

# Misc. Issues

---

## Table of Contents

Misc. Issues.....	1
Coding FAQ.....	2
Form/Control.....	2
When I create a form and I put , for example, a label control on it, how can I access to properties of this control?.....	2
SQL.....	2
How much SQL do I need to know to use it?.....	3
The basic concept of SQL support in KBasic.....	3
Table Designer.....	3
Table Editor.....	4
Relations.....	4
Query By Example.....	4
Form.....	4
Report.....	4
Using SQL in forms.....	4
Controls and SQL.....	4
SQLName.....	4
SQLRelation.....	5
Form.....	5
Other Control.....	5
Primary Key.....	6
Using SQL for filling data without binding.....	6
Hidden SQL Operating.....	6
Import/Export.....	6
SQLite (Embedded).....	6
Additional information.....	6
DLL.....	6
Example about using the native color dialog of Windows.....	7
Playing MIDI music files on Windows.....	8
Playing WAVE sound files on Windows.....	8
IDE.....	9
Naming Forms and other objects.....	9
Mac.....	9
Creating custom menubars on Mac.....	9
Structure of a menubar file of KBasic.....	9
Menubar properties.....	10
Menus + properties.....	10
ParentControl As String.....	10
ParentIndex As Integer.....	10
Child menus.....	10
Menuitems.....	11
Caption As String.....	11
Icon As String.....	11
Key As String.....	11
Enabled As Boolean.....	11

Checked As Boolean (not supported yet).....	11
Separator As Boolean.....	11
StatusTip As String.....	11
Tag As String.....	11
Menubar code.....	11
Example.....	12

# Coding FAQ

---

## Form/Control

When I create a form and I put , for example, a label control on it, how can I access to properties of this control?

There are several examples showing how to access controls. In general, it is done like in VB6.

You can access the control by using its name, e.g. you have got a Label named "Label0".

Use "Label0" to access this control (in fact it is declared as a variable of type Label by KBasic automatically: Dim Label0 As Label)

Create a new project and a new form, add a Label to it (its name must be Label0, be sure that is correctly set) and add the following code to the form.

```
Sub Form_OnOpen()
    Label0.Caption = "Hello World!"
End Sub
```

Run the project. The caption of the Label will change at runtime.

# SQL

---

**This is an overview about the possibility to use SQL databases.**

The Qt documentation in C++ of SQL can be read here: <http://doc.trolltech.com/4.3/qtsql.html>

You do not need to have a running database server: If you have selected "Embedded (SQLite)" in the database connection dialog, the builtin SQLite database, no setup is needed for this database.

The following databases are (theoretically) supported:

- **QPSQL = PostgreSQL Driver**
- QMYSQL = MySQL Driver
- QODBC = ODBC Driver (includes Microsoft SQL Server)
- QDB2 = IBM DB2
- QIBASE = Borland InterBase Driver
- QOCI = Oracle Call Interface Driver
- QTDS = Sybase Adaptive Server

If the database driver is available, KBasic will show it in the database connection dialog. Which database driver is present, depends on the platform you are using, e.g. QPSQL is currently available on all platforms.

**Note: You can use much more databases, like MS Access or MS SQL Server, by using the ODBC interface driver.**

Currently, if you would like to use MySQL on Windows, you have to use the QODBC driver. The use of PostgreSQL is recommended, because it is free to use and available on all platforms. It is KBasic's test database as well.

## How much SQL do I need to know to use it?

---

No much knowledge about SQL is needed, because KBasic does all the hard parts for you automatically, e.g. it manages for you the records of a database table providing common functionality for your applications like editing, deleting or adding new data. Nevertheless it is possible to use SQL commands like in many other programming languages to gain low-level access to your database.

