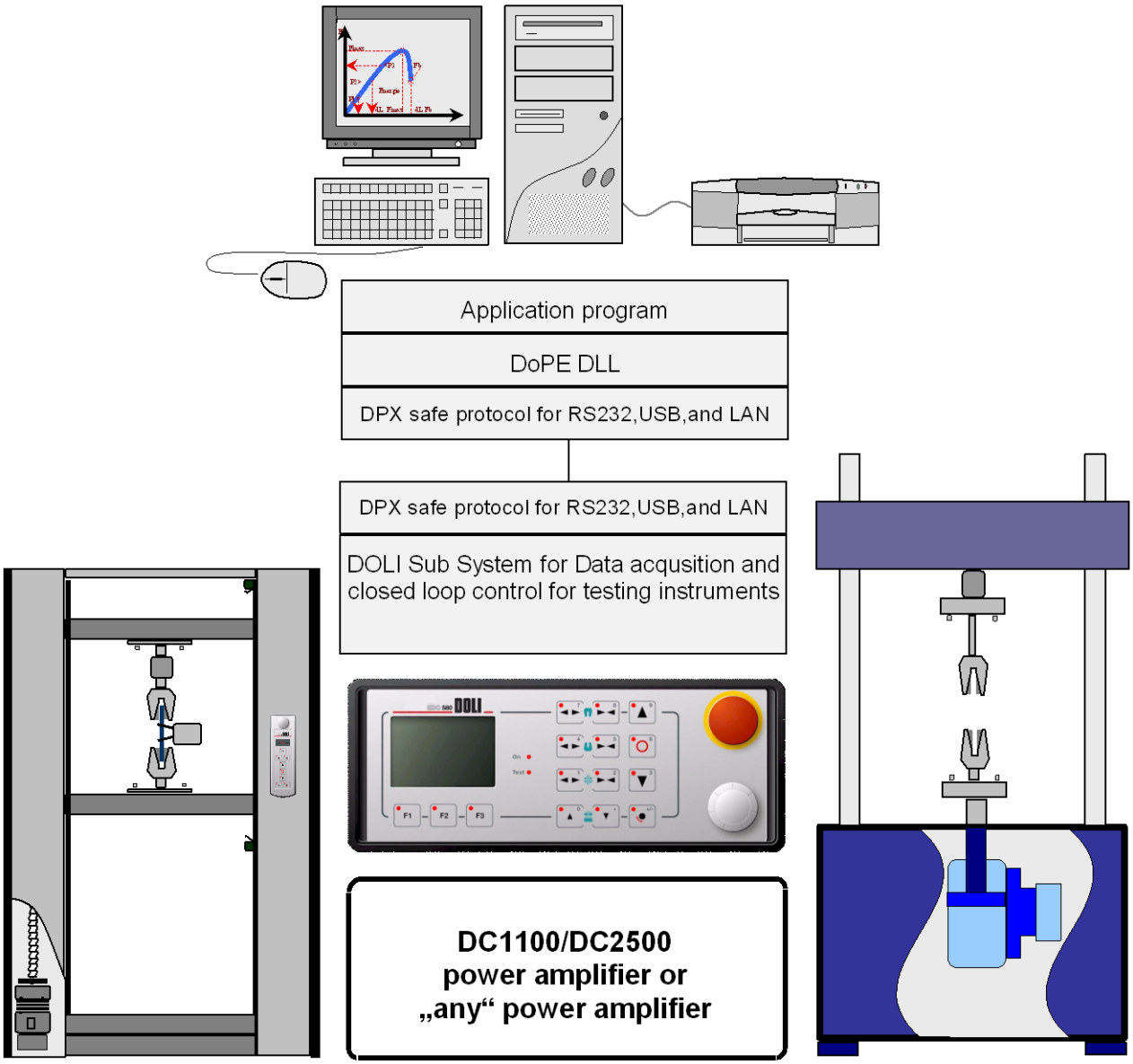
DoPE
Obsolete functions



DOLI Elektronik GmbH

Corporate Name:	DOLI Elektronik GmbH
Headquarters:	Adi-Maislinger-Straße 7, 81373 München, Deutschland ☎ (+49)-089-20 243-0 Fax (+49)-089-20 243 243 E-Mail info@doli.de
Production:	Ulmer Straße 34, 89584 Ehingen, Deutschland ☎ (+49)-07391-58039-0 Fax (+49)-07391-58039-71 E-Mail ehi@doli.de
Sales:	Mühlstr. 26, 55271 Stadecken-Elsheim, Deutschland ☎ (+49)-06130-944250 Fax (+49)-06130-944251 E-Mail sales@doli.de
Internet:	www.doli.de

