VB Scripting for CATIA V5

How to Program CATIA Macros

Emmett Ross

Fourth Edition

Copyright Information

Disclaimer

Although the author has attempted to exhaustively research all sources to ensure the accuracy and completeness of information on the subject matter, the author assumes no responsibility for errors, inaccuracies, omissions, or any other inconsistencies herein. The data contained herein is for informational purposes only and is not represented to be error free. Information may be rendered inaccurate by changes made to the subject of the material, such as the applicable software. No consequential damages can be sought against the author for the use of these materials by any third parties or for any direct or indirect result of that use. The purpose of this text is to complement and supplement other texts and resources. You are urged to read all the available literature, learn as much as you can and adapt the information to your particular needs. There may be mistakes within this manual. Therefore, the text should only be used as a general and introductory guide and not as the sole source for CATIA macro programming. The information contained herein is intended to be of general interest to you and is provided “as is”, and it does not address the circumstances of any particular individual or entity. Nothing herein constitutes professional advice, nor does it constitute a comprehensive or complete statement of the issues discussed thereto. Readers should also be aware that internet websites listed in this work may have changed or disappeared between when this work was written and when it is read.

The author also assumes a general understanding of how to use CATIA V5 including geometry creation and various workbenches (mainly Part Design, Generative Shape Design, and Assembly Design). To learn more about CATIA please refer to the resources in the Appendix for more information about how to use CATIA or where to go to get further answers or advice. I welcome any comments you may have regarding this book. To contact me please email: emmett@scripting4v5.com

The Author's Story

Learning how to program CATIA macros helped save my career.

You've probably found your way to this book from my website, or maybe a friend passed it along to you. Either way, I'm happy you're here. I wrote this guide because when I was in a time of desperate need, not too long ago, learning how to write CATIA macros helped save my professional career. I was working as a CAD engineer but was just beginning to learn how to use CATIA. Everyone else at my company was much more experienced than I was, therefore I was expendable. When the economy took a turn for the worse, and coworkers began getting laid off, I feared for my job and my family's future.

I needed a way to set myself apart to prove my value to the team. Learning how to write macros in CATIA gave me a huge advantage over my coworkers and helped to quickly earn my colleague's respect, leading not only to me keeping the job but also to quicker promotions, along with more job freedom and flexibility. Not only did it help me bounce back from a low point but it opened my eyes to the world of automation and the opportunities that it can create for an engineer's career. *Scripting4v5.com* and *VB Scripting for CATIA V5* are my way of giving back for all of the fortunate things that have happened to me ever since.

I want this content to provide anyone "walking in off the street" the knowledge to be able to write their first CATIA macro with as little pain as possible. This book is a guide, and the purpose of this guide is to do just that - ***guide you***. It will take you through the process of turning your ideas into automated programs. If I can help just one person learn one thing that will help them in their career and/or life, the countless hours I have spent writing this book will have been totally worth it. If at any point while you're reading this guide and you have any questions, please don't hesitate to contact me. Even if you don't have any questions, I'd love for you to come by and say hello! If you want to reach me in private you can email me at emmett@scripting4v5.com.

-Emmett Ross

Chapter 1: Introduction to CATIA Programming

Are you tired of repeating those same time-consuming CATIA processes over and over? Worn out by thousands of mouse clicks? Don't you wish there were a better way to do things? What if you could rid yourself those hundreds of headaches by teaching yourself how to program macros while impressing your bosses and coworkers in the process? **VB Scripting for CATIA V5** is the ultimate guide to teach you how to write macros for CATIA V5!

Through a series of example codes and tutorials you'll learn how to unleash the full power and potential of CATIA V5. No programming experience is required! There are many CAD engineers, designers, and technicians who want to write macros but simply don't have the time or money to go to an expensive third party training class. This text will cover the core items to help teach beginners important concepts needed to create custom CATIA macros. More importantly, you'll learn how to solve problems and what to do when you get stuck. Once you begin to see the patterns you'll be flying along on your own in no time.

Book Format

Everyone learns in different ways. Therefore, **VB Scripting for CATIA V5** is comprised of different types of exercises in order to address all the different learning styles.

- **Tutorials:** Step-by-step instructions will show you exactly how to create the example macros.
- **Exercises:** Basic steps and an end goal of a problem are given and the programmer must figure out the best way to solve it. Solutions in the back of the book.
- **Quizzes:** The end of each chapter features a quiz between 5 and 15 questions comprising multiple choice, fill in the blank, or true or false questions to test your knowledge The solutions are located in the back of the book.
- **Real-world examples:** Where possible I will show you how the code can be applied in the real-world of CAD engineering.

What is a Macro and why do we use them?

A macro is a series of functions written in a programming language that is grouped in a single command to perform the requested task automatically. If you perform a task repeatedly you can take advantage of a macro to automate the task. Why do manual labor

when you can simply press a button instead? Macros are used to save time and reduce the possibility of human error by automating repetitive processes. Other advantages include standardization, improving efficiency, expanding CATIA's capabilities, and streamlining procedures. Macros use programming but you don't need to be a programmer or have programming knowledge to use them (though it certainly does help).

The application of automation in the design process is virtually unlimited. Some real world examples of CATIA automation at work:

- Batch script for the conversion of CATDrawing files to PDF
- Batch script to convert CATParts to STP files
- Import of points from an Excel spreadsheet to a 3D CAD model
- Export of data from CATIA model to a bill of material spreadsheet
- Automatic geometry creation from selection
- Automatic drawing creation
- Custom functions

And so on and so on. The possibilities are nearly limitless.

Terms and Definitions

The following is a list of terms and their definitions which will be used frequently throughout this text. It is recommended that you become familiar with them if you aren't already. A quick reference of acronyms is listed in the appendices of this book as well.

Integrated Development Environment (IDE) is a computer application to help programmers develop software and typically consists of a source code editor, debugger, build automation tools, object browser, and a compiler or interpreter. IDEs typically have built-in syntax checkers, color coded schemes, and automatic code completion. The Visual Basic Editor in CATIA and Excel is an example of an IDE.

Graphical User Interface (GUI) is a way for humans to interact with computers with graphical elements such as windows, menus, toolbars, icons, etc. which can be manipulated by a mouse. The VBA editor is a perfect example.

Command Line Interface (CLI) is a way for humans to interact with computers through text only and is accessed solely by a keyboard. The most common example is MS-DOS.

Component Application Architecture (CAA, CAA V5, or CNext) is the **Application Programming Interface (API)** or technological infrastructure designed to support Dassault Systèmes products. It is an open development platform enabling programmers to develop and integrate their own applications for CATIA or other Dassault Systèmes products. CAA V5 is faster and more powerful than VB. It provides access to interfaces not available to Visual Basic but is harder to learn. C++ is the primary language. A single source code is used for both Windows and UNIX. CAA **Rapid Application Development**

Environment (RADE) provides a workbench to develop PLM applications using the component object model object oriented programming. CAA is beyond the current scope of this text.

