ProMoS Development

Version 1.5

Table of Contents

Foreword

Printed: Dezember 2008 in Belp, Switzerland.

Publisher
MST Systemtechnik AG

Managing Editor
Christoph Müller

Technical Editors
Zdenek Sulc
Christoph Müller

Chapter 1

1 Introduction

Introduction

Top Next

ViSi.Plus works only on Windows NT, Win2K, WinXP and Vista. Windows 95/98 is not supportet.

The idea behind ViSi.Plus is, that a data handler manages all information of the system. We use ViSi.Plus to visualize processes in the industry. But the are also a lot of other possibilities.

ViSi.Plus is a collection of diverse programs. Every program is independent. Threre are two proprietary data bases.

- **DMS** - **D**ata **M**anagement **S**ystem 
  Fast data base without disk-access. The data are stored only in RAM memory. that's why is the system is very fast and usefull in process control systems. When the DMS starts, the data as well as all configuration settings are loaded in memory. You can save the data on disk, if you need it.

- **PDBS** - **P**rocess **D**ata**B**ase **S**ystem 
  Disc oriented data base for historical data. Alarms and protocols will be as well administrated by the PDBS.

There are connections to both data-management systems, so you can write your own PLC-communication-program, or a program that gets some actual data from the DMS or historical ones from the PDBS.

For DMS there are two different kinds of functions:

- Access with callback-function (ConnectDMS, DisconnectDMS, RegisterDMS, UnregisterDMS, etc.)
- Direct access (DMS_Connect, DMS_Close, DMS_ReadBIT, DMS_WriteBIT, etc.)

You can implement your programs in different programing languages. We have done all our programs with MS-Visual C++, but you can use other languages as Delphi or Visual Basic.

Chapter II

2 DMS Functions

Use callback access to DMS data if you want to be notified and updated automatically (by callback event) when registered data variables have changed. Use following functions:

- ConnectDMS(...) // data change callback notification
- ConnectDMSidle(...) // as ConnectDMS() plus additional notification when data not yet ready to deliver
- DisconnectDMS(...) // close connection (callback)

- RegisterDMS(...) // register DMS data point (variable), about which value changes you want to be notified
- UnregisterDMS(...) // terminate (unsubscribe) notification

- SendPipe(...) // write (general, raw) data to DMS
- SendPrgName(...) // label your callback connection (useful for debug purposes, identification of broken connections, etc.)

- SendBitDMS(...) // write data to BIT-type DMS variable and return immediately (non-blocking mode, does not wait for result)
- SendBysDMS(...) // .. BYS = byte signed
- SendByuDMS(...) // .. BYU = byte unsigned
- SendWosDMS(...) // .. WOS = word signed
- SendWouDMS(...) // .. WOU = word unsigned
- SendDwsDMS(...) // .. DWS = Double word signed
- SendDwuDMS(...) // .. DWU = Double word unsigned
- SendFltDMS(...) // .. FLT = float
- SendStrDMS(...) // .. STR = string (max. 80 characters long)

To access DMS data directly through a connection channel use following functions:

- DMS_Connect(...) // open a connection channel to DMS
- DMS_Close()

- DMS_Send(...) // write (general, raw) data to DMS
- DMS_SendCode(...) // send message (commend, control code) to DMS

- DMS_SendPrgName(...) // label your connection channel (useful for debug purposes, identification of broken connections, etc.)
- DMS_SaveTree(...) // save appropriate DMS sub-tree as a file

- DMS_Create(...) // create a DMS data point (this data point will be deleted when connection closes)
- DMS_CreateRemanent(...) // create a DMS data point (this data point remains when connection closes)

- DMS_Delete(...) // delete a DMS data point
- DMS_Rename(...) // rename a DMS data point

- DMS_ReadType(...) // read type of a DMS data point (bit, word, byte, string, etc.)
- DMS_GetSPS(...)

- DMS_WriteBIT(...) // write data to BIT-type DMS variable and return when data written (blocking mode, wait for result)
- DMS_WriteBYS(...) // .. BYS = byte signed
- DMS_WriteBYU(...) // .. BYU = byte unsigned
- DMS_WriteWOS(...) // .. WOS = word signed

- DMS_WriteWOU(...) // .. WOU = word unsigned
- DMS_WriteDWS(...) // .. DWS = Double word signed
- DMS_WriteDWU(...) // .. DWU = Double word unsigned
- DMS_WriteFLT(...) // .. FLT = float
- DMS_WriteSTR(...) // .. STR = string (max. 80 characters long)

- DMS_ReadBIT(...) // read data from BIT-type DMS variable
- DMS_ReadBYS(...) // .. BYS = byte signed
- DMS_ReadBYU(...) // .. BYU = byte unsigned
- DMS_ReadWOS(...) // .. WOS = word signed
- DMS_ReadWOU(...) // .. WOU = word unsigned
- DMS_ReadDWS(...) // .. DWS = Double word signed
- DMS_ReadDWU(...) // .. DWU = Double word unsigned
- DMS_ReadFLT(...) // .. FLT = float
- DMS_ReadSTR(...) // .. STR = string (max. 80 characters long)

- DMS_GetRights(...) // read rights of a DMS variable (data point access signs = READ_ONLY, READ_WRITE, REMANENT, etc.)
- DMS_SetRights(...) // set rights of a DMS variable

- DMS_FindNames(...) //
- DMS_Find(...)
- DMS_FindNext(... //
- DMS_FindVal(...)
- DMS_FindNextVal(...)
- DMS_FindInt(...)
- DMS_FindFlt(...)
- DMS_FindStr(...)

- DMS_FindMessageFilter(...) //
- DMS_FindMessage(...) //
- DMS_FindNextMessage(...) //
- DMS_ResetMessage(...) //
- DMS_FindValMessage(...) //
- DMS_ResetValMessage(...) //

- DMS_GetNames(...) //
- DMS_GetNextName(...) //

- DMS_Login(...) //
- DMS_ReadUser(...)

- DMS_Error(...)

2.1 Choosing a Network Protocol

There are two possibilities of communication connectivity:

- Named Pipes:
Protocol developed for local area networks.

All ViSi.Plus access functions were originally developed for named pipes only.

- TCP/IP Sockets:
Common protocol widely used over the Internet. It communicates across interconnected networks of computers that have diverse hardware architectures and various operating systems.

ViSi.Plus access functions which are dedicated for TCP/IP sockets have the "...Ex" suffix added to their function names.

2.2 Main Functions

Main functions Top Previous Next

2.2.1 ConnectDMS

ConnectDMS Top Previous Next

ConnectDMS	Create a connection to the DMS.	
Prototype	`int _stdcall ConnectDMS(TCHAR *lpszPipename, client *cl, int (*fun_ptr)(comm *msg));`	
Returnvalue	int	LastError. 0, if everything is OK. See Win32 SDK-documentation (winerror.h).
Parameter	TCHAR *lpszPipename	Pipe-Name, include PC-name (. for local machines) Sample: \\.\PIPE\PROMOS-DMS (for a local PC) \\PC3\PIPE\PROMOS-DMS (for a remote PC, called 'PC3')
	client *cl	Client-Structure (this structure is only used internal). The function SendDMS() uses this structure.
	int (*fun_ptr)(comm *msg)	Function-pointer to a callback function This is the only function with _cdecl (and not _stdcall). Be carefull with other languages (e.g. Delphi).
Remarks	The count of connection is not limited. Each program can make more than one connection. If you connect more than one connection, please wait some milliseconds to allow the DMS to create the thread.	

Samples Visual C++

```
#include   "handle.h"
client   dms;
char    lpszPipename[41];
wsprintf(lpszPipename,"\\\\%s\\PIPE\\%s",pcname,"PROMOS-DMS");
if((ConnectDMS(lpszPipename,&dms,MessageHandler)) != 0){
   AfxMessageBox("Can't find DMS");
   return;
}
```

Delphi

```
cl                  : TClient;
Pipename          : Tpipename;
err                 : Integer;

Pipename   := '\\.\pipe\PROMOS-DMS';
err   := ConnectDMS(Pipename, cl, HandleMessage);
```

See also SendDMS, RegisterDMS, SendDMS

2.2.2 DisconnectDMS

DisconnectDMS

Top Previous Next

Disconnect DMS	Disconnect from DMS.	
Prototype	`int _stdcall DisonnectDMS(client *cl);`	
Returnvalue	int	LastError. 0, if everything is OK. See Win32 SDK-documentation (winerror.h).
Parameter	client *cl	Client-Structure (this structure is only used internal).The function SendDMS() uses this structure.You get the structure by calling ConnectDMS();
Remarks	Before closing a connection you have to unregister every registered datapoint. If the application ends without disconnecting the DMS close the connection (e.g. in case of an application crash), but it's a better programing style to disconnect within the program	

Samples Visual C++

```
#include "handle.h"
client   dms;
char lpszPipename[41];
wsprintf(lpszPipename,"\\\\%s\\pipe\\%s",pcname,"PROMOS-DMS");
if((ConnectDMS(lpszPipename,&dms,MessageHandler)) != 0){
   AfxMessageBox("Can't find DMS");
   return;
}
...
DisconnectDMS(&dms);
```

Delphi

```
err := DisconnectDMS(cl);
```

See also SendDMS, RegisterDMS, SendDMS

2.2.3 SendDMS

SendDMS

Top Previous Next

SendDMS	Send data to the DMS	
Prototype	`int _stdcall SendDMS(comm *msg,client *cl);`	
Returnvalue	int	LastError. 0, if everything is OK. See Win32 SDK-documentation (winerror.h).
Parameter	Message* msg	Data to send (see Message structure)
	client *cl	Client-Structure (this structure is only used internal)
Remarks	If you use this function to set or read values, the answers are responded thrue the callback-function. Use the DMS_*-functions to read and write values, it‘s much simpler. The ‘Client' should be a global variable (we recognized some problems using local variables).	

