

Risø-R-1553(EN)

## Wilmar Deliverable D6.2 (e)

### Wilmar Planning Tool VBA documentation

Helge V. Larsen

**Author:** Helge V. Larsen  
**Title:** Wilmar Planning Tool, VBA documentation  
**Department:** Systems Analysis Department

**Risø-R-1553(EN)**  
**January 2006**

**Abstract:**

This is a documentation of the VBA (Visual Basic for Applications) in the Wilmar Planning Tool. VBA is used in the Wilmar User Shell (an Excel workbook) and in the three Access databases that hold input, scenario and output data.

The User Shell controls the operation of the Wilmar Planning Tool. In the User Shell various control parameters are set, and then a macro in the Input Database is run that writes input files for the Joint market Model and the Long Term Model. Afterwards these models can be started from the User Shell. Finally, the User Shell can start a macro in the Output Database that imports the output files from the models.

**ISSN 0106-2840**  
**ISBN 87-550-3512-4**

**Contract no.:**  
ENK5-CT-2002-00663

**Group's own reg. no.:**  
1200152

**Pages: 38**  
**Tables: 0**  
**References: 0**

Risø National Laboratory  
Information Service Department  
P.O.Box 49  
DK-4000 Roskilde  
Denmark  
Telephone +45 46774004  
[bibl@risoe.dk](mailto:bibl@risoe.dk)  
Fax +45 46774013  
[www.risoe.dk](http://www.risoe.dk)

# Contents

<b>Preface</b>	<b>5</b>
<b>1 Introduction</b>	<b>6</b>
<b>2 User shell</b>	<b>6</b>
1.1 Sheets	8
1.1.1 Sheet Shell	8
1.1.2 Sheet Def	8
1.1.3 Sheet Splash	9
1.1.4 Sheet Blank	9
1.2 Modules	9
1.2.1 Module myPublic	9
1.2.2 Module Start_Stop	11
1.2.3 Module aShell	12
1.2.4 Module Wilmar	13
1.2.5 Module Write_Queries	14
1.2.6 Module Del_Case	14
1.2.7 Module RunGAMS_1	15
1.2.8 Module RunGAMS_2	15
1.2.9 Module OptPeriod	15
1.2.10 Module Import	16
1.2.11 Module Misc	16
1.2.12 Module DB_Open_Close	17
1.2.13 Module HVL_ADO	17
1.3 Forms	18
1.3.1 Form Wilmar_Shell	18
1.3.2 Form Opt_Period	20
1.3.3 Form Show_File	20
<b>3 Input database</b>	<b>20</b>
1.4 Modules	20
1.4.1 Module myPublic	20
1.4.2 Module AutoExec	21
1.4.3 Module Write_Queries	21
1.4.4 Module Import	23
1.4.5 Module Misc	23
1.4.6 Module HVL_ADO	24
1.5 Macros	25
1.5.1 Macro AutoExec	25
1.5.2 Macro HVL_Write_Database_Index	25
1.5.3 Macro HVL_Write_Queries_All	25
1.5.4 Macro HVL_Write_Queries_JMM	25
1.5.5 Macro HVL_Write_Queries_LTM	25
1.5.6 Macro HVL_Write_Queries_WP5	25
1.5.7 Macro Form Write Queries – Makros	25
1.5.8 Macro UpdateCaseValueBaseTime_Sim	25
1.6 Form modules	25
1.6.1 Form Log_Form	25
1.6.2 Form AutoClose	25
1.6.3 Form Form Write Queries	26

<b>4 Output database</b>	<b>26</b>
1.7 Modules	26
1.7.1 Module myPublic	26
1.7.2 Module AutoExec	26
1.7.3 Module Del_Case	27
1.7.4 Module Import_TextFiles	28
1.7.5 Module Misc	28
1.7.6 Module HVL_ADO	29
1.8 Macros	30
1.8.1 Macro AutoExec	30
1.8.2 Macro Delete_Case	30
1.8.3 Macro Delete_Case_From_Excel	30
1.8.4 Macro HVL_Import_from_GAMS	30
1.8.5 Macro HVL_Write_Database_Index	30
1.9 Form modules	30
1.9.1 Form Delete Case	30
1.9.2 Form AutoClose	30
1.9.3 Form Choice	31
1.9.4 Form CompareCases	31
1.9.5 Form Show_Results	31
1.9.6 Form View_Balance	31
1.9.7 Form View_Production	32
1.9.8 Form View_Transmission	32
<b>5 Scenario database</b>	<b>32</b>
1.10 Modules	32
1.10.1 Module myPublic	32
1.10.2 Module AutoExec	32
1.10.3 Module Misc	33
1.10.4 Module HVL_ADO	33
1.11 Macros	34
1.11.1 Macro AutoExec	34
1.12 Form modules	34
1.12.1 Form AutoClose	34
<b>Index</b>	<b>35</b>

## **Preface**

This is a documentation of the VBA (Visual Basic for Applications) in the Wilmar Planning Tool. VBA is used in the Wilmar User Shell (an Excel workbook) and in the three Access databases that hold input, scenario and output data.

The User Shell controls the operation of the Wilmar Planning Tool. In the User Shell various control parameters are set, and then a macro in the Input Database is run that writes input files for the Joint market Model and the Long Term Model. Afterwards these models can be started from the User Shell. Finally, the User Shell can start a macro in the Output Database that imports the output files from the models.

# 1 Introduction

This is a documentation of the VBA (Visual Basic for Applications) in the Wilmar Planning Tool. VBA is used in the Wilmar User Shell (an Excel workbook) and in the three Access databases that hold input, scenario and output data.

The User Shell controls the operation of the Wilmar Planning Tool as shown in Figure 1. In the User Shell various control parameters are set, and then a macro in the Input Database is run that writes input files for the Joint market Model and the Long Term Model. Afterwards these models can be started from the User Shell. Finally, the User Shell can start a macro in the Output Database that imports the output files from the models. In the present version of the Planning Tool the User Shell does not control the Scenario Tree Creation Model that should be run prior to using the rest of the Planning Tool.

In Section 2 the User Shell is dealt with (sheets, VBA modules and forms) and in Sections 3 to 5 the three databases are described (VBA modules, macros and forms).

Finally, an index is found at the end of this report. For all VBA modules, routines (*Subs* and *Functions*), macros and forms the index tells on which page to find this entry. Also the sheets in the Excel workbook holding the User Shell are included.

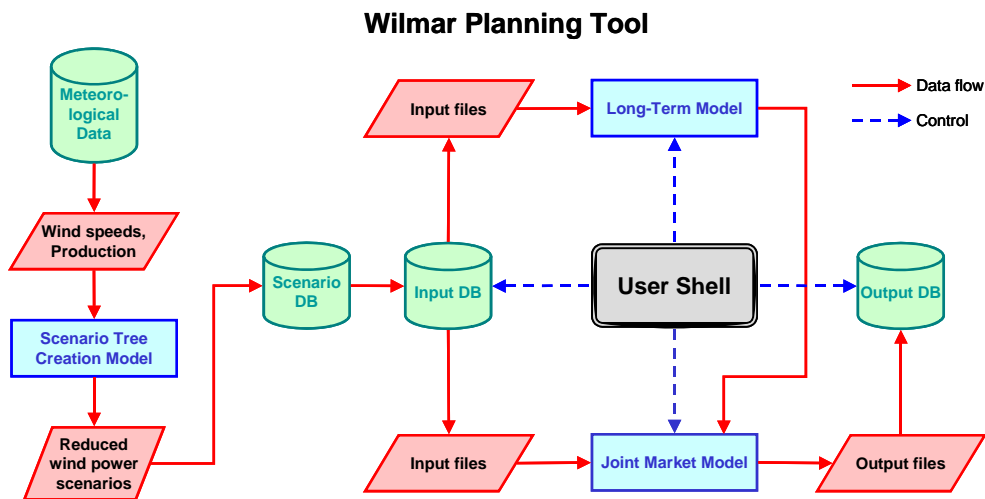


Figure 1. Wilmar Planning Tool.