Object Oriented Programming (OOP) is where programmers define not only the data type of a data structure, but also the types of operations, or functions, that can be applied to the data structure. An object in software is a structure that consists of data fields and subroutines or functions. Everything in CATIA is an object; the data fields are called *Properties* and the subroutines and functions are called *Methods*. All the data and functions have owners which are the objects to which they belong. A thorough understanding of OOP is critical to your success in macro programming. More on this later.

Component Object Model (COM) is a Microsoft technology that enables sharing of binary code across different applications and languages. CATIA V5 is COM enabled software. Codes for COM objects or components can be called, initiated, or created at any time because they are stored in DLL files and registered in the Windows registry. If CATIA calls Excel, CATIA is then the client and Excel is the server, or the one that provides *services* to the client.

VB talks to CATIA through **Dynamic Linked Libraries (DLL).** DLLs are compiled files that contain all of the functions that make CATIA V5 perform an action. For example, when you select the "point" function in CATIA, the program calls a function inside one of the dll files that performs the action of creating a point in the V5 database. These files are both compiled and encrypted (or "mangled") and are located in the UNLOAD directory for CATIA V5 (*C:\Program Files\Dassault Systemes\B20\intel_1\code\bin*). Encryption is a method by which software companies can ensure that others cannot access the function inside the dlls. You cannot directly call the dlls from outside applications, therefore extra programming needs to be done to allow the dlls to be exposed to Windows and the COM object model. This is done via Type Library Files.

Type Library Files (TLB) are files necessary for exposing functions to Windows by acting as maps which point to the functions inside of the dll files that make CATIA V5 work. The TLB files are also located in the UNLOAD directory for V5. Any external application needs to have access to these files. The complete process is: VB Application -> Type Libraries -> Dynamic Linked Libraries -> CNext. How to create references to type libraries is shown in later chapters of this text.

Universal Unique Identifier (UUID) - Every CATPart and CATProduct contains a UUID. Basically, CATIA identifies files based on their file name and their UUID. Where problems occur are when two pieces of data have the same UUID. Compounding the problem, the UUID can't be viewed or edited with any current CATIA function. There are cases when two files may have different names but share the same UUID. This causes a problem when dealing with **Product Data Management (PDM)** systems, like SmartTeam. It is recommended to create new UUIDs whenever possible. Actions which will **create new UUID** include:

- File + New
- File + New From
- File + SaveAs - option save as new document
- INSERT New Product
- INSERT New Part
- Document Template Creation

Actions which will **keep the same UUID** for each include:

- File + Open
- File + Save Management
- File + Save
- File + SaveAs
- Send to directory
- File + CLOSE
- File + Save
- File + Save ALL

CATIA Macro Languages

Just like most countries on this planet have their own native languages, software programs have their own programming language. Many of these are very similar to each other so learning elements that are common between all programming languages will help you transition from one to another if you need to! For example, after learning to program in CATIA and Excel, I was able (with some help from some tutorials on YouTube) figure out how to program some basic Android applications in Java. That's powerful stuff!

CATIA V5 automation was originally designed for VB6, VBA, and VBScript. Microsoft no longer officially supports VB6 as it has been replaced by VB.net, which is supported by CATIA V5 R16 and onwards. VB6 is more complex but also more powerful than VBA, as is VBA over VBScript and CATScript. Macro languages supported by CATIA and discussed in this text are VBScript, CATScript, and VBA, all derivatives of Visual Basic used in scripting.

CATScript is Dassault Systèmes' portable version of VBScript and is very similar to it. CATScript macros CAN run on UNIX systems. It is a sequential programming language and non-GUI oriented. Regular text editors (like Notepad) can be used for coding. Advantages of writing CATScript macros include free to use, macro recording, personal time saving operations, and rapid deployment. The disadvantages of CATScript are limited flexibility and difficult to debug. The file extension is **.CATScript**.

VBScript is a subset of the Visual Basic Programming language (VBA). All elements of VBScript are present in VBA, but some VBA elements are not implemented in VBScript. The result of the slimming down process is a very small language that is easy to use. VBScript (officially, "Microsoft Visual Basic Scripting Edition") was originally designed to run in Web applications such as Internet Explorer. One of the advantages of VBScript (in

common with other scripting languages) is that it's written in plain, ordinary ASCII text. That means that your 'development environment' can be something as simple as Notepad. CATIA objects can be called but no type is used as the system tries to dynamically call methods and properties of objects. It can be used on both Windows and UNIX versions of CATIA. The disadvantage of VBScript is it's slow, is limited for interface development, and has the least functionality. The file extension is **.catvbs**.

VBScript (MS VBScript) and CATScript are very similar with the major difference being variable declaration. Many programmers believe it is better to declare all variables As String, As Integer, etc. to better keep track of each variable type.

Visual Basic (VB or VB6) is the full and complete version. Derived from BASIC, VB6 programs can generate independent programs, can create ActiveX and servers, and can be compiled. VB programs run in their own memory space.

VBA (Visual Basic for Applications) is another subset of Visual Basic and is hosted in applications such as CATIA (after V5R8), Microsoft Word, Excel, etc. VBA provides a complete programming environment with an editor, debugger, and help object viewer. Declaring the object library used is allowed. In CATIA, VBA has the full VB6 syntax and IDE, which is similar to VBA in Excel. It is event driven, GUI oriented, and has full IDE yet cannot run a program WITHOUT the host application running (meaning it runs as a DLL in the same memory space as the host application). The advantages of using CATvba macros include using the GUI, building forms, and the debugging ability of the macro editor. The disadvantage is VBA programs cannot be compiled into executables or DLLs and they do not run in their own memory space. The extension is **.catvba**.

Visual Basic.NET (VB.net) is Microsoft's designated successor to VB6 and has been supported by Dassault Systèmes since V5R16. VB.net is event driven, has IDE, and is used for building GUI but is not COM (though it can call COM objects). The syntax is different from VB6. Code can be compiled into .exe or .asp files. There are many issues encountered when switching from VB6 to VB.net, such as new syntax, new IDE, new GUI controls, and a new Install Shield, therefore fully automatic conversions are near impossible. Compiled languages like VB.net aren't completely necessary because most automation can be done in VBA. VB.net will not be discussed in this text.

There are two primary methods a macro communicates with CATIA: in-process or out of process.

In Process: The first method a macro communicates with CATIA is when the VB application is ran from within the CATIA process in the computer memory. CATIA essentially freezes while the macro is running and the allocation memory is wiped clean after each run of the program so passing data between subsequent runs is impossible. To access and create in-process macros go to Tools > Macros but please note the only options are VBScript, CATScript, or VBA. Most of the examples in this text are In Process macros.

Out of Process: The other communication method is called out of process where the program runs in its own process in the computer memory. The application could be run

from Excel, Word, Windows Explorer, etc. CATIA is fully active while the program is running. VB.net or VB6 can also be used.