Samples Visual C++

```
#include "handle.h"
client   dms;   // Global variable
Message   msg;

msg.message_id   = ID_REGISTER;
strcpy(msg.point_name, Name);
msg.obj_id   = nr;
msg.status   = ID_REGISTER;
SendDMS(&msg,&dms);
```

Delphi

```
Themsg.message_id   := ID_REGISTER;
Themsg.point_name    := 'Gruppe1:Motor1:Temperatur';
Themsg.status   := ID_REGISTER;
Themsg.obj_id    := 4800;
err   := SendPipe(Themsg, cl);
```

See also ConnectDMS

2.2.4 Callback function

Callback function

Top Previous Next

Callback function	The callback function is called, when the program has registered a datapoint and the value of the point has changed (e.g. changes from the PLC-driver). The function is called automaticaly, you just have to put a function-pointer in the ConnectDMS-command. In the parameter Message you get the new values including the obj_id (see RegisterDMS-command).	
Prototype	`cdecl int MessageHandler(Message* msg);`	
Returnvalue	int	Not used.
Parameter	Message* msg	Data from the DMS.
Remarks	This is the only function with _cdecl. Be carefull to declare it in other languages the same way. This function is running in a thread and not in the main function. Be careful to set values in the GUI, because the GUI runs in another thread !	

Samples Visual C++

```
int MessageHandler(Message *msg)
{
   if(msg->message_id == ID_INFORM) {

// Do something with the data
switch(List[msg->obj_id].type) {
   case COLOR_CHANGE_VG :
   if(List[msg->obj_id].mObj->InitColorVG == -1) return 0;
   if(msg->value.val_BIT)
   List[msg->obj_id].mObj->m_logpen.lopnColor =
   List[msg->obj_id].mObj->InitVG->Color1;
   else
   List[msg->obj_id].mObj->m_logpen.lopnColor =
   List[msg->obj_id].mObj->InitVG->Color0;
   break;
   case COLOR_CHANGE_BG :
   ...
   break;

   case COLOR_CHANGE_TXT :
   ...
   break;
   }

   // Post message in mainthread (just in this sample !)
   PostMessage(List[msg->obj_id].hWnd,WM_USER, msg->obj_id,0);
   }
   return 0;
}
```

Delphi

```
function HandleMessage(var msg: TDMSMessage): Integer; cdecl;
   var Value   : TValueType;
   ausgabe   : String;
begin

   Form1.Edit2.Text := IntToStr(msg.obj_id)+' :' + msg.point_name;

   case msg.obj_id of
   4800 :   begin
   Form1.Edit3.Text := FloatToStr(msg.value.val_FLT);
   end;
   4801 :   begin
   Form1.Edit5.Text := FloatToStr(msg.value.val_FLT);
   end;
end;

   Form1.Edit5.Text := FloatToStr(msg.value.val_FLT);
end;
```

See also RegisterDMS, UnregisterDMS, SendDMS

2.2.5 RegisterDMS

RegisterDMS

Top Previous Next

RegisterDMS	Register a datapoint. Every change in the datapoint-value will involve the callback function.	
Prototype	`int RegisterDMS(client* dms, char* name, int obj_id);`	
Returnvalue	int	LastError. 0, if everything is OK. See Win32 SDK-documentation (winerror.h).
Parameter	client *cl	Client-Structure (this structure is only used internal)
	char* name	Datapoint-name
	int obj_id	ID, to check, which value has been changed. This ID is a value, that you can use for whatever you need (e.g. index in an array).
Remarks	Use the obj_id to manage the changed data in the callback-function. It's much faster than to check the DMS-name everytime the callback-function is involved You have to unregister every registered datapoint before quitting the program..	

Samples Visual C++

```
#include "handle.h"
client   dms;
RegisterDMS(&dms, "Gruppe1:Motor1:Zustand", 1000);
```

Delphi

```
RegisterDMS(cl, 'Gruppe1:Motor1:Temperatur', 4800);
RegisterDMS(cl, 'Gruppe1:Motor2:Temperatur', 4801);
```

See also ConnectDMS, UnregisterDMS

2.2.6 UnregisterDMS

UnregisterDMS

Top Previous Next

UnregisterDMS Unregister a datapoint.

Prototype `int UnregisterDMS(client* dms, char* name, int obj_id);`

Returnvalue	int	LastError. 0, if everything is OK. See Win32 SDK-documentation (winerror.h).
Parameter	client *cl	Client-Structure (this structure is only used internal)
	char* name	Datapoint-name
	int obj_id	ID (the same as used in RegisterDMS)

Remarks You have to unregister every registered datapoint before quitting the program.

Samples Visual C++

```
#include "handle.h"
BOOL CMainFrame::DestroyWindow()
{
   // Alle angemeldeten Datenpunkte abmelden
   for(int i=0; i < Alarm.GetSize(); i++) {
   UnregisterDMS(&dms, Alarm[i].DMSName.GetBuffer(MAX_NAME),
   Alarm[i].Index);
   }
   DisconnectDMS(&dms);

   return CFrameWnd::DestroyWindow();
}
```

Delphi

```
procedure TForm1.Button3Click(Sender: TObject);
begin
UnregisterDMS(cl, 'Gruppe1:Motor1:Temperatur', 4800);
UnregisterDMS(cl, 'Gruppe1:Motor2:Temperatur', 4801);
DisconnectDMS(cl);
end;
```

See also ConnectDMS, RegisterDMS

2.3 Access Functions

Access functions

Top Previous Next

DMS_Connect
DMS_Close
DMS_Error
DMS_Create
DMS_CreatePoint
DMS_Remove
DMS_Delete
DMS_SendCode
DMS_ReadType
DMS_ReadBIT
DMS_ReadBYS
DMS_ReadBYU
DMS_ReadWOS
DMS_ReadWOU
DMS_ReadDWS
DMS_ReadDWU
DMS_ReadFLT
DMS_ReadSTR
DMS_WriteBIT
DMS_WriteBYS
DMS_WriteBYU
DMS_WriteWOS
DMS_WriteWOU
DMS_WriteDWS
DMS_WriteDWU
DMS_WriteFLT
DMS_WriteSTR

DMS_GetRights
DMS_SetRights
DMS_FindMessage
DMS_FindNextMessage
DMS_GetNames
DMS_GetNextName

2.3.1 DMSConnect

DMSConnect

Top Previous Next

Make a connection to a DMS. This connection use the same thread as the main program. No callback-functions will be involved.

Prototype

```
int _stdcall DMS_Connect(TCHAR *lpszPipename, HANDLE& pipe);
```

Returnvalue

int		Zero, if sucessfull, see Win32 SDK-documentation.

Parameter

TCHAR*	pipename	Pipename including PC-name (on networks)
HANDLE& pipe		Handle

Remarks

You have to initialise the handle to NULL, before calling the function. If the handle has a value, the function does not create a connection to the DMS.

Samples Visual C++

```
if(!PcName)
   sprintf(lpszPipename,"\\\\.\\pipe\\PROMOS-DMS");
else
   sprintf(lpszPipename,"\\\\%s\\pipe\\PROMOS-DMS", PcName);

Connect = FALSE;
pipe = NULL;         // Important !

if(DMS_Connect(lpszPipename, pipe)) {
   return;
}
Connect = TRUE;
```

Delphi

```
procedure TForm1.FormCreate(Sender: TObject);
var retval   : LongInt;
   t   : TText;   //   TText = array [0 ..80] of char;
begin

   handle    := 0;      // Important !
   t   := '\\.\pipe\PROMOS-DMS';
   retval   := DMS_Connect(t, handle);

end;
```

See also DMS_Close

2.3.2 DMSClose

DMSClose

Top Previous Next

Close a connection to the DMS.

