

# Using the CONIFERS Dynamic Link Library (DLL) from Visual Basic for Applications in Microsoft Access

Jeff D. Hamann<sup>1</sup>  
Forest Informatics, Inc.  
Corvallis, Oregon 97331  
[jeff.hamann@forestinformatics.com](mailto:jeff.hamann@forestinformatics.com)

August 25, 2008

# 1 Introduction

This tutorial is designed to familiarize you with using the CONIFERS Dynamic Link Library (DLL). Using a Microsoft Access database to store a small forest inventory, the goal is to project the inventory into the future, using simple programming constructs in the Visual Basic for Applications environment. We expend little or no effort on the minutiae of Microsoft Visual Basic or fundamental programming constructs such as `for-loops`, or `if-else-endif` code blocks assuming the reader moderately versed in procedural programming (i.e. FORTRAN, SAS, VB, R/S+, C, etc.). Our goal is to present the basic procedures for interfacing with the DLL itself, and in order to do so, we use a Microsoft Access database and Visual Basic for Applications, with some basic utilizes specific for this inventory, to read and write data, translate species codes between our internal representation (e.g. "BM") and the functional species in the CONIFERS (i.e. `integers`), and to call functions within the library itself.

The first tutorial is to familiarize you with installing the Visual Basic for Applications Module (i.e. `conifers.bas`) that allows you to interface the CONIFERS growth model using Visual Basic. Next, we map our species codes to those available in CONIFERS by using some simple Visual Basic code to map our species table to the CONIFERS function species and use the interface function, `set_species_map`. In the third tutorial, we load data into the CONIFERS data types, by calling a utility function, and impute missing values using the CONIFERS equations using the `fill_in_missing_values` function. In tutorial four, we call the `ex_update_total_summaries` function in CONIFERS to compute summary statistics, and in tutorial five, we finally project data using the CONIFERS growth model using the `project_sample` function. In tutorial six, we create a table to store our current and projected summaries, add the code to open and delete any current contents of our projected summaries table, and insert new records into the empty table for each of the projected stands. Last, in tutorial seven, we demonstrate projecting multiple stands, updating the database annually, and perform thinning for each projected sample.

Throughout this tutorial, there will be large blocks of "source code". If you are familiar with Visual Basic for Applications (i.e. VBA), you may want to skip over them. If not, try reading over the code to gain an understanding of the syntax. Specifically, a "\_" represents a line continuation. Source code (i.e. variables, VBA functions, CONIFERS functions), and is presented in `courier`, and when new code is to be added to previous code, we include a "»»" in the left margin.

## 2 First Tutorial: Installing the VBA Interface Module

Open the Visual Basic Editor in Microsoft Access. To get to the Visual Basic Editor, go to the Module View in the Database View and click the "New" button. To install the interface module, import the file named "conifers.bas", using the "File, Import..." menu option from within the Visual Basic Editor. When you have successfully imported the interface module, you should have a screen similar to the one shown in Figure 1. If not, either double click on

the module, or use the "View module code" menu item.

The interface module contains user defined constants (i.e. `Public Const`), user defined data types (e.g. `Type PLANT_RECORD ... End Type`), and the interface functions (e.g. subroutines) themselves, used to communicate with the the CONIFERS library. These are the building blocks for all functions associated with CONIFERS.

### 3 Second Tutorial: Setting the Species Map

In this tutorial, we map our species codes to those available in CONIFERS. We create an array of `SPECIES_MAP_RECORDs` which will serve as our functional species map into CONIFERS, and assign values of the `SPECIES_MAP_RECORD's` elements from the species definitions in our inventory database. We use a few basic Visual Basic routines to obtain the sizes we need for our arrays of user defined data types and finally assign the species map using the `conifers` interface function, `set_species_map`.

The method which CONIFERS handles multiple species is by mapping real, or nominal, species to "functional species", since most growth models only handle a few species. The seven variables for a nominal species are defined in the user defined data type `SPECIES_MAP_RECORD`:

```
Type SPECIES_MAP_RECORD
    idx As Long
    fsp_idx As Long
    endemic_mortality As Double
    max_sdi As Double
    browse_damage As Double
    mechanical_damage As Double
    genetic_worth_h As Double
    genetic_worth_d As Double
End Type
```

so that an array of which are used in the function:

```
Public Declare Sub set_species_map Lib _
    "conifers.dll" _
    (ByRef return_code As Long, _
     ByVal model_variant As Long, _
     ByVal n_species As Long, _
     ByRef species_ptr As SPECIES_MAP_RECORD)
```

`set_species_map` to set the CONIFERS functional species to your species map. Once you have set the species map, you need only define the species `idx` in your array of `PLANT_RECORDs`.

A simple technique to initialize the species map is to perform a query from your own species table. We begin with a practice script to select a set of species from the species table, looping over a Microsoft ADO `Recordset`, determining how large to make our array of `SPECIES_MAP_RECORD`, and finally closing the open `Recordset`:

