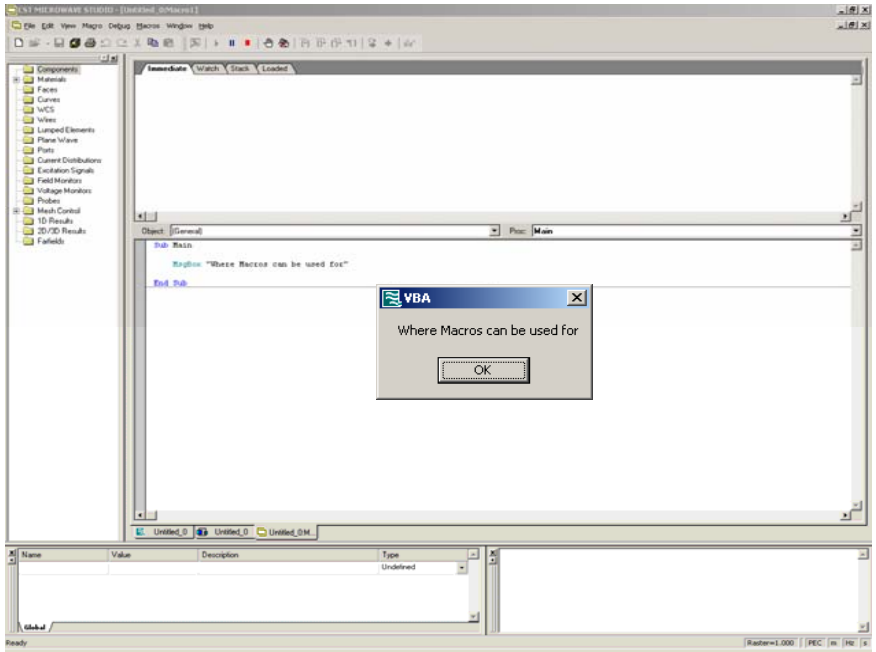
CST STUDIO SUITE™ 2006B
Application Note

Introduction in VBA Macro Usage and Programming



Existing Macros
Different Macro Types
Templates
Structure of a Macro
Example

Outline

- Why macro programming?
- Existing macros
- Different types of macros
- Creating and testing new macros
  - The integrated development environment (IDE)
  - Structure of a macro
  - How to create a macro?
- Getting more information

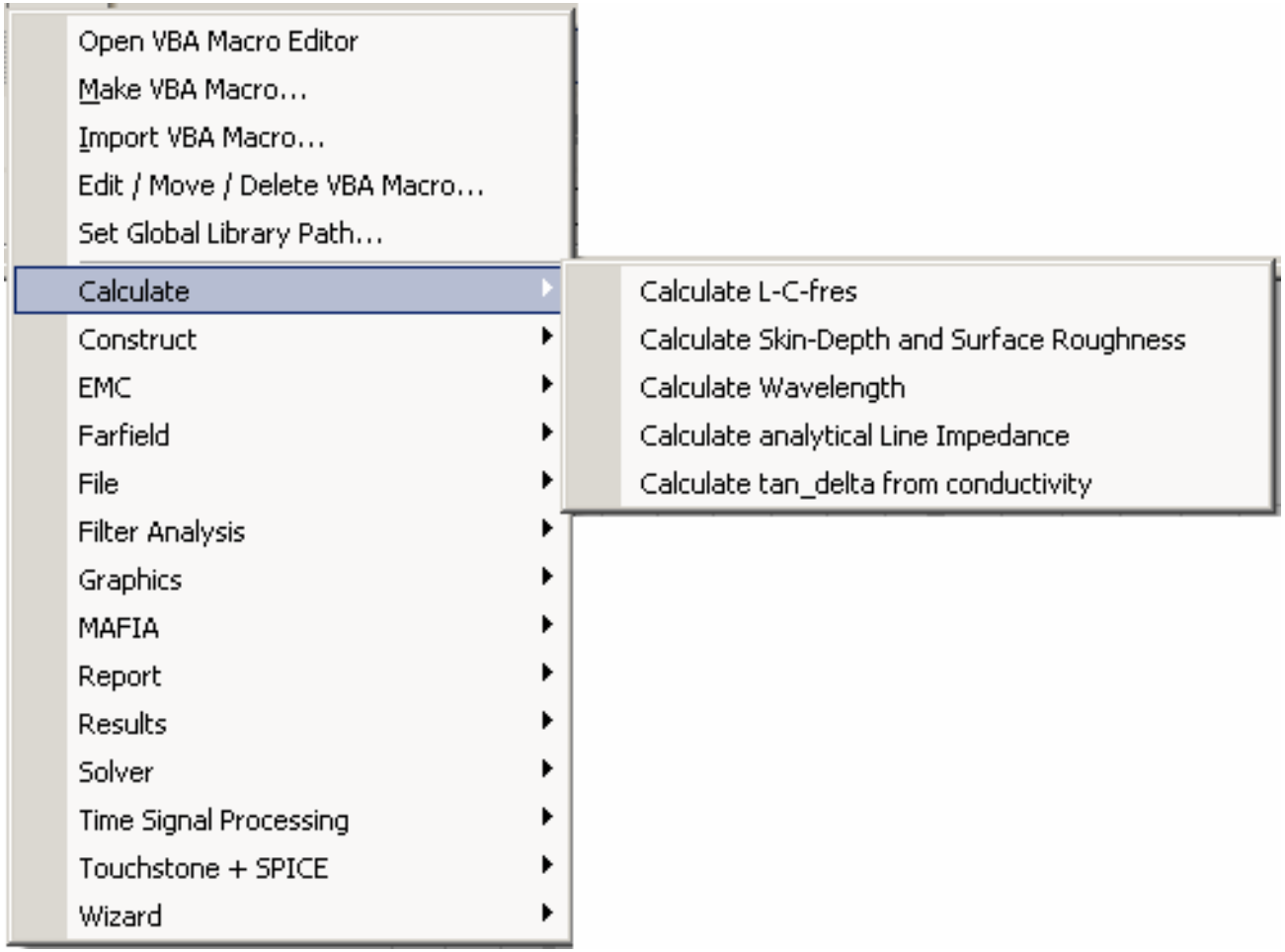
Why Macro Programming?

- Automate common tasks, save time, increase productivity
- Extend the program‘s capabilities, e.g. post processing, optimization
- Customize the program for particular applications
- Make advanced functionality available to less experienced users

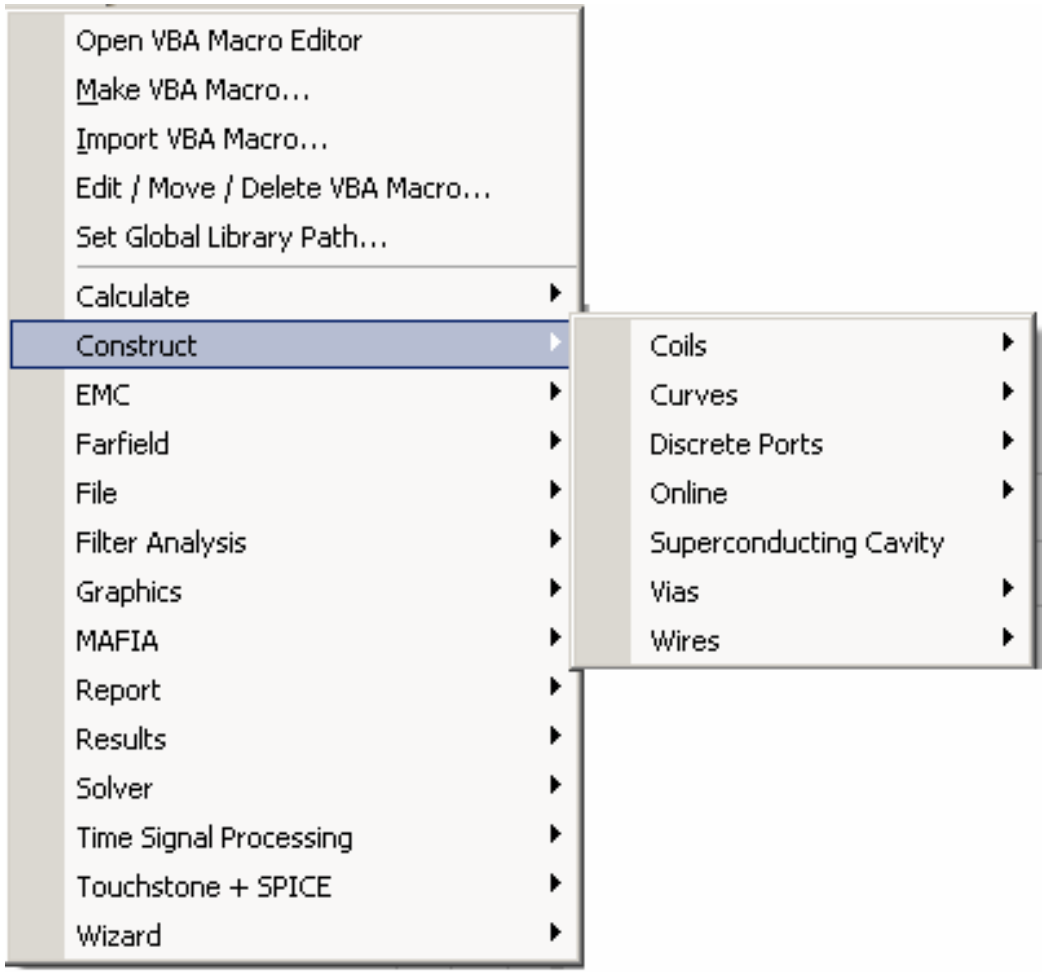
CST STUDIO SUITE™ macro language:

- Compatible to the widely used VBA (Visual Basic for Applications)
- COM based
  - CST STUDIO SUITE™ can be controlled by other applications
  - CST STUDIO SUITE™ can control other applications