## 2 User shell

When the Excel workbook holding the Wilmar user shell is opened, a toolbar with one button is created, c.f. Figure 2, and then a userform from which the Wilmar Planning Tool can be controlled is displayed, see Figure 3. This userform can be closed by clicking its upper right corner and opened again by clicking the button on the Wilmar toolbar.



*Figure 2. Wilmar toolbar.*

The form consists of a number of comboboxes and checkboxes where the user can select some scenario variables and the size of the system to optimize, e.g. which countries and which time period to include. When all choices have been made the whole setup can be saved in the input database by giving it a name in the bottom combobox and subsequently clicking the button “Save case”. An old case definition is written from the database if it is chosen in the combobox. A case definition can be deleted by the button marked “Delete case definition”. When the button “Write model data” is clicked, a dialog box is opened where the user can choose a model after which all queries in the input database related to this model are run, and the result is written to text files for later use as input to the model. If the user clicks the button “Run model” another dialog box is opened where the user can specify the optimization period whereupon the Joint market Model is started. The models read the text files mentioned above and produce a new set of output text files that can be imported to the output database by clicking the button “Read output”.

Figure 3. Userform controlling the Planning Tool.

In addition to the documentation of the VBA modules, also a short description of the sheets in the user shell is presented in the sequel.

## 1.1 Sheets

### 1.1.1 Sheet Shell

Gray empty sheet used as background for form Wilmar\_Shell.

### 1.1.2 Sheet Def

The user gives model name, paths to databases and GAMS executable, and defines all text strings to be shown in the user shell, i.e. the following data elements are specified :

- Model name.
- Path and name of :
  - Wilmar input database.
  - Wilmar scenario database.
  - Wilmar output database.
- Path of :



GAMS system (not the model).

Text strings used to define form Wilmar\_Shell (the user shell).

File names and paths can be written directly in the spreadsheet cells or the user can click one of the buttons to display a “browse for file/folder” window.

The syntax of the form definition is rather straightforward. Text strings and number of choices for Combo Boxes can be changed. Such changes are reflected immediately in the form. (It might be necessary to change the size of various elements in the form, if the length of text strings is changed considerably. This is not done by VBA.)

All information in sheet Def must be placed in the correct cells whose addresses are given in module myPublic. The number of frames, checkboxes and comboboxes must correspond to the design of form Wilmar\_Shell.

### **1.1.3 Sheet Splash**

Green splash screen used when starting Wilmar. Displays the text “Wilmar rules”.

### **1.1.4 Sheet Blank**

Green empty screen used when starting Wilmar.

## **1.2 Modules**

### **1.2.1 Module myPublic**

#### *Global constants*

This module sets various global constants :

Sheet names.

Positions in sheet Def for various information :

Path and name of :

Wilmar input database.

Wilmar scenario database.

Wilmar output database.

Start of shell definition.

Path of GAMS system (not model).

Folders for the GAMS model – relative to the folder holding the User Shell.

File extensions for various GAMS files.

Name of various dedicated GAMS files.

Information on which queries are exported for the JMM, LTM and WP5 models, respectively, and information on where to place the files.

Name of default case (that cannot be deleted from the user shell).

Link table used to enable communication between user shell and databases :

Table name (“Access-Excel Link”).

Field names.

Field values for specific information.

Name of database macros run from the user shell :

Input Database

HVL\_Write\_Queries\_JMM

HVL\_Write\_Queries\_LTM

HVL\_Write\_Queries\_WP5  
HVL\_Write\_Queries\_All  
Output Database  
Delete\_Case\_From\_Excel  
HVL\_Import\_from\_GAMS

Database tables and queries, names and other information related to case definitions :

Table for case definitions (“Case definition”).  
Query "CrossTab Case definition".  
Table "Base Scope (for Scenario Shell)"  
Table "Base Hypothesis" (Frame 1 choices).  
Table "Case selected".  
Table “Base Hypothesis” with Scope values for ComboBoxes in frame 1.

Booleans to control the amount of debug information.

Booleans to control the amount of confirmation by the user.

Definition of the Wilmar Toolbar, e.g. which VBA routine to call when the button is clicked.

***Global variables :***

This module defines global variables that are assigned values in Sub HVL\_Initialize\_1 below and in other modules :

Wilmar folder, i.e. the folder for the Excel workbook holding the user shell.

Full path (folder and file name) :

Input database.  
Output database.  
Scenario database.  
GAMS executable.  
GAMIDE executable.

Database object variables :

Connection to database.  
Access Input database.  
Access Output database.  
Access Scenario database.

User shell definition :

Database field names.  
Database data.  
Text strings used to define all elements of the form Wilmar\_Shell (comboboxes, checkboxes, titles, captions, frames).  
Object variables to hold various elements in the form.

***Sub HVL\_Initialize\_1 :***

Called from Sub Auto\_Open.

Field names for various tables related to case definitions are assigned values.

Read from sheet Def using the positions (global constants) mentioned above :

Model name.  
Database names and paths (input, output and scenario databases).

GAMS and GAMSIDE path.

### **1.2.2 Module Start\_Stop**

In this module are placed routines called when opening and closing the workbook.

#### ***Sub Auto\_Open :***

Automatically called at start-up.

Display the splash screen.

Build the Wilmar toolbar with a button for calling Sub HVL\_Load\_Wilmar\_Shell.

Call HVL\_ButtonFace (to define the face of the button).

Initialize :

Call HVL\_Initialize\_1.

Call HVL\_Initialize\_2.

Call HVL\_Initialize\_3.

Check for missing references (libraries) :

Call HVL\_Find\_Missing\_References.

Load the user shell :

Call HVL\_Load\_Wilmar\_Shell.

#### ***Sub Auto\_Close :***

Automatically called when the Excel file is closed.

Delete the toolbar.

Close input and output databases.

#### ***Sub HVL\_ButtonFace :***

Define the button-face of the button on the toolbar.

#### ***Function HVL\_Initialize\_2 :***

Check that path and name exist for :

Input database.

Output database.

Scenario database.

In Input database : Set link to Scenario database.

Call HVL\_Set\_Path\_To\_Linked\_Tables\_InDB.

In Output database : Set link to Input database.

Call HVL\_Set\_Path\_To\_Linked\_Tables\_OutDB.

Check that paths for various model folders exist.

Check that path and name exist for :

GAMS model file.

GAMS project file.

GAMS executable.

GAMSIDE executable.

#### ***Function HVL\_Initialize\_3 :***

Update values in table "Access-Excel Link" in the input database :

Folders for include files for JMM, LTM and WP5 models.

Update values in table "Access-Excel Link" in the output database :

Folder for output files from JMM model.

***Function HVL\_Set\_Path\_To\_Linked\_Tables\_InDB :***

Open an instance of Access and connect this to the input database.

Loop through all tables to identify linked tables.

All links are updated to point to the actual scenario database.

Close the input database.

***Function HVL\_Set\_Path\_To\_Linked\_Tables\_OutDB :***

Open an instance of Access and connect this to the output database.

Loop through all tables to identify linked tables.

All links are updated to point to the actual input database.

Close the output database.

**1.2.3 Module aShell**

In this module are placed routines used to open and present the userform Wilmar\_Shell.

***Sub HVL\_Load\_Wilmar\_Shell :***

Called by Sub Auto\_Open and by the button on the Wilmar toolbar.

