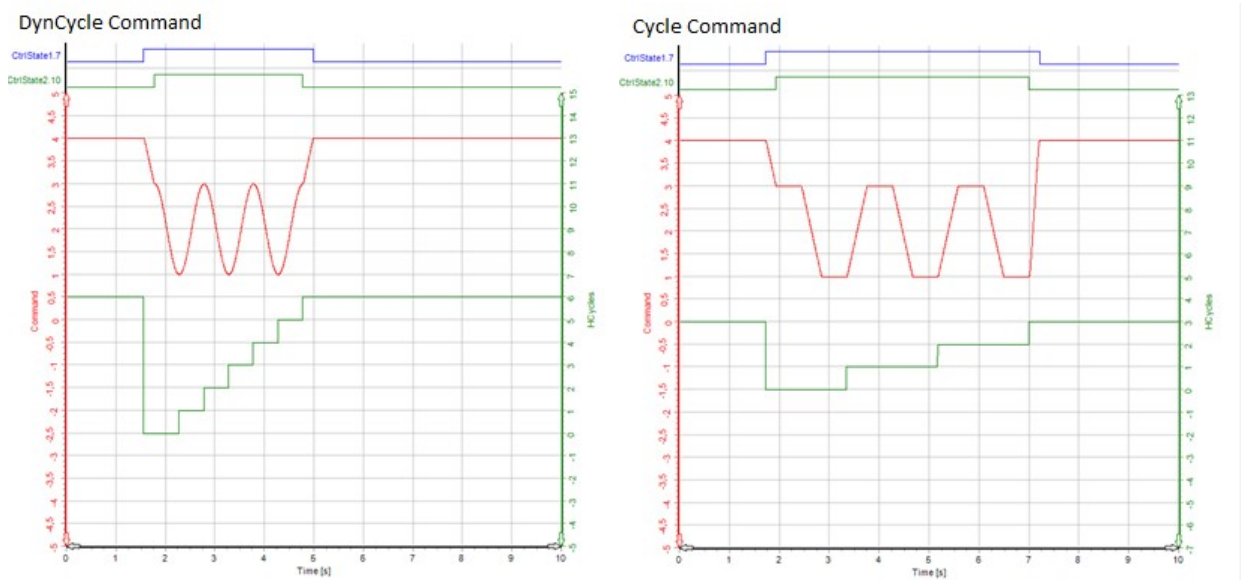




No unified cycle handling

Per default the DynCycle and Cycle commands handle the cycle counter different than PcCmd.

- Cycles count resets to zero at the beginning at the movement command (CtrlState1.7 rising edge).
- Cycles count increases at the **end of each (half)cycle**.
- 'cycles active' resets to low at the end of the last cycle



(PcCmd has allways unified cycle handling)

9.4 Safety Shield

In order to protect the user from injures, the EDC contains a shield control. Two different shield types are supported, simple and lockable shields. For more information see the EDC Installation Manual.

9.4.1 DoPEShieldLimit(Sync)

Activate the shield supervision.

Function declaration			Description	
extern unsigned DLLAPI	DoPEShieldLimit (		Function returns Error constant (DoPERR_xxxx)	
DoPE_HANDLE	DoPEHdl		DoPE link handle	
unsigned short	SensorNo		Sensor Number	
double	UprLock		For all load values higher than this the shield must be closed and will be locked.	Unit
double	UprUnLock		Upper limit for unlock the shield. (e.g. 10 % lower than UprLock)	Unit
double	LwrUnLock		Lower limit for unlock the shield (e.g. 10 % higher than LwrLock).	Unit
double	LwrLock		For all load values lower than this the shield must be closed and willbe locked.	Unit
unsigned short	CtrlLimit		Control mode for speed limit (normally X-head)	
double	SpeedLimit		Maximum allowed speed if shield is open.	m/s
unsigned short	CtrlAction		Control mode for selected action	
unsigned short	Action		This action will be activated if the limits are reached FB_HALT: Halt machine in 'CtrlAction' control mode FB_STOP: Switch off closed loop control. (Use this action for machines without position sensor)	
WORD	*lpusTAN	(not for Sync. version)	Pointer to transaction number (not for Sync. version).	

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 affect these vales!

9.4.2 DoPEShieldEnable(Sync)

Activate / Deactivate the shield supervision.

Minimum requirements: EDC120/60, 9123.015, DoPE 2.65

Function declaration			Description	
extern unsigned DLLAPI	DoPEShieldEnable (		Function returns Error constant (DoPERR_xxxx)	
DoPE_HANDLE	DoPEHdl		DoPE link handle	
unsigned	Enable		!= 0 enables 0 disables shield supervision	
WORD	*lpusTAN	(not for Sync. version)	Pointer to transaction number (not for Sync. version).	

9.4.3 DoPEShieldDisable(Sync)

Deactivate the shield supervision.

Function declaration			Description	
extern unsigned DLLAPI	DoPEShieldDisable (		Function returns Error constant (DoPERR_xxxx)	
DoPE_HANDLE	DoPEHdl		DoPE link handle	
WORD	*lpusTAN	(not for Sync. version)	Pointer to transaction number (not for Sync. version).	

9.4.4 DoPEShieldLock(Sync)

Lock or unlock the shield.

Function declaration			Description	
extern unsigned DLLAPI	DoPEShieldLock (		Function returns Error constant (DoPERR_xxxx)	
DoPE_HANDLE	DoPEHdl		DoPE link handle	
WORD	State		State of Shield lock. (TRUE = Lock, FALSE = UNLOCK)	
WORD	*lpusTAN	(not for Sync. version)	Pointer to transaction number (not for Sync. version).	

9.5 Channel supervision

9.5.1 DoPESetCheck(Sync)

9.5.2 DoPESetCheckX(Sync)

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, the current supervision and all other supervisions (which are not marked with the CHK_NOCLEAR bit) are deactivated.

Minimum requirements:

DoPESetCheckX(Sync)	EDC5/25/100, EdcApp 9056.o, DoPE 2.12
ACTION_UP/ACTION_DOWN	EDC5/25/100, EdcApp 9056.o, DoPE 2.21
ACTION_DRIVE_OFF	EDC60/120, EdcApp 9132.011 / 9133.006, DoPE 2.58

Function declaration			Description	
extern unsigned DLLAPI	DoPESetCheck(X) (		Function returns Error constant (DoPERR_xxxx)	
DoPE_HANDLE	DoPEHdl		DoPE link handle	
unsigned short	CheckId		ID of this check, use the CheckId constants (CHK_ID0 ... CHK_ID9)	
unsigned short	SensorNo		Sensor to be supervised	
double	Limit		Limit to be supervised	Unit
double	Tare (only for DoPESetCheckX)		Tare to be subtracted for CHK_PERCENT_MAX/MIN mode	Unit
unsigned short	Mode		Mode, how the limit is detected. (see below)	
unsigned short	Action		This action will be activated if the check hits. (see below)	
unsigned short	Ctrl		Control mode for selected action(CTRL_xxx)	
double	Acc		Acceleration	Unit/s²
double	Speed		Maximum speed	Unit/s
double	Dec		Deceleration	Unit/s²
double	Destination		Final destination	Unit
WORD	*lpusTAN	(not for Sync. version)	Pointer to transaction number (not for Sync. version).	

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

Possible actions:

ACTION_HALT	HALT with default deceleration
ACTION_HALT_A	HALT with specified deceleration
ACTION_POS	Go to a position with default deceleration

ACTION_POS_A	Go to a position with specified deceleration
ACTION_XPCONT	Change control mode, go on with current speed
ACTION_STOP	Switch off closed loop control. (Use this action for machines without position sensor)
ACTION_NOACTION	No action, only message to host
ACTION_SETOL	Set command output (only in open loop structure)
ACTION_SHALT	Immediate Halt in X-head position control
ACTION_UP	Move Up with specified speed
ACTION_DOWN	Move Down with specified speed
ACTION_DRIVE_OFF	Switch drive OFF

9.5.3 DoPEClrCheck(Sync)

Deactivate a measuring channel supervision				
Function declaration			**Description**	
extern unsigned DLLAPI DoPE_HANDLE unsigned short WORD	DoPEClrCheck (DoPEHdl CheckId *lpusTAN	(not for Sync. version)	Function returns Error constant (DoPERR_xxxx) DoPE link handle ID of this check, use the CheckId constants (CHK_ID0 ... CHK_ID9) Pointer to transaction number (not for Sync. version).	

9.5.4 DoPESetCheckLimit(Sync)

Activate 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 depends on the device selected in EDC-Set-up.

4INC/4IO	**XxxB**	**Bit 0, Pin 32**
EDC100	**X2**	**Bit 14, Pin 6**
EDC60/120/220/222/580	**X2**	**Bit 3, Pin 8**
EDCs with firmware 9133.008 or newer		**see DoPEIOGrip configuration**

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 affect these values!

Function declaration			**Description**	
extern unsigned DLLAPI DoPE_HANDLE unsigned short double double double double WORD	DoPESetCheckLimit (DoPEHdl SensorNo UprLimitSet UprLimitReset LwrLimitReset LwrLimitSet *lpusTAN	(not for Sync. version)	Function returns Error constant (DoPERR_xxxx) DoPE link handle Defines the sensor to be supervised. For all values above, the digital output will be set. For all values below, the digital output will be reset. For all values above, the digital output will be reset For all values below, the digital output will be set. Pointer to transaction number (not for Sync. version).	

9.5.5 DoPEClrCheckLimit(Sync)

Deactivate check limit function. The digital output will be set inactive.				
Function declaration			**Description**	
extern unsigned DLLAPI DoPE_HANDLE WORD	DoPEClrCheckLimit (DoPEHdl *lpusTAN	(not for Sync. version)	Function returns Error constant (DoPERR_xxxx) DoPE link handle Pointer to transaction number (not for Sync. version).	

9.5.6 DoPESetCheckLimitIO(Sync)

Set / reset measuring channel supervision IO.				
Minimum requirements: EDC120/60, EdcApp 9123.011, DoPE 2.59				
Function declaration			**Description**	
extern unsigned DLLAPI DoPE_HANDLE WORD WORD	DoPESetCheckLimitIO (DoPEHdl Value *lpusTAN	(not for Sync. version)	Function returns Error constant (DoPERR_xxxx) DoPE link handle 0 = reset, 1 = Set Pointer to transaction number (not for Sync. version).	

9.6 Controller Parameter

All controller parameter can be modified at any time!

9.6.1 DoPEPosPID (Sync)

Set parameter for closed loop position controller

Minimum requirements: DoPE 2.23

Function declaration			Description
extern unsigned DLLAPI	DoPEPosPID (		Function returns Error constant (DoPERR_xxxx)
DoPE_HANDLE	DoPEHdl		DoPE link handle
unsigned short	MoveCtrl		Control mode (CTRL_xxx)
unsigned long	P		Proportional gain of the position controller
unsigned short	I		Integration time constant
unsigned short	D		Derivative time constant
WORD	*IpusTAN	(not for Sync. version)	Pointer to transaction number (not for Sync. version).

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!

9.6.2 DoPERdPosPID

Get parameter for closed loop position controller

Minimum requirements: EDC580/220, EdcApp 9133.29, DoPE 2.71

Function declaration			Description
extern unsigned DLLAPI	DoPEPosPID (		Function returns Error constant (DoPERR_xxxx)
DoPE_HANDLE	DoPEHdl		DoPE link handle
unsigned short	MoveCtrl		Control mode (CTRL_xxx)
unsigned	HighPressure		!=0 get parameter for high pressure 0 get parameter for low pressure
unsigned long	*P		Pointer to Proportional gain of the position controller
unsigned short	*I		Pointer to Integration time constant
unsigned short	*D		Pointer to Derivative time constant
WORD	*IpusTAN	(not for Sync. version)	Pointer to transaction number (not for Sync. version).

9.6.3 DoPEWrPosPID (Sync)

Set parameter for closed loop position controller

Minimum requirements: EDC580/220, EdcApp 9133.29, DoPE 2.71

Function declaration			Description
extern unsigned DLLAPI	DoPEPosPID (		Function returns Error constant (DoPERR_xxxx)
DoPE_HANDLE	DoPEHdl		DoPE link handle
unsigned short	MoveCtrl		Control mode (CTRL_xxx)
unsigned	HighPressure		!=0 set parameter for high pressure 0 set parameter for low pressure
unsigned long	P		Proportional gain of the position controller
unsigned short	I		Integration time constant
unsigned short	D		Derivative time constant
WORD	*IpusTAN	(not for Sync. version)	Pointer to transaction number (not for Sync. version).

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!

9.6.4 DoPESpeedPID(Sync)

Set parameter for closed loop speed controller.			
Minimum requirements: DoPE 2.23			
Function declaration		**Description**	
extern unsigned DLLAPI	DoPESpeedPID (	Function returns Error constant (DoPERR_xxxx)	
DoPE_HANDLE	DoPEHdl	DoPE link handle	
unsigned short	MoveCtrl	Control mode (CTRL_xxx)	
unsigned long	P	Proportional gain of the speed controller	
unsigned short	I	Integration time constant	
unsigned short	D	Derivative time constant	
WORD	*lpusTAN (not for Sync. version)	Pointer to transaction number (not for Sync. version).	

9.6.5 DoPERdSpeedPID

Get parameter for closed loop speed controller			
Minimum requirements: EDC580/220, EdcApp 9133.029, DoPE 2.71			
Function declaration		**Description**	
extern unsigned DLLAPI	DoPEPosPID (	Function returns Error constant (DoPERR_xxxx)	
DoPE_HANDLE	DoPEHdl	DoPE link handle	
unsigned short	MoveCtrl	Control mode (CTRL_xxx)	
unsigned	HighPressure	!=0 get parameter for high pressure 0 get parameter for low pressure	
unsigned long	*P	Pointer to Proportional gain of the speed controller	
unsigned short	*I	Pointer to Integration time constant	
unsigned short	*D	Pointer to Derivative time constant	

9.6.6 DoPEWrSpeedPID (Sync)

Set parameter for closed loop speed controller			
Minimum requirements: EDC580/220, EdcApp 9133.029, DoPE 2.71			
Function declaration		**Description**	
extern unsigned DLLAPI	DoPEPosPID (	Function returns Error constant (DoPERR_xxxx)	
DoPE_HANDLE	DoPEHdl	DoPE link handle	
unsigned short	MoveCtrl	Control mode (CTRL_xxx)	
unsigned	HighPressure	!=0 set parameter for high pressure 0 set parameter for low pressure	
unsigned long	P	Proportional gain of the speed controller	
unsigned short	I	Integration time constant	
unsigned short	D	Derivative time constant	
WORD	*lpusTAN (not for Sync. version)	Pointer to transaction number (not for Sync. version).	
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!			

9.6.7 DoPEFeedForward (Sync)

Set feed forward parameter.

Minimum requirements: EDC580/220, EdcApp 9133.008, DoPE 2.59

Function declaration		Description	
extern unsigned DLLAPI	DoPEFeedForward (	Function returns Error constant (DoPERR_xxxx)	
DoPE_HANDLE	DoPEHdl	DoPE link handle	
WORD	MoveCtrl	Control mode (CTRL_xxx)	
WORD	SpeedFFP	FeedForward for Speed in % of current speed	
WORD	PosDelay	Delay for Position command	0.1ms
		Acceleration feed forward for future use!	
WORD	AccFFP	FeedForward for Acc in % of current acceleration	
WORD	SpeedDelay	Delay for Speed command	0.1ms
WORD	*IpusTAN (not for Sync. version)	Pointer to transaction number (not for Sync. version).	