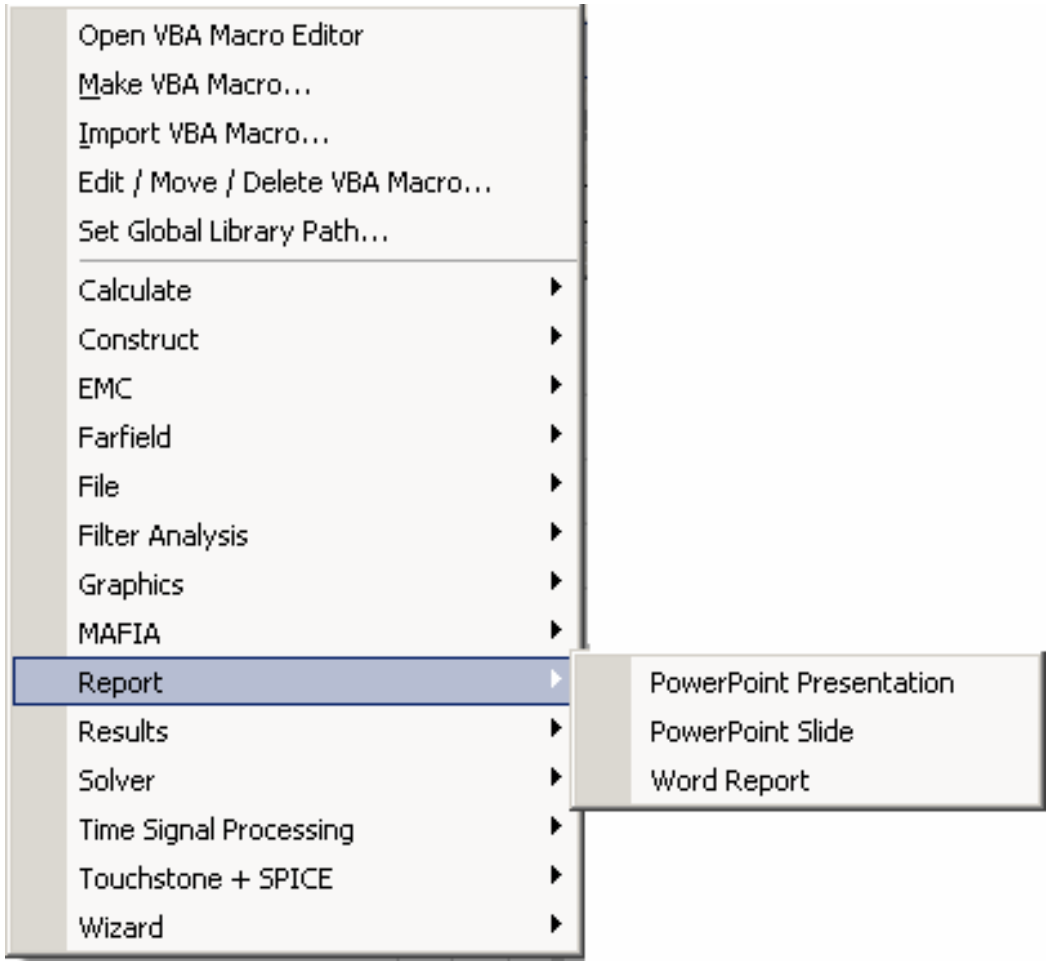
Useful, predefined macros



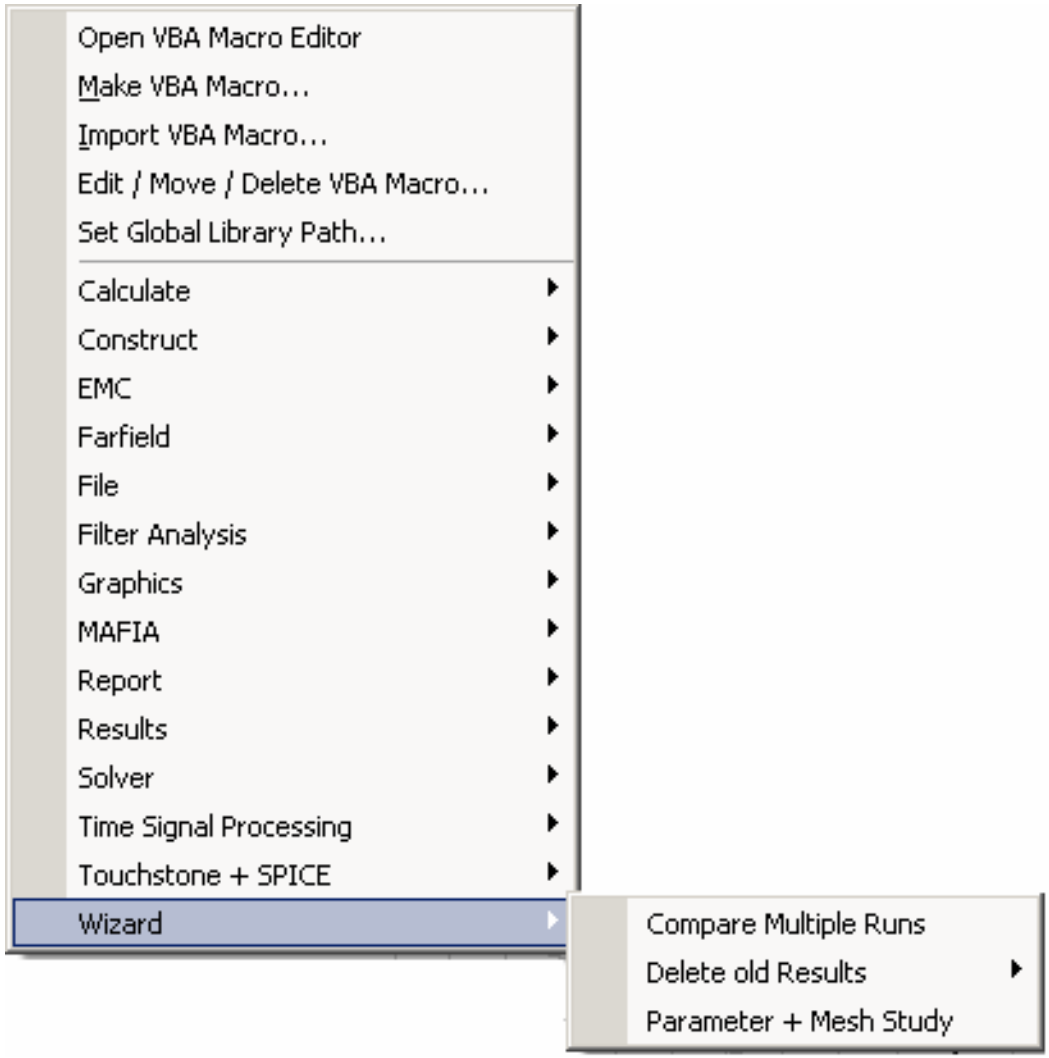
Useful, predefined macros



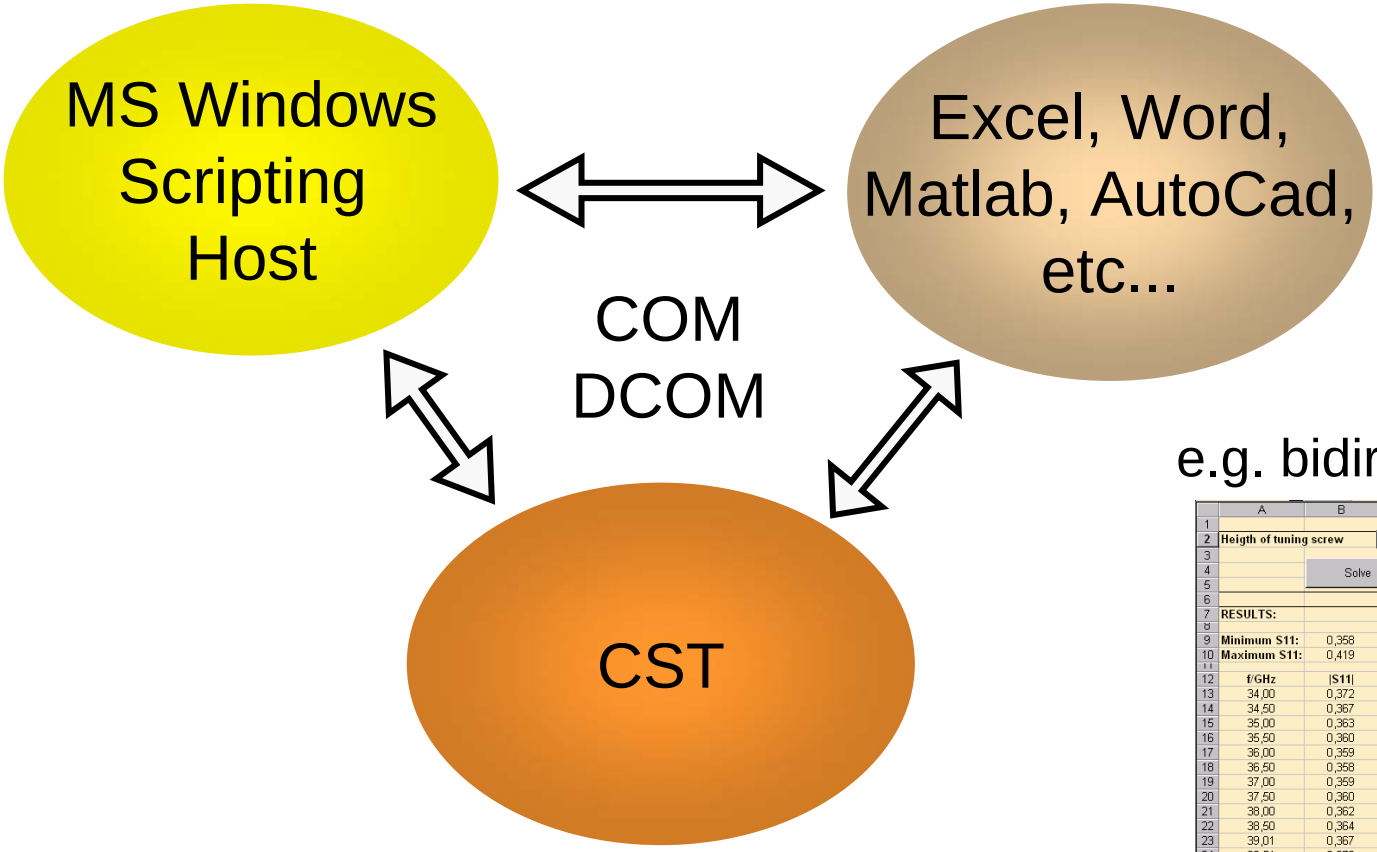
Useful, predefined macros



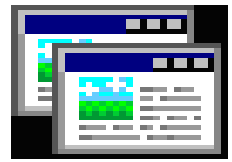
Useful, predefined macros



Integration Into Workflow

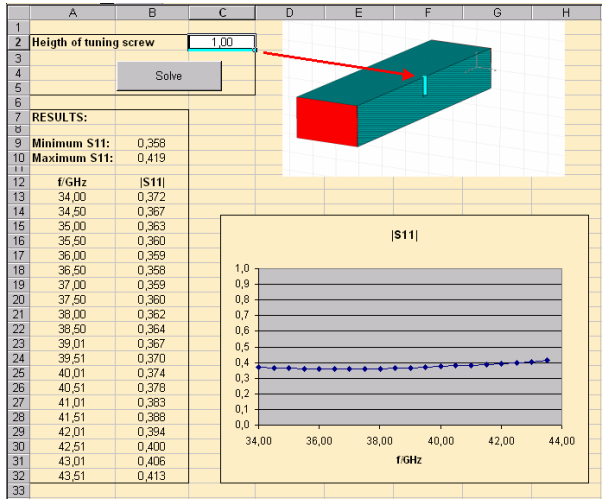


e.g. ppt-Reports



reports

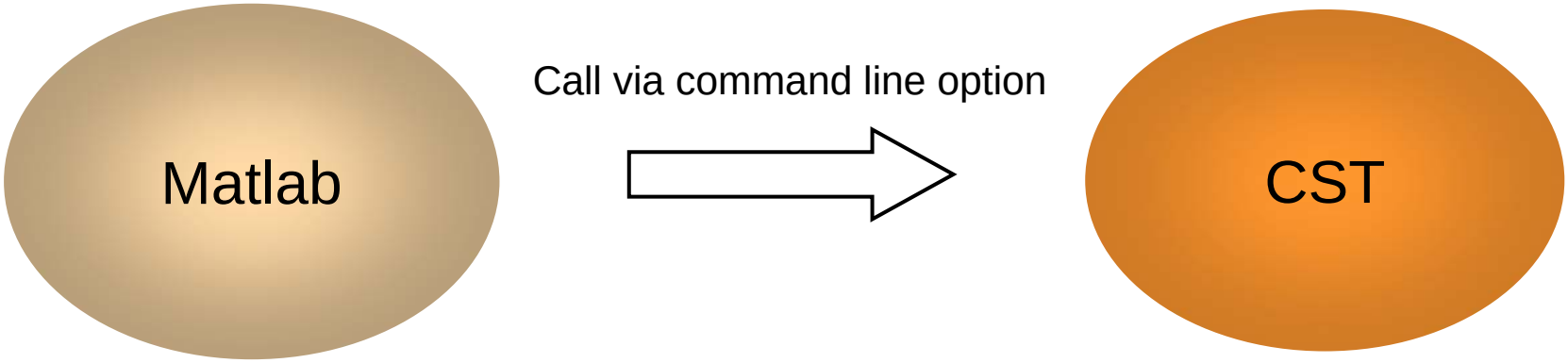
e.g. bidirectional Excel link



CST STUDIO SUITE™ can be both:
OLE client and server

OLE: CommunicationStandard for Data Exchange

Integration Into Workflow



Call: `! "C:\Program Files\CST STUDIO SUITE 2006\CST DESIGN ENVIRONMENT.exe" –m D:\MBK\Start_CST_5.bas`

Execute command in Matlab

CST MWS is called within the DESIGN ENVIRONMENT

Path of the VBA script

```
Sub Main
    OpenFile("D:\MBK\test1\test1.mod")
    Solver.Start
    Save
End Sub
```

Opens an existing CST MWS file

Start of Transient Solver

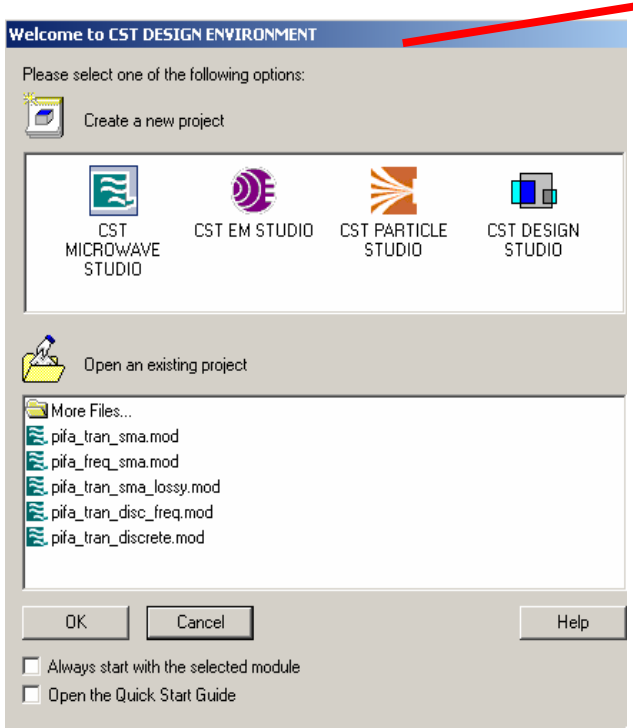
Saves results and gives control back to Matlab

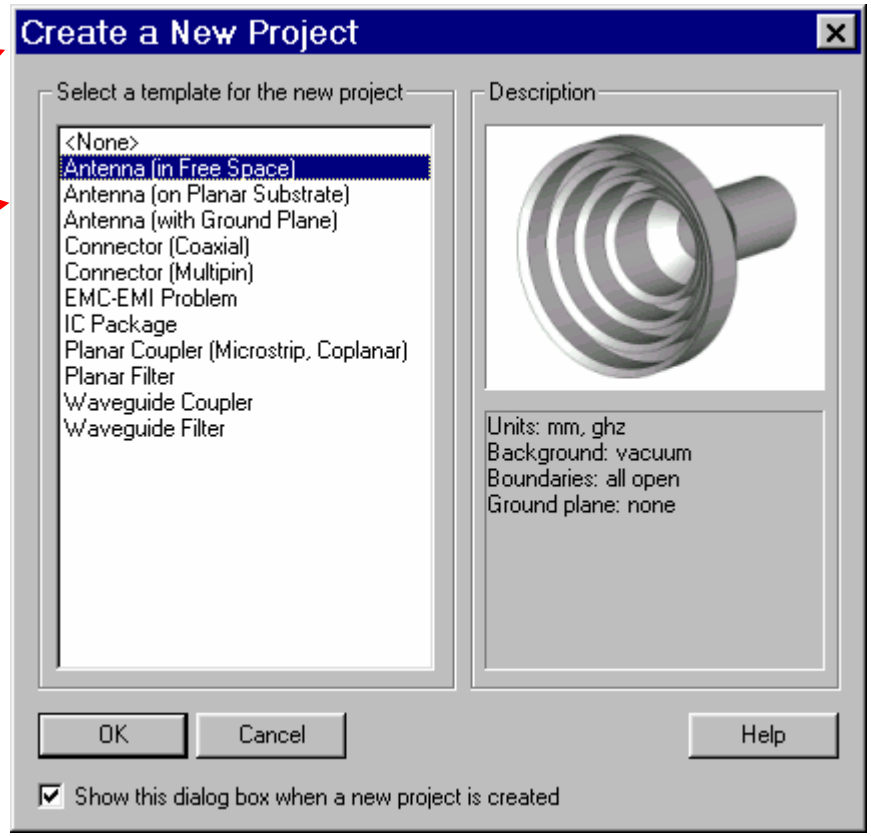
Project Templates

Customize the default settings for particular types of applications.