Call HVL\_Initialize\_1 :

Field names for various tables are set.

Read information from sheet Def.

Call HVL\_Read\_CaseNames. See below.

Call HVL\_Read\_Shell\_Definition. See below.

Load the form.

Call HVL\_Set\_Shell\_Definition. See below.

Call HVL\_Update\_Shell. See below.

Call HVL\_Initialize\_Form. See below.

Show the form.

Call HVL\_Close\_DatabaseConnection\_ADO to close database connection.

***Function HVL\_Read\_CaseNames :***

Read names of case definitions in query qTable = "CrossTab Case definitions".

The names are placed in variable CaseList.

Call HVL\_BubbleSort to sort the list.

***Function HVL\_Read\_Shell\_Definition :***

Call HVL\_Read\_Frame\_1\_Choices :

Read choice lists for comboboxes in frame 1 from database and write it to sheet ShDef.

Read from sheet Def text strings to be used in form Wilmar\_Shell.

Put the shell definition in array variables.

***Function HVL\_Read\_Frame\_1\_Choices :***

Read choice lists for comboboxes in frame 1 from database table "Base Hypothesis".

Write it to sheet ShDef.

***Function HVL\_Set\_Shell\_Definition :***

Define object variables for each element in the form.

Text strings are moved from array variables to these object variables. This sets the text strings in form Wilmar\_Shell.

If the design of form Wilmar\_Shell is changed, this function should be changed accordingly.

***Function HVL\_Update\_Shell :***

Read CaseName from the form, i.e. from objCoBo\_Choice(0,1).

If CaseName is not in database query "CrossTab Case Definition" (i.e. it has been changed by the user) :

The choice list for CoBo\_01 is updated.

If CaseName is in database query "CrossTab Case Definition"

Read all choices for this casename in the input database, and put them into the form.

***Function HVL\_Initialize\_Form :***

Make things visible or invisible.

**1.2.4 Module Wilmar**

In this module are placed routines used when a case definition is saved to the database and when it is run, i.e. when queries are exported to text files.

***Sub HVL\_Run\_Wilmar :***

Is called when the button "Save case" or the button "Write model data" in form Wilmar\_Shell is clicked.

Call HVL\_Run\_Wilmar\_1. See below.

Call HVL\_Run\_Wilmar\_2. See below.

Call HVL\_Run\_Wilmar\_3. See below.

If the button "Write model data" was clicked :

Call HVL\_Write\_Queries. See module Write\_Queries.

Close database connection.

***Function HVL\_Run\_Wilmar\_1 :***

Get choices from object variables (the choices made by the user).

Put choices into array variables.

Put choices into sheet Def.

***Function HVL\_Run\_Wilmar\_2 :***

Write the choices to database table "Case definition".

If this is a new case definition :

Write CaseName to table "CaseID".

The AutoNumber field "CaseID" is automatically assigned a value.

Call HVL\_Update\_Shell.

***Function HVL\_Run\_Wilmar\_3 :***

Empty table "Case selected".

Write CaseName to table "Case selected".

### **1.2.5 Module Write\_Queries**

In this module is placed a routine used when queries are exported to text files.

***Function HVL\_Write\_Queries :***

Called by Sub HVL\_Run\_Wilmar.

The user is asked which model to run (JMM, LTM, WP5 or all).

Close the input database – if open.

Open an instance of Access and connect this to the input database.

Run the appropriate Access macro stored in the input database :

HVL\_Write\_Queries\_JMM

HVL\_Write\_Queries\_LTM

HVL\_Write\_Queries\_WP5

HVL\_Write\_Queries\_All

### **1.2.6 Module Del\_Case**

In this module are placed routines used to delete a case definition from the input database and the corresponding output data from the output database.

***Sub HVL\_Delete\_Case\_A :***

Is called when the button "Delete case definition" is clicked.

Make the combobox "Choose case definition" and the button "Delete" visible.

Build the drop-down list for the combobox.

***Sub HVL\_Delete\_Case\_B :***

Is called when the button "Delete" is clicked.

Read from object variable the name of the case definition to delete.

Get a confirmation from the user.

Delete this case definition from tables "Case definition" and "CaseID" in the input database.

Delete this case from the output database :

Delete record "Case\_to\_Delete" from table "Access-Excel Link".

Write record "Case\_to\_Delete" in table "Access-Excel Link".

Open an instance of Access and connect this to the output database.

Delete the case by running Access macro "Delete\_Case\_From\_Excel" stored in the output database.

Update the list of existing case names by calling HVL\_Read\_CaseNames. See above.

Rebuild choice list for the CaseName combobox, i.e. objCoBo\_01.

Call HVL\_Update\_Shell. See above.

### **1.2.7 Module RunGAMS\_1**

In this module and in module RunGAMS\_2 are placed routines used to call GAMS to run the JMM.

#### ***Sub Run\_GAMS :***

Call HVL\_Planning\_Loops\_1 in module OptPeriod to show form "Opt\_Period".

When the form is closed, HVL\_Planning\_Loops\_2 is called from the click event routine of the "OK" button. Then control is returned to Sub Run\_GAMS.

Identify path for the GAMS model.

Remove process folders from previous GAMS runs.

Identify path for GAMS output.

Identify path name for various GAMS files :

Executable file.

Listing file.

Project file.

Status file.

Run GAMS by calling Function GAMSrun in module RunGAMS\_2.

Show the status file by calling HVL\_Show\_File in module Misc.

If GAMS returned an error code :

The error message is displayed by calling Function GamsErrorMessage in module RunGAMS\_2.

Ask the user whether the GAMS IDE should be opened.

If so wished open the GAMS IDE.

### **1.2.8 Module RunGAMS\_2**

This is heavily inspired (i.e. copied) from McCarl's ExternalInCharge.xls, and subsequently slightly changed.

See "GAMS User Guide: 2004, Version 21.3, Bruce A.McCarl, February 14, 2004". If it is installed you can access it from the GamsIde : Help / McCarl Guide.

### **1.2.9 Module OptPeriod**

In this module are placed routines used in connection with form Opt\_Period.

#### ***Sub HVL\_Planning\_Loops\_1 :***

Is called from Sub Run\_GAMS.

Set text and comboboxes in form "Opt\_Period".

Load the form.

Show the form.

#### ***Sub HVL\_Planning\_Loops\_2 :***

Is called from the click event routine of the OK button in the "Opt\_Period" form.

Path and names for the two output files "LoopRuns.inc" and "StartLoop.inc" are identified.

LoopRuns and StartLoop are retrieved from the two comboboxes in form "Opt\_Period".

It is checked that the values are legal.

Form "Opt\_Period" is unloaded.

LoopRuns and StartLoop are written to files "LoopRuns.inc" and "StartLoop.inc" (GAMS include files).

#### **1.2.10 Module Import**

In this module is placed a function used to import GAMS output files into the output database.

##### ***Function HVL\_Import\_Txt\_Files :***

Close the output database – if open.

Open an instance of Access and connect this to the output database.

Run Access macro "HVL\_Import\_from\_GAMS" stored in the output database.

#### **1.2.11 Module Misc**

In this module are placed various general routines.

##### ***Function HVL\_Find\_Missing\_References :***

Loops through the workbook's references and displays a message if a reference is missing.

A missing reference is most often caused by moving the workbook to an older version of Microsoft Office.

For instance when moving from Microsoft Office XP to Microsoft Office 2000, there will be a missing reference to "Microsoft Access 10.0 Object Library" that should be replaced by a reference to "Microsoft Access 9.0 Object Library".

##### ***Sub HVL\_Show\_File :***

For this to function a reference to "Microsoft Scripting Runtime" is needed.

Loads the form "Show\_File" that holds a textbox.

Assigns a connection between the textbox and a text file.