Prototype	`int _stdcall DMS_Close(HANDLE pipe);`	
Returnvalue	int	Zero, if sucessfull, see Win32 SDK-documentation.
Parameter	HANDLE pipe	Handle (the same as from DMS_Connect
Remarks	Use DMS_Connect() to use this function. You cannot close a connection opened by ConnectDMS().	

Samples Visual C++

```
if(pipe)
   DMS_Close(pipe);
```

Delphi

```
if(handle <> 0) then
   DMS_Close(handle);
```

See also DMS_Connect

2.3.3 DMS_Error

DMS_Error

Top Previous Next

Gets the last error.

Prototype

```
int _stdcall DMS_Error(void);
```

Returnvalue

int	LastError. 0, if everything is OK.
INVALID_MSG	The message was invalid (e.g. wrong message_id)
POINT_NOT_EXIST	The datapoint is not in the DMS.
NO_MEMORY	The DMS can't allocate memory.
	See Win32 SDK-documentation.

Parameter None

Remarks Some functions sets an error-value. It's not implemented in every function yet.

Samples Visual C++

```
int Err = DMS_Error();
```

Delphi

```
var lasterr   : LongInt;
begin
   lasterr   := DMS_Error;
end;
```

See also all DMS_... -functions

2.3.4 DMS_Create

DMS_Create

Top Previous Next

Create a new datapoint in the DMS.

Prototype `int _stdcall DMS_Create(HANDLE pipe, TCHAR *name, char type);`

Returnvalue	int	Zero, if sucessfull, see Win32 SDK-documentation.
Parameter	HANDLE pipe	Handle from DMS_Connect().
	TCHAR* Name	DMS-pointname
	char type	Type of datapoint (see types)

Remarks Use DMS_Connect() to use this function.

Samples Visual C++

```
int ret = DMS_Create(pipe, "MA:MT:500:Value", ID_FLT );
```

Delphi

```
lasterr := DMS_Create(handle,'Motor:37:On', char(ID_BIT));
if(lasterr <> 0) then
   MessageDlg('Fehler DMS_Create', mtError, [mbOK], 0);
```

See also DMS_CreatePoint

2.3.5 DMS_Create Point

DMS_Create Point

Top Previous Next

Create a new datapoint with rights in the DMS.

Prototype

```
int _stdcall DMS_Create(HANDLE pipe, TCHAR *name, char type,
    char rights);
```

Returnvalue

int	Zero, if sucessfull, see Win32 SDK-documentation.

Parameter

HANDLE pipe	Handle from DMS_Connect().
TCHAR* Name	DMS-pointname
char type	Type of datapoint (see types)
rights	Access rights READ_ONLY Other processes can't write values. READ_WRITE Every process can write values. REMANENT Only remanent values are stored on disk. The REMANENT-Flag should be ORed with the other flags.

Remarks

Use DMS_Connect() to use this function.

The rights are not fully implemented yet, but will be in a further version of DMS. The REMANTENT-flag is used to save DMS-values on disk.

Samples

Visual C++

```
int ret = DMS_CreatePoint(pipe, "MA:MT:500:Value", ID_FLT,
   READ_WRITE | REMANENT);
```

Delphi

```
lasterr := DMS_CreatePoint(handle,'Motor:38:On', char(ID_BIT),
   char(READ_WRITE));
if(lasterr <> 0) then
   MessageDlg('Fehler DMS_CreatePoint', mtError, [mbOK], 0);
```

See also

DMS_Create

2.3.6 DMS_Remove

DMS_Remove

Top Previous Next

Remove a datapoint from the DMS-informing list. The point itself is not removed.

Prototype `int _stdcall DMS_Remove(HANDLE pipe, TCHAR *name);`

Returnvalue	int	Zero, if sucessfull, see Win32 SDK-documentation.
Parameter	HANDLE pipe	Handle from DMS_Connect().
	TCHAR* Name	DMS-pointname

Remarks Use DMS_Connect() to use this function.

Samples Visual C++

```
int ret = DMS_Remove(pipe, "MA:MT:500:Value");
```

Delphi

```
retval := DMS_Remove(handle, 'MA:MT:500:Value');
```

See also DMS_Create, DMS_CreatePoint, DMS_Delete

2.3.7 DMS_Delete

DMS_Delete

Top Previous Next

Remove a datapoint from the DMS.

Prototype `int _stdcall DMS_Delete(HANDLE pipe, TCHAR *name);`

Returnvalue	int	Zero, if sucessfull, see Win32 SDK-documentation.
Parameter	HANDLE pipe	Handle from DMS_Connect().
	TCHAR* Name	DMS-pointname

Remarks Use DMS_Connect() to use this function.

You can only remove datapoints, if no other process is registered on this point.

Samples Visual C++

```
int ret = DMS_Delete(pipe, "MA:MT:500:Value");
```

Delphi

```
lasterr := DMS_Delete(handle,'Motor:38:On');
if(lasterr <> 0) then
   MessageDlg('Fehler DMS_Delete', mtError, [mbOK], 0);
```

See also

2.3.8 DMS_SendCode

DMS_SendCode

Top Previous Next

Here are some commands like 'Save DMS values' that requieres a simple code.

Prototype

```
int _stdcall DMS_SendCode(HANDLE pipe, int code);
```

Returnvalue int — Zero, if sucessfull, see Win32 SDK-documentation.

Parameter

HANDLE pipe — Handle from DMS_Connect().

int code — One of the following constants:

ID_REBUILD	Rebuilds the tree in the DMS.
ID_LOAD_BMO	Load the templates
ID_SAVE_BMO	Save the templates
ID_SAVE_DMS	Save the full tree on disk
ID_DMS_SHOW	Make the DMS-window visible
ID_DMS_HIDE	Make the DMS-window hidden

Remarks Use DMS_Connect() to use this function.

Samples Visual C++

```
int ret = DMS_SendCode(pipe, ID_REBUILT);
```

Rebulds the DMS-tree.

Delphi

```
DMS_SendCode(handle, ID_REBUILD);
```

See also

2.3.9 DMS_ReadType

DMS_ReadType

Top Previous Next

Gets the type of a datapoint.

Prototype

```
int _stdcall DMS_ReadType(HANDLE pipe, TCHAR *name);
```

Returnvalue int

Type of the datapoint:

ID_NONE	No datapoint found
ID_BIT	Boolean
ID_BYS	Byte signed
ID_BYU	Byte unsigned
ID_WOS	Word signed
ID_WOU	Word unsigned
ID_DWS	Double Word signed
ID_DWU	Double Word unsigned
ID_FLT	Double (floating point value)
ID_STR	String (max. MAX_NAME characters)

Parameter

HANDLE	pipe	Handle from DMS_Connect().
TCHAR*	name	Datapoint-name.

Remarks Use DMS_Connect() to use this function.

Samples Visual C++

```
int type = DMS_ReadType(pipe, "Gruppe1:Motor1:Temperatur");
```

Delphi

```
typ := DMS_ReadType(handle, 'Gruppe1:Motor1:Temperatur');
case typ of
   1 :   MessageDlg('Datatype 1: BIT',   mtInformation, [mbOK], 0);
   2 :   MessageDlg('Datatype 2: BYS',   mtInformation, [mbOK], 0);
   3 :   MessageDlg('Datatype 3: BYU',   mtInformation, [mbOK], 0);
   4 :   MessageDlg('Datatype 4: WOS',   mtInformation, [mbOK], 0);
   5 :   MessageDlg('Datatype 5: WOU',   mtInformation, [mbOK], 0);
   6 :   MessageDlg('Datatype 6: DWS',   mtInformation, [mbOK], 0);
   7:   MessageDlg('Datatype 7: DWU',   mtInformation, [mbOK], 0);
   8:   MessageDlg('Datatype 8: FLT',   mtInformation, [mbOK], 0);
   9:   MessageDlg('Datatype 9: STR',   mtInformation, [mbOK], 0);
   else
   MessageDlg('Datatype 0: NONE',   mtInformation, [mbOK], 0);
end;
```

See also DMS_Read*, DMS_Write*

2.3.10 DMS_ReadBIT

DMS_ReadBIT

Top Previous Next

Gets the value of a datapoint.

Prototype

```
int _stdcall DMS_ReadBIT(HANDLE pipe, TCHAR *name, tBIT& value);
```

Returnvalue	int	0, if everything is o.k., otherwise LastError (see WIN-SDK-documentation (winerror.h)).
Parameter	HANDLE pipe	Handle from DMS_Connect().
	TCHAR* name	Datapoint-name.
	tBIT& value	The value will be returned in this variable.