The process of creating and using your own custom database is described by the following steps:

- First you decide, which tables you would like to use in your database. Therefore, you use the database administration tool, which comes with your database and create a new database, user password and other configurations you might need. After that you add the needed tables to this database and set up the configuration in KBasic's database connection dialog, either in KBasic (using Import CSV or creating plain tables) or in the administration tool. Now, you may be able to see the database tables inside KBasic by selecting 'Query By Example'.
- You may change the database structure like adding new fields to your tables, dropping tables or adding indexes to your tables by using the administration tool, which comes with your database software, but this is possible in KBasic as well.
- You may tell the relationship of your tables and their fields to KBasic in the 'Relations' dialog. This is only needed, if you would like to use the 'Query By Example' dialog or KBasic's table editor. If relations is set up correctly, the query by example or table editor tool will consider the information when building the sql statements for you.
- It is not needed either to use 'Relations' nor 'Query By Example' most database work will be fine without them.

**If you use the administration tool of your database to create your tables you need to know how to create a primary key, which is named 'kbreccordid' all lowercase! and set as 'int' datatype. KBasic and it is not declared KBasic's SQL feature will NOT function.**

## The basic concept of SQL support in KBasic

---

**You may change the database structure like adding new fields to your tables, dropping tables or adding indexes using the administration tool, which comes with your database software, though it is possible to manage the database structure directly in KBasic (recommended).**

Every [Form](#) is able to represent a database table, which can be changed and accessed in many ways. Normally, you define the table you would like to use in your form only by setting the right table name of [SQLName](#) property of your form. After that you decide, which of the fields of that table you would like to have in your form visible and accessible to your user. You may do this, by adding a new [Control](#) for each field of that database table and connect it to the database table field by setting the [SQLName](#) to the name of the field you have declared in your database administration tool. By using the Form Wizard, KBasic does this job for you.

### Table Designer

This is only needed, if you would like to manage your database tables from within KBasic.

Name of fields must be written as it would be a variable in KBasic. It is recommended to write all fields lowercase for SQL compatibility. Spaces are not allowed.

## Table Editor

This is only needed, if you would like to change the data of database tables from within KBasic.

## Relations

This is only needed, if you would like to use the 'Query By Example' dialog. If relations is set up correctly, the query by example or table edit tool will consider the information when building the sql statements for you.

## Query By Example

This tool is provided that you can visually design your sql queries without knowing sql syntax. It is possible to save queries in your project.

## Form

The forms do all the hard sql support work for you. You create a [Form](#), adds information about the sql table to use, and add some controls. Every control need to know to which database table field it is pointing to. You decide, which actions the user of your application might do like allowing changing existing records or deleting. That's it. Your database application is ready to run!

## Report

The reports are useful, when you would like to print your database records.

## Using SQL in forms

---

- **SQLName** - contains the SQL statement to retrieve the data for the form or simple the table name. Currently, only single table selection is supported, which is sufficient for most circumstances. Write the name of the table you would like to use or use a select statement, e.g. "SELECT \* FROM myTable"

### Example

This is for a form: Selecting all fields from table sql in the order by the field values of textbox.

```
SELECT * FROM sql ORDER BY textbox
```

- **SQLInsert** - If this value is set to false. SQL inserts are not allowed.
- **SQLUpdate** - If this value is set to false. SQL updates are not allowed.
- **SQLDelete** - If this value is set to false. SQL deletes are not allowed.

## Controls and SQL

---

### SQLName

- **SQLName** - contains simple the field name. Currently, only the selection of one table is supported.

## Example

This is for a textbox control. The sql table, to which the form of textbox controls points to, contains a database field 'name':

```
name
```

See property [SQLName](#) of Control for more information.

## SQLRelation

It is possible to use 1:n or m:n relations.

### Form

#### m:n relation

It is possible to embed a form into another form using the control ChildControl. By so, you may connect the two sql statements of that form by setting SQLRelation of the child form. You may access the current record of the parent form by using the \$ and the field name like \$kbreccordid. The SQLRelation contains a valid SQL select statement, which returns a list of id values used to read the records for the child form.