9.6.8 DoPERdFeedForward

Getfeed forward parameter

Minimum requirements: EDC580/220, EdcApp 9133.029, DoPE 2.71

Function declaration		Description	
extern unsigned DLLAPI	DoPEPosPID (	Function returns Error constant (DoPERR_xxxx)	
DoPE_HANDLE	DoPEHdl	DoPE link handle	
unsigned short	MoveCtrl	Control mode (CTRL_xxx)	
unsigned	HighPressure	!=0 get parameter for high pressure 0 get parameter for low pressure	
WORD	*SpeedFFP	Pointer to FeedForward for Speed in % of current speed	
WORD	*PosDelay	Pointer to Delay for Position command	
		Acceleration feed forward for future use!	
WORD	*AccFFP	Pointer to FeedForward for Acc in % of current acceleration	
WORD	*SpeedDelay	Pointer to Delay for Speed command	

9.6.9 DoPEWrFeedForward (Sync)

Set feed forward parameter

Minimum requirements: EDC580/220, EdcApp 9133.029, DoPE 2.71

Function declaration		Description	
extern unsigned DLLAPI	DoPEPosPID (	Function returns Error constant (DoPERR_xxxx)	
DoPE_HANDLE	DoPEHdl	DoPE link handle	
unsigned short	MoveCtrl	Control mode (CTRL_xxx)	
unsigned	HighPressure	!=0 set parameter for high pressure 0 set parameter for low pressure	
WORD	SpeedFFP	FeedForward for Speed in % of current speed	
WORD	PosDelay	Delay for Position command	
		Acceleration feed forward for future use!	
WORD	AccFFP	FeedForward for Acc in % of current acceleration	
WORD	SpeedDelay	Delay for Speed command	
WORD	*IpusTAN (not for Sync. version)	Pointer to transaction number (not for Sync. version).	

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!

9.6.10 DoPEOptimizeFeedForward (Sync)

Optimize feed forward parameter.
For optimization you specify e.g. a cosine cycle and the optimization function will find SpeedFFP and PosDelay values.

Minimum requirements: EDC580/220, EdcApp 9133.008, DoPE 2.59

Function declaration			Description	
extern unsigned DLLAPI	DoPEOptimizeFeedForward(		Function returns Error constant (DoPERR_xxxx)	
DoPE_HANDLE	DoPEHdl		DoPE link handle	
unsigned short	MoveCtrl		Control mode (CTRL_xxx)	
unsigned short	Mode		Waveform: OPTIMIZE_COSINE, OPTIMIZE_TRIANGLE, or OPTIMIZE_RECTANGLE	
double	Offset		Offset	Unit
double	Amplitude		Amplitude	Unit
double	Frequency		Frequency	Hz
WORD	*lpusTAN	(not for Sync. version)	Pointer to transaction number (not for Sync. version).	

9.6.11 DoPEPosFeedForward(Sync)

Set feed forward gain of closed loop controller.

Minimum requirements: EDC580/220, EdcApp 9133.08, DoPE 2.23

Function declaration			Description	
extern unsigned DLLAPI	DoPEPosFeedForward (		Function returns Error constant (DoPERR_xxxx)	
DoPE_HANDLE	DoPEHdl		DoPE link handle	
unsigned short	MoveCtrl		Control mode (CTRL_xxx)	
unsigned long	P		Gain for feed forward control	
WORD	*lpusTAN	(not for Sync. version)	Pointer to transaction number (not for Sync. version).	

9.6.12 DoPECurrentPID(Sync)

Set parameter for the external current closed loop controller.

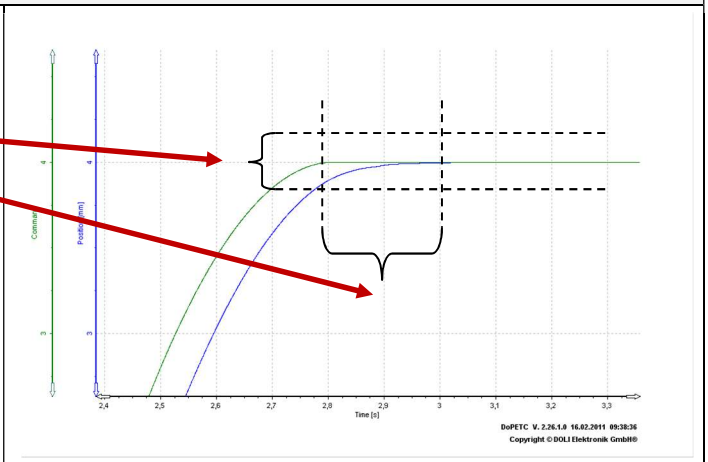
Minimum requirements:
EDC580/220 with DC160, DC320, DC1100 and DC2500, EdcApp 9133.06, DoPE 2.57

Function declaration			Description	
extern unsigned DLLAPI	DoPECurrentPID (		Function returns Error constant (DoPERR_xxxx)	
DoPE_HANDLE	DoPEHdl		DoPE link handle	
unsigned short	MoveCtrl		Control mode (CTRL_xxx)	
unsigned long	P		Proportional gain of the current controller	
unsigned short	I		Integration time constant	
unsigned short	D		Derivative time constant	
WORD	*lpusTAN	(not for Sync. version)	Pointer to transaction number (not for Sync. version).	

9.6.13 DoPEDestWnd(Sync)

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 the specified destination window.
If the actual value reaches the destination window within the specified time the message “CMSG_POS” is sent to the application program.
If it does not reach it, the message “CMSG_POS_ERR” is sent.
By default, the destination limit is set to:

For absolute sensors	WndSize = 0.2% of the nominal sensor range.
For incremental position sensors	WndSize = 0.05 mm
For incremental extension sensors	WndSize = 0.005 mm
For all sensors	WndTime = 0.5 s.



Function declaration			Description	
extern unsigned DLLAPI	DoPEDestWnd (		Function returns Error constant (DoPERR_xxxx)	
DoPE_HANDLE	DoPEHdl		DoPE link handle	
unsigned short	MoveCtrl		Control mode (CTRL_xxx)	
double	WndSize		Size of destination window	Unit
double	WndTime		Time for destination window	s
WORD	*lpusTAN	(not for Sync. version)	Pointer to transaction number (not for Sync. version).	

9.6.14 DoPESft(Sync)

Definitions of limits supervised by software (softend).
If Reaction is defined as REACT_ACTION all moving commands will limit destinations to this softends.
In contrast to the range limits of the measurement channels, softends are working limits that can be changed at any time.
By default, all softends are set to the range limits and REACT_STATUS.
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 affect these vales!
Note: The function of softend may be different with EDC-firmware version before 9133.010

Function declaration				Description		
extern unsigned DLLAPI	DoPESft(			Function returns Error constant (DoPERR_xxxx)		
DoPE_HANDLE	DoPEHdl			DoPE link handle		
unsigned short	MoveCtrl			Control mode (CTRL_xxx)		
double	UpperSft			Upper soft limit		Unit
double	LowerSft			Lower soft limit		Unit
unsigned short	Reaction			REACT_STATUS	No reaction if softend is reached, only status bits are set	
				REACT_ACTION	Halt in X-head position control if softend is reached	
WORD	*lpusTAN	(not for Sync. version)		Pointer to transaction number (not for Sync. version).		

9.6.15 DoPECtrlError(Sync)

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.

Function declaration				Description		
extern unsigned DLLAPI	DoPECtrlError(			Function returns Error constant (DoPERR_xxxx)		
DoPE_HANDLE	DoPEHdl			DoPE link handle		
unsigned short	MoveCtrl			Control mode (CTRL_xxx)		
double	Error			Maximum error signal		Unit
unsigned short	Reaction			Reaction if error is reached:		Unit
				REACT_STATUS	No reaction if softend is reached, only status bits are set	
				REACT_ACTION	If CTRL_POSEmergency off. Else Halt in X-head position control	
WORD	*lpusTAN	(not for Sync. version)		Pointer to transaction number (not for Sync. version).		

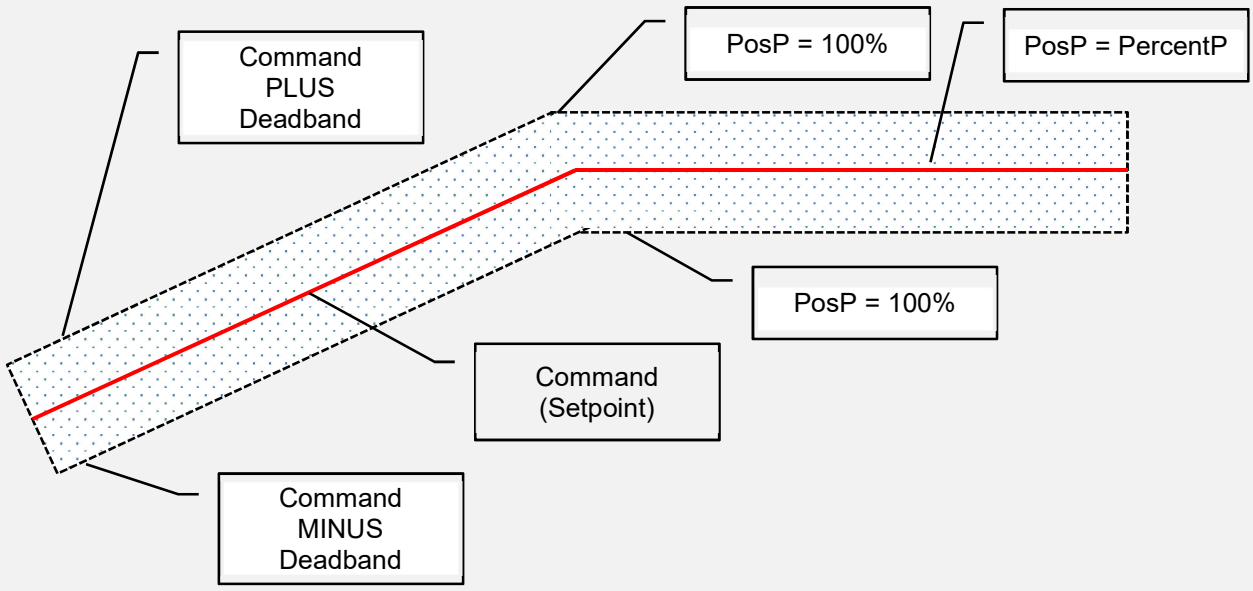
9.6.16 DoPECtrlSpeedTimeBase(Sync)

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. 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.

Function declaration				Description	
extern unsigned DLLAPI	DoPECtrlSpeedTimeBase(			Function returns Error constant (DoPERR_xxxx)	
DoPE_HANDLE	DoPEHdl			DoPE link handle	
double	Time			Time base in for speed calculation Note: for EdcApp versions up to 9140.015: internally only 1, 2, 4, 8, 16, 32, 64, 128 x position controller time base will be used!	s
WORD	*lpusTAN	(not for Sync. version)		Pointer to transaction number (not for Sync. version).	

9.6.17 DoPEDeadbandCtrl(Sync)

Set parameter for error deadband controller.
A deadband can be used to reduce, or eliminate a limit cycle due to friction.
Inside an area “setpoint plus Deadband and setpoint minus Deadband” the gain of position controller is continuously reduced from PosP = 100% to a minimum value (PercentP of PosP = 100%).
Note: a well adjusted feed forward is needed for a good function of Deadband control!



Minimum requirements: EDC580V/220V, EdcApp 9140.009, DoPE 2.79

Function declaration		Description	
extern unsigned DLLAPI	DoPEDeadbandCtrl (	Function returns Error constant (DoPERR_xxxx)	
DoPE_HANDLE	DoPEHdl	DoPE link handle	
unsigned short	MoveCtrl	Control mode (CTRL_xxx)	
double	Deadband	Width of error deadband around setpoint	
unsigned short	PercentP)	Smallest P inside deadband PercentP range is 0..100% 100% disables error deadband controller 0% sets the PosP to the smallest value (1)	[%PosP]

9.6.18 DoPERdCtrlParameter

Read closed loop controller parameter

Minimum requirements: EDC580/220, EdcApp 9133.008, DoPE 2.59
Deadband, PercentP EDC580V/220V, EdcApp 9140.009, DoPE 2.79

Function declaration		Description	
extern unsigned DLLAPI	DoPERdCtrlParameter (	Function returns Error constant (DoPERR_xxxx)	
DoPE_HANDLE	DoPEHdl	DoPE link handle	
unsigned short	MoveCtrl	Control mode (CTRL_xxx)	
DoPECtrlParameter	*CtrlParameter)	Pointer for DoPECtrlParameter structure	
typedef struct Controller parameter			
{// **Controller parameter**			
unsigned long	PosP	Pos. contr. P: gain	[No]

unsigned short	PosI	Pos. contr. I: time constant	[No]
unsigned short	PosD	Pos. contr. D: time constant	[No]
unsigned long	PosFFP	Pos. Feed Forward P	[No]
unsigned long	SpeedP	Speed contr. P: gain	[No]
unsigned short	SpeedI	Speed contr. I: time constant	[No]
unsigned short	SpeedD	Speed contr. D: time constant	[No]
unsigned short	SpeedFFP	Speed feed forward	[No]
unsigned short	PosDelay	Delay for Command	[No]
unsigned short	AccFFP	Acceleration contr. P: gain	[No]
unsigned short	SpeedDelay	Delay for SpeedCommand	[No]
double	Acceleration	default acceleration	[Unit/s²]
double	Speed	speed limit	[Unit/s]
double	Deviation	Max. deviation of controller	[Unit]
unsigned short	DevReaction	Reaction if deviation exceeded	[No]
double	DestinationWnd	Size of 1/2 destination window	[Unit]
double	DestinationTime	Time until controlled channel must reach destination window	s
double	UpperSoftEnd	Upper softend	[Unit]
double	LowerSoftEnd	Lower softend	[Unit]
unsigned short	SoftEndReaktion	Reaction if softend is reached	[No]
// numerical limitations for acceleration and speed parameters			
double	MinAcceleration	minimum acceleration	[Unit/s²]
double	MaxAcceleration	maximum acceleration	[Unit/s²]
double	MinDeceleration	minimum deceleration	[Unit/s²]
double	MaxDeceleration	maximum deceleration	[Unit/s²]
double	MinSpeed	minimum speed	[Unit/s]
double	MaxSpeed	maximum speed	[Unit/s]
// error dead band controller parameter			
double	Deadband	Error deadband around setpoint	[Unit]
unsigned short	PercentP	Smallest P inside deadband	[%PosP]
} DoPECtrlParameter			

9.6.19 DoPESetNominalAccSpeed(Sync)

Set nominal values for the position generator.			
Minimum requirements: EDC220/580, EdcApp 9133.008, DoPE 2.82			
Function declaration		**Description**	
extern unsigned DLLAPI DoPE_HANDLE Unsigned short double double WORD	DoPESetNominalAccSpeed (DoPEHdl MoveCtrl Acc Speed * lpusTAN)	Function returns Error constant (DoPERR_xxxx) DoPE link handle Control mode (CTRL_xxx) Acceleration Nominal speed Pointer to transaction number (not for Sync. version).	

9.7 Calibration

9.7.1 DoPECal(Sync)

Compensate drifts (zero and gain) of the measuring channel. It takes about 0.5 s to compensate these drifts. Note: During compensation the sensor is not measured!!!			
Function declaration		**Description**	
extern unsigned DLLAPI DoPE_HANDLE unsigned short WORD	DoPECal (DoPEHdl SensorBits *lpusTAN (not for Sync. version)	Function returns Error constant (DoPERR_xxxx) DoPE link handle Bit 0 ... 15 define the sensor Number to be calibrated Bit 0 = 1 Calibrate Sensor 0 Bit 1 = 1 Calibrate Sensor 1 ... Pointer to transaction number (not for Sync. version).	

9.7.2 DoPEZeroCal(Sync)

Compensate only zero offset drift. It takes about 0.2 s to compensate zero offset drift. Note: During compensation the sensor is not measured!!!			
Function declaration		**Description**	
extern unsigned DLLAPI DoPE_HANDLE unsigned short WORD	DoPEZeroCal (DoPEHdl SensorBits *lpusTAN (not for Sync. version)	Function returns Error constant (DoPERR_xxxx) DoPE link handle Bit 0 ... 15 define the sensor Number to be calibrated Bit 0 = 1 Calibrate Sensor 0 Bit 1 = 1 Calibrate Sensor 1 ... Pointer to transaction number (not for Sync. version).	

9.8 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 grips. The Basic-Tare value is stored in the machine set-up inside EDC. After switching mains power off / on, the Basic-Tare value is still valid.

The **DoPESetTare** function can be used at any time. This tare value will be lost after reset.

9.8.1 DoPESetBasicTare(Sync)

Set basic tare of the measuring channel.
Note: This function clears the ordinary tare value

Function declaration		Description	
extern unsigned DLLAPI	DoPESetBasicTare (	Function returns Error constant (DoPERR_xxxx)	
DoPE_HANDLE	DoPEHdl	DoPE link handle	
unsigned short	SensorNo	Sensor number	
unsigned short	Mode	BASICTARE_SET: BasicTare represents the desired measuring value. This is useful to set crosshead position for systemswith encoder. BASICTARE_SUBTRACT: BasicTare will be subtracted. This is useful to compensate the weight of grips.	
double	BasicTare	Value for BasicTare	Unit
WORD	*lpusTANFirst	Pointer to transaction number (not for Sync. version).	
WORD	*lpusTANLast		

9.8.2 DoPESetTare

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

Function declaration		Description	
extern unsigned DLLAPI	DoPESetTare (	Function returns Error constant (DoPERR_xxxx)	
DoPE_HANDLE	DoPEHdl	DoPE link handle	
unsigned short	SensorNo	Sensor number	
double	Tare	Value to be subtracted	Unit

9.8.3 DoPEGetBasicTare

Read basic tare value of the measuring channel.