At the beginning: **File -> New...**

or later: **File -> Select Template...**





The template library can be easily extended

Project Templates

- Input some commands you often use for your CST MWS structures, e.g.:
  - Frequency range, units, Background-mat + boundaries
  - Definition of materials (parameters, favorite colours ...)
  - Working plane settings (especially **snapping** !)
  - Monitors at favorite frequencies, ...
- Open the history list
- Mark the commands, press „Macro“
- Give a name to your macro, e.g. „File / My defaults“
- Click „Make globally available“, then „OK“

Project Templates

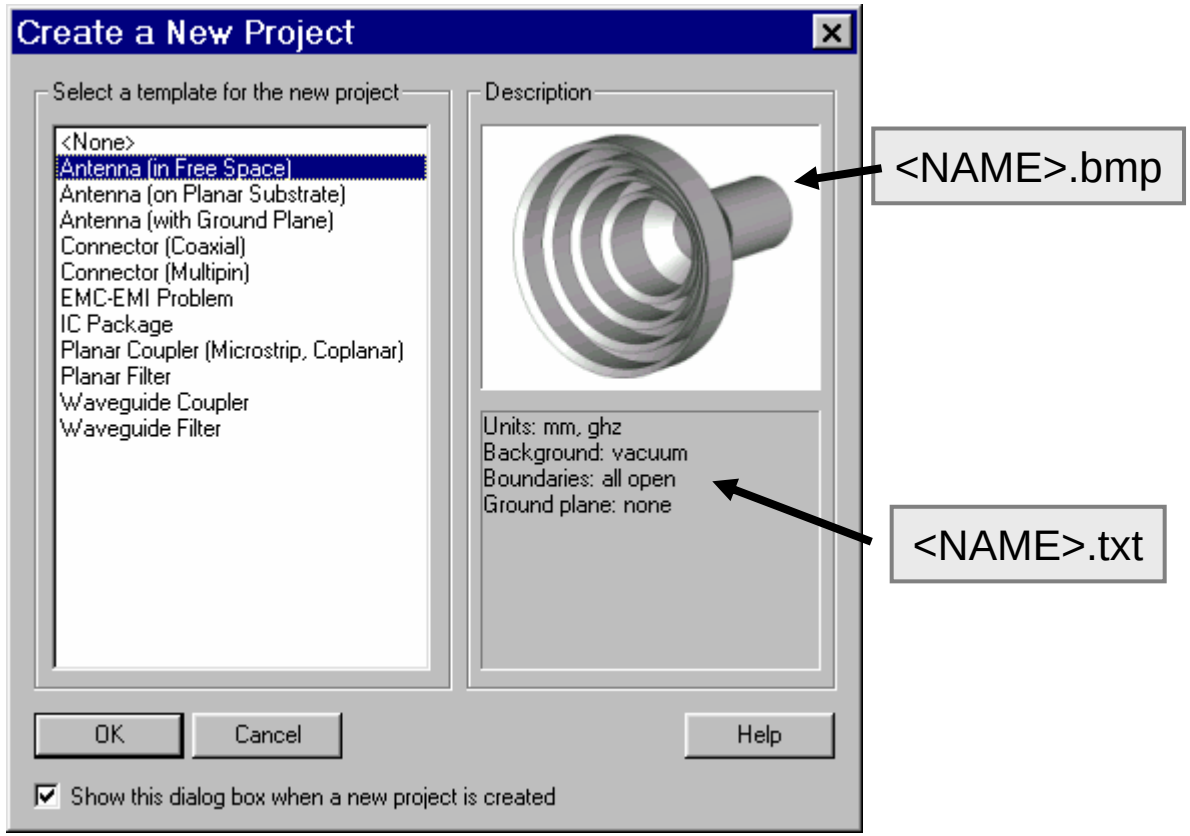
Source-Files are stored in
<GLOBALMACROPATH>\New Project Templates

Each template <NAME> consists of 3 files:

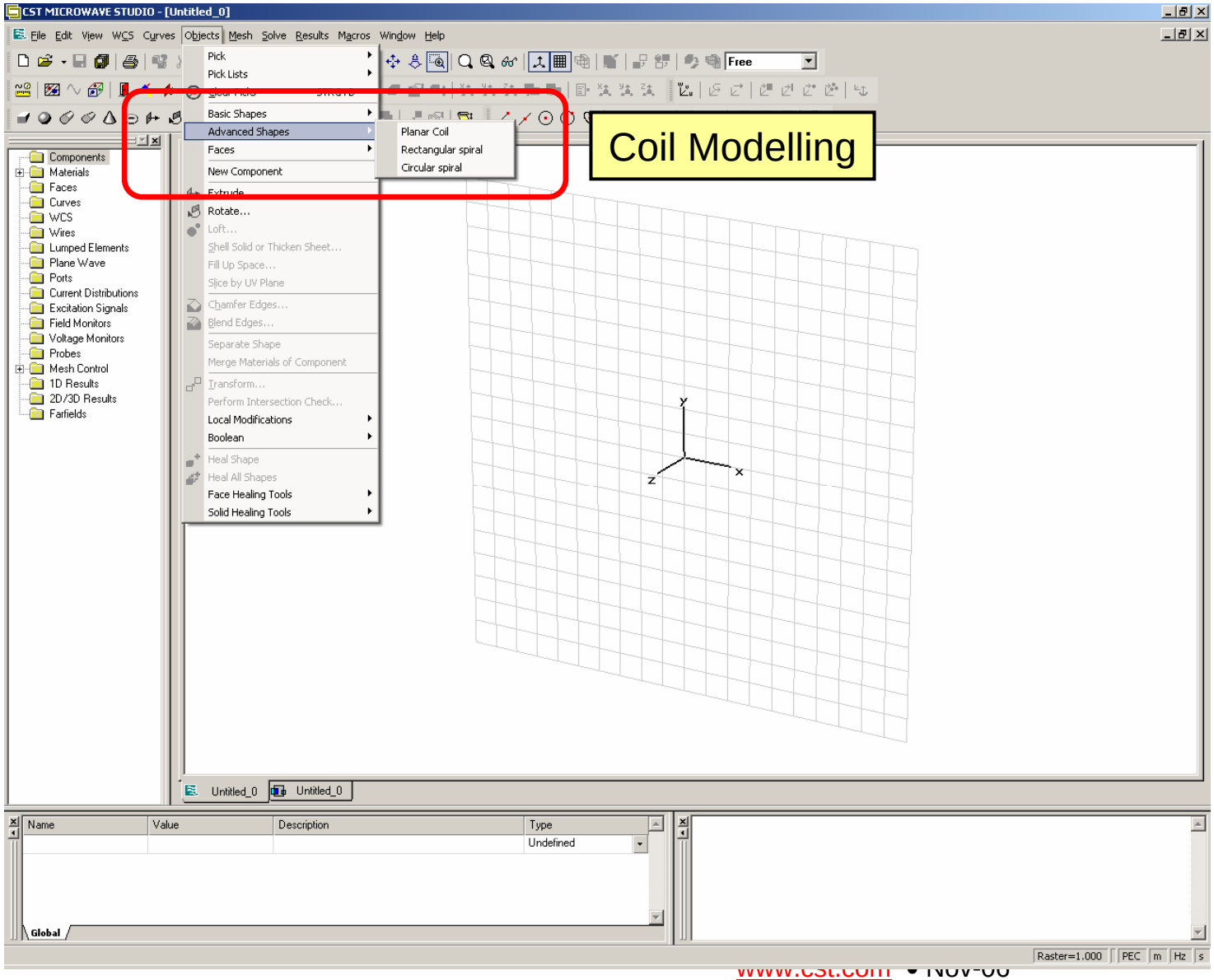
1) <NAME>.tpl (required)
(contains VBA-commands)

2) <NAME>.bmp (optional)
(contains displayed picture)

3) <NAME>.txt (optional)
(contains displayed description)



Customize PullDown Menus via menu.cfg



Customize PullDown Menus via menu.cfg

menu.cfg - Example

```
Objects\Advanced Shapes[+Basic Shapes]\Planar Coil
COMMANDMACRO
macro_construct_coil.bas

Objects\Advanced Shapes[+Basic Shapes]\Rectangular spiral
STRUCTUREMACRO
macro.939
```

Note:

- menu.cfg and macro.### or .bas in globalmacropath
- STRUCTUREMACRO goes into history list,
- COMMANDMACRO (*) will not be entered in the history list
- [+Basic Shapes] -> new entry will be inserted after "Basic Shapes"

Outline

- Why macro programming?
- Existing macros
- Different types of macros
- Creating and testing new macros
  - The integrated development environment (IDE)
  - Structure of an CST MWS macro
  - How to create a macro?
- Getting more information

The Different Types of Macros

Different types depending on the functionality

- Macros for structure generation
  - Project templates
  - Structure macros
- Macros for advanced control
  - Control macros
- Macros for extending functionality
  - User defined excitation functions
  - User defined parameter sweep watches
  - User defined goal functions
  - Result templates for customized post-processing

The Different Types of Macros

Structure macros and control macros

- Structure macros

```
' Construct / Coils / Trapezoidal-Spiral
```

  - Modify the structure
  - Stored in the history list for parametric model definition
  - Example: Creation of advanced geometry, e.g. spirals,…

- Control macros

```
' *Calculate / Calculate Wavelength
```

  - Do not modify the structure
  - Do not need to be stored in the history list
  - Example: Specific post processing calculations, e.g. group delay, TDR, etc....
  - User defined goal functions, etc. can be considered as a special type of control macro

The Different Types of Macros

Project macros and global macros

- Project macros
  - Can be either command macros or structure macros
  - Specific for a particular project
  - Stored with the project. Not available for other projects as well

- Global macros
  - Can be either command macros or structure macros
  - Generally useful
  - Stored in a global location (*Global Macro Path*). Can be shared accross projects

The Different Types of Macros

Result Templates for customized postprocessing

- Store in <globalmacropath>/Result Templates/1D/my_template.rtp or …/0D/my_template.rtp
- Will be evaluated after each solver run.
- Can perform just an action or return 1D or 0D values.

Performs fixed combine results

```
Option Explicit

Function Evaluate1D() As Object

    With CombineResults
        .Reset
        .SetMonitorType ("frequency")
        .SetOffsetType ("phase")
        .EnableAutomaticLabeling (True)
        'fixed combine results for two modes excited
        'at the same port, phase shift 90 degree
        .SetPortModeValues (1, 1, 0.5, 0.0)
        .SetPortModeValues (1, 2, 0.5, 90.0)
        .Run
    End With


   Set Evaluate1D = Result1D("")
   Evaluate1D.Initialize 1

End Function
```

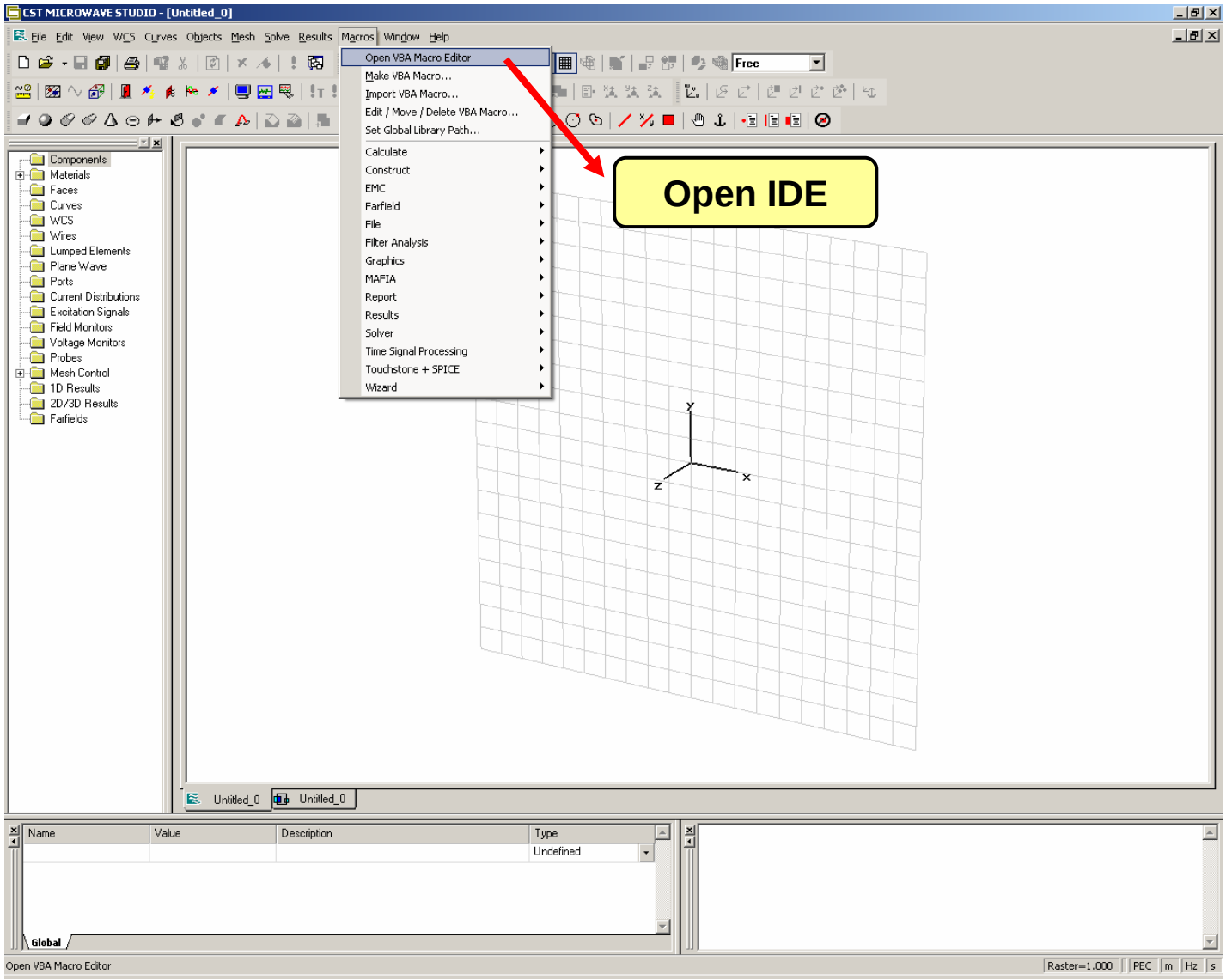
Returns a value

```
Option Explicit

Function Evaluate0D() As Double

    Dim resultkey As String
    Dim cst_value As Double

    Dim dfactor As Double

    If (CInt(GetScriptSetting("multiply","0")) = 0 ) Then
        ' no
        dfactor = 1.0
    Else
        'yes
        dfactor = Evaluate(GetScriptSetting("factor",""))
    End If

   Evaluate0D = dfactor * (Mesh.GetNx-1)*(Mesh.GetNy-1)*(Mesh.GetNz-1)

End Function
```

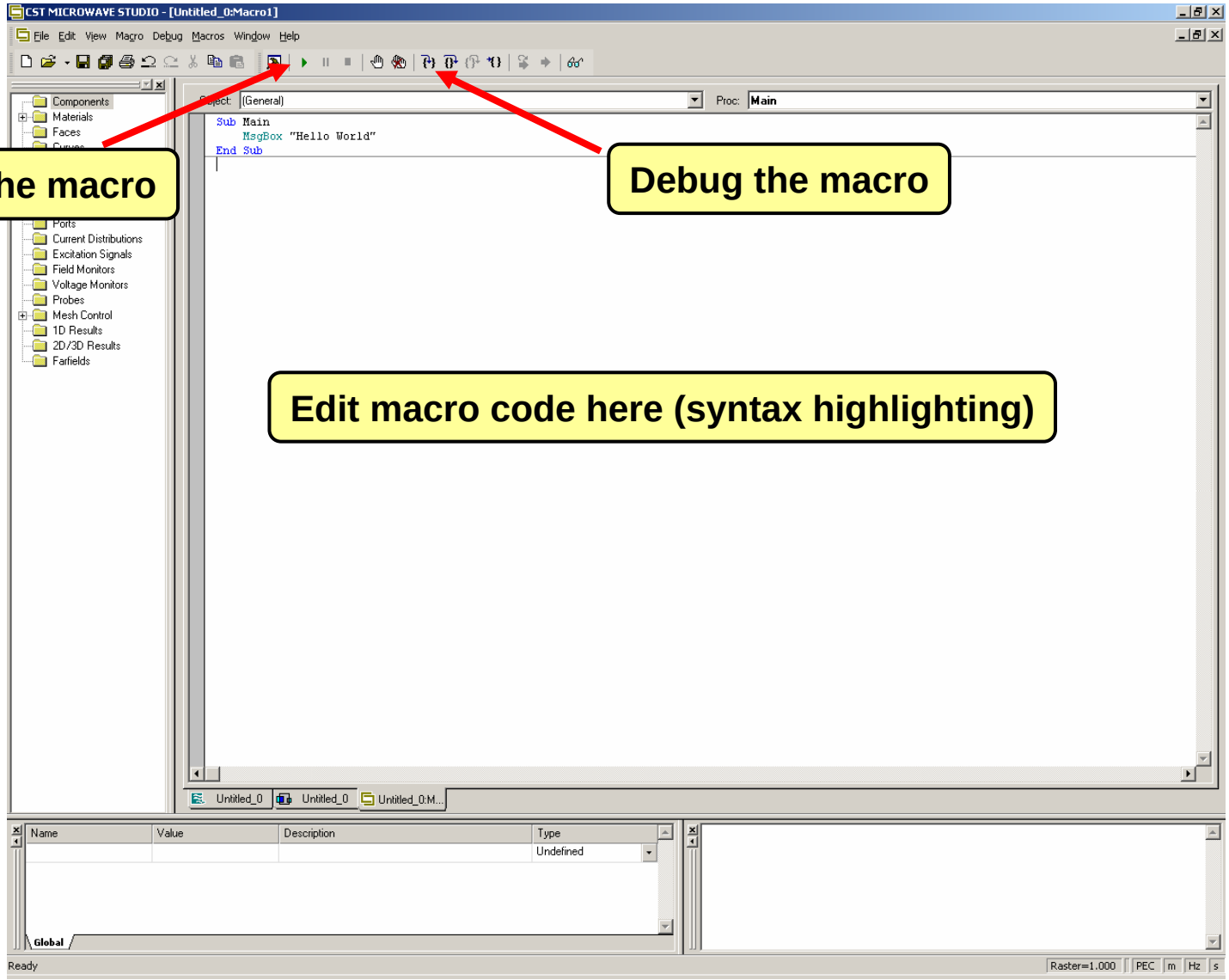

Outline

- Why macro programming?
- Existing macros
- Different types of macros
- Creating and testing new macros
  - The integrated development environment (IDE)
  - Structure of an CST MWS macro
  - How to create a macro?
- Getting more information

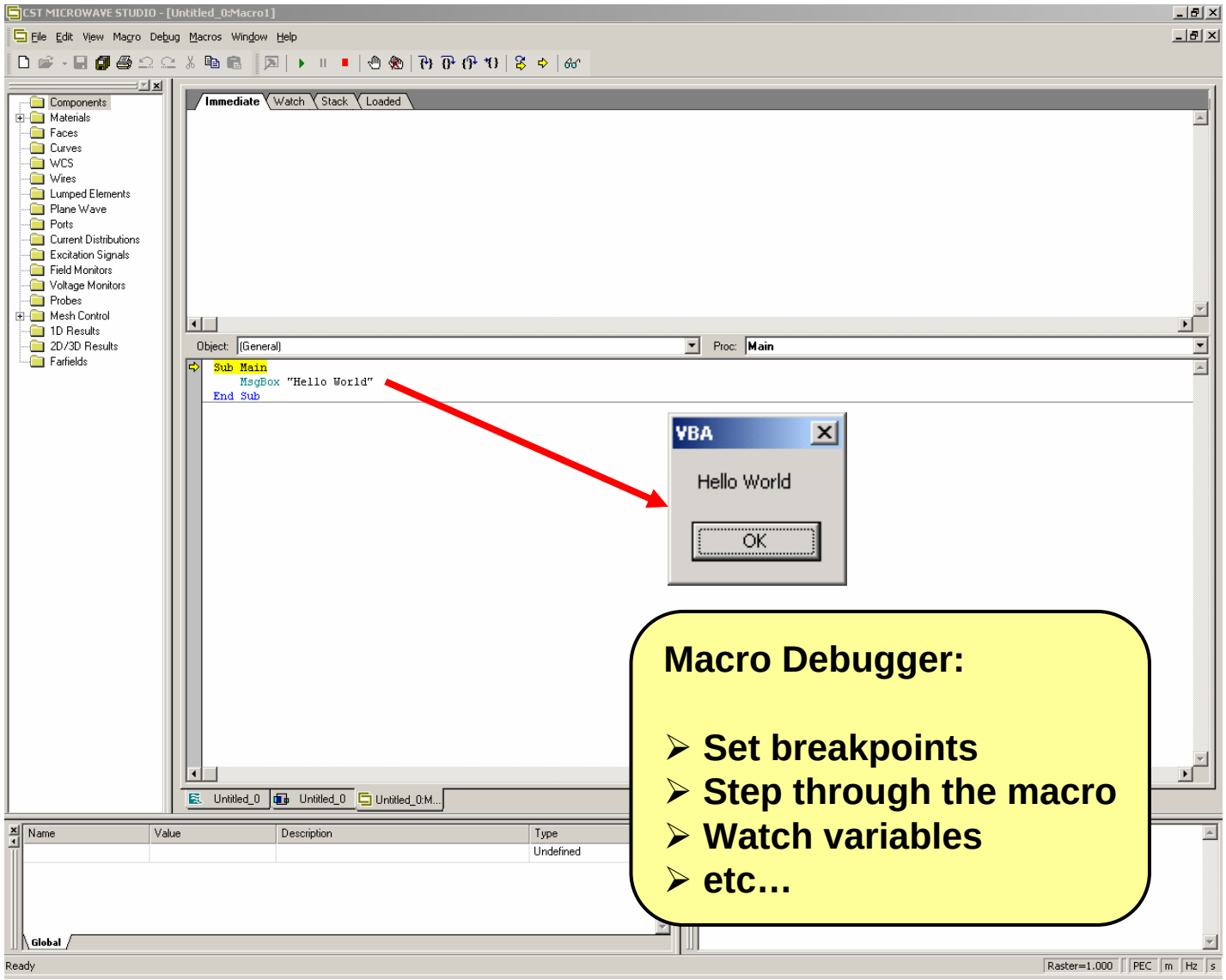
The Integrated Development Environment (IDE)



The Integrated Development Environment (IDE)



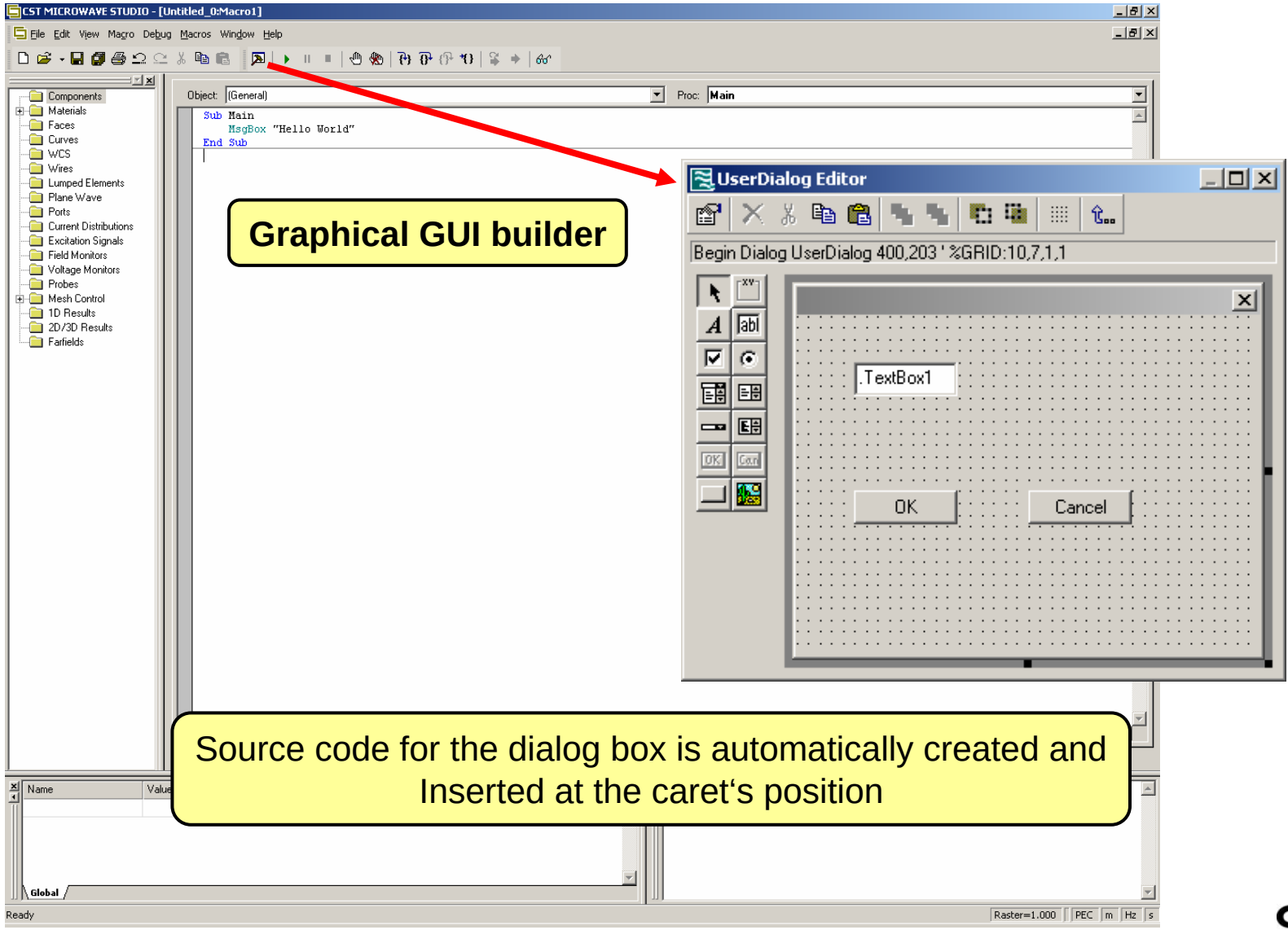
The Integrated Development Environment (IDE)