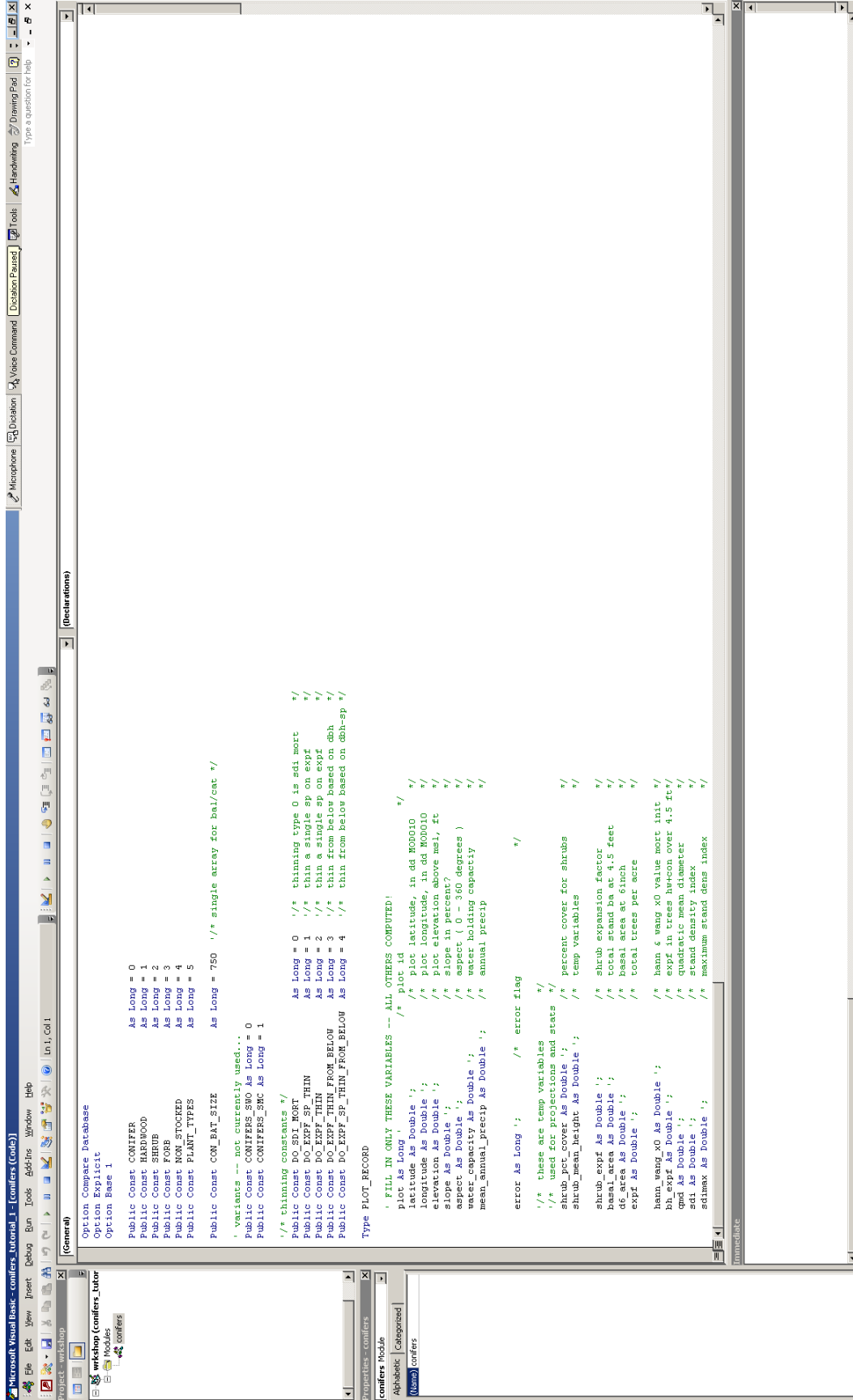


Figure 1: CONFIFERS Visual Basic for Applications interface module.

```

Sub main()

    Dim sp_recs() As conifers.SPECIES_MAP_RECORD    ' conifers data definition (type)
    Dim species_rs As New ADODB.Recordset          ' vba object
    Dim n_species As Long

    ' issue an sql statement to open the table (or some subset)
    ' the variant is SMC_CONIFERS (variant = 1)
    species_rs.Open "select * from species where variant = 1", _
        CurrentProject.Connection, adOpenKeyset, adLockOptimistic

    ' get a count of the species recordsets
    species_rs.MoveLast
    n_species = species_rs.RecordCount
    ReDim sp_recs(n_species)

    ' move the first record in the record set
    ' and loop over the access records and initialize the conifers.SPECIES_RECORD

    ' close the recordset (ado)
    species_rs.Close
    Set species_rs = Nothing

End Sub

```

Just under the notes to yourself about looping over the records, insert the code to extract the values from the recordset (from the database) to the array we **ReDim**'d. The code requires we go back to the first record in the recordset, and then loop over each record, and assign the values to the elements in our array:

```

    ' move the first record in the record set
    species_rs.MoveFirst
    i = 1
    Do While (Not species_rs.EOF)

        Debug.Print "Species = "; species_rs("idx").Value; " "; _
            species_rs("fsp_idx").Value

        ' initialize the elements of the conifers.SPECIES_MAP_RECORD
        sp_recs(i).idx = species_rs("idx").Value
        sp_recs(i).fsp_idx = species_rs("fsp_idx").Value
        sp_recs(i).endemic_mortality = species_rs("endemic_mortality").Value
        sp_recs(i).max_sdi = species_rs("max_sdi").Value
        sp_recs(i).browse_damage = species_rs("browse_damage").Value
        sp_recs(i).mechanical_damage = species_rs("mechanical_damage").Value
    
```

```

    sp_recs(i).genetic_worth_h = species_rs("genetic_worth_h").Value
    sp_recs(i).genetic_worth_d = species_rs("genetic_worth_d").Value
    i = i + 1
    species_rs.MoveNext
Loop

```

and after closing the Recordset, calling the set\_species\_map function in the CONIFERS library:

```

' finally, call the conifers shared library function (set_species_map)
Call conifers.set_species_map(return_code, conifers.CONIFERS_SMC, n_species, sp_recs(1))
If (return_code) Then
    Debug.Print "species map not initialized -- stopping"
    Exit Sub
End If

```

so that the final, and complete subroutine looks like,

```

Sub main()

    Dim i As Long
    Dim return_code As Long

    Dim sp_recs() As conifers.SPECIES_MAP_RECORD    ' conifers data definition (type)
    Dim species_rs As New ADODB.Recordset          ' vba object
    Dim n_species As Long

    ' issue an sql statement to open the table (or some subset)
    species_rs.Open "select * from species where variant = 1", _
        CurrentProject.Connection, adOpenKeyset, adLockOptimistic

    ' get a count of the species recordsets
    species_rs.MoveLast
    n_species = species_rs.RecordCount
    ReDim sp_recs(n_species)

    ' move the first record in the record set
    ' and loop over the access records and initialize the conifers.SPECIES_RECORD
    species_rs.MoveFirst
    i = 1
    Do While (Not species_rs.EOF)

        ' print some basic data to make sure you can see it.
        Debug.Print "Variant = "; species_rs("variant").Value; _
            " Species = "; species_rs("idx").Value; " "; species_rs("fsp_idx").Value
    
```

```

' initialize the elements of the conifers.SPECIES_MAP_RECORD
sp_recs(i).idx = species_rs("idx").Value
sp_recs(i).fsp_idx = species_rs("fsp_idx").Value
sp_recs(i).endemic_mortality = species_rs("endemic_mortality").Value
sp_recs(i).max_sdi = species_rs("max_sdi").Value
sp_recs(i).browse_damage = species_rs("browse_damage").Value
sp_recs(i).mechanical_damage = species_rs("mechanical_damage").Value
sp_recs(i).genetic_worth_h = species_rs("genetic_worth_h").Value
sp_recs(i).genetic_worth_d = species_rs("genetic_worth_d").Value
i = i + 1
species_rs.MoveNext
Loop

' close the species recordset (ado)
species_rs.Close
Set species_rs = Nothing

' finally, call the conifers shared library function (set_species_map)
Call conifers.set_species_map(return_code, conifers.CONIFERS_SMC, n_species, sp_recs(1))
If (return_code) Then
    Debug.Print "species map not initialized -- stopping"
    Exit Sub
End If

Debug.Print n_species; " species initialized in species map"

End Sub

```

which when we run the script, yields:

```

Variant = 1 Species = 0 0
Variant = 1 Species = 1 1
Variant = 1 Species = 2 2
3 species initialized in species map

```

in the Immediate window as shown in Figure 2.

Depending on how your data is structured, you may use this simple technique to load data into CONIFERS user defined data types or you can write any number of several subroutines to meet your needs for the remainder of the tutorial. For convenience we have included a utility module with several helper subroutines for this specific database. Of course, you are free to modify them or create your own from scratch.