Function declaration		Description	
extern unsigned DLLAPI	DoPEGetBasicTare (	Function returns Error constant (DoPERR_xxxx)	
DoPE_HANDLE	DoPEHdl	DoPE link handle	
unsigned short	SensorNo	Sensor number	
double	*BasicTare	Pointer for BasicTare	Unit

9.8.4 DoPEGetTare

Read tare value of the measuring channel.

Function declaration		Description	
extern unsigned DLLAPI	DoPEGetTare (	Function returns Error constant (DoPERR_xxxx)	
DoPE_HANDLE	DoPEHdl	DoPE link handle	
unsigned short	SensorNo	Sensor number	
double	*Tare	Pointer for Tare	Unit

9.9 Reference signal handling of incremental sensors

Incremental sensors like encoder or linear gauges may use the build in reference to set the counter-value. Encoders 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:

- These functions are not supported by EDC5/25/100.
- The following hardware interfaces support this function:
  4INC(channel A and C), 2INC (channel A), X7 on EDC220/222/580 or newer.

9.9.1 DoPESetRefSignalMode

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.

Minimum requirements: EDC120/60, EdcApp 9123.i, DoPE 2.23

Function declaration	Description
extern unsigned DLLAPI DoPESetRefSignalMode (DoPE_HANDLE DoPEHdl unsigned short SensorNo unsigned short Mode WORD * lpusTAN)	Function returns Error constant (DoPERR_xxxx) DoPE link handle Sensor Number Mode reference signal messages will be reported: REFSIG_NON: never REFSIG_ON: always REFSIG_ONCE: only once Pointer to transaction number (not for Sync. version).
typedef struct DoPERefSignalMsg/ { unsigned short MsgId; double Time; unsigned short SensorNo; double Position; } DoPERefSignalM;	 ID of message System time for the message Control mode of position Position

9.9.2 DoPESetRefSignalTare

Set the measuring channel to the tare value at the next occurrence of the reference signal.

Minimum requirements: EDC120/60, EdcApp 9123.i, DoPE 2.23

Function declaration	Description
extern unsigned DLLAPI DoPESetRefSignalTare (DoPE_HANDLE DoPEHdl unsigned short SensorNo unsigned short Mode double Tare WORD * lpusTAN)	Function returns Error constant (DoPERR_xxxx) DoPE link handle Sensor Number REFSIG_TARE: set the measuring channel to the tare value REFSIG_NON: don't affect the measuring channel Tare value for the measuring channel. This tare value will be stored in EDC's non volatileBasicTarememory.This function clears the current basic tare and ordinary tare valueat the next occurrence of the reference signal. Pointer to transaction number (not for Sync. version).

9.10 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.

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

9.10.1 DoPEConfCMcSpeed(Sync)

Configure calculated speed to measuring data record.
A 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!

Function declaration		Description	
extern unsigned DLLAPI	DoPEConfCMcSpeed (	Function returns Error constant (DoPERR_xxxx)	
DoPE_HANDLE	DoPEHdl	DoPE link handle	
unsigned short	CalculatedSensorNo	Sensor number for the calculated speed value(SENSOR_4 to SENSOR_15)	
unsigned short	SensorNo	Sensor Number of the physical channel for speed calculation	
double	IntegrationTime	Integration time (only for analogue channels)	s
double	Timebase	Time base for speed calculation (maximum 2.56 sec)	s
WORD	* lpusTANFirst	Pointer to transaction number (not for Sync. version).	
WORD	* lpusTANLast)		

9.10.2 DoPEConfCMcCommandSpeed(Sync)

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!

Function declaration		Description	
extern unsigned DLLAPI	DoPEConfCMcCommandSpeed(	Function returns Error constant (DoPERR_xxxx)	
DoPE_HANDLE	DoPEHdl	DoPE link handle	
unsigned short	CalculatedSensorNo	Sensor number for the calculated speed value(SENSOR_4 to SENSOR_15)	
double	Timebase	Time base for speed calculation (maximum 2.56 sec)	s
WORD	* lpusTANFirst	Pointer to transaction number (not for Sync. version).	
WORD	* lpusTANLast)		

9.10.3 DoPEConfCMcGradient(Sync)

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!

Function declaration		Description	
extern unsigned DLLAPI	DoPEConfCMcGradient (	Function returns Error constant (DoPERR_xxxx)	
DoPE_HANDLE	DoPEHdl	DoPE link handle	
unsigned short	CalculatedSensorNo	Sensor number for the calculated speed value(SENSOR_4 to SENSOR_15)	
unsigned short	DividentSensorNo	Sensor Number of the dividend	
unsigned short	DivisorSensorNo	Sensor Number of the divisor	
double	IntegrationTime	Integration time for both dividend and divisor (only for analogue channels)	s
double	Timebase	Time base for speed calculation (maximum 2.56 sec)	s
WORD	* lpusTANFirst	Pointer to transaction number (not for Sync. version).	
WORD	* lpusTANLast)		

9.10.4 DoPEClearCMc(Sync)

Clear calculated measuring channel

Function declaration		Description	
extern unsigned DLLAPI	DoPEClearCMc (	Function returns Error constant (DoPERR_xxxx)	
DoPE_HANDLE	DoPEHdl	DoPE link handle	
unsigned short	CalculatedSensorNo	Sensor number for the calculated speed value(SENSOR_4 to SENSOR_15)	
WORD	* lpusTANFirst	Pointer to transaction number (not for Sync. version).	
WORD	* lpusTANLast)		

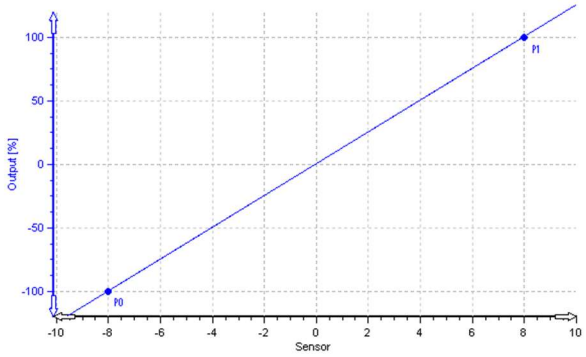
9.10.5 DoPEMc2Output(Sync)

Output of a measured value to an analogue output channel. Any measured channel may be scaled and via a DAC converted to an analogue signal.
Attention: MC2OUT_MODE_3POINTS is available only for EDC580V/220V (and above)

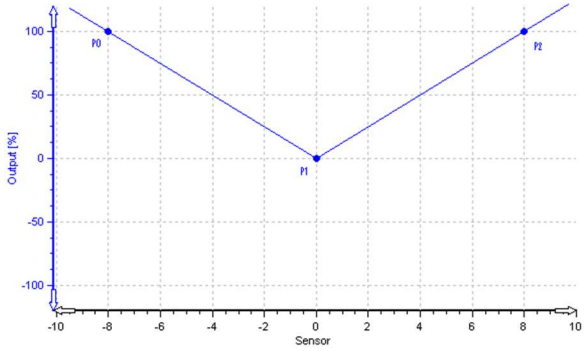
Function declaration		Description	
extern unsigned DLLAPI	DoPEMc2Output (	Function returns Error constant (DoPERR_xxxx)	
DoPE_HANDLE	DoPEHdl	DoPE link handle	
unsigned short	Mode	Mode flags (see below)	
unsigned short	Priority	Output method (see below)	
unsigned short	SensorNo	Sensor number	
unsigned short	Output	Number of analogue output channel	Unit
double	SensorPoints[3]	Sensor points table	%
double	OutputPoints[3]	Output points table	
WORD	* lpusTAN)	Pointer to transaction number (not for Sync. version).	

Definition of Mode:

MC2OUT_MODE_OFF	Stop output to this analogue output channel.
MC2OUT_MODE_2POINTS	Output definition by 2 points.



MC2OUT_MODE_3POINTS Output definition by 3 points.



Definition of Priority:

MC2OUT_PRIORITY _HIGH	Output of this channel every 20 ms.
MC2OUT_PRIORITY _LOW	Every 20 ms only one of the channels with this priority is calculated and given output.
MC2OUT_PRIORITY _BURST	Output of this channel every speed controller cycle (only supported for 4ADA).

9.10.6 DoPEConfPeakValue(Sync)

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. These calculated channels are not allowed as the command channel in DoPEExts2 or DoPEFDPoti command!

Function declaration		Description	
extern unsigned DLLAPI	DoPEConfPeakValue (	Function returns Error constant (DoPERR_xxxx)	
DoPE_HANDLE	DoPEHdl	DoPE link handle	
unsigned short	PositionMin	Position of Minimum value of XHead Position	
unsigned short	PositionMax	Position of Maximum value of XHead Position	
unsigned short	LoadMin	Position of Minimum value of Load	
unsigned short	LoadMax	Position of Maximum value of Load	
unsigned short	ExtensionMin	Position of Minimum value of Extension	
unsigned short	ExtensionMax	Position of Maximum value of Extension	
WORD	* lpusTANFirst	Pointer to transaction number (not for Sync. version).	
WORD	* lpusTANLast)		

9.10.7 DoPEPeakValueTime(Sync)

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

Function declaration		Description	
extern unsigned DLLAPI	DoPEPeakValueTime (	Function returns Error constant (DoPERR_xxxx)	
DoPE_HANDLE	DoPEHdl	DoPE link handle	
double	Time	Reset time for peak value detection	s
WORD	* lpusTANFirst	Pointer to transaction number (not for Sync. version).	
WORD	* lpusTANLast)		

9.11 Sensor Correction

9.11.1 DoPESetSensorCorrection (Sync)

Set sensor correction table.

Minimum requirements: EDC580V/220V, EdcApp 9140.002, DoPE 2.74

Function declaration		Description	
extern unsigned DLLAPI	DoPESetSensorCorrection(	Function returns Error constant (DoPERR_xxxx)	
DoPE_HANDLE	DoPEHdl	DoPE link handle	
WORD	CalculatedSensor	Sensor number of the corresponding Calculated Measuring Channel (with formula F_SensorCorrection)	
DoPESensorCorrection	*CorrTable	Pointer to sensor correction table.	
WORD	* lpusTAN)	Pointer to transaction number (not for Sync. version).	

```
SENSOR_CORR_MAX    32                          /* maximum number of sensor correction points */

typedef struct
{
  DoPE_HANDLE     DoPEHdl;
  unsigned        CorrNo;                       /* Number of valid entries */
  double          S1Correction [SENSOR_CORR_MAX]; /* Correction values for S1 */
  double          S2Value [SENSOR_CORR_MAX];    /* S2 values (must be in ascending order) */
} DoPESensorCorrection;
```

9.11.2 DoPESetStiffnessCorrection (Sync)

Set stiffness correction table.

Minimum requirements: EDC580/220, EdcApp 9133.029, DoPE 2.71

Function declaration		Description	
extern unsigned DLLAPI	DoPESetStiffnessCorrection(	Function returns Error constant (DoPERR_xxxx)	
DoPE_HANDLE	DoPEHdl	DoPE link handle	
DoPEStiffnessCorrection	*CorrTable	Pointer to stiffness correction table.	
WORD	* lpusTAN)	Pointer to transaction number (not for Sync. version).	

```
STIFF_CORR_MAX    32                          /* maximum number of stiffness correction points */

typedef struct
{
  DoPE_HANDLE     DoPEHdl;
  unsigned        CorrNo;                     /* Number of valid entries */
  double          Load[STIFF_CORR_MAX];       /* Load value */
  double          Deformation[STIFF_CORR_MAX];  /* Machine deformation at load value */
} DoPEStiffnessCorrection;
```

9.12 Serial Sensors

9.12.1 DoPEWrSensorMsg (Sync)

Write a message to a sensor.The message is not interpreted by DoPE!
Maximum message length is 80 Bytes.
Note: This function isnot available for EDC5/25/100.

Minimum requirements: EDC120/60, EdcApp 9123.i, DoPE 2.23

Function declaration		Description	
extern unsigned DLLAPI	DoPEWrSensorMsg(	Function returns Error constant (DoPERR_xxxx)	
DoPE_HANDLE	DoPEHdl	DoPE link handle	
unsigned short	SensorNo	Sensor Number	
void	* Buffer	Pointer to message to transmit	
unsigned	Length	Message length in Byte's.	
WORD	* lpusTAN)	Pointer to transaction number (not for Sync. version).	

9.12.2 DoPESerialSensorDef (Sync)

Definition for serial sensor.
A message for a serial sensor will be transmitted after:

Minimum requirements: EDC120/60, EdcApp 9123.v, DoPE 2.29

Function declaration		Description	
extern unsigned DLLAPI	DoPESerialSensorDef (	Function returns Error constant (DoPERR_xxxx)	
DoPE_HANDLE	DoPEHdl	DoPE link handle	
unsigned short	SensorNo	Sensor Number	
double	Timeout	Timeout, 0 = no timeout	s
unsigned short	MaxLength	Transmit after Number of bytes received (must be <= SERSEN_TRANSFER)	
char	EndChar1	First Endcharacter	
char	EndChar2	Second Endcharacter	
WORD	EndCharMode	Mode for detection end character (see below)	
WORD	* lpusTAN)	Pointer to transaction number (not for Sync. version).	

SERSEN_ENDCHAR_NO	No Endcharacter active, message will be transmitted after timeout, or MaxLength
SERSEN_ENDCHAR_1	Detect only EndChar1
SERSEN_ENDCHAR_1_OR_2	Detect EndChar1 or EndChar1
SERSEN_ENDCHAR_1_AND_2	Detect Sequence EndChar1 + EndChar2

SERSEN_ENDCHAR_1_PLUS1	Detect EndChar1 plus one Character

9.12.3 DoPESetSerialSensor (Sync)

Set value (e.g. Temperature) for serial channel.

Minimum requirements: EDC120/60, EdcApp 9123.v, DoPE 2.29

Function declaration		Description	
extern unsigned DLLAPI	DoPESetSerialSensor (	Function returns Error constant (DoPERR_xxxx)	
DoPE_HANDLE	DoPEHdl	DoPE link handle	
unsigned short	SensorNo	Sensor Number	
unsigned short	Mode	SERSEN_SET_COMMAND: Set Command value SERSEN_SET_FEEDBACK: Set Feedback value	
double	Value	Value to set	
double	Speed	Speed for ramp	Unit/s
WORD	* lpusTAN)	Pointer to transaction number (not for Sync. version).	

9.12.4 DoPESetSerialSensorTransparent (Sync)

Set serial channel to transparent mode or the previously selected protocol.

Minimum requirements: EDC580/220, EdcApp 9133.021, DoPE 2.66

Function declaration		Description
extern unsigned DLLAPI	DoPESetSerialSensorTransparent (	Function returns Error constant (DoPERR_xxxx)
DoPE_HANDLE	DoPEHdl	DoPE link handle
unsigned short	SensorNo	Sensor Number
unsigned short	Mode	SERSEN_SET_PROTOCOL:Set sensor to the protocol (Set-up) SERSEN_SET_TRANSPARENT: Set sensor to transparent mode
WORD	* lpusTAN)	Pointer to transaction number (not for Sync. version).

9.13 Debug Messages

For service purposes it is sometimes necessary to record the debug output of the EDC and send it to the DOLI support team. To facilitate this, the OnDebugMsg Event Handler were implemented.

9.13.1 DoPEDebugMsgEnable (Sync)

Enable or disable debug messages.
Attention: Unfortunately, the debug messages influence the runtime behavior of the application and should be turned off by default and turned on only in e.g. a remote support session

Minimum requirements: EDC580V/220V, DoPE 2.77

Function declaration		Description
extern unsigned DLLAPI	DoPEDebugMsgEnable(	Function returns Error constant (DoPERR_xxxx)
DoPE_HANDLE	DoPEHdl	DoPE link handle
unsigned	Enable	!=0 enables =0 disables debug messages.
WORD	* lpusTAN)	Pointer to transaction number (not for Sync. version).