How to Create Macros

Macros within CATIA are created by two primary methods:

1. Using the macro recorder or
2. Writing custom code with the macro editor

Once a macro is created, there are multiple ways to open the macros window to run your macros:

1. Go to Tools>Macro>Macros
2. CATIA macros window keyboard shortcut: **Alt+F8**
3. Assign or create your own icon for each macro
4. Visual Basic Editor (VBE) shortcut: **Alt+F11**

If the macro editor cannot be opened, talk to your system administrator because it may not have been installed. No extra license is required to run macros, though sometimes licenses for special workbenches are needed if the code uses a function or method from a particular workbench.

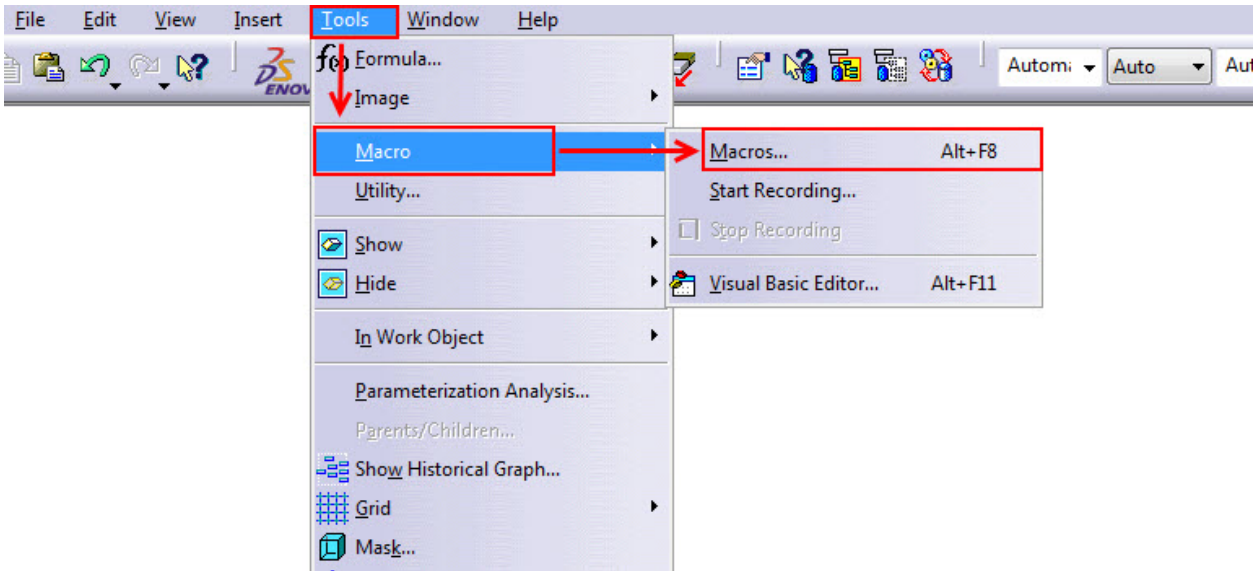
Macro Libraries

CATIA macros are stored in macro libraries in one of three locations: Folders (vbscript and CATScript), Project files (catvba), or documents such as CATParts, CATProducts, and CATDrawings. Only one of these macro libraries can be used at a time. When creating a new macro library, the folder or path location must already exist. If not you will get an error message.

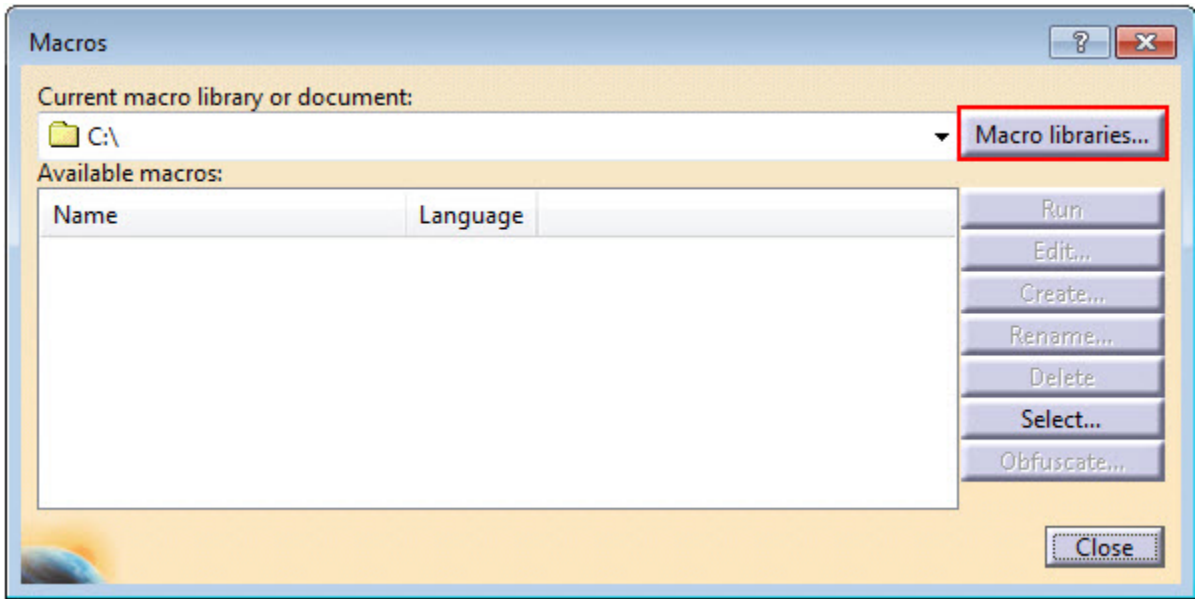
Tutorial 1.1: Create a new macro library

Use the following steps to create a new macro library or setup an existing one:

