

# Scripting Development Environment

AWR's Script Development Environment (SDE) allows you to create and execute macros and utilities to automate complex tasks from within the AWRDE. The SDE supports scripting in Visual Basic and includes syntax highlighting and powerful debugging capabilities.

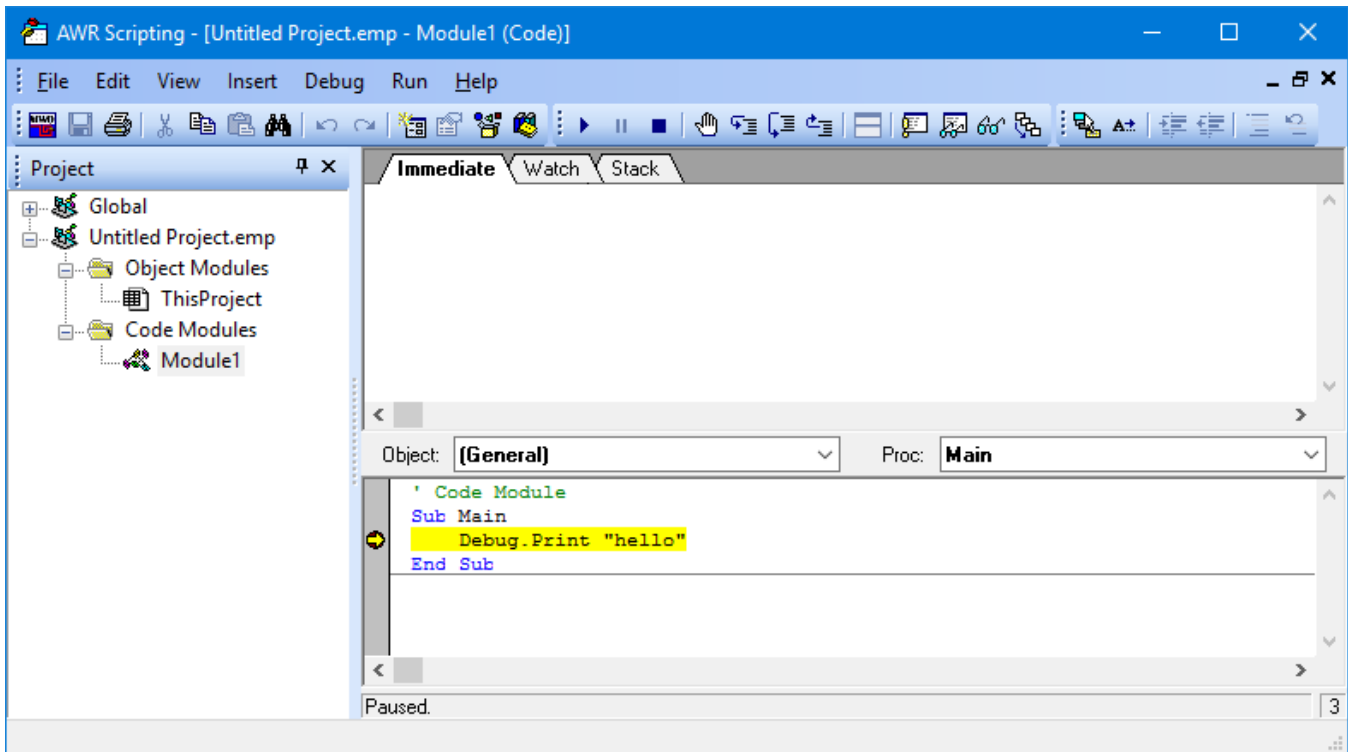
The Scripting Editor contains BASIC script modules that you can run from the program by choosing **Scripts > Global Scripts** or **Scripts > Project Scripts** directly, or from within the SDE.

The language used in scripts is SAX Basic, a subset of Microsoft® Visual Basic® for Applications. SAX Basic language online Help is available here: [AWR\\_V13\\_API\\_Diagram.pdf](#)

- [Components of the Script Development Environment](#)
- [Project Browser](#)
  - [Local Modules](#)
  - [Global Modules](#)
  - [Speed Menus](#)
- [Developing Scripts](#)
  - [References](#)
  - [Object Browser](#)
  - [Object and Proc Lists](#)
  - [Status Bar](#)
  - [Auto-Complete](#)
  - [Referencing Subroutines and Functions in Other Files](#)
    - [Stand-Alone File on Disk](#)
    - [Another Script Loaded in the SDE](#)
  - [Getting Help](#)
  - [Debugging Scripts](#)
    - [Break Bar](#)
    - [Stepping Through Code](#)
    - [Immediate Window](#)
    - [Watch Window](#)
    - [Stack Window](#)
    - [Debug.Print Statement](#)
  - [User Forms](#)
- [Running Scripts from the AWR Design Environment](#)
- [Menus](#)
  - [Edit Area Speed Menus](#)
  - [File Menu](#)
  - [Edit Menu](#)
  - [View Menu](#)
  - [Insert Menu](#)
  - [Debug Menu](#)
  - [Run Menu](#)
  - [Help Menu](#)

## Components of the Script Development Environment

To open the SDE, choose **Tools > Scripting Editor** or click the Scripting Editor toolbar button. The AWR SDE window opens.



The following table describes the components of the SDE:

Name	Description
Project	Script Project Browser (or simply, Project Browser). Allows you to create scripts to automate tasks within the AWRDE. The scripts display as two subnodes. The first subnode, <b>Global</b> , contains global modules. The second subnode, Untitled Project.emp contains project-specific modules also called local modules.
Menus	A set of menus: File, Edit, View, Insert, Debug, Run, and Help. Most of the menu choices and commands are also available as buttons on the toolbar.
Script workspace	The area in which you design, run, and debug scripts.
Immediate /Watch/Stack window	The area in which you can print the results of a running script and watch the value of the variables during the debugging process.
Toolbars (Standard, Debug, and Edit)	A row of buttons that provides shortcuts for editing, running, and debugging scripts; creating new Code Modules; accessing VBA Object Browser Help; adding available references to a Project; and for accessing Immediate, Watch and Quick Watch Windows. To view a tooltip for a particular toolbar command, pass the cursor over the button.

## Project Browser

The scripting browser is along the left side of the SDE and organizes the scripts available in the project. Scripts can be either global or local to a project. If a script is general purpose, you should save it as a global script. If the script only works with a specific project, you should save it as a local script.

### Local Modules

Shown in the scripting browser with the name of the project, in this example "Untitled Project.emp." Local scripts are saved in the project, so if you copy your project file to another machine, your scripts are automatically included.

### Global Modules

Scripts stored in the **Scripts** or **ScriptUser** folders are included in this. From the AWR Design Environment, select **Help > Show Files/Directories** to find these folders. **Scripts** folder contains the factory scripts shipped with the product installer. **ScriptUser** contains global scripts that the user manages. Global scripts are available to all projects. The Global subnode contains all global scripts that are stored separately in individual \*.bas files located in specific folders on your computer. You can add a new \*.bas file to your **ScriptUser** folder, then right-click the Global node and choose **Check For New Files** to make the new script available.

To define more than one global script folder, add the following line in your *User.ini* file:

```
[File Locations]
Scripts=$DEFAULT;<path>
```

where <path> is the full or relative path to the folder to search for additional \*.bas files. You can add more than one path by separating paths with semi-colons. PDKs can also specify a scripts file location, so scripts can be shipped with PDKs and loaded automatically as global scripts.

## Speed Menus

Right-click in the scripting browser to get the following menu commands:

Command	Description
View Code	Opens that code module for editing.
Run Sub Main	Run the Main subroutine in that code module.
Insert Module	Insert a new code module.
Rename	Rename the selected module.
Remove	Remove (delete) the selected module.
Export	Save the module to disk as a .bas file.
Print	Print the selected module.
Import	Import a module from disk (only in Local Modules). You will be prompted to browse for a .bas file to import
Check for New Files	Add any new files in the global locations to the SDE (only in Global Modules).

## Developing Scripts

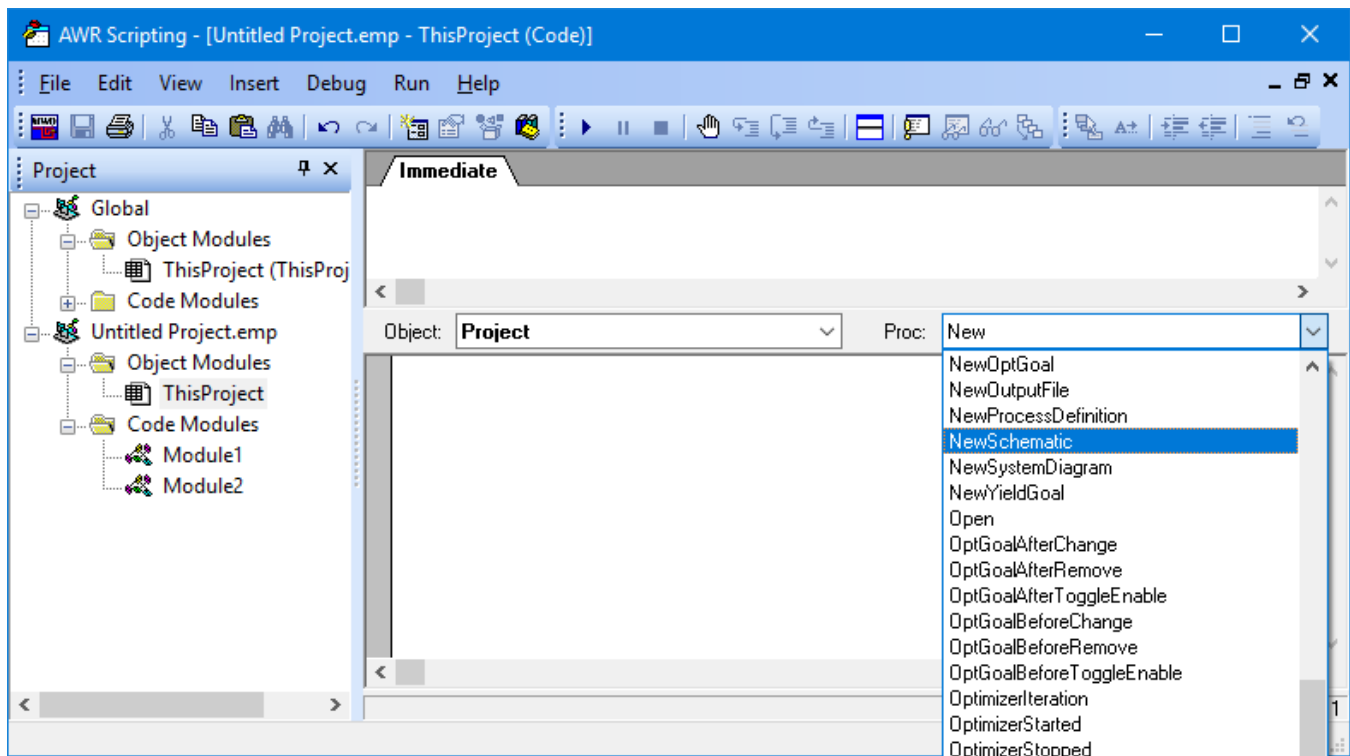
The code can either be in code modules or object modules. Code modules contain procedures that run from start to finish when executed. This is useful for creating macros or procedures for automating tasks that you would normally perform manually. Each module will begin with a 'Sub Main', and each code module is listed the project browser as a separate item. To create a module, right-click the Global or the local node and choose **Insert Module**. A new module is added to the node, and a new window will be opened with the 'Sub Main' defined.

```
' Code Module
Sub Main

End Sub
```

An Object module runs code in response to events generated by the AWRDE object. Object modules do not have a Sub Main that executes when they run. Instead, they contain special functions called Event Handlers that execute in response to events that objects start. When run, the Basic engine waits for an event to occur, such as adding a schematic to the project. When the event occurs, the associated event handler code executes.

To enter the code that responds to an event, expand the project tree for the local scripts, into the **Object Modules** node and double click on the node named ThisProject. **Note:** we do not support global object modules at this time. In the code editor window, select the object that starts the event from the object list (this will be called **Project** for events from the AWR API), and then select an event from the **Proc** list. After you select an event, a handler procedure is added to the module, and you can add any additional code you want to execute. The following figures show this process.



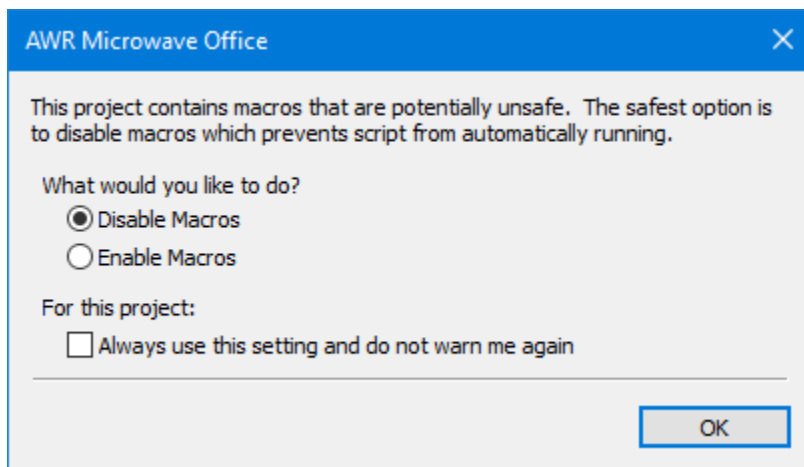
The following code is added to the code editor window.

```
Private Sub Project_NewSchematic(Schematic As MWOffice.Schematic)

End Sub
```

Before the code can respond to events, you must run the module by clicking the **Run** button on the toolbar.

Next time you open the project containing object modules, a dialog box will appear asking if you would like to enable or disable macros.



If you choose **Disable Macros**, the object module will not be running. You can immediately edit the code, and you will need to click **Run** on the toolbar to be able to respond to an event.

If you choose **Enable Macros**, the object module will be running. You will need to click **End** on the toolbar to be able to edit the code, and the AWRDE will be automatically responding to events.

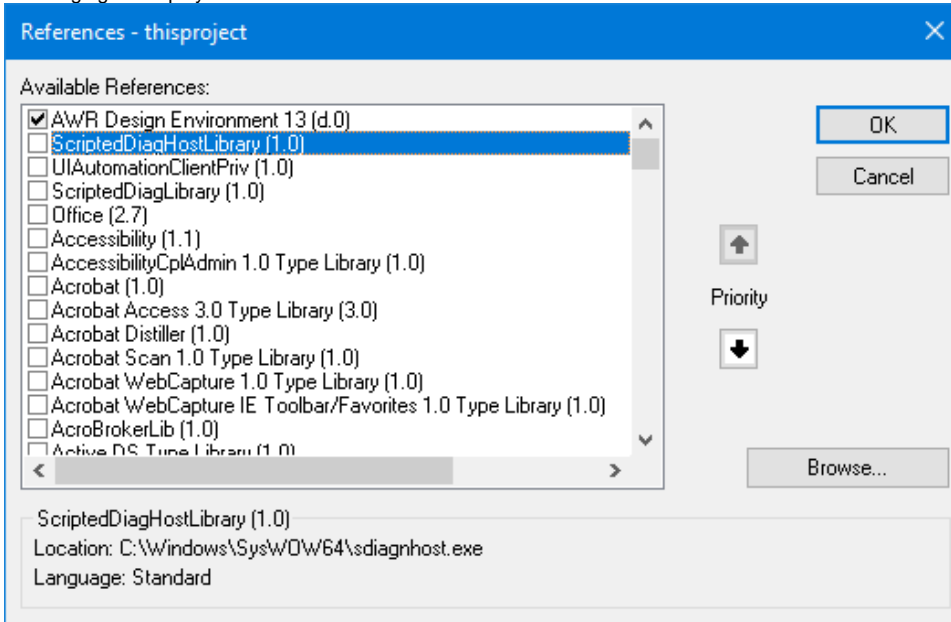
If you set the option, **Always use the settings and do not warn again**, the dialog will not show again for that project. If you need to change the settings and to have the dialog, there will be an entry in the **user.ini** file. Search for the project name in the **user.ini** and remove the entry for the warning about the script.

```
[test_project.emp]  
EnableMacros=-1
```

## References

To work with different automation clients and their objects, you should create a reference to a COM components type library using the following steps in the SDE:

1. Click the References button on the toolbar or select **Edit > References** from the menus. A list of references similar to those shown in the following figure displays.



2. Select the checkbox next to the reference you want to add.
3. If the desired reference is not listed, click the Browse button and locate the type of library you want to use.

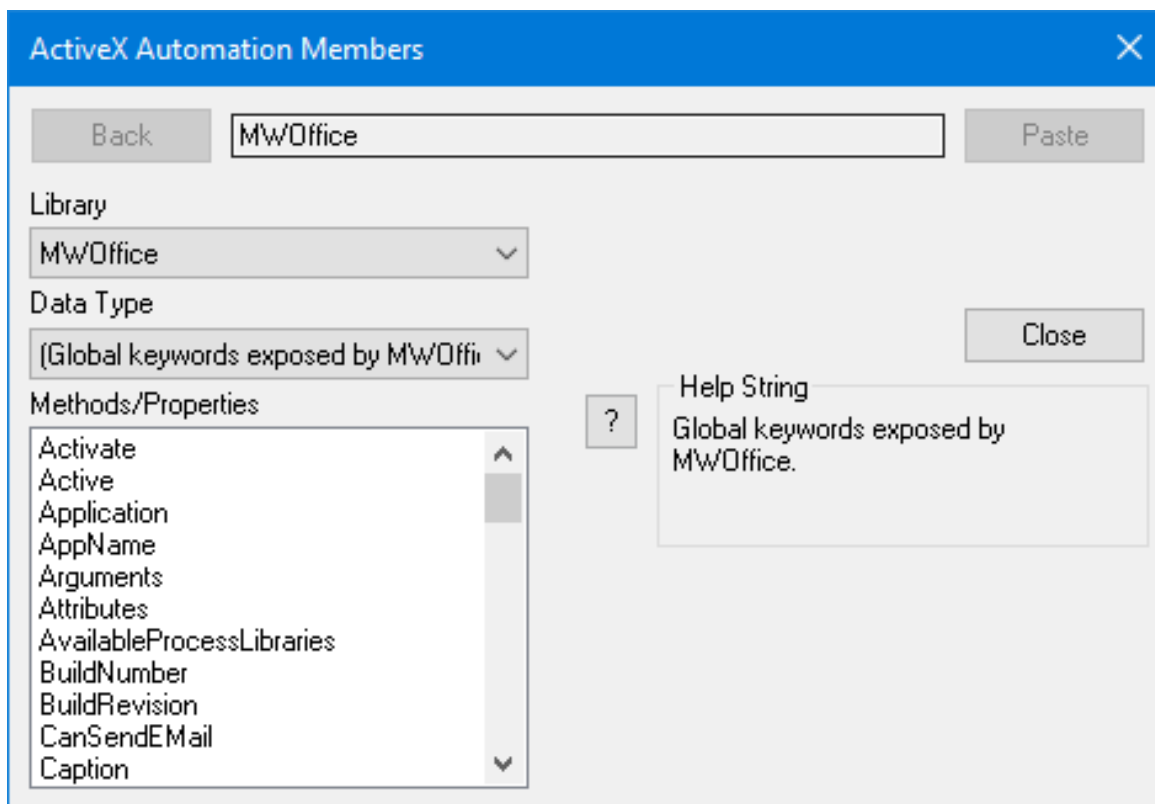
The AWRDE object library is selected by default and references must be set for the object module and each code module separately. Each checked reference is searched in order from top to bottom.

## Object Browser

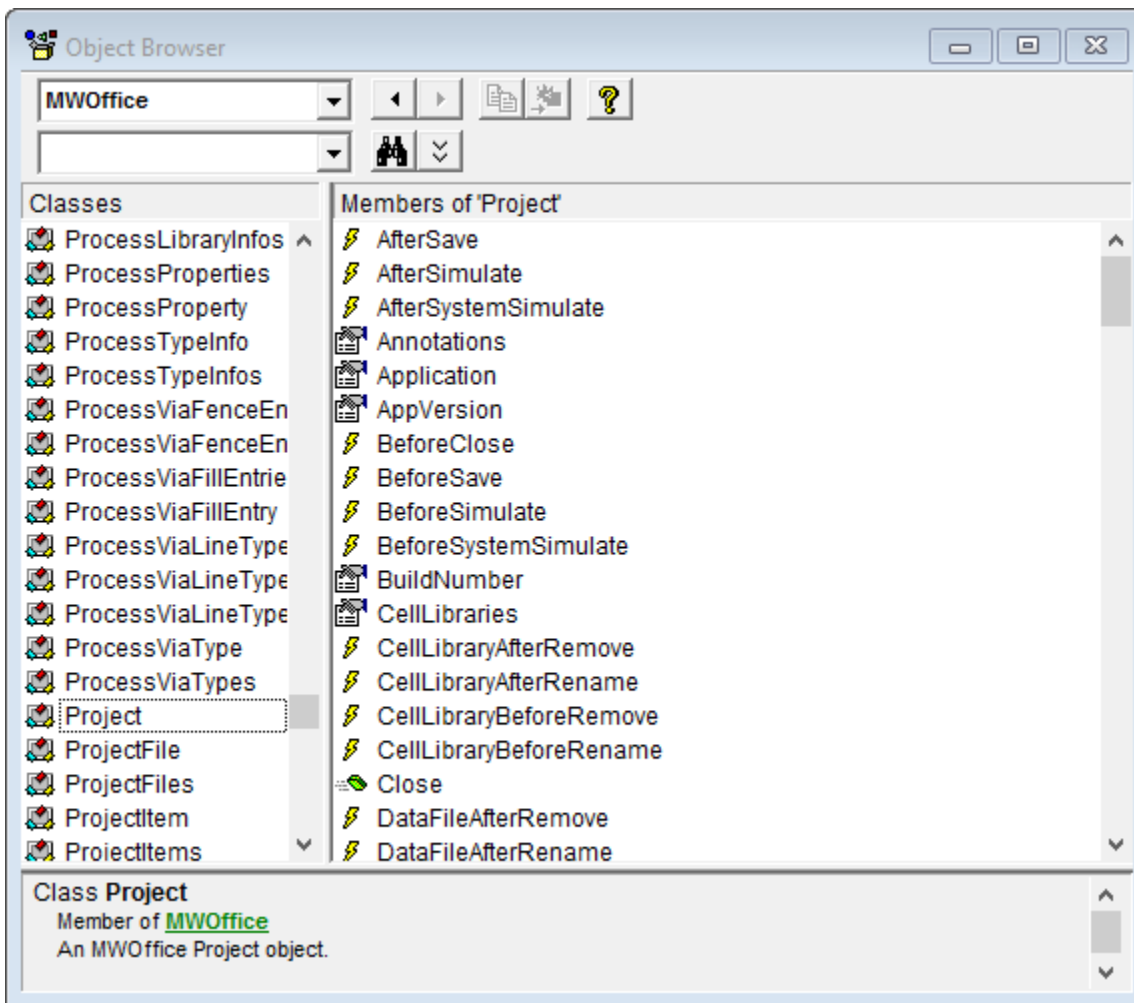
The Object Browser shows information about all the special data types that are available.

To open the object browser in the SDE, click the object browser button on the toolbar or choose **View > Object Browser** from the menus

The following figure shows the object browser with the AWRDE (MWOffice) type library selected.

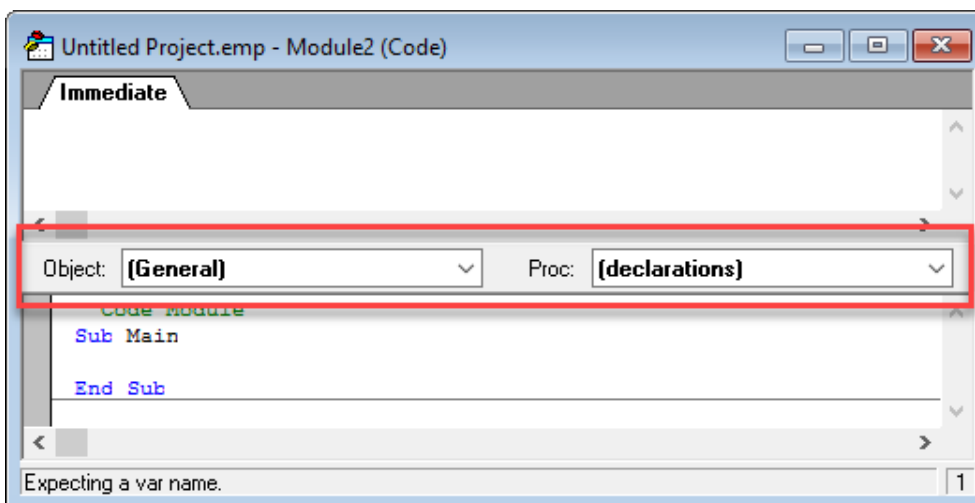


Note: more modern SDEs might provide more insightful object browsers. For example, creating a script in Microsoft Excel, adding a reference to the AWR type library and then using the object browser shows like below.



## Object and Proc Lists

The **Object** and **Proc** lists are shown at the top of any code or object module.



The object list shows all the objects in the current module

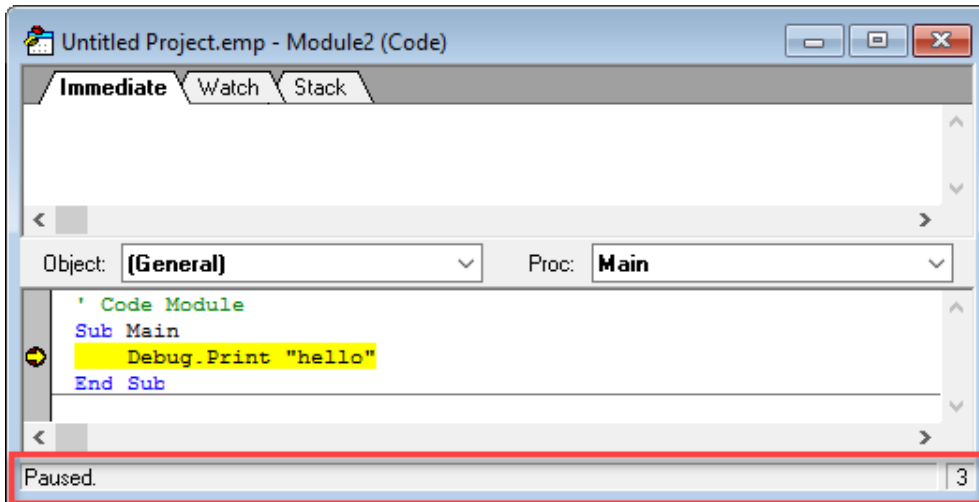
- The **(general)** object groups all of the procedures which are not part of any specific object.
- When in an object module, you can also select **Project**

The **Proc** list shows all the procedures (list of all subroutines and functions) for the current object.

- Selecting a procedure that is not bold inserts the proper procedure definition for that procedure, this mostly applies to object modules.
- Selecting a procedure that is bold will arrange the file such that the selected procedure is in view.
- **(declarations)** takes you to the beginning of the file for where global variables can be defined.

## Status Bar

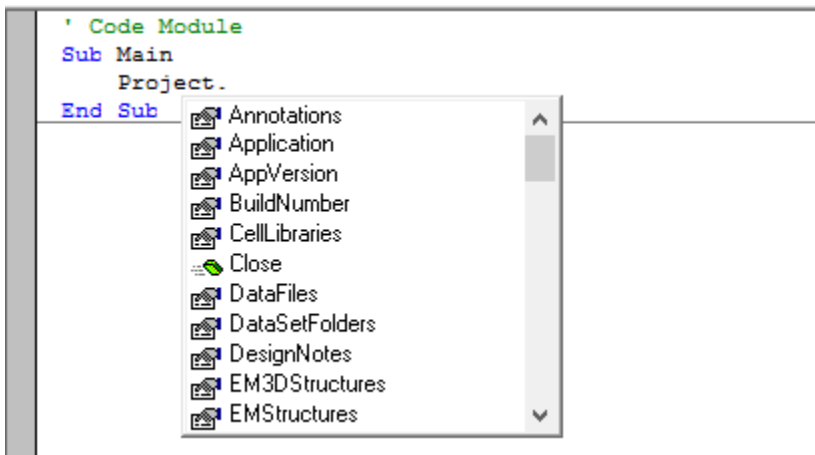
The status bar gives more information about what is happening with the code. The right side of the status bar also lists the file line number for the cursor.



## Auto-Complete

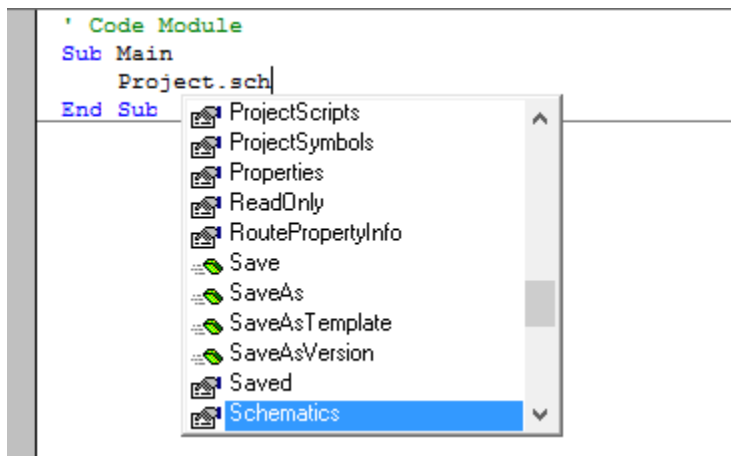
Auto-completion in the IDE uses the object browser information to show the current object's methods and properties.

For example, when you type "project.", you will get a list of the available objects available under the Project object.



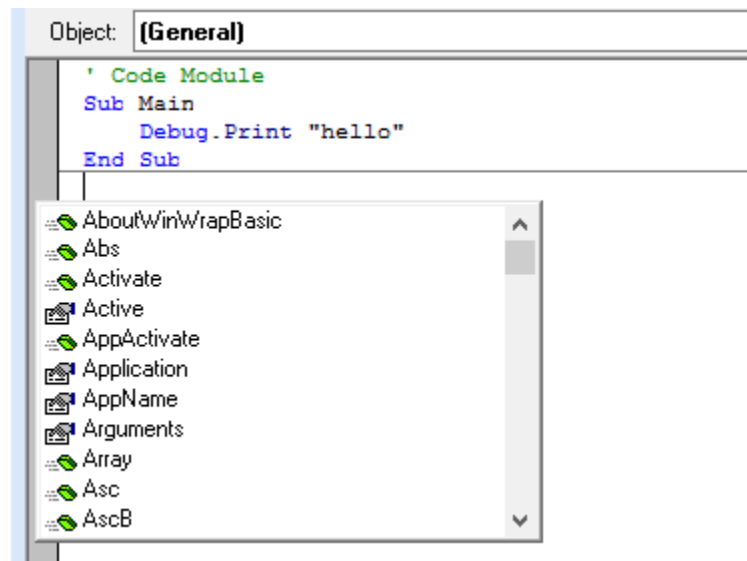
As you continue typing, it will match the closest child object and will be highlighted. For example, typing "project.sch" will show as below.





Notice the object that is highlighted. If you press the **Tab** key, that object is completed.

To see language extensions, built-in instructions/functions, and user-defined procedures/variables press **Ctrl-Space** on a blank line in the IDE.



## Referencing Subroutines and Functions in Other Files

If you want to reference subroutines or functions in other \*.bas files, there are several options. Referencing code can be a good way to track common groups of code you want to reuse and easily update.

## Stand-Alone File on Disk

To reference subroutines and functions in a stand-alone file

Use the following in your script:

```
'#Uses "global.bas"
Sub main

End Sub
```

The path to the file can be an absolute path or a path relative to the AWR Project file in which you are working. The disadvantage of this approach is that you cannot debug into any referenced functions.

## Another Script Loaded in the SDE

To reference subroutines and functions in another script loaded in the SDE for that type (Global or Project)

Use the following in your script:

```
'#Uses "*global"
Sub main

End Sub
```

Notice a few differences. There is no path to the file. You use a "\*" before the name. You do not use the '.bas' extