Database Management System
DDE Server Engine
VBDB
version 1.10

COPYRIGHT NOTICE(S)

ABOUT VBDB

VBDB is a database engine which operates as a *DDE server*. This means that any windows application which supports DDE can use VBDB. Visual Basic supports DDE, and in fact VBDB was written entirely in Visual Basic making use of Windows system level calls via the windows API functions.

VBDB is a suite of programs, forms and routines that allow the Visual Basic programmer access to the industry standard database file format used by dBASE. Using VBDB you may create, edit, modify, access or manage database files, indexes and memo files -- all from within your Visual Basic program.

To use VBDB simply add the DBACCESS.BAS module and DBACCESS.FRM to your program and your off! Include other forms from this library as needed to add features to your program like Browse, Display Structure and others. It's really that easy.

The following caveat applies to these routines: This version supports dBASE III, III+ and IV -- as long as NO specific functions of dBASE IV are used (like the floating point record type, for example).

VBDB processes number as strings. Make your string a number using the Visual Basic Str$() function BEFORE processing it, and convert it back into a number using the Visual Basic Val() function.

VBDB supports stand-alone or multi-user (Local Area Network) functions as well as multiple

open databases. Multiple programs can simultaneously access the same or different files.

The DDE server engine is distributed as a compiled program. VBDB.EXE may be distributed with your application and has the same license and restriction requirements as VBRUN100.DLL from Microsoft. And in fact, VBDB, when purchased, comes with VBRUN100.DLL, as VBDB requires VBRUN100.DLL to operate.

SPECIFICATIONS

VBDB uses the dBASE III+ format for all database files, index files and memo files. VBDB is compatible with dBASE IV& as no version specific functions of dBASE IV are used.

Field Types	Character	Numeric	Logical	Date	Memo
Max length	255	19	1	8 fixed	10 fixed

Max records	: 2,147,483,647
Max record size	: 4,096 bytes
Max number fields/record	: 128
Field name size	: 1 to 10 bytes
Max key length	: 100 bytes
Max memo size	: 32K

PROGRAMS & FORMS LICENSED FROM

Marquis Computing
135 Chestnut St.,
Bridgewater NJ 08807
CompuServe 76120,2413
Phone (201) 707-1316

Proudly written in pure BASIC by Hank Marquis.

WARRANTY

Marquis Computing makes no warranty of merchantability or fitness for a particular purpose. Marquis Computing does not guarantee that this product will operate error free or without any loss of data. **MARQUIS COMPUTING MAKES NO WARRANTIES EXPRESSED OR IMPLIED OF ANY SORT REGARDING THIS PRODUCT.**
MARQUIS COMPUTING IS NOT LIABLE FOR ANY DAMAGES RESULTING FROM THE USE OR MISUSE OF THESE ROUTINES IN ANY FASHION.

GUARANTEE OF SATISFACTION

If you have paid for this library, and are unsatisfied with it, delete all copies of all routines in this library from any disks you may have, return all code along with all copies,

documentation and receipt(s) to Marquis Computing for a full and cheerful refund.

YOU HEREBY AGREE THAT

1) You will not sell or distribute this source code as raw source code or object code. You will not distribute forms as un-compiled forms. That you will in no way distribute these routines in any manner other than compiled executable programs.

2) You may only sell executable programs which contain compiled routines which use these routines.

3) You can make as many copies of this library as you want, for any purpose which does not conflict with paragraph 1 or 2 above.

4) Copies may be made for distribution to BBS's and other shareware distribution channels providing that all files in this package are also distributed and Marquis Computing is so notified.

ABOUT MARQUIS COMPUTING

If you find these routines useful, helpful and in general worthwhile; and plan to use them in your programs or for other purposes, I would appreciate a contribution of $50.00 -- as you see fit and your means permit.

When you register VBDB you will receive phone & mail support as well as updates and bug fixes.

Your patronage allows me to create other such libraries. Thanks for taking the time to read this documentation.

Sincerely,
Hank Marquis
Marquis Computing

MAKING MODIFICATIONS TO VBDB

There are many subs and functions included inVBVB. Most are for your direct use. The general purpose and high-level routine are made for you to use with little or no understanding of how they work -- if you don't care to study the code, you can still use them.