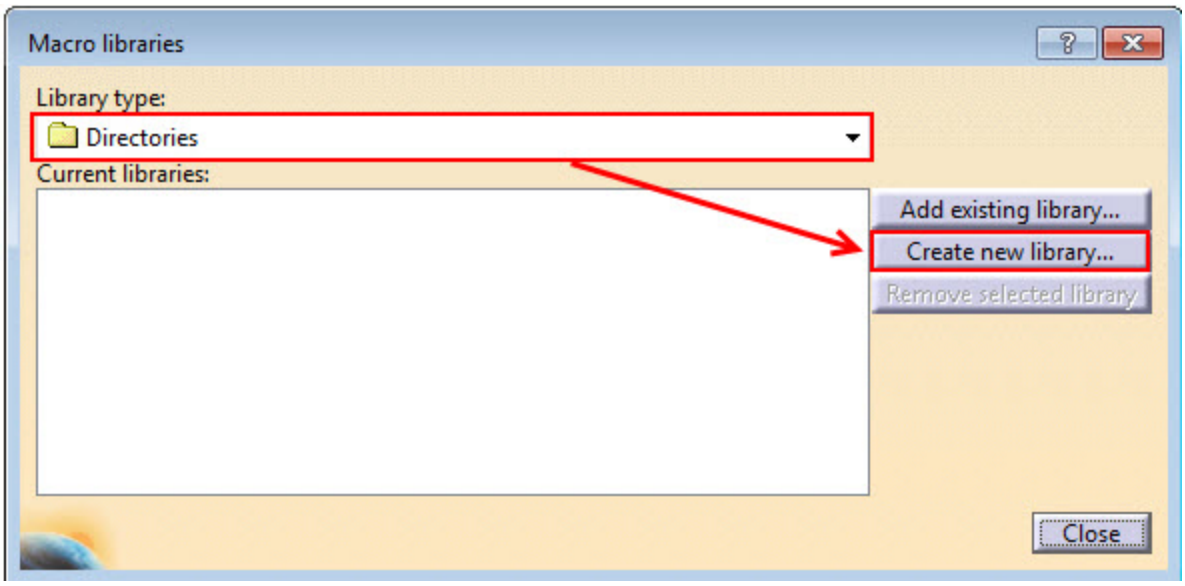
1. Go to Tools>Macro>Macros



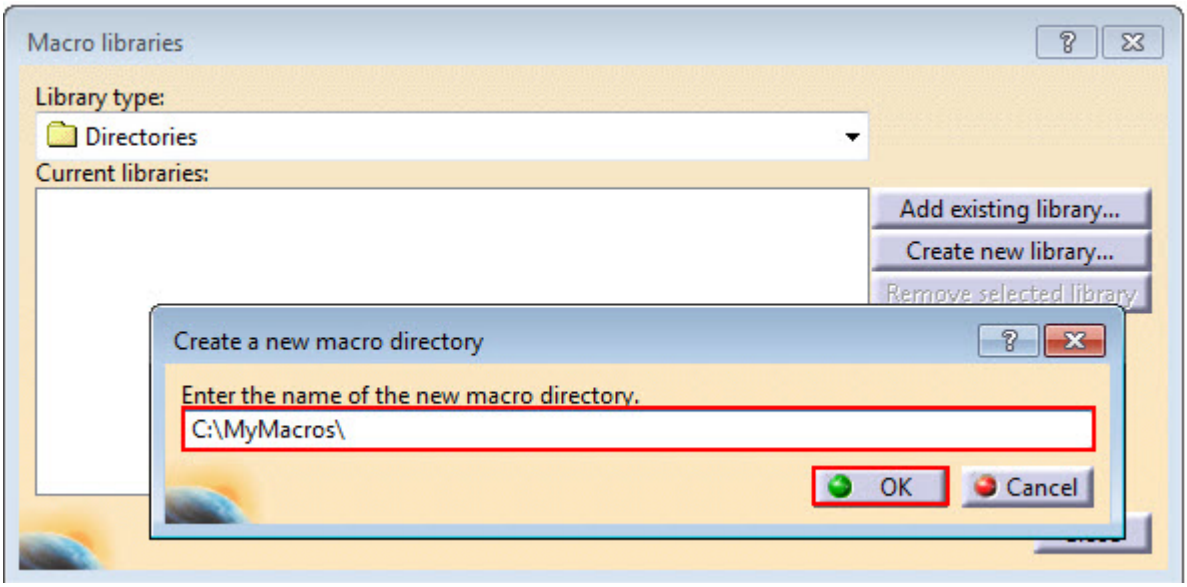
2. Click "Macro libraries…"



3. Ensure the Library type is set to "Directories" then click "Create new library…"



4. Type in the file location where you are planning on saving all of your CATIA macros then click OK.



5. Close the macros libraries window. This is where you can create CATScript macros. If you were setting up an existing library (add existing library versus Create new library) you would see a list of .CATScript files here. You only need to do this once as the library should load even after restarting CATIA.

CATIA Macro Syntax

Syntax is defined as the ordering of and relationship between the words and other structural elements in phrases and sentences. You can think of it as a particular layout of words and symbols. Each programming language is composed of its own syntax. Learning

the syntax of each programming language is crucial to creating successful macros. Think of it like this: when you see an email address (emmett@scripting4v5.com) you immediately identify it as an email address. Why is this? An email address has a correct structure in the language of the internet, its syntax. Syntax enables the programming language to understand what it is you are trying to do. Listed below are some of the key syntax features of CATIA macro programming:

- **Case Sensitivity:** By default, VBA is not case sensitive and does not differentiate between upper-case and lower-case spelling of words.
- **Comments:** Add comments to your statements using an apostrophe ('), either at the beginning of a separate line, or at the end of a statement. It is recommended that you add comments wherever possible to make your scripts easier to understand and maintain, especially if another user has to make changes to it later on down the road.
- **Indentation:** Indent or out dent script to reflect the logical structure and nesting of the statements to make it easier to read.
- **Parentheses:** It is important to use parentheses correctly in any statement to achieve the desired result and to avoid errors.
- **Semicolon (:):** Inserting a semicolon allows you to write multiple commands on the same line of code.
- **Single Quotation(‘):** To return a single quotation mark which does not indicate a comment (needed in a formula for example), you'll have to use the Chr function. Chr() is a built-in VBA function that returns the character that corresponds to the numeric value of its argument using the ASCII coding scheme. If you provide Chr with an integer value (such as 39) it will report back the character that corresponds to that value. The ASCII value of 39 is the single quote mark. Chr(34) is for the double quote mark. This is shown in an example later on in this text.
- **Spaces:** Add extra blank spaces to your script to improve clarity. These spaces are ignored by VBA.
- **Text Strings:** When a value is entered as a text string you must add quotation marks before and after the string. You can concatenate, or combine, multiple strings using a plus (+) or ampersand (&) sign: txtString = “This value is “+ TestValue. Return the left, middle, or right of a string using: left(sting, digits), mid(string, digits), or right(string, digits). Get the complete length of a string with len(string). To figure out if a string contains another string use Instr(). Convert numeric strings and vice versa using str(num) or val(string).
- **Underscore(_):** Carry over to next line (line concatenation symbol).

Syntax is often the most frequent cause of a macro program not working and throwing an error. There is a correct way to write your code. It takes practice, patience, and persistence to understand but over time it will become second nature to you. If you follow the rules the programming languages will understand you otherwise you will get errors.

Dealing with syntax can be tough which is why programmers have developed tools to help create programs correctly. These are called IDE, or integrated development environments (remember that definition defined earlier?). The Visual Basic editor in CATIA is the perfect

example (open it with Alt+F11). It contains a built-in syntax checker and works much like a grammar checker found in software like Microsoft Word and will even give you hints about what it *thinks* you meant.

