ACD/Labs

ACD/ChemBasic

Version for Microsoft Windows

Tutorial

Getting Started with ChemBasic

Table of Contents

Before You Begin

Thank you for using our free software designed to expand the functionality of ACD/ChemSketch. We have endeavored to produce an easy to use software development kit (SDK).

About This Tutorial

Completion of this tutorial should give you the tools needed to get started programming your own ChemBasic applications and interfacing them to ACD/ChemSketch.

The screen shots shown throughout this reference manual have been taken with a relatively small window size.

The colors and other properties of the window elements described throughout this manual correspond to the Windows Classic Theme (with dismissed gradients) of the operating system's Display Properties.

This tutorial is provided in electronic form, readable with Adobe software. If you cannot locate an index topic you need, please do a text string search for the relevant word or phrase, or related words.

This tutorial assumes that you have a basic familiarity with mouse and file manipulation in Microsoft Windows.

Mouse Conventions

You may perform several actions during your work with this software; the following specific words are used to describe them:

- ***Point to*** means move the mouse pointer to an item.
- ***Click*** means point to an item, and quickly press and release the left mouse button.
- ***Right-click*** means point to an item, and quickly press and release the right mouse button.
- ***Double-click*** means point to an item, and quickly press and release the left mouse button twice.
- ***Drag*** means point to an item, press and hold down the left mouse button while you move the item.
- ***Select*** means highlight or make an interface element active either by clicking it or dragging over it (other actions are possible if specified in documentation). If used in "select the check box", it means that the check box should be marked with a tick (as opposed to "clear the check box" when the check box should be cleared, without a mark).

For More Information…

To see the latest in ACD/Labs software and services, please visit our Web site at

http://www.acdlabs.com

Our Web site is being accessed at the rate of tens of thousands of "hits" per day. There's a reason for this: much is offered through our Web site. We offer free ChemSketch, an ACD/Log*P* Freeware Add-on for ChemSketch, a free ISIS 3D Add-in, free ChemDraw extensions, and a free 2-week demo key for "Interactive Laboratory" sessions where you can run test calculations using Java applets without purchasing software. There are TechSmith Camtasia-based movies which show the operation of many of our software packages (especially ChemSketch). The movies can be run from the ACD/Labs software folder, \\MOVIES.

We are constantly updating the information on our Web site. The Web site will tell you at which scientific conferences you can visit the ACD/Labs booth. You can browse the Frequently Asked Questions page or drop in and "chat" on our newsgroup, which can also be reached via our Web site.

If you would like to stay informed of the latest developments in chemical software at ACD/Labs, please be sure to sign up for e-mail broadcasts at our Web page:

http://www.acdlabs.com/newsletters

How to Contact Us

We are accessible through our Web site, phone, fax, and regular mail, but by far the most popular way to contact us is via electronic mail. Questions on pricing, sales, availability, and general issues should be directed to:

info@acdlabs.com

Technical and scientific support issues should be addressed by visiting:

http://www.acdlabs.com/support

Please tell us the name of the software purchaser; the product name, version number, build number, and license ID of the product you are contacting us about (from the **Help** menu, choose **About** to find this information); as well as a description of the problem you are having. If applicable, please tell us the name of the distributor from whom you purchased the software.

Online Updates

Updates of our Desktop and Enterprise products are made throughout the year. These intermediate releases (bringing the actual version number of a program, for example, from *N*.00 to *N*.01) often contain new functionality along with additional bug fixes and support for new file formats. To check if there is a new update available and to have this sent to you, please contact your local agent or our Technical Support Department. Before calling, we recommend that you have ready the name of the software purchaser, the product name, version number, build number, and license ID of the product you are contacting us about. All Desktop ACD/Labs software contains the capability to have software updates delivered online. You will need the registration numbers of the software and an Internet connection from the same computer on which the software is installed. For more information, refer to the *ACD/Updater User's Guide* located in the ACD/Labs documentation folder, \\DOCS\UP_CLNT.PDF.

1. Introduction

1.1 About ACD/ChemBasic

ACD/ChemBasic is a simple, convenient, and functionally rich *programming language* for the presentation and manipulation of molecular structure, related objects, and all the content of ACD/Labs' current and future programs. ChemBasic is founded on, and fully integrated with, ACD/Labs existing functionality. At the same time, ChemBasic has all of the things a programming language should have: numeric and string variables, arrays, flow control and conditional operators, input/output procedures, *etc*.

ChemBasic inherits from *generic BASIC* and some of its extensions. Most evident is a flavor of Microsoft's Visual Basic for Applications (VBA) and, to some extent, extended ANSI's Full BASIC. However, there are significant simplifications in the main procedural language framework, as well as extensions in subject-specific parts.

To be a universal application that extends the capability of the existing program set, ChemBasic is designed as an *object-oriented language*. This means that all of the chemistry-related things are described as *objects*—that is, specific data structures which correspond to molecules, conformations, *etc*. There is a number of predefined *methods* to manipulate these objects. The hierarchy of the objects and methods is the very essence of ChemBasic.

ACD/ChemSketch goodies no longer use the sequential one-item input form dialog boxes. Now you can design the multi-item input forms for the ChemBasic programs using ACD/Forms Manager. Note that pop-down menus, default input values, automated dating, and obligatory fields are easily available through the Forms Manager.

ACD/ChemBasic can be accessed from the ChemSketch window, whether you are in the Structure or Draw mode. The ChemBasic Organizer is a tool in ACD/ChemSketch that helps you to keep track of your ChemBasic programs. Depending on how crowded your toolbars are, you can set a ChemBasic button to be available on the Structure toolbar, both the Structure and Drawing toolbars, or on the ChemBasic toolbar only.

1.2 Why Do We Use ChemBasic?

Every chemist who deals with a computer on a regular basis has a set of routine operations to be carried out on their molecule, sample, spectrum, *etc*. Thanks to the intuitive interface of ACD/Labs programs, you can significantly reduce the number of repetitive actions. For example, in the ACD/SpecManager modules, macro programming now is a part of the standard toolset. ACD/Labs software has had proven success in many areas such as molecular structure manipulation, substructure searching, optimization of graphical representation (2D cleaning), 3D optimization, calculation of molecular properties, prediction of NMR spectra, *etc*.

However, even with the most up-to-date set of applications, you can come across a number of situations where you *have to* repeat some steps time and again… Have you ever thought how nice it would be to issue one simple menu command in order to perform a series of steps automatically? A common instance is examining of the effect of a certain substituent on a group of 20 compounds systematically: you input first one structure and then another, constantly calculating, adding the substituent, recalculating, and recording—for each compound.

To solve the problem, ACD/ChemBasic is here! It brings customization of our software right to your desktop.

1.3 What Does It Do?

ACD/ChemBasic is designed to (a) automate routine operations within ACD/Labs and (b) extend ACD/Labs functionality. It allows you to customize ACD/Labs software for your own purposes, or construct an interface for external programs to the suite of ACD/Labs programs.

A simple short ChemBasic program can, for example, retrieve a molecule from the ChemSketch window, calculate some molecular property, replace one fragment of it with another, and then recalculate the property. Or, a ChemBasic program can automatically perform conformational analysis and depict the most probable 3D-structures for the molecules you indicated. Or anything else… you are limited only by your imagination!

In this tutorial, we shall consider the following simple yet illustrative examples:

- Retrieve all molecular structures drawn on a ChemSketch page and generate their IUPAC names, one after another [Chapter 2].
- Create a series of homologous oxycarbonic acids, perform a 3D-optimization, depict the resulting structures in the ChemSketch window, and mark all of the structures in which the hydroxyl group is separated from the carboxyl carbon by a distance of less than 5 Angstroms [Chapter 3].

1.4 Show Me Examples!

A savvy programmer knows that often the best way to start is to find a template, and then modify it. Some real-life examples of ChemBasic (source codes) are bundled with ACD/ChemBasic. The ChemBasic sample programs are as follows:

- ***Label Printer***—creates and prints out glassware labels for the structures drawn in a ChemSketch session.
- ***SugarNames***—transforms short-hand string notation of a specific class of structures (sugars) to the chemical structures in the correct conformations; that is, it takes a formula like •GalNAc(alpha1-3)Gal(beta1-4)[Fuc(alpha1-3)]Glc(betaOMe) and supplies you with the most probable 3D-structures.
- ***LogK Predictor***—predicts stability constants for the compellation of amine- and/or carboxylato-containing organic molecules with up to 31 metal ions.
- ***Molecular 3D-editor***—is a versatile set of powerful utilities which makes building and modifying 3-dimensional molecular models easy (change torsional angle, invert enantiomeric center, analyze ring puckering and polyhedral shapes, export and import in various molecular formats, and much more).

An additional source for the examples of ChemBasic programs is a set of GOODIES which are now distributed with ACD/ChemSketch. These are described in Chapter 5. Each Goody is a separate .BAS program that can be copied and modified to suit your purposes.

1.5 Why Is It "Basic"?

ACD/ChemBasic is based on the good old "Beginners All-purpose Symbolic Instruction Code" (the original acronym for BASIC). Basic is widely accepted as the most convenient language for non-programmers; however, it is also powerful enough for programmers. Modern versions of BASIC have proved their success in various fields—it is enough to mention Microsoft Visual Basic and Visual Basic for Applications.

ACD/ChemBasic is the Chemical "Beginners All-purpose Symbolic Instruction Code". It inherits from native BASIC, simplicity and ease of use—for the chemists, not for the programming guru; it adds to BASIC certain chemistry-related objects in a manner that is "common sense" to chemists.

1.6 How Does It Work?

Technically, ChemBasic is not only a programming language; it also consists of a language interpreter and run-time library (RTL) which are integrated with ACD/ChemSketch. To run a ChemBasic application, you should have ACD/ChemSketch running. You can create a ChemBasic program with any text editor and save it to a disk file. To run this file, in the ChemSketch window, from the **File** menu, choose **Run ChemBasic**, and then in the **Run ChemBasic** dialog box that appears, specify the name and location of the program (.BAS), and click **OK**.

2. Getting Started: Say Hello!

2.1 Objectives

In this chapter, you will learn:

- How to create a ChemBasic program;
- What are ChemBasic variables and data types;
- How to design a dialog for users of your program;
- How to access ACD/ChemSketch pages and their content;
- How to implement logical control and loops in a ChemBasic program; and
- How to generate IUPAC names for the molecules.

Traditionally, one's first program should say Hello to the world. Since the ChemBasic world is inhabited with molecules, let's have our first application say Hello to the molecules!

2.2 Create the Program Framework

First of all, we will create a framework of the application. A ChemBasic program consists of functions and subroutines (procedures). A subroutine is a section of the program that performs a particular task. Programs consist of modules, each of which contains one or more routines. The term routine is synonymous with procedure, function, and subroutine. All these start with the key word **`Function`** or **`Sub`** and end with the **`End Function`** or **`End Sub`** statements. There can be many procedures in a ChemBasic program but one and only one function should start with the special name **`Main`**. It may be the first procedure in the program, but not necessarily.

> **Note** For the information on what is the difference between a **`Function`** and a **`Sub`**, refer to the *ACD/ChemBasic online Help* located in the ACD/Labs software folder (CHEMBAS.CHM).

In our simple program, the framework for the program will be:

```
Function Main As String

      'The code will be written here

End Function
```

> **Important** ChemBasic does not take into account whether you use capital or lower-case letters in the keywords or variable names. `Function`, `function`, `FUNCTION`, and even `fUnCtIoN` mean exactly the same. So, it is up to you how to write them.
> In this tutorial, all the keywords begin with a capitalized letter, which is common in various flavors of BASIC.

There are two more things to explain: first, the keywords in our function (**`As String`**) which appear after the declaration **`Function Main`**. This means that the function will return a string value. Functions always return some value after execution, in contrast to subroutines.

So, the special variable **`Main`** will contain a string value after the function is executed. The calling program may use the value that is returned by the function for some particular purpose. The calling program for **`Function Main`** is ChemBasic itself; it will use the value in calculation or display the value to the user in an information box, which appears on the screen just after completing the ChemBasic application. A good habit in ChemBasic programming is to declare the main function as **`String`** (you *may* also declare it as **`Integer`**, **`Double`**, *etc*.) because this gives you an opportunity to embed any error messages or final remarks into that string.

The second thing concerning the above three lines of code is the comment line. Everything between the apostrophe (‘) and the end-of-line will be ignored by Basic. This is very useful feature, allowing you to clarify your programs and make them much more readable, when you are trying to make sense of them after a month or two.

> **Note** There exist two other forms of comment: the **REM** statement and the multi-line C-style `/* …. */` comments. For more information, refer to the *ACD/ChemBasic online Help* located in the ACD/Labs software folder (CHEMBAS.CHM).

2.3 Hello, Everybody!

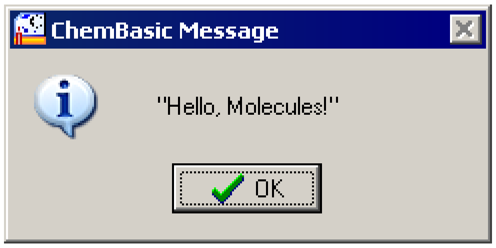
Now, we can fill in the above template with real code.

The simplest way to say Hello is just as follows:

```
Function Main as String

            Main="Hello, Molecules!"

End Function
```

So, to get started, create a text file HELLO.BAS that contains exactly the three above lines. Save it, and start ACD/ChemSketch. From the **File** menu, choose **Run ChemBasic**. In the **Run ChemBasic** dialog box that appears, specify the location of the created HELLO.BAS file, and then click **OK**.

Your Hello will appear on the screen as:



"Hello, Molecules!"

The trick is that, as we mentioned a page or so above, ChemBasic will show the value returned by the main function in its “info box” after executing the application.

Naturally, it is not a truly honest way. Such a trick may be used only once per program (because only one main function may be present). Let us design our greetings in a more universally-applicable way. ChemBasic has special built-in functions that support dialog with the user: **`MessageBox`**, **`UserIOBox`**, and **`UserIO`**. We shall use **`MessageBox`** for our purposes.

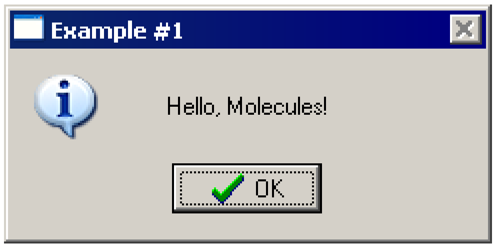
Note For more information on other Input/Output functions, refer to the *ACD/ChemBasic online Help* located in the ACD/Labs software folder (CHEMBAS.CHM).

Let's write the code:

```
Function Main As String
      Dim Answer As Integer

            Answer=MessageBox("Hello, Molecules!","Example #1", MBB_OK
            +MBI_INFORMATION)
      Main="Demo completed"

End Function
```

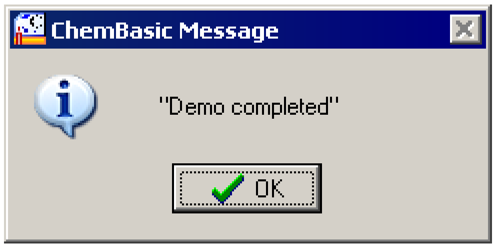
Important Currently, ChemBasic does not support multi-line statements. You have to write long text in a single line. However, in this tutorial, for readability purposes, long texts may appear over several lines. In such cases, all the lines except for the first one will start directly from the left margin of the page. Be careful when transferring such lines into the ChemBasic executable BAS files—you should merge the lines into a single one.

Once you have transferred the above code to HELLO.BAS and saved it as a text file (taking care to avoid multiple line statements), run this program. You will see:



Hello, Molecules!

and then:



"Demo completed"

The first dialog is exactly what we wish. The second one displays the result of the **`Main`** function that we assigned in the same way as it was done in the first sample code.

In this second program, the statement **`Dim`** which appears just after the **`Function`** header declares the variable **`Answer`** which is of the type **`Integer`**. What does it mean?

Variable is a name that you assign to some storage place in memory. Once introduced, it may be then filled (or re-filled) and used for any purposes. Each variable should belong to a pre-defined data type, which differ by internal formats and allowed manipulation methods.

Conventional data types include:

• **`Boolean`**	Boolean variable values are only True or False.
• **`Integer`**	16-bit-long integer number
• **`Long`**	32-bit-long integer number
• **`Double`**	64-bit-long floating-point number
• **`String`**	variable-length string

> **`Note`** For more information on the data types and declarations, refer to the *ACD/ChemBasic online Help* located in the ACD/Labs software folder (CHEMBAS.CHM).

In ChemBasic, all of the variables that you use in the program should be explicitly declared. If you use a non-declared variable, it will be considered an error. To declare a variable, you use the **`Dim`** statement, just as shown above for the integer variable **`Answer`**.

Returning to our example program. In the **`Integer`** type of variable, we need to receive the value returned by the function **`MessageBox`**. This value is a return code—it indicates which button the user has just clicked in the dialog message box. At the moment, its value does not concern us greatly. However, there is no possibility to invoke a function without explicit assignment of its value to some variable. In a further example below, we will use this return code.

A few words on the function itself. **`MessageBox`** displays a standard dialog box, which contains the message (first parameter in a call to function**`;`** here `"Hello, Molecules!"`) and the header (second parameter; here `"Example #1"`). The buttons appearing in the dialog box are regulated by the value of the third parameter (here the user will see only the **OK** button and an exclamation sign (ℹ) indicating the information box).

> **`Note`** For the information on the list of parameters and button controls of the function **`MessageBox`**, refer to the *ACD/ChemBasic online Help* located in the ACD/Labs software folder (CHEMBAS.CHM).

2.4 ChemBasic Objects

However, your first Hello to the molecules was not too polite—we said it to all of them at once. Everyone has a name, though… how about asking for the molecule's name and sending her (him) a personalized message? Let's say "Hello, propane!" just like "Hello, John!" Before we do so, though, we need to understand how molecules and other things are different from simple numbers and strings as represented in ChemBasic.

Because it is a super-set of the BASIC programming language, ACD/ChemBasic includes not only the data types available in standard BASIC, but also certain extensions that are both general-purpose and chemistry-related. All extensions are represented by objects, that is, predefined data structures useful for dealing with sketch pages, molecules, conformations, *etc*. In short, an object is any entity you manipulate which cannot be described as a simple primitive of conventional data types like `integer`, `single`, `Boolean`, *etc.*

There is one generic object type, named `Object`, which is used to manipulate a given entity. It is intended to refer to any actual object of a predefined type. The exact nature of the object is hidden in its type name. The types (and names) are defined within ChemBasic itself; you cannot create your own type. Often (in fact, always), there is no need to indicate a particular object type—ChemBasic determines it when you invoke some function for creating or getting the object. All you deal with are object variable names which thus serve as handles for manipulating entities. For example, the line:

```
            Dim Doc As Object
```

will reserve an object variable named “`Doc`”.

Each object has its own set of methods, that is, procedures which you may apply to it. These procedures are carefully designed for ease of manipulating this particular object (type), without annoying details. Only they are applicable to the object. If you try something else, you will get an error.

Consider an analogy with conventional numbers. Imagine that you have two conventional string variables, **`a`** and **`b`**. Let **`a`** be "good" and **`b`** be "job". Applying the addition operation to them, **`a+b`**, gives "good job". However, if you try division, **`a/b`**, this is an absurd action, and you will get an error. In fact, addition is a method which conventional BASIC allows you to apply to strings, whereas division is not permitted for strings. Furthermore, BASIC strictly defines all of the operations and functions (methods) which are applicable to string objects. It is the type of variable that you have set which distinguishes between legal and forbidden operations. For example, if you set **`a`** and **`b`** as the conventional integer type, they could have the values, say, "10" and "2". Both addition and division would make sense for this type and are thus permitted. Likewise, ChemBasic permits certain things to be done for some of its own objects but not for others.

Methods are simply special kinds of BASIC functions and subprograms that take an object’s arguments. They are invoked by typing a period (“.”) followed by the method’s name just after the object's own name:

```
      SomeObject.ThisObjectMethod(parameters)
```

This may seem like a digression, but we are interested in ChemBasic objects which represent molecules in the ChemSketch window. And in order to access them, we need objects that represent the ChemSketch window itself and its content.

2.5 ChemSketch Pages and Collections

In ChemBasic terms, your currently opened ChemSketch document is represented by the object called **`Document`**, which is a *collection* of objects of the type **`Page`**. This means that **`Document`** is a set of member objects, and that each of them is a **`Page`**; you may easily count them, get any member object, delete it, or add a new one by invoking the collection’s (here, the **`Document`**’s) own methods.

> **`Note`** For a more detailed discussion of collections, refer to the *ACD/ChemBasic online Help* located in the ACD/Labs software folder (CHEMBAS.CHM).