Remarks Use DMS_Connect() to use this function.

Samples Visual C++

```
tBIT   value;
int ret = DMS_ReadBIT(pipe, "Gruppe1:Motor1:Ein", value);
```

Delphi

```
procedure TForm1.Timer1Timer(Sender: TObject);
var wert : tBIT;
begin

   DMS_ReadBIT(handle, 'Gruppe1:Motor1:Ein', wert);
   if(wert) then
     Zustand.Caption := 'Ein'
   else
     Zustand.Caption := 'Aus';

end;
```

See also DMS_Read*, DMS_Write*

2.3.11 DMS_ReadBYS

DMS_ReadBYS

Top Previous Next

Gets the value of a datapoint.

Prototype

```
int _stdcall DMS_ReadBYS(HANDLE pipe, TCHAR *name, tBYS& value);
```

Returnvalue	int	0, if everything is o.k., otherwise LastError (see WIN-SDK-documentation (winerror.h)).
Parameter	HANDLE pipe	Handle from DMS_Connect().
	TCHAR* name	Datapoint-name.
	tBYS& value	The value will be returned in this variable.

Remarks Use DMS_Connect() to use this function.

Samples Visual C++

```
tBYS   value;
int ret = DMS_ReadBYS(pipe, "Gruppe1:Motor1:Temperatur", value);
```

Delphi

```
var value : tBYS;

ret := DMS_ReadBYS(handle, ‘Gruppe1:Motor1:Temperatur‘, value);
```

See also DMS_Read*, DMS_Write*

2.3.12 DMS_ReadBYU

DMS_ReadBYU

Top Previous Next

Gets the value of a datapoint.

Prototype `int _stdcall DMS_ReadBYU(HANDLE pipe, TCHAR *name, tBYU& value);`

Returnvalue	int		0, if everything is o.k., otherwise LastError (see WIN-SDK-documentation (winerror.h)).
Parameter	HANDLE	pipe	Handle from DMS_Connect().
	TCHAR*	name	Datapoint-name.
	tBYU&	value	The value will be returned in this variable.

Remarks Use DMS_Connect() to use this function.

Samples Visual C++

```
tBYU   value;
int ret = DMS_ReadBYU(pipe, "Gruppe1:Motor1:Temperatur", value);
```

Delphi

```
var value : tBYU;

ret := DMS_ReadBYU(handle, ‘Gruppe1:Motor1:Temperatur‘, value);
```

See also DMS_Read*, DMS_Write*

2.3.13 DMS_ReadWOS

DMS_ReadWOS

Top Previous Next

Gets the value of a datapoint.

Prototype

```
int _stdcall DMS_ReadWOS(HANDLE pipe, TCHAR *name, tWOS& value);
```

Returnvalue	int		0, if everything is o.k., otherwise LastError (see WIN-SDK-documentation (winerror.h)).
Parameter	HANDLE	pipe	Handle from DMS_Connect().
	TCHAR*	name	Datapoint-name.
	tWOS&	value	The value will be returned in this variable.

Remarks Use DMS_Connect() to use this function.

Samples Visual C++

```
tWOS   value;
int ret = DMS_ReadWOS(pipe, "Gruppe1:Motor1:Temperatur", value);
```

Delphi

```
var value : tWOS;

ret := DMS_ReadWOS(handle, ‘Gruppe1:Motor1:Temperatur‘, value);
```

See also DMS_Read*, DMS_Write*

2.3.14 DMS_ReadWOU

DMS_ReadWOU

Top Previous Next

Gets the value of a datapoint.

Prototype `int _stdcall DMS_ReadWOU(HANDLE pipe, TCHAR *name, tWOU& value);`

Returnvalue	int		0, if everything is o.k., otherwise LastError (see WIN-SDK-documentation (winerror.h)).
Parameter	HANDLE	pipe	Handle from DMS_Connect().
	TCHAR*	name	Datapoint-name.
	tWOU&	value	The value will be returned in this variable.

Remarks Use DMS_Connect() to use this function.

Samples Visual C++

```
tWOU   value;
int ret = DMS_ReadWOU(pipe, "Gruppe1:Motor1:Temperatur", value);
```

Delphi

```
var value : tWOU;

ret := DMS_ReadWOU(handle, ‘Gruppe1:Motor1:Temperatur‘, value);
```

See also DMS_Read*, DMS_Write*

2.3.15 DMS_ReadDWS

DMS_ReadDWS

Top Previous Next

Gets the value of a datapoint.

Prototype

```
int _stdcall DMS_ReadDWS(HANDLE pipe, TCHAR *name, tDWS& value);
```

Returnvalue	int		0, if everything is o.k., otherwise LastError (see WIN-SDK-documentation (winerror.h)).
Parameter	HANDLE	pipe	Handle from DMS_Connect().
	TCHAR*	name	Datapoint-name.
	tDWS&	value	The value will be returned in this variable.

Remarks Use DMS_Connect() to use this function.

Samples Visual C++

```
tDWS   value;
int ret = DMS_ReadDWS(pipe, "Gruppe1:Motor1:Temperatur", value);
```

Delphi

```
var   value : tDWS;

ret := DMS_ReadDWS(handle, ‘Gruppe1:Motor1:Temperatur‘, value);
```

See also DMS_Read*, DMS_Write*

2.3.16 DMS_ReadDWU

DMS_ReadDWU

Top Previous Next

Gets the value of a datapoint.

Prototype

```
int _stdcall DMS_ReadDWU(HANDLE pipe, TCHAR *name, tDWU& value);
```

Returnvalue	int		0, if everything is o.k., otherwise LastError (see WIN-SDK-documentation (winerror.h)).
Parameter	HANDLE	pipe	Handle from DMS_Connect().
	TCHAR*	name	Datapoint-name.
	tDWU&	value	The value will be returned in this variable.

Remarks Use DMS_Connect() to use this function.

Samples Visual C++

```
tDWU   value;
int ret = DMS_ReadDWU(pipe, "Gruppe1:Motor1:Temperatur", value);
```

Delphi

```
ret   := DMS_ReadDWU(handle, ‘Gruppe1:Motor1:Temperatur‘, value);
```

See also DMS_Read*, DMS_Write*

2.3.17 DMS_ReadFLT

DMS_ReadFLT

Top Previous Next

Gets the value of a datapoint.

Prototype `int _stdcall DMS_ReadFLT(HANDLE pipe, TCHAR *name, tFLT& value);`

Returnvalue	int		0, if everything is o.k., otherwise LastError (see WIN-SDK-documentation (winerror.h)).
Parameter	HANDLE	pipe	Handle from DMS_Connect().
	TCHAR*	name	Datapoint-name.
	tFLT&	value	The value will be returned in this variable.

Remarks Use DMS_Connect() to use this function.

Samples Visual C++

```
tFLT   value;
int ret = DMS_ReadFLT(pipe, "Gruppe1:Motor1:Temperatur", value);
```

Delphi

```
procedure TForm1.Timer1Timer(Sender: TObject);
var dwert: tFLT;

begin

  DMS_ReadFLT(handle, 'Gruppe1:Motor1:Temperatur', dwert);
  Temperatur.Caption := FloatToStr(dwert);

end;
```

See also DMS_Read*, DMS_Write*

2.3.18 DMS_ReadSTR

DMS_ReadSTR

Top Previous Next

Gets the stringvalue of a datapoint.

Prototype

```
int _stdcall DMS_ReadSTR(HANDLE pipe, TCHAR *name, tSTR* value);
```

Returnvalue int — 0, if everything is o.k., otherwise LastError (see WIN-SDK-documentation (winerror.h)).

Parameter

HANDLE	pipe	Handle from DMS_Connect().
TCHAR*	name	Datapoint-name.
tSTR&	value	The value will be returned in this variable.

Remarks Use DMS_Connect() to use this function.

Samples Visual C++

```
tSTR   value[MAX_NAME];
int ret = DMS_ReadSTR(pipe, "Gruppe1:Motor1:Temperatur", value);
```

Delphi

```
procedure TForm1.Button8Click(Sender: TObject);
var text : tSTR;
begin
   DMS_ReadSTR(handle, 'Gruppe1:Motor1:BEZEICHNUNG', text);
   Button8.Caption := text;
end;
```

See also DMS_Read*, DMS_Write*

2.3.19 DMS_WriteBIT

DMS_WriteBIT

Top Previous Next

Set the value of a datapoint.

Prototype `int _stdcall DMS_WriteBIT(HANDLE pipe, TCHAR *name, tBIT bwert);`

Returnvalue	int		0, if everything is o.k., otherwise LastError (see WIN-SDK-documentation (winerror.h)).
Parameter	HANDLE	pipe	Handle from DMS_Connect().
	TCHAR*	name	Datapoint-name.
	tBIT	value	The value to be set in the DMS.