Program Structure and Format

Macros in CATIA use **sequential programming**, meaning when you run your macro program, the code will be read by the computer line by line, generally from top to bottom and from left to right, just like reading a book. The program flows down the code until it hits a decision point where it may have to jump to a different portion of the program or re-run lines of code that it already read. The program chooses what lines of code to execute based on the state of the variables you have defined.

Macros can also be **event driven** meaning the execution of the program depends on an event, such as a user clicking a button with the mouse, an object being selected, certain text or characters being typed into a form, etc. One common decision point is an If...Else statement. If the variable equals x then do y, else do z. Another one is the while loop which will execute the lines of code within the loop over and over until the condition becomes false. There are more examples of decision points and control structures but we'll get to those later on in the text.

It's a good idea to maintain the same format and order on every single program you create. It's very useful to keep your code clean and tidy, especially if other programmers are going to work on it later. Remember; keep the code as simple as possible. Always try to use as few lines of code as possible (which you'll be glad you did when you have to troubleshoot one of your macros). You can reduce the number of lines of code using a colon (:). A common macro format may look something like this:

```
'my name
'date last revised
'who the macro was created for
'description of what the macro does
'————–separator————-
Main code begins here, usually with Sub CATMain
        'define variables, constants, etc.
        'error handling
        'check if part or product is in design mode or not
        'update part, drawing, etc.
        'double check to make sure all loops are closed
End Sub
```

Macro Recording

One method for creating macros is by recording your mouse actions. For macros recorded in a folder or in a CATPart or CATProduct, Dim statements (declarations) will be recorded for CATScript but not for MSVBScript. For macros recorded in a .catvba library, "MS VBA" is the only choice. Macros **cannot** be recorded while in the drafting workbench. A few things to keep in mind when recording a macro:

- DON'T: Switch workbenches while recording a macro.
- DON'T: Record more than is absolutely necessary.
- DON'T: Use the UNDO button when recording a macro.
- DO: Be aware of CATSettings when recording.
- DO: Exit sketches before stopping recording.
- DO: Check each macro after it's recorded.

Always UNDO what you just recorded and run the macro (you are able to undo CATIA macros after you've run them which is a good way to check if they work as expected or not). If the macro works from within CATIA and repeats what you just did, then the macro obviously works fine. If it does NOT work from within CATIA, you need to fix it. If it does NOT work from within CATIA it will NOT work once you cut and paste it into a VB application.

Look through the recorded macro. Many times extra lines of code are added which are not necessary. This is based on the order of steps that you do as you record the macro. These unnecessary lines can be removed. Recorded macros do not contain any comments or explanations of what is happening in the code and input parameters are never recorded.

For example, if a macro is recorded to zoom in and then zoom out it might display the following code:

```
Dim viewpoint3D As viewpoint3D
Set viewpoint3D = Viewer3D.viewpoint3D
Viewer3D.ZoomIn
Set viewpoint3D = Viewer3D.viewpoint3D
Viewer3D.ZoomOut
Set viewpoint3D = Viewer3D.viewpoint3D
```

Notice how the "Set Viewpoint" command appears multiple times? This is unnecessary in this situation. The viewpoint only needs to be set once after the Dim statement (setting and declaring will be explained in more detail soon).

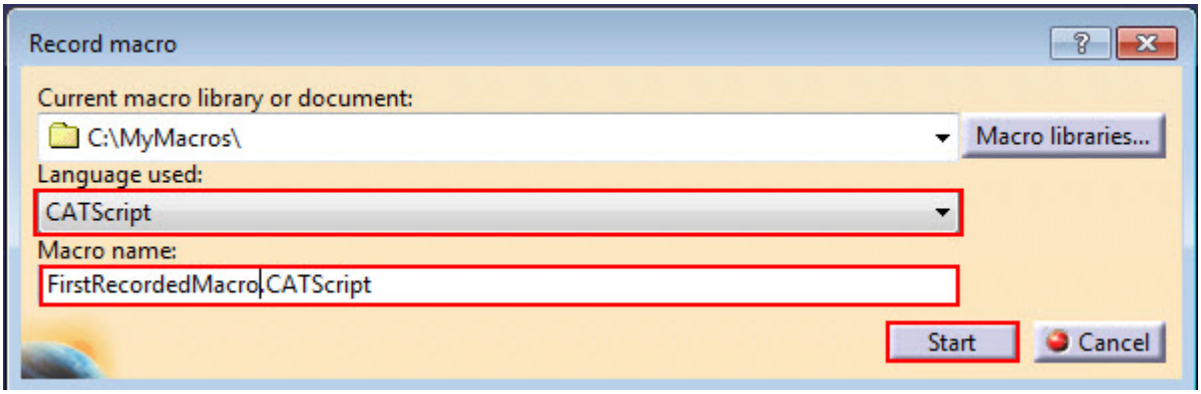
Often times you might record a macro with a CATPart active and open it in its own window. All goes smoothly and the macro replays fine. Then, the next day you replay the

macro again but this time you may have some other document type open or maybe a part is open but it is in a product assembly. Usually, the macro will fail because when the code was recorded a CATPart was the active document but now it is not. You need to add your own error handling to the code to ensure this doesn't happen. This is one advantage to writing custom code and knowing the fundamentals of CATIA macro programming.

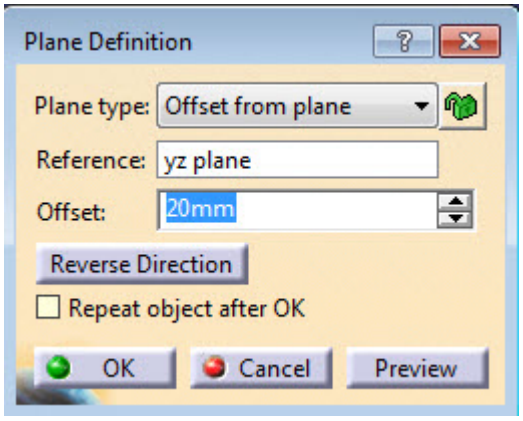
Tutorial 1.2: Record a macro

In this tutorial you'll record some actions then examine the recorded code.

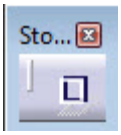
1. Open a new CATPart. The name of the part doesn't matter.
2. Start recording a macro by going to: Tools > Macro > Start Recording.
3. Set the current macro library to the directory you setup in Tutorial #1. Choose CATScript as the language used and give your macro a name, such as FirstRecordedMacro.CATScript, then click start to begin recording your mouse actions.