Macro Debugger:

- **Set breakpoints**
- **Step through the macro**
- **Watch variables**
- **etc…**

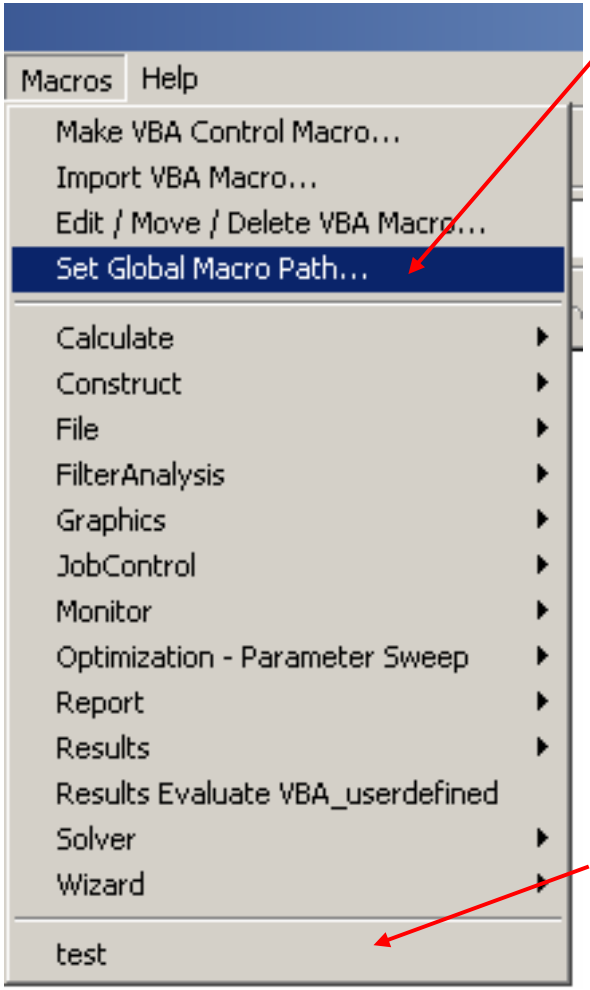
The Integrated Development Environment (IDE)



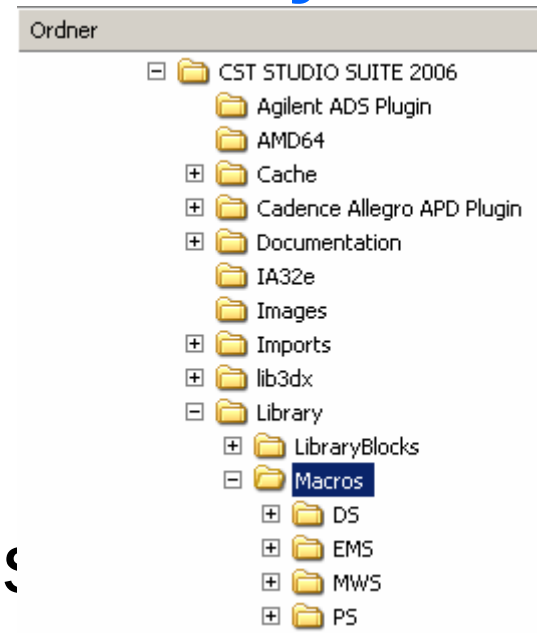
Outline

- Why macro programming?
- Existing macros
- Different types of macros
- Creating and testing new macros
  - The integrated development environment (IDE)
  - Structure of an CST MWS macro
  - How to create a macro?
- Getting more information

Where macro files are stored



- Global macros:
  stored in the Global macro path
  (**`INST_DIR\Library\Macros`**)



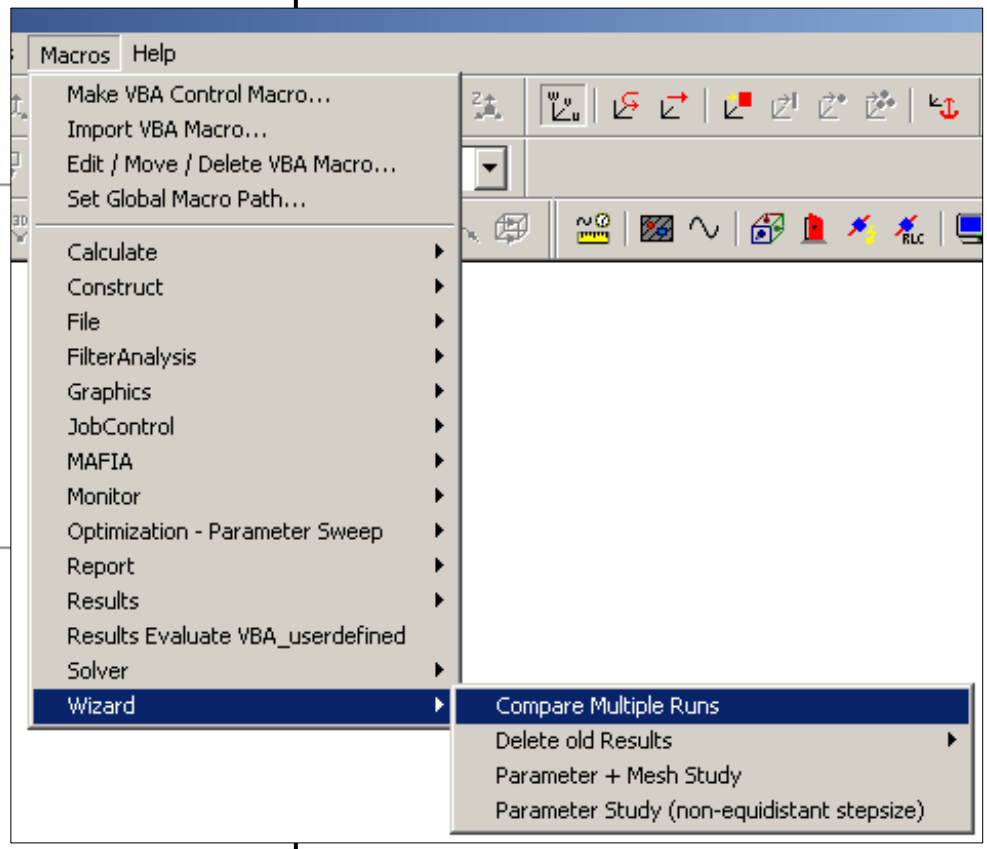
- Local macros
  stored in the same path as the project

Structure of a CST STUDIO SUITE™ Macro

```
' *Wizard / Compare Multiple Runs
' !!! Do not change the line above !!!
' macro.550
'
' macro allows postprocessing of result-cache or other models

Option Explicit

'#include "vba_globals.lib"
'#include "mws_evaluate-results.lib"

Public datafile models As String


Sub Main ()

    SpecifyModelFiles

    Dim sMacrobase As String

    . . .

End Sub


Sub SpecifyModelFiles ()

    If GetApplicationName = "MWS" Then
        sFileExt = ".mod"
    End If

    . . .
End Sub
```

Macro name

Slash is used to define a place in the macro hierarchy



Structure of a CST STUDIO SUITE™ Macro

```
' *Wizard / Compare Multiple Runs
' !!! Do not change the line above !!!
' macro.550
'
' macro allows postprocessing of result-cache or other models

Option Explicit

'#include "vba_globals.lib"
'#include "mws_evaluate-results.lib"

Public datafile models As String


Sub Main ()

    SpecifyModelFiles

    Dim sMacrobase As String

    . . .

End Sub


Sub SpecifyModelFiles ()

    If GetApplicationName = "MWS" Then
        sFileExt = ".mod"
    End If

    . . .
End Sub
```

Comments – start with '

Structure of a CST STUDIO SUITE™ Macro

```
' *Wizard / Compare Multiple Runs
' !!! Do not change the line above !!!
' macro.550
'
' macro allows postprocessing of result-cache or other models

Option Explicit

'#include "vba_globals.lib"
'#include "mws_evaluate-results.lib"

Public datafile models As String


Sub Main ()

    SpecifyModelFiles

    Dim sMacrobase As String

    . . .

End Sub


Sub SpecifyModelFiles ()

    If GetApplicationName = "MWS" Then
        sFileExt = ".mod"
    End If

    . . .
End Sub
```

Options:

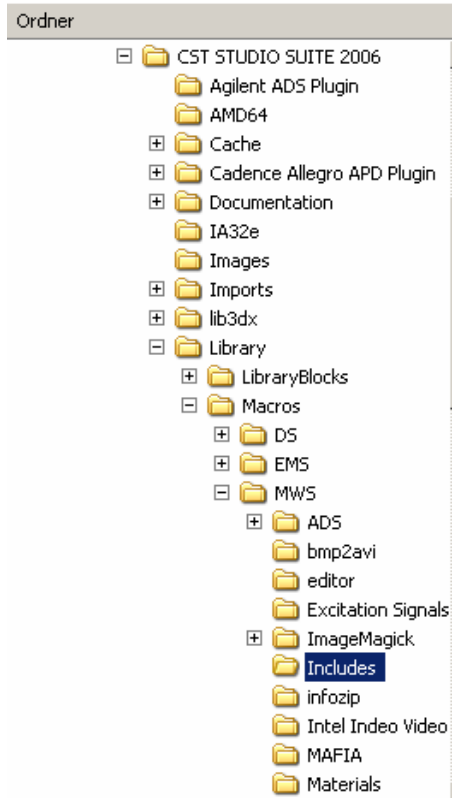
Option Explicit – Force declaration of all variables

Option Private Module – Public variables invisible from outside the project

Structure of a CST STUDIO SUITE™ Macro

```
' *Wizard / Compare Multiple Runs
' !!! Do not change the line above !!!
' macro.550
'
' macro allows postprocessing of result-cache or other models

Option Explicit

'#include "vba_globals.lib"
'#include "mws_evaluate-results.lib"

Public datafile models As String


Sub Main ()

    SpecifyModelFiles

    Dim sMacrobase As String

    . . .

End Sub


Sub SpecifyModelFiles ()

    If GetApplicationName = "MWS" Then
        sFileExt = ".mod"
    End If

    . . .
End Sub
```

Included libraries



Structure of a CST STUDIO SUITE™ Macro

```
' *Wizard / Compare Multiple Runs
' !!! Do not change the line above !!!
' macro.550
'
' macro allows postprocessing of result-cache or other models

Option Explicit

'#include "vba_globals.lib"
'#include "mws_evaluate-results.lib"

Public datafile models As String


Sub Main ()

    SpecifyModelFiles

    Dim sMacrobase As String

    . . .

End Sub


Sub SpecifyModelFiles ()

    If GetApplicationName = "MWS" Then
        sFileExt = ".mod"
    End If

    . . .
End Sub
```

Public variables

Structure of a CST STUDIO SUITE™ Macro

```
' *Wizard / Compare Multiple Runs
' !!! Do not change the line above !!!
' macro.550
'
' macro allows postprocessing of result-cache or other models

Option Explicit

'#include "vba_globals.lib"
'#include "mws_evaluate-results.lib"

Public datafile models As String


Sub Main ()

    SpecifyModelFiles

    Dim sMacrobase As String

    . . .

End Sub


Sub SpecifyModelFiles ()

    If GetApplicationName = "MWS" Then
        sFileExt = ".mod"
    End If

    . . .
End Sub
```

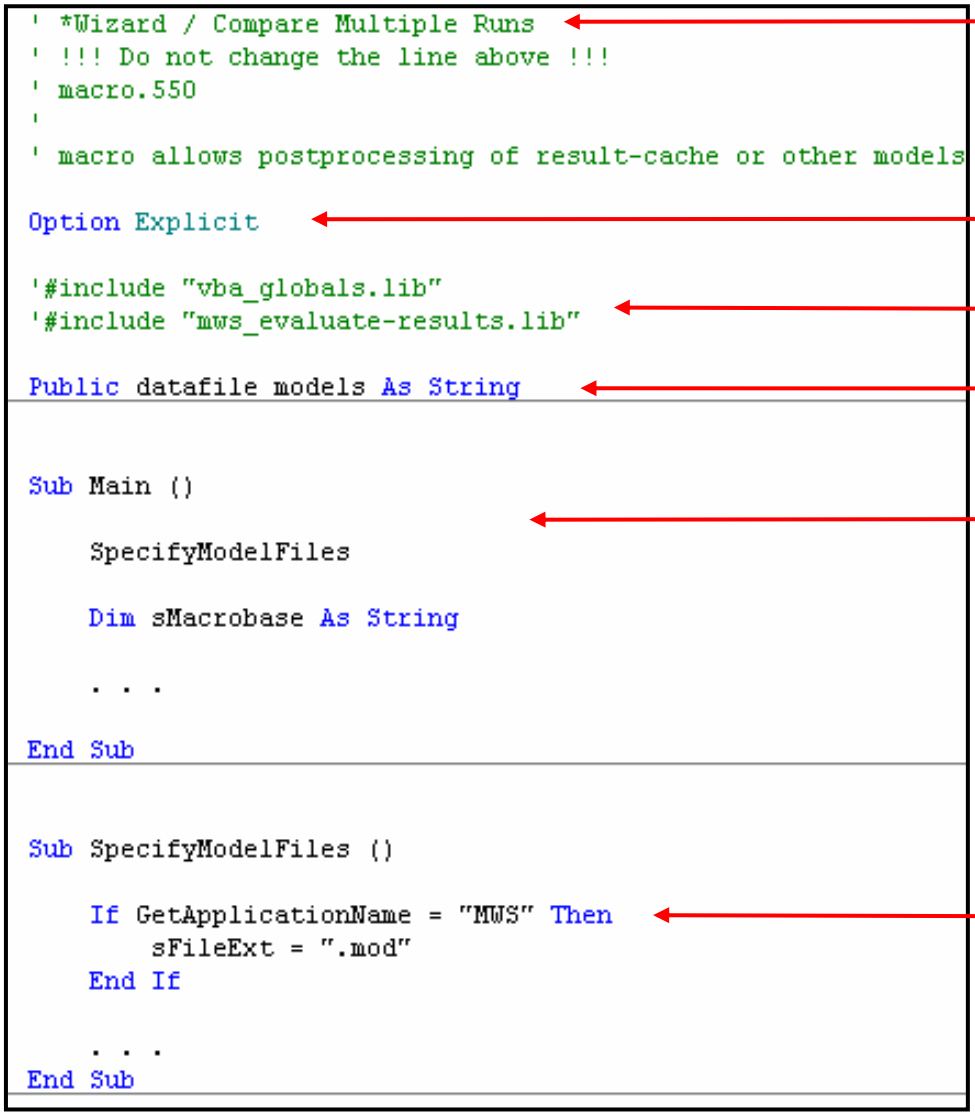
Every macro contains at least the subroutine „Main“ exept Result Templates!