9.13.2 DoPESendDebugCommand (Sync)

Send a command to the debug interface.
The “?” command prints a list of the available commands to the debug interface..

Minimum requirements: EDC580V/220V, DoPE 2.77

Function declaration		Description
extern unsigned DLLAPI	DoPEDebugMsgEnable(	Function returns Error constant (DoPERR_xxxx)
DoPE_HANDLE	DoPEHdl	DoPE link handle
char	*Text	Pointer to zero terminated text to transmit.
WORD	* lpusTAN)	Pointer to transaction number (not for Sync. version).

10 Input / Output-Commands

10.1 Analogue output

10.1.1 DoPESetOutput(Sync)

Set an analogue output channel. The output channel must be assigned in EDC set-up.			
Function declaration		**Description**	
extern unsigned DLLAPI DoPE_HANDLE unsigned short double WORD	DoPESetOutput (DoPEHdl Output Value * lpusTAN)	Function returns Error constant (DoPERR_xxxx) DoPE link handle Number of analogue output channel New value of output channelin % of max. value Pointer to transaction number (not for Sync. version).	 %

10.1.2 DoPESetOutChannelOffset (Sync)

Set an analogue output channel offset.			
Minimum requirements: EDC120/60, EdcApp 9123.i, DoPE 2.23			
Function declaration		**Description**	
extern unsigned DLLAPI DoPE_HANDLE unsigned short double WORD	DoPESetOutChannelOffset(DoPEHdl Output Offset * lpusTAN)	Function returns Error constant (DoPERR_xxxx) DoPE link handle Number of analogue output channel New Offset value of output channelin % of max. value Pointer to transaction number (not for Sync. version).	 %

10.1.3 DoPESetDither(Sync)

Set an analogue output channel dither. Some DOLI output amplifier like D03I, D16I, D32I have a digital dither generator function. The dither amplitude and frequency can be set by this command.			
Minimum requirements: EDC120/60, DoPE 2.21			
Function declaration		**Description**	
extern unsigned DLLAPI DoPE_HANDLE unsigned short double double WORD	DoPESetDither (DoPEHdl Output Frequency Amplitude * lpusTAN)	Function returns Error constant (DoPERR_xxxx) DoPE link handle Number of analogue output channel Dither frequency in Hz Dither amplitude in % of nominal output current Pointer to transaction number (not for Sync. version).	 Hz %

10.1.4 DoPEOfflineActionOutput (Sync)

Definition of an action for an initialized 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 reinitialize, and DoPEOfflineActionBitOutput command has no effect.			
Minimum requirements: EDC120/60, EdcApp 9123.s, DoPE 2.27			
Function declaration		**Description**	
extern unsigned DLLAPI DoPE_HANDLE unsigned short unsigned short	DoPEOfflineActionOutput (DoPEHdl Output Mode	Function returns Error constant (DoPERR_xxxx) DoPE link handle Number of analogue output channel. If closed loop control, Output channel 0 cannot be selected! DO_NOTHING: Don't modify this digital output USE_INIT_VALUE: Use Initial value from set-up	

	double	Value	USE_VALUE: Use defined value Output value used in USE_VALUE mode in % of max. value
	WORD	* lpusTAN)	Pointer to transaction number (not for Sync. version).

10.2 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 an EDC.

For Digital input/output bits that are not often used, DoPE provides functions to read or write these devices without having initialized it (not assigned in EDC set-up).

10.2.1 DoPESetIoCompatibilityMode (Sync)

Set the compatibility mode for bit input and output channels.
All EDC's before EDC220/580 generation used negative logic for some digital inputs and outputs (you must send a '0' to activate the output). Since DoPE version V2.51 DoPE provides a positive logic for all EDC generations.
By default the positive logic conversion is active. "Older" application programs, can switch off this conversion by enabling the IoCompatibilityMode.

Minimum requirements: DoPE 2.51

Function declaration			Description	
extern unsigned DLLAPI		DoPESetIoCompatibilityMode (	Function returns Error constant (DoPERR_xxxx)	
	DoPE_HANDLE	DoPEHdl	DoPE link handle	
	unsigned short	SensorNo	Sensor Number	
	unsigned	Enable	!=0 enables 0 disables compatibility mode	
	WORD	* lpusTAN)	Pointer to transaction number (not for Sync. version).	

10.2.2 DoPESetB(Sync)

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.

Function declaration			Description	
extern unsigned DLLAPI		DoPESetB (	Function returns Error constant (DoPERR_xxxx)	
	DoPE_HANDLE	DoPEHdl	DoPE link handle	
	unsigned short	BitOutputNo	Number of bit output device	
	unsigned short	SetB	These bits will be set	
	unsigned short	ResB	These bits will be set	
	unsigned short	FlashB	These bits 'flash'	
	WORD	* lpusTAN)	Pointer to transaction number (not for Sync. version).	

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.

10.2.3 DoPECalOut(Sync)

Activate / deactivate calibration output on EDC. The calibration contact may be used to trigger a reference measuring system during load calibration (see also DoPEIOMisc).		
Function declaration	**Description**	
extern unsigned DLLAPI DoPECalOut (DoPE_HANDLE DoPEHdl unsigned short Cal WORD * lpusTAN)	Function returns Error constant (DoPERR_xxxx) DoPE link handle !=0 activate Calibration output 0 deactivate Calibration output Pointer to transaction number (not for Sync. version).	

10.2.4 DoPEBeep(Sync)

Activate / deactivate beep on EDC.		
Function declaration	**Description**	
extern unsigned DLLAPI DoPEBeep (DoPE_HANDLE DoPEHdl unsigned short Beep WORD * lpusTAN)	Function returns Error constant (DoPERR_xxxx) DoPE link handle !=0 activate Beep 0 deactivate Beep Pointer to transaction number (not for Sync. version).	

10.2.5 DoPEUniOut(Sync)

Activate/Deactivate universal digital output bits at EDC.These digital outputs may be used for general purposes.The universal digital output bits are located at X2. EDC5/25 does not have digital output bits. EDC100 offers 4 output bits (bit 0 and 1 are reserved), all newer EDC's have 8 bits. Note: If the parameter "MaIo" in EDC Set-up->GeneralData->MaIo) is set to "YES", the EDC uses some of these output bits to give out the active machine set-up number. Firmware versions before 9133.008 uses bit 0 and 1, firmware 9133.008 or newer uses one bit for every setup, starting with bit 0.		
Function declaration	**Description**	
extern unsigned DLLAPI DoPEUniOut(DoPE_HANDLE DoPEHdl unsigned short Output WORD * lpusTAN)	Function returns Error constant (DoPERR_xxxx) DoPE link handle Bit 0 ... 15 represent the digital outputs Pointer to transaction number (not for Sync. version).	

10.2.6 DoPEBypass(Sync)

Activate/Deactivate bypass output bits at EDC.		
Function declaration	**Description**	
extern unsigned DLLAPI DoPEBypass (DoPE_HANDLE DoPEHdl unsigned short Bypass WORD * lpusTAN)	Function returns Error constant (DoPERR_xxxx) DoPE link handle !=0 activates bypass output 0 deactivates bypass output Pointer to transaction number (not for Sync. version).	

10.2.7 DoPERdBitInput

Read a digital input device.
Note: This command will also work on not initialized I/O-devices.

Function declaration		Description	
extern unsigned DLLAPI	DoPERdBitInput (	Function returns Error constant (DoPERR_xxxx)	
DoPE_HANDLE	DoPEHdl	DoPE link handle	
WORD	Connector	Connector number, for the I/O device Use connector constants from header file like “CON_X2”	
WORD	*Value	Data pointer to store result	
WORD	* lpusTAN)	Pointer to transaction number (not for Sync. version).	

10.2.8 DoPEWrBitOutput(Sync)

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

Function declaration		Description	
extern unsigned DLLAPI	DoPEWrBitOutput(	Function returns Error constant (DoPERR_xxxx)	
DoPE_HANDLE	DoPEHdl	DoPE link handle	
WORD	Connector	Connector number, for the I/O device Use connector constants from header file like “CON_X2”	
WORD	Value	Data to be written to the output device.	
WORD	* lpusTAN)	Pointer to transaction number (not for Sync. version).	

10.2.9 DoPEOfflineActionBitOutput (Sync)

Definition of an action for an initialized 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 reinitialize, and DoPEOfflineActionBitOutput command has no effect.

Minimum requirements: EDC120/60, EdcApp 9123.s, DoPE 2.27

Function declaration		Description	
extern unsigned DLLAPI	DoPEOfflineActionBitOutput (	Function returns Error constant (DoPERR_xxxx)	
DoPE_HANDLE	DoPEHdl	DoPE link handle	
unsigned short	BitOutputNo	Number of bit output device	
unsigned short	Mode	DO_NOTHING: Don't modify this digital output USE_INIT_VALUE: Use Initial value from set-up after offline USE_VALUE: Use defined value after offline	
WORD	Value	Use defined value after offline.	
WORD	* lpusTAN)	Pointer to transaction number (not for Sync. version).	

10.3 IOSignals

EDCs with firmware version 9133.008 and later support the following functions:

- Control of grips
- Control of extensometer
- Fixed XHead adjustment
- High/Low pressure selection
- SHalt signal (since firmware 9133.024)
- Miscellaneous control I/O (since firmware 9133.024)

All of these functions must be configured in the EDC set-up. The control functions may be activated via keys on RMC1/RMC7, or via commands from PC.

Each control function uses digital I/O with certain functions. The I/O bits for one function must be configured as a whole block to any I/O device. The first bit of a block may be assigned to any bit of the I/O device.

The number of reserved In- and Output bits for each block are identical, even if not all are currently used. The Miscellaneous Input- and Output-bits may be defined separately.

Note: For more detailed information please refer to EDC Installation Manual.

10.3.1 DoPEIOGripEnable (Sync)

Enable or disable grip IO handling.		
Minimum requirements: EDC580/220, EdcApp 9133.008, DoPE 2.59		
Function declaration	**Description**	
extern unsigned DLLAPI DoPEIOGripEnable (DoPE_HANDLE DoPEHdl Unsigned Enable WORD * lpusTAN)	Function returns Error constant (DoPERR_xxxx) DoPE link handle !=0 enables 0 disables grip IO handling Pointer to transaction number (not for Sync. version).	

10.3.2 DoPEIOGripSet (Sync)

Perform a grip action.		
Minimum requirements: EDC580/220, EdcApp 9133.008, DoPE 2.59		
Function declaration	**Description**	
extern unsigned DLLAPI DoPEIOGripSet (DoPE_HANDLE DoPEHdl unsigned Grip unsigned Action double Pressure WORD * lpusTAN)	Function returns Error constant (DoPERR_xxxx) DoPE link handle Grip to use(use IO_GRIP_ defines) Action to perform (use IO_GRIP_ACTION_ defines) Pressure to be applied Pointer to transaction number (not for Sync. version).	 %

Constants for Grip

IO_GRIP_UPPER
IO_GRIP_LOWER
IO_GRIP_BOTH

Constants for Action

IO_GRIP_ACTION_OPEN
IO_GRIP_ACTION_CLOSE
IO_GRIP_ACTION_HIGH_PRESSURE1
IO_GRIP_ACTION_HIGH_PRESSURE2

10.3.3 DoPEIOGripPressure(Sync)

Set grip parameter.			
Minimum requirements: EDC580/220, EdcApp 9133.008, DoPE 2.59			
Function declaration		**Description**	
extern unsigned DLLAPI	DoPEIOGripPressure (	Function returns Error constant (DoPERR_xxxx)	
DoPE_HANDLE	DoPEHdl	DoPE link handle	
double	LowPressure	Low pressure value	%
double	HighPressure	High pressure value	%
double	RampTime	Time to switch between low and high pressure	s
WORD	* lpusTAN)	Pointer to transaction number (not for Sync. version).	

10.3.4 DoPEIOExtEnable(Sync)

Enable or disable extensometer IO handling.			
Minimum requirements: EDC580/220, EdcApp 9133.008, DoPE 2.59			
Function declaration		**Description**	
extern unsigned DLLAPI	DoPEIOExtEnable (	Function returns Error constant (DoPERR_xxxx)	
DoPE_HANDLE	DoPEHdl	DoPE link handle	
unsigned	Enable	!=0 enables 0 disables extensometer IO handling	
WORD	* lpusTAN)	Pointer to transaction number (not for Sync. version).	

10.3.5 DoPEIOExtSet (Sync)

Perform an extensometer action.			
Minimum requirements: EDC580/220, EdcApp 9133.008, DoPE 2.59			
Function declaration		**Description**	
extern unsigned DLLAPI	DoPEIOExtSet(	Function returns Error constant (DoPERR_xxxx)	
DoPE_HANDLE	DoPEHdl	DoPE link handle	
unsigned	Ext	Grip to use(use IO_EXT_ defines)	
unsigned	Action	Action to perform(use IO_EXT_ACTION_ defines)	
WORD	* lpusTAN)	Pointer to transaction number (not for Sync. version).	

IO_EXT_UPPER
IO_EXT_LOWER
IO_EXT_BOTH

IO_EXT_ACTION_OPEN
IO_EXT_ACTION_CLOSE

10.3.6 DoPEIOFixedXHeadEnable (Sync)

Enable or disable fixed cross head IO handling.			
Minimum requirements: EDC580/220, EdcApp 9133.008, DoPE 2.59			
Function declaration		**Description**	
extern unsigned DLLAPI DoPE_HANDLE unsigned WORD	DoPEIOFixedXHeadEnable (DoPEHdl Enable * lpusTAN)	Function returns Error constant (DoPERR_xxxx) DoPE link handle !=0 enables 0 disables fixed cross head IO handling Pointer to transaction number (not for Sync. version).	

10.3.7 DoPEIOFixedXHeadSet (Sync)

Move fixed cross head.			
Minimum requirements: EDC580/220, EdcApp 9133.008, DoPE 2.59			
Function declaration		**Description**	
extern unsigned DLLAPI DoPE_HANDLE unsigned WORD	DoPEIOFixedXHeadSet (DoPEHdl Direction * lpusTAN)	Function returns Error constant (DoPERR_xxxx) DoPE link handle Direction of movement (use MOVE_HALT, MOVE_UP, MOVE_DOWN defines) Pointer to transaction number (not for Sync. version).	

10.3.8 DoPEIOHighPressureEnable (Sync)

Enable or disable high pressure IO handling.			
Minimum requirements: EDC580/220, EdcApp 9133.008, DoPE 2.59			
Function declaration		**Description**	
extern unsigned DLLAPI DoPE_HANDLE unsigned WORD	DoPEIOHighPressureEnable (DoPEHdl Enable * lpusTAN)	Function returns Error constant (DoPERR_xxxx) DoPE link handle !=0 enables 0 disables high pressure IO handling Pointer to transaction number (not for Sync. version).	

10.3.9 DoPEIOHighPressureSet (Sync)

Set high or low pressure.			
Minimum requirements: EDC580/220, EdcApp 9133.008, DoPE 2.59			
Function declaration		**Description**	
extern unsigned DLLAPI DoPE_HANDLE unsigned WORD	DoPEIOHighPressureSet (DoPEHdl Enable * lpusTAN)	Function returns Error constant (DoPERR_xxxx) DoPE link handle !=0 selects high pressure 0 selects low pressure Pointer to transaction number (not for Sync. version).	

10.4 Display Commands

DoPE offers functions to write data to the EDC display. The display is subdivided into:

1. Headline section
2. Function key section
3. Value section for two values
4. Beam section

Following functions can be used to write data to the display.

Note: The display should not be updated faster than 3 times per second!

10.4.1 DoPEDspClear(Sync)

Clear LCD-display at EDC front panel.		
Function declaration	**Description**	
extern unsigned DLLAPI DoPEDspClear (DoPE_HANDLE DoPEHdl WORD * lpusTAN)	Function returns Error constant (DoPERR_xxxx) DoPE link handle Pointer to transaction number (not for Sync. version).	