Shows the form.

##### ***Function HVL\_DirExists :***

General function : Tells whether a folder exists.

##### ***Function HVL\_FileExists :***

General function : Tells whether a file exists.

##### ***Function HVL\_GetFolder :***

For this to function a reference to "Microsoft Shell Controls And Automation" is needed.

Opens a window with a folder tree.



Returns a string containing the folder selected by the user.

***Sub HVL\_BubbleSort :***

General function : Sorts a string array using bubble sort algorithm.

***Function HVL\_Mm\_to\_Month :***

General function : Converts month number to month name, e.g. "06" to "June".

***Function HVL\_Month\_to\_Mm :***

General function : Converts month name to month number, e.g. "June" to "06".

**1.2.12 Module DB\_Open\_Close**

In this module are placed routines used to open or close a database object.

A reference to "Microsoft Access 9.0 Object Library" is needed. In English Excel this is done as follows :

- Start the VBA editor.
- Activate the correct file, i.e. the Wilmar user shell.
- Choose menu Tools / References...
- Check "Microsoft Access 9.0 Object Library" (or whatever version you have).
- Click OK.

***Function Open\_Access\_InDB :***

Open input database.

***Function Quit\_Access\_InDB :***

Close input database.

***Function Open\_Access\_OutDB :***

Open output database.

***Function Quit\_Access\_OutDB :***

Close output database.

***Function Open\_Access\_ScenDB :***

Open scenario database.

***Function Quit\_Access\_ScenDB :***

Close scenario database.

**1.2.13 Module HVL\_ADO**

In this module are placed functions to connect to an Access database through ADO and to read, write and delete datasets. ADO = ActiveX Data Objects.

References to the following libraries are needed :

- "Microsoft ActiveX Data Objects Recordset 2.8 Library" and
- "Microsoft ADO Ext. 2.8 for DLL and Security".

In English Excel this is done as follows :

- Start the VBA editor.
- Activate the correct file.
- Choose menu Tools / References...

Check "Microsoft ActiveX Data Objects 2.8 Library" (or whatever version you have).

Check " Microsoft ADO Ext. 2.8 for DLL and Security " (or whatever version you have).

Click OK.

When building SQL statements for ADO, use % as wild-card character.

***Function HVL\_Exist\_ADO :***

Tell whether dataset(s) exist.

***Function HVL\_Count\_ADO :***

Count the number of datasets in a database table.

***Function HVL\_Read\_ADO :***

Read all datasets from a database. All fields, no filter.

***Function HVL\_Read\_All\_ADO :***

Read data from a database. Selected fields, no filter.

***Function HVL\_Read\_Filter\_ADO :***

Read one or more datasets from a database. Selected fields, filtered.

***Function HVL\_Write\_ADO :***

Write one dataset to a database.

***Function HVL\_WriteN\_ADO :***

Write several datasets to a database.

***Function HVL\_Update\_ADO :***

Update (change) an existing dataset in a database table.

***Function HVL\_Delete\_ADO :***

Delete one or more datasets from a database by building and executing an SQL statement.

***Function HVL\_Delete\_ADO\_Old :***

Delete one or more datasets from a database. A recordset is used to delete records one after another.

***Function HVL\_Create\_DatabaseConnection\_ADO :***

Open connection to a database.

***Function HVL\_Close\_DatabaseConnection\_ADO***

Close connection to a database.

## **1.3 Forms**

### **1.3.1 Form Wilmar\_Shell**

This is the Wilmar user interface. It consists of 6 frames (frame 0 to frame 5) each holding a number of labels, combo-boxes and check-boxes. All elements have been

named in a systematic way, e.g. ChBo\_35 is check-box # 5 in frame # 3, and CoBo\_14 is combo-box # 4 in frame # 1. Frame 0 holds the case name.

Moreover, the form has some buttons and an extra combobox :

- “Save case” button.
- “Write model data” button.
- “Run model” button.
- “Read output” button.
- “Delete case” button.
- A combobox used to select the case definition to delete.
- A “Delete” button.

The form has a number of event routines that call routines described above. The main event routines are mentioned in the sequel.

The form itself :

***Sub UserForm\_Activate***

Call HVL\_Close\_DatabaseConnection\_ADO

***Sub UserForm\_Terminate***

Call HVL\_Close\_DatabaseConnection\_ADO

Frame 0, Name of Case :

***Sub CoBo\_01\_DropButtonClick***

Call HVL\_Update\_Shell

***Sub CoBo\_01\_AfterUpdate***

Call HVL\_Update\_Shell

Frame 2, Time period analyzed :

***Sub ChBo\_21\_Change***

Call HVL\_Initialize\_Form

***Sub ChBo\_22\_Change***

Call HVL\_Initialize\_Form

Save Case :

***Sub Save\_Case\_Click***

Call HVL\_Run\_Wilmar("Save")

Write model data :

***Sub Write\_model\_data\_Click***

Call HVL\_Run\_Wilmar("Run")

Delete Case :

***Sub Del\_Case\_Click***

Call HVL\_Delete\_Case\_A

Delete :

***Sub Del\_Click***

Call HVL\_Delete\_Case\_B

Run Model :

***Sub Run\_Model\_Click***

Call Run\_GAMS

Read Output :

***Sub Read\_Output\_Click***

Call HVL\_Import\_from\_GAMS

### **1.3.2 Form Opt\_Period**

This form is loaded by Sub HVL\_Planning\_Loops\_1 in module OptPeriod. It is used to specify two control parameters for the optimization. The values are chosen from comboboxes. Some help information is presented in two textboxes (actually labels). A button is used to close the form.

Event routine :

***Sub CommandButton1\_Click***

Call HVL\_Planning\_Loops\_2

### **1.3.3 Form Show\_File**

This form is loaded by Sub HVL\_Show\_File in module Misc. It is used to present a text file to the user. It consists of a textbox for the file and a button to close the form.

Event routine :

***Sub CloseButton\_Click***

Unload the form.

## **3 Input database**

### **1.4 Modules**

#### **1.4.1 Module myPublic**

***Global constants***

This module sets various global constants :

Link table used to enable communication between user shell and database :

Table name (“Access-Excel Link”).

Field names.

Field values for specific information.

JMM, LTM and GAMS include files, respectively :

Start of names of queries to be exported.

Output folder ID to look up in the link table.

Output file extension.

Write Time and CaseName in top of files (True/False).

Use scaling (True/False).

If scaling :

Table to be scaled.

Subquery with scale factors.

Names of various action queries to be used when writing queries.

Name of table holding the name of the selected case.

Form used to show a log :  
Name of log form.  
Log window caption.

Developer's username.

#### **1.4.2 Module AutoExec**

In this module are placed routines called when opening and closing the database.

##### ***Function HVL\_AutoExec :***

Automatically called by macro AutoExec at start-up.

Wilmar toolbar :

- Create the Wilmar toolbar.
- Add button for writing GAMS include files for the JMM model.
- Add button for writing GAMS include files for the LTM model.
- Add button for writing GAMS include files for the WP5 model.
- Add button for writing GAMS include files for the all models.
- Add button for writing database index to file.
- If User = Developer :
  - Add an extra button for exporting VBA modules.

Check for missing references (libraries) :

- Call HVL\_Find\_Missing\_References.

Open form AutoClose as hidden.

##### ***Sub HVL\_AutoClose :***

Automatically called by UnLoad event routine of form AutoClose.

Here is placed VBA to be executed when the database is closed.

If User = Developer :

- Delete the extra button for exporting VBA modules.

##### ***Sub HVL\_Find\_Missing\_References :***

Loops through the database's references and displays a message if a reference is missing.

A missing reference is most often caused by moving the database to an older version of Microsoft Office.