Contents

1 OBSOLETE FUNCTIONS 4
1.1 Open a connection 4
1.2 Establishing a connection with call-back function 4
1.3 Establishing a connection using Windows message passing 5
1.4 Messages from DoPE to the application program 6
1.5 DoPESetNotification 7
2 OBSOLETE POSITIONING COMMANDS 9
2.1 Approach of an external destination position 9
2.1.1 DoPEPosG1(Sync) 9
2.1.2 DoPEPosG1_A(Sync) 10
2.1.3 DoPEPosD1(Sync) 11
2.1.4 DoPEPosD1_A(Sync) 12
2.2 Positioning to an external destination position 13
2.2.1 DoPEPosG2(Sync) 13
2.2.2 DoPEPosG2_A(Sync) 14
2.2.3 DoPEPosD2(Sync) 15
2.2.4 DoPEPosD2_A(Sync) 16
3 OBSOLETE MESSAGES 17
3.1 DoPESendMsg(Sync) 17
3.2 DoPEGetMsg 17
3.3 Command error messages 17
3.4 Run-time error messages 19
3.5 Movement control messages 20
3.6 Sensor message (only serial sensors) 22
3.7 DoPEGetData 23
4 OBSOLETE SETUP COMMANDS 24
4.1 DoPEOpenSetup(Sync) 24
4.2 DoPECloseSetup(Sync) 24
4.3 DoPERdSensorDef 24
4.4 DoPEWrSensorDef(Sync) 24
4.5 DoPERdCtrlSensorDef / DoPERdCtrlSensorDefHigh 25
4.6 DoPEWrCtrlSensorDef(Sync) / DoPEWrCtrlSensorDefHigh(Sync) 25
4.7 DoPERdOutChannelDef 26
4.8 DoPEWrOutChannelDef(Sync) 26
4.9 DoPERdBitInDef 27
4.10 DoPEWrBitInDef(Sync) 27
4.11 DoPERdBitOutDef 27
4.12 DoPEWrBitOutDef(Sync) 28
4.13 DoPERdMachineDef 28
4.14 DoPEWrMachineDef(Sync) 29
4.15 DoPERdLinTbl 29
4.16 DoPEWrLinTbl(Sync) 30
1 VERSIONS 31

1 Obsolete functions

All Functions described in this document are still available in DoPE, but we do not recommend these functions for new application programs. Most of this functions and methods are replaced by others as described in the DoPE documentation.

1.1 Open a connection

The application program establishes a connection by calling the **DoPEOpenLink** function. This function creates a thread with a priority depending on the priority of the application thread.