10.4.2 DoPEDspHeadLine / DoPEwDspHeadLine (Sync)

Display HeadLine on LCD-display at EDC front panel.		
Minimum requirements: DoPEwDspHeadLine: DoPE 2.65		
Function declaration	**Description**	
extern unsigned DLLAPI DoPEDspHeadLine (DoPE_HANDLE DoPEHdl char * HeadLine WORD * lpusTAN)	Function returns Error constant (DoPERR_xxxx) DoPE link handle Character string Pointer to transaction number (not for Sync. version).	
extern unsigned DLLAPI DoPEwDspHeadLine (DoPE_HANDLE DoPEHdl wchar_t * HeadLine WORD * lpusTAN)	Function returns Error constant (DoPERR_xxxx) DoPE link handle Wide character string Pointer to transaction number (not for Sync. version).	
DSP_FKEYSLINE_LEN (22) Number of characters in Value field (including terminating zero '\0')		

10.4.3 DoPEDspFKeys / DoPEwDspFKeys (Sync)

Display Function-Keys on LCD-display at EDC front panel.			
Minimum requirements: DoPEwDspFKeys: DoPE 2.65			
Function declaration		**Description**	
extern unsigned DLLAPI DoPE_HANDLE char WORD	DoPEDspFKeys (DoPEHdl * FKeys * lpusTAN)	Function returns Error constant (DoPERR_xxxx) DoPE link handle Character string Pointer to transaction number (not for Sync. version).	
extern unsigned DLLAPI DoPE_HANDLE wchar_t WORD	DoPEwDspFKeys(DoPEHdl * FKeys * lpusTAN)	Function returns Error constant (DoPERR_xxxx) DoPE link handle Wide Character string Pointer to transaction number (not for Sync. version).	
DSP_FKEYSLINE_LEN (22) Number of characters in Value field (including terminating zero '\0')			

10.4.4 DoPEDspMValue / DoPEwDspMValue (Sync)

Display values with dimension on LCD-display at EDC front panel.			
Minimum requirements: DoPEwDspMValue: DoPE 2.65			
Function declaration		**Description**	
extern unsigned DLLAPI DoPE_HANDLE char char char char WORD	DoPEDspMValue (DoPEHdl * Value1 * Value2 * Dim1 * Dim2 * lpusTAN)	Function returns Error constant (DoPERR_xxxx) DoPE link handle Character string for first value Character string for second value Character string for first dimension Character string for second dimension Pointer to transaction number (not for Sync. version).	
extern unsigned DLLAPI DoPE_HANDLE wchar_t wchar_t wchar_t wchar_t WORD	DoPEwDspMValue (DoPEHdl * Value1 * Value2 * Dim1 * Dim2 * lpusTAN)	Function returns Error constant (DoPERR_xxxx) DoPE link handle Wide character string for first value Wide character string for second value Wide character string for first dimension Wide character string for second dimension Pointer to transaction number (not for Sync. version).	
DSP_VALUE_LEN (10) Number of characters in Value field (including terminating zero '\0') DSP_DIM_LEN (7) Number of characters in Dim field (including terminating zero '\0')			

10.4.5 DoPEDspBeamScreen(Sync)

Display frame **and** beam on LCD-display at EDC front panel.			
Function declaration		**Description**	
extern unsigned DLLAPI DoPE_HANDLE short WORD	DoPEDspBeamScreen(DoPEHdl Value * lpusTAN)	Function returns Error constant (DoPERR_xxxx) DoPE link handle Value of the beam in % Pointer to transaction number (not for Sync. version).	%

10.4.6 DoPEDspBeamValue(Sync)

Display beam on LCD-display at EDC front panel.			
Function declaration		**Description**	
extern unsigned DLLAPI DoPE_HANDLE short WORD	DoPEDspBeamValue(DoPEHdl Value * lpusTAN)	Function returns Error constant (DoPERR_xxxx) DoPE link handle Value of the beam in % Pointer to transaction number (not for Sync. version).	%

11 EDC keyboard interface

DoPE offers some function to handlekeys and LED's of EDC keyboards (Front-Panel and RMC).

11.1 DoPERdNumberOfKeyboards

Read the number of connected keyboards.		
Minimum requirements: EDC580/220, DoPE 2.51		
Function declaration	**Description**	
extern unsigned DLLAPI DoPERdNumberOfKeyboards (DoPE_HANDLE DoPEHdl unsigned *pNumber	Function returns Error constant (DoPERR_xxxx) DoPE link handle Pointer to storage for the number of keyboards	

11.2 DoPESetKeyShiftState (Sync)

Set the keyboard shift state.		
Minimum requirements: EDC580/220, DoPE 2.51		
Function declaration	**Description**	
extern unsigned DLLAPI DoPESetKeyShiftState (DoPE_HANDLE DoPEHdl unsigned Enable WORD * lpusTAN)	Function returns Error constant (DoPERR_xxxx) DoPE link handle 0=disable shift state, else enable shift state Pointer to transaction number (not for Sync. version).	

11.3 DoPERdKeyShiftState

Read the keyboard shift state.		
Minimum requirements: EDC580/220, DoPE 2.51		
Function declaration	**Description**	
extern unsigned DLLAPI DoPERdKeyShiftState (DoPE_HANDLE DoPEHdl unsigned *pEnabled	Function returns Error constant (DoPERR_xxxx) DoPE link handle Pointer to storage for keyboard shift state	

11.4 DoPESetLed (Sync)

Switch On/Off LED's at the EDC keyboard.		
Minimum requirements: DoPE 2.51		
Function declaration	**Description**	
extern unsigned DLLAPI DoPESetLed (DoPE_HANDLE DoPEHdl unsigned __int64 LedOn unsigned __int64 LedOff unsigned __int64 LedFlash WORD OemLedOn WORD OemLedOff WORD OemLedFlash WORD * lpusTAN)	Function returns Error constant (DoPERR_xxxx) DoPE link handle These LED's will be set These LED's will be reset These LED's 'flash' These LED's will be set These LED's will be reset These LED's 'flash' Pointer to transaction number (not for Sync. version).	
The parameters will be processed in the followingsequence (important with conflicting data): 1. Flashing LED's. (lowest) 2. Resetting of the LED's. 3. Setting of the LED's. (highest)		

11.5 DoPELedMask

Set the LED matrix bit for a given LED code.Each LED is represented by a bit in the LED matrix. This function convertsa LED code to the matrix bit and sets the bit in the LED or OEM LED matrix.		
Minimum requirements: DoPE 2.51		
Function declaration	**Description**	
extern unsigned DLLAPI DoPELedMask (DoPE_HANDLE DoPEHdl __int32 LedCode unsigned State unsigned __int64 *pLedMask WORD *pOemLedMask	Function returns Error constant (DoPERR_xxxx) DoPE link handle LED code to convert 0=clear bit, else set bit Pointer to the LED matrix Pointer to the OEM-LED matrix (NULL pointers are accepted)	

11.6 DoPECurrentKeys

Read the current keys.		
Minimum requirements: DoPE 2.51		
Function declaration	**Description**	
extern unsigned DLLAPI DoPECurrentKeys(DoPE_HANDLE DoPEHdl unsigned __int64 *pKeys WORD *pOemKeys	Function returns Error constant (DoPERR_xxxx) DoPE link handle current key matrix current OEM-key matrix	

11.7 DoPEKeyPressed

Check if the key with the given key code is pressed.	
Minimum requirements: DoPE 2.51	
Function declaration	**Description**
extern intDLLAPI DoPEKeyPressed (DoPE_HANDLE DoPEHdl __int32 KeyCode unsigned __int64 Keys WORD OemKeys	Function returns 0 if key is not pressed, else key is pressed DoPE link handle Key code to check Key matrix OEM-Key matrix

11.8 DoPEKeyGet

Get the key code of a pressed key.Each pressed key is represented by a set bit in the key matrix. This functionconverts a bit from the key matrix to its key code and resets the bit inthe key matrix.	
Minimum requirements: DoPE 2.51	
Function declaration	**Description**
extern intDLLAPI DoPEKeyGet (DoPE_HANDLE DoPEHdl unsigned __int64 *pKeys WORD *OemKeys	The key code of a pressed key or DoPE_KEY_NONE DoPE link handle Pointer to the key matrix Pointer to the OEM-Key matrix

11.9 Definition of keys

Key constant	Key code	Description
DOLI keys		
DoPE_KEY_NONE	-1	invalid (no key pressed)
DoPE_KEY_HALT	0x00	'HALT'
DoPE_KEY_UP	0x01	'UP'
DoPE_KEY_DOWN	0x02	'DOWN'
DoPE_KEY_DPOTI	0x03	'DigiPoti pressed'
DoPE_KEY_F1	0x04	'F1' function key
DoPE_KEY_F2	0x05	'F2' function key
DoPE_KEY_F3	0x06	'F3' function key
DoPE_KEY_ON	0x07	'ON'
DoPE_KEY_TEST	0x08	'TEST'
DoPE_KEY_UPR_GRIP_OPEN	0x09	'Upper Grip Open'
DoPE_KEY_UPR_GRIP_CLOSE	0x0A	'Upper Grip Close'
DoPE_KEY_LWR_GRIP_OPEN	0x0B	'Lower Grip Open'
DoPE_KEY_LWR_GRIP_CLOSE	0x0C	'Lower Grip Close'
DoPE_KEY_EXTMETER_OPEN	0x0D	'Extensometer Open'
DoPE_KEY_EXTMETER_CLOSE	0x0E	'Extensometer Close'
DoPE_KEY_FIXED_XHEAD_UP	0x0F	'fixed X-Head Up'
DoPE_KEY_FIXED_XHEAD_DOWN	0x10	'fixed X-Head Down'
DoPE_KEY_DPOTI_CONTROL	0x11	'DigiPoti Speed-Position Mode'
DoPE_KEY_BF1	0x12	'blind F1' function key (no LCD)
DoPE_KEY_BF2	0x13	'blind F2' function key (no LCD)
DoPE_KEY_BF3	0x14	'blind F3' function key (no LCD)
DoPE_KEY_HIGH_PRESSURE	0x15	'Activate High Pressure'
DoPE_KEY_LOW_PRESSURE	0x16	'Activate Low Pressure'
DoPE_KEY_OFF	0x17	'OFF'
DoPE_KEY_UP_FAST	0x18	'Move UP fast'
DoPE_KEY_DOWN_FAST	0x19	'Move DOWN fast'
DoPE_KEY_TEST_STOP	0x1A	'Test stop'
DoPE_KEY_TEST_RETURN	0x1B	'Test return'
DoPE_KEY_TEST_HALT_CONTINUE	0x1C	'Test halt / continue'
DoPE_KEY_reserved1D	0x1D	
DoPE_KEY_reserved1E	0x1E	
DoPE_KEY_reserved1F	0x1F	
DoPE_KEY_reserved20	0x20	
DoPE_KEY_CTRL_SENSOR3	0x21	
DoPE_KEY_CTRL_SENSOR4	0x22	
DoPE_KEY_CTRL_SENSOR5	0x23	
DoPE_KEY_CTRL_SENSOR6	0x24	
DoPE_KEY_CTRL_SENSOR7	0x25	
DoPE_KEY_CTRL_SENSOR8	0x26	
DoPE_KEY_CTRL_SENSOR9	0x27	
DoPE_KEY_CTRL_SENSOR10	0x28	
DoPE_KEY_CTRL_SENSOR11	0x29	
DoPE_KEY_CTRL_SENSOR12	0x2A	
DoPE_KEY_CTRL_SENSOR13	0x2B	
DoPE_KEY_CTRL_SENSOR14	0x2C	
DoPE_KEY_CTRL_SENSOR15	0x2D	
DoPE_KEY_DP	0x2E	'.' period (decimal point)
DoPE_KEY_SIGN	0x2F	'±' sign
DoPE_KEY_0	0x30	'0'
DoPE_KEY_1	0x31	'1'
DoPE_KEY_2	0x32	'2'
DoPE_KEY_3	0x33	'3'

DoPE_KEY_4	0x34	'4'
DoPE_KEY_5	0x35	'5'
DoPE_KEY_6	0x36	'6'
DoPE_KEY_7	0x37	'7'
DoPE_KEY_8	0x38	'8'
DoPE_KEY_9	0x39	'9'
DoPE_KEY_CTRL_POS	0x3A	
DoPE_KEY_CTRL_LOAD	0x3B	
DoPE_KEY_CTRL_EXTENSION	0x3C	
DoPE_KEY_reserved3D	0x3D	
DoPE_KEY_reserved3E	0x3E	
DoPE_KEY_CONNECT	0x3F	
OEM keys		
DoPE_KEY_OEM0	0x40	
DoPE_KEY_OEM1	0x41	
DoPE_KEY_OEM2	0x42	
DoPE_KEY_OEM3	0x43	
DoPE_KEY_OEM4	0x44	
DoPE_KEY_OEM5	0x45	
DoPE_KEY_OEM6	0x46	
DoPE_KEY_OEM7	0x47	
DoPE_KEY_OEM8	0x48	
DoPE_KEY_OEM9	0x49	
DoPE_KEY_OEMA	0x4A	
DoPE_KEY_OEMB	0x4B	
DoPE_KEY_OEMC	0x4C	
DoPE_KEY_OEMD	0x4D	
DoPE_KEY_OEME	0x4E	
DoPE_KEY_OEMF	0x4F	

11.10Definition of LED's

LED constant	LED refers to key
DOLI LED's	
DoPE_LED_HALT	DoPE_KEY_HALT
DoPE_LED_UP	DoPE_KEY_UP
DoPE_LED_DOWN	DoPE_KEY_DOWN
DoPE_LED_DPOTI	DoPE_KEY_DPOTI
DoPE_LED_F1	DoPE_KEY_F1
DoPE_LED_F2	DoPE_KEY_F2
DoPE_LED_F3	DoPE_KEY_F3
DoPE_LED_ON	DoPE_KEY_ON
DoPE_LED_TEST	DoPE_KEY_TEST
DoPE_LED_UPR_GRIP_OPEN	DoPE_KEY_UPR_GRIP_OPEN
DoPE_LED_UPR_GRIP_CLOSE	DoPE_KEY_UPR_GRIP_CLOSE
DoPE_LED_LWR_GRIP_OPEN	DoPE_KEY_LWR_GRIP_OPEN
DoPE_LED_LWR_GRIP_CLOSE	DoPE_KEY_LWR_GRIP_CLOSE
DoPE_LED_EXTMETER_OPEN	DoPE_KEY_EXTMETER_OPEN
DoPE_LED_EXTMETER_CLOSE	DoPE_KEY_EXTMETER_CLOSE
DoPE_LED_FIXED_XHEAD_UP	DoPE_KEY_FIXED_XHEAD_UP
DoPE_LED_FIXED_XHEAD_DOWN	DoPE_KEY_FIXED_XHEAD_DOWN
DoPE_LED_DPOTI_CONTROL	DoPE_KEY_DPOTI_CONTROL
DoPE_LED_BF1	DoPE_KEY_BF1
DoPE_LED_BF2	DoPE_KEY_BF2
DoPE_LED_BF3	DoPE_KEY_BF3
DoPE_LED_HIGH_PRESSURE	DoPE_KEY_HIGH_PRESSURE
DoPE_LED_LOW_PRESSURE	DoPE_KEY_LOW_PRESSURE
DoPE_LED_OFF	DoPE_KEY_OFF
DoPE_LED_UP_FAST	DoPE_KEY_UP_FAST
DoPE_LED_DOWN_FAST	DoPE_KEY_DOWN_FAST
DoPE_LED_TEST_STOP	DoPE_KEY_TEST_STOP
DoPE_LED_TEST_RETURN	DoPE_KEY_TEST_RETURN
DoPE_LED_TEST_HALT_CONTINUE	DoPE_KEY_TEST_HALT_CONTINUE
DoPE_LED_reserved1D	DoPE_KEY_reserved1D
DoPE_LED_reserved1E	DoPE_KEY_reserved1E
DoPE_LED_reserved1F	DoPE_KEY_reserved1F
DoPE_LED_reserved20	DoPE_KEY_reserved20
DoPE_LED_CTRL_SENSOR3	DoPE_KEY_reserved21
DoPE_LED_CTRL_SENSOR4	DoPE_KEY_reserved22
DoPE_LED_CTRL_SENSOR5	DoPE_KEY_reserved23
DoPE_LED_CTRL_SENSOR6	DoPE_KEY_reserved24
DoPE_LED_CTRL_SENSOR7	DoPE_KEY_reserved25
DoPE_LED_CTRL_SENSOR8	DoPE_KEY_reserved26
DoPE_LED_CTRL_SENSOR9	DoPE_KEY_reserved27
DoPE_LED_CTRL_SENSOR10	DoPE_KEY_reserved28
DoPE_LED_CTRL_SENSOR11	DoPE_KEY_reserved29
DoPE_LED_CTRL_SENSOR12	DoPE_KEY_reserved2A
DoPE_LED_CTRL_SENSOR13	DoPE_KEY_reserved2B
DoPE_LED_CTRL_SENSOR14	DoPE_KEY_reserved2C
DoPE_LED_CTRL_SENSOR15	DoPE_KEY_reserved2D
DoPE_LED_DP	DoPE_KEY_DP
DoPE_LED_SIGN	DoPE_KEY_SIGN
DoPE_LED_0	DoPE_KEY_0
DoPE_LED_1	DoPE_KEY_1
DoPE_LED_2	DoPE_KEY_2
DoPE_LED_3	DoPE_KEY_3
DoPE_LED_4	DoPE_KEY_4