Remarks Use DMS_Connect() to use this function.

Samples Visual C++

```
tBIT   value = TRUE;
int ret = DMS_WriteBIT(pipe, "Gruppe1:Motor1:On", value);
```

Delphi

```
procedure TForm1.Button10Click(Sender: TObject);
var value : tBIT;
begin
   DMS_ReadBIT(handle, 'Gruppe1:Motor3:Ein', value);
   if(value) then
   value := FALSE
   else
   value := TRUE;

   DMS_WriteBIT(handle, 'Gruppe1:Motor3:Ein', value);
end;
```

See also DMS_Read*, DMS_Write*

2.3.20 DMS_WriteBYS

DMS_WriteBYS

Top Previous Next

Set the value of a datapoint.

Prototype `int _stdcall DMS_WriteBYS(HANDLE pipe, TCHAR *name, tBYS bwert);`

Returnvalue	int		0, if everything is o.k., otherwise LastError (see WIN-SDK-documentation (winerror.h)).
Parameter	HANDLE	pipe	Handle from DMS_Connect().
	TCHAR*	name	Datapoint-name.
	tBYS	value	The value to be set in the DMS.

Remarks Use DMS_Connect() to use this function.

Samples Visual C++

```
tBYS   value = 7;
int ret = DMS_WriteBYS(pipe, "Gruppe1:Motor1:Test", value);
```

Delphi

```
var value : tBYS;

value := 44;
DMS_WriteBYS(handle, 'Gruppe1:Motor1:Test', value);
```

See also DMS_Read*, DMS_Write*

2.3.21 DMS_Write_BYU

DMS_Write_BYU

Top Previous Next

Set the value of a datapoint.

Prototype `int _stdcall DMS_WriteBYU(HANDLE pipe, TCHAR *name, tBYU bwert);`

Returnvalue	int		0, if everything is o.k., otherwise LastError (see WIN-SDK-documentation (winerror.h)).
Parameter	HANDLE	pipe	Handle from DMS_Connect().
	TCHAR*	name	Datapoint-name.
	tBYU	value	The value to be set in the DMS.

Remarks Use DMS_Connect() to use this function.

Samples Visual C++

```
tBYU   value = 255;
int ret = DMS_WriteBYU(pipe, "Gruppe1:Motor1:Default", value);
```

Delphi

```
var value : tBYU;

value := 255;
DMS_WriteBYU(handle, 'Gruppe1:Motor1:Test', value);
```

See also DMS_Read*, DMS_Write*

2.3.22 DMS_WriteWOS

DMS_WriteWOS

Top Previous Next

Set the value of a datapoint.

Prototype `int _stdcall DMS_WriteWOS(HANDLE pipe, TCHAR *name, tWOS bwert);`

Returnvalue	int		0, if everything is o.k., otherwise LastError (see WIN-SDK-documentation (winerror.h)).
Parameter	HANDLE	pipe	Handle from DMS_Connect().
	TCHAR*	name	Datapoint-name.
	tWOS	value	The value to be set in the DMS.

Remarks Use DMS_Connect() to use this function.

Samples Visual C++

```
tBWOS   value = 0x4A;
int ret   = DMS_WriteWOS(pipe, "Gruppe1:Motor1:Value", value);
```

Delphi

```
var value : tWOS;

value := 44;
DMS_WriteWOS(handle, 'Gruppe1:Motor1:Test', value);
```

See also DMS_Read*, DMS_Write*

2.3.23 DMS_WriteWOU

DMS_WriteWOU

Top Previous Next

Set the value of a datapoint.

Prototype

```
int _stdcall DMS_WriteWOU(HANDLE pipe, TCHAR *name, tWOU bwert);
```

Returnvalue	int	0, if everything is o.k., otherwise LastError (see WIN-SDK-documentation (winerror.h)).
Parameter	HANDLE pipe	Handle from DMS_Connect().
	TCHAR* name	Datapoint-name.
	tWOU value	The value to be set in the DMS.

Remarks Use DMS_Connect() to use this function.

Samples Visual C++

```
tWOU   value = 12345;
int ret = DMS_WriteWOU(pipe, "Gruppe1:Motor1:Value", value);
```

Delphi

```
var value : tWOU;

value := 44444;
DMS_WriteWOU(handle, 'Gruppe1:Motor1:Test', value);
```

See also DMS_Read*, DMS_Write*

2.3.24 DMS_WriteDWS

DMS_WriteDWS

Top Previous Next

Set the value of a datapoint.

Prototype `int _stdcall DMS_WriteDWS(HANDLE pipe, TCHAR *name, tDWS bwert);`

Returnvalue	int		0, if everything is o.k., otherwise LastError (see WIN-SDK-documentation (winerror.h)).
Parameter	HANDLE	pipe	Handle from DMS_Connect().
	TCHAR*	name	Datapoint-name.
	tDWS	value	The value to be set in the DMS.

Remarks Use DMS_Connect() to use this function.

Samples Visual C++

```
tDWS   value = 9876;
int ret = DMS_WriteDWS(pipe, "Gruppe1:Motor1:Count", value);
```

Delphi

```
var value : tDWS;

value := 0;
DMS_WriteDWS(handle, 'Gruppe1:Motor1:Test', value);
```

See also DMS_Read*, DMS_Write*

2.3.25 DMS_WriteDWU

DMS_WriteDWU

Top Previous Next

Set the value of a datapoint.

Prototype `int _stdcall DMS_WriteDWU(HANDLE pipe, TCHAR *name, tDWU bwert);`

Returnvalue	int		0, if everything is o.k., otherwise LastError (see WIN-SDK-documentation (winerror.h)).
Parameter	HANDLE	pipe	Handle from DMS_Connect().
	TCHAR*	name	Datapoint-name.
	tDWU	value	The value to be set in the DMS.

Remarks Use DMS_Connect() to use this function.

Samples Visual C++

```
tDWU   value = 888888;
int ret = DMS_WriteDWU(pipe, "Gruppe1:Motor1:Count", value);
```

Delphi

```
var value : tDWU;

value := 44;
DMS_WriteDWU(handle, 'Gruppe1:Motor1:Test', value);
```

See also DMS_Read*, DMS_Write*

2.3.26 DMS_WriteFLT

DMS_WriteFLT

Top Previous Next

Set the value of a datapoint.

Prototype `int _stdcall DMS_WriteFLT(HANDLE pipe, TCHAR *name, tFLT bwert);`

Returnvalue	int		0, if everything is o.k., otherwise LastError (see WIN-SDK-documentation (winerror.h)).
Parameter	HANDLE	pipe	Handle from DMS_Connect().
	TCHAR*	name	Datapoint-name.
	tFLT	value	The value to be set in the DMS.

Remarks Use DMS_Connect() to use this function.

Samples Visual C++

```
tFLT   value = 3.14159;
int ret = DMS_WriteFLT(pipe, "Gruppe1:Motor1:Temp", value);
```

Delphi

```
procedure TForm1.Temp3Click(Sender: TObject);
begin
   DMS_WriteFLT(handle, 'Gruppe1:Motor3:Temperatur', 23.44);
end;
```

See also DMS_Read*, DMS_Write*

2.3.27 DMS_WriteSTR

DMS_WriteSTR

Top Previous Next

Set the value of a datapoint.

Prototype `int _stdcall DMS_WriteSTR(HANDLE pipe, TCHAR *name, tSTR* swert);`

Returnvalue	int		0, if everything is o.k., otherwise LastError (see WIN-SDK-documentation (winerror.h)).
Parameter	HANDLE	pipe	Handle from DMS_Connect().
	TCHAR*	name	Datapoint-name.
	tSTR	value	The value to be set in the DMS.

Remarks Use DMS_Connect() to use this function.

Samples Visual C++

```
int ret = DMS_WriteSTR(pipe, "Gruppe1:Motor1:Text", "Hello");
```

Delphi

```
var text : tSTR;

text := 'This is a test';
DMS_WriteSTR(handle, 'Gruppe1:Motor1:BEZEICHNUNG', text);
```

See also DMS_Read*, DMS_Write*

2.3.28 DMS_GetRights

DMS_GetRights

Top Previous Next

Read the rights of a datapoint.

Prototype

```
char _stdcall DMS_GetRights(HANDLE pipe, TCHAR *Name);
```

Returnvalue

READ_ONLY	
READ_WRITE	
REMANENT	(ORed with READ_ONLY or READ_WRITE)

Parameter

HANDLE	pipe	Handle from DMS_Connect().
TCHAR*	name	Datapoint-name.

Remarks

Use DMS_Connect() to use this function.

Samples

Visual C++

```
char rights = DMS_GetRights(pipe, "Gruppe1:Motor1:On")
if(rights & READ_WRITE) {
   ...
}
```

Delphi

```
var rights : char;
begin
   rights := DMS_GetRights(handle, 'Gruppe1:Motor1:Ein');
end;
```

See also

DMS_SetRights

2.3.29 DMS_SetRights

DMS_SetRights

Top Previous Next

Set the rights of a datapoint.

Prototype

```
Char _stdcall DMS_SetRights(HANDLE pipe, TCHAR *Name,
                            char rights);
```

Returnvalue

READ_ONLY	
READ_WRITE	
REMANENT	(ORed with READ_ONLY or READ_WRITE)

Parameter

HANDLE	pipe	Handle from DMS_Connect().
TCHAR*	name	Datapoint-name.
char	rights	Rights-flags (see return-value)

Remarks

Use DMS_Connect() to use this function.

Samples

Visual C++

```
char rights = DMS_SetRights(pipe, "Gruppe1:Motor1:On",
      READ_WRITE | REMANENT)
```

Delphi

```
rights := char(READ_ONLY);
DMS_SetRights(handle, 'Gruppe1:Motor1:Ein', rights);
```

See also

2.3.30 DMS_FindMessage

DMS_FindMessage

Top Previous Next

Get point-data from DMS to an internal array. To access the data use the function DMS_FindNextMessage().

Prototype `int _stdcall DMS_FindMessage(HANDLE pipe, TCHAR *Name);`

Returnvalue int — Count of points, that correspond with Name

Parameter

HANDLE	pipe	Handle from DMS_Create().
TCHAR*	name	Requested DMS-points

Remarks Use DMS_Connect() to use this function.

Samples Visual C++

```
int Count = DMS_FindMessage(pipe, "*");

Counts all DMS data points
```

Delphi

```
public
  handle : THandle;
end;

count:=DMS_FindMessage(handle,'system');

Counts the data points with the name system
```

See also DMS_FindNextMessage

2.3.31 DMS_FindNextMessage

DMS_FindNextMessage

Top Previous Next

Get point-data from DMS that are retreaves with the function DMS_FindMessage().

Prototype `TCHAR* _stdcall DMS_FindNextMessage(Message* msg);`

Returnvalue TCHAR * DMS-name

Parameter Message* msg Pointer to a DMS-message (see Message-structure).

Remarks Use DMS_Connect() to use this function.

Samples Visual C++

```
Message msg;
int Count = DMS_FindMessage(pipe, "TEST");
for(int i=0; i< Count; i++){
   DMS_FindNextMessage(&msg);
   printf("\nName %s  Type: %d", msg.point_name, msg.type);
}

Finds all DMS-names with 'TEST' and print the DMS-path and type on the screen.
```

Delphi

```
var

   msgA  : TDMSMessage;

count:=DMS_FindMessage(handle,'Error');
For i:=1 to count do begin
   DMS_FindNextMessage(msgA);
   memo2.Lines[i]:=memo1.Lines[i]+msgA.point_name+' : '+   inttostr(msgA.ValueType);
end;


Finds all DMS-names with 'Error' and print the DMS-path and type on the screen.
```

See also DMS_FindMessage

2.3.32 DMS_GetNames

DMS_GetNames

Top Previous Next

Search the sons of a datapoint.

Prototype `int _stdcall DMS_GetNames(HANDLE pipe, TCHAR *name);`

Returnvalue	int	ret	0, if everything is OK, otherwise LastError. Last error (see Win32 SDK documentation)
Parameter	HANDLE	pipe	Handle from DMS_Connect().
	TCHAR*	name	String with a datapoint-name. If the string is empty (""), the function return the root-entries.

Remarks Use DMS_Connect() to use this function.

Samples Visual C++

```
char   DMSName[MAX_NAME];
strcpy(DMSName,"System");

if(DMS_GetNames(DMS, DMSName)== 0) {

   TCHAR *p;

   p = DMS_GetNextName(DMS, son);
   int listcount = 0;
   while(*p) {
   m_DMS_List.AddString(p);
   if(++listcount >= MAX_LIST) break;
   p = DMS_GetNextName(DMS, son);
   }
}

Every datapoint below 'System' is copied in a stringlist (Date, Time ...).
```

Delphi

```
err := DMS_GetNames(handle,'System:Alm');

err is equal zero if the data point 'System:Alm' exists
```

See also DMS_GetNextName

2.3.33 DMS_GetNextName

DMS_GetNextName

Top Previous Next

Retrieve the datapoint-name.

Prototype `TCHAR* _stdcall DMS_GetNextName(HANDLE pipe, int& sons);`

Returnvalue	TCHAR* name	A pointer to the name. If the datapoint has sons, the second parameter is set to 1.
Parameter	HANDLE pipe	Handle from DMS_Connect().
	int& sons	if the datapoints has sons, this value will be set to 1.

Remarks Use DMS_Connect() to use this function.

Samples Visual C++

```
char   DMSName[MAX_NAME];
strcpy(DMSName,"System");
if(DMS_GetNames(DMS,DMSName)== 0) {

   TCHAR *p;

   p = DMS_GetNextName(DMS, son);
   int listcount = 0;
   while(*p) {
   m_DMS_List.AddString(p);
   if(++listcount >= MAX_LIST) break;
   p = DMS_GetNextName(DMS, son);
   }
}
```

Every datapoint below 'System' is copied in a stringlist (Date, Time ...). If the variable 'son' is set to 1, then the datapoint have a son. You can call a function recursive to obtain the whole DMS-tree.

Delphi

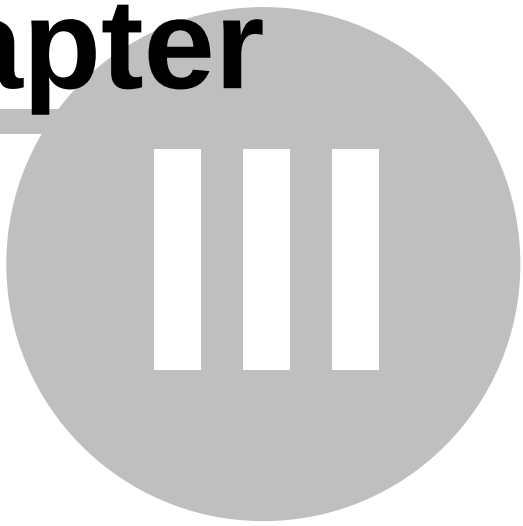
```
var   err,son,   i   : longint;
   st   : String;
   p   : pchar;

err := DMS_GetNames(handle,'System:ALM');
p   := DMS_GetNextName(handle, son);
st  := string(p);
while st <> '' do begin
      memo1.Lines[i]:=memo1.Lines[i]+p+' ';
      p := DMS_GetNextName(handle, son);
      st :=string(p);
end;
inc(i);
```

Every datapoint below 'System:ALM' is copied in a stringlist (Date, Time ...). If the variable 'son' is set to 1, then the datapoint have a son. You can call a function recursive to obtain the whole DMS-tree.

See also DMS_GetNames

Chapter



3 PDBS Functions

PDBS-Functions

Top Previous Next

To access history data use one of the following functions:

- PdbsConnect(...);
- PdbsDisconnect(...);
- PdbsGetData(...);
- PdbsGetCount(...);

To access protocoll-data use one of the following functions:

- PDBS_Open(...);
- PDBS_Close(...);
- PDBS_IsOpen(...);
- PDBS_GetCount(...);
- PDBS_GetTimeCount(...);

- PDBS_Append(...);

- PDBS_GetBulkData(...);
- PDBS_Move(...);
- PDBS_MoveNext(...);
- PDBS_MovePrev(...);
- PDBS_SetFilterDMS(...);
- PDBS_SetFilterText(...);
- PDBS_ClearFilter(...);
- PDBS_MoveTime(...);

To access alarm-data use the following functions:

- PDBS_GetAlarm(...);
- PDBS_PutAlarm(...);
- PDBS_FilterAlarm(...);

3.1 Main Functions

Main functions

Top Previous Next

3.1.1 PdbsConnect

PdbsConnect

Top Previous Next

Connects to the PDBS-manager-task.

Prototype HANDLE _stdcall PdbsConnect(TCHAR* pcname);

Returnvalue	HANDLE	if everything is OK., 0, if the connection can't be done. Use GetLastError() to get extended error information.
Parameter	TCHAR* pcname	The PC-name can be any PC-name in the network. To connect to the local PDBS use "." or the local PC-name.

Remarks Use DMS_Connect() to use this function.

Samples Visual C++

```
HANDLE pdbs;

pdbs = PdbsConnect("."); // Local PC
if(!pdbs) {
   AfxMessageBox("Can't connect to PDBS");
}
```

See also PdbsDisconnect()

3.1.2 PdbsDisconnect

PdbsDisconnect

Top Previous Next

Disconnects from the PDBS-task.