For example, the text:

```
      DocumentObject.Count
```

forces ChemBasic to count the members of the collection.

You can write something like:

```
Dim Npages As Integer 'reserve an integer variable
Dim Doc As Object       'reserve an object variable

      'Get Doc somewhere
      Npages=Doc.Count  'count pages
      Doc.AddEmpty      'add blank page
      Doc.AtRemove(1)   'remove 1st page of document
```

> **Note** For the description of the collection methods **AddEmpty** and **AtRemove**, refer to the *ACD/ChemBasic online Help* located in the ACD/Labs software folder (CHEMBAS.CHM).

One useful tip: the special statement **With** drastically reduces the volume of code. After you write **With SomeObject**, you do not need to mention **SomeObject** when invoking its methods. You only need to type the method names preceded by a point. Re-writing our example:

```
Dim Npages as Integer 'reserve an integer variable
Dim Doc as Object       'reserve an object variable

      'Get Doc somewhere …

      With Doc

            Npages=.Count     'count pages
            .AddEmpty         'add blank page
            .AtRemove(1)      'remove first page of document

      End With
```

> **Tip** Currently, you may not use the nested **With** statements in ChemBasic.

The special function **ActiveDocument** returns a handle of the active document. So, the following code does all of its actions with your current document:

```
Dim Npages As Integer 'reserve an integer variable

      With ActiveDocument

            Npages=.Count     'count pages
            .AddEmpty         'add blank page
            .AtRemove(1)      'remove first page of document

      End With
```

(Note that we omitted the object variable and simply mentioned **ActiveDocument** in the statement. If you do define **ActiveDocument** using “`Dim ActiveDocument as Object`”, then you will get an error message about an uninitialized variable.)

Enough on documents; now on to pages. The **`Page`** object is a member of a collection, as you have seen. But it also is a collection itself; moreover, it contains a number of collections corresponding to different kinds of ChemSketch drawings (text boxes, rectangles, *etc*.). There are **`Page`** object methods, which return those daughter collections. What is of interest to us now is the method **`Diagrams`** of **`Page`** which returns a **`Diagrams`** object—a collection of all molecular diagrams drawn on that page. Once you get a collection, you can add, remove, or get any particular member objects.

2.6 Hello, Mr. Propane!

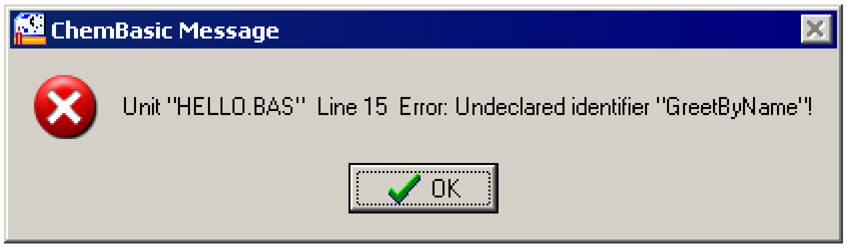
Back to our mission. As a first attempt, we shall greet the first molecule in the ChemSketch active window. To accomplish this task, we access the active page of the active document and get its first molecular diagram. For convenience, we will encapsulate all the work with diagrams in a separate subroutine; our main function will be responsible only for asking the user if he or she would like to address ChemSketch's molecules personally and then, if the answer is "yes", calling a separate procedure for doing this. Look at:

```
      Function Main as String
      Dim Answer as Integer

      'Say hello to all

      Answer=MessageBox("Hello, Molecules!","Example #1", MBB_OK +
MBI_INFORMATION)

      'Ask user if (s)he wants to pronounce the name(s)

      Answer=MessageBox("Say Hello to all? ","Example #1", MBB_YESNO +
MBI_QUESTION)

      If Answer=MBR_YES Then

      'yes, (s)he wants to be greeted by name
      Call GreetByName
      Main=" Everybody welcomed. "

      Else

      'No, leave this molecule out
      Main="Sorry, we leave them. "

      End If

      End Function
```

In this example, we applied the previously unused value which is returned by **`MessageBox`**. Also, we implemented a *conditional statement* **`If… Then… Else`** to treat alternative answers from the user.

> **Note** For more information on the conditional statements **`If… Then… Else`** and **`Select Case…`**, refer to the *ACD/ChemBasic online Help* located in the ACD/Labs software folder (CHEMBAS.CHM).

If you create a new text file called HELLO.BAS from the above example, and from the **File** menu, choose **Run ChemBasic** when inside of the ChemSketch window, you will receive an error message about an undeclared identifier:

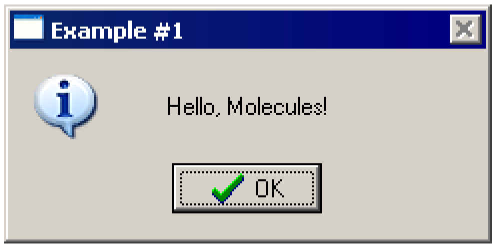


Unit "HELLO.BAS" Line 15 Error: Undeclared identifier "GreetByName"!

This just means that we are missing the **`GreetByName`** subroutine. Here is a first draft of how we plan to treat the diagram:

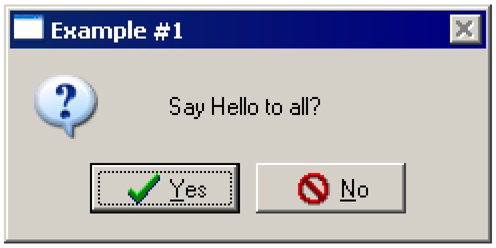
```
      Sub GreetByName
      Dim Diagram as Object, Answer as Integer, MolName as String

            With ActiveDocument.ActivePage.Diagrams

      MolName=.Item(1).GetIUPACName
      Answer=MessageBox("Hello, "+MolName+"!","Example #1",
MBB_OK+MBI_EXCLAMATION)

            End With
      End Sub
```

Add this subroutine to the main function in HELLO.BAS, draw a molecule such as propane in the ChemSketch window, and then run the ChemBasic program. You will first see:



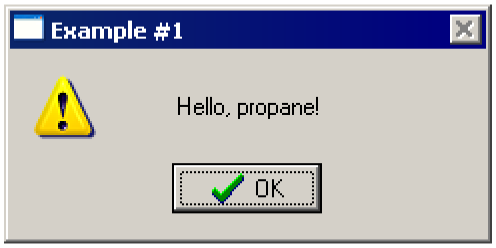
Hello, Molecules!

and then after clicking **OK** a question appears:



Say Hello to all?

Click **Yes**, and then after a pause the message will be displayed:



Hello, propane!

Let's look at the **`GreetByName`** subroutine more thoroughly. First, we retrieved the Diagrams collection through the call **`ActiveDocument.ActivePage.Diagrams`** in the **`With`** statement. Second, we retrieved a structural diagram from that collection using the Item method. Which structural diagram? The first, as you can see from the number in parentheses: "`Item(1)`". And, finally, we applied a method called **`GetIUPACName`** to the Diagram object. This method returns the string which contains the IUPAC name generated by ACD/ChemBasic Run-Time Library (in fact, most if not all of the object methods are internally translated into RTL calls). Note that our code is fully equivalent to a full form (without the **`With`** shortcut):

```
MolName=ActiveDocument.ActivePage.Diagrams.Item(1).GetIUPACName
```

> **Note** The **`GetIUPACName`** method returns the right name only if you have the ACD/Name Freeware add-on or ACD/Name commercial version installed, otherwise an empty string is returned. If you want to work through a very similar example, but one that is entirely contained in the ChemSketch properties, please use the **`GetMolWeight`** method. The HELLOMW.BAS example was written using this. The code is given in the Appendix.

Let us repeat, for clarity, what ChemBasic does here:

1. Retrieves the ACD/ChemSketch active opened document;
2. Retrieves its active page (the visible ChemSketch window);
3. Retrieves all molecular diagrams from the page;
4. Extracts the first diagram from the collection;
5. Applies the **`GetIUPACName`** method to that diagram; and
6. Outputs the name and greeting.

> **Note** For more information on the methods of the Diagram object, refer to the *ACD/ChemBasic online Help* located in the ACD/Labs software folder (CHEMBAS.CHM).

2.7 Hello to Several Individuals

The final step we should make is to address all the molecules on the page, not only the first one. To do so, we should implement one of the ChemBasic cycle-building constructions. The most common form in other programming languages is the **`For… To`** statement:

```
    Dim i As Integer

    For i=1 To 10
    Do_Something
Next I
```

This means that all of the statements enclosed between the **`For... To`** and **`Next`** statements will be repeated 10 times, and the variable `i` (*loop counter*) will be step-by-step incremented by one, starting from unity, until it reaches a value of ten, at which point the loop is exited. In our example, we should repeat a cycle for each diagram. So, first we count the number of diagrams:

```
      Sub GreetByName

      Dim Answer as Integer, MolName as String
      Dim i,Ndiagrams as Integer

            With ActiveDocument.ActivePage.Diagrams

                  'Count the diagrams
                  Ndiagrams=.Count

            For i=1 To Ndiagrams

                  MolName=.Item(i).GetIUPACName

                  Answer=MessageBox("Hello, "+MolName+"!","Example
#1",MBB_OKCANCEL+MBI_EXCLAMATION)

                  If Answer= MBR_CANCEL  Then Exit For

            Next i

            End With

      End Sub
```

Replace the subroutine in HELLO.BAS by the above subroutine, and save the text file. Draw another molecule on the ChemSketch page that contains propane. When you run this ChemBasic program, you will see each molecule individually greeted.

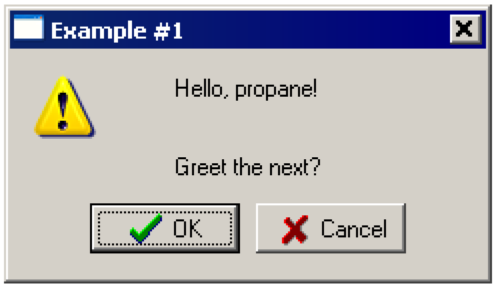
Note, however, that we have no real need for the loop counter **`i`** in the subroutine. All we need are the members of a collection. For just such a case, ChemBasic has the so-called *object-loop* **`For Each`**:

```
      For Each MemberObject In CollectionObject

      MemberObject.CallSomeMethod

      Next MemberObject
```

This means that all of the statements enclosed between the **`For Each… In`** and **`Next`** statements will be repeated with each member object of a collection. So, we re-write our code as:

```
      Sub GreetByName
      Dim Diagram as Object, Answer as Integer
      Dim NextName,Tail as String

            Tail=Chr(13)+Chr(13)+Chr(13)+"Greet the next?"

            With ActiveDocument.ActivePage.Diagrams


            For Each Diagram In .Self

                  NextName=Diagram.GetIUPACName

                  Answer=MessageBox("Hello, "+NextName+"!"+Tail,"Example
#1", MBB_OKCANCEL+MBI_EXCLAMATION)

            If Answer= MBR_CANCEL Then Exit For

            Next Diagram

            End With
End Sub
```

Save the code into a file and run it after drawing, say, propane, ethane, and methane on a single ChemSketch page. The first individual greeting looks like:


Example #1

Hello, propane!

Greet the next?

Note the method **`.Self`** in the above subroutine. It is intended to return the object itself when applied to any object—quite a convenient thing in combination with the **`With`** construction, as shown.

Note For information on the loop statements, refer to the *ACD/ChemBasic online Help* located in the ACD/Labs software folder (CHEMBAS.CHM).

Now is the final step. The situation might occur where the RTL cannot return the IUPAC name, for whatever reason. In such a case, **`GetIUPACName`** will return an empty string or a string containing spaces. We should check for this and either skip the molecule or call it by another name. This is implemented as follows:

```
      If Trim(MolName)="" Then MolName=" though I can not pronounce your
name"
```

(The ChemBasic function **`Trim`** removes the leading and trailing spaces from an argument string).

Whew… It seems that we finally said Hello to all them! Congratulations!

Note See the full text of the program in the Appendix.

3. The Fruits of the Molecular Tree

3.1 Objectives

In this chapter, you will learn:

- What the ChemBasic molecular objects tree is;
- How to access its branches;
- How to create new molecular objects;
- How to link molecular objects with ChemSketch document pages; and
- How to analyze molecules and conformations.

3.2 Design Statement

Let us consider a real-life example. Suppose we are interested in the oxycarbonic acids, of the family $HO–(CH_2)_n–COOH$, which have a short distance between the carbonyl carbon and the hydroxy oxygen. Such a situation is possible if we are designing receptors for these acids and are interested in simultaneously binding a molecule by the carboxy and hydroxy ends.

With ACD/ChemBasic, this leads to the following design statement:

- Generate the molecules $HO–(CH_2)_n–COOH$ with n, say, from 1 to 10;
- Find their optimized structures;
- Find the distances (H)O–C(OOH) and, if the distance is less than 5 Angstroms, mark the structure; and
- Show all of the structures on one ChemSketch page.

To accomplish the mission, take a look at ChemBasic molecular objects.

3.3 Molecular Objects Reference

To describe the molecular world, ChemBasic uses the whole tree of its own molecular objects. They are related to, but not identical with, ChemSketch molecular diagrams (**`Diagram`** objects). All we are dealing with in ChemSketch is some depictions of molecules—their 2D or 3D "shadows". Atoms in ChemSketch diagrams have labels, screen coordinates (in centimeters or inches), and so forth. Atoms in ChemBasic 'molecules' have nuclear charge and atomic masses, world 3D coordinates (in angstroms), *etc*. Naturally, these objects are related and this ensures the opportunity to create a molecular pattern in ChemSketch. But to deal with this pattern as with a real molecule, we should convert it into a ChemBasic entity.

> **`Note`** The difference between ChemSketch and ChemBasic molecular objects was created intentionally. ChemSketch contains many features, such as the automated control of atomic valence, which are very convenient for the drawing application. However, these controlled features restrict the freedom of operation with molecules, which is necessary in molecule-manipulating language. So, the ChemSketch Diagrams and ChemBasic molecular tree are clearly separated but there exists a rigorous mechanism of translation back and forth between them.

In the ChemBasic world, everything is composed of atoms (**`Atom`** object). Atoms may be collected into groups, which are called **`Assembly`** objects. The connection scheme of the atoms of a given assembly is called **`Molecule`**. A collection of atomic world (x, y, z Angstroms) coordinates is called **`Conformation`**. Any **`Molecule`** or **`Conformation`** is based on some **`Assembly`**.

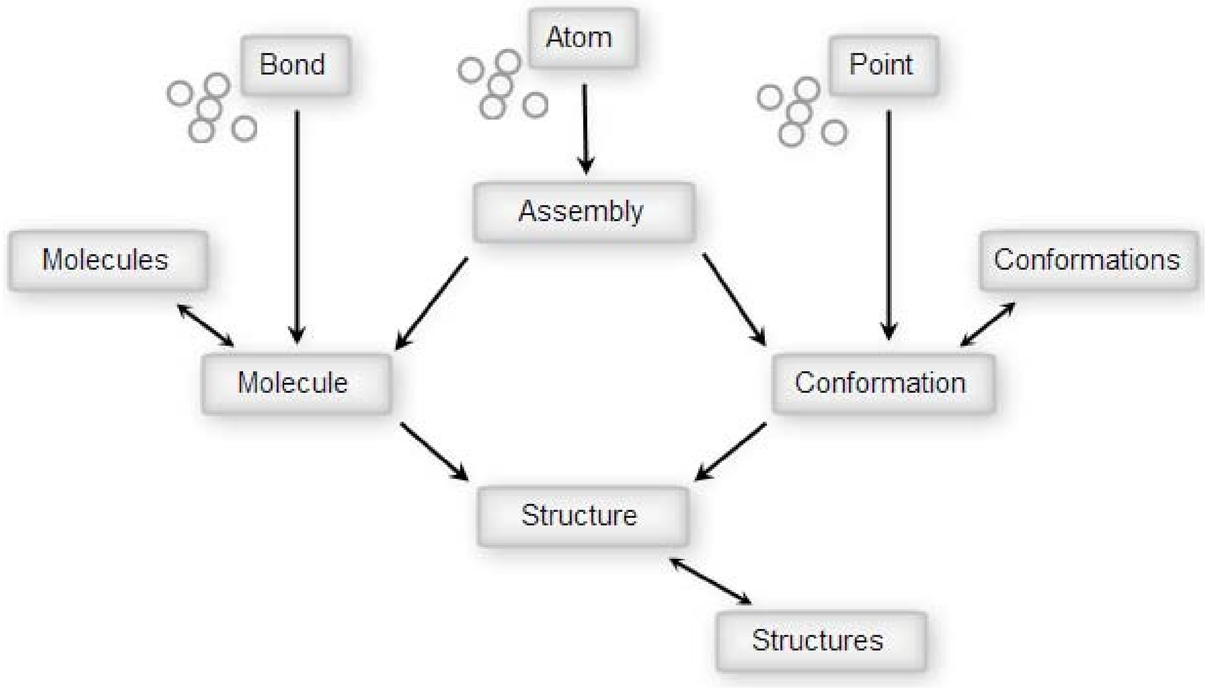
ChemBasic automatically traces and updates the **`Molecule`** and **`Conformation`** collections (*autocollections*), which contain all the molecules and conformations attached to a particular **`Assembly`**. You may associate some **`Molecule`** and **`Conformation`** objects attached to the same **`Assembly`**. The resulting composite object is called **`Structure`**. The **`Structure`** objects are also auto-collected into a **`Structures`** object, attached to the **`Assembly`**. Finally, all of the **`Assembly`** objects in the current ChemBasic session are auto-collected into the object **`Assemblies`**. Thus, you may access any **`Assembly`** (and its derived tree) as **`Assemblies.Item(i)`**.

> **`Note`** In fact, ACD/ChemBasic manages its own pool of molecular data objects and collections. All of your operations are translated into operations with objects of this pool just like ACD/ChemSketch does with its drawings and pages. This allows you to keep the above-mentioned auto-collections. The predefined molecular-level objects create an abstraction layer which takes care of your objects—the abstraction layer manages memory, traces objects disappearance, their mutual influence, *etc*. However, you do not need to concern yourself with the internal mechanics—you deal only with your own molecular trees.

This paradigm, though somewhat unusual at first glance, follows the way we describe the molecular world verbally. The overall basis is atoms and their groups. The same group of atoms may have different connectivities. It means that you have a convenient way to describe isomers as different **`Molecule`** objects attached to the same **`Assembly`**. Also, the atoms in the group may have different coordinates, and you have a convenient way to describe the conformers as different **`Conformation`** objects attached to the same **`Assembly`**. Note that neither the conformation refers to the bonding scheme, nor the valence scheme refers to the atomic coordinates. To integrate these kind of data, you should introduce one more object, **`Structure`**.

As you may see, all of the objects form a sort of tree. (Or would *forest* be more appropriate?)

ChemBasic's Molecular Tree



Let us examine the branches of the tree in more detail.

Atom

The most fundamental object in the hierarchy. It represents isolated unconnected atoms. Atoms may be collected into an **`Assembly`** object.

Bond

Indicates the connection of two atoms. Bonds may be collected into a **`Molecule`** object. The same bond may be included in many molecules. To create a bond, use the ChemBasic **`NewBond`** function. To destroy a bond, use the **`Kill`** function. To create or break the bond in one particular molecule, use methods of the **`Molecule`** object.

Point

Represents a point in space occupied by an atom. Points are collected into a **`Conformation`** object. Points may not be removed from or added to a **`Conformations`** collection. The user can access points directly or through a **`Conformation`**.

Assembly

A group of **`Atom`** objects which serves as a basis for all molecular entities. Any atom can be associated with many assemblies. One may deal simultaneously with, say, naphthalene and its constituent benzene rings as with three different molecules based on three assemblies made from the same atoms (*i.e.*, having the same handles). Any **`Assembly`** may have multiple attached **`Conformation`**, **`Molecule`**, and **`Structure`** objects, all of which are (auto) collected into the **`Conformations`**, **`Molecules`**, and **`Structures`** collections. If the base **`Assembly`** is modified, this involves all the child objects. For example, removing an **`Atom`** from the **`Assembly`** automatically destroys all the related **`Point`** and **`Bond`** objects and updates all other related objects.

Molecule

A valence scheme built at the top of the atomic **`Assembly`**. Typically, it is connected but may be disconnected as well. Although **`Molecule`** encapsulates chemically meaningful information on the bonding among atoms, it does not include information about the 3D-structure. To access the latter, you should refer to a separate object **`Conformation`**. Technically speaking, **`Molecule`** is, primarily, a (auto)collection of the **`Bond`** objects. This collection is strictly associated with its base **`Assembly`**. It changes if the parent assembly is modified. For example, removing an **`Atom`** from an **`Assembly`** automatically deletes all of the related **`Bond`** objects and updates **`Molecule`** (bonds collection).