DoPE_LED_5	DoPE_KEY_5
DoPE_LED_6	DoPE_KEY_6
DoPE_LED_7	DoPE_KEY_7
DoPE_LED_8	DoPE_KEY_8
DoPE_LED_9	DoPE_KEY_9
DoPE_LED_CTRL_POS	DoPE_KEY_reserved3A
DoPE_LED_CTRL_LOAD	DoPE_KEY_reserved3B
DoPE_LED_CTRL_EXTENSION	DoPE_KEY_reserved3C
DoPE_LED_reserved3D	DoPE_KEY_reserved3D
DoPE_LED_reserved3E	DoPE_KEY_reserved3E
DoPE_LED_CONNECT	DoPE_KEY_CONNECT
OEM- LED's	
DoPE_LED_OEM0	DoPE_KEY_OEM0
DoPE_LED_OEM1	DoPE_KEY_OEM1
DoPE_LED_OEM2	DoPE_KEY_OEM2
DoPE_LED_OEM3	DoPE_KEY_OEM3
DoPE_LED_OEM4	DoPE_KEY_OEM4
DoPE_LED_OEM5	DoPE_KEY_OEM5
DoPE_LED_OEM6	DoPE_KEY_OEM6
DoPE_LED_OEM7	DoPE_KEY_OEM7
DoPE_LED_OEM8	DoPE_KEY_OEM8
DoPE_LED_OEM9	DoPE_KEY_OEM9
DoPE_LED_OEMA	DoPE_KEY_OEMA
DoPE_LED_OEMB	DoPE_KEY_OEMB
DoPE_LED_OEMC	DoPE_KEY_OEMC
DoPE_LED_OEMD	DoPE_KEY_OEMD
DoPE_LED_OEME	DoPE_KEY_OEME
DoPE_LED_OEMF	DoPE_KEY_OEMF

12 SetupCommands

You can read or write machine set-up data.

Set-up data from number one onwards stored inside EDC in an 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 initialize EDC.

12.1 DoPERdSetupAll(Sync)

Read the total set-up structure.			
Function declaration		**Description**	
extern unsigned DLLAPI DoPE_HANDLE unsigned short DoPESetup WORD WORD	DoPERdSetupAll (DoPEHdl SetupNo *Setup * lpusTANFirst * lpusTANLast	Function returns Error constant (DoPERR_xxxx) DoPE link handle Set-up Number Pointer to set-up data structure Pointer to transaction number (not for Sync. version).	

12.2 DoPEWrSetupAll(Sync)

Write the total set-up structure.			
Function declaration		**Description**	
extern unsigned DLLAPI DoPE_HANDLE unsigned short DoPESetup WORD WORD	DoPEWrSetupAll (DoPEHdl SetupNo *Setup * lpusTANFirst * lpusTANLast	Function returns Error constant (DoPERR_xxxx) DoPE link handle Set-up Number Pointer to set-up data structure Pointer to transaction number (not for Sync. version).	

DoPESetup structure:

```
typedef struct                          /* EDC Setup Data                      */
{                                       /* ----------------------------------- */
  DoPESenDef       SDef[MAX_MC];        /* Sensor definition data              */
  DoPECtrlSenDef   CSDef[MAX_CTRL];     /* Control-Sensor definition           */
  DoPECtrlSenDef   CSDefHigh[MAX_CTRL]; /* Control-Sensor def. (high pressure)*/ //EDC220+
  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;         /* Linearization table FALSE values    */
  DoPELinTblTrue   LinTblTrue;          /* Linearization table TRUE values     */
  DoPEIOSignals    IOSignals;           /* IO signal definition                */ //EDC220+
  DoPEMainMenu     MainMenu[MAX_MAIN_MENU]; /* EDC main menu definition        */ //EDC220+
  DoPERmcDef       RmcDef;              /* RMC definition                      */ //EDC220+
} DoPESetup;
```

12.3 DoPESetupScale

Sets the set-up user scale.			
Minimum requirements: DoPE 2.20			
Function declaration		**Description**	
extern unsigned DLLAPI DoPE_HANDLE unsigned short UserScale WORD	DoPESetupScale (DoPEHdl SetupNo US * lpusTAN)	Function returns Error constant (DoPERR_xxxx) DoPE link handle Set-up Number User scale for all set-up sensor data 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. Pointer to transaction number (not for Sync. version).	

12.4 DoPERdSetupNumber

Read currently selected set-up number.			
Function declaration		**Description**	
extern unsigned DLLAPI DoPE_HANDLE unsigned short WORD	DoPERdSetupNumber (DoPEHdl *SetupNo * lpusTAN)	Function returns Error constant (DoPERR_xxxx) DoPE link handle Pointer to store set-up number Pointer to transaction number (not for Sync. version).	

12.5 DoPERdGeneralData

Read general data of opened set-up.			
Function declaration		**Description**	
extern unsigned DLLAPI DoPE_HANDLE DoPEGeneralData WORD	DoPERdGeneralData (DoPEHdl * GeneralData * lpusTAN)	Function returns Error constant (DoPERR_xxxx) DoPE link handle Pointer for DoPEGeneralData structure Pointer to transaction number (not for Sync. version).	

12.6 DoPEWrGeneralData(Sync)

Write general data to opened set-up.

Function declaration		Description
extern unsigned DLLAPI	DoPEWrGeneralData (	Function returns Error constant (DoPERR_xxxx)
DoPE_HANDLE	DoPEHdl	DoPE link handle
DoPEGeneralData	* GeneralData	Pointer to DoPEGeneralData structure
WORD	* lpusTAN)	Pointer to transaction number (not for Sync. version).

DoPEGeneralData structure:

```
typedef struct                          /* General data                          */
{                                       /* ------------------------------------ */
  WORD    MachineNo;                    /* Number of active machines       [No]*/
  WORD    MachineNoIo;                  /* Machine select IO's config.     [No]*/
                                        /* (see MACHINE_NO_IO_... definitions)*/
  WORD    Supervisor;                   /* Supervisor mode                 [No]*/ //EDC120-
  WORD    SuperPassword;                /* Supervisor Password(0=inactive) [No]*/
  WORD    UserPassword;                 /* User       Password(0=inactive) [No]*/
  WORD    Logo;                         /* DOLI Logo (0 = inactive)        [No]*/
  DWORD   nRmc;                         /* Required number of RMC's        [No]*/ //EDC220+
  DWORD   Language;                     /* Language                        [No]*/ //EDC220+
  DWORD   FunctionID;                   /* User defined function ID        [No]*/ //EDC220+
  WORD    SyncOption;                   /* Sync option (SYNC_OPTION_xxx)   [No]*/ //EDC220+
  WORD    MachineNoIoBitConnector;      /* BitIn/BitOut connector          [No]*/ //EDC220+
  WORD    MachineNoIoBitNo;             /* First bit of block          [0..15]*/ //EDC220+
} DoPEGeneralData;
```

12.7 DoPESelSetup

Select a machine set-up and initialize.
Set-up number 1 to 4 (8) are stored inside EDC in an EEPROM. Set-up number 0 is the working set-up. If set-up 1 to 4 (8) 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 initialize the system.TheDoPESelSetup function calls DoPEInitialize to do system initialization.

Function declaration		Description
extern unsigned DLLAPI	DoPESelSetup (	Function returns Error constant (DoPERR_xxxx)
DoPE_HANDLE	DoPEHdl	DoPE link handle
unsigned short	SetupNo	Set-up No.
UserScale	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.If US = NULL, user scale will be set to 1.0
WORD	* lpusTANFirst	Pointer to transaction number (not for Sync. version).
WORD	* lpusTANLast)	

12.8 DoPEInitialize

Initialize 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.

Function declaration		Description
extern unsigned DLLAPI	DoPEInitialize (	Function returns Error constant (DoPERR_xxxx)
DoPE_HANDLE	DoPEHdl,	DoPE link handle
WORD	*lpusTANFirst,	TAN's are used to identify initialization errors sent in addition to the synchronized return code.
WORD	*lpusTANLast,)	(This will be implemented in future EDC versions.In Version 2.11 we have synchronized return codes but onlyone error code sent on the PE_INITIALIZE command.)

12.9 DoPEInitializeResetXHead(Synch)

Initialize System with selected setup data and reset the crosshead position.	
Minimum requirements: EDC580/220, EdcApp 9133.002, DoPE 2.50	
Function declaration	**Description**
extern unsigned DLLAPI DoPEInitializeResetXHead (DoPE_HANDLE DoPEHdl, WORD *lpusTANFirst, WORD *lpusTANLast,)	Function returns Error constant (DoPERR_xxxx) DoPE link handle TAN's are used to identify initialization errors sent inaddition to the synchronized return code. (This will be implemented in future EDC versions.In Version 2.11 we have synchronized return codes but onlyone error code sent on the PE_INITIALIZE command.)

12.10DoPERdSensorInfo

Read summary sensor information's.

Function declaration		Description	
extern unsigned DLLAPI	DoPERdSensorInfo (	Function returns Error constant (DoPERR_xxxx)	
DoPE_HANDLE	DoPEHdl	DoPE link handle	
unsigned short	SensorNo	Sensor Number.	
DoPESumSenInfo	*Info	Pointer for SumSenInfo structure.	

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;
```

12.11DoPERdSysUserData

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

Function declaration		Description	
extern unsigned DLLAPI	DoPERdSysUserData (	Function returns Error constant (DoPERR_xxxx)	
DoPE_HANDLE	DoPEHdl	DoPE link handle	
BYTE	*SysUsrData	Pointer for SYSEEPROM user data	
unsigned	Length	User data buffer length in BYTE's	

12.12DoPEWrSysUserData(Sync)

Write system user data (16 Byte).

Function declaration		Description	
extern unsigned DLLAPI	DoPEWrSysUserData (	Function returns Error constant (DoPERR_xxxx)	
DoPE_HANDLE	DoPEHdl	DoPE link handle	
BYTE	*SysUsrData	Pointer for SYSEEPROM user data	
unsigned	Length	User data buffer length in BYTE's	
WORD	*lpusTAN)	Pointer to transaction number (not for Sync. version).	

12.13DoPERdSenUserData

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!

Function declaration		Description	
extern unsigned DLLAPI	DoPERdSenUserData (	Function returns Error constant (DoPERR_xxxx)	
DoPE_HANDLE	DoPEHdl	DoPE link handle	
WORD	SensorNo	Sensor Number	
BYTE	*SenUsrData	Pointer for sensor user data	
unsigned	Length	User data buffer length in BYTE's	

12.14DoPEWrSenUserData(Sync)

Write sensor user data (128 Byte).

Function declaration		Description	
extern unsigned DLLAPI	DoPEWrSenUserData (	Function returns Error constant (DoPERR_xxxx)	
DoPE_HANDLE	DoPEHdl	DoPE link handle	
WORD	SensorNo	Sensor Number	
BYTE	*SenUsrData	Pointer for sensor user data	
unsigned	Length	User data buffer length in BYTE's	
WORD	*lpusTAN)	Pointer to transaction number (not for Sync. version).	

12.15DoPERdSensorUserData

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!

Function declaration		Description	
extern unsigned DLLAPI	DoPERdSensorUserData (	Function returns Error constant (DoPERR_xxxx)	
DoPE_HANDLE	DoPEHdl	DoPE link handle	
WORD	Connector	Connector number of sensor	
BYTE	*SenUsrData	Pointer for sensor user data	
unsigned	Length	User data buffer length in BYTE's	

12.16DoPEWrSensorUserData

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).

Function declaration		Description	
extern unsigned DLLAPI	DoPEWrSensorUserData (	Function returns Error constant (DoPERR_xxxx)	
DoPE_HANDLE	DoPEHdl	DoPE link handle	
WORD	Connector	Connector number of sensor	
BYTE	*SenUsrData	Pointer for sensor user data	
unsigned	Length	User data buffer length in BYTE's	
WORD	*lpusTAN)	Pointer to transaction number (not for Sync. version).	

12.17DoPERdModuleInfo / DoPEwRdModuleInfo

Read module info.

Minimum requirements:
DoPERdModuleInfo: DoPE 2.50
DoPEwRdModuleInfo: DoPE 2.65

Function declaration	Description
extern unsigned DLLAPI DoPERdModuleInfo (DoPE_HANDLE DoPEHdl unsigned ModulNo DoPEModuleInfo *Info	Function returns Error constant (DoPERR_xxxx) DoPE link handle Module number (0 ... MAX_MODULE-1) Pointer to storage for the module info

```
typedef struct                          /* Definition of module info          */
{                                       /* ---------------------------------- */
  DWORD  ID;                            /* Module hardware ID             [No]*/
  DWORD  Revision;                      /* Module hardware revision       [No]*/
  DWORD  DeviceID;                      /* Device ID                      [No]*/
  DWORD  FunctionID;                    /* Function ID (from MachineDef)  [No]*/
  DWORD  SerNr;                         /* Serial number                  [No]*/
  DWORD  Status;                        /* Module status                  [No]*/
  char   Name[MODINFO_NAME_LEN];        /* Module name (e.g. "2INC 1742.000(-)") */
} DoPEModuleInfo;
```

Function declaration	Description
extern unsigned DLLAPI DoPEwRdModuleInfo (DoPE_HANDLE DoPEHdl unsigned ModulNo DoPEwModuleInfo *Info	Function returns Error constant (DoPERR_xxxx) DoPE link handle Module number (0 ... MAX_MODULE-1) Pointer to storage for the module info

```
typedef struct                          /* Definition of module info          */
{                                       /* ---------------------------------- */
  DWORD  ID;                           /* Module hardware ID             [No]*/
  DWORD  Revision;                     /* Module hardware revision       [No]*/
  DWORD  DeviceID;                     /* Device ID                      [No]*/
  DWORD  FunctionID;                   /* Function ID (from MachineDef)  [No]*/
  DWORD  SerNr;                        /* Serial number                  [No]*/
  DWORD  Status;                       /* Module status                  [No]*/
wchar_tName[MODINFO_NAME_LEN];        /* Module name (e.g. "2INC 1742.000(-)") */
} DoPEModuleInfo;
```

12.18DoPERdDriveInfo / DoPEwRdDriveInfo

Read drive info.

Minimum requirements:
DoPERdDriveInfo DoPE 2.57
DoPEwRdDriveInfo DoPE 2.65

Function declaration		Description
extern unsigned DLLAPI	DoPERdDriveInfo(	Function returns Error constant (DoPERR_xxxx)
DoPE_HANDLE	DoPEHdl	DoPE link handle
WORD	Connector	Module number (0 ... MAX_MODULE-1)
DoPEDriveInfo	*Info	Pointer to storage for the drive info

```
typedef struct                          /* Definition of drive info           */
{                                       /* ---------------------------------- */
  DWORD  ID;                            /* Drive hardware ID              [No]*/
  DWORD  Revision;                      /* Drive hardware revision        [No]*/
  DWORD  SerNr;                         /* Serial number                  [No]*/
  double NominalCurrent;                /* Nominal current                 [A]*/
  double MinCurrent;                    /* Minimum current                 [A]*/
  double MaxCurrent;                    /* Maximum current                 [A]*/
  double MaxCurrentTime;                /* Maximum current time (I²t)      [s]*/
  int    NominalVoltageCount;           /* Number of nominal voltages     [No]*/
  double NominalVoltage[MAX_NOMV];      /* Nominal voltage                 [V]*/
  double MaxPower;                      /* Maximum power                   [W]*/
  double FeedbackPower;                 /* Average feedback power          [W]*/
  double MinDither;                     /* Minimum dither frequency       [Hz]*/
  double MaxDither;                     /* Maximum dither frequency       [Hz]*/
  double CurrentControllerCycle;        /* Current controller cycle time   [s]*/
                                        /* (0 = analogue or not adjustable)   */
  char   Name[DRIVEINFO_NAME_LEN];      /* Drive name (e.g. "DC2500 1860.000(-)") */
} DoPEDriveInfo;
```

Function declaration		Description
extern unsigned DLLAPI	DoPEwRdDriveInfo(	Function returns Error constant (DoPERR_xxxx)
DoPE_HANDLE	DoPEHdl	DoPE link handle
WORD	Connector	Module number (0 ... MAX_MODULE-1)
DoPEDriveInfo	*Info	Pointer to storage for the drive info