Structure of a CST STUDIO SUITE™ Macro

```
' *Wizard / Compare Multiple Runs
' !!! Do not change the line above !!!
' macro.550
'
' macro allows postprocessing of result-cache or other models

Option Explicit

'#include "vba_globals.lib"
'#include "mws_evaluate-results.lib"

Public datafile models As String


Sub Main ()

    SpecifyModelFiles

    Dim sMacrobase As String

    . . .

End Sub


Sub SpecifyModelFiles ()

    If GetApplicationName = "MWS" Then
        sFileExt = ".mod"
    End If

    . . .
End Sub
```

Other needed functions and subroutines

Structure of a CST STUDIO SUITE™ Macro



```
' *Wizard / Compare Multiple Runs
' !!! Do not change the line above !!!
' macro.550
'
' macro allows postprocessing of result-cache or other models

Option Explicit

'#include "vba_globals.lib"
'#include "mws_evaluate-results.lib"

Public datafile models As String


Sub Main ()

    SpecifyModelFiles

    Dim sMacrobase As String

    . . .

End Sub


Sub SpecifyModelFiles ()

    If GetApplicationName = "MWS" Then
        sFileExt = ".mod"
    End If

    . . .
End Sub
```

Macro name

Comments – start with '

Options

Included libraries

Public variables

Every macro contains at least the subroutine „Main“ exept Result Templates!

Other needed functions and subroutines

Structure of a Result Template

```
Option Explicit

'#include "vba_globals.lib"

Function Define(sName As String, bCreate As Boolean, bNameChanged As Boolean) As Boolean

    Define = True

    Begin Dialog UserDialog 400,203 ' %GRID:10,7,1,1
        Text 40,63,90,14,"Number:",.Text3
        TextBox 160,63,90,21,.nant

        ...

    End Dialog
    Dim dlg As UserDialog

    dlg.nant = GetScriptSetting("NumberOfAntennas","1")

    ...

    StoreScriptSetting("NumberOfAntennas",dlg.nant)

End Function
'------------------------------------------------------------------------------------

Function EvaluateID() As Object

    Set EvaluateID = ResultID("")

    SelectTreeItem "Farfields\farfield (f=7.55) [1]"

    Dim nant As Integer

    nant       = CInt(GetScriptSetting("NumberOfAntennas","1"))

    ...

End Function
```

Defines all necessary input data

Structure of a Result Template

```
Option Explicit

'#include "vba_globals.lib"

Function Define(sName As String, bCreate As Boolean, bNameChanged As Boolean) As Boolean

    Define = True

    Begin Dialog UserDialog 400,203 ' %GRID:10,7,1,1
        Text 40,63,90,14,"Number:",.Text3
        TextBox 160,63,90,21,.nant

        ...

    End Dialog
    Dim dlg As UserDialog

    dlg.nant = GetScriptSetting("NumberOfAntennas","1")

    ...

    StoreScriptSetting("NumberOfAntennas",dlg.nant)

End Function
'------------------------------------------------------------------------------------

Function EvaluateID() As Object

    Set EvaluateID = ResultID("")

    SelectTreeItem "Farfields\farfield (f=7.55) [1]"

    Dim nant As Integer

    nant       = CInt(GetScriptSetting("NumberOfAntennas","1"))

    ...

End Function
```

Dialog built by GUI builder

Structure of a Result Template

```
Option Explicit

'#include "vba_globals.lib"

Function Define(sName As String, bCreate As Boolean, bNameChanged As Boolean) As Boolean

    Define = True

    Begin Dialog UserDialog 400,203 ' %GRID:10,7,1,1
        Text 40,63,90,14,"Number:",.Text3
        TextBox 160,63,90,21,.nant

        ...

    End Dialog
    Dim dlg As UserDialog

    dlg.nant = GetScriptSetting("NumberOfAntennas","1")

    ...

    StoreScriptSetting("NumberOfAntennas",dlg.nant)

End Function
'------------------------------------------------------------------------------------

Function EvaluateID() As Object

    Set EvaluateID = ResultID("")

    SelectTreeItem "Farfields\farfield (f=7.55) [1]"

    Dim nant As Integer

    nant       = CInt(GetScriptSetting("NumberOfAntennas","1"))

    ...

End Function
```

Gets default settings/ already stored settings

Structure of a Result Template

```
Option Explicit

'#include "vba_globals.lib"

Function Define(sName As String, bCreate As Boolean, bNameChanged As Boolean) As Boolean

    Define = True

    Begin Dialog UserDialog 400,203 ' %GRID:10,7,1,1
        Text 40,63,90,14,"Number:",.Text3
        TextBox 160,63,90,21,.nant

        ...

    End Dialog
    Dim dlg As UserDialog

    dlg.nant = GetScriptSetting("NumberOfAntennas","1")

    ...

    StoreScriptSetting("NumberOfAntennas",dlg.nant)

End Function
'------------------------------------------------------------------------------------------

Function EvaluateID() As Object

    Set EvaluateID = ResultID("")

    SelectTreeItem "Farfields\farfield (f=7.55) [1]"

    Dim nant As Integer

    nant       = CInt(GetScriptSetting("NumberOfAntennas","1"))

    ...

End Function
```

Stores Settings in common script

Structure of a Result Template

```
Option Explicit

'#include "vba_globals.lib"

Function Define(sName As String, bCreate As Boolean, bNameChanged As Boolean) As Boolean

    Define = True

    Begin Dialog UserDialog 400,203 ' %GRID:10,7,1,1
        Text 40,63,90,14,"Number:",.Text3
        TextBox 160,63,90,21,.nant

        ...

    End Dialog
    Dim dlg As UserDialog

    dlg.nant = GetScriptSetting("NumberOfAntennas","1")

    ...

    StoreScriptSetting("NumberOfAntennas",dlg.nant)

End Function
'------------------------------------------------------------------------------------

Function EvaluateID() As Object

    Set EvaluateID = ResultID("")

    SelectTreeItem "Farfields\farfield (f=7.55) [1]"

    Dim nant As Integer

    nant       = CInt(GetScriptSetting("NumberOfAntennas","1"))

    ...

End Function
```

Mandatory function for a 1D Template

Outline

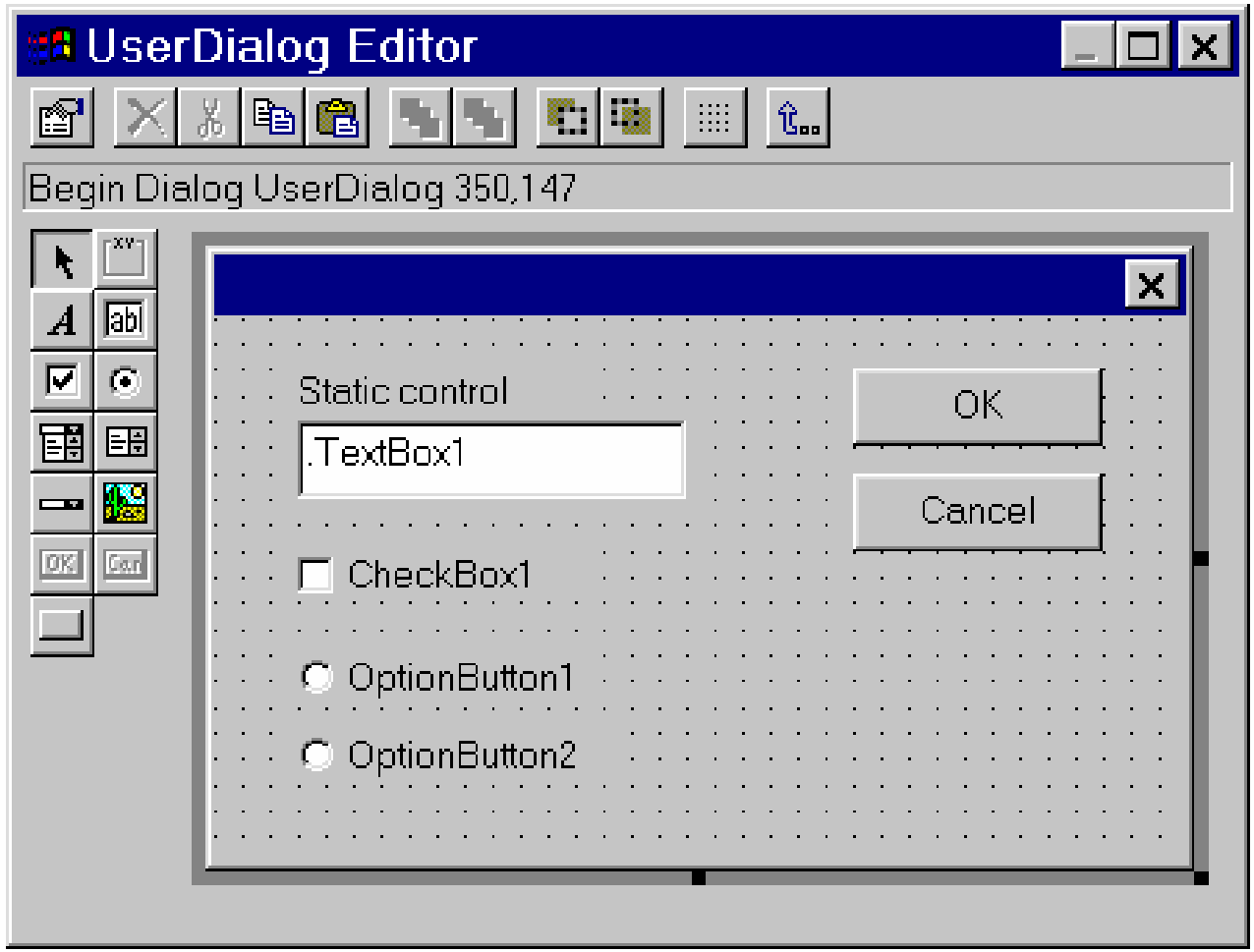
- Why macro programming?
- Existing macros
- Different types of macros
- Creating and testing new macros
  - The integrated development environment (IDE)
  - Structure of an CST MWS macro
  - How to create a macro?
- Getting more information

How to Create Macros?

There exist different ways to create a macro

- Copy and modify an existing macro
  - Project Templates
  - Result Templates
  - Preloaded macro examples
- Go to the history list, select lines and press „*Macro*“
- Use *Macro* ⇨ *Make VBA Control Macro*
- Let CST MWS create the macro‘s framework by pressing „Edit“ for
  - User defined excitation function
  - User defined parameter sweep watch
  - User defined optimizer goal function

Integrated development environment GUI-builder

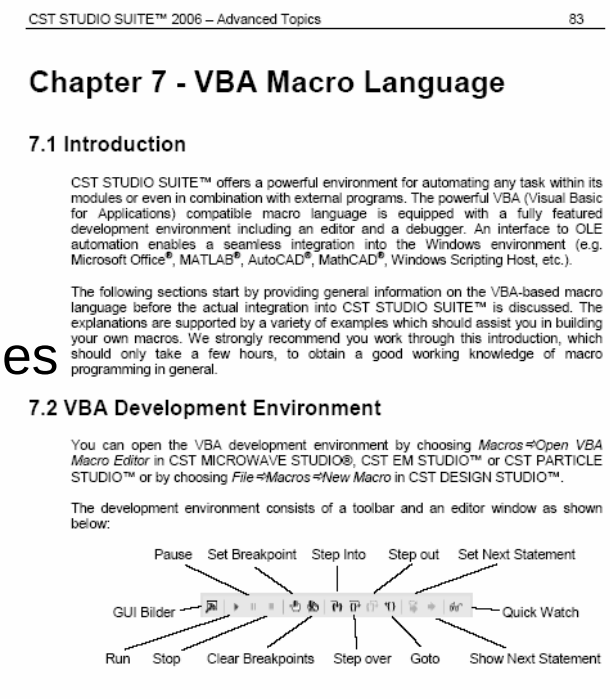


Outline

- Why macro programming?
- Existing macros
- Different types of macros
- Creating and testing new macros
  - The integrated development environment (IDE)
  - Structure of an CST MWS macro
  - How to create a macro?
- Getting more information

Getting More Information

- Carefully read the *Advanced Topics* Manual
- Check the VBA online manual
  (*Help ⇨VBA Macro Language*)
- Reference VBA programming from text
- Have a look at the pre-loaded macro examples
- Visit a special training class on macro programming
- Learning by doing….