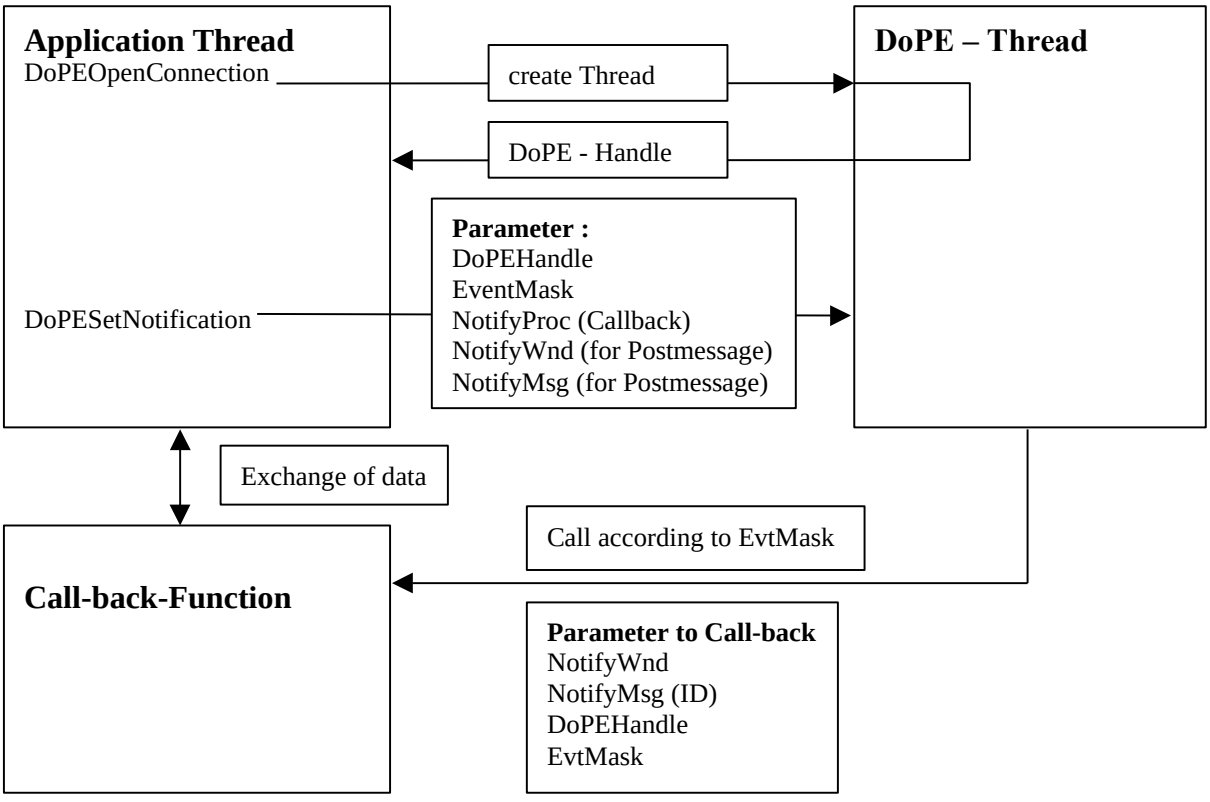
If the application thread has the priority THREAD_PRIORITY_NORMAL, DoPE thread will use the priority THREAD_PRIORITY_ABOVE_NORMAL.

If the application thread has the priority THREAD_PRIORITY_ABOVE_NORMAL, DoPE thread will use the priority THREAD_PRIORITY_TIME_CRITICAL.

The application thread should not start another thread with a higher priority than its own. Otherwise the communication with the EDC might be delayed. This will cause unpredictable problems.

By this mechanism it is safeguarded that DoPE and hence the communication with the EDC has always a higher priority than the application program.

1.2 Establishing a connection with call-back function



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

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

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

This call-back mechanism enables the application program to do some “real time” work independent from Windows message passing. The application programmer must be aware of the fact that this call-back function can interrupt the application program at any point. Care must be taken of communication between application program and call-back function. The application programmer must use a Windows mechanism like MUTEX to synchronise communication between application program and call-back.

The application programme should not call lengthy functions like graphics, file handling etc. Due to the high priority of the call-back function, only absolutely necessary tasks should be done inside the call-back function.

Tasks like:

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

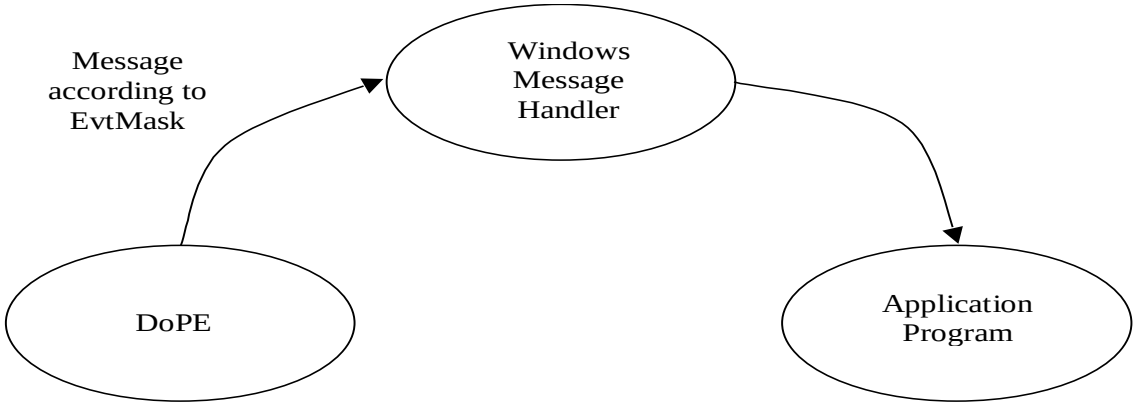
may be done inside the call-back function.