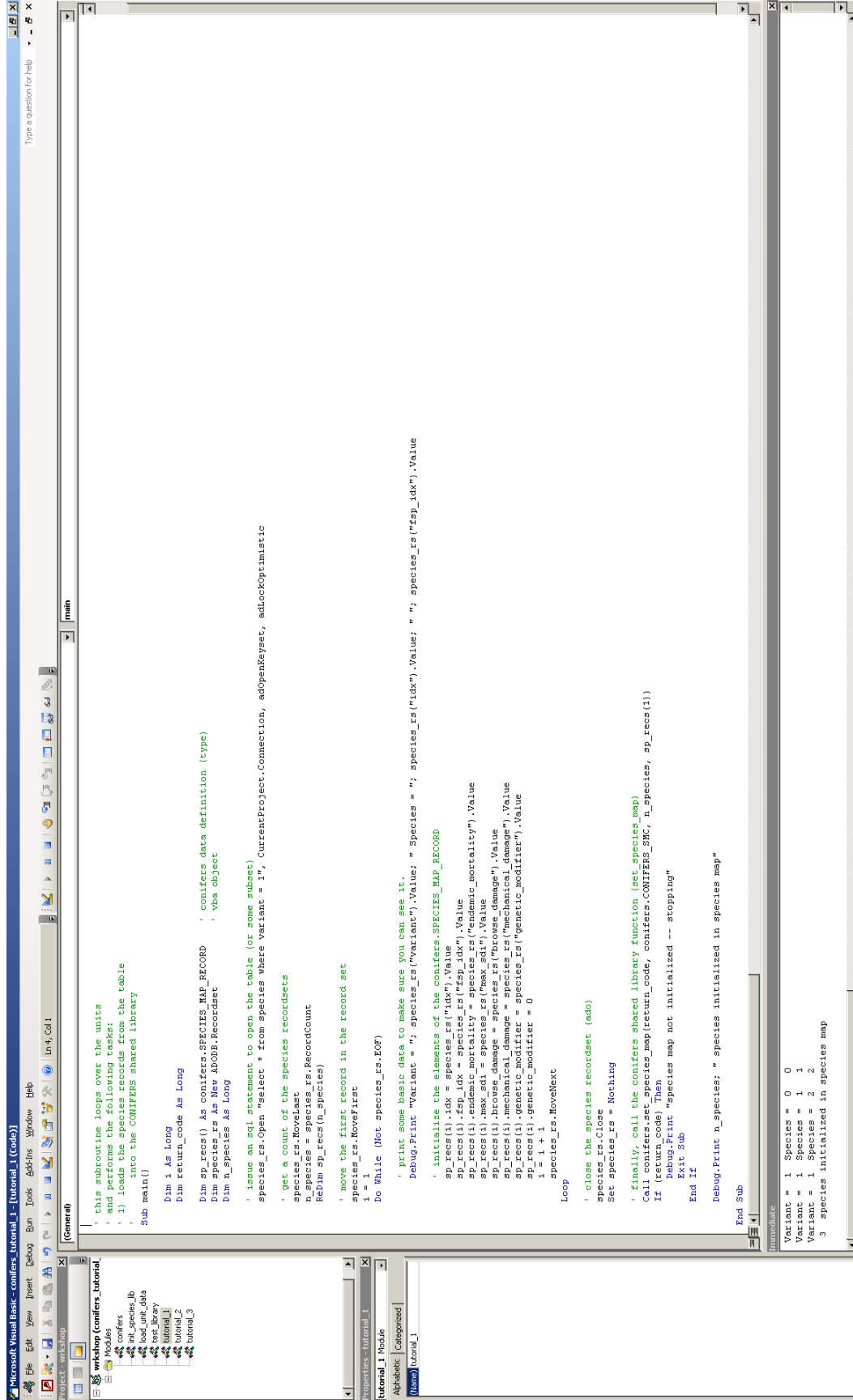


Figure 2: Immediate window output



## 4 Third Tutorial: Loading Data

Let's begin by importing the `workshop_utils.bas` module we included to support the tutorial database. In this module, we include functions to load our species table into the CONIFERS library, which we used in the last tutorial, a function to translate our species into CONIFERS functional species, and a function to allocate and assign arrays of CONIFERS data types from the plot and plant records in our database. In short, these functions are the data interface functions between the tutorial database and the modules we present here.

First, create a new module and construct a basic skeleton (simple subroutine) for looping over the `stands`. Your subroutine should have a couple of variables for the number of plots and plant records, as well as variable sized arrays to hold the `n_plots` plots and `n_plants` plants. You'll need variables for your database interface (e.g. `ADODB`), code to open the species recordset, and code to open and loop over the units.

```
Sub main()

    Dim return_code As Long
    Dim n_plots As Long
    Dim n_plants As Long
    Dim plots() As conifers.PLOT_RECORD      ' conifers data definition (type)
    Dim plants() As conifers.PLANT_RECORD    ' conifers data definition (type)

    ' load the plot and plants for stand 1
    Dim sql_str As String
    Dim stands_rs As New ADODB.Recordset     ' vba object
    Dim species_rs As New ADODB.Recordset    ' vba object
    Dim conn As ADODB.Connection

    ' open the current connection
    Set conn = CurrentProject.Connection

    ' initialize the species in the shared library
    Set species_rs = New ADODB.Recordset
    sql_str = "species"
    species_rs.Open sql_str, conn, adOpenKeyset, adLockOptimistic, adCmdTableDirect
    species_rs.Index = "sp_code"

    ' issue an sql statement to open the table (or some subset)
    sql_str = "select * from unit"
    stands_rs.Open sql_str, CurrentProject.Connection, adOpenKeyset, adLockOptimistic

    Do While (Not stands_rs.EOF)

        ' initialize the species map in the library from the species table
        Call init_species_lib.load_species(stands_rs("variant").Value)
```

```

    ' grow the sample

    ' print summaries

    ' write out the results to the database

    stands_rs.MoveNext
Loop

    species_rs.Close
    Set species_rs = Nothing

End Sub

```

Make sure you're code is working correctly by examining the output in the Immediate window. Once you have verified your code is working correctly, call the utility function `load_stand`, to load data from the current `stand_rs` record, resize the `plant` and `plot` arrays, and finally, assign values from the database to the resized arrays. To make sure that we have loaded the data correctly, add a `Debug.Print` statement to report the number of plots, and plant loaded from the units, as well as the name of the tree and plots able where the name in the units table represents the file name for the original conifers file.

```

Sub main()

    Dim i As Long
    Dim return_code As Long
    Dim age As Long
    Dim n_plots As Long
    Dim n_plants As Long
    Dim plots() As conifers.PLOT_RECORD      ' conifers data definition (type)
    Dim plants() As conifers.PLANT_RECORD    ' conifers data definition (type)
    Dim summary_rec As conifers.SUMMARY_RECORD ' conifers data definition (type)

    ' load the plot and plants for stand 1
    Dim sql_str As String
    Dim stands_rs As New ADODB.Recordset      ' vba object
    Dim species_rs As New ADODB.Recordset     ' vba object
    Dim conn As ADODB.Connection

    ' open the current connection
    Set conn = CurrentProject.Connection

    'initialize the species in the shared library
    Set species_rs = New ADODB.Recordset
    sql_str = "species"

```

```

species_rs.Open sql_str, conn, adOpenKeyset, adLockOptimistic, adCmdTableDirect
species_rs.Index = "sp_code"

' issue an sql statement to open the table (or some subset)
sql_str = "select * from unit"
stands_rs.Open sql_str, CurrentProject.Connection, adOpenKeyset, adLockOptimistic

Do While (Not stands_rs.EOF)

    ' initialize the species map in the library from the species table
    Call workshop_utils.load_species(stands_rs("variant").Value)

    ' load the data for this database...
    Call workshop_utils.load_stand(species_rs, stands_rs, _
                                   n_plots, plots, n_plants, plants)
    ' print out the results to the intermediate window
    Debug.Print "loaded "; _
                n_plots; " plots and "; n_plants; " plants from # "; _
                stands_rs("Stand").Value; _
                "named "; stands_rs("name").Value; _
                " return_code = "; return_code

    ' grow the sample

    ' use the conifers summary function...
    ' you may not want to use this one for your particular project

    ' write out the results to the database

    stands_rs.MoveNext
Loop

species_rs.Close
Set species_rs = Nothing

End Sub

```

Now that we have successfully loaded data, let's call the CONIFERS library function to impute missing values called `fill_in_missing_values`, by placing the function call between the `load_data` function call and the `Debug.Print` statement.

```

Sub main()

    Dim i As Long
    Dim return_code As Long
    Dim age As Long