For instance when moving from Microsoft Office XP to Microsoft Office 2000, there will be a missing reference to "Microsoft Access 10.0 Object Library" that should be replaced by a reference to "Microsoft Access 9.0 Object Library".

##### ***Sub Delete\_Toolbar :***

Delete the wilmar toolbar. Used by developer.

##### ***Sub ToolBar\_Buttons :***

Create toolbars and displays 1200 button faces. Used by developer.

#### **1.4.3 Module Write\_Queries**

In this module are placed routines used when queries are exported to text files.

##### ***Function HVL\_Write\_Queries\_JMM :***

Open the log form.

Run the action queries defined in module myPublic by calling HVL\_Run\_Action\_Queries.

Call HVL\_Write\_Queries\_0 ("JMM"), see below.

Ask the user whether the log form should be closed – and close it if so wished.

***Function HVL\_Write\_Queries\_LTM :***

Open the log form.

Run the action queries defined in module myPublic by calling HVL\_Run\_Action\_Queries.

Call HVL\_Write\_Queries\_0 ("LTM"), see below.

Ask the user whether the log form should be closed – and close it if so wished.

***Function HVL\_Write\_Queries\_WP5 :***

Open the log form.

Run the action queries defined in module myPublic by calling HVL\_Run\_Action\_Queries.

Call HVL\_Write\_Queries\_0 ("WP5"), see below.

Ask the user whether the log form should be closed – and close it if so wished.

***Function HVL\_Write\_Queries\_All :***

Open the log form.

Run the action queries defined in module myPublic by calling HVL\_Run\_Action\_Queries.

Call HVL\_Write\_Queries\_0 ("JMM"), see below.

Call HVL\_Write\_Queries\_0 ("LTM"), see below.

Call HVL\_Write\_Queries\_0 ("WP5"), see below.

Ask the user whether the log form should be closed – and close it if so wished.

***Sub HVL\_Run\_Action\_Queries :***

Run the action queries defined in module myPublic.

***Function HVL\_Write\_Queries\_0(Model) :***

Depending on Model ("JMM", "LTM" or "WP5"), choose the correct values from module myPublic :

Start of names of queries to be exported.

Output folder ID to look up in the link table.

Output file extension.

Write Time and CaseName in top of files (True/False).

Call HVL\_Write\_Queries\_1.

***Function HVL\_Case\_Selected :***

Retrieve the name of the selected case from table "Case selected".

***Function HVL\_Write\_Queries\_1 :***

Retrieve the output folder from the link table “Access-Excel Link”.

Find all queries whos name starts with the correct string.

Call HVL\_ExportDelim\_ADO for each such query.

***Sub HVL\_ExportDelim\_ADO :***

Read scale factors from database – if needed (see module myPublic).

Open output text file.

Write time stamp and case name to the file.

Open a recordset that retrieves all data from the query.

Call HVL\_ExportDelim\_1.

***Sub HVL\_ExportDelim\_1 :***

Write the content of the recordset to the file.

During this :

    Scaling is done for the last field in each record – if needed.

    Decimal comma is replaced by period.

**1.4.4 Module Import**

In this module are placed routines for importing text files. The routines are not used in the present version, but could be used as inspiration if data at a later time should be imported.

***Sub HVL\_Import\_NordPool :***

Call HVL\_Import\_NordPool\_1 to import various text files to database tables using import specifications. Not used.

***Sub HVL\_Import\_NordPool\_1 :***

Not used.

**1.4.5 Module Misc**

In this module are placed various general routines.

***Sub HVL\_Export\_VBA\_Access :***

Export all VBA modules to text files and write an index file with information on the modules.

***Sub HVL\_Export\_All\_Modules :***

Export all VBA modules to text files and write a more simple index file than HVL\_Export\_VBA\_Access does.

***Function HVL\_Write\_Database\_Index :***

Write a comprehensive index of the database : Tables, queries (incl. SQL), forms, reports, modules, and macros.

***Sub HVL\_BubbleSort :***

General function : Sorts a 1-dimensional string array using bubble sort algorithm.

***Sub HVL\_BubbleSort2 :***

General function : Sorts a 2-dimensional string array using bubble sort algorithm.

***Function HVL\_GetFolder :***

Open a window with a folder tree and return a string containing the folder selected by the user.

For this to function a reference to "Microsoft Shell Controls And Automation" is needed.

***Function HVL\_DirExist :***

General function : Tell whether a folder exists.

***Sub HVL\_Log\_Open :***

Open the log form used to show a log.

***Function HVL\_Log\_IsOpen :***

Tell whether the log form is open.

***Sub HVL\_Log\_Write :***

Write a text string to the bottom of the textbox in the log form.

***Sub HVL\_Log\_Empty :***

Delete all text in the log form.

***Sub HVL\_Log\_Close :***

Close the log form.

***Sub HVL\_Log\_Close\_1 :***

Ask the user whether the log form should be closed.

If yes, then the form is closed.

***Function WEEKNR :***

Input : Date (data type : Long). Returns the week number.

**1.4.6 Module HVL\_ADO**

In this module are placed functions to connect to an Access database through ADO and to read, write and delete datasets. ADO = ActiveX Data Objects.

References to the following libraries are needed :

- “Microsoft ActiveX Data Objects Recordset 2.8 Library” and
- “Microsoft ADO Ext. 2.8 for DLL and Security”.

In English Excel this is done as follows :

- Start the VBA editor.
- Activate the correct file.
- Choose menu Tools / References...
- Check "Microsoft ActiveX Data Objects 2.8 Library" (or whatever version you have).
- Check " Microsoft ADO Ext. 2.8 for DLL and Security " (or whatever version you have).
- Click OK.



See Module HVL\_ADO in the user shell (the Excel workbook) for information on individual routines.

## **1.5 Macros**

Macros can be executed in various ways :

- A macro can be assigned to a toolbar button.
- By clicking the Run button in the main database window.
- In VBA by using “DoCmd.RunMacro Macro\_Name”.
- From Excel using the VBA statement “AccessObject.DoCmd.RunMacro Macro\_Name”.

### **1.5.1 Macro AutoExec**

This macro is always run when the database is opened – because of its name AutoExec. It calls Function HVL\_AutoExec (see module AutoExec).

### **1.5.2 Macro HVL\_Write\_Database\_Index**

Call Function HVL\_Write\_Database\_Index (see module Misc).

### **1.5.3 Macro HVL\_Write\_Queries\_All**

Call Function HVL\_Write\_Queries\_All (see module Write\_Queries).

### **1.5.4 Macro HVL\_Write\_Queries\_JMM**

Call Function HVL\_Write\_Queries\_JMM (see module Write\_Queries).

### **1.5.5 Macro HVL\_Write\_Queries\_LTM**

Call Function HVL\_Write\_Queries\_LTM (see module Write\_Queries).

### **1.5.6 Macro HVL\_Write\_Queries\_WP5**

Call Function HVL\_Write\_Queries\_WP5 (see module Write\_Queries).

### **1.5.7 Macro Form Write Queries – Makros**

Obsolete ?

Call Function HVL\_Write\_Queries\_JMM (see module Write\_Queries).

Perform various requeries.

### **1.5.8 Macro UpdateCaseValueBaseTime\_Sim**

Obsolete ?

Open four queries.

## **1.6 Form modules**

“Form modules” contain event routines for forms.

### **1.6.1 Form Log\_Form**

This form is used to present a text file to the user. It consists of a textbox for the file and a button to close the form.

Event routines : None.

### **1.6.2 Form AutoClose**

This form is hidden, i.e. it cannot be seen in the main database window ! It is opened when the database is opened (see module AutoExec). The UnLoad event routine Form\_Unload is activated when the database is closed.