Conformation

Represents the position of the **`Assembly`** atoms in the space. The reason to separate conformation from assembly is that the assembly can have many conformations. In fact, what we mean here by conformation is only a snapshot of the atomic positions (which may not be stable or chemically reasonable). The assembly may have as many conformations as you wish. The **`Conformation`** is strictly associated with a single **`Assembly`** (base object). Any **`Assembly`** may have multiple attached **`Conformation`** objects, which are (auto) collected into the **`Conformations`** object that is always attached to the **`Assembly`**. The **`Conformation`** derived (child) objects are the **`Structure`** objects. Any **`Conformation`** may have multiple attached **`Structure`** objects which are (auto)collected into the **`Structures`** object, that in its turn is always attached to an **`Assembly`**. The **`Conformation`** changes if the parent **`Assembly`** is modified. If the latter is deleted, all of the conformations are immediately destroyed. Changing **`Conformation`** modifies all of the derived **`Structures`**. For example, removing an **`Atom`** from an **`Assembly`** automatically deletes all of the related **`Point`** objects and updates all of the derived **`Conformation`** objects (**`Point`** collections). Changing the atomic coordinates in **`Conformation`** will influence all of the derived **`Structures`**. Technically speaking, **`Conformation`** is, primarily, (auto)collection of the **`Point`** objects.

Structure

The child of **`Conformation`** and **`Molecule`** based on the common **`Assembly`** of atoms. To access the parent objects, use the reference methods of **`Structure`** (see below).

3.4 How to Create and Destroy Objects

To create an atom, use the ChemBasic function **`NewAtom`**. Since **`Atom`** underlies all other molecular objects, killing the **`Atom`** object is allowed but dangerous. **`Atom.Kill`** will modify all of the related assemblies, molecules, conformations, and structures. The **`Bond`** and **`Point`** objects that use this atom will be destroyed forever. So, to destroy an atom, use the function **`Kill`**. To remove an atom from one particular assembly, use the **`AtRemove`** function. To create a bond or point, use **`NewBond`** or **`NewPoint`**.

To create a new **`Assembly`**, use the method **`AddEmpty`** for the ChemBasic **`Assemblies`** automatic collection. **`Assembly`** may also be created by importing. Note that, once created, the ChemBasic **`Assembly`** holds the references to the related **`Molecule`** and **`Conformation`** `objects`, and lives its own life.

To create a new **`Molecule`** object at a given **`Assembly`**, use the **`AddEmpty`** method for the **`Molecules`** collection attached to this assembly.

To create a new **`Conformation`** object for a given **`Assembly`**, use the **`AddEmpty`** method for the **`Conformations`** collection attached to this assembly. Note that the **`Point`** objects of an empty conformation contain filled **`Atom`** fields and zero coordinates. You may also create a **`Conformation`** object by invoking the 3D-Optimizer or by importing a file with coordinates.

To create a new **`Structure`**, use the **`Derive`** method of the **`Structures`** collection, associated with the given **`Assembly`**.

3.5 Growing a Molecular Tree

Now, we return to our mission. Our first task is to generate the molecules $HO–(CH_2)_n–COOH$ with n from 1 to 10.

Let us, as in the previous example, place all of the molecular-treatment work in a special **`Sub`** called **`GenerateAndAnalyze`**:

```
Function Main As String

      Call GenerateAndAnalyze
      Main="Completed OK."

End Function

Sub GenerateAndAnalyze

      'Generate - optimize - depict – analyze

End Sub
```

Now fill in the Subroutine framework to get the expanded subroutine below. The code below creates the necessary ten oxycarbonic acids:

```
Sub GenerateAndAnalyze
'Generate - optimize - depict – analyze

Dim n,i as Integer
Dim Ent,Mol,Oh,Cox,O1ox,O2ox,AtomCurr,AtomPrev As Object
Const nmax=10

  'In a loop ....

For n=1 To nmax
            'Generate

      'Create a new assembly

      Ent=Assemblies.AddEmpty
      Mol=Ent.Molecules.AddEmpty

      'Add OH (note that we will not add implicit hydrogen)

      Oh=NewAtom(8)     :       Ent.Add(Oh)

      'Add COOH

      Cox=NewAtom(6)    :       Ent.Add(Cox)
      O1ox=NewAtom(8)   :       Ent.Add(O1ox)
      O2ox=NewAtom(8)   :       Ent.Add(O2ox)

      'Make bonds

      Mol.AddBond(Cox,O1ox,2)
      Mol.AddBond(Cox,O2ox,1)
      AtomPrev=Oh

      'Add (CH2)n

For i=1 To n

            AtomCurr=NewAtom(6)
            Mol.AddBond(AtomCurr,AtomPrev,1)
            AtomPrev=AtomCurr

Next i

Mol.AddBond(AtomPrev,Cox,1)
Next n

End Sub
```

To test this loop, create a new file OXYCARB.BAS which contains the simple main function, and the `GenerateAndAnalyze` subroutine as shown. When running it, you will get a clear return code, but don't be surprised when no molecules appear… we have not yet shown how a molecule can be displayed.

What takes place here? We loop through a cycle with an incremented counter **`n`**, which will be the number of methylenes between the ends of a molecule. At each pass of the cycle, we, first of all create a new assembly:

```
Ent=Assemblies.AddEmpty
```

and a new molecule

```
Mol=Ent.Molecules.AddEmpty
```

At the moment, both of them are empty. Now, we can add atoms to the assembly. To do this, we create one atom and add other atoms. (Note that as a parameter **`NewAtom`** receives the element number of the atom to be created):

```
'Add OH  (note that we will not add implicit hydrogen)

Oh=NewAtom(8)      :       Ent.Add(Oh)

'Add COOH

Cox=NewAtom(6)     :       Ent.Add(Cox)
O1ox=NewAtom(8)    :       Ent.Add(O1ox)
O2ox=NewAtom(8)    :       Ent.Add(O2ox)
```

Then, we add bonds to the molecule built at the top of assembly. We could do it as with the atoms, through subsequently issuing **`NewBond`** and **`MoleculeObject.Add`**. However, here a more economical way is shown (note that **`AddBond`** performs both of the functions, and it receives as a parameter the bond order of the bond to be created):

```
'Make bonds

Mol.AddBond(Cox,O1ox,2)
Mol.AddBond(Cox,O2ox,1)
```

`Note` For the explanation of **`NewAtom`**, **`NewBond`**, **`AddBond`**, and related functions, refer to the *ACD/ChemBasic online Help* located in the ACD/Labs software folder (CHEMBAS.CHM).

We just added oxy and carboxy moieties and the bonds at the very last. Adding and connecting methylene groups are more intricate:

```
    AtomPrev=Oh

    'Add (CH2)n

    For i=1 to n

      AtomCurr=NewAtom(6)
      Mol.AddBond(AtomCurr,AtomPrev,1)
      AtomPrev=AtomCurr

    Next i

    Mol.AddBond(AtomPrev,Cox,1)
```

What does this mean? We added carbon after carbon in a cycle (**`For i=1 To n`**) and connected each one to the previous atom (**`AtomPrev`**) of the chain. Look at the code. The atom previous to the cycle is hydroxyl oxygen. So, first methylene will be connected to this oxygen. Then, that methylene becomes **`AtomPrev`**. So, the next, second methylene, which will be added in the second pass of the cycle, will appear connected to the first one, and so on, until our limit (**`n`**) of carbons is reached. Finally, we make a bond between the last methylene in the chain and a carboxy carbon (**`Cox`**).

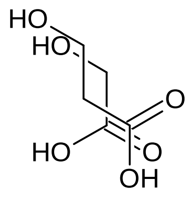
3.5.1 Taking Stock

At this point, you might want the reassurance of seeing your molecule in the ChemSketch window. To do this, two modifications can be done to your OXYCARB.BAS file created above.

1. Set **`nmax`** to *1* instead of *10*; note that this is temporary and for testing purposes only;
2. Insert the type declaration `Dim Page, Diagram as Object` into the subroutine `GenerateAndAnalyze`; and
3. Insert the following three lines of code just before the end of the subroutine:

```
    Page=ActiveDocument.ActivePage
    Diagram=Page.Diagrams.AddEmpty
    Diagram.Depict(Mol)
```

Now, when you run OXYCARB.BAS, you will see one molecule created in the upper left corner of the ChemSketch page. It's not too fancy—in fact, all of the atoms lie on top of one another, but the molecule does exist:



This is a big comfort, and now we can move on to the rest of the task! You can set **`nmax`** back to *10*.

3.5.2 Optimization and Measurement

So, now we are ready for the next tasks—finding the optimized structure and calculating the distance (H)O–C(OOH). The necessary code is quite simple. First, we issue the method **`Do3Doptimize`**, which is applied to the **`Molecule`** object. This method invokes ACD/3D Optimizer, just the same as in ChemSketch. It creates the **`Conformation`** object filled in with optimized atomic coordinates, associates it with the original **`Molecule`** and returns the resulting **`Structure`**:

```
      'Optimize a structure

  Struc=Mol.Do3DOptimize(0.1)
```

Then we apply a so called **`GetDist`** method for the **`Structure`** object to calculate the inter-atomic distance (of course, somewhere at the top of the **`Sub`** we should declare the necessary variable **`r`**):

```
   'Check out the distance
   r=Struc.GetDist(Cox,Oh)
```

`Note` For a more information on the **`Do3Doptimize`**, **`GetDist`**, and related methods, refer to the *ACD/ChemBasic online Help* located in the ACD/Labs software folder (CHEMBAS.CHM).

And finally, we mark the atoms with a label ($) if the distance is less than 5 Å:

```
      If (r<=5.0) Then

            Cox.SetName("$")
            Oh.SetName("$")

      End If
```

The **`SetName`** method sets the name of the object. For an atom, the name is just a label string, which is typically empty, but may contain an arbitrary text. This text will be displayed in the ChemSketch molecular diagram, when we translate our **`Structure`** to **`Diagram`**.

`Important` Be aware that the name of an atom has nothing in common with the atom's element number.

3.5.3 Refining the Output

Now we can do the structure-to-diagram translation and create a proper drawing. This is intended to replace the diagram depiction lines that were added above. The necessary steps are:

1. Create an empty (default) ChemSketch diagram;

```
      'Depict as a ChemSketch diagram

      Diagram=ActiveDocument.ActivePage.Diagrams.AddEmpty
```

2. Depict the structure on this diagram; and

```
Diagram.Depict(Struc)
```

3. Resize the diagram for a more readable format:

```
'Set diagram size to fit all diagrams to a page

Page=ActiveDocument.ActivePage

'Read out page parameters

ph=Page.GetHeight
pw=Page.GetWidth

l=Int(pw/3)
t=200+(n-1)*Int(ph/(nmax+1))
w=Int(pw/6)
h=Int(0.9*ph/(nmax+1))

Diagram.SetBound(l,t,w,h)
```

Some words of explanation; the construction

```
Diagram=ActiveDocument.ActivePage.Diagrams.AddEmpty
Diagram.Depict(Struc)
```

is a ChemBasic idiom for displaying a Structure object in the ChemSketch window.

A related idiom for the inverse task, converting *from* **`Diagram`** to **`Structure`**, requires issuing the **`Assemblies`** method **`AddFromCS`** and extracting the first **`Structure`** attached to the **`Assembly`** that has just been received:

```
Structure=Assemblies.AddFromCS(DiagramObject).Structures.Item(1)
```

> **`Note`** For more detailed information on **`Depict`** and **`GetFromCS`**, refer to the *ACD/ChemBasic online Help* located in the ACD/Labs software folder (CHEMBAS.CHM).

3.5.4 Resizing the Molecular Diagram

To resize the molecular diagram, it is required to get the position and dimension of the rectangle which embeds a diagram, a frame (the default, here). This is accomplished with the **`GetBound`** method, which is applicable to any ChemSketch drawing object, because any of these objects is placed into an invisible frame. The method parameters are the left and top coordinates of the frame, as well as its width and height. After obtaining them, you can calculate a more proper size and position of a frame, and make resizing with **`SetBound`**. The arithmetic, presented in the above code, serves simply to fit our **`nmax`** molecules to the ChemSketch page (whose dimensions are available through methods and **`GetHeight`**).

Note For the more detailed description of **`SetBound`**, **`GetBound`**, **`GetFont`**, **`SetFont`**, and **`SetFontSize`** methods, refer to the *ACD/ChemBasic online Help* located in the ACD/Labs software folder (CHEMBAS.CHM).

Finally, as a matter of convenience, we may add the text, indicating the interatomic distance of interest just near the molecule's picture. For this purpose, to the ChemSketch page we add an empty **`TextBox`** object, then fill it with the necessary information and arrange it properly:

```
'Add a textbox indicating the distance
'carbonyl carbon ---- hydroxy oxygen

TBox=Page.TextBoxes.AddEmpty
TBox.SetContent("Cox---Ooh "+FStr(r,10,4))
TBox.SetBound(l+Int(1.1*w),t+Int(h/2),w,h)
```

Note For the more detailed description of the **`TextBoxes`**, **`SetContent`**, and **`GetContent`** methods and the **`FStr`** function (which simply converts the number to string), refer to the *ACD/ChemBasic online Help* located in the ACD/Labs software folder (CHEMBAS.CHM).

Now, our travel is finished. The full text of the program is given in Appendix (it contains some additional code lines with declarations and checks for the potential error cases). Please run it and see what happens.

Enjoy!

4. Using ACD/ChemBasic in ACD/ChemSketch

4.1 Objectives

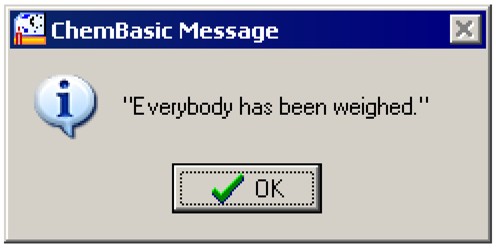
This chapter does not deal with the ACD/ChemBasic language. It only aims to show you how a pre-existing ChemBasic program will fit into the ACD/ChemSketch interface. In this Chapter, you will learn how to:

- Run a ChemBasic program from the ChemSketch window;
- Add a button to the ChemSketch window, to run a ChemBasic program easily;
- Change the active mode (Structure or both Draw and Structure) for the ChemBasic program;
- Suppress a ChemBasic button (deactivation); and
- Delete the button from the ChemSketch window.

The example of the ChemBasic program that we will work with is HELLOMW.BAS, and the corresponding button picture is HELLOMW.BMP. Please take a moment to specify the location of these files; the default location is within the \\EXAMPLES\CHEMBAS\SAMPLES folder.

4.2 Running ChemBasic Programs in ChemSketch

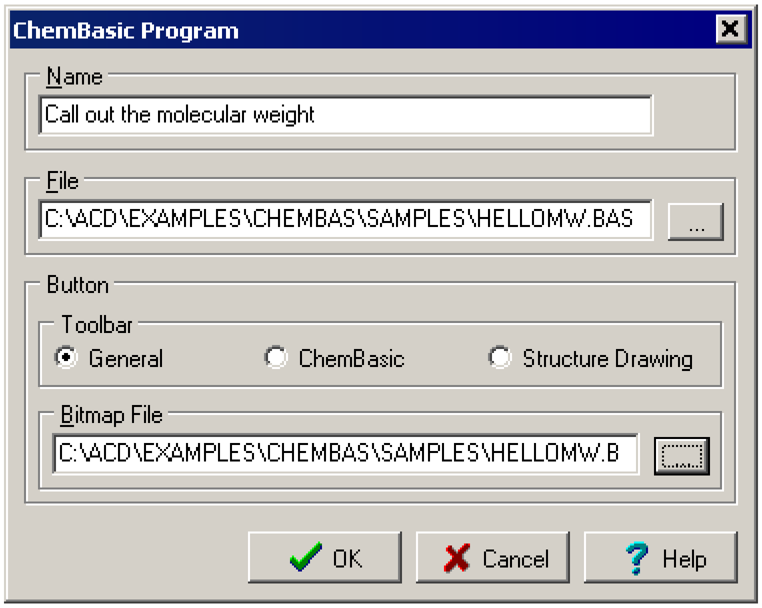
1. Open ACD/ChemSketch. Verify that you are in the Structure mode.
2. Click once to place methane, CH_4, within the ChemSketch drawing area. (It could be any other molecule.)
3. From the **File** menu, choose **Run ChemBasic**.
4. In the **Run ChemBasic** dialog box that appears, specify the name and location of HELLOMW.BAS. Click **OK**.
5. You will see a series of message boxes, including one which gives the molecular weight of the drawn compound.
6. The last message, “Everybody has been weighed” signals the end of the ChemBasic program:



"Everybody has been weighed."

4.3 ChemBasic Programs As Buttons

1. From the **Options** menu, choose **ChemBasic Organizer**.
2. Click **New** to display the **ChemBasic Program** dialog box.
3. In the **ChemBasic Program** dialog box, in the **Name** box, enter a name for the program you will be linking to the ChemSketch interface. In the **File** and **Bitmap File** boxes you are prompted for the locations of the ChemBasic program file and the small picture file. Click the **Browse** buttons and specify the location of the HelloMW files in place:



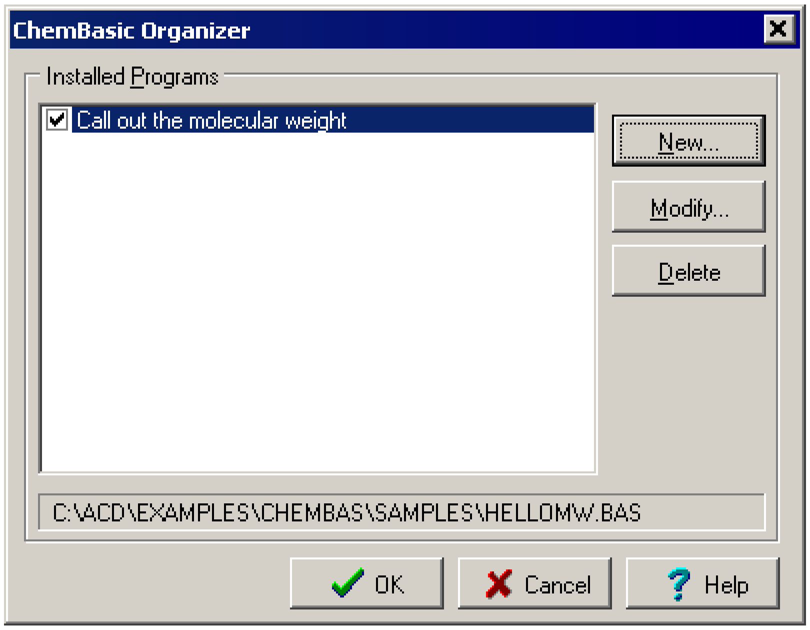
Name: Call out the molecular weight

File: C:\ACD\EXAMPLES\CHEMBAS\SAMPLES\HELLOMW.BAS

Toolbar: [✓ General] ChemBasic Structure Drawing

Bitmap File: C:\ACD\EXAMPLES\CHEMBAS\SAMPLES\HELLOMW.B

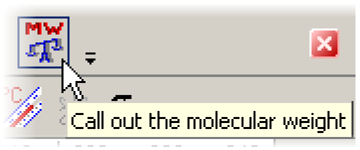
4. Make sure that the **General** option is selected in the **Toolbar** area and click **OK**. The **ChemBasic Organizer** dialog box appears:



☑ Call out the molecular weight

C:\ACD\EXAMPLES\CHEMBAS\SAMPLES\HELLOMW.BAS

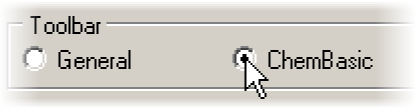
5. Click **OK** and you will notice a new button (MW) appears on the General toolbar of the ChemSketch window.
6. Point to the button. Note that the yellow hint that appears contains the name you typed into the **ChemBasic Program** dialog box:



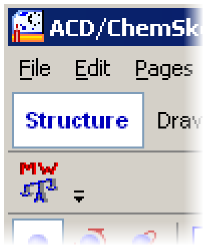
7. Click once on the new button to verify that the ChemBasic program works the same way it did in the previous section.

4.4 Changing Button Placement and Active Mode

1. On the General toolbar, click **Draw** (Draw) to switch to the Draw mode. The button is still there. That is because you the General toolbar you have selected is common for both modes.
2. From the **Options** menu, choose **ChemBasic Organizer**.
3. In the **Installed Programs** area, click the program to select it, and then click **Modify** (Modify...).
4. In the **Toolbar** area of the **ChemBasic Program** dialog box, choose **ChemBasic**, and click **OK**.