```



Figure 3: Loaded unit data from the database in the Immediate window.

```

Dim n_plots As Long
Dim n_plants As Long
Dim plots() As conifers.PLOT_RECORD      ' conifers data definition (type)
Dim plants() As conifers.PLANT_RECORD    ' conifers data definition (type)
Dim summary_rec As conifers.SUMMARY_RECORD ' conifers data definition (type)

' load the plot and plants for stand 1
Dim sql_str As String
Dim stands_rs As New ADODB.Recordset      ' vba object
Dim species_rs As New ADODB.Recordset    ' vba object
Dim conn As ADODB.Connection

' open the current connection
Set conn = CurrentProject.Connection

'initialize the species in the shared library
Set species_rs = New ADODB.Recordset
sql_str = "species"
species_rs.Open sql_str, conn, adOpenKeyset, adLockOptimistic, adCmdTableDirect
species_rs.Index = "sp_code"

' issue an sql statement to open the table (or some subset)
sql_str = "select * from unit"
stands_rs.Open sql_str, CurrentProject.Connection, adOpenKeyset, adLockOptimistic

Do While (Not stands_rs.EOF)

    ' initialize the species map in the library from the species table
    Call workshop_utils.load_species(stands_rs("variant").Value)

    ' load the data for this database...
    Call workshop_utils.load_stand(species_rs, stands_rs, _
        n_plots, plots, n_plants, plants)

    ' call the CONIFERS library function to fill in any missing values
    'Call conifers.ex_fill_in_missing_values(return_code, _
        stands_rs("variant").Value,
        n_plants, plants(1), _
        n_plots, plots(1), _
        11.78, 5.6, 40#)

    ' print out the results to the intermediate window
    Debug.Print "loaded "; _
        n_plots; " plots and "; n_plants; " plants from # "; _
        stands_rs("Stand").Value; _

```

```

        "named "; stands_rs("name").Value; _
        " return_code = "; return_code

    ' grow the sample

    ' write out the results to the database

    stands_rs.MoveNext
Loop

    species_rs.Close
    Set species_rs = Nothing

End Sub

```

Finally, let's put a breakpoint at the `Debug.Print` statement so we can examine individual attributes of the plant records to verify that the values were filled in as we intended. Run the macro, and when the breakpoint is reached, Visual Basic for Applications will stop, allowing you to add variables to the watch window (Figure 5). To add variables to the watch window, right-click on the variable and using the pull-down-menu, select the "Add to Watch..." menu item.

## 5 Fourth Tutorial: Computing Summaries

We could construct our own code to loop over plant and plot records and create any statistics we need by using some simple programming constructs. While this may be what you ultimately do for your inventory, in this tutorial, we call the `ex_update_total_summaries` function in CONIFERS to compute summary statistics for an array of plot and plant records. The `ex_update_total_summaries` function fills in values in a `SUMMARY_RECORD`.

Now that you have the code skeleton for your simple application, you can either add directly to, or add to a copy of, the application we developed in the previous tutorial, by simply adding a CONIFERS `SUMMARY_RECORD` data type and the call to `ex_update_total_summaries`.

```

Sub main()

    Dim i As Long
    Dim return_code As Long
    Dim age As Long
    Dim n_plots As Long
    Dim n_plants As Long
    Dim plots() As conifers.PLOT_RECORD      ' conifers data definition (type)
    Dim plants() As conifers.PLANT_RECORD    ' conifers data definition (type)
    Dim summary_rec As conifers.SUMMARY_RECORD ' conifers data definition (type)

```



Figure 4: Output from fill\_in\_missing\_values function in the Immediate window.

