PowerTABLE 2.00 - Pre Release

bytes + letters
software technology

postfach 16 27
66716 Saarlouis, germany

tel: +49 (68 31) 96 45 96
fax: +49 (68 31) 96 45 97

100275.3554@compuserve.com

(Help file is not complete in this version)

Main Window

(No help available in this version)

Database Window

(No help available in this version)

Open/Create Database Dialog

(No help available in this version)

Edit Window

(No help available in this version)

SQL Window

(No help available in this version)

Table Layout

(No help available in this version)

About

(No help available in this version)

Convert Database

(No help available in this version)

Options Dialog

(No help available in this version)

Shareware Version

bytes + letters
software technology hilger
postfach 16 27

66716 Saarlouis, germany

First: If you have problems with this tool or if you find a bug - please, please inform us about it!!!

That´s the only way to make a bug-free product.

Our e-mail adress is:

CompuServe-ID: 100275, 3554

If you decide to use PowerTable in the future (and if you don´t want to see the About-Dialog anymore) you must register it first.

There are two possibilities to do this:

Via CompuServe

GO SWREG

The product number is 7790. You can also find it with the keyword »POWERTABLE«. The registration fee is US $55. After we have received your registration, you´ll get a User ID and a Serial Number. After you typed in these values, there will be no more limits and no more messages.

By Mail

Our adress is:

bytes + letters
software technology hilger
Postfach 16 27

66716 Saarlouis
Germany

Registration >>

If you are from anywhere in the world ....
please enclose a cheque of US$59. A few days later, you´ll get the registration number and ID by fax or mail. As you like.
Thanks a lot.

Wenn Sie aus Deutschland sind, ...
dann senden Sie bitte Bargeld oder einen Scheck in Höhe von DM 79,-- an unsere Postadresse. In diesem Fall erhalten Sie eine Registrierungsbestätigung, bzw. eine Quittung mit ausgewiesener MwSt. per Post oder per Fax. Ganz wie Sie wünschen.
Herzlichen Dank.

Registration

To
bytes + letters
software technology hilger
Postfach 1627
66716 Saarlouis
Germany

Registering PowerTable, Version 2.00

Please send a registration ID and a serial number to:

Title _____
Name _____
Street _____
City/Zip _____
State, Country _____

Phone _____
Fax _____
E-Mail _____

I´ve enclosed:

( ) Cheque, No. _____
( ) Cash

Sincerely _____

Procedures and Functions

One sourcecode file is shipped with PowerTable. This file contains several routines and functions which you need to use the enhanced features of PowerTable in your application: PowerTable supports version control and field descriptions.

Version Control Functions

PowerTable supports database format independent version control. These are descriptions of the functions which you need to use this feature.

db_HasVersion ()

Purpose: Determines if a database contains version information or not.

Prototype in: PWRTBL.BAS (in this help file)

Usage: bool = db_hasVersion (db as database, fAdd%)

Parameters: *db* - the database
fAdd - Append flag

Remarks: If the *fAdd* parameter is set to *True*, the function creates a version information table and returns *True* if it was successful.

Returns: *True* if it contains version information, *False* if not

db_GetSetVersion ()

Purpose: Reads or writes version information

Prototype in: PWRTBL.BAS (in this help file)

Usage: db_getSetVersion db, version, versionDate, revision, fSet%)

Parameters: *db* - the database
version - the version of the file (user defined value)
versiondate **-** versions date (user defined value)
revision - database revision number
fSet **-** *True* to write version information, *False* to read them

Remarks: The revision parameter cannot be set, only retrieved. It is updated each time you call this routine to rewrite the version information.