Toolbar
General [✓ ChemBasic]

5. Click **OK** again to leave the **ChemBasic Organizer** dialog box. The position of the button has changed in the ChemSketch window. Now it is located on its own toolbar:



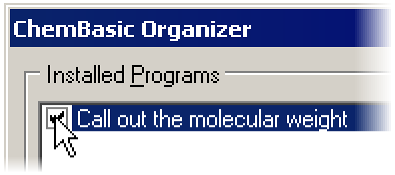
6. Verify that the button works by clicking it, and ensuring that the molecular weights are stated, even though the Draw mode is active.
7. From the **Options** menu, choose **ChemBasic Organizer**.
8. In the **Installed Programs** area, click the program to select it, and then click **Modify** (Modify...).
9. In the **Toolbar** area of the **ChemBasic Program** dialog box, choose **Structure Drawing** and click **OK** twice to close the dialog boxes.

10. In the Draw mode of the ChemSketch window, the button has apparently disappeared. On the General toolbar, click **Structure** to return to Structure mode. You will see that the button is still there on the Structure toolbar.
11. Once more, verify that the program works correctly by clicking on the button. You will see that the molecular weights of all the molecules currently displayed in the structure drawing area are stated.

4.5 Deactivating a ChemBasic Button

To deactivate a button is to remove it from the ChemSketch window, without deleting the information on the program location from the interface. It just suppresses the display of the ChemBasic program. For example, you may have some ChemBasic programs that simplify documents preparation, but that buttons you do not want to be displayed when you are called upon to give a formal presentation.

1. From the **Options** menu, choose **ChemBasic Organizer**.
2. In the **Installed Programs** area, click the check box beside the program to deselect it:



3. Click **OK** to leave the **ChemBasic Organizer** dialog box.
4. In the ChemSketch screen, the button should no longer be displayed, regardless of whether you are in the Structure or Draw mode. However, it is still listed in the **ChemBasic Organizer** dialog box.

4.6 Deleting a ChemBasic Button

1. To delete a button from the ChemSketch interface, from the **Options** menu, choose **ChemBasic Organizer**.
2. In the **Installed Programs** area, click the program to select it, and then click **Delete** .
3. In the ACD/ChemSketch question message that appears, click **Yes**, and then click **OK** to leave the **ChemBasic Organizer** dialog box.
4. In the ChemSketch window, the button is no longer displayed and the information on the program location is removed from the interface.

5. ChemBasic Goodies for ChemSketch

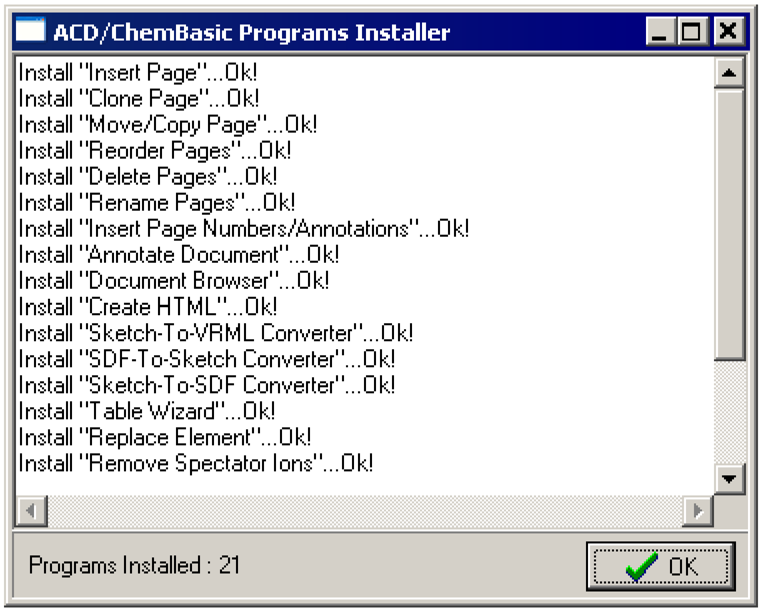
5.1 What are "Goodies"?

The "Goodies" are additional buttons that extend the functionality of ACD/ChemSketch. They are actually a set of 22 ACD/ChemBasic programs. These helpful buttons are already included in most ChemSketch installations. To check if you already have them, look for all .BAS files within the \\EXAMPLES\CHEMBAS\GOODIES folder. They are easy to install and to use. You will improve the versatility of your copy of ChemSketch and learn more about ACD/ChemBasic at the same time.

Note You can remove some or all of these new buttons from your ChemSketch toolbar at any time.

5.2 Automatic Installation of the Goodies

At the time of installing ACD/ChemSketch, the user is prompted to install the Goodies. If **Yes** is chosen, the Goodies are installed:



```
Install "Insert Page"...Ok!
Install "Clone Page"...Ok!
Install "Move/Copy Page"...Ok!
Install "Reorder Pages"...Ok!
Install "Delete Pages"...Ok!
Install "Rename Pages"...Ok!
Install "Insert Page Numbers/Annotations"...Ok!
Install "Annotate Document"...Ok!
Install "Document Browser"...Ok!
Install "Create HTML"...Ok!
Install "Sketch-To-VRML Converter"...Ok!
Install "SDF-To-Sketch Converter"...Ok!
Install "Sketch-To-SDF Converter"...Ok!
Install "Table Wizard"...Ok!
Install "Replace Element"...Ok!
Install "Remove Spectator Ions"...Ok!
```

Programs Installed : 21

This has caused the CBINSTAL.EXE program to run, after the other ACD/Labs software has been installed.

Alternatively, you can install Goodies later by running CBINSTAL.EXE as described below.

Information on the installed goodies is written to the Windows registry so it is personalized to the user. The presence of the Goodies buttons is reflected in any version of ACD/ChemSketch that you use on your local drive.

5.2.1 CBINSTAL.EXE and CBINSTAL.INF

For automatic installation of a batch of ChemBasic programs (not only goodies) into the ChemSketch interface, you can use the CBINSTAL.EXE file located in the ACD/Labs installation folder This installer takes information about the programs and associated pictures to be installed from the CBINSTAL.INF file (located in the \\EXAMPLES\CHEMBAS\GOODIES) that contains the following information:

```
Insert Page*inspage.bas*inspage.bmp*main*1
Clone Page*clnpage.bas*clnpage.bmp*main*1
...
...
Peptide Builder*PEPBUILD\pepbuild.bas*PEPBUILD\pepbuild.bmp*main*1
Carbohydrate Builder*SUGARSK\sugarsk.bas*SUGARSK\sugarsk.bmp*main*1
Nucleic Acid Builder*nuclbld.bas*nuclbld.bmp*main*1
```

Where the format is:

*Name on yellow tag***FileName***.bas****picture***.bmp*****main****toolbar number*

The **.bas** and **.bmp** files can be provided with the path names if necessary.

The *toolbar number* corresponds to the following toolbars:

1—ChemBasic toolbar that is located right below the general toolbar;

2—Structure toolbar (in this case the button will not be visible in the Draw mode); and

3—General toolbar.

Note Modifying the CBINSTAL.INF file can be useful if you want to add new ChemBasic programs to be automatically installed into ACD/ChemSketch.

5.3 The Goodies

In this section we present a list of the available Goodies, organized by their function.

5.3.1 Document Organization Goodies

Goody	Button	Description
Insert Page		Inserts a blank page at any place within your ChemSketch document.
Clone Page		Clones a page (including its contents) for the given number of times—very useful for filling in the document with page templates, tables, titles, and so on.
Move/ Copy Page		Reorders pages in your document by moving or copying of a current page.
Reorder Pages		Allows to cut-and-paste or copy-and-paste a sequence of pages to a new position within the same document.
Delete Pages		Deletes a range of pages.
Rename Pages		Changes name of the page with the given number. *<u>Notes:</u>* • Useful in conjunction with Annotate Document (above). • This will not put text on your ChemSketch page.
Insert Page Numbers/ Annotations		Inserts page numbers or complex annotations in your document. Note that the annotation will be inserted at the bottom-left corner of the page. *<u>Note:</u>* • This will place markings in your ChemSketch document itself.
Annotate Document		Annotates your documents based on the content of the leftmost top textbox on each page. *<u>Notes:</u>* • Changes show up in the status bar page indicator. • Great for managing large documents and presentations. • This will not put text on your ChemSketch page.
Document Browser		Automatically looks through the directories to find ChemSketch documents containing the query text string.

5.3.2 File Formats & Table Formatting

Goody	Button	Description
Create HTML	HTML	Exports all of the selected pages of a current document into an HTML file, which you may then view with your favorite web browser.
Sketch-to-VRML Converter	VRML	Exports all of the molecules in a current page into a VRML 2.0 file, which you may then view with Cosmo, GLView, or any other VRML browser. *Notes:* • Places the resulting WRL file in the same directory with SK2VRML.BAS. • By client request.
SDF-to-Sketch Converter	SDF	Imports the data (molecules, texts, *etc*.) from a file in MDL's SDfile format into a ChemSketch document. From 1–20 structures per page. *Notes:* • LIMITATION: no more than 100 pages available in a ChemSketch document. • A common method for looking at a client's problem file. • The proportion of font and structure size may be strange, so use "Properties: Normal Style" and/or 2D Clean to adjust.
Sketch-To-SDF Converter	SDF	Exports all of the structures from the current page or from the whole document into an SDfile. *Notes:* • A logical extension of the **Export to MOL** command. • Only properties exported are FW & Formula.
Table Wizard	▦	Creates tables or/and aligns objects according to a specified number of rows and columns. *Note:* • Helpful to use in conjunction with SDfile-to-Sketch converter.
Remove Spectator Ions (Desalt)	SDF SDF	An SDfile that contains one or more salt structure entries can be changed to a "one-molecule-per-entry" SDfile. This button removes the smallest ion, either by MW or by number of atoms. For example, sodium acetate will have the sodium atom removed, and acetic acid will remain behind. (The molecule left behind is put into neutral form.) *Notes:* • A special sample file, SALTS.SDF with 5 salts in it is placed in the Goodies directory for testing. • Use the Import SDfile Goody to double-check NEWFILE.SDF.
Label Printer	⚗	Quickly creates labels for chemicals and prints them on the Avery Standard (45 template sheets included) or your own sheets. *Notes:* • If you use a file template such as AS_8663.SK2 (10 labels on a page) this is also an easy way to get a table.