Others routines are not really meant to be called by you or your programs directly even though you may indeed call them. Some are pretty special and you probably won't need to call them. However, other VBDB routines call them often in the pursuit of their operations so DO NOT modify anything that says do not modify -- unless you REALLY know what you are doing!

All are commented so you can see how they work though. In general, the whole point of doing this kind of thing in Visual Basic is so that you ARE able to make changes and study the code and all that. Just be careful, and keep a copy of these routines stored away incase you make a mistake and mess-up.

REVISION & VERSION HISTORY

VBDB.DOC, Microsoft Write format, version 1.0, update to documentation 7/21/91.
VBDB.EXE, Windows 3.0 DDE Server, version 1.00; re-written from QB4.5 source code.
VBDB.EXE, re coded to use Windows API calls to read/write files; allowing up to 255 open files per instance of VBDB.EXE, version 1.10.

Please note that this version of VBDB, 1.10, is incomplete and under development. As such all functions relating to index and memo file operations are not finished.

USING VBDB

Using VBDB is very easy -- in fact, all you need to do to use VBDB is to include the DBACCESS.FRM and DBACCESS.BAS into your application. Then, make sure that the VBDB.EXE database server engine is in a sub-directory which is in your path statement.

VBDB.EXE must be loaded onto your hardisk, under a directory which is in your path statement, or the default directory from which a client application will run.

Then, simply use the DBACCESS.BAS routines. In general, your program will operate as follows:

```
....your code here....

        Status = LogonServer ()       'access the VBDB DBF server engine

....more code here....

        CALL OpenDataBase (FileName$, Handle, Mode, Status)  'open up a database

....more code here

        CALL GetRecord (Handle, Record&, RecData$, Status)  'get a record

....more code here....

        CALL CloseDataBase (FileName$, Mode, Status) 'close the database

        Status = LogoffServer ()      'turn off the DDE link to the server

        END
```

Only one call to LogonServer is needed per program occurance. Calling LogoffServer will unload the VBDB.EXE program from the windows environment, if there are no other applications using it.

NOTE:
Use Handle as the file indicator for all future file accesses. You cannot use Handle with Visual Basic file operations! VBDB does not use Visual Basic file processes -- instead VBDB uses Windows API calls for all file operations. Thus, VBDB supports many more that 15 files (VB limit). The actual number is less than 255, defaults to 20, and may be set by use of the SetNoHandles call.

Two optional environment variables allow you to customize the amount of memory VBDB uses as well as tailor VBDB to your application.

DBCLIENTS determines the number of client applications that VBDB will support. The default is 2.

DBDBFS determines the maximum number of Windows file handles that that VBDB will allow. This in turn limits the number of open files. The fewer open files, the less memory needed by VBDB. The default is 10. Do not set less than 6-10 or VBDB will fail as several (as many as 6) Windows handles are used by Windows and/or VBDB itself.

To use these optional variables, include a line in your autoexec.bat file similar to the following. The following allow 10 client applications and limits the number of databas handles to 10.

```
Set DBDBFS=10
Set DBCLIENTS=2
```

ROUTINES CONTAINED IN VBDB

The following routines comprise VBDB. The are composed of general purpose routines as well as those specific to database manipulation. Below they are grouped according to functionality. In general, VBDB contains low-level and high-level routines. Low-level routines perform a specific very defined function, for example, determining the size of a database header. High-level routine combine several low-level operations into a functional element, for example, opening a database file.

Several "intelligent forms" are also included. These forms provide a function, for example, getting user input to create a database definition. These forms are typically self-supporting and do not use other programs or forms. Some, however, do make use other forms or sub routines.

Following this listing, each function or sub-routine is documented. Given is the routine name, arguments, VBDB program module (.BAS, .FRM, .GBL or .MAK) and a description.

In all cases, the routine itself is commented and you are urged to look there for more detailed information and other cautions or notes.

DDE SERVER MANAGEMENT

This section describes the various functions provided for management of the DDE link between a client and a server using VBDB.

LogonServer (Status)

Establishes a DDE hot-link to the database server engine. If the server is not running under windows, it will be run. Returns TRUE (-1) is log on was successful, or FALSE (0) if unsuccessful.