Prototype BOOL _stdcall PdbsDisconnect(HANDLE hpdbs);

Returnvalue If the function succeeds, the return value is nonzero.
If the function fails, the return value is zero.
To get extended error information, call GetLastError.

Parameter HANDLE hpdbs — Handle from PdbsConnect().

Remarks

Samples *Visual C++*

```
HANDLE pdbs;

pdbs = PdbsConnect("."); // Local PC
if(!pdbs) {
   AfxMessageBox("Can‘t connect to PDBS");
}

...

If(pdbs)
   PdbsDisconnect(pdbs);
```

See also PdbsConnect

3.2 History Data

History-Data

Top Previous Next

History-data are sampled by the HDA-programm (History-Data-Aquisition). Use PET (ProMoS Engineering Tool) to define, which data are to be trended.

```
HANDLE _stdcall PdbsConnect(TCHAR* pcname);

BOOL _stdcall PdbsDisconnect(HANDLE hPipe);

int _stdcall PdbsGetData(HANDLE hPipe, TCHAR* dmsname, time_t start, time_t end,
int count, DBData* Data);

int _stdcall PdbsGetCount(HANDLE hPipe, TCHAR* dmsname, time_t start, time_t end);
```

To get history-data use the PdbsConnect-function to get a handle. Every call use this handle.

3.2.1 PdbsGetData

PdbsGetData

Top Previous Next

Get history data.

Prototype

```
int _stdcall PdbsGetData(HANDLE hPdbs, TCHAR* dmsname,
time_t start, time_t end, int count, DBData* Data);
```

Returnvalue int count of data.

Parameter

HANDLE	hPdbs	Handle from PdbsConnect()
TCHAR	dmsname	Name of datapoint
time_t	start	Starttime (seconds since 1.1.1970)
time_t	end	Endtime (seconds since 1.1.1970)
int	count	Number of data to retreive
DBData*	data	Array to store the data

Remarks The structure of DBData is

```
typedef struct {
    time_t      zeit;      // timestamp
    float       wert;      // value
    int         status;    // State
}DBData;
```

The size of the array has to be at least the double of the count +1, because the PDBS will return the highest and the lowest value per count.
Example: If you retreive 100 data in a time period of one month (10000 data on the disk), the function will return 200 values, with the minimum and maximum values for a period of 1/100 of a month.

Samples *Visual C++*

```
DBData data[1001];
time_t   start, end;

end   = time(NULL);
start   = end - 86400;   // 24 hours

PdbsGetData(hPdbs, "Gruppe1:Motor1:Temperatur",
      start, end, 500, &data);
```

See also PdbsGetCount()

3.2.2 PdbsGetCount

PdbsGetCount

Top Previous Next

Get the count of data between two timestamps.

Prototype

```
int _stdcall PdbsGetCount(HANDLE hPdbs, TCHAR* dmsname,
time_t start, time_t end);
```

Returnvalue int count of data

Parameter

HANDLE hPdbs		Handle form PdbsConnect()
TCHAR* dmsname		Name of datapoint
time_t	start	starttime (seconds since 1.1.1970)
time_t	end	endtime (seconds since 1.1.1970)

Remarks

Samples Visual C++

```
time_t   start, end;

end = time(NULL);
start = time - 86400:

int count = PdbsGetCount(hPdbs, "Gruppe1:Motor1:Temperatur", start, end);
```

See also

3.2.3 PdbsGetLastData

PdbsGetLastData

Top Previous Next

Get the last data from a history database.

Prototype `BOOL PdbsGetLastData(HANDLE hPipe, TCHAR* DMSName, DBData* Data);`

Returnvalue int count of data

Parameter

HANDLE hPdbs	Handle form PdbsConnect()
TCHAR* dmsname	Name of datapoint
DBData* Data	Pointer to a data structure

```
typedef struct {
time_t          zeit;     // Time of last entry
float           wert;      // Last value
int             status;     // State of last entry
}DBData;
```

Remarks

Samples Visual C++

```
DBData        data;
if(PdbsGetLastData(pdbs, "MT:500:Ist", &data)) {
  if(!data.zeit)
    cout << "\nNo timestamp found!" << endl;
  else {
    CString Output;
    CTime   Zeit = data.zeit;
    Output.Format("%s -> %0.3f", Zeit.Format("%d.%m.%y %H:%M:%S"), data.wert);
    cout << "\nZeit: " << Output.GetBuffer(30) << endl;
  }
}
```

See also

3.2.4 PdbsAppendTrd

PdbsAppendTrd

Top Previous Next

Append new data to a history-database.

Prototype

```
BOOL PdbsAppendTrd(HANDLE hPipe, TCHAR* DMSName, DBData* Data);
```

Returnvalue TRUE, if data are found
FALSE, if no data found

Parameter

HANDLE	hPdbs	Handle form PdbsConnect()
TCHAR*	dmsname	Name of datapoint
DBData*	Data	Data to strore (DBData see PdbsGetLastData())

Remarks It's not possible to insert data at a specific time. The timestamp of the appended data must be higher than the last existing data in the database. Use PdbsGetLastData() to check the last saved data.

The state stored in DBData can be one of the following states:

CYCLE_SAVE	Data came from a cyclic recording
CHANGE	Data are stored because of a value change
DIFF	Data are strored because of a difference in values
NEWFILE	Data will be stored after checking, if file exists
NEWDATA	Data are marked, that the recording was interrupted

States can be ORed (Sample: CHANGE | NEWDATA).

Samples Visual C++

```
DBData   trddata;
trddata.zeit   = time(NULL);   // Timestamp
trddata.wert   = (float)1.23;   // Value
trddata.status = CHANGE;   // State

if(PdbsAppendTrd(pdbs, "MT:500:Ist", &trddata))
  cout << "\nData appended..." << endl;
else
  cout << "\nERROR PDBSAppendTrd()" << endl;
```

See also PdbsGetLastData()

3.3 Alarm Data

Alarm-data Top Previous Next

3.3.1 PDBS_GetAlarm

PDBS_GetAlarm Top Previous Next

Gets every active alarm.

Prototype int_stdcall PDBS_GetAlarm(HANDLE hPdbs, PDBSData* Data)

Returnvalue int 0, if everything is OK.
otherwise GetLastError()

Parameter

HANDLE hPdbs	Handle from PdbsConnect()
PDBSData* data	Array of data to store the records

Remarks The routine returns all alarms on a system (new alarms, quit alarms, gone alarms). If an alarm is quitted but still pending, the alarm only get a new state. If an alarm has gone, but is not quitted yet, the alarm remains in the list.

There are some DMS-datapoints to detect, if the alarmlist has changed. (System:ALM:NewAlarm and System:ALM:Count).

Samples Visual C++

```
PDBSData*   Data = new PDBSData[MAX_DATA];

int count = PDBS_GetAlarm(pdbs, Data);


Structure of PDBSData

#define   PDMSDATASIZE   256


typedef struct {
   time_t   reftime;   // Alarmtime
   int   milli;
   TCHAR   DMSName[MAX_NAME+3];
   int   status;
   TCHAR   Text[PDMSDATASIZE - (20 +
   (MAX_NAME+3))];
   int   group;   // Alarmgroup
   int   pri;   // Priority
}PDBSData;

Status has one of the folowing states:

ALARM_KOMMT   New alarm
ALARM_GEHT   Alarm has gone
ALARM_QUIT   Alarm was quitted by a user
```

See also

3.3.2 PDBS_PutAlarm

PDBS_PutAlarm

Top Previous Next

Stores a new alarm in the alarmlist (only active alarms).

Prototype int _stdcall PDBS_GetAlarm(HANDLE hPipe, PDBSData* Data)

Returnvalue int 0, if everything is OK.

Parameter

HANDLE	hPdbs	Handle from PdbsConnect()
PDBSData*	data	Record-data

Remarks The data are not stored in a database (on disk). PDBS stores all active alarms in a internal RAM-based array.

Samples

See also

3.3.3 PDBS_FilterAlarm

PDBS_FilterAlarm

Top Previous Next

Set a filter.

Prototype int _stdcall PDBS_FilterAlarm(HANDLE hPipe, Filtereinstellung* Data)

Returnvalue int 0, if everything is OK.

Parameter

HANDLE	hPdbs	Handle from PdbsConnect()
Filtereinstellung*	Data	Record-data

Remarks Structure of Filtereinstellung

```
typedef struct {
   BOOL   ADMSName;
   BOOL   AZeit;
   BOOL   AGruppe;
   BOOL   APri;
   TCHAR   DMSName[MAX_NAME];
   int   Gruppe;
   int   Pri;
   time_t   Startzeit;
   time_t   Endzeit;
}Filtereinstellung;
```

This function is NOT TESTET yet !!!