The form is introduced to be able to run some VBA code when the database is closed. It calls HVL\_AutoClose where VBA code to be executed when the database is closed should be placed.

***Sub Form\_Unload :***

Call HVL\_AutoClose.

**1.6.3 Form Form Write Queries**

Obsolete ?

## **4 Output database**

### **1.7 Modules**

#### **1.7.1 Module myPublic**

This module sets various global constants :

Link table used to enable communication between user shell and database :

Table name (“Access-Excel Link”).

Field names.

Field values for specific information.

File mask for GAMS output files that should be imported.

Information on naming convention for Import specification for importing text files.

Table holding case names and Ids :

Table name.

Field for case name.

Field for case Id.

Names of tables that are not using case Id but case name as key field.

Name of update query to be run when importing data.

Table to be filled by append query :

Name of table.

Name of append query.

Form used to show a log :

Name of log form.

Log window caption.

Name of form opened at start up.

This data element is only used to close the form again if User = Developer.

A form is opened at startup by a specification in menu Tools / Startup...

Developer’s username.

#### **1.7.2 Module AutoExec**

In this module are placed routines called when opening and closing the database.

***Function HVL\_AutoExec :***

Automatically called by macro AutoExec at start-up.

Form "Show\_Results" is loaded before this function is run. This is controlled by : Menu Tools / Startup... : Display Form/Page.

Wilmar toolbar :

- Create the Wilmar toolbar.

- Add button for importing GAMS output.

- Add button for writing database index to file.

- If User = Developer :

  - Add an extra button for exporting VBA modules.

  - Close the form "Show\_Results" that was loaded at start-up.

Check for missing references (libraries) :

- Call HVL\_Find\_Missing\_References.

Open form AutoClose as hidden.

#### ***Sub HVL\_AutoClose :***

Automatically called by UnLoad event routine of form AutoClose.

Here is placed VBA to be executed when the database is closed.

If User = Developer :

- Delete the extra button for exporting VBA modules.

#### ***Sub HVL\_Find\_Missing\_References :***

Loops through the database's references and displays a message if a reference is missing.

A missing reference is most often caused by moving the database to an older version of Microsoft Office.

For instance when moving from Microsoft Office XP to Microsoft Office 2000, there will be a missing reference to "Microsoft Access 10.0 Object Library" that should be replaced by a reference to "Microsoft Access 9.0 Object Library".

#### ***Sub Delete\_Toolbar :***

Deletes the wilmar toolbar. Used by developer.

#### ***Sub ToolBar\_Buttons :***

Creates toolbars and displays 1200 button faces. Used by developer.

### **1.7.3 Module Del\_Case**

In this module are placed routines used when deleting a case definition.

#### ***Function HVL\_Delete\_Case\_A :***

Deletes a case from the output database.

This function is called from event routine Delete\_Chosen\_Case\_Click and from function HVL\_Delete\_Case\_From\_Excel.

CaseName is argument to the function. CaseID is read from table "CaseNames".

Get folder for GAMS output files from table "Access-Excel Link".

Call HVL\_Delete\_Case to delete the case.

#### ***Function HVL\_Delete\_Case\_From\_Excel :***

This function is used when the delete operation is invoked from the user shell that writes the case name to the table "Access-Excel Link".

Get case name from table "Access-Excel Link" – and remove the record from the table.

Call HVL\_Delete\_Case\_A to delete the case.

***Function HVL\_Delete\_Case :***

This function carries out the actual deletion by deleting all information on CaseName from the output database.

This could in principle be done by deleting CaseName from table Tbl\_CaseNames. Because of relationships between tables, CaseName will then automatically be deleted from all other tables. But this method cannot be used because of "File sharing lock count exceeded". Therefore the following method is used.

Delete all information on CaseId from all tables that have Fld\_CaseId as key. During this skip :

Tbl\_CaseNames (to avoid "File sharing lock count exceeded").

Tables having Fld\_CaseName as key (Tbl\_TechnologyData and Tbl\_UnitGroupsInCase).

Delete CaseName from tables :

Tbl\_TechnologyData.

Tbl\_UnitGroupsInCase.

Tbl\_Append1 (that was filled by append query AppendQuery1 when the data was imported).

CaseTable.

#### **1.7.4 Module Import\_TextFiles**

In this module is placed a routine used to import GAMS output.

***Function HVL\_Import\_from\_GAMS :***

Get folder for GAMS output files from table LinkTable = "Access-Excel Link".

Get CaseName and CaseId from file CaseFile.

Check that the case is in the database.

If the case is in the database :

The user is asked whether the new data should replace the existing data, or they should be appended.

If replace : Delete the case by calling function HVL\_Delete\_Case.

Import data to table CaseTable.

Import data to other tables.

Update Choice.ChoiceCaseName1 to present case by running the update query defined in module myPublic.

Append data to table "TechData" by running query "Append\_TechData".

#### **1.7.5 Module Misc**

In this module are placed various general routines.

***Sub HVL\_Export\_VBA\_Access :***

Exports all VBA modules to text files. Writes an index file with information on the modules.

***Sub HVL\_Export\_All\_Modules :***

Exports all VBA modules to text files. Writes a more simple index file than HVL\_Export\_VBA\_Access does.

***Function HVL\_Write\_Database\_Index :***

Writes a comprehensive index of the database : Tables, queries (incl. SQL), forms, reports, modules, and macros.

***Sub HVL\_BubbleSort :***

General function : Sorts a 1-dimensional string array using bubble sort algorithm.

***Sub HVL\_BubbleSort2 :***

General function : Sorts a 2-dimensional string array using bubble sort algorithm.

***Function HVL\_GetFolder :***

Open a window with a folder tree and return a string containing the folder selected by the user.

For this to function a reference to "Microsoft Shell Controls And Automation" is needed.

***Function HVL\_DirExist :***

General function : Tell whether a folder exists.

***Sub HVL\_Log\_Open :***

Open the log form used to show a log.

***Function HVL\_Log\_IsOpen :***

Tell whether the log form is open.

***Sub HVL\_Log\_Write :***

Write a text string to the bottom of the textbox in the log form.

***Sub HVL\_Log\_Empty :***

Delete all text in the log form.

***Sub HVL\_Log\_Close :***

Close the log form.

***Sub HVL\_Log\_Close\_1 :***

Ask the user whether the log form should be closed.

If yes, then the form is closed.

***Sub Juha\_Empty\_All\_Tables :***

This function totally empties the tables that are partly emptied by function HVL\_Delete\_Case. Used by developer.

### **1.7.6 Module HVL\_ADO**

In this module are placed functions to connect to an Access database through ADO and to read, write and delete datasets. ADO = ActiveX Data Objects.

References to the following libraries are needed :

“Microsoft ActiveX Data Objects Recordset 2.8 Library” and  
“Microsoft ADO Ext. 2.8 for DLL and Security”.

In English Excel this is done as follows :

Start the VBA editor.

Activate the correct file.

Choose menu Tools / References...

Check "Microsoft ActiveX Data Objects 2.8 Library" (or whatever version you have).

Check " Microsoft ADO Ext. 2.8 for DLL and Security " (or whatever version you have).

Click OK.

See Module HVL\_ADO in the user shell (the Excel workbook) for information on individual routines.

## **1.8 Macros**

Macros can be executed in various ways :

- A macro can be assigned to a toolbar button.
- By clicking the Run button in the main database window.
- In VBA by using “DoCmd.RunMacro Macro\_Name”.
- From Excel using the VBA statement “AccessObject.DoCmd.RunMacro Macro\_Name”.

### **1.8.1 Macro AutoExec**

This macro is always run when the database is opened – because of its name AutoExec.

Call Function HVL\_AutoExec (see module AutoExec).