LogoffServer ()

FUNCTION, closes a DDE hot-link which had been setup via the LogonServer call. It will close VBDB.EXE if there are no other applications using it. Returns TRUE (-1) is log off was successful, or FALSE (0) if unsuccessful.

DBAccess (CmdStr$)

SUB, Actual routine which send/recieves data via the DDE links. CmdStr$ is the command line the DDE server is to execute.

DBALinkUp ()

FUNCTION, Returns TRUE (-1) if the DDE server is on-line, or FALSE (0) if the server is off-line.

DBALoaded ()

FUNCTION, Returns TRUE (-1) if the VBDB server is loaded and running, or FALSE (0) if not.

DATABASE FILE USAGE (DBF)

This section describes the various functions provided for database functions of DBACCESS module of VBDB. All calls are designed to be compatible with popular BASIC database management software calling syntax.

Sub OpenDBF (Handle, Status, FileName$, dbftype, Mode)

Opens FileName$ as a database. Returns *Handle* as the file handle. Use *Handle* for all subsequent calls to perform operations on this database. dbftype% is a dummy paramater provided for syntax compatibility with popular database libraries. *Mode* controls how the database is opened. Valid options for *Mode* include:

0/4 = normal open - read/write
1/5 = create a new database if it does not already exist
2/6 = create a new database erasing the old if it already exists
3/7 = not implemented, invalid function.

Status is returned as the error condition where 0 (FALSE) means no error has occured. You can assume that the call was successful. *Status* <> 0 means the call experienced some error. The value of Status equates to an error; see the Error Messages section for a listing of VBDB errors.

NOTE: VBDB inherently supports multiple users with or without LAN software. All VBDB modes are direct-write-to-disk mode with no buffering.

Sub CloseDBF (Handle, Status, Mode)

Closes FileName$, flushes all buffers to disk. *Mode* controls the type of close where:

0 = normal close, update header
1 = close with no header update

See OpenDatabase for *Status* values.

Sub CreateDBF (DbfName$, Handle%, Fld$(), Mode%, Status%)

Creates and opens new a database file named *DBFName$*. *Handle%* is the operating system handle for this file. *Mode%* controls the method used for creation where *Mode%*=1 will over-write an existing database of the same name or *Mode%*=0 will abort if a file exists with the same name. *Flds$()* is a multi-demensional array that contains field definitions. ***I strongly urge you to use the DefineDataBase form to develop the Flds$() array***. However, you can create this array yourself. It has the following characteristics:

Flds$(0,0) = total number of fields in this array

Flds$(n,1) = decimal flag (-1 or 0) which indicates if this field has decmials
Flds$(n,2) = field length
Flds$(n,3) = field type (String * 1) which is C,M,N,D or L
Flds$(n,4) = field name (String * 12) which is this fields name

Combines the functions of OpenDBF, DefineSTR, CommitSTR, CloseDBF and OpenDBF into a single call for automated database creation when the Flds$() array is already built. See OpenDatabase for *Status* values.

Sub PutREC (Handle%, Status%, Record&, RecData$)

Adds record Record& containing data Record$ to database file *Handle%*. See OpenDatabase for *Status* values.

Sub PutFLD (Handle%, Status%, FldNum%, FldName$, FldData$, RecData$)

Writes the data in *FldData$* into field *FldNum%* of record *Record$*. The data is not written to disk file until the next call to PutRec. If *FldNum%* and *FldName$* are passed, *FldNum%* has priority. See OpenDatabase for *Status* values.

Sub GetREC (Handle%, Status%, Rec&, RecData$)

Retrieves database record *Record&* from file *Handle%*. *RecData$* contains the database record after the call. See OpenDatabase for *Status* values.

Sub GetFLD (Handle%, Status%, FldNum%, FldName$, FldData$, RecData$)

Retrieves *FldData$* from Field *FldNUm%* from record-string *Record$*. If *FldNum%* and *FldName$* are passed, *FldNum%* has priority. See OpenDatabase for *Status* values.