Samples

See also

3.4 Protocol Data

Protocol-data

Top Previous Next

Protocol-data are stored by the PRT-manager (prtmng.exe).

To access data (or store new protocoll-data) use the following functions:

```
int _stdcall PDBS_Open(HANDLE hPipe, TCHAR* Filename);
int _stdcall PDBS_Close(HANDLE hPipe, TCHAR* Filename);
int _stdcall PDBS_IsOpen(HANDLE hPipe, TCHAR* Filename);
int _stdcall PDBS_GetCount(HANDLE hPipe, TCHAR* Filename);
int _stdcall PDBS_GetTimeCount(HANDLE hPipe, TCHAR* Filename, time_t start, time_t
end);

int _stdcall PDBS_Append(HANDLE hPipe, TCHAR* Filename, PDBSData* Data);

int _stdcall PDBS_GetBulkData(HANDLE hPipe, TCHAR* Filename, int position, int
count, PDBSData* Data);
int _stdcall PDBS_Move(HANDLE hPipe, TCHAR* Filename, int position, PDBSData*
Data);
int _stdcall PDBS_MoveNext(HANDLE hPipe, TCHAR* Filename, PDBSData* Data);
int _stdcall PDBS_MovePrev(HANDLE hPipe, TCHAR* Filename, PDBSData* Data);
int _stdcall PDBS_SetFilterDMS(HANDLE hPipe, TCHAR* Filename, TCHAR* DMSName);
int _stdcall PDBS_SetFilterText(HANDLE hPipe,        TCHAR* Filename, TCHAR*
Text);
int _stdcall PDBS_ClearFilter(HANDLE hPipe,        TCHAR* Filename);
int _stdcall PDBS_MoveTime(HANDLE hPipe, TCHAR* Filename,        time_t zeit);
```

3.4.1 PDBS_Open

PDBS_Open

Top Previous Next

Open a protocol-database.

Prototype int _stdcall PDBS_Open(HANDLE hPdbs, TCHAR* Filename)

Returnvalue int 1, if file is open
0, if file is closed, or error

Parameter

HANDLE	hPdbs	Handle from PdbsConnect()
TACHR*	Filename	Filename to open

Remarks You don‘t have to open the protocol-database, if you want to read or append data. If the file is not open, the PDBS-task will open it automaticaly.

By opening a file, the file will be opened by the PDBS-task and not by your program.

Use this function only when you use Move, MoveNext, MovePrev-functions.

Those functions are not testet yet !!!

Samples

See also PDBS_Close()

3.4.2 PDBS_Close

PDBS_Close

Top Previous Next

Close a protocol-database.

Prototype int _stdcall PDBS_Close(HANDLE hPipe, TCHAR* Filename)

Returnvalue int 0, if everything is OK.
1, if still open

Parameter

HANDLE	hPdbs	Handle from PdbsConnect()
TACHR*	Filename	Filename to open

Remarks

Samples

See also

3.4.3 PDBS_IsOpen

PDBS_IsOpen

Top Previous Next

Tests, if a database is opened.

Prototype int _stdcall PDBS_IsOpen(HANDLE hPipe, TCHAR* Filename)

Returnvalue int 1, if file is open
0, if file is closed, or error

Parameter

HANDLE	hPdbs	Handle from PdbsConnect()
TACHR*	Filename	Filename to open

Remarks

Samples

See also

3.4.4 PDBS_GetCount

PDBS_GetCount

Top Previous Next

Gets the count of records in a database.

Prototype int _stdcall PDBS_GetCount(HANDLE hPipe, TCHAR* Filename)

Returnvalue	int		Status of the file (1 = open, 0 = closed)
Parameter	HANDLE	hPdbs	Handle from PdbsConnect()
	TACHR*	Filename	Filename to open

Remarks

Samples

See also

3.4.5 PDBS_Append

PDBS_Append

Top Previous Next

Append a record to a protocoll-database.

Prototype

int _stdcall PDBS_Append(HANDLE hPipe, TCHAR* Filename, PDBSData* Data)

Returnvalue

int 0, if everything is OK.
See GetLastError to get error informations.

Parameter

HANDLE	hPdbs	Handle from PdbsConnect()
TACHR*	Filename	Filename to open
PDBSData*	Data	Record to append

Remarks

Structure of protocoll-data

```
#define PDMSDATASIZE 256

typedef struct {
      time_t       reftime;
      int           milli;
      TCHAR    DMSName[MAX_NAME+3];
      int           status;
      TCHAR    Text[PDMSDATASIZE - (20 + (MAX_NAME+3))];
      int           group;
      int           pri;       // Priorität
}PDBSData;

// 20 = 5* sizeof(time_t) or (int)
```

Samples

See also

3.4.6 PDBS_GetBulkData

PDBS_GetBulkData

Top Previous Next

Get protocoll-data.

Prototype

int _stdcall PDBS_GetBulkData(HANDLE hPipe, TCHAR* Filename,int position, int count, PDBSData* Data)

Returnvalue

int count

Parameter

HANDLE	hPdbs	Handle from PdbsConnect()
TACHR*	Filename	Filename to open
int	position	Startposition in file
int	count	Count (number of protocoll-records to retreive)
PDBSData*	data	Array of data to store the records

Remarks

Structure of protocoll-data

```
#define       PDMSDATASIZE       256

typedef struct {
      time_t       reftime;
      int           milli;
      TCHAR     DMSName[MAX_NAME+3];
      int           status;
      TCHAR     Text[PDMSDATASIZE - (20 + (MAX_NAME+3))];
      int           group;
      int           pri;       // Priorität
}PDBSData;
```

Samples

See also

Chapter IV

4 Examples

4.1 Delphi

Delphi

Top Previous Next

Use the unit promos

```
uses
Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs, promos,
StdCtrls;
```

In your main class you have to define a Thandle like

```
public
{ Public-Deklarationen }
handle : THandle;
```

In the FormCreate()-method you create a connection to the DMS.

```
DMS_Connect('\\.\pipe\PROMOS-DMS', handle);
```

If you use the callback function you have to declare a function like the following:

```
function HandleMessage(var msg: TDMSMessage): Integer; cdecl;
begin

case msg.obj_id of
4800  :  begin
    Form1.Edit3.Text := FloatToStr(msg.value.val_FLT);
    end;
4801  :  begin
    Form1.Edit5.Text := FloatToStr(msg.value.val_FLT);
    end;
end;

HandleMessage := 0;
end;
```

and to create a connection with

```
ConnectDMS(Pipename, cl, HandleMessage);
```

The cl-variable is usualy a global variable of type TClient

```
cl        : TClient;
```

See the Delphi-sample for more details.

Chapter V

5 Structures

Appendix

Top Previous

Structures (Records)

Message communicating over pipe:

C++

```
struct message{
int   message_id;   //message id.
TCHAR   point_name[MAX_NAME];   //name of a D-point
short   status;   /status word
int   reply_id;   //id. for reply from server
char   rights;   //rights for a D-point
int   obj_id;   //object Id. for registering
char   type;   //value type
struct   value_type value;   //value.
time_t   time;   // Last Update
unsigned short   millitm;   // Last Update (Millisekunden)
};

typedef struct message   Message;


//struct of D-point value for double - till char
struct   value_type{
tBIT   val_BIT;   // ID_BIT
tBYS   val_BYS;   // ID_BYS
tWO   Sval_WOS;   // ID_WOS
tDWS   val_DWS;   / ID_BYU
tWOU   val_WOU;   //// ID_DWS
tBYU   val_BYU;   / ID_WOU
tDWU   val_DWU;   // ID_DWU
tFLT   val_FLT;   // ID_FLT
tSTR   val_STR[MAX_TEXT];   // ID_STR
};
```

Delphi

// structure to save DMS-data

```
TValueType = record
val_BIT: tBIT;
val_BYS: tBYS;
val_WOS: tWOS;
val_DWS: tDWS;
val_BYU: tBYU;
val_WOU: tWOU;
val_DWU: tDWU;
val_FLT: tFLT;
val_STR: tSTR;
end;

// Message (callback)
TDMSMessage = Record
message_id: LongInt;   {message id}
point_name: array [0..80] of char;   {name of a D-point}
status: Word;   {status word}
reply_id: LongInt;   {id. for reply from server}
rights: Byte;   {rights for a D-point}
obj_id: LongInt;   {object Id. for registering}
ValueType: Byte;   {value type}
value: TvalueType;   {value: grösstes Unionelement}
time : LongInt;   {LastUpdate}
milli: Word;   {Zeitstempel Millisekunden}
end;
```