```
typedef struct                          /* Definition of drive info           */
{                                       /* ---------------------------------- */
  DWORD  ID;                            /* Drive hardware ID              [No]*/
  DWORD  Revision;                      /* Drive hardware revision        [No]*/
  DWORD  SerNr;                         /* Serial number                  [No]*/
  double NominalCurrent;                /* Nominal current                 [A]*/
  double MinCurrent;                    /* Minimum current                 [A]*/
  double MaxCurrent;                    /* Maximum current                 [A]*/
  double MaxCurrentTime;                /* Maximum current time (I²t)      [s]*/
  int    NominalVoltageCount;           /* Number of nominal voltages     [No]*/
  double NominalVoltage[MAX_NOMV];      /* Nominal voltage                 [V]*/
  double MaxPower;                      /* Maximum power                   [W]*/
  double FeedbackPower;                 /* Average feedback power          [W]*/
  double MinDither;                     /* Minimum dither frequency       [Hz]*/
  double MaxDither;                     /* Maximum dither frequency       [Hz]*/
  double CurrentControllerCycle;        /* Current controller cycle time   [s]*/
                  /* (0 = analogue or not adjustable)   */
wchar_tName[DRIVEINFO_NAME_LEN];    /* Drive name (e.g. "DC2500 1860.000(-)") */
} DoPEDriveInfo;
```

13 Sensor EEPROM Handling

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

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

These functions work on initialized and not initialized 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.
   Analyze the sensor class!
3. For DoPESensorHeaderData.Class = SEN_ANALOGUE useDoPERdSensorAnalogueData,
   for DoPESensorHeaderData.Class =SEN_INC useDoPERdSensorIncData,
   for DoPESensorHeaderData.Class =SEN_ABS useDoPERdSensorAbsData
   to read the sensor class specific data.
4. Modify data.
   Only the **bold type**parameter in the sensor data structures is 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!**

13.1 DoPERdSensorConKey

Read sensor plug connected and key state.			
Minimum requirements: DoPE 2.23			
Function declaration		**Description**	
extern unsigned DLLAPI	DoPERdSensorConKey (	Function returns Error constant (DoPERR_xxxx)	
DoPE_HANDLE	DoPEHdl	DoPE link handle	
WORD	Connector	Connector number of sensor	
WORD	*Connected	Pointer to the sensor plug connected state (0=not connected, 1=connected)	
WORD	* KeyPressed	Pointer to the sensor plug key state (0=not pressed, 1=pressed	

13.2 DoPERdSensorHeaderData

Read sensor EEPROM data header.

Minimum requirements: DoPE 2.23

Function declaration	Description
extern unsigned DLLAPI DoPERdSensorHeaderData (DoPE_HANDLE DoPEHdl WORD Connector DoPESensorHeaderData *SenHdrData	Function returns Error constant (DoPERR_xxxx) DoPE link handle Connector number of sensor Pointer to sensor data header structure

13.3 DoPEWrSensorHeaderData

Write sensor EEPROM data header.

Minimum requirements: DoPE 2.76

Function declaration	Description
extern unsigned DLLAPI DoPEWrSensorHeaderData (DoPE_HANDLE DoPEHdl WORD Connector DoPESensorHeaderData *SenHdrData	Function returns Error constant (DoPERR_xxxx) DoPE link handle Connector number of sensor Pointer to sensor data header structure

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;
                                           Write protected

                                        /* 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      */
```

13.4 DoPERdSensorAnalogueData

Read analogue sensor data.

Minimum requirements: DoPE 2.23

Function declaration		Description	
extern unsigned DLLAPI DoPERdSensorAnalogueData (		Function returns Error constant (DoPERR_xxxx)	
DoPE_HANDLE	DoPEHdl	DoPE link handle	
WORD	Connector	Connector number of sensor	
DoPESensorAnalogueData	*SenAnalogueData	Pointer to analogue sensor data structure	

13.5 DoPEWrSensorAnalogueData

Write analogue sensor data.

Minimum requirements: DoPE 2.23

Function declaration		Description	
extern unsigned DLLAPI DoPEWrSensorAnalogueData (		Function returns Error constant (DoPERR_xxxx)	
DoPE_HANDLE	DoPEHdl	DoPE link handle	
WORD	Connector	Connector number of sensor	
DoPESensorAnalogueData	*SenAnalogueData	Pointer to analogue sensor data structure	

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;
                                        Write protected
```

13.6 DoPERdSensorIncData

Read incremental sensor data.

Minimum requirements: DoPE 2.23

Function declaration		Description	
extern unsigned DLLAPI	DoPERdSensorIncData (	Function returns Error constant (DoPERR_xxxx)	
DoPE_HANDLE	DoPEHdl	DoPE link handle	
WORD	Connector	Connector number of sensor	
DoPESensorIncData	*SenIncData	Pointer to incremental sensor data structure	

13.7 DoPEWrSensorIncData

Write incremental sensor data.

Minimum requirements: DoPE 2.23

Function declaration		Description	
extern unsigned DLLAPI	DoPEWrSensorIncData (	Function returns Error constant (DoPERR_xxxx)	
DoPE_HANDLE	DoPEHdl	DoPE link handle	
WORD	Connector	Connector number of sensor	
DoPESensorIncData	*SenIncData	Pointer to incremental sensor data structure	

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;                /* Transducer 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;                    /* Transducer 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 transducer[Hz]*/
  WORD   Sign;                          /* Invert sign of channel        [1/0]*/
  BYTE   NegLimit;                      /* Range limit - min.              [%]*/
  BYTE   PosLimit;                      /* Range limit - max.              [%]*/
} DoPESensorIncData;
                                        Write protected
```

13.8 DoPERdSensorAbsData

Read absolute sensor data.

Minimum requirements: DoPE 2.23

Function declaration		Description	
extern unsigned DLLAPI DoPERdSensorAbsData (		Function returns Error constant (DoPERR_xxxx)	
DoPE_HANDLE	DoPEHdl	DoPE link handle	
WORD	Connector	Connector number of sensor	
DoPESensorAbsData	*SenAbsData	Pointer to absolute sensor data structure	

13.9 DoPEWrSensorAbsData

Write absolute sensor data.

Minimum requirements: DoPE 2.23

Function declaration		Description	
extern unsigned DLLAPI DoPEWrSensorAbsData (		Function returns Error constant (DoPERR_xxxx)	
DoPE_HANDLE	DoPEHdl	DoPE link handle	
WORD	Connector	Connector number of sensor	
DoPESensorAbsData	*SenAbsData	Pointer to absolute sensor data structure	

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   [ms/10]*/
  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;                    /* Tansducer sgnal type  SIG_xxx  [No]*/
                                        /* or SIG_SSI_GENERIC + code + number */
                                        /* of data bits                       */
  float  LimitFrequency;                /* Limit frequency of the transducer[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;
                                        Write protected
```

13.10DoPESetSsiGenericSignalType

Build the signal type of a SSI absolute sensor by it's generic data.			
Minimum requirements: DoPE 2.66			
Function declaration		**Description**	
extern unsigned DLLAPI BYTE unsigned unsigned	DoPESetSsiGenericSignalType(*SignalType Code Bits	Function returns Error constant (DoPERR_xxxx) Pointer to signal type variable Code of the sensor (0:binary, !=0:gray code) Number of databits of the sensor	

13.11DoPESsiGenericSignalTypeInfo

Get the generic data of a SSI absolute sensor by it's signal type.			
Minimum requirements: DoPE 2.66			
Function declaration		**Description**	
extern unsigned DLLAPI BYTE unsigned unsigned	DoPESsiGenericSignalTypeInfo(SignalType * Code * Bits	Function returns Error constant (DoPERR_xxxx) Signal type of the SSI sensor Pointer to storage for code of the sensor (code = 0:binary, !=0:gray code) Pointer to storage for number of databits of the sensor	

14 Reestablishing a Connection

For long-term experiments, it may be useful to close an open link without interrupting a running movement command. At a later time the connection can be resumed. For this DoPE provides two funtions. One enables the so called ReInitialize Mode and retreves the current state of the connection in the ReInitialize Data. The other one recovers the state of the connetion with this data.

14.1 DoPEReInitializeEnable

Activates the ReInitialize Mode and provides ReInitialize Data. This data will be passed to the DoPEReInitialize function to restore the communication link to the current state.
DoPESelSetup and the DoPEinitialize functions set the the ReInitialize Mode to the disabled state.

Minimum requirements: EDC220V/580V, EdcApp 9140.21, DoPE 2.84

Function declaration	Description
extern unsigned DLLAPI DoPEReInitializeEnable (DoPE_HANDLE DoPEHdl unsigned Enable DoPEReInitializeData* ReInitData)	Function returns Error constant (DoPERR_xxxx) DoPE link handle !=0 enables, =0 disables the ReInitialize mode pointer to storage for ReInitialize data

```
#define DoPEREINITIALIZEDATA_LEN 0x8000

typedef
  BYTE DoPEReInitializeData [DoPEREINITIALIEDATA_LEN];  /* ReInitialize data array [No]*/
```

14.2 DoPEReInitialize

Recover a previously disconnected communication link.

Attention: DoPEReInitialize checks that:
- the EDC is still in ReInitialize Mode.
- the DoPEAPIVERSION of the current DoPE session matches the one used when the ReInitData were retreaved.
- the DeviceID didn't change.
- the ReInitData contains valid data (CRC check).

Minimum requirements: EDC220V/580V, EdcApp 9140.21, DoPE 2.84

Function declaration	Description
extern unsigned DLLAPI DoPEReInitialize (DoPE_HANDLE DoPEHdl DoPEReInitializeData *ReInitData)	Function returns Error constant (DoPERR_xxxx) DoPE link handle pointer to ReInitialize data

Here a "C" sample program:

First session of the PC Program:

```
// open the communication link
DoPEErr = DoPEErr = DoPEOpenFunctionID( ... &DoPEHdl );
if( DoPEErr != DoPERR_NOERROR)
  //add errorhandling here
    return 0;

// select a setup
DoPEErr = DoPESelSetup ( DoPEHdl, 1, UserScale, NULL, NULL );
if(DoPEErr != DoPERR_NOERROR)
  //add errorhandling here

// add event handler and other application stuff
```

```
…

// switch drive on
DoPEErr = DoPEOn ( DoPEHdl );
if(DoPEErr != DoPERR_NOERROR)
  //add errorhandling here

// start a long term test
DoPEErr = DoPEDynCycle ( DoPEHdl, … HalfCycles = 10000000, … );
if(DoPEErr != DoPERR_NOERROR)
  //add errorhandling here

// enable the Reinitialize Mode and keep the ReInitialize Data
DoPEReInitializeData ReInitData;
DoPEErr = DoPEReInitializeMode ( DoPEHdl, true, &ReInitData );
if(DoPEErr != DoPERR_NOERROR)
  //add errorhandling here

…save ReInitDat to Disk…


// Close the communication link
DoPEErr = DoPECloseLink ( &DoPEHdl );
if(DoPEErr != DoPERR_NOERROR)
  //add errorhandling here
```

Recovery session of the PC Program:

```
// open the communication link
DoPEErr = DoPEErr = DoPEOpenFunctionID( ... &DoPEHdl );
if( DoPEErr != DoPERR_NOERROR)
  //add errorhandling here
    return 0;

// recover communication link with previously stored ReInitData
…load ReInitData from disk
DoPEErr = DoPEReInitialize ( DoPEHdl, &ReInitData );
if(DoPEErr != DoPERR_NOERROR)
  //add errorhandling here