Examples:

Parent form: **SQLName**

```
SELECT * FROM movie
```

Child form: **SQLName**

```
SELECT * FROM actor
```

Child form: **SQLRelation**

```
SELECT actorid FROM actor_movie WHERE movieid=$kbreccordid
```

After each change of the current record of the parent control, the SQLRelation sql statement of the child form is executed. This returns the id values used as kbreccordid values within the SQLName select statement of the child form. Actually, a WHERE clause is added to the end with those id values and the query is refreshed and a new set of records is returned.

### Other Control

#### 1:n relation

Take the value from SQLName and search it in SQLRelation in the field kbreccordid of the table city and return name and code sorted by name.

```
SELECT kbreccordid, name, code FROM city ORDER BY name
```

First you write kbreccordid (which is the foreign key) and database field you would like to show instead. The information is displayed using [SQLName](#) and [SQLRelation](#) combined.

This is useful, if you would like to show a more humanreadably text instead of a foreign key.

If you do not write kbreccordid as first selected field, KBasic will automatically do it for you.

See property [SQLRelation](#) of Control for more information.

## Primary Key

One TextBox control must be set to “kbrecordid”, otherwise the auto-sql handling of KBasic will fail. If SQLName of one control is “kbrecordid”, the value will be auto-incremented by KBasic and used as primary key. It is recommended to set this control as hidden by ‘Visible = False’.

All of your tables must have kbrecordid as of type int and set as primary key, otherwise the auto sql handling of KBasic will fail.

## Using SQL for filling data without binding

---

- SQL - contains the SQL statement to retrieve the data for the control.

See property [SQL](#) of Control for more information.

## Hidden SQL Operating

---

- If you need to change sql records without notice for the user, one way would be to create a form with controls and set the SQLName’s of the controls. Open this form using [OpenHidden](#) and use it like you would use a visible form with sql controls and bindings.
- Another way would be directly creating the needed sql statements.
- Third possibility is the use of the class [Query](#).

## Import/Export

---

It is possible to import or export database tables provided as comma separated values (CSV files). The separator might be ',' or ';' and the first line may contain the names of the fields.

## SQLite (Embedded)

---

KBasic supports SQLite, so that you do not need to setup a database server. It uses SQLite as embedded database engine and you may create database files locally on your machine.

For more information, see <http://www.sqlite.org/>

## Additional information

---

- Always name your fields lowercase

## DLL

---

**This is an overview about the possibility to use the WinAPI or any DLL files inside your KBasic code on Windows.**

In KBasic you may use the syntax of VB6 to declare DLL calls, but it is strongly recommend to use the new syntax, because it makes it easier to work with and helps you to get organized your code.

Before you can use any DLL file you must tell KBasic, which DLL file you would like to use and which functions are available through the DLL file, after that you can call that routine as you would call any KBasic function.

Warning! If you use predeclared DECLARE statements of VB6, be aware that the size of the datatypes differs between VB6 and KBasic. You have to change it.