4. CATIA is now recording all of your actions. Create a plane, offset 20mm from the default yz plane.



5. Click the Stop Recording button to stop recording the macro.



6. Click the Undo button to undo the plane creation macro you just recorded.
7. Alt+F8 to open the macros window. Select your FirstRecordedMacro code then click Run. The macro should re-create a 20mm offset plane just as you did before.
8. If you open the macro window again but this time select edit, the code should look something like this:

```
Language="VBSCRIPT"
Sub CATMain()

Dim partDocument1 As Document
Set partDocument1 = CATIA.ActiveDocument

Dim part1 As Part
Set part1 = partDocument1.Part

Dim hybridShapeFactory1 As Factory
Set hybridShapeFactory1 = part1.HybridShapeFactory

Dim originElements1 As OriginElements
Set originElements1 = part1.OriginElements

Dim hybridShapePlaneExplicit1 As AnyObject
Set hybridShapePlaneExplicit1 = originElements1.PlaneYZ

Dim reference1 As Reference
Set reference1 = part1.CreateReferenceFromObject(hybridShapePlaneExplicit1)

Dim hybridShapePlaneOffset1 As HybridShapePlaneOffset
Set hybridShapePlaneOffset1 = hybridShapeFactory1.AddNewPlaneOffset(reference1,
20.000000, False)

Dim hybridBodies1 As HybridBodies
Set hybridBodies1 = part1.HybridBodies

Dim hybridBody1 As HybridBody
Set hybridBody1 = hybridBodies1.Item("Geometrical Set.1")

hybridBody1.AppendHybridShape hybridShapePlaneOffset1
part1.InWorkObject = hybridShapePlaneOffset1
part1.Update
End Sub
```

Congratulations! You've recorded your first macro. But what does all the code mean? How do you make sense of it? Hopefully by the end of this book you'll understand every line of code and will be able to recreate the program without using the macro recorder.

CATIA Macro Variable Naming

Variables make up the backbone of any programming language. Basically, variables store information that can be accessed later by referring to a name or "variable." Let's say you have an input box in your code where the user will enter a part number. You refer to the input box as *componentName*. In other words, *componentName* is a symbolic name, or word, for the input box, the variable. You can then ask the question in your code "What value does the variable *componentName* contain?" Variables can then be declared as a type to let us know what we can do with that information. The value of a variable will not necessarily be constant throughout the duration a program is running.

How you decide to name your variables is very important. In the previous example, instead of giving the input box a variable name of *componentName* we could have called it *ImASuperAwesomeProgrammer*. However, this doesn't make a lot of sense and doesn't help us, especially if we're looking at the code months from now or another programmer has to modify the code.

There are a few rules we must follow. Variable names, also known as identifiers, must be less than 255 characters, they cannot start with a number, and special characters such as spaces and periods are forbidden. Avoid naming conflicts. Two variables cannot have the same name.

- Good Variable Names: *dimensionOne, dimension1*
- Bad Variable Names: *dimension One, 1dimension*

Variables have different types. To store your name in a variable, you would use a **String** variable. To store your age then that type of variable would be an **Integer**. How much money you make in a year would be declared as a **Double**. Declaring variables as a specific type tells CATIA the information being stored will be "a certain way." Storing a number as an Integer means it will be a whole number without a decimal place, like 5, 60, or -4000. If you try to store $58.76 as an integer you will get an error. Two variables that are both Integers can be added or subtracted to each other (50-5=45) but it wouldn't make sense to try to add two strings together (Emmett-David=???). Declaring variables as a type give us a sense of what can and can't be done with them.

Variable Type	Default Value	Description
String	Null	Text (letters, numbers, spaces, punctuation)
Integer	0	Whole number
Double	0	Any number, positive or negative
Long	0	A 4-byte integer
Boolean	FALSE	Logical statement like true or false, yes or no, etc.
CATVariant	Null	Index of a list of objects
CATBStr	Null	String of CATIA expressions

There is a naming convention called “Hungarian notation” where a letter is placed in front of the variable name to notate what type of data the variable is. For example:

- o=object (i.e. oPartDoc)
- s=selection (i.e sSelection1)
- str=string (i.e. strFilename)
- i=integer (i.e iCount)
- rng=Range (i.e. rngField)

Many programmers use “int” or “n” for integer and “o” or “obj” for object. Either or works as long as your notation is consistent and easy for others to understand.

Dimming and Setting

Variables are "dimmed" (declared) as a type, either a "primitive" type (single, double, integer, string, etc.) or an object type (more complex). If you don’t specify the type of variable VBA declares the variable as a Variant type which can accept any type of variable. In rare instances you’ll have a good reason for using a Variant. But most of the time, you should use explicit variable types. One reason for doing so is that your code will run faster but the primary reason is that you’ll reduce your coding errors due to the fact that CATIA enforces the type of variable you specify.

The second reason that it’s always a good idea to declare your variable types explicitly is that it helps to document your intentions when you write code. This documentation is critical if you look at your code after several months have passed. Knowing whether a variable was intended to contain a number or text can make it much easier for you to read your old code, find errors, and then continue coding.

The "Dim" command is used to allocate a variable of any type. Primitive variables are populated with a value. Declaring variables with a ‘Dim’ statement helps save memory while running the program. If you have two variables declared as integers you could subtract or add them. But if you have two variables stored as names it wouldn’t make sense to subtract them because that just doesn’t mean anything! Declaring the type of variable allows you to make sense of what a variable can and cannot do.

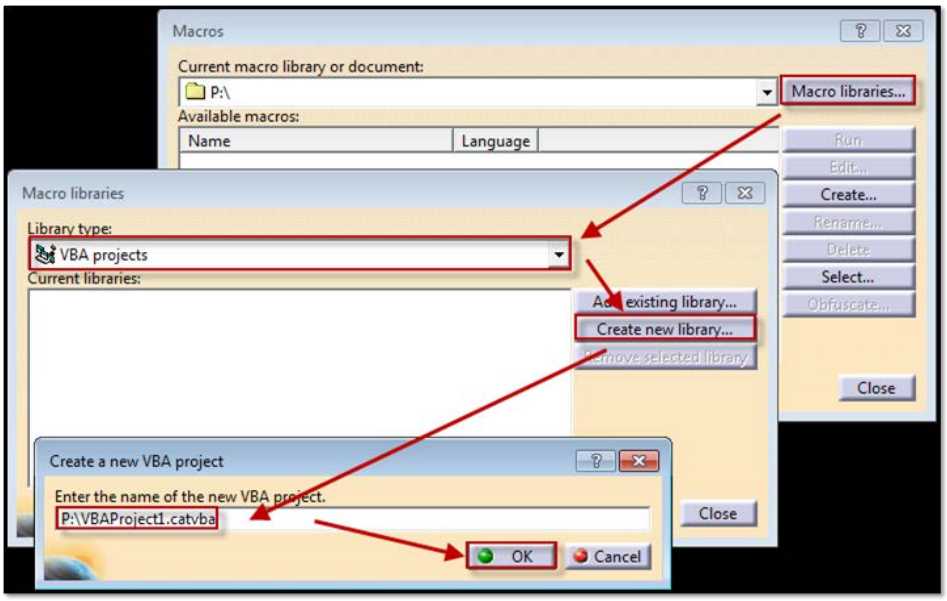
Multiple variables of the same type can be declared all at once by using commas (Dim iNum1, iNum2, As Integer). However, in VB6 you need a separate line of code for every Dim statement (meaning you can’t use `Dim w, h As String` it must be written as `Dim w As String: Dim h As String`).