Returns: Nothing

Example:

Updating all version information

```
db_getSetVersion db, 1.00.000, now, 0, True
```

Updating the revision number only

```
db_getSetVersion db, Null, Null, 0, True
```

Getting version information

```
db_getSetVersion db, ver$, verDate$, revision&, False
```

Description Functions

PowerTable supports database format independent table and field descriptions. These are descriptions of the functions which you need to use this feature.

db_GetDes ()

Purpose: Returns the description of a field

Prototype in: PWRTBL.BAS (in this help file)

Usage: description$ = db_getDes(db, tablename$, fieldname$)

Parameters: *db* - database
tablename - name of the table
fieldname - name of the field. If this parameter has the value (table), the table description will be returned.

Returns: String with field description

db_SetDes ()

Purpose: Sets the description of a field

Prototype in: PWRTBL.BAS (in this help file)

Usage: db_setDes db, tablename$, fieldname$, description$, tag$

Parameters: *db* - database
tablename - name of the table
fieldname - name of the field
description - description for the field
tag **-** additional Information

Returns: Nothing.

db_HasDes ()

Purpose: Returns *True* if the database contains field-descriptions, *False* if not.

Prototype in: PWRTBL.BAS (in this help file)

Usage: bool = db_hasDes (db, fAdd%)

Parameters: *db* - the database
fAdd **-** If this Parameter is *True*, the function will add a description table to the database.

Returns: *True* if the database contains field descriptions, *False* if not.

db_deleteTable ()

Purpose: Deletes a table and its field descriptions

Prototype in: PWRTBL.BAS (in this help file)

Usage: db_deleteTable db, tablename$

Parameters: *db* **-** the database
tablename$ - the table to delete

Returns: Nothing.

Other Functions

db_AddIndex()

Purpose: Function to add a new index to an openeded database

Prototype in: PWRTBL.BAS (in this help file)

Syntax: bool = db_AddIndex% (db As Database, tablename$, ixn$, ixf$, ixu%, ixp%)

Parameters: *db* - the database object variable
tablename - the name of the sourcetable
ixn - the name for the new index
ixf - list of fields for the new index
ixu - *True* if it´s a unique index, *False* if not
ixp - *True* if it´s a primary index, *False* if not

Returns: *True* if index was successfully generated.
False if an error has occured.

Example:

```
dim db as database, tb as table
set db=opendatabase ("MYDATA.MDB")
set tb=db.opentable("MYTABLE")
if db_AddIndex ( db, tb, "NewIndex","+Field1;+Field2",False,False ) then
    msgBox "Ok"
else
    msgBox "Error"
end if
```

dbf_CompactDatabase()

Purpose: Compacts an entire xBase database

Prototype in: PWRTBL.BAS (in this help file)

Syntax: bool = dbf_CompactDatabase (dbName$, dbConnect$)

Parameters: *dbName* - the name/directory of the database
dbConnect - the connect-property, either "dbase iv;" or "foxpro 2.5;"

Returns: *True* if successful, *False* if not

Remarks: The database will be opened for exclusive usage.

Example:

```
tf = dbf_CompactDataBase ("C:\MYPROJ","dbase iv;")
```

dbf_CompactTable()

Purpose: Compacts a single xBase table

Prototype in: PWRTBL.BAS (in this help file)

Syntax: bool = dbf_CompactTable (dBName$, dBConnect$, tablename$)

Parameters: *dbName* - the name/directory of the database
dbConnect - the connect-property, either "dbase iv;" or "foxpro 2.5;"
tableName - Name of the table

Returns: *True* if successful, *False* if not

Remarks: The database will be opened for exclusive usage.

Example:

```
tf = dbf_CompactTable ( "C:\MYPROJ","foxpro 2.5;","CUSTOM" )
```

ini_InitDataAccess()

Purpose: Initializes VBRUN300.DLL for access to external databases

Prototype in: PWRTBL.BAS (in this help file)

Syntax: ini_initDataAccess

Parameters: (none)

Returns: Nothing

Remarks: This routine creates or corrects an .INI file automatically, if you want to use external tables in your program. The name of the created .INI-file is the name of your application and it´s stored into your applications directory.

Call this routine at the entry of your program, and the message: "Installable ISAM not found!" is history.