1.3 Establishing a connection using Windows message passing

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

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

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



1.4 Messages from DoPE to the application program

Besides measured values, messages are send from the EDC via DoPE to the application program.
If enabled by DoPESetNotification, a incoming message will create a DoPEEVT_RXAVAIL event which is send to the application program via call-back or PostMessage mechanism.

In both cases the real message must be read by DoPEGetMsg function.

There are three major types of messages possible:

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

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

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

1.5 DoPESetNotification

DoPE_HANDLE	DoPEHdl
unsigned	EventMask
NPROC	*NotifyProc
HWND	NotifyWnd
UINT	NotifyMsg

Select the events you are interested in.

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

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

HWND	NotifyWnd
UINT	NotifyMsg
WPARAM	DoPEHdl
LPARAM	Event

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

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

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

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

Possible events for EventMask:

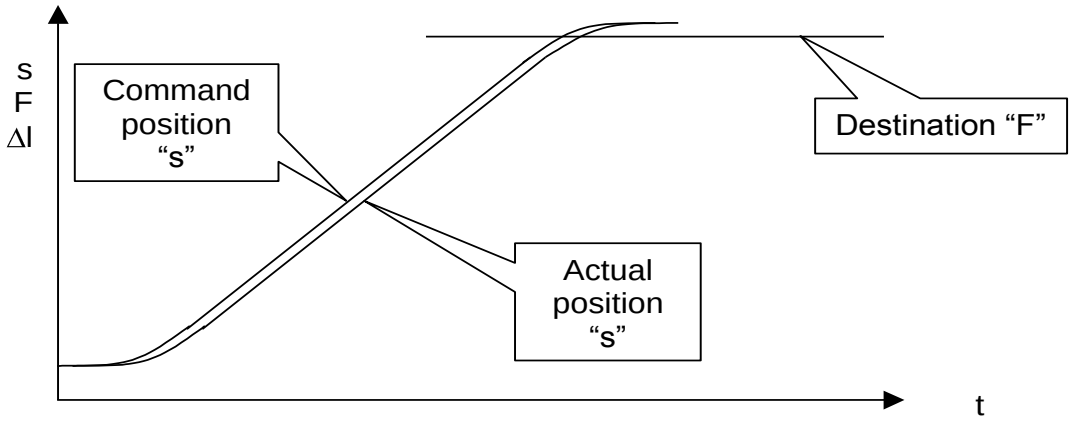
Event	Description	Action
DoPEEVT_RXAVAIL	New message received.	Read message by DoPEGetMsg
DoPEEVT_DATAVAIL	New measured data record available.	Read data by DoPEGetData
DoPEEVT_DATAOVERFLOW	Overflow of data record. The oldest record was overwritten.	Read all old data by repetitively calling DoPEGetData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

All needed events may be combined with an OR operation.