After a variable is declared through dimming then it is “Set” or given a value. Do NOT use Set to assign a variable to an object property (`MyCamera3D.ViewPoint3D`) or an intrinsic type variable (`mystring = “directory”`) For object variables, the "Set" command is used to "point" the variable to the object. Within the program, the variable then represents that object unless it is "Set" to a different one.

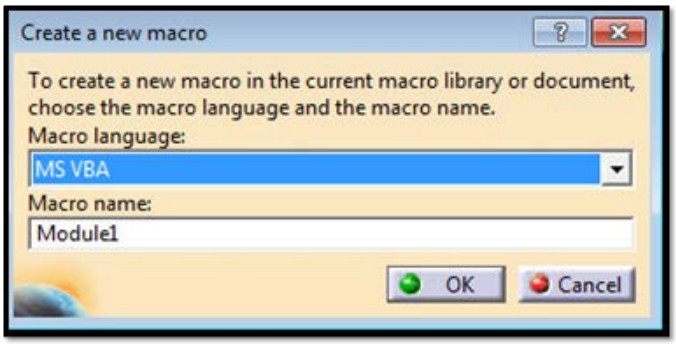
Tutorial 1.3: Create Your First Custom "Hello World" Macro

It's time to follow along to create your first CATIA macro. In the programming world, the term "Hello World" typically refers to the most basic program that can be written in a given language. Think of it like this, if you can write a Hello World CATIA macro, this means you are able to setup the necessary tools, write the code, compile the macro, and run the program. Not a bad start!

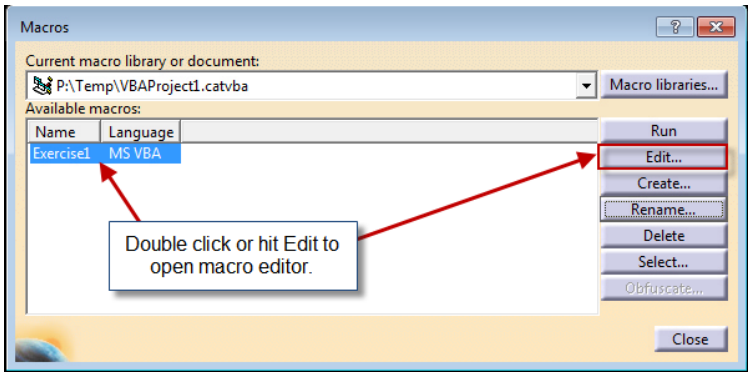
1. To begin, create a new macro library using VBA Projects instead of Directories.



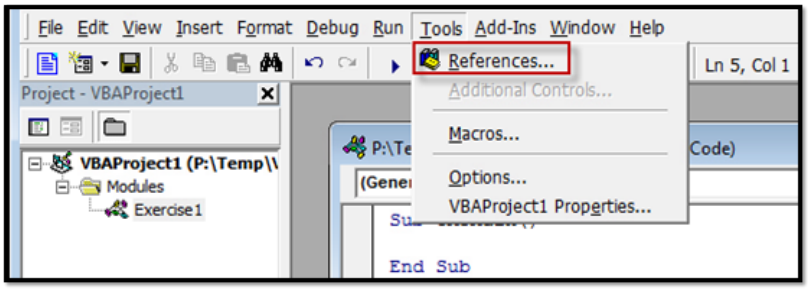
2. Create a new macro using the MS VBA language (most examples in this book are either VBA or CATScript).



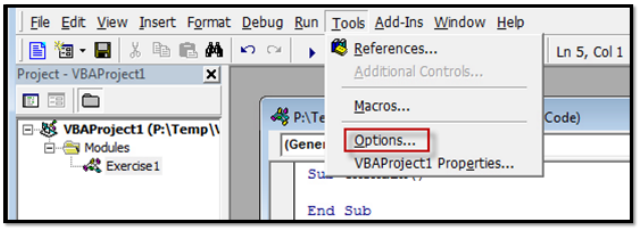
3. Rename the macro to Exercise1 then double click or hit edit to begin editing the macro in the VBA editor.



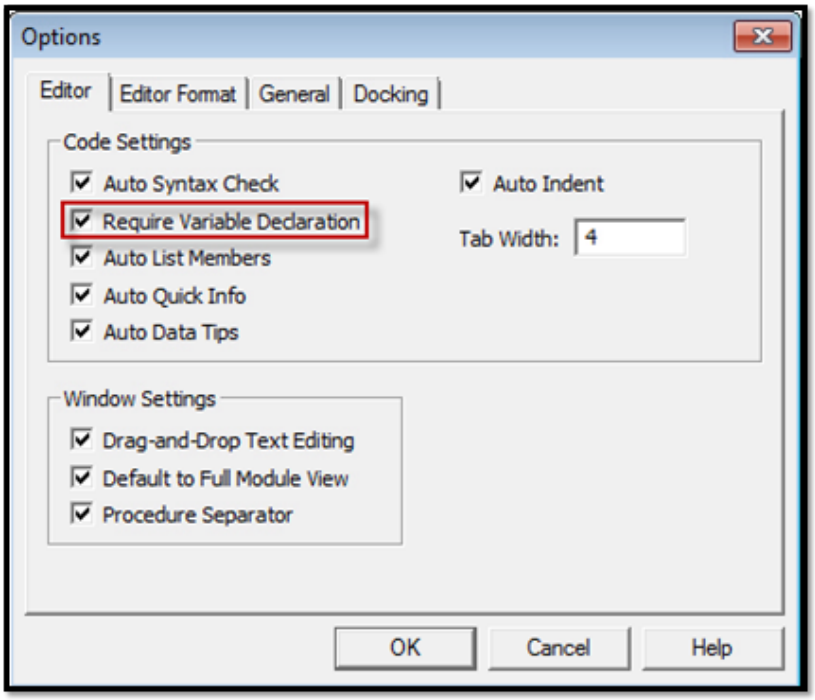
4. Go to Tools > References and make sure all available CATIA references are selected.

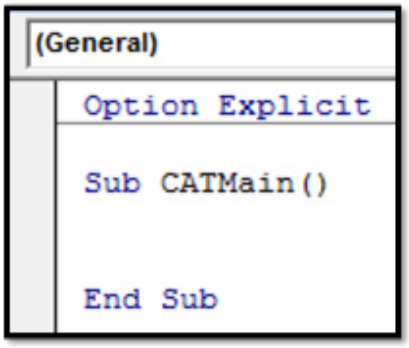


5. By default, CATIA doesn't force you to declare your variables. However, you should change that setting immediately. In CATIA (or Excel or other software), press Alt+F11 to launch the Visual Basic Editor (VBE). Go to Tools>Options.