CST STUDIO SUITE™ 2006 – Advanced Topics 83

Chapter 7 - VBA Macro Language

7.1 Introduction

CST STUDIO SUITE™ offers a powerful environment for automating any task within its modules or even in combination with external programs. The powerful VBA (Visual Basic for Applications) compatible macro language is equipped with a fully featured development environment including an editor and a debugger. An interface to OLE automation enables a seamless integration into the Windows environment (e.g. Microsoft Office®, MATLAB®, AutoCAD®, MathCAD®, Windows Scripting Host, etc.).

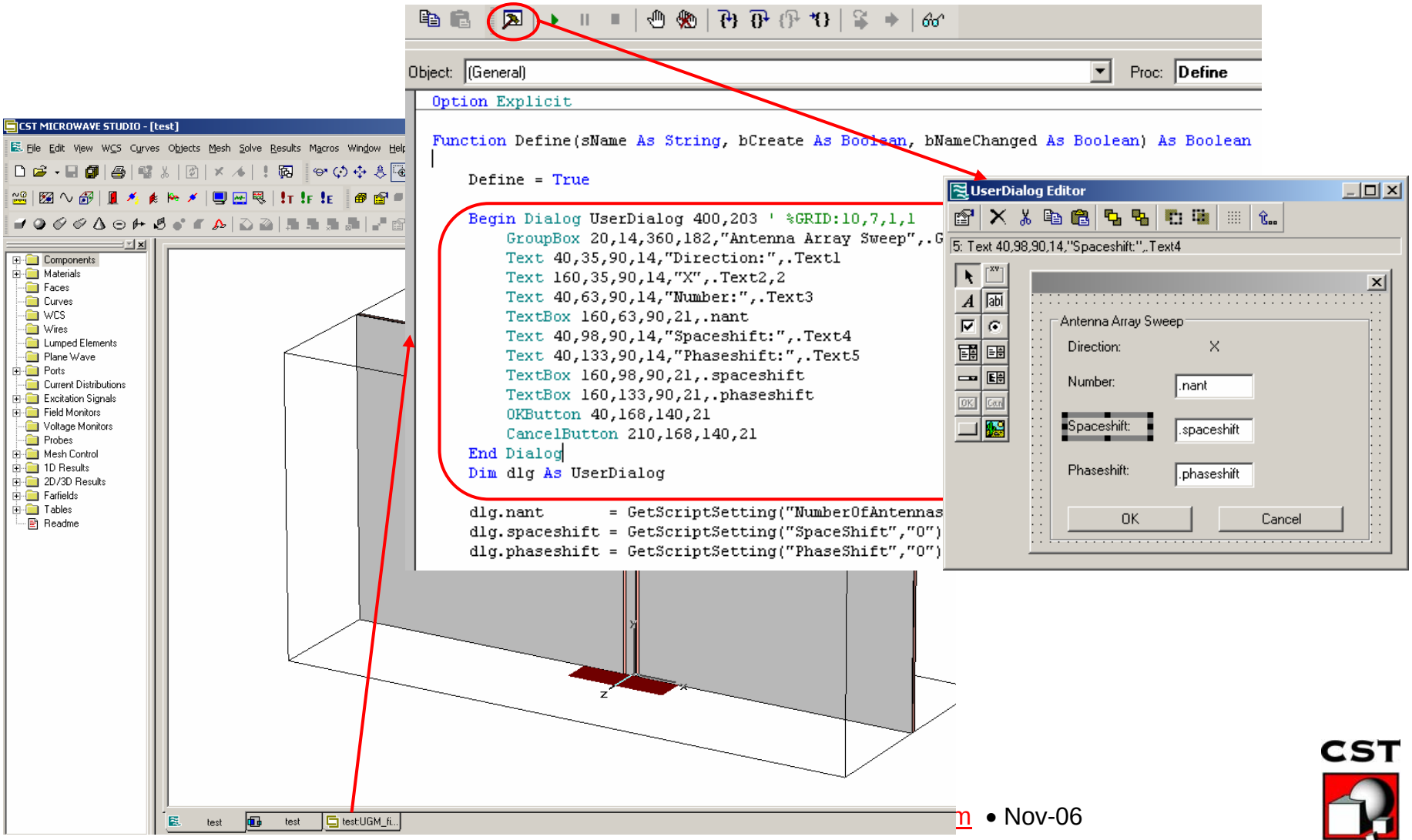
The following sections start by providing general information on the VBA-based macro language before the actual integration into CST STUDIO SUITE™ is discussed. The explanations are supported by a variety of examples which should assist you in building your own macros. We strongly recommend you work through this introduction, which should only take a few hours, to obtain a good working knowledge of macro programming in general.

7.2 VBA Development Environment

You can open the VBA development environment by choosing *Macros⇨Open VBA Macro Editor* in CST MICROWAVE STUDIO®, CST EM STUDIO™ or CST PARTICLE STUDIO™ or by choosing *File⇨Macros⇨New Macro* in CST DESIGN STUDIO™.

The development environment consists of a toolbar and an editor window as shown below:

Example: Farfield Plot Sweep



Example: Farfield Plot Sweep

```
Function Define(sName As String, bCreate As Boolean, bNameChanged As Boolean) As Boolean

        Define = True

        Begin Dialog UserDialog 400,203 ' %GRID:10,7,1,1
                        GroupBox 20,14,360,182,"Antenna Array Sweep",.GroupBox1
                        Text 40,35,90,14,"Direction:",.Text1
                        Text 160,35,90,14,"X",.Text2,2
                        Text 40,63,90,14,"Number:",.Text3
                        TextBox 160,63,90,21,.nant
                        Text 40,98,90,14,"Spaceshift:",.Text4
                        Text 40,133,90,14,"Phaseshift:",.Text5
                        TextBox 160,98,90,21,.spaceshift
                        TextBox 160,133,90,21,.phaseshift
                        OKButton 40,168,140,21
                        CancelButton 210,168,140,21
        End Dialog
        Dim dlg As UserDialog
```

User dialog editor

```
        dlg.nant      = GetScriptSetting("NumberOfAntennas","1")
        dlg.spaceshift = GetScriptSetting("SpaceShift","0")
        dlg.phaseshift = GetScriptSetting("PhaseShift","0")
```

Default values

```
        If (Not Dialog(dlg)) Then

                        ' The user left the dialog box without pressing Ok.
                        Define = False
        Else

                        ' The user properly left the dialog box by pressing Ok.
                        Define = True

                        ' Store the script settings into the database for later reuse by either the define function (for modifications)
                        ' or the evaluate function.

                        StoreScriptSetting("NumberOfAntennas",dlg.nant)
                        StoreScriptSetting("SpaceShift",dlg.spaceshift)
                        StoreScriptSetting("PhaseShift",dlg.phaseshift)
        End If

End Function
```

Store Settings

CST

```
Function Evaluate1D() As Object
    Set Evaluate1D = Result1D("")
    Dim cst_value As Double
    Open "Gain_vs_Number_of_Antennas.sig" For Output As #1
    SelectTreeItem "Farfields\farfield (f=2.92) [1]"

    Dim nant As Integer
    Dim spaceshift As Double, phaseshift As Double
    nant       = CInt(GetScriptSetting("NumberOfAntennas","1"))
    spaceshift = CDbl(GetScriptSetting("SpaceShift","0"))
    phaseshift = CDbl(GetScriptSetting("PhaseShift","0"))

    With FarfieldPlot
            .Plottype "3d"
            .Step "5"
            .UseFarfieldApproximation "True"
            .SetPlotMode  "gain"
    End With

    Dim I As Integer
    For I=1 To nant STEP 1
            With FarfieldArray
                    .Reset
                    .UseArray (True)
                    .Arraytype ("rectangular")
                    .XSet (I,spaceshift,phaseshift)
                    .SetList
            End With

            FarfieldPlot.plot
            Plot.Update

            cst_value = FarfieldPlot.GetMax

            Print #1, CStr(I) + "      " + CStr(cst_value)
            Evaluate1D.AppendXY I, cst_value

    Next I
    Evaluate1D.Xlabel "Number of Antennas"
End Function
```