## Example about using the native color dialog of Windows.

```
' DLL USING (new style)
' Warning! If you use predeclared DECLARE statements of VB6, be aware
' that the size of the datatypes differs between VB6 and KBasic,
' namely Long in VB6 must be Integer in KBasic! You have to change it.

Class comdlg32 Alias Lib "comdlg32.dll"

    Static Function ChooseColor_Dlg Alias "ChooseColorA" _
        (lpcc As CHOOSECOLOR_TYPE) As Integer

    Type CHOOSECOLOR_TYPE
        lStructSize As Integer
        hwndOwner As Integer
        hInstance As Integer
        rgbResult As Integer
        lpCustColors As Integer
        flags As Integer
        lCustData As Integer
        lpfnHook As Integer
        lpTemplateName As String
    End Type

    ' Anwender kann alle Farben wählen
    Const CC_ANYCOLOR = &H100
    ' Nachrichten können "abgefangen" werden
    Const CC_ENABLEHOOK = &H10
    ' Dialogbox Template
    Const CC_ENABLETEMPLATE = &H20
    ' Benutzt Template, ignoriert aber den Template-Namen
    Const CC_ENABLETEMPLATEHANDLE = &H40
    ' Vollauswahl aller Farben anzeigen
    Const CC_FULLOPEN = &H2
    ' Deaktiviert den Button zum Öffnen der Dialogbox-Erweiterung
    Const CC_PREVENTFULLOPEN = &H4
    ' Vorgabe einer Standard-Farbe
    Const CC_RGBINIT = &H1
    ' Hilfe-Button anzeigen
    Const CC_SHOWHELP = &H8
    ' nur Grundfarben auswählbar
    Const CC_SOLIDCOLOR = &H80

End Class

Dim CC_T As comdlg32.CHOOSECOLOR_TYPE, Retval As Integer
Dim BDF(16) As Integer

'Dim k As String
'CC_T.lpTemplateName = AddressOf(k)

'CC_T.lpTemplateName = "fdgfg"
'Print CC_T.lpTemplateName

'Einige Farben vordefinieren (Benutzerdefinierte Farben)
BDF(0) = RGB(255, 255, 255)
BDF(1) = RGB(125, 125, 125)
BDF(2) = RGB(90, 90, 90)

'Print Len(CC_T) 'Strukturgröße
With CC_T
    .lStructSize = Len(CC_T) 'Strukturgröße
    .hInstance = 0 'App.hInstance 'Anwendungs-Instanz
    .hwndOwner = 0 'Me.hWnd 'Fenster-Handle
    .flags = comdlg32.CC_RGBINIT Or comdlg32.CC_ANYCOLOR Or comdlg32.CC_FULLOPEN Or
comdlg32.CC_PREVENTFULLOPEN 'Flags
    .rgbResult = RGB(0, 255, 0) 'Farbe voreinstellen
    .lpCustColors = AddressOf(BDF(0)) 'Benutzerdefinierte Farben zuweisen
```

```

End With

Retval = comdlg32.ChooseColor_Dlg(CC_T) 'Dialog anzeigen

If Retval <> 0 Then
  MsgBox Hex$(CC_T.rgbResult) 'gewählte Farbe als Hintergrund setzen
Else
  MsgBox "Das Auswählen einer Farbe ist fehlgeschlagen," & _
    "oder Sie haben Abbrechen gedrückt", kbCritical, "Fehler"
End If

```

## Playing MIDI music files on Windows

```

' DLL USING (old style)
' Warning! If you use predeclared DECLARE statements of VB6, be aware
' that the size of the datatypes differs between VB6 and KBasic,
' namely Long in VB6 must be Integer in KBasic! You have to change it.

' Play midi file using the windows api. Not portable!
' Be sure that the midi files are correctly named to the install path of KBasic
' in this example!

Declare Function mciSendString Lib "winmm.dll" Alias "mciSendStringA" _
  (ByVal lpszCommand As String, ByVal lpszReturnString As String, _
  ByVal cchReturnLength As Integer, ByVal hwndCallback As Integer) As Integer

Dim s As String
Dim k As String
Dim r As Integer

k = Space(1024)

r = mciSendString("close all", k, Len(k), 0)

Randomize Timer

Select Case Int(RND * 4) + 1
  Case 1
    s = "Open c:\kbasic15\examples\test\mond_1.mid Type sequencer Alias MUSIC"
  Case 2
    s = "Open c:\kbasic15\examples\test\mond_3.mid Type sequencer Alias MUSIC"
  Case 3
    s = "Open c:\kbasic15\examples\test\pathetique_1.mid Type sequencer Alias MUSIC"
  Case 4
    s = "Open c:\kbasic15\examples\test\pathetique_2.mid Type sequencer Alias MUSIC"
End Select

r = mciSendString(s, k, Len(k), 0)

If r = 0 Then
  r = mciSendString("play MUSIC from 0", k, Len(k), 0)
End If