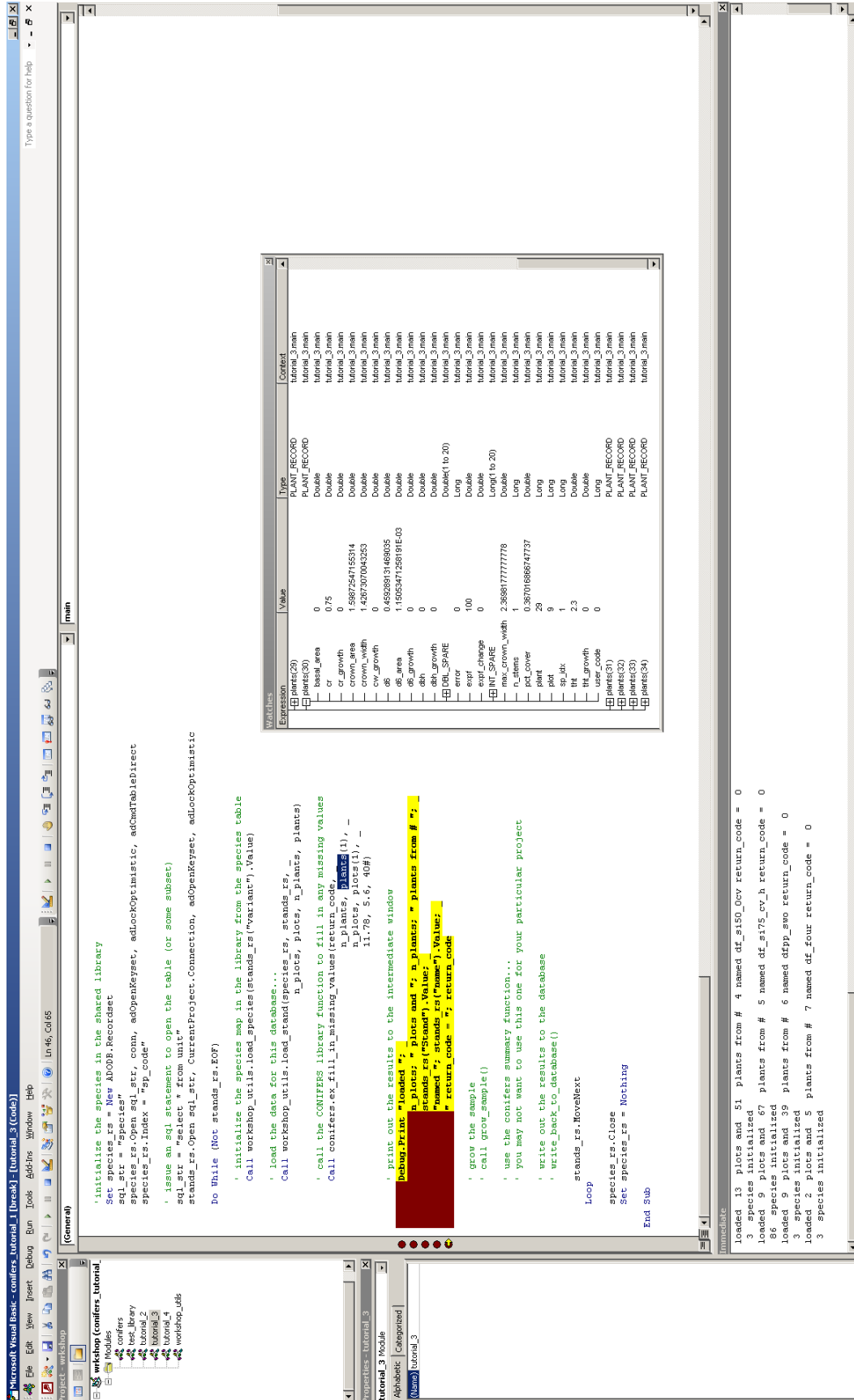


Figure 5: Using the watch window to examine variables in the debugging window.



```

' load the plot and plants for stand 1
Dim sql_str As String
Dim stands_rs As New ADODB.Recordset           ' vba object
Dim species_rs As New ADODB.Recordset        ' vba object
Dim conn As ADODB.Connection

' open the current connection
Set conn = CurrentProject.Connection

'initialize the species in the shared library
Set species_rs = New ADODB.Recordset
sql_str = "species"
species_rs.Open sql_str, conn, adOpenKeyset, adLockOptimistic, adCmdTableDirect
species_rs.Index = "sp_code"

' issue an sql statement to open the table (or some subset)
sql_str = "select * from unit"
stands_rs.Open sql_str, CurrentProject.Connection, adOpenKeyset, adLockOptimistic

Do While (Not stands_rs.EOF)

    ' initialize the species map in the library from the species table
    Call workshop_utils.load_species(stands_rs("variant").Value)

    ' load the data for this database...
    Call workshop_utils.load_stand(species_rs, stands_rs, _
                                   n_plots, plots, n_plants, plants)

    ' call the CONIFERS library function to fill in any missing values
    'Call conifers.ex_fill_in_missing_values(return_code, _
                                             stands_rs("variant").Value,
                                             n_plants, plants(1), _
                                             n_plots, plots(1), _
                                             11.78, 5.6, 40#)

    ' print out the results to the intermediate window
    Debug.Print "loaded "; _
                n_plots; " plots and "; n_plants; " plants from # "; _
                stands_rs("Stand").Value; _
                "named "; stands_rs("name").Value; _
                " return_code = "; return_code

    ' compute the total stats for the all the species together...
    'Call conifers.ex_update_total_summaries(return_code, _
                                             n_plots, _

```

```

        n_plants, _
        plants(1), _
        summary_rec)

' print out the results to the intermediate window
'age = stands_rs("age").Value
'Debug.Print "age = "; age; _
        "rc = "; return_code; _
        "tree_expf = "; summary_rec.tree_expf; _
        " bh_expf = "; summary_rec.bh_expf; _
        "basal_area ="; summary_rec.basal_area; _
        "qmd = "; summary_rec.qmd

' grow the sample

' use the conifers summary function...
' you may not want to use this one for your particular project

' write out the results to the database

        stands_rs.MoveNext
Loop

        species_rs.Close
Set species_rs = Nothing

End Sub

```

Now we've successfully obtained summary statistics for the current set of data, we can then encapsulate this code block within a for-loop to call the `project_sample` function and ultimately transfer back into the database, or to some other application.

For brevity, we no longer put the entire subroutine in the tutorial. These modules are available in the installation and can be imported into the Visual Basic editor, using the "File, Import..." menu item as we did in the first few tutorials.

## 6 Fifth Tutorial: Growing

In this tutorial, we are going to project data using the CONIFERS growth model. To accomplish this, we need to add some variables that are used to control the growth model:

```

Dim x0 As Double
Dim endemic_mortality As Long
Dim sdi_mortality As Long

```



Figure 6: Computing summary statistics using the update\_total\_summaries interface function.

```

Dim use_random_error As Long
Dim random_seed As Long
Dim use_precip_in_hg As Long
Dim use_genetic_gains As Long
Dim age As Long
Dim current_year As Long
Dim n_years As Long

```

```

' set the control variables
x0 = 0# ' for mortality
n_years = 20 ' project for 20 years

```

```

'endemic_mortality = 1 can be 0/1 - 0 means no background
' mortality, if 1 then mortality will then depend on morts_ptr.background values
'sdi_mortality = 1 can be 0/1 - 0 means no sdi competition
' induced mortality, mortality will then depend on morts_ptr.max_sdi values
endemic_mortality = 1
sdi_mortality = 1
use_random_error = 0
random_seed = 0
use_precip_in_hg = 0
use_genetic_gains = 0

```

where variable `n_years` is the number of years that projected and can range from 0 to  $2^{32}$  to facilitate any number of projection lengths. For this workshop will simply project stands forward for twenty (20) years and export the two summaries: one summary at time "age" and twenty years into the future.

```

' grow the sample
Call conifers.project_sample( _
    return_code, _
    n_years, _
    n_plants, _
    plants(1), _
    n_plots, _
    plots(1), _
    x0, _
    endemic_mortality, _
    sdi_mortality, _
    use_random_error, _
    random_seed, _
    use_precip_in_hg, _
    stands_rs("variant").Value, _
    use_genetic_gains, _
    age, _

```

```

        current_year)
Debug.Print return_code

' use the conifers summary function...
' you may not want to use this one for your particular project
Call ex_update_total_summaries(return_code, _
    n_plots, _
    n_plants, _
    plants(1), _
    summary_rec)
Debug.Print "age = "; age; _
           "rc = "; return_code; _
           "tree_expf = "; summary_rec.tree_expf; _
           " bh_expf = "; summary_rec.bh_expf; _
           "basal_area ="; summary_rec.basal_area; _
           "qmd = "; summary_rec.qmd

```

The `Debug.Print` statement prints out the summaries at the end of the projection and your screen should look something like Figure 7.

Instead of projecting a sample for one 20-year-period, we could have projected for twenty one-year periods (annual) and checked the age iteration or any number of stand conditions that are tracked in the `SUMMARY_RECORD` to control the execution path of the subroutine. For example, if we wanted to thin at age 15, we could use an `if-then-endif` statement to check for age (e.g. `age==15`) and performed some thinning function using your own code, or one of the thinning functions in the `CONIFERS` module.

## 7 Sixth Tutorial: Writing Data back to the Database

This tutorial involves creating a table to store our current and projected summaries, adding the code to open and delete any current contents of our projected summaries table, and a few lines of code to assign field values from our `SUMMARY_RECORD`.

Using the design view, create a table definition that includes stand, period, quadratic mean diameter (`qmd`) and trees per acre (`tree_expf`), and the total number of trees per acre over 4.5' tall, the breast height trees per acre (`bh_expf`) and the stand basal area (`basal_area`) (Figure 8):

Don't forget to change the data types to the appropriate field size (e.g. longer for integers and doubles for floating point numbers). Save the table by naming it `projected_summaries`. Don't worry about having a primary key, this is a tutorial. If you want to add a primary key, add it so that it includes the stand and period.

We now need to add the recordset variable for the projected summaries like we did with the species table the other tables:

```
Dim prj_sum_rs As New ADODB.Recordset      ' vba object
```