2 Obsolete positioning commands

2.1 Approach of an external destination position

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



2.1.1 DoPEPosG1(Sync)

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

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

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

2.1.2 DoPEPosG1_A(Sync)

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

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

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

2.1.3 DoPEPosD1(Sync)

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

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

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

2.1.4 DoPEPosD1_A(Sync)

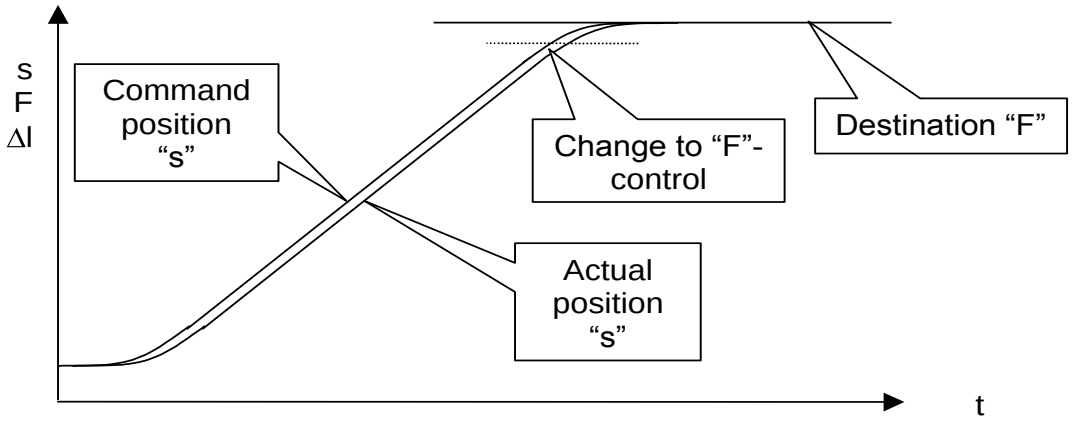
DoPE_HANDLE	DoPEHdl
unsigned short	MoveCtrl
double	Acc
double	Speed
double	DecLimit
double	Limit
double	DecDest
unsigned short	DestCtrl
double	Destination
WORD	*lpusTAN

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

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

2.2 Positioning to an external destination position

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



2.2.1 DoPEPosG2(Sync)

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

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

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

2.2.2 DoPEPosG2_A(Sync)

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

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

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

2.2.3 DoPEPosD2(Sync)

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

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

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

2.2.4 DoPEPosD2_A(Sync)

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

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

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

3 Obsolete messages

3.1 DoPESendMsg(Sync)

DoPE_HANDLE	DoPEHdl
void	*Buffer
unsigned	Length
WORD	*lpusTAN

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

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

In:	DoPE_HANDLE	DoPEHdl	DoPE link handle
	Buffer		Pointer to message to transmit
	Length		Message length in bytes
	WORD	lpusTAN	Pointer to transaction number.
Returns:			Error constant (DoPERR_xxxx)

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

3.2 DoPEGetMsg

DoPE_HANDLE	DoPEHdl
void	*Buffer
unsigned	BufSize
unsigned	*Length

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

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

3.3 Command error messages

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

Asynchronous DoPE commands do not wait. The error message must be read by DoPEGetMsg function.

Structure of command error messages:

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

List of command errors:

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

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

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

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

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

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

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

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

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

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

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

CMD_ERROR_CST (1012)
Wrong controller Structure for this command.

CMD_ERROR_NIM (1013)
Command not implemented.

CMD_ERROR_MSGNO (2001)
Unknown message ID.

CMD_ERROR_VERSION (2003)

Wrong PE interface version.

CMD_ERROR_OPEN (2004)
Set-up not opened.

CMD_ERROR_MEMORY (2005)
Not enough memory.

3.4 Run-time error messages

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

Structure of run-time error message:

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

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

In later DoPE versions there will be more detailed error messages.

List of run-time errors:

RTE_ERROR_S (101)
X-Head position controller deviation error.
The difference between command and measured value is too big. Increase “deviation” parameter in “DoPECtrlSenDef” set-up structure or optimise control loop.

RTE_ERROR_F (102)
Load controller deviation error.

RTE_ERROR_E (103)
Extension controller deviation error.

RTE_EMOVE_END (104)
Emergency movement is finished but run-time error is still active.

RTE_DRIVE_OFF (201)
Drive was switched off.

RTE_E_MOVE_RQ (202)
Drive was switched off. A emergency drive is possible to clear this situation.

RTE_UPPER_LIMIT_SWITCH (203)
Drive was switched off due to an active upper limit switch.

RTE_LOWER_LIMIT_SWITCH (204)
Drive was switched off due to an active lower limit switch.

RTE_STOP (205)
Drive not ready.

RTE_DF_KEY (206)
"Drive enable" output signal was withdrawn due to an inactive input signal "Drive enable".

RTE_SHALT (207)
Machine was halted in position control due to a active input signal. This normally happens after "STOP" key at EDC front panel was pressed.