5.3.3 Chemistry Goodies

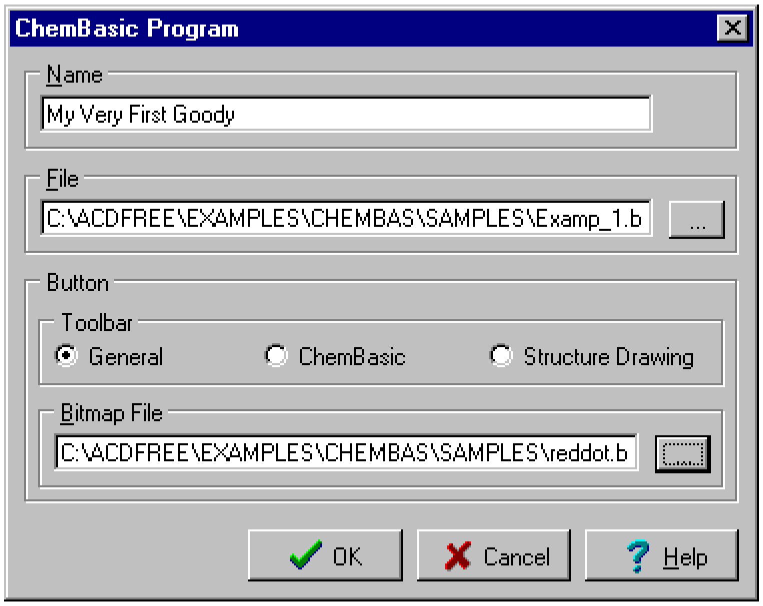
Goody	Button	Description
InChI	InChI	Generates unique identifier and pastes it into the ChemSketch window or transfers the structure into the original InChI program created and distributed by International Union of Pure and Applied Chemistry (IUPAC). For more information, refer to the *Integrating InChI into ACD/ChemSketch* document (http://www.acdlabs.com/download/technotes/80/draw_db/inchi.pdf). *Note:* • Available only as a free download from our Web site.
Column Selector	ColSel	Searches a knowledge base of the most utilized columns in order to locate the ones that have the properties best suited to the separation at hand. *Note:* • Compares the selected column with the list or two selected columns between themselves. For information on how to work with this tool, refer to the COLSEL_W.PDF file that you can find in the (\\DOCS) folder.
Replace Element	H→F	Replaces all of the atoms of a given type with atoms of another type in a chemical structure. *Notes:* • Useful for drawing perfluorinated structures. • Can have only a single structure present. • Partial perfluorination—not yet available.
Solution Calculator	1M	Calculates the weight of a compound required for preparing a solution of the user-defined volume and molar concentration. *Notes:* • Note that program execution is possible only with a single structure on the page. • Print out of solution output—not yet available.
Peptide Builder	Ala	Builds a 3D peptide structure from the amino acid sequence. *Note:* • Both types of input, Gly-Ala-Ser-Pro and GASP, accepted.
Carbo-hydrate Builder		Builds a structure from carbohydrate abbreviated names.
Nucleic Acid Builder		Builds a 3D nucleic acid (DNA, RNA) structure (one or two chains) from your input sequence.

5.4 Make Your Own "Goody"

5.4.1 Super Simple

The simplest way to create your own Goody is to use one of the example ChemBasic programs already made up.

1. In the \\EXAMPLES\CHEMBAS\SAMPLES folder, find an example .BAS program.
2. Create your own .BMP using MS Paint (or related software). The size of the button should be about 20 by 20 pixels (about 0.2 inches or 0.63 cm square).
3. In the ChemSketch window, from the **Options** menu, choose **ChemBasic Organizer**.
4. In the **ChemBasic Organizer** dialog box, click **New** New... .
5. In the **ChemBasic Program** dialog box that appears, type the name of the Goody in the first line, then browse to find the locations of the .BAS and .BMP files.



Name: My Very First Goody

File: C:\ACDFREE\EXAMPLES\CHEMBAS\SAMPLES\Examp_1.b

Button

Toolbar: [✓ General] ChemBasic Structure Drawing

Bitmap File: C:\ACDFREE\EXAMPLES\CHEMBAS\SAMPLES\reddot.b

6. Click **OK**. In the ChemSketch window, you will see that the button has appeared. Try it out!

5.4.2 Fairly Simple

The next simplest way to create your own Goodie is to *construct your own* ChemBasic program "the easy way." Recommended procedure is:

1. Find a Goodie or example ChemBasic program that is similar to what you want to do. Copy and rename the program.
2. Open the copied program with CBEDIT.EXE. Use the *ACD/ChemBasic online Help* (CHEMBAS.CHM) that contains many examples extensively. Functions are listed but are highly constrained by the modules that are available. You will see GetIUPACName but not GetLogP.
3. Run the program from *within* CBEdit by clicking **Run** ⚡.

   Note The errors are flagged line by line—VERY helpful.
4. You might want to refer to another program at the same time, and fortunately, unlike Wordpad, multiple files *can* be opened in the CBEdit editor—just click on the tab of the program that you want to see.
5. Once the .BAS program runs smoothly, follow steps 2–7 of the preceding section to make the button available in the ChemSketch window.

Appendix A. Full Text of Sample Programs

HELLOMW.BAS

```
Function Main As String
  Dim Answer As Integer
  Answer=MessageBox("My, you are looking well!","Example #1", MBB_OK
+MBI_EXCLAMATION)
  Answer=MessageBox("Have you been on a diet?","Example #1", MBB_YESNO +
MBI_QUESTION)
  If Answer= MBR_YES Then
    Call GreetThemAll
    Main="Everybody has been weighed."
  Else
    Main="Sorry, time to leave."
  End If
End Function

Sub GreetThemAll
Dim Diagram As Object, Answer As Integer, MolWeight As Double, Tail As
String
Tail=Chr(13)+Chr(13)+Chr(13)+"Greet the next?"
With ActiveDocument.ActivePage.Diagrams
  For Each Diagram in .Self
    MolWeight=Diagram.GetMolWeight
    If MolWeight=0.0 Then MolWeight=-99.99
    Answer=MessageBox("Hmmm, you weigh"+LTrim(FStr(MolWeight,12,4))+"
a.u.!!"+Tail,"Example #1a", MBB_YESNO+MBI_EXCLAMATION)
    If Answer= MBR_NO  Then Exit For
  Next Diagram
End With
End Sub
```

HELLO.BAS

```
'This program
'gets all the molecules in current window of ChemSketch;
'generates their IUPAC names
'and says a personal Hello to each of them

Function Main As String
  Dim Answer As Integer
  Answer=MessageBox("Hello, Molecules!","Example #1", MBB_OK
+MBI_EXCLAMATION)
  Answer=MessageBox("Say Hello to all?","Example #1", MBB_YESNO +
MBI_QUESTION)
  If Answer= MBR_YES Then
    Call GreetThemAll
    Main="Everybody welcomed."
  Else
    Main="Sorry, we leave them."
  End If
End Function

Sub GreetThemAll
Dim Diagram as Object, Answer as Integer, MolName, Tail as String
Tail=Chr(13)+Chr(13)+Chr(13)+"Greet the next?"
With ActiveDocument.ActivePage.Diagrams
  For Each Diagram In .Self
    MolName=Diagram.GetIUPACName
    If Trim(MolName)="" Then MolName=" though I can not pronounce your
name"
    Answer=MessageBox("Hello,"+MolName+"!"+Tail,"Example #1",
MBB_OKCANCEL+MBI_EXCLAMATION)
    if Answer= MBR_CANCEL  Then Exit For
  Next Diagram
End With
End Sub
```

OXCARBOX.BAS

```
' This program
' (i) fills in current window of ChemSketch  with the molecules
'     of general formula HO-(CH2)n-COOH, n=1-10,
' (ii) performs 3D-optimization for all them and
' (iii) marks those which contain a hydroxyl group not further than
'       3 Angstroms from a carbonyl oxygen

Const nmax=10

Function Main As String
Dim Entities (1) As Object
 Call GenerateAndAnalyze
Main="Completed OK."
End Function

Sub GenerateAndAnalyze
'Generate - optimize - depict - analyze
Dim n,i,ph,pw,l,t,w,h as Integer, r As Double
Dim Ent,Mol,Oh, Cox,O1ox,O2ox,AtomCurr,AtomPrev, Page,Diagram, Struc,Font,
TBox As Object

  'Get a page
  Page=ActiveDocument.ActivePage
  'If page is not empty, create new page
  If Page.Drawings.Count>0 Then Page=ActiveDocument.AddEmpty
  'Read out page parameters
  ph=Page.GetHeight
  pw=Page.GetWidth

  'In a loop ....

  For n=1 to nmax

    'Generate

    'Create new assembly
    Ent=Assemblies.AddEmpty
    Mol=Ent.Molecules.AddEmpty
    'Add OH (note that we will not add implicit hydrogen)
    Oh=NewAtom(8)     :       Ent.Add(Oh)
    'Add COOH
    Cox=NewAtom(6)    :       Ent.Add(Cox)
    O1ox=NewAtom(8)   :       Ent.Add(O1ox)
    O2ox=NewAtom(8)   :       Ent.Add(O2ox)
    'Make bonds
    Mol.AddBond(Cox,O1ox,2)
    Mol.AddBond(Cox,O2ox,1)
    AtomPrev=Oh
    'Add (CH2)n
    For i=1 To n
      AtomCurr=NewAtom(6)
      Mol.AddBond(AtomCurr,AtomPrev,1)
      AtomPrev=AtomCurr
    Next i
      Mol.AddBond(AtomPrev,Cox,1)

    'Optimize a structure

    Struc=Mol.Do3DOptimize(0.1)

    'Check out the distance

    r=Struc.GetDist(Cox,Oh)
    If (r<=5.0) Then
      Cox.SetName("$")
      Oh.SetName("$")
    End If

    'Depict as a ChemSketch diagram

    Diagram=Page.Diagrams.AddEmpty
    Diagram.Depict(Struc)
    'Set diagram size to fit all diagrams to a page
    l=Int(pw/3)
    t=200+(n-1)*Int(ph/(nmax+1))
    w=Int(pw/6)
    h=Int(0.9*ph/(nmax+1))
    Diagram.SetBound(l,t,w,h)
    'Set font for atoms symbols
    Font=Diagram.GetFont
    Font.SetFontSize(24)
    Diagram.SetFont(Font)

    'Add a textbox indicating the distance
            ‘carbonyl carbon ---- hydroxy oxygen
    TBox=Page.TextBoxes.AddEmpty
    TBox.SetContent("Cox---Ooh "+FStr(r,10,4))
    TBox.SetBound(l+Int(1.1*w),t+Int(h/2),w,h)


  Next n


End Sub
```