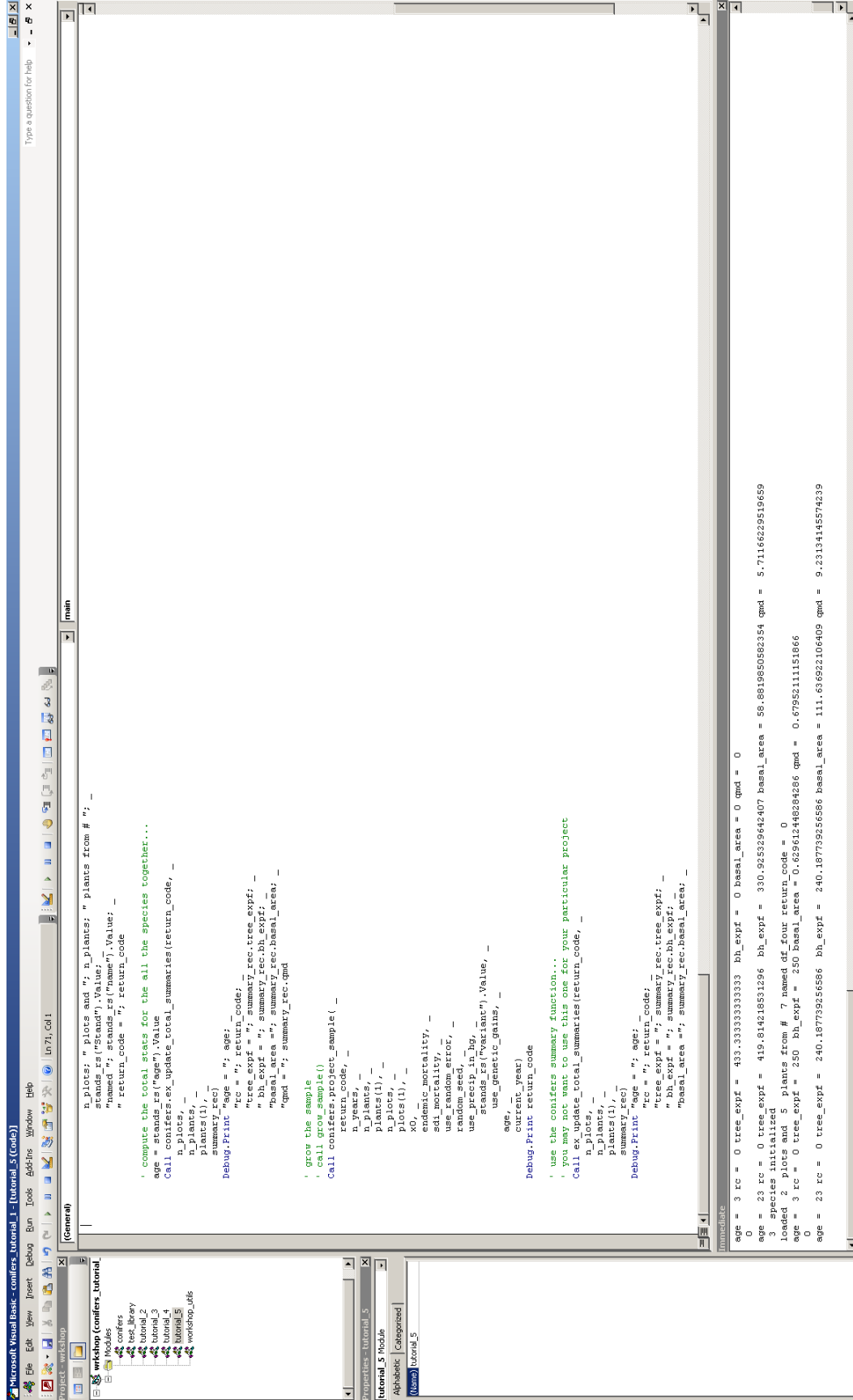


Figure 7: Projecting the units for a single 20 year projection.

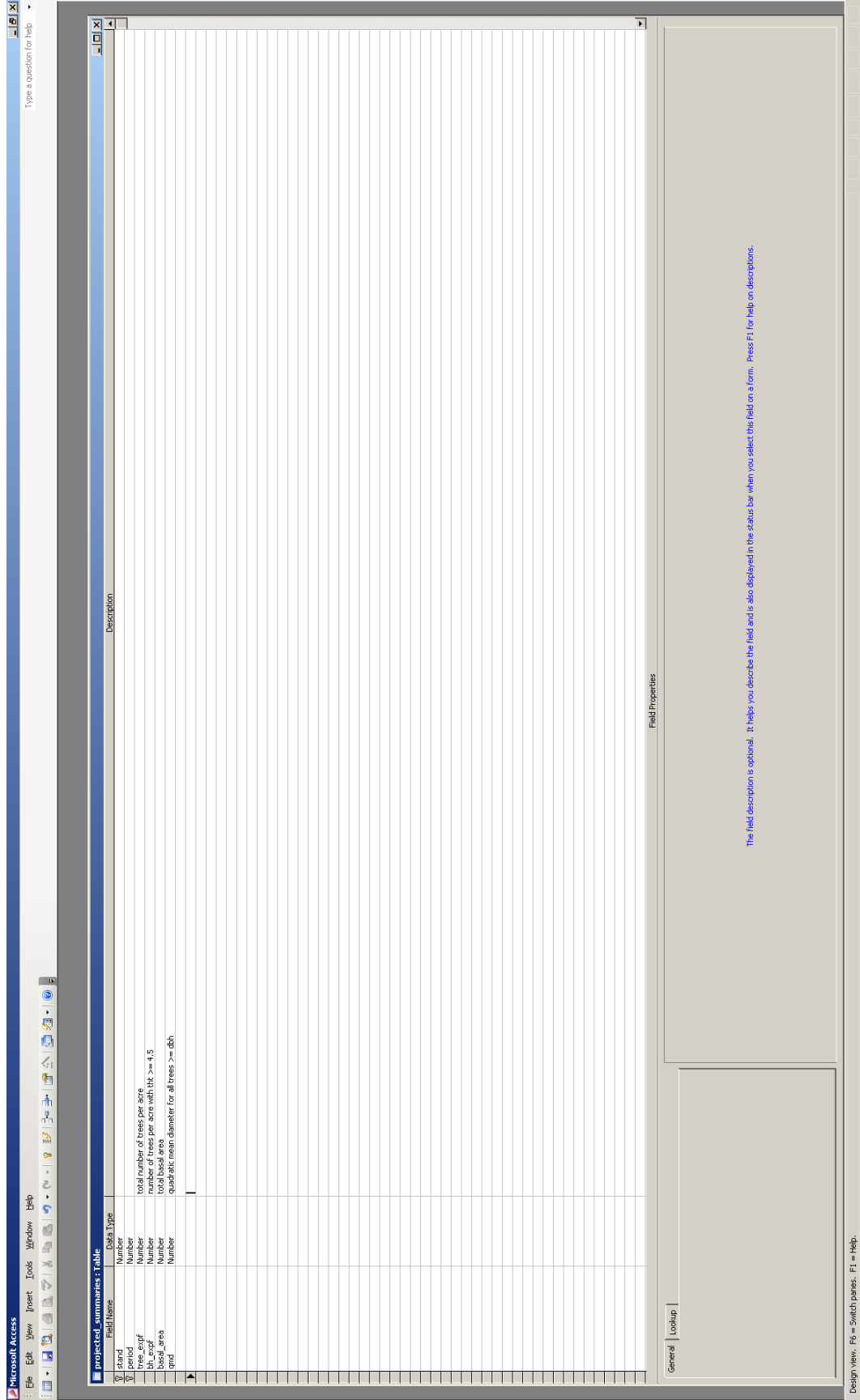


Figure 8: New table definition for storing the projected summaries.

and add a statement to execute an SQL statement handled by the connection (i.e. `conn`) to delete all records from projected summaries, and then open the empty recordset:

```
'initialize the species in the shared library
  Set species_rs = New ADODB.Recordset
  sql_str = "species"
  species_rs.Open sql_str, conn, adOpenKeyset, adLockOptimistic, adCmdTableDirect
  species_rs.Index = "sp_code"

' delete any records from the projected_summaries table
>>  conn.Execute "delete * from projected_summaries"

>>  ' open the project summaries recordset
>>  Set prj_sum_rs = New ADODB.Recordset
>>  sql_str = "projected_summaries"
>>  prj_sum_rs.Open sql_str, conn, adOpenKeyset, adLockOptimistic, adCmdTable

' issue an sql statement to open the table (or some subset)
sql_str = "select * from unit"
stands_rs.Open sql_str, CurrentProject.Connection, adOpenKeyset, adLockOptimistic
```