RTE_UPPER_LIMIT (301)
Upper range limit exceeded.

RTE_LOWER_LIMIT (302)
Lower range limit exceeded.

RTE_ERROR_UNHANDLED (999)
Unknown runtime error.

3.5 Movement control messages

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

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

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

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

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

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

List of movement control messages:

CMSG_POS (1) use DoPEMCM

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

CMSG_UPPER_SFT (2) use DoPESftM

CMSG_LOWER_SFT (3) use DoPESftM

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

CMSG_POS_ERR (4) use DoPEMCM

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

CMSG_TPOS (6) use DoPEMCM

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

CMSG_TPOS_ERR (7) use DoPEMCM

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

CMSG_LPOS (8) use DoPEMCM

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

CMSG_LPOS_ERR (9) use DoPEMCM

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

CMSG_OFFSET (49) use DoPEOffsCM

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

CMSG_CHECK (51) use DoPECheckM

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

CMSG_CHECK_ERR (52) use DoPECheckM

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

CMSG_SHIELD (53) use DoPECheckM

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

CMSG_SHIELD_ERR (54) use DoPECheckM

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

CMSG_REFSIGNAL (55) use DoPERefSignalM

Reference Signal of a incremental sensor occurred.

3.6 Sensor message (only serial sensors)

A serial “sensor” may be connected via RS232 (X62A – X62D) to EDC60/120, EDC220/222 and EDC580.

If it is specified with “NoProtocol”, it’s data are not interpreted by EDC or Dope, but send to the application program. Following message will be given to the application program:

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

3.7 DoPEGetData

Get 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 **Fehler! Textmarke nicht definiert.Fehler! Textmarke nicht definiert.**.	
Function declaration	**Description**
extern unsigned DLLAPI DoPEInitializeResetXHead (DoPE_HANDLE DoPEHdl, DoPEData *Sample)	Function returns Error constant (DoPERR_xxxx) DoPE link handle Pointer to storage for data record.

4 Obsolete setup commands

4.1 DoPEOpenSetup(Sync)

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

In:	DoPE_HANDLE	DoPEHdl	DoPE link handle
	unsigned short	SetupNo	Set-up Number
	WORD	*lpusTAN	Pointer to transaction number.
Out:	*lpusTAN		Pointer to TAN used from this command
Returns:			Error constant (DoPERR_xxxx)

4.2 DoPECloseSetup(Sync)

Closes previously opened set-up.

In:	DoPE_HANDLE	DoPEHdl	DoPE link handle
	WORD	*lpusTAN	Pointer to transaction number.
Out:	*lpusTAN		Pointer to TAN used from this command
Returns:			Error constant (DoPERR_xxxx)

4.3 DoPERdSensorDef

Read sensor definitions of opened set-up.

In:	DoPE_HANDLE	DoPEHdl	DoPE link handle
	unsigned short	SensorNo	Sensor Number
	DoPESenDef	*SensorDef	Pointer for DoPESenDef structure
Out:	*SetupNo		Setup No.
Returns:			Error constant (DoPERR_xxxx)

4.4 DoPEWrSensorDef(Sync)

Write Sensor definitions to opened set-up.

In:	DoPE_HANDLE	DoPEHdl	DoPE link handle
	unsigned short	SensorNo	Sensor Number
	DoPESenDef	*SensorDef	Pointer for Sensor Definition structure
	WORD	*lpusTAN	Pointer to transaction number (not for Sync. version).
Out:	*lpusTAN		Pointer to TAN used from this command
Returns:		Error constant (DoPERR_xxxx)	