// add event handler and other application stuff
…
```

15 Default measuring data record

unsigned long	Cycles	Cycle counter
double	Time	Time from subsystem
double	Position	X-Head position
double	Load	Load
double	Extension	Extension
double	Sensor3	Sensor 3 measuring channel
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 15.1
unsigned short	OutSignals	Logical output signals definition see15.2
unsigned short	CtrlState1	Controller status WORD 1 definition see 15.3
unsigned short	CtrlState2	Controller status WORD 2 definition see 15.4
unsigned short	UpperLimits	Upper range limits exceeded
unsigned short	LowerLimits	Lower range limits exceeded
unsigned short	SysState0	System status WORD 0
unsigned short	ActiveCtrl	Active control channel (Bit0=Position…Bit15=Sensor15)
unsigned short	UpperSft	Upper soft limit active (Bit0=Position…Bit15=Sensor15)
unsigned short	LowerSft	Lower soft limit active (Bit0=Position…Bit15=Sensor15)
unsigned short	SensorConnected	Sensor plug connected state (Bit0=Position…Bit15=Sensor15)
unsigned short	SensorKeyPressed	Sensor plug key state (Bit0=Position…Bit15=Sensor15)
double	Test1	Configured test value 1 (Command)
double	Test2	Configured test value 2 (Feedback)
double	Test3	Configured test value 3 (Output)
unsigned short	Keys	Current state of EDC front panel keys (obsolete, use OnKey handler instead)
unsigned short	NewKeys	New keys (obsolete, use OnKey handler instead)
unsigned short	GoneKeys	Gone keys (obsolete, use OnKey handler instead)

15.1 Logical input signals

The logical input signals are safety relevant digital inputs. They are configured during initialization 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 data record (InSignals).

Bit number	Constant	Description
0	IN_SIG_DRIVE_OFF	Drive off or emergency off 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 or similar safety equipment.
1	IN_SIG_E_MOVE_RQ	Emergency off, emergency movement required This signal also forces the machine control into the emergency stop state, but can be masked during a driving free movement.
2	IN_SIG_UPPER_LIMIT_SWITCH	Upper hard-limit switch active This signal force the machine control into the emergency stop state and can be masked during a driving free movement.
3	IN_SIG_LOWER_LIMIT_SWITCH	Lower hard-limit switch active This signal force the machine control into the emergency stop state and can be masked during a driving free movement.
4	IN_SIG_NO_CTRL	Drive not ready (stop) This signal reports the state of the connected drive unit. If set, the drive is not ready.
5	IN_SIG_DF_KEY	Drive free withdrawn by key
6	IN_SIG_SHALT	Signal S-HALT activated 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.
7	IN_SIG_REFERENZ	X-head reference switch This signal reports the state a reference switch at the X-Head.
8	IN_SIG_SH_CLOSED	Shield closed This signal is set when a connected protective shield is closed.
9	IN_SIG_SH_LOCKED	Shield locked This signal is set when a connected protective shield is locked.
10	IN_SIG_SH_INACTIVE	Shield function inactive This signal is set when the connected protective shield is disabled.
11	IN_SIG_IO_SHALT_UPPER	IO-Signal ShaltUpper
12	IN_SIG_IO_SHALT_LOWER	IO-Signal ShaltLower
13	IN_SIG_CPU_EMERGENCY_OFF	Internal emergency off
14, 15		reserved

15.2 Logical output signals

The logical output signals are digital outputs to control the drive and associated devices. They are configured during initialization. The state of the logical output signals is maintained in the measuring data record (OutSignals).

Bit number	Constant	Description
0	O_SIG_DRIVE_ON	Drive ON output 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 Free' (see below) without delay.
1	O_SIG_DRIVE_FREE	Drive enable output With this signal, the external drive may be enabled and disabled.
2	O_SIG_BRAKE	Brake output 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.
3	O_SIG_E_MOVE	Emergency movement output 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.
4	O_SIG_BYPASS	Bypass output
5	O_SIG_LIMIT	Load limit for grip reached This signal is used by the measuring channel supervision activated by DoPESetCheckLimit command.
6	O_SIG_SH_LOCK	Lock Shield The signal 'Lock Screen' is used by the screen (shield) supervision functions to lock the shield.
7	O_SIG_ON_RELAY	ON Relay output
8		reserved
9	O_SIG_INTERNAL_DRIVE_ENABLE	Internal Drive enable /disable
10	O_SIG_READY_TO_MOVE	Controller is ready for positioning .commands
11	O_SIG_EDC_READY	EDC ready
12 … 15		reserved

15.3 Controller Status WORD 1

Status word 1 of the closed loop controller (CtrlState1)		
Bit number	**Constant**	**Description**
0	CTRL_STATE_POS	X-Head control is active
1	CTRL_STATE_LOAD	Load control is active
2	CTRL_STATE_EXT	Extension control is active
3		Reserved
4	CTRL_HALT	Command generator halts (controlled halt in S/F/E).
5	CTRL_DOWN	Movement DOWN
6	CTRL_UP	Movement UP
7	CTRL_MOVE	X-Head moves (not halted) 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.
8	CTRL_READY	Moving command will be accepted
9	CTRL_FREE	Waiting for free signal. The EDC waits for the drive enable through the Host software or from the user.
10	CTRL_INIT_E	Emergency movement has to be activated
11	CTRL_SFTSET	Change of softends allowed
12		Reserved
13	CTRL_SYNCHWAIT	Synch State: 1 = wait for Start
14	CTRL_SLAVE	Synch State: 0 = Master 1 = Slave
15	CTRL_E_ACTIVE	Emergency movement active

15.4 Controller Status WORD 2

Status word 2 of the closed loop controller (CtrlState2)		
Bit number	**Constant**	**Description**
0	CTRL_LOWER_SFT_S	Lower position softend exceeded
1	CTRL_LOWER_SFT_F	Lower load softend exceeded
2	CTRL_LOWER_SFT_E	Lower extension softend exceeded
3		reserved
4	CTRL_UPPER_SFT_S	Upperpositionsoftend exceeded
5	CTRL_UPPER_SFT_F	Upperloadsoftend exceeded
6	CTRL_UPPER_SFT_E	Upperextensionsoftend exceeded
7		reserved
8		reserved
9		reserved
10	CTRL_CYCLES_ACTIVE	Cycle command active (cosine, rectangle, triangle, cycles, dyn_cycles)
11	CTRL_HIGH_PRESSURE	High pressure active
12	CTRL_CALIBRATION	Calibration of analogue channels is currently active.
13	CTRL_LOWER_LIMIT	Lower range limit of any sensor is active.
14	CTRL_UPPER_LIMIT	Upper range limit of any sensor is active.
15	CTRL_ERROR	Deviation position controller

15.5 Controller Status ActiveCtrl

ActiveCtrl shows the currently active sensor for closed loop control	
Bit number	**Description**
0	X-Head control is active
1	Load control is active
2	Extension control is active
3	Sensor3 control is active
4	Sensor4 control is active
5	Sensor5 control is active
6	Sensor6 control is active
7	Sensor7 control is active
8	Sensor8 control is active
9	Sensor9 control is active
10	Sensor10 control is active
11	Sensor11 control is active
12	Sensor12 control is active
13	Sensor13 control is active
14	Sensor14 control is active
15	Sensor15 control is active

16 DoPE System message

Number	Message constant	Description
0	SYSTEM_MSG_NOERROR	No error
General runtime errors		
10000	SYSTEM_MSG_UNKNOWN	Unknown runtime error
10001	SYSTEM_MSG_POWERAMP_ERROR	Power amplifier error
10002	SYSTEM_MSG_DRIVE_ERROR	Drive error
10003	SYSTEM_MSG_DRIVE_NOTREADY	Drive not ready
10004	SYSTEM_MSG_DRIVE_RELEASE	Drive release withdrawn
10005	SYSTEM_MSG_UPPER_SENLIMIT	Upper sensor limit exceeded
10006	SYSTEM_MSG_LOWER_SENLIMIT	Lower sensor limit exceeded
10007	SYSTEM_MSG_LIMITSWITCH	Upper or lower hard-limit switch
10008	SYSTEM_MSG_DEVIATION	Controller deviation error
10009	SYSTEM_MSG_NODIGIPOTI	No DigiPoti configurated
10010	SYSTEM_MSG_NORMC	No remote control plugged
10011	SYSTEM_MSG_EMERGENCY_OFF	Emergency switch activated
10012	SYSTEM_MSG_NOMEMORY	not enough memory
10013	SYSTEM_MSG_NOSHIELD	Shield not found
10014	SYSTEM_MSG_X2T_OFFLINE	X2T module offline
10015	SYSTEM_MSG_MAX_RESULT_FILES	Max. number of result files reached
System EEPROM errors (flash disk)		
10100	SYSTEM_MSG_SYSEEP_CRC	data CRC error
10101	SYSTEM_MSG_SYSEEP_NOBLOCK	block not found
10102	SYSTEM_MSG_SYSEEP_BLOCKLENGTH	wrong block length
10103	SYSTEM_MSG_SYSEEP_NOMEMORY	not enough EEPROM memory
10104	SYSTEM_MSG_SYSEEP_BIOSNOFUNC	BIOS: function not found
10105	SYSTEM_MSG_SYSEEP_BIOSNODEVICE	BIOS: device error
10106	SYSTEM_MSG_SYSEEP_BIOSPARA	BIOS: parameter error
10107	SYSTEM_MSG_SYSEEP_BIOSREAD	BIOS: read error
10108	SYSTEM_MSG_SYSEEP_BIOSWRITE	BIOS: write error
10109	SYSTEM_MSG_SYSEEP_BIOSREENT	BIOS: reentrance error
General initialization errors		
10200	SYSTEM_MSG_INIT_ENDDEV	Error at initialization level 1
10201	SYSTEM_MSG_INIT_ENDINI	Error at initialization level 2
10202	SYSTEM_MSG_INIT_SYSTIME	Wrong system time
10203	SYSTEM_MSG_INIT_POSCTRLTIME	Wrong position controller time
10204	SYSTEM_MSG_INIT_DATA_TRANSMISSION_RATE	Wrong data transmission rate
10205	SYSTEM_MSG_INIT_GRIP	Gripinitialization error
10206	SYSTEM_MSG_INIT_SHIELD	Shield initialization error
10207	SYSTEM_MSG_INIT_MAINBOARD	EDC main board initialization error
10208	SYSTEM_MSG_INIT_MODULE_ERROR	Module error
10209	SYSTEM_MSG_INIT_CTRL	Error closed loop controller
Sensor initialization errors		
10300	SYSTEM_MSG_SEN_EEP_NOTFOUND	Sensor EEPROM not found
10301	SYSTEM_MSG_SEN_EEP_BCC	Sensor EEPROM data BCC error
10302	SYSTEM_MSG_SEN_EEP_CLASS	Unknown sensor EEPROM class
10303	SYSTEM_MSG_SEN_NOTFOUND	Sensor not found
10304	SYSTEM_MSG_SEN_INITBYTE	Initialization word error
10305	SYSTEM_MSG_SEN_INIT	Sensor initialization error
10306	SYSTEM_MSG_SEN_PARA	Wrong sensor parameter
10307	SYSTEM_MSG_SEN_CORR	Wrong sensor correction value
10308	SYSTEM_MSG_SEN_MCINTEGR	Wrong integration time for measured channel values
10309	SYSTEM_MSG_SEN_CTRLINTEGR	Wrong integration time for closed loop control
10310	SYSTEM_MSG_SEN_LIMIT	Wrong sensor limits

10311	SYSTEM_MSG_SEN_NOMINALACC	Wrong nominal acceleration
10312	SYSTEM_MSG_SEN_NOMINALSPEED	Wrong nominal speed
10313	SYSTEM_MSG_SEN_POS_CTRL	Wrong parameter for position closed loop control
10314	SYSTEM_MSG_SEN_SPEED_CTRL	Wrong parameter for speed closed loop control
10315	SYSTEM_MSG_SEN_BIOS	Wrong BIOS Version
10316	SYSTEM_MSG_SEN_OFFSET	Wrong sensor offset
10317	SYSTEM_MSG_SEN_SCALE	Wrong sensor scale
Output channel initialization errors		
10400	SYSTEM_MSG_OC_EEP_NOTFOUND	Output channel EEPROM not found
10401	SYSTEM_MSG_OC_EEP_BCC	Output channel EEPROM data BCC error
10402	SYSTEM_MSG_OC_EEP_CLASS	Unknown Output channel EEPROM class
10403	SYSTEM_MSG_OC_INIT	Output channel initialization error
10404	SYSTEM_MSG_OC_VOLTAGE	Wrong voltage for DDAxx
10405	SYSTEM_MSG_OC_CURRENT	Wrong current for DDAxx
10406	SYSTEM_MSG_OC_POWER	Wrong power for DDAxx
10407	SYSTEM_MSG_OC_PARA	Wrong output channel parameter
10408	SYSTEM_MSG_OC_MAX_CURR_TIME	Wrong max current time for I²T
10409	SYSTEM_MSG_OC_DITHER_FREQ	Wrong dither frequency
10410	SYSTEM_MSG_OC_DITHER_AMPL	Wrong dither amplitude
10411	SYSTEM_MSG_OC_DITHER_INIT	dither initialization error
10412	SYSTEM_MSG_OC_CURRENT_CTRL	Wrong parameter for current closed loop control
10413	SYSTEM_MSG_OC_MC2OUTPUT	Mc2Output initialization error
10414	SYSTEM_MSG_OC_MC2OUTPUT	Mc2Output mode not 153mplemented
Bit input/output initialization errors		
10500	SYSTEM_MSG_BIN_INIT	Bit input initialization error
10600	SYSTEM_MSG_BOUT_INIT	Bit output initialization error
IO-Signals initialization errors		
10700	SYSTEM_MSG_IO_MACHINE_CONNECTOR	IOMachine: connector error
10710	SYSTEM_MSG_IO_GRIP_MODE	IOGrip: Mode not implemented
10711	SYSTEM_MSG_IO_GRIP_BIN	IOGrip: Bit input error
10712	SYSTEM_MSG_IO_GRIP_BOUT	IOGrip: Bit output error
10713	SYSTEM_MSG_IO_GRIP_OC	IOGrip: Output channel error
10714	SYSTEM_MSG_IO_GRIP_LIMIT	IOGrip: Limit error
10715	SYSTEM_MSG_IO_GRIP_OLD_LIMIT	IOGrip: Conflict with old grip limit
10730	SYSTEM_MSG_IO_EXT_MODE	IOExtensometer: Mode not implemented
10731	SYSTEM_MSG_IO_EXT_BIN	IOExtensometer: Bit input error
10732	SYSTEM_MSG_IO_EXT_BOUT	IOExtensometer: Bit output error
10740	SYSTEM_MSG_IO_FIXED_XHEAD_MODE	IOFixedXHead: Mode not implemented
10741	SYSTEM_MSG_IO_FIXED_XHEAD_BIN	IOFixedXHead: Bit input error
10742	SYSTEM_MSG_IO_FIXED_XHEAD_BOUT	IOFixedXHead: Bit output error
10750	SYSTEM_MSG_IO_HIGH_PRESSURE_MODE	IOHighPressure: Mode not implemented
10751	SYSTEM_MSG_IO_HIGH_PRESSURE_BIN	IOHighPressure: Bit input error
10752	SYSTEM_MSG_IO_HIGH_PRESSURE_BOUT	IOHighPressure: Bit output error
10753	SYSTEM_MSG_IO_HIGH_PRESSURE_OC	IOHighPressure: Output channel error
10780	SYSTEM_MSG_IO_MISC_MODE	IOMisc: Mode not implemented
10781	SYSTEM_MSG_IO_MISC_BIN	IOMisc: Bit input error
10782	SYSTEM_MSG_IO_MISC_BOUT	IOMisc: Bit output error
10790	SYSTEM_MSG_IO_MISC_TEMPERATURE1	IOMisc: temperature1 (warning)
10791	SYSTEM_MSG_IO_MISC_TEMPERATURE2	IOMisc: temperature2 (emergency off)
10792	SYSTEM_MSG_IO_MISC_OIL_LEVEL	IOMisc: oil level(emergency off)
10793	SYSTEM_MSG_IO_MISC_OIL_FILTER	IOMisc: oil filter (warning)
10794	SYSTEM_MSG_IO_MISC_POWER_FAIL	IOMisc: power fail (emergency off)
10850	SYSTEM_MSG_IO_SHALT_MODE	IOSHalt: Mode not implemented
10851	SYSTEM_MSG_IO_SHALT_BIN	IOSHalt: Bit input error
10900	SYSTEM_MSG_LANGUAGE_READ	Language file: read error
10901	SYSTEM_MSG_LANGUAGE_VERSION	Language file: wrong version
10902	SYSTEM_MSG_LANGUAGE_SYNTAX	Language file: syntax errors

17 DoPE Error constants

Error number	Error constant	Description
0	DoPERR_NOERROR	No error
1	DoPERR_NOFLOAT	No float in WIN16 callback
2	DoPERR_SYNC	Synchronization to callback failed
3	DoPERR_TIMEOUT	Timeout at await answer
4	DoPERR_NOFNC	Function not implemented
5	DoPERR_VERSION	No compatible Version EDC-DoPE
6	DoPERR_INIT	Initialization Error Subsystem
7	DoPERR_PARAMETER	Invalid parameter
8	DoPERR_SETUPOPEN	Set-up open error
9	DoPERR_RTE_UNHANDLED	Unhandled runtime error
Command errors		
1001	DoPERR_CMD_PARCORR	Error in parameter (corrected)
1003	DoPERR_CMD_PAR	Error in parameter. Not correctable
1004	DoPERR_CMD_XMOVE	X-Head is not halted
1005	DoPERR_CMD_INITSEQ	Sequence in init not observed
1006	DoPERR_CMD_NOTINIT	Controller part not initialized
1007	DoPERR_CMD_DIR	Movement direction not possible
1008	DoPERR_CMD_TMP	Required resource not available
1009	DoPERR_CMD_RUNTIME	Run time error active
1010	DoPERR_CMD_INTERN	Internal error in subsystem
1011	DoPERR_CMD_MEM	Insufficient memory
1012	DoPERR_CMD_CST	Wrong controller Structure
1013	DoPERR_CMD_NIM	Command not implemented
2001	DoPERR_CMD_MSGNO	Unknown message number
2003	DoPERR_CMD_VERSION	Wrong PE interface version
2004	DoPERR_CMD_OPEN	Set-up not opened
2005	DoPERR_CMD_MEMORY	Not enough memory
Machine normalization errors		
0x4001	DoPERR_PARMS	Parameter Error
0x4002	DoPERR_ZERODIV	Division by ZERO
0x4003	DoPERR_OVFLOW	Overflow
0x4004	DoPERR_NIN	Not Initialized
Low level communication errors		
0x8001	DoPERR_NODATA	No receiver data available
0x8002	DoPERR_NOBUFFER	No transmitter buffer available
0x8003	DoPERR_OFFLINE	Connection is offline
0x8004	DoPERR_HANDLE	Invalid DoPE handle
0x8005	DoPERR_MSGSIZE	Message to long
0x8007	DoPERR_NOMEM	Not enough heap memory
0x8008	DoPERR_BADPORT	Invalid device ID
0x8009	DoPERR_BAUDRATE	Invalid baud rate
0x800A	DoPERR_OPEN	Device already in use
0x800B	DoPERR_HARDWARE	Device not present
0x800C	DoPERR_NOTOPEN	Connection not open
0x800D	DoPERR_PORTLIMIT	Unused
0x800E	DoPERR_NOTIMER	No timer for timeout check
0x800F	DoPERR_NODRIVER	No driver available
0x8010	DoPERR_NOTHREAD	Win32: Thread creation failed
0x8011	DoPERR_BADOS	Not supported operating system
0x8012	DoPERR_THUNK	Win32: Thread creation failed
-1	DoPERR_INTERNAL	Internal driver error

Versions

Version	Changes	Date	Name
2.24		2004	SAH
2.69	Update to DoPE Version 2.69	18.11.2010	SAH
2.71	Update to DoPE Version 2.71	02.12.2011	HEG
2.73	Update to DoPE Version 2.73	30.07.2012	HEG
2.74	Update to DoPE Version 2.74	28.09.2012	HEG
2.76	Update to DoPE Version 2.76	26.07.2013	HEG
2.77	Update to DoPE Version 2.77	25.10.2013	HEG
2.78	reserved	04.02.2014	HEG
2.79	Update to DoPE Version 2.79	25.03.2014	HEG
2.80	Update to DoPE Version 2.80	19.05.2014	HEG
2.82	Update to DoPE Version 2.82	28.09.2015	HEG
2.83	Update to DoPE Version 2.83	04.03.2016	HEG
2.84	Update to DoPE Version 2.84	18.08.2016	HEG
2.88	DoPEREINITIALIZEDAT_LEN corrected (must be 0x8000)	13.04.2018	HEG