Needs to be adapted, here fixed

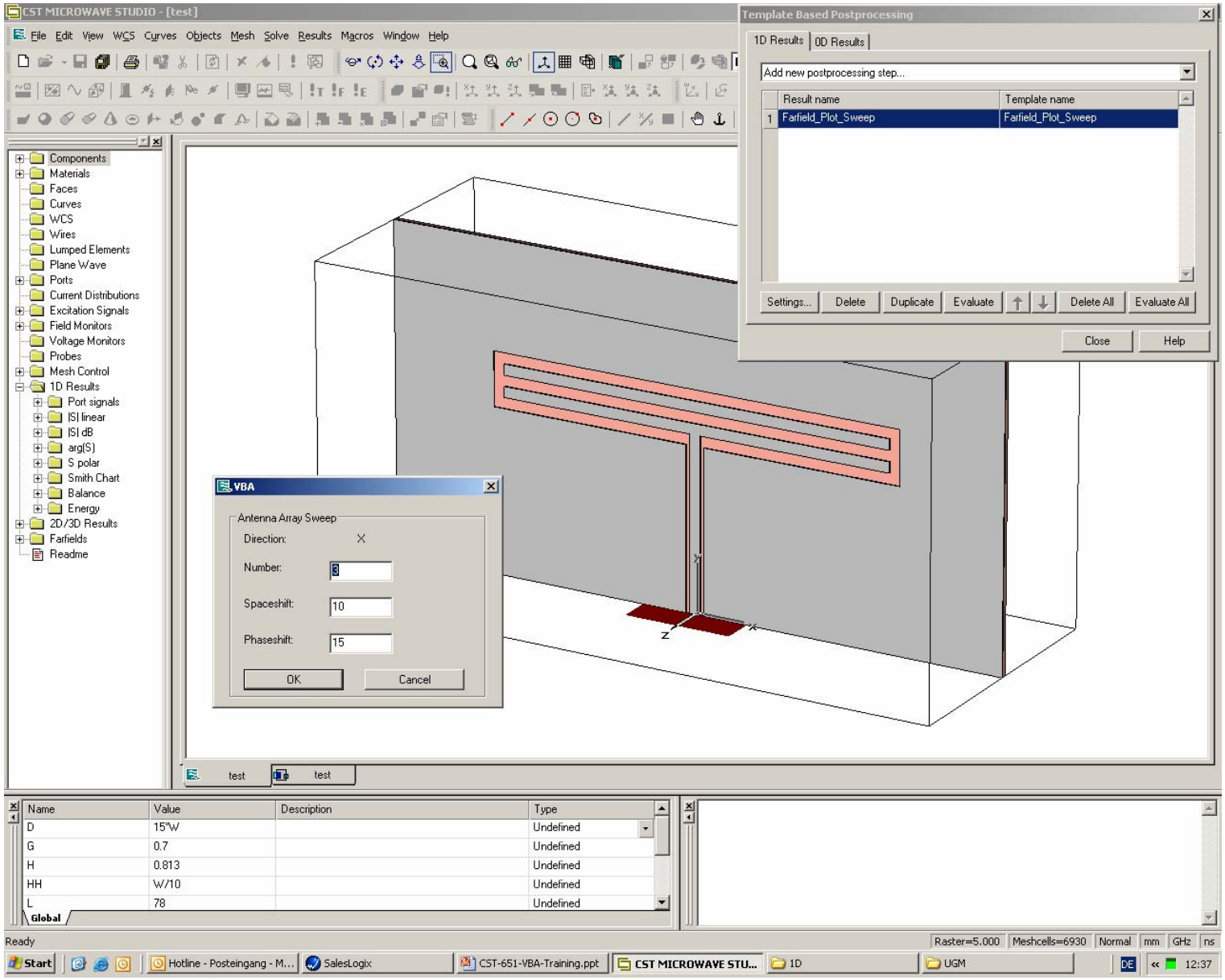
Read previously defined settings

Select plot to be evaluated

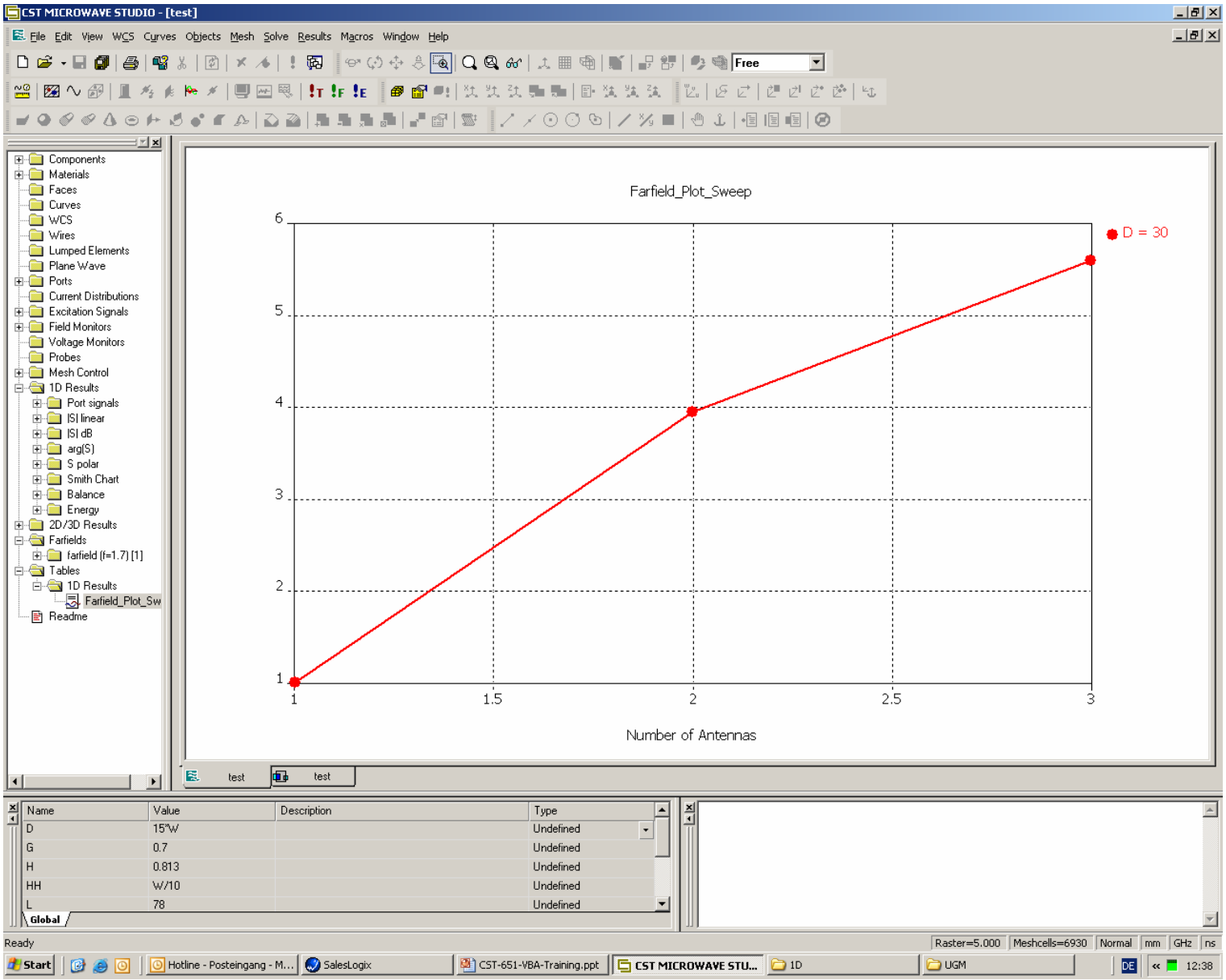
Change array settings and update

Evaluate value of interest and add to 1D results

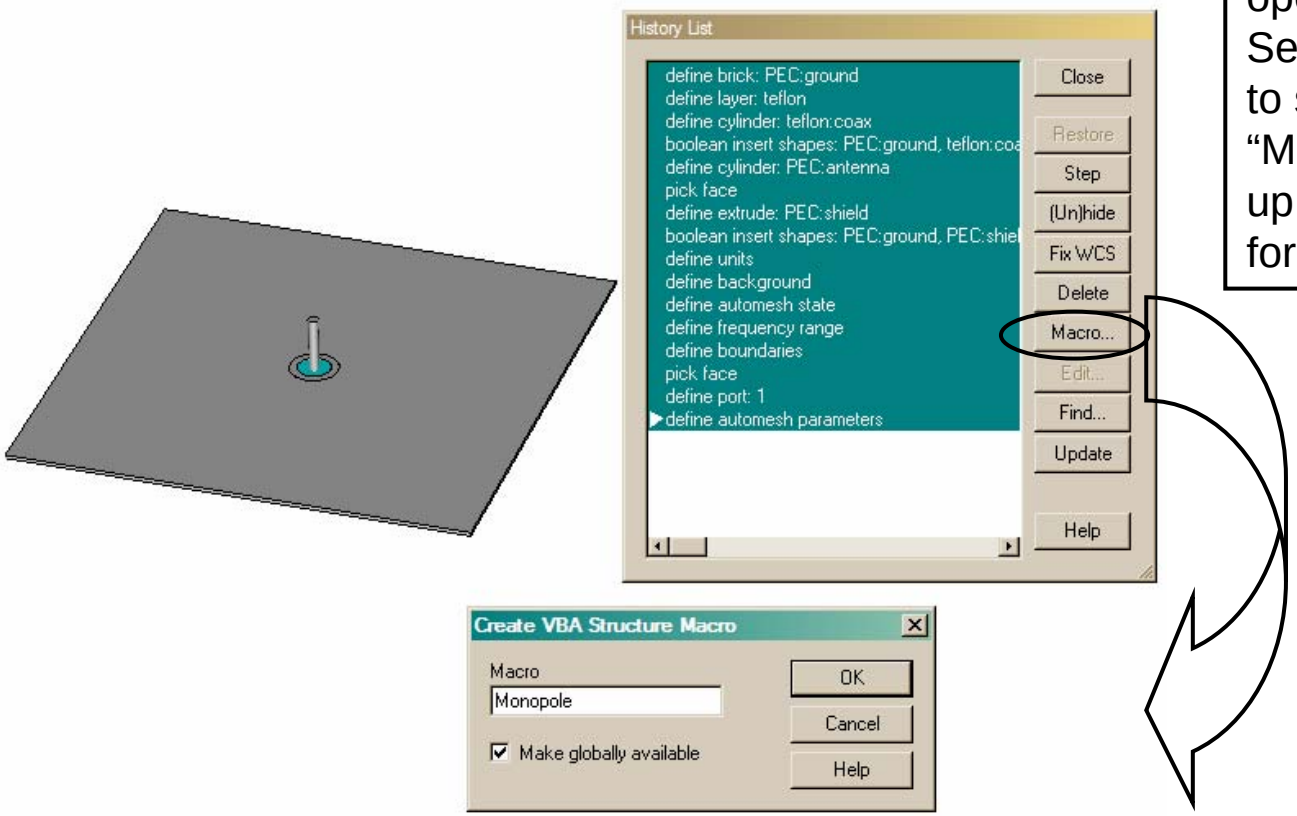
Example: Farfield Plot Sweep



Example: Farfield Plot Sweep



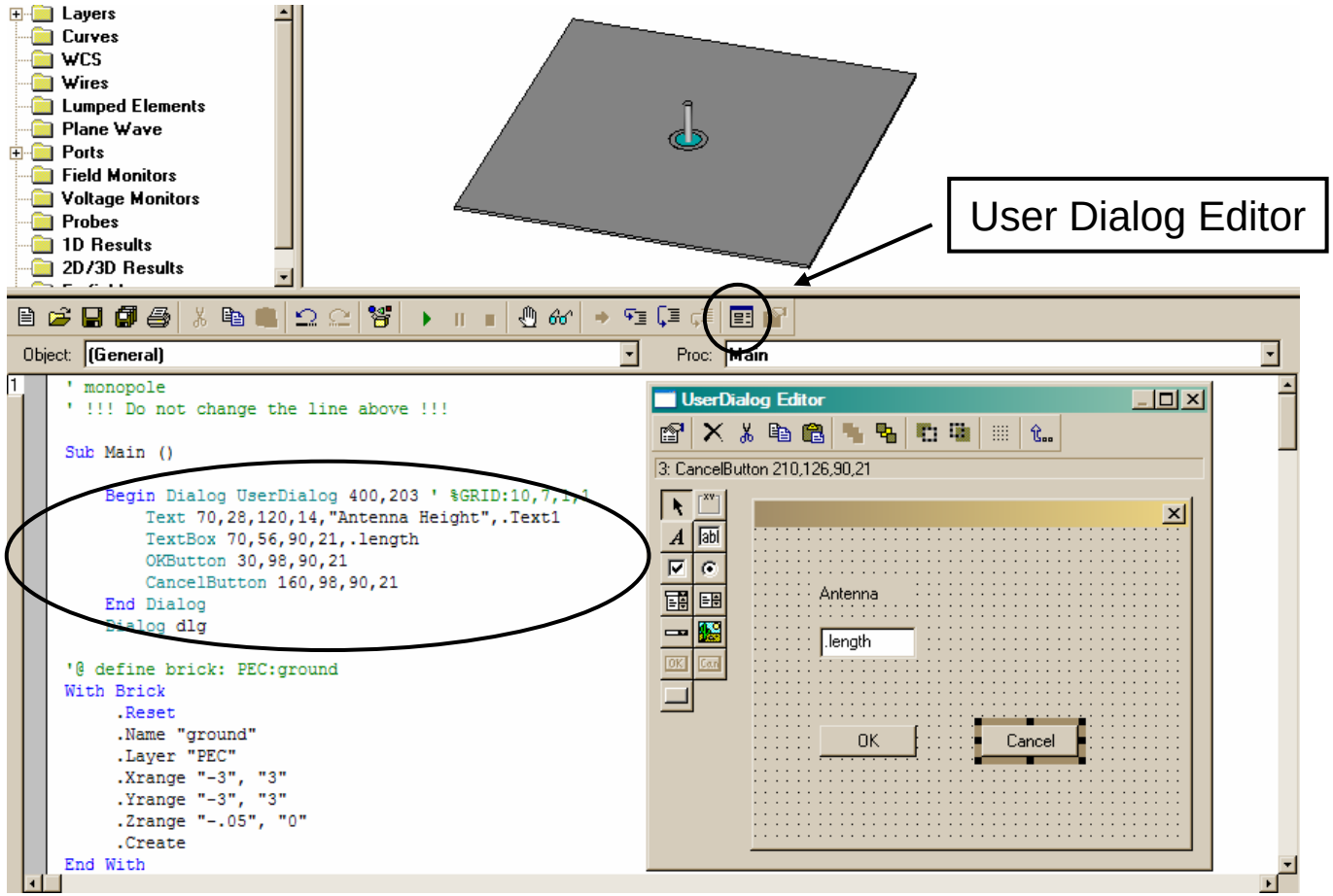
Example – Monopole Antenna



After completing a model, open the History List. Select the steps you wish to save as a macro, select "Macro," which will bring up an additional window for naming the macro.

Example – Monopole Antenna

After saving, macro will be visible from editor. Select the User Dialog Editor to create macro language for dialog box.



Example – Monopole Antenna

```
Sub Main ()

        Begin Dialog UserDialog 400,203 ' %GRID:10,7,1,1
                Text 70,28,120,14,"Antenna Height",.Text1
                TextBox 70,56,90,21,.length
                OKButton 30,98,90,21
                CancelButton 160,98,90,21
        End Dialog

        Dim dlg As UserDialog
        BeginHide
        Dialog dlg
        Assign "dlg.length"
        EndHide

....

'@ define cylinder: PEC:antenna
With Cylinder
    .Reset
    .Name "antenna"
    .Layer "PEC"
    .OuterRadius ".0635"
    .InnerRadius "0"
    .Axis "z"
    .Zrange "-.5", Evaluate(dlg.length)
```

3 Things to remember…

BeginHide / EndHide

Assign “dlg.length”

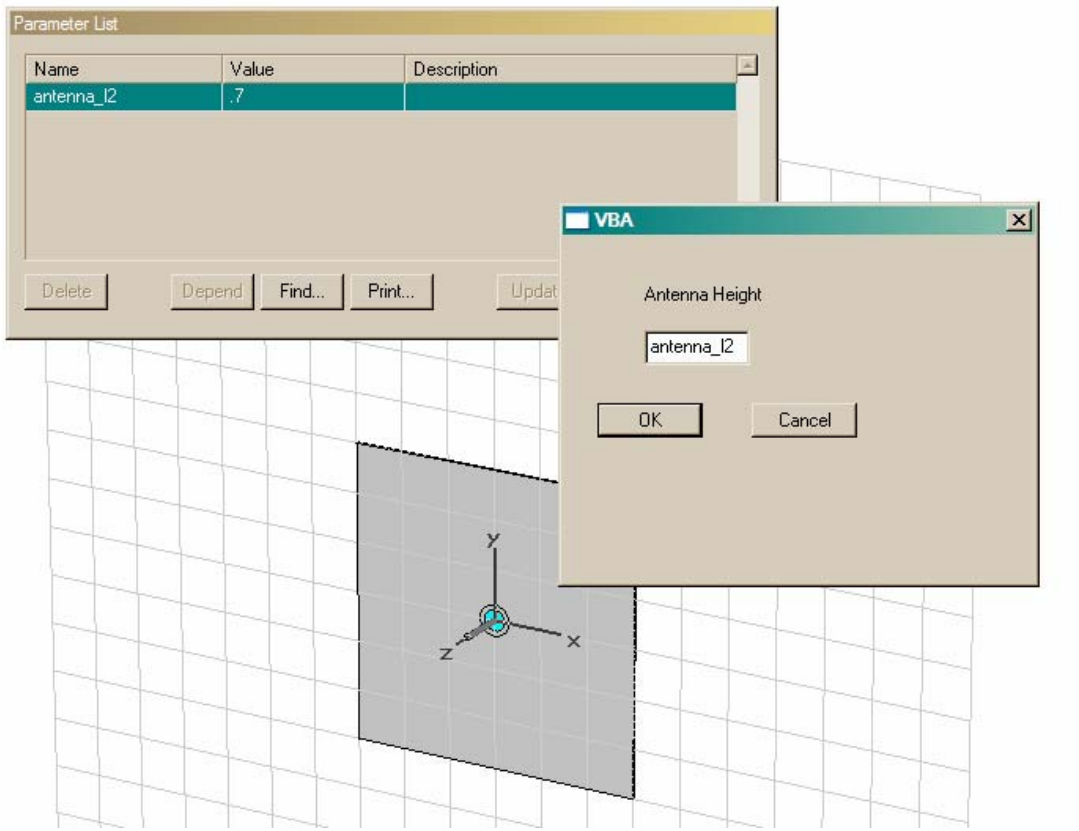
Evaluate(dlg.length)

“Hide” the region not to be written in the history list, in this case, the execution of the dialog box.

“Assign” the variable from the dialog box for the rest of the macro to use.

“Evaluate” this variable in the parameter of interests.

Example – Monopole Antenna



When the macro prompts for the value, a parameter can be used for parametrics and optimization.

Summary

- Automate common tasks to increase productivity
- Extend the program‘s capabilities
- Integrated Development Environment available
- Structure macros and command macros
- Project macros and global macros
- Copy and modify existing macros
- Let CST MWS create the macro‘s framework
- Refer to the *Advanced Topics* for more information