DoPESenDef structure:

```
typedef struct                          /* Sensor definition data             */
{                                       /* ---------------------------------- */
  WORD   Connector;                     /* Connector number of sensor     [No]*/
  WORD   Sign;                          /* Invert sign of sensor         [1/0]*/
  WORD   CtrlChannel;                   /* Activate control channel      [1/0]*/
  WORD   LimitCtrl;                     /* Stop if limit exceeded        [1/0]*/
  WORD   ConnectedCtrl;                 /* Stop if disconnected          [1/0]*/
  WORD   UseEeprom;                     /* Use sensor EEPROM data        [1/0]*/
                                        /* only for analogue sensors:         */
```

```
  double Integr;                    /* Time of integration             [s]*/
                                    /* only for sensors without EEPROM:   */
  WORD   Init;                      /* Sensor init                   [No]*/
  double NominalValue;              /* Nominal value of sensor      [Unit]*/
  WORD   Unit;                      /* Unit of sensor UNIT_xxx         [No]*/
  double Offset;                    /* Offset of sensor             [Unit]*/
  double UpperLimit;                /* Upper range limit of sensor     [%]*/
  double LowerLimit;                /* Lower range limit of sensor     [%]*/
                                    /* only for incremental sensors:      */
  double Scale;                     /* Scale of sensor    [inc/revolution]*/
                                    /*               or [Unit/revolution]*/
  double Correction;                /* Correction value of sensor     [No]*/
} DoPESenDef;
```

4.5 DoPERdCtrlSensorDef / DoPERdCtrlSensorDefHigh

Read definitions of control sensor for low and high pressure.

In:	DoPE_HANDLE	DoPEHdl	DoPE link handle
	unsigned short	SensorNo	Sensor Number
	DoPECtrlSenDef	*CtrlSensorDef	Pointer for DoPECtrlSenDef structure
Out:	* CtrlSensorDef		DoPECtrlSenDef structure
Returns:			Error constant (DoPERR_xxxx)

4.6 DoPEWrCtrlSensorDef(Sync) / DoPEWrCtrlSensorDefHigh(Sync)

Write definitions of control sensor for low and high pressure.

In:	DoPE_HANDLE	DoPEHdl	DoPE link handle
	unsigned short	SensorNo	Sensor Number
	DoPECtrlSenDef	*CtrlSensorDef	Pointer for Control-Sensor Definition structure
	WORD	*lpusTAN	Pointer to transaction number (not for Sync. version).
Out:	*lpusTAN		Pointer to TAN used from this command
Returns:			Error constant (DoPERR_xxxx)

DoPECtrlSenDef structure:

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

4.7 DoPERdOutChannelDef

Read analogue output channel definitions of opened set-up.

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

4.8 DoPEWrOutChannelDef(Sync)

Write analogue output channel definitions to opened set-up.

In:	DoPE_HANDLE	DoPEHdl	DoPE link handle
	unsigned short	OutChNo	Analogue Output channel no
	DoPEOutChaDef	* CtrlSensorDef	Pointer for DoPEOutChaDef structure
	WORD	*lpusTAN	Pointer to transaction number (not for Sync. version).
Out:	*lpusTAN		Pointer to TAN used from this command
Returns:			Error constant (DoPERR_xxxx)

DoPEOutChaDef structure:

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

4.9 DoPERdBitInDef

Read Bit input definitions of opened set-up.

In:	DoPE_HANDLE	DoPEHdl	DoPE link handle
	unsigned short	BitInNo	Output channel no.
	DoPEBitInDef	*BitInDef	Pointer for DoPEBitInDef structure
Out:	*BitInDef		DoPEBitInDef structure
Returns:			Error constant (DoPERR_xxxx)

4.10 DoPEWrBitInDef(Sync)

Write Bit input definitions to opened set-up.

In:	DoPE_HANDLE	DoPEHdl	DoPE link handle
	unsigned short	BitInNo	Bit Input channel no
	DoPEBitInDef	*BitInDef	Pointer for DoPEBitInDef structure
	WORD	lpusTAN	Pointer to transaction number (not for Sync. version).
Out:	*lpusTAN		Pointer to TAN used from this command
Returns:			Error constant (DoPERR_xxxx)

DoPEBitInDef structure:

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

4.11 DoPERdBitOutDef

Read Bit output definitions of opened set-up.

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

4.12 DoPEWrBitOutDef(Sync)

Write Bit output definitions to opened set-up.

In:	DoPE_HANDLE	DoPEHdl	DoPE link handle
	unsigned short	BitOutNo	Bit Output channel no
	DoPEBitOutDef	*DoPEBitOutDef	Pointer for DoPEBitOutDef structure
	WORD	*lpusTAN	Pointer to transaction number (not for Sync. version).
Out:	*lpusTAN		Pointer to TAN used from this command
Returns:			Error constant (DoPERR_xxxx)

DoPEBitOutDef structure:

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

4.13 DoPERdMachineDef

Read definitions of active machine of opened set-up.