Since we are only interested in two time periods for this tutorial ( $t=0,1$ ), place a `update_total_summaries` and the code to add a new record to the table, function call after the `Debug.Print` statement immediately after the `fill_in_missing_values`,

```
Debug.Print "age = "; age; _
      "rc = "; return_code; _
      "tree_expf = "; summary_rec.tree_expf; _
      " bh_expf = "; summary_rec.bh_expf; _
      "basal_area ="; summary_rec.basal_area; _
      "qmd = "; summary_rec.qmd

>>  ' write the data to the database
>>  prj_sum_rs.AddNew
>>  prj_sum_rs("stand").Value = stands_rs("stand").Value
>>  prj_sum_rs("period").Value = 0
>>  prj_sum_rs("tree_expf").Value = summary_rec.tree_expf
>>  prj_sum_rs("qmd").Value = summary_rec.qmd
>>  prj_sum_rs.Update

' grow the sample
' call grow_sample()
Call conifers.project_sample( _
      return_code, _
      n_years, _
```



```

        n_plants, _
        plants(1), _
        n_plots, _
        plots(1), _
        x0, _
        endemic_mortality, _
        sdi_mortality, _
        use_random_error, _
        random_seed, _
        use_precip_in_hg, _
            stands_rs("variant").Value, _
            use_genetic_gains, _
        age, _
        current_year)
Debug.Print return_code

```

and a set of calls to `update_total_summaries` and `AddNew/Update` code blocks after the `post-project_sample` function call:

```

' write out the results to the database
prj_sum_rs.AddNew
prj_sum_rs("stand").Value = stands_rs("stand").Value
prj_sum_rs("period").Value = 1
prj_sum_rs("tree_expf").Value = summary_rec.tree_expf
prj_sum_rs("qmd").Value = summary_rec.qmd
prj_sum_rs.Update

```

The output should be similar to Figure 9.

## 8 Seventh Tutorial: Simulation

Replace the single projection with a for-loop that projects the sample for twenty one-year periods, beginning by changing the value of `n_years` from 20 to 1 at the top of the subroutine where we initialize variables,

```

x0 = 0# ' for mortality
>>  n_years = 1 ' project for 1 years
'endemic_mortality = 1 can be 0/1 - 0 means no background
' mortality, if 1 then mortality will then depend on morts_ptr.background values
'sdi_mortality = 1 can be 0/1 - 0 means no sdi competition
' induced mortality, mortality will then depend on morts_ptr.max_sdi values
endemic_mortality = 1
sdi_mortality = 1
use_random_error = 0

```

Microsoft Access

File Edit View Insert Format Records Tools Window Help

Type a question for help

projected\_summaries : table

stand	period	tree_expf	bh_expf	basal_area	qmd
0	275	0	0	0	0
1	1	264.206513182	264.206513182	100.118233133	8.33529805269685
2	0	350	0	0	0
3	1	336.262834959	336.262834959	90.3214033619	7.01765915045344
4	0	316.686666667	0	0	0
5	1	304.237803058	304.237803058	122.327936796	8.58603278884698
6	0	588.461538462	0	0	0
7	1	567.629354114	509.937046422	106.639707711	6.19769720395362
8	0	744.444444444	0	0	0
9	1	641.987470463	641.987470463	138.388795645	6.28671096882142
10	0	433.333333333	0	0	0
11	1	419.814218931	330.92529642	59.1825800893	5.72622327164046
12	0	250	250	0.62861244628	0.6795211151686
13	1	240.187739257	240.187739257	111.636922106	9.23134145574239
14	0	0	0	0	0
15	0	0	0	0	0

Record: 14 of 14

Datasheet View

Figure 9: Projected summaries before and at the end of the projection.

```

random_seed = 0
use_precip_in_hg = 0
use_genetic_gains = 0

```

, and adding the for-loop,

```

' write the data to the database
prj_sum_rs.AddNew
prj_sum_rs("stand").Value = stands_rs("stand").Value
prj_sum_rs("period").Value = 0
prj_sum_rs("tree_expf").Value = summary_rec.tree_expf
prj_sum_rs("qmd").Value = summary_rec.qmd
prj_sum_rs.Update

' grow the sample
>> For i = 1 To 20
    Call conifers.project_sample( _
        return_code, _
        n_years, _
        n_plants, _
        plants(1), _
        n_plots, _
        plots(1), _
        x0, _
        endemic_mortality, _
        sdi_mortality, _
        use_random_error, _
        random_seed, _
        use_precip_in_hg, _
        stands_rs("variant").Value, _
        use_genetic_gains, _
        age, _
        current_year)

' use the conifers summary function...
' you may not want to use this one for your particular project
Call ex_update_total_summaries(return_code, _
    n_plots, _
    n_plants, _
    plants(1), _
    summary_rec)

' write out the results to the database
prj_sum_rs.AddNew
prj_sum_rs("stand").Value = stands_rs("stand").Value
>> prj_sum_rs("period").Value = i

```

```

    prj_sum_rs("tree_expf").Value = summary_rec.tree_expf
    prj_sum_rs("qmd").Value = summary_rec.qmd
    prj_sum_rs.Update

>>     Next i

        stands_rs.MoveNext
Loop

```

which yields the following projected\_summaries contents:

Finally, ensure that include a pre-commercial thinning to reduce the density at age 13, we first add an if-then-elseif block to the loop where we check the age of the stand:

For i = 1 To 20

```

    ' grow the sample for a single year
    Call conifers.project_sample( _
        return_code, _
        n_years, _
        n_plants, _
        plants(1), _
        n_plots, _
        plots(1), _
        x0, _
        endemic_mortality, _
        sdi_mortality, _
        use_random_error, _
        random_seed, _
        use_precip_in_hg, _
        stands_rs("variant").Value, _
        use_genetic_gains, _
        age, _
        current_year)

    ' when the stand is 13 years old, perform a pre-commercial thinning
>>     If (age = 13) Then
>>         Call conifers.thin(return_code, _
>>             n_plants, _
>>             plants(1), _
>>             n_plots, _
>>             plots(1), _
>>             0, _
>>             conifers.DO_EXPF_THIN, _
>>             280#, _
>>             expf_removed, _
>>             ba_removed)

```

Microsoft Access

File Edit View Insert Format Records Tools Window Help

Type a question for help

Record: 14 of 147

Record: 14 of 147

stand	period	tree_expf	bh_expf	basal_area	qmd
4	21	323.155097302	265.46278961	61.6176230299	6.52360495176071
4	22	322.62471723	264.931864030	66.0372352943	6.86167080033273
5	3	744.444444444	0	0	0
5	4	742.955555556	33.2666666667	0	0
5	5	741.466644444	386.4016	0.04693022776	0.146961000307582
5	6	739.986705156	739.986705156	0.97241153764	0.4906950681546826
5	7	738.506731745	738.506731745	3.48624303969	0.930331866278718
5	8	737.029716282	737.029716282	7.8322807484	1.3998443954103
5	9	735.555658845	735.555658845	14.474920877	1.89948848118702
5	10	734.084547527	734.084547527	23.2775617095	2.41119318539195
5	11	732.616378432	732.616378432	33.7969638524	2.90828415618408
5	12	731.151145676	731.151145676	45.4426036411	3.37570374972836
5	13	290	280	22.1847873201	3.81140346529789
5	14	279.44	279.44	28.3456822345	4.31256323480891
5	15	278.88112	278.88112	34.9292689470	4.792053156924063
5	16	278.3235776	278.3235776	41.7648368491	5.24651396455255
5	17	277.766711044	277.766711044	48.7747621351	5.67405621366627
5	18	277.21177622	277.21177622	56.7603795592	6.073956384433197
5	19	276.656755267	276.656755267	62.7041966027	6.446351300403846
5	20	276.103441757	276.103441757	69.4697641485	6.79200965535801
5	21	275.551234873	275.551234873	76.0199474664	7.11211723919412
5	22	275.000132403	275.000132403	82.3143703955	7.408114466880705
5	23	274.450132139	274.450132139	88.326565376	7.68157542818225
6	3	433.333333333	0	0	0
6	4	432.644444444	0	0	0
6	5	431.956333333	99.6004	0	0
6	6	431.270797244	176.713243022	0.07495517652	0.27807048147192
6	7	430.586193428	264.539724804	0.33978520505	0.484566152338095
6	8	429.902639139	341.01375025	0.88925871325	0.683242587511291
6	9	429.220611638	340.331727249	1.7403861191	0.96819878207578
6	10	428.539548193	339.651069304	3.00956492721	1.27444307844637
6	11	427.860546074	338.971571785	4.7041730666	1.59513042980883
6	12	427.18270256	338.293813671	6.8642448333	1.92741296240107
6	13	360.201797025	261.341298816	7.6704748822	2.31976186313027
6	14	349.707497123	300.818680334	10.1741639715	2.6743385530108
6	15	349.182658906	300.266971018	13.0956194353	3.03927989380611
6	16	348.665256964	299.776377076	16.4078974124	3.40300675504263
6	17	348.145713210	299.258934321	20.1353407304	3.77365651068638
6	18	347.627198952	298.739310673	24.2579483779	4.1468095715812
6	19	347.109723840	298.220834051	28.727929154	4.5117188481484
6	20	346.592581072	297.70432389	33.5515667488	4.8825553684962
6	21	346.078724727	297.186865598	38.6619168866	5.24959046570917

Dashboard View

Figure 10: Projected annual summaries before the thinning.

```

>>         End If

' use the conifers summary function...
' you may not want to use this one for your particular project
Call ex_update_total_summaries(return_code, _
    n_plots, _
    n_plants, _
    plants(1), _
    summary_rec)

' write out the results to the database
prj_sum_rs.AddNew
prj_sum_rs("stand").Value = stands_rs("stand").Value
prj_sum_rs("period").Value = age
prj_sum_rs("tree_expf").Value = summary_rec.tree_expf
prj_sum_rs("bh_expf").Value = summary_rec.bh_expf
prj_sum_rs("qmd").Value = summary_rec.qmd
prj_sum_rs.Update

Next i

```

and add the two variables we need to return the number of plants and the respective basal area removed, we need to add two variables that are updated when we call the thinning function:

```

Dim expf_removed As Double
Dim ba_removed As Double

```

To perform the thinning, call the `thin` function with the `DO_EXPF_THIN` argument for the type of thinning. To perform for other types of thinning, see the on-line documentation or look in the interface module for the constants defined associated with thinning.

When we run this "scenario", we get the following results in the `projected_summaries` table:

## 9 Final Thoughts

In these seven tutorials, we have focused on moving data into and out of the CONIFERS library. While these are the basic building blocks, there are no restrictions on how you put these commands together for inventory updates, simulation, and optimization.

While the GUI for the CONIFERS growth model may seem have more functionality, it is limited in it's flexibility and utility for simulation or optimization. The advantage of the DLL are that the user is in complete control. The downside, of course, is also, that the user is in complete control. In fact, all of the functions that CONIFERS itself uses are encapsulated in the functions available using this interface. Of course, you are not limited

Microsoft Access

File Edit View Insert Format Records Tools Window Help

Type a question for help

Record: 14 of 14

View: Datasheet View

Record: 14 of 14

stand	period	tree_expf	bh_expf	basal_area	qcmd
1	4	510.95964350	96.9104591029	5.95750479500857	6.19769720933862
2	5	509.937046422	106.639707711	0	0
3	4	742.955555556	0	0	0
4	5	386.4016	0.04693022776	0.146961000307582	0.146961000307582
5	6	739.986705156	739.986705156	0.97241153764	0.490950681546826
6	7	739.986705156	739.986705156	3.48624303969	0.930331866278718
7	8	737.029718282	737.029718282	7.8322807484	1.3969473954103
8	9	735.555555556	735.555555556	14.4749228877	1.89946848118702
9	10	734.084547527	734.084547527	23.2775617095	2.41119318539195
10	11	732.616378432	732.616378432	33.7996638524	2.90528415518408
11	12	731.151145676	731.151145676	45.4426036411	3.37570374972836
12	13	729.688433384	729.688433384	57.6250659281	3.80516232896247
13	14	728.229465697	728.229465697	69.8512628437	4.19362289761079
14	15	726.773006766	726.773006766	81.7603623923	4.541594633309797
15	16	725.319460753	725.319460753	93.1186234815	4.85165243247385
16	17	723.868821831	723.868821831	103.794043362	5.127345451063008
17	18	722.421084187	722.421084187	113.726688064	5.37250088367319
18	19	720.976242019	720.976242019	122.913696036	5.59062982314586
19	20	697.489011865	697.489011865	127.34474723	5.78573444382568
20	21	676.804325009	676.804325009	131.376681122	5.96573401020407
21	22	658.547957954	658.547957954	135.070689088	6.13230351249419
22	23	641.987470463	641.987470463	138.388795645	6.28671056882142
23	24	433.333333333	0	0	0
24	25	432.644444444	0	0	0
25	26	431.958333333	99.6004	0	0
26	27	431.27097244	176.713243022	0.07495517652	0.27887048147192
27	28	430.586193428	264.539724804	0.33978520506	0.484566152338095
28	29	429.902639139	341.01375025	0.88925871325	0.693242597511291
29	30	429.20611638	340.331727449	1.7403961191	0.96819878207578
30	31	428.539548193	339.651059304	3.00985482721	1.27443378344637
31	32	427.880546074	338.971757185	4.70417306656	1.59513042980983
32	33	427.18270256	338.293813671	6.8642448333	1.92741236240107
33	34	426.536114932	337.617226043	9.48467688988	2.26952659872452
34	35	425.838890480	336.941991991	12.6136576467	2.6198882330176
35	36	425.159596497	336.268107808	16.2525422697	2.976930657421958
36	37	424.484646282	335.595571393	20.3965345996	3.33840781978847
37	38	423.813289139	334.924380260	25.0418721475	3.70251099001433
38	39	423.143203979	334.25453149	30.193386843	4.06775460789048
39	40	422.471691316	333.589020247	35.59396459	4.42934463681684
40	41	421.807249271	332.938603682	41.603963595	4.7856891303851
41	42	421.14190157	332.253012681	47.8468412559	5.1386033786852

Figure 11: Projected annual summaries.

by the functions made available in the DLL. You can create complex scenarios with more complex programming structures and should you wish to obtain additional metrics (e.g. volumes), you can construct subroutines to perform these procedures as well. You are free to use this code and modify it to meet your needs. Enjoy.

## 10 FAQ

### 10.1 Errors

#### 10.1.1 CONIFERS crashes Access

A defect was found in `conifers.bas` (i.e. DLL interface module) that was not updated in the repository before release. The DLL interface subroutine for `fill_in_missing_values` did not include the correct number of arguments (i.e. added the variant argument). The modules, tables, and tutorial documents were updated to correspond with the DLL.