```

## Playing WAVE sound files on Windows

```

' DLL USING (old style)
' Warning! If you use predeclared DECLARE statements of VB6, be aware
' that the size of the datatypes differs between VB6 and KBasic,
' namely Long in VB6 must be Integer in KBasic! You have to change it.

' Play wav file using the windows api. Not portable!
' Be sure that the wav files are correctly named to the install path of KBasic
' in this example!

Declare Function PlaySound Lib "winmm.dll" Alias "PlaySoundA" _

```



```
(lpzName As String, ByVal hModule As Integer, ByVal dwFlags As Integer) As Integer  
Dim s As String  
Randomize Timer  
Select Case Int(RND * 2) + 1  
  Case 1  
    s = "c:\kbasic14\ide\gong.wav"  
  
  Case 2  
    s = "c:\kbasic14\ide\neon_light.wav"  
End Select  
Dim r = PlaySound(s, 0, 0)
```

# IDE

## Naming Forms and other objects

---

- You must NOT use any keywords or builtins of KBasic as name of forms or other objects, otherwise the compilation of your program will fail.
- Names of objects like forms or reports are case sensitive, so you must write it everytime the same way.
- Name prefixes for the objects: frm for forms, tb for tables, q for queries. Examples are frmAddress, tbAddress, qAddress

# Mac

## Creating custom menubars on Mac

---

KBasic's look and feel is equally on every platform, but there is one exception. The menubar designer is not present on Mac, because of the limitation of the menubar provided by Qt on Mac. On Mac the Qt menubar is actually a native Mac OS X menubar and is not created by Qt itself, therefore there is no feature of having two menubars at once on screen. But this feature is needed for KBasic's menubar designer and works well on Windows and Linux.

Therefore, if you would like to create a custom menubar, you have to write it in plain text, what your menubar should look alike. When you click on new menubar on Mac a plain text editor appears in which you may enter the needed information. Don't worry it is really easy to create such a file and it is worth it!

By the way, menubar files created with the menubar designer of KBasic on Windows and Linux works on Mac as well.

## Structure of a menubar file of KBasic

The file's format looks like the following

- Menubar properties
- Menubar items + properties
- Menubar code

Comments are not allowed, so do not use REM or '.

## Menubar properties

There are no properties for menubars yet, but you need to write two lines, which encloses the menubaritems.

```
Begin MenuBar *  
    ' place your menubaritems here  
End MenuBar
```

## Menus + properties

Write the following lines for every menu and change NAME to the right one.

```
Begin Menu NAME  
    ' place your properties here  
End
```

## Layout properties

### ParentControl As String

Describes if this menubaritem is a main menubaritem on the bar (ParentControl = "") or a child menubaritem, then ParentControl contains the name of the parent menubaritem.

- ParentControl = "MenuBarItem0"

### ParentIndex As Integer

Together with ParentControl, ParentIndex describes the position of this menubaritem in the list of the child menubaritems of ParentControl. ParentIndex starts with 0 then 1, 2, 3 and so on.

- ParentIndex = 0

## Example

```
DO NOT COPY THIS EXAMPLE, BECAUSE IT CONTAINS SOME COMMENTS, WHICH ARE NOT ALLOWED  
  
Begin Menu MenuBarItem0  
    Caption = "Edit"  
    Enabled = True  
    ParentIndex = 0          ' THIS IS THE FIRST MENUBARITEM, because there is no ParentControl  
End                          ' (means ParentControl = ""), it is placed on the menubar on top.  
  
Begin Menu MenuBarItem1  
    Caption = "File"  
    Enabled = True  
    ParentControl = "MenuBarItem0" ' THIS IS THE FIRST MENUBARITEM, of the menu of MenuBarItem0  
declared above  
    ParentIndex = 0          ' set to position 0  
End
```

## Child menus

Working like normal menubaritems, but visually create a new menu, when the user enters this menubaritem.