### **1.8.2 Macro Delete\_Case**

Open form Delete\_Case.

### **1.8.3 Macro Delete\_Case\_From\_Excel**

Call Function HVL\_Delete\_Case\_From\_Excel (see module Del\_Case).

### **1.8.4 Macro HVL\_Import\_from\_GAMS**

Call Function HVL\_Import\_from\_GAMS (see module Import\_TextFiles).

### **1.8.5 Macro HVL\_Write\_Database\_Index**

Call Function HVL\_Write\_Database\_Index (see module Misc).

## **1.9 Form modules**

“Form modules” contain event routines for forms.

### **1.9.1 Form Delete Case**

*Sub Delete\_Chosen\_Case\_Click :*

The user is asked if he really wants to delete the case from the output database.

If Yes : The case is deleted by calling HVL\_Delete\_Case\_A.

### **1.9.2 Form AutoClose**

This form is hidden, i.e. it cannot be seen in the main database window ! It is opened when the database is opened (see module AutoExec). The UnLoad event routine Form\_Unload is activated when the database is closed.

The form is introduced to be able to run some VBA code when the database is closed. It calls HVL\_AutoClose where VBA code to be executed when the database is closed should be placed.

#### ***Sub Form\_Unload :***

Call HVL\_AutoClose.

#### **1.9.3 Form Choice**

***Sub ChoiceCaseName\_Change***

***Sub ChoiceEndTime\_Change***

***Sub ChoiceRegion\_Change***

***Sub ChoiceStartTime\_Change***

#### **1.9.4 Form CompareCases**

***Sub ChoiceCaseName\_Change***

***Sub ChoiceCaseName2\_Change***

***Sub ChoiceEndTime\_Change***

***Sub ChoiceRegion\_Change***

***Sub ChoiceStartTime\_Change***

#### **1.9.5 Form Show\_Results**

***Sub ActivateChoice\_Click***

***Sub ChoiceCaseName1\_Change***

***Sub ChoiceArea\_Change***

***Sub ChoiceCaseName2\_Change***

***Sub ChoiceEndTime\_Change***

***Sub ChoiceRegion\_Change***

***Sub ChoiceRegion2\_Change***

***Sub ChoiceStartTime\_Change***

***Sub ChoiceUnitGroup\_Change***

***Sub Activate\_View\_Production\_Click***

***Sub Activiate\_View\_Transmission\_Click***

***Sub Activate\_CompareCases\_Click***

***Sub Activate\_View\_Balance\_Click***

***Sub Activate\_ProfitUnitGroup\_Click***

***Sub Activate\_Bal\_Energy\_Click***

#### **1.9.6 Form View\_Balance**

*Sub ChoiceCaseName\_Change*

*Sub ChoiceEndTime\_Change*

*Sub ChoiceStartTime\_Change*

*Sub Combo26\_Change*

### **1.9.7 Form View\_Production**

*Sub ChoiceCaseName1\_Change*

*Sub ChoiceUnitGroup\_Change*

*Sub ChoiceEndTime\_Change*

*Sub ChoiceStartTime\_Change*

### **1.9.8 Form View\_Transmission**

*Sub ChoiceCaseName1\_Change*

*Sub ChoiceRegion\_Change*

*Sub ChoiceRegion2\_Change*

*Sub ChoiceEndTime\_Change*

*Sub ChoiceStartTime\_Change*

## **5 Scenario database**

### **1.10 Modules**

#### **1.10.1 Module myPublic**

This module sets a global constant : Developer's username.

#### **1.10.2 Module AutoExec**

In this module are placed routines called when opening and closing the database.

##### ***Function HVL\_AutoExec :***

Automatically called by macro AutoExec at start-up.

Wilmar toolbar :

    Create the Wilmar toolbar.

    Add button for writing database index to file.

    If User = Developer :

        Add an extra button for exporting VBA modules.

Check for missing references (libraries) :

    Call HVL\_Find\_Missing\_References.

Open form AutoClose as hidden.

##### ***Sub HVL\_AutoClose :***

Automatically called by UnLoad event routine of form AutoClose.

Here is placed VBA to be executed when the database is closed.

If User = Developer :

    Delete the extra button for exporting VBA modules.



***Sub HVL\_Find\_Missing\_References :***

Loops through the database's references and displays a message if a reference is missing. A missing reference is most often caused by moving the database to an older version of Microsoft Office.

For instance when moving from Microsoft Office XP to Microsoft Office 2000, there will be a missing reference to "Microsoft Access 10.0 Object Library" that should be replaced by a reference to "Microsoft Access 9.0 Object Library".

***Sub Delete\_Toolbar :***

Deletes the wilmar toolbar. Used by developer.

***Sub ToolBar\_Buttons :***

Creates toolbars and displays 1200 button faces. Used by developer.

**1.10.3 Module Misc**

In this module are placed various general routines.

***Sub HVL\_Export\_VBA\_Access :***

Exports all VBA modules to text files. Writes an index file with information on the modules.

***Sub HVL\_Export\_All\_Modules :***

Exports all VBA modules to text files. Writes a more simple index file than HVL\_Export\_VBA\_Access does.

***Function HVL\_Write\_Database\_Index :***

Writes a comprehensive index of the database : Tables, queries (incl. SQL), forms, reports, modules, and macros.

***Sub HVL\_BubbleSort :***

General function : Sorts a 1-dimensional string array using bubble sort algorithm.

***Sub HVL\_BubbleSort2 :***

General function : Sorts a 2-dimensional string array using bubble sort algorithm.

***Function HVL\_GetFolder :***

Open a window with a folder tree and return a string containing the folder selected by the user.

For this to function a reference to "Microsoft Shell Controls And Automation" is needed.

***Function HVL\_DirExist :***

General function : Tell whether a folder exists.

***Function WEEKNR :***

Finds the week number for any date.

**1.10.4 Module HVL\_ADO**

In this module are placed functions to connect to an Access database through ADO and to read, write and delete datasets. ADO = ActiveX Data Objects.

References to the following libraries are needed :

“Microsoft ActiveX Data Objects Recordset 2.8 Library” and  
“Microsoft ADO Ext. 2.8 for DLL and Security”.

In English Excel this is done as follows :

Start the VBA editor.

Activate the correct file.

Choose menu Tools / References...

Check "Microsoft ActiveX Data Objects 2.8 Library" (or whatever version you have).

Check " Microsoft ADO Ext. 2.8 for DLL and Security " (or whatever version you have).

Click OK.

See Module HVL\_ADO in the user shell (the Excel workbook) for information on individual routines.

## 1.11 Macros

Macros can be executed in various ways :

- A macro can be assigned to a toolbar button.
- By clicking the Run button in the main database window.
- In VBA by using DoCmd.RunMacro Macro\_Name.
- From Excel using the VBA statement  
AccessObject.DoCmd.RunMacro Macro\_Name.

### 1.11.1 Macro AutoExec

This macro is always run when the database is opened – because of its name AutoExec.

Call Function HVL\_AutoExec (see module AutoExec).

## 1.12 Form modules

“Form modules” contain event routines for forms.

### 1.12.1 Form AutoClose

This form is hidden, i.e. it cannot be seen in the main database window ! It is opened when the database is opened (see module AutoExec). The UnLoad event routine Form\_Unload is activated when the database is closed.

The form is introduced to be able to run some VBA code when the database is closed. It calls HVL\_AutoClose where VBA code to be executed when the database is closed should be placed.

***Sub Form\_Unload :***

Call HVL\_AutoClose.