In:	DoPE_HANDLE	DoPEHdl	DoPE link handle
	DoPEMachineDef	*MachineDef	Pointer for DoPEMachineDef structure
Out:	* MachineDef		DoPEMachineDef structure
Returns:			Error constant (DoPERR_xxxx)

4.14 DoPEWrMachineDef(Sync)

Write definitions of active machine to opened set-up.

In:	DoPE_HANDLE	DoPEHdl	DoPE link handle
	DoPEMachineDef	*MachineDef	Pointer for DoPEMachineDef structure
	WORD	*lpusTAN	Pointer to transaction number (not for Sync. version).
Out:	*lpusTAN		Pointer to TAN used from this command
Returns:			Error constant (DoPERR_xxxx)

DoPEMachineDef structure:

```
typedef struct                          /* Definition of machine data          */
{                                       /* ---------------------------------- */
  double SpeedCtrlTime;                 /* Speed controller cycle time     [s]*/
                                        /*                    [0.2ms .. 2.5ms]*/
  double PosCtrlTime;                   /* Position controller cycle time  [s]*/
  WORD   CtrlStructure;                 /* Closed loop control structure  [No]*/
  double DataTransmissionRate;          /* Data transmission rate          [s]*/
  WORD   Mode;                          /* Data acquisition or control   [1/0]*/
  WORD   UnUsed3;                       /* Unused                              */
  WORD   XheadDir;                      /* Machine moves up/down         [1/0]*/
                                        /* with positive output signal         */
  double UnUsed4;                       /* Unused                              */
  double EncXheadRatio;                 /* Ratio encoder-Xhead     [Rev/Unit] */
  double BrakeOpen;                     /* Delay time to open brake after  [s]*/
                                        /* closed loop control is active       */
  double BrakeClose;                    /* Delay time to close brake before[s]*/
                                        /* closed loop control is deactivated */
  double PistonArea;                    /* Area of piston for hydraulic   [m²]*/
  double LoadMax;                       /* Max. load capacity of machine   [N]*/
  double Load100;                       /* Nominal load of machine         [N]*/
  double Stiffness;                     /* Over all stiffness of machine [N/m]*/
  short  UnUsed1;                       /* Unused                              */
  short  UnUsed2;                       /* Unused                              */
  WORD   GripConnector;                 /* Grip: connector number         [No]*/
  WORD   GripChannel;                   /* Grip: channel                  [No]*/
  WORD   GripActive;                    /* Grip: active Low/High          [No]*/
  WORD   ShieldConnector;               /* Shield: connector number       [No]*/
  WORD   ShieldType;                    /* Shield: type simple/locked     [No]*/
  double ShieldTimeout;                 /* Shield: timeout                 [s]*/
  WORD   CtrlOnMode;                    /* CTRL ON mode: 0=NoCtrl, 1=Ctrl     */
  double FixValue;                      /* Fixing  output value           [%]*/
  double InitValue;                     /* Initial output value           [%]*/
  double ReturnValue;                   /* Return  output value           [%]*/
} DoPEMachineDef;
```

4.15 DoPERdLinTbl

Read linearisation table of opened set-up.

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

4.16 DoPEWrLinTbl(Sync)

Write linearisation table to opened set-up.

In:	DoPE_HANDLE	DoPEHdl	DoPE link handle
	DoPELinTblFalse	*LinTblFalse	Pointer to measured values structure
	DoPELinTblTrue	*LinTblTrue	Pointer to reference values structure
	WORD	*lpusTANFirst	Pointer to first Transaction-Number (not for Sync. version)
	WORD	*lpusTANLast	Pointer to last Transaction-Number (not for Sync. version)
Out:	*lpusTANFirst		Pointer to first TAN used from this command
	*lpusTANLast		Pointer to last TAN used from this command
Returns:			Error constant (DoPERR_xxxx)

DoPELinTblFalse and DoPELinTblTrue structure:

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

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

1 Versions

Version	Changes	Date	Name
1	First edition		