Example:

```
sub main
        ini_initDataAccess
        .
        mdiForm.show 1
        .
        .
end sub

      - or -

sub frmMain_load ()
        ini_initDataAccess
        .
        .
        .
end sub
```

ini_do()

Purpose: Reads or writes a profile value from/to your private .INI file

Prototype in: PWRTBL.BAS (in this help file)

Syntax: ini_do section$, key$, value, fRW%

Parameters: *section* - the section where the string is located. If section is "", the previous name is *used again.*

key - the key name
value - the returned or written value
fRW - *True* to write the value, *False* to read it

Returns: value, if fRW was *False*

Remarks: This function does not require a filename. The name of the .INI-file is the

name of your application and it´s stored into your applications directory

Example:

```
ini_do "Data-Access","Username",uname,False
msgBox "Your Name is " & uname
```

This sequence would read a section like that:

```
[Data-Access]
Username=Mr.Big
```

SourceCode of PWRTBL.BAS

```
Option Explicit
Const TFD = "_tfd"
Const VER = "_version"

Declare Function WritePrivateProfileString Lib "Kernel" (ByVal lpApplicationName As String, ByVal
lpKeyName As String, ByVal lpString As String, ByVal lplFileName As String) As Integer
Declare Function GetPrivateProfileString Lib "Kernel" (ByVal lpApplicationName As String, ByVal
lpKeyName As String, ByVal lpDefault As String, ByVal lpReturnedString As String, ByVal nSize As
Integer, ByVal lpFileName As String) As Integer
Declare Function GetWindowsDirectory Lib "Kernel" (ByVal p$, ByVal s%) As Integer
Declare Function GetSystemDirectory Lib "kernel" (ByVal p$, ByVal s%) As Integer

Const DB_BOOLEAN = 1
Const DB_BYTE = 2
Const DB_INTEGER = 3
Const DB_LONG = 4
Const DB_CURRENCY = 5
Const DB_SINGLE = 6
Const DB_DOUBLE = 7
Const DB_DATE = 8
Const DB_TEXT = 10
Const DB_LONGBINARY = 11
Const DB_MEMO = 12

Dim privateIniFile$, intLine$
Dim retval As String * 255
dim fieldtag$

Sub db_addField (tbl$, td As TableDefs, tn As TableDef, fName$, fTyp%, fSize%, fCnt%)
    Dim fld As New Field
    On Error Resume Next
    Err = False
    If fName = "" Then Exit Sub
    fld.Name = fName
    fld.Type = fTyp
    If fTyp = 10 Then fld.Size = fSize
    If Err Then MsgBox Error$, 48
    Err = False
    fld.Attributes = IIf(fCnt, 16 + 32, 32)
    If Err Then MsgBox Error$, 48
    If td Is Nothing Then
        tn.Fields.Append fld
    Else
        td(tbl).Fields.Append fld
    End If
    If Err Then MsgBox Error$, 48

End Sub


Function db_addindex% (db As Database, tablename$, ixn$, ixf$, ixu%, ixp%)
    Dim nindx As New Index
    On Error Resume Next
    Err = False
    If Len(ixn) = 0 Or Len(ixf) = 0 Then
        db_addindex = True
        Exit Function
    End If
    nindx.Name = ixn
```

```
    nindx.Fields = ixf
    nindx.Primary = ixp
    nindx.Unique = ixu

    db.TableDefs(tablename).Indexes.Append nindx

    If Err Then
        MsgBox Error$ & Chr$(13) & "Table: " & tablename & Chr$(13) & "Index: " & ixn, 48
    Else
        db_addindex = True
    End If
End Function

Function db_getDes$ (db As Database, tablename$, fieldname$)
    Static t As table, db2 As Database
    fieldTag = ""
    On Local Error Resume Next
    If Not db Is db2 Then
        Set db2 = db
        Set t = db2.OpenTable(TFD)
        t.Index = "TF"
    End If
    If t Is Nothing Then Exit Function

    t.Seek "=", LCase$(tablename), LCase(fieldname)
    If t.NoMatch = False Then
        If Not IsNull(t("Text")) Then
            db_getDes = t("Text")
            fieldTag = t("Tag")
        End If
    End If
End Function

Function db_getTag$ ()
    db_getTag = fieldTag
End Function

Sub db_getsetVersion (db As Database, version, versionDate, revision&, fSet%)
    Dim t As table, r&
    On Local Error Resume Next

    Set t = db.OpenTable(VER)
    If t Is Nothing Then Exit Sub

    r = t("Revision")
    If fSet = False Then