Sub GetFLDS (Handle%, Status%, NumFlds%, Flds$(), RecNum&)

Returns *NumFlds%*, which is the number of fields in a record, and then parses the record

string into an array. Much faster and easier than using GetREC and then using multiple GetFLD calls. Passed *RecNum&* , a valid record number of open database *Handle%* , returns Flds$() -- which contains all the records field data. Element 1 of Flds$() is field 1, element n is field n etc. See OpenDatabase for *Status* values.

Sub CommitSTR (Handle%, Status%)
Finishes creation of a database by actually writing the header to a database. See OpenDatabase for *Status* values.

Sub DefineSTR (Handle%, FldNum%, FldName$, FldType$, FldLen%, Decimal%)
Used to send information to the DDE server to define a database. It called once for each that is to be in the new database. Handle% is the database file handle returned from an OpenDBF call using mode 2, FldNum% is this fields number, FldName$ is this fields name, FldType$ is a one character string indicating field type (C,N,M,D,L), FldLen% is this fields length, Decimal % indicates the number of decimal points used by this field (if any) and is only valid when FldType$ is "N" for numeric.

NOTE: See CreateDBF for more information.

Sub StatusDBF (Handle,% FileName$, dbftype$, DBTPtr%, NumRecs&, NumFlds%, RecLen%, UpDate$, Status%)
Returns information about database Handle%.

INDEX FILE USAGE (NDX)

This section describes the various functions provided for database functions of DBACCESS module of VBDB.

Sub GetKEY (Handle%, Status%, Key$, Record&, Mode%)
Returns a record number based on the value of Key$. Mode control the find key mode where:

```
 0 = find record based on value of Key$
-1= return the record of the last physical key - does not use Key$
+1 = return the record of the next physical key - does not use Key$
+3 = find next key above Key$
-2 = seek to last key
+2= seek to first key
```

Only Mode 0 is implemented at this time

Sub OpenNDX(Handle%, Status%, NDXName$, NDXType%, NDXMode %, KeyExp$, KeyLen%, KeyType%, Mode%)
Used to open an index file and return information about the index. Mode controls open mode where:

```
0/4=normal open mode
```

CloseNDX(Handle%, Status%)
Closes an open index.

At the present time only GetKey , OpenNDX and CloseNDX are functional. Other Index file functions to be added at a later date

DelKey (Index%, KeyVal$, Record&)
AddKey (Index%, KeyVal$, Record&)

MEMO FILE USAGE (DBT)

This section describes the various functions provided for database functions of DBACCESS module of VBDB.

Index file functions to be added at a later date

OpenDBT
CloseDBT
GetMEMO
PutMEMO

FORMS & USAGE

This section describes the various forms found in VBDB and the function they provide. It does not (always) decribe the sub-routines contained within a form. In general, main sub routines used in/by a form are defined, general purpose routines are not.

DEFINEDB.FRM, DefineDataBase -- controls database creation & validation
DISPDBIN.FRM, DispDBInfo -- displays a DBFs structure
GFILEBOX.FRM, GetFileBox -- a general purpose file name retreival program
BROWSER.FRM, Browse -- Record browsing sub routine
DBACCESS.FRM, DBA -- the DDE link transfer objects DO NOT MODIFY!
DEMOVBDB.FRM, Demo program form main menu
INDEXDEM.FRM, BrowseNDX -- form to allow demonstration of the GetKEY routine

ERROR MESSAGES AND CONDITIONS

VBDB has a built-in error handler. When a call fails, or executes sucessfully, an error status variable is returned. This variable is typically named *Status%.* When *Status%* is non-zero, an error has occured and the value is the error number that occured. VBDB traps DOS, Windows, Visual Basic and VBDB errors. Only VBDB errors are presented below. Other errors (i.e., Visual Basic errors) are covered in the respective manuals. When *Status%* is 0 (FALSE) then no error occured and the call executed sucessfully.

Below are the VBDB specific error codes returned in *Status%*.

100	can't close file
101	can't open file
102	file seek failure
103	file read failure
104	file write failure
105	can't create file
106	can't delete file
3000	end of database file
3001	invalid database size
3002	file is not a valid DBF file
3003	invalid record size
3004	field not found
3005	invalid field name
3006	invalid field size
3007	duplicate field name
3008	field length too long
3009	invalid field type
3010	field name too long
3011	can't close database
3100	key not found in index