# Index

Name	Type	Page
Activate_Bal_Energy_Click	Sub	31
Activate_CompareCases_Click	Sub	31
Activate_ProfitUnitGroup_Click	Sub	31
Activate_View_Balance_Click	Sub	31
Activate_View_Production_Click	Sub	31
ActivateChoice_Click	Sub	31
Activate_View_Transmission_Click	Sub	31
aShell	Module	12
Auto_Close	Sub	11
Auto_Open	Sub	11
AutoClose	Form	25, 30, 34
AutoExec	Module	21
AutoExec	Macro	25
AutoExec	Module	26
AutoExec	Macro	30
AutoExec	Module	32
AutoExec	Macro	34
Blank	Sheet	9
ChBo_21_Change	Sub	19
ChBo_22_Change	Sub	19
Choice	Form	31
ChoiceArea_Change	Sub	31
ChoiceCaseName_Change	Sub	31, 31, 32
ChoiceCaseName1_Change	Sub	31, 32, 32
ChoiceCaseName2_Change	Sub	31, 31
ChoiceEndTime_Change	Sub	31, 31, 31, 32, 32, 32
ChoiceRegion_Change	Sub	31, 31, 31, 32
ChoiceRegion2_Change	Sub	31, 32
ChoiceStartTime_Change	Sub	31, 31, 31, 32, 32, 32
ChoiceUnitGroup_Change	Sub	31, 32
CloseButton_Click	Sub	20
CoBo_01_AfterUpdate	Sub	19
CoBo_01_DropButton_Click	Sub	19
Combo26_Change	Sub	32
CommandButton1_Click	Sub	20
CompareCases	Form	31
DB_Open_Close	Module	17
Def	Sheet	8
Del_Case	Module	14, 27
Del_Case_Click	Sub	19
Del_Click	Sub	19
Delete Case	Form	30
Delete_Case	Macro	30
Delete_Case_From_Excel	Macro	30

<b>Name</b>	<b>Type</b>	<b>Page</b>
Delete_Chosen_Case_Click	Sub	30
Delete_Toolbar	Sub	21, 27, 33
Form Write Queries	Form	26
Form Write Queries – Makros	Macro	25
Form_Unload	Sub	26, 31, 34
HVL_ADO	Module	17, 24, 29, 33
HVL_AutoClose	Sub	21, 27, 32
HVL_AutoExec	Function	21, 26, 32
HVL_BubbleSort	Sub	17, 23, 29, 33
HVL_BubbleSort2	Sub	24, 29, 33
HVL_ButtonFace	Sub	11
HVL_Case_Selected	Function	22
HVL_Close_DatabaseConnection_ADO	Function	18
HVL_Count_ADO	Function	18
HVL_Create_DatabaseConnection_ADO	Function	18
HVL_Delete_ADO	Function	18
HVL_Delete_ADO_Old	Function	18
HVL_Delete_Case	Function	28
HVL_Delete_Case_A	Sub	14
HVL_Delete_Case_A	Function	27
HVL_Delete_Case_B	Sub	14
HVL_Delete_Case_From_Excel	Function	27
HVL_DirExist	Function	24, 29, 33
HVL_DirExists	Function	16
HVL_Exist_ADO	Function	18
HVL_Export_All_Modules	Sub	23, 29, 33
HVL_Export_VBA_Access	Sub	23, 28, 33
HVL_ExportDelim_1	Sub	23
HVL_ExportDelim_ADO	Sub	23
HVL_FileExists	Function	16
HVL_Find_Missing_References	Function	16
HVL_Find_Missing_References	Sub	21, 27, 33
HVL_GetFolder	Function	16, 24, 29, 33
HVL_Import_from_GAMS	Function	28
HVL_Import_from_GAMS	Macro	30
HVL_Import_NordPool	Sub	23
HVL_Import_NordPool_1	Sub	23
HVL_Import_Txt_Files	Function	16
HVL_Initialize_1	Sub	10
HVL_Initialize_2	Function	11
HVL_Initialize_3	Function	11
HVL_Initialize_Form	Function	13
HVL_Load_Wilmar_Shell	Sub	12
HVL_Log_Close	Sub	24, 29
HVL_Log_Close_1	Sub	24, 29
HVL_Log_Empty	Sub	24, 29
HVL_Log_IsOpen	Function	24, 29
HVL_Log_Open	Sub	24, 29
HVL_Log_Write	Sub	24, 29

<b>Name</b>	<b>Type</b>	<b>Page</b>
HVL_Mm_to_Month	Function	17
HVL_Month_to_Mm	Function	17
HVL_Planning_Loops_1	Sub	15
HVL_Planning_Loops_2	Sub	15
HVL_Read_ADO	Function	18
HVL_Read_All_ADO	Function	18
HVL_Read_CaseNames	Function	12
HVL_Read_Filter_ADO	Function	18
HVL_Read_Frame_1_Choices	Function	13
HVL_Read_Shell_Definition	Function	12
HVL_Run_Action_Queries	Sub	22
HVL_Run_Wilmar	Sub	13
HVL_Run_Wilmar_1	Function	13
HVL_Run_Wilmar_2	Function	13
HVL_Run_Wilmar_3	Function	14
HVL_Set_Path_To_Linked_Tables_InDB	Function	12
HVL_Set_Path_To_Linked_Tables_OutDB	Function	12
HVL_Set_Shell_Definition	Function	13
HVL_Show_File	Sub	16
HVL_Update_ADO	Function	18
HVL_Update_Shell	Function	13
HVL_Write_ADO	Function	18
HVL_Write_Database_Index	Function	23
HVL_Write_Database_Index	Macro	25
HVL_Write_Database_Index	Function	29
HVL_Write_Database_Index	Macro	30
HVL_Write_Database_Index	Function	33
HVL_Write_Queries	Function	14
HVL_Write_Queries_0(Model)	Function	22
HVL_Write_Queries_1	Function	23
HVL_Write_Queries_All	Function	22
HVL_Write_Queries_All	Macro	25
HVL_Write_Queries_JMM	Function	21
HVL_Write_Queries_JMM	Macro	25
HVL_Write_Queries_LTM	Function	22
HVL_Write_Queries_LTM	Macro	25
HVL_Write_Queries_WP5	Function	22
HVL_Write_Queries_WP5	Macro	25
HVL_WriteN_ADO	Function	18
Import	Module	16, 23
Import_TextFiles	Module	28
Juha_Empty_All_Tables	Sub	29
Log_Form	Form	25
Misc	Module	16, 23, 28, 33
modules	Form	25, 30, 34
myPublic	Module	9, 20, 26, 32
Open_Access_InDB	Function	17
Open_Access_OutDB	Function	17
Open_Access_ScenDB	Function	17

<b>Name</b>	<b>Type</b>	<b>Page</b>
Opt_Period	Form	20
OptPeriod	Module	15
Quit_Access_InDB	Function	17
Quit_Access_OutDB	Function	17
Quit_Access_ScenDB	Function	17
Read_Output_Click	Sub	20
Run_GAMS	Sub	15
Run_Model_Click	Sub	20
RunGAMS_1	Module	15
RunGAMS_2	Module	15
Save_Case_Click	Sub	19
Shell	Sheet	8
Show_File	Form	20
Show_Results	Form	31
Splash	Sheet	9
Start_Stop	Module	11
ToolBar_Buttons	Sub	21, 27, 33
UpdateCaseValueBaseTime_Sim	Macro	25
UserForm_Activate	Sub	19
UserForm_Terminate	Sub	19
View_Balance	Form	31
View_Production	Form	32
View_Transmission	Form	32
WEEKNR	Function	24, 33
Wilmar	Module	13
Wilmar_Shell	Form	18
Write_model_data_Click	Sub	18
Write_Queries	Module	14, 21

Risø's research is aimed at solving concrete problems in the society.

Research targets are set through continuous dialogue with business, the political system and researchers.

The effects of our research are sustainable energy supply and new technology for the health sector.