        version = t("Version")
        versionDate = t("Date")
        revision = r
    Else
        t.Edit
        If version <> Null Then
            t("Version") = CStr(version)
        End If
        If versionDate <> Null Then
            t("Date") = DateValue(versionDate)
        End If
        t("Revision") = r + 1
        t.Update
    End If
```

```
    t.Close

End Sub


Function db_hasDes% (db As Database, fAdd%)
    On Error Resume Next
    Dim n%
    db_hasDes = False
    n = db.TableDefs(TFD).Fields.Count
    If n Then
        db_hasDes = True
        Exit Function
    End If

    Err = 0
    If fAdd Then
        Dim tn As New TableDef
        tn.Name = TFD
        db_addField TFD, Nothing, tn, "Table", DB_TEXT, 32, 0
        db_addField TFD, Nothing, tn, "Field", DB_TEXT, 32, 0
        db_addField TFD, Nothing, tn, "Text", DB_TEXT, 254, 0
        db_addField TFD, Nothing, tn, "Tag", DB_TEXT, 20, 0
        db.TableDefs.Append tn
        If db_addindex(db, TFD, "TF", "+Table;+Field", 1, 1) Then
        End If
        db_hasDes = True
    End If

End Function


Function db_hasVersion% (db As Database, fAdd%)
    On Error Resume Next
    Dim n%, t As table
    n = db.TableDefs(VER).Fields.Count
    If n Then
        db_hasVersion = True
        Exit Function
    End If

    Err = 0
    If fAdd Then
        Dim tn As New TableDef
        tn.Name = VER
        db_addField VER, Nothing, tn, "Version", DB_TEXT, 10, 0
        db_addField VER, Nothing, tn, "Date", DB_DATE, 8, 0
        db_addField VER, Nothing, tn, "Copyright", DB_TEXT, 64, 0
        db_addField VER, Nothing, tn, "Text", DB_TEXT, 64, 0
        db_addField VER, Nothing, tn, "Tag", DB_TEXT, 10, 0
        db_addField VER, Nothing, tn, "Revision", DB_LONG, 4, 0
        db_addField VER, Nothing, tn, "Memo", DB_MEMO, 4, 0
        db.TableDefs.Append tn

        Set t = db.OpenTable(VER)
        If t Is Nothing Then Exit Function
        t.AddNew
        t("Version") = "1.00.000"
        t("Date") = Now
        t("Revision") = 0
```

```
        t.Update
        db_hasVersion = True
    End If


End Function


Sub db_SetDes (db As Database, tablename$, fieldname$, fieldnamenew$, Text$, tag$)
    Static t As table, db2 As Database
    Dim tf$
    On Local Error Resume Next
    If Not db Is db2 Then
        Set db2 = db
        Set t = db2.OpenTable(TFD)
        t.Index = "TF"
    End If
    If t Is Nothing Then Exit Sub

    t.Seek "=", LCase$(tablename), LCase$(fieldname)
    If t.NoMatch Then
        t.AddNew
    Else
        t.Edit
    End If
    t("Table") = LCase$(tablename)
    t("Field") = LCase$(fieldnamenew)
    t("Text") = Text
    t("Tag") = tag
    t.Update

End Sub

Function dbf_compactDatabase% (dbName$, dbc$)
    Dim db As Database
    Dim tn$, i%

    On Error Resume Next
    Err = False

    For i = 0 To 10000
        Err = False
        Set db = OpenDatabase(dbName$, True, False, dbc)
        If Err Then
            MsgBox Error$, 48
            Exit For
        End If
        If i + 1 > db.TableDefs.Count Then Exit For
        tn = db.TableDefs(i).Name
        db.Close
        If dbf_compactTable(dbName, dbc, tn) = False Then Exit Function
    Next
    dbf_compactDatabase = True
End Function

Function dbf_compactTable% (ByVal dbPath$, dbc$, tablename$)
    Dim db As Database, iSuf$, mSuf$
    Dim ox As Indexes, oxc%, i%

    On Error Resume Next
    Err = False
```

```
    Set db = OpenDatabase(dbPath, True, False, dbc)
    If Err Then
        MsgBox Error$, 48
        Exit Function
    End If
    If LCase$(dbc) = "dbase iv;" Then
        iSuf = ".MDX"
        mSuf = ".DBT"
    Else
        iSuf = ".CDX"
        mSuf = ".FPT"
    End If

    If Right$(dbPath, 1) <> "\" Then dbPath = dbPath & "\"
    GoSub dbComp_killTemp

    screen.MousePointer = 11
    db.Execute ("SELECT * INTO temptbl0 from " & tablename)
    If Err Then
        MsgBox Error$, 48