Write ChildMenu instead of Menu as menubaritem.

```
Begin ChildMenu MenuBarItem4  
    Caption = "Custom ChildMenu"
```

```
Enabled = True
ParentControl = "MenuBarItem1"
ParentIndex = 1
End
```

## Menuitems

Write MenuItem instead of Menu as menubartype.

```
Begin MenuItem MenuBarItem4
Caption = "Custom menu entry"
Enabled = True
ParentControl = "MenuBarItem1"
ParentIndex = 1
End
```

## Data properties

### Caption As String

- Caption = "Hello World!"

### Icon As String

- Icon = "my.png"

### Key As String

- Key = "CTRL+R"

### Enabled As Boolean

- Enabled = True
- Enabled = False

### Checked As Boolean (not supported yet)

- Checked = True
- Checked = False

### Separator As Boolean

- Separator = True
- Separator = False

### StatusTip As String

- StatusTip = "Text to be displayed in statusbar, when mouse cursor is over this menubaritem."

### Tag As String

- Tag = "Custom information, normally you do not need this."

## Menubar code

Add your code lines after the line:

```
End MenuBar
```

Adding event procedures is really simply. Just write a new sub with the name of the menuitem and “\_OnEvent()”. Add your code inside and you are done.

```
Sub NAME_OnEvent()  
    ' your code here  
End Sub  
  
Sub MenuBarItem2_OnEvent()  
    ' your code here  
End Sub
```

Add two more functions for the about box and help/contents dialog.

```
Sub About_OnEvent()  
    ' your code here  
End Sub  
  
Sub Contents_OnEvent()  
    ' your code here  
End Sub
```

## Example

```
Begin MenuBar *  
  
    Begin Menu MenuBarItem0  
        Caption = "Edit"  
        Enabled = True  
        ParentIndex = 1  
    End  
  
    Begin Menu MenuBarItem1  
        Caption = "File"  
        Enabled = True  
        ParentIndex = 0  
    End  
  
    Begin MenuItem MenuBarItem2  
        Caption = "CustomItem"  
        Enabled = True  
        ParentControl = "MenuBarItem0"  
        ParentIndex = 0  
    End  
  
    Begin MenuItem MenuBarItem3  
        Caption = "More"  
        Enabled = True  
        ParentControl = "MenuBarItem0"  
        ParentIndex = 1  
    End  
  
    Begin ChildMenu MenuBarItem4  
        Caption = "Custom ChildMenu"  
        Enabled = True  
        ParentControl = "MenuBarItem1"  
        ParentIndex = 1  
    End  
  
    Begin MenuItem MenuBarItem5  
        Caption = "Close"  
        Enabled = True  
        ParentControl = "MenuBarItem1"  
        ParentIndex = 2  
    End  
  
    Begin MenuItem MenuBarItem10  
        Caption = "Print"  
        Enabled = True  
        ParentControl = "MenuBarItem4"  
        ParentIndex = 1  
    End  
  
    Begin MenuItem MenuBarItem11  
        Caption = "Hide"
```

```
    Enabled = True
    ParentControl = "MenuBarItem4"
    ParentIndex = 2
End

Begin MenuItem MenuBarItem12
    Caption = "Open"
    Enabled = True
    ParentControl = "MenuBarItem1"
    ParentIndex = 0
End

End MenuBar

Sub About_OnEvent()
    Print "you selected About"
End Sub

Sub Contents_OnEvent()
    Print "you selected Contents"
End Sub

Sub MenuBarItem2_OnEvent()
    Print "you selected CustomItem"
End Sub

Sub MenuBarItem3_OnEvent()
    Print "you selected More"
End Sub

Sub MenuBarItem12_OnEvent()
    Print "you selected Open"
End Sub

Sub MenuBarItem5_OnEvent()
    Print "you selected Close"
End Sub

Sub MenuBarItem10_OnEvent()
    Print "you selected Print"
End Sub

Sub MenuBarItem11_OnEvent()
    Print "you selected Hide"
End Sub
```