6. In the Editor tab of the Options dialog, make sure that there's a check mark in front of "*Require Variable Declaration*"





7. After you click OK, insert a new Module. When you do so, you'll see Option Explicit as the first line of the module. This line tells CATIA that all variables must be declared making your code easier to read and follow. Option Explicit will be the first line in our code.
8. A CATIA VB program or "macro" consists of a "Subroutine" named CATMain(). CATIA only recognizes Sub CATMain as the entry point to any VBA application. Every Sub must end with an End Sub. Between the two Sub statements, type the text as shown:

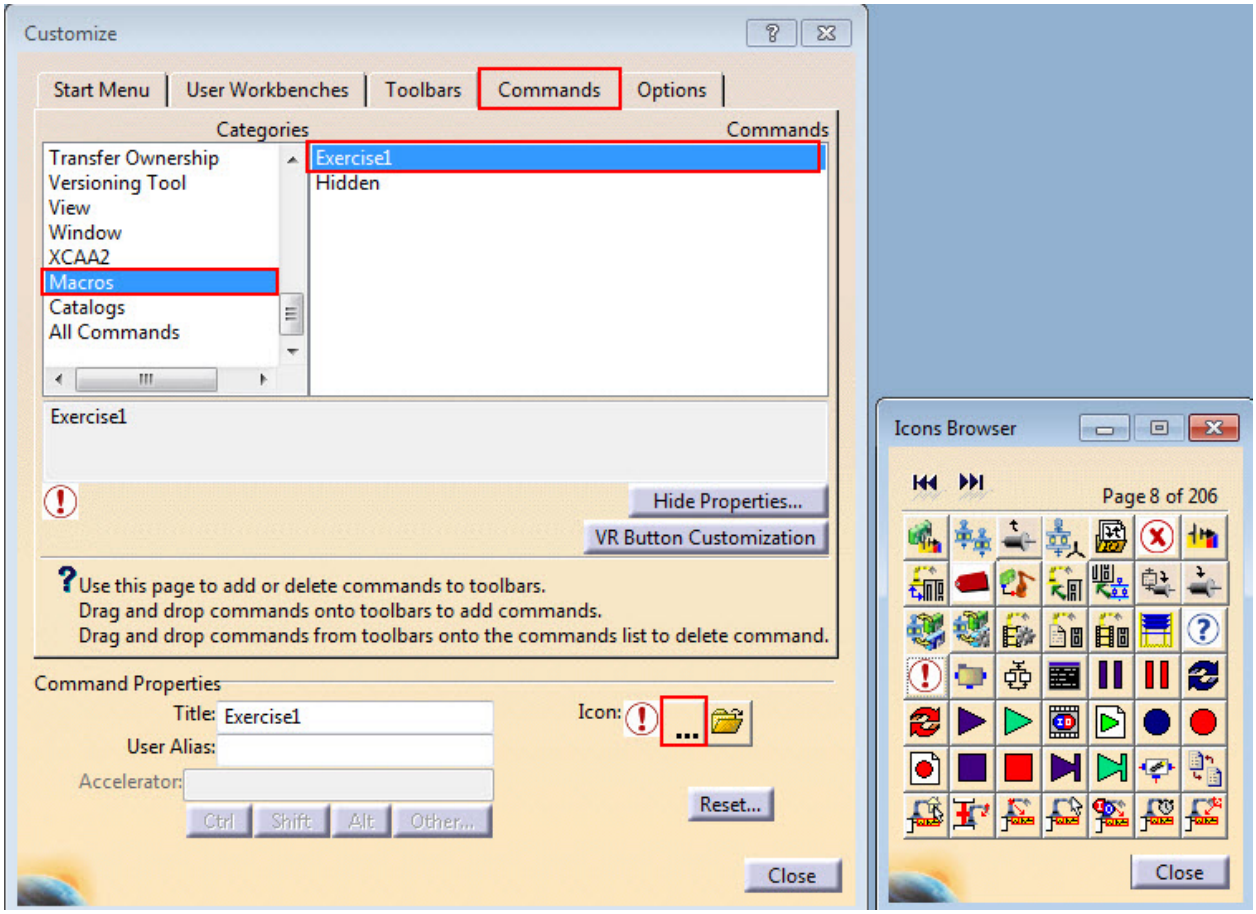
```
Option Explicit
Sub CATMain()

'declare a variable as a String type
Dim strMessage As String

'define the String variable as the text Hello World
strMessage = "Hello world!"

'have a message box display the text in CATIA
Msgbox strMessage
End Sub
```

9. Save then run the macro by clicking the play icon, going to the run menu, or by pressing F5. A message box should pop-up displaying “Hello world!”. Click ok to end the program. Voila! You have just setup, coded, and ran your first CATIA macro, your first accomplishment as a programmer.

Tutorial 1.4: Create a Macro Icon

Instead of having to open the macro window to run your program every time, it can be much quicker to assign an icon to macro and place it on a toolbar for quick access. To assign an icon for your recently created Hello World macro:

1. Go to Tools>Customize then the Commands tab.
2. Scroll down to Macros and click "Show Properties." Under Commands, select the Hello World macro then click the "…" box. The icon browser pops up. There are over 9,000 unique icons to choose from (or you can create your own). Select one that makes sense to you and is different from any others you currently use then hit Close.



3. Finally, drag and drop the macro file from the command window to whatever toolbar you would like the icon to appear on (such as "Graphic Properties" or create a new one).

Now you can click the Icon to run your macro! You can also setup custom keyboard shortcuts from within the same window as well or create your own custom toolbar by going to Tools>Customize>Toolbars>New.

Chapter 1 Quiz

Test your knowledge of the material presented in Chapter 1.

1. True or False: Macros can be recorded in the Drafting workbench.

2. What is the keyboard shortcut to open the CATIA macros window?

3. What is the keyboard shortcut to open the Visual Basic Editor?

4. How does code generally flow in a CATIA CATScript macro?

   a. From bottom to top, left to right

   b. From top to bottom, left to right

   c. From top to bottom, right to left

   d. From bottom to top, right to left

5. True or false: 1MainComponent is a good variable name.

6. What does IDE stand for?

7. What variable type would I use to store this data: "My birthday is June 1st."

8. What is the default value of String type?

9. What is the default value of Integer type?

10. True or False: File+SaveAs will **keep the same UUID**.

11. If you need to store this data: -298.056 what variable type would you use?

12. If you're going to make a variable name of *myName* and assign a value of Emmett Ross, what type would you use?

13. What type of variable would you use to store this data: 76?

14. If using CATIA V5, what will the following code's message box output be?

```
Sub CATMain()
MsgBox CATIA.Caption
End Sub
```

15. For VBA's syntax, which statement is true:
    a. VBA is not case sensitive
    b. Comments are denoted using a \ symbol
    c. Semicolon(:) is carry over to the next line
    d. All of the above