        GoTo dbComp_exit
    End If

    Set ox = db.TableDefs(tablename).Indexes
    oxc = ox.Count - 1
    For i = 0 To oxc
        If db_addindex(db, "temptbl0", CStr(ox(i).Name), CStr(ox(i).Fields), CInt(ox(i).Unique),
CInt(ox(i).Primary)) = False Then
            GoTo dbComp_exit
        End If
    Next
    If Err = False Then
        Kill dbPath & tablename & ".DBF"
        Kill dbPath & tablename & mSuf
        Kill dbPath & tablename & iSuf
        Name dbPath & "temptbl0" & iSuf As dbPath & tablename & iSuf
        Name dbPath & "temptbl0.dbf" As dbPath & tablename & ".DBF"
        Name dbPath & "temptbl0" & mSuf As dbPath & tablename & mSuf
        GoSub dbComp_killTemp
    End If

    dbf_compactTable = True

dbComp_exit:
    db.Close
    screen.MousePointer = 0
    Exit Function

dbComp_killTemp:
    Kill dbPath & "temptbl0.dbf"
    Kill dbPath & "temptbl0" & mSuf
    Kill dbPath & "temptbl0" & iSuf
    Err = False
    Return

End Function


Function getSystemDir$ ()
    Dim winpath As String * 145, l%
    l% = GetSystemDirectory(winpath, 145)
```

```
    If l Then
        If Mid$(winpath, l, 1) <> "\" Then
            Mid$(winpath, l + 1) = "\"
            l = l + 1
        End If
        getSystemDir$ = UCase$(Left$(winpath, l%))
    End If
End Function

Function getWindowsDir$ ()
    Dim winpath As String * 145, l%
    l% = GetWindowsDirectory(winpath, 145)
    If l Then
        If Mid$(winpath, l, 1) <> "\" Then
            Mid$(winpath, l + 1) = "\"
            l = l + 1
        End If
        getWindowsDir$ = UCase$(Left$(winpath, l%))
    End If
End Function


Sub ini_do (section$, key$, v, ByVal flag%)
    Static sec$
    Dim r%
    If Len(privateIniFile$) = 0 Then
        privateIniFile$ = app.Path & "\" & app.EXEName & ".INI"
    End If

    If Len(section) Then sec = section
    If flag% Then
        r% = WritePrivateProfileString(sec, key$, v, privateIniFile$)
    Else
        r% = GetPrivateProfileString(sec, key, "", retval, 255, privateIniFile$)
        If r% = 0 Then
            v = ""
        Else
            v = Left(retval, r%)
        End If
    End If
End Sub


Sub ini_initDataAccess ()
    Dim sysdir$, windir$, result$, old$, v$

    sysdir = getSystemdir()
    windir = getWindowsDir()

    ini_do "Installable ISAMs", "dbase IV", result, False

  ' Write Data-Acess Information to the INI-File
  ' -------------------------------------------

    If len(dir$(sysdir & "msajt200.dll")) Then
        v = "200.dll"
    Else
        v = "110.dll"
    End If
```

```
        ' Installable ISAMs
          ini_do "", "Btrieve", sysdir & "btrv" & v, True
          ini_do "", "Foxpro 2.0", sysdir & "xbs" & v, True
          ini_do "", "Foxpro 2.5", sysdir & "xbs" & v, True
          ini_do "", "dbase III", sysdir & "xbs" & v, True
          ini_do "", "dbase IV", sysdir & "xbs" & v, True
          ini_do "", "Paradox 3.X", sysdir & "pdx" & v, True

        ' Installable ISAMs - options
          ini_do "ISAM", "PageTimeOut", 5, True
          ini_do "", "MaxBufferSize", 128, True
          ini_do "", "LockRetry", 20, True
          ini_do "", "CommitLockRetry", 20, True
          ini_do "", "ReadAheadPages", 16, True

        ' dBase ISAM
          ini_do "dBASE ISAM", "Deleted", "ON", True
          ini_do "", "PageTimeOut", 600, True
          ini_do "", "CollatingSequence", "International", True

        ' Paradox ISAM
          ini_do "Paradox ISAM", "CollatingSequence", "International", True
          ini_do "", "PageTimeOut", 600, True

        ' Btrieve ISAM
          ini_do "Btrieve ISAM", "PageTimeOut", 600, True

          old = privateIniFile
          privateIniFile = windir & "WIN.INI"
          ini_do "BTRIEVE", "Options", "/m:64 /p:4096 /b:16 /f:20 /l:40 /n:12 /t:" & windir &
"BTRIEVE.TRN", True

          privateIniFile = old


      SetDataAccessOption 1, privateIniFile

End Sub
```

Other Products

The following list is dated on 29 September 1995

Lisa	Easy to use ISAM index manager for PowerBasic. Useful for huge and small database projects. Very highspeed and very small.	DOS	German, English	1.20	DM 59,-- US$ 39,-- (SWREG #4592)
PowerTools I, II	Tools for PowerBasic. Menus, dialogs, windows, mouse, system, control funcions, etc. The most successful PowerBasic toolbox.	DOS	German	2.00	DM 89,--
PowerTools I, II Sourcecodes	Sourcecodes for PowerTools for PowerBasic.	DOS	German	2.00	DM 80,--
ThreedPak´95 (FormEx-Ctl3d/VBX)	3D power package for Visual Basic 3.0/4.0 under Windows 3.1 and Windows 95. Independent managing of 3D effects. Supports all standard controls and all custom controls. All styles are configurable. Several extras. Includes other controls such as: button/toolbar, tab control, panel control, property control, FormEx control. replaces up to 10 custom controls.	Win 3.1, Win 95	German, English	2.00	DM 69,-- US$ 49,-- (SWREG #4543)
SmartHelp One	Tool to create Windows help with MS Word. For all Windows programming languages. Generates header files for Visual Basic, C/C++ and PowerBasic automatically. Moreover it manages VB projects independently and adapts help context ID properties of all Forms of a project automatically It supports format templates, and that way an easy creation of help files and manuals at the same time.	Win 3.1, Win 95	German, English	1.01	DM 49,-- US$ 35,-- (SWREG #4490)
PowerTable	Utility to manage/change easily databases and their structure. It supports following formats: Access, dBase IV, FoxPro, Paradox und Btrieve and shortly SQL Server and other ODBC formats. Converting and copying tables via drag & drop,	Win 3.1, Win 95	English	2.00	DM 79,-- US$ 59,-- (SWREG #7790)

undo functions, field description, version control and more.

(All prices plus delivery costs and the value-added. Delivery only after payment by cheque or cash. For foreign orders, only the price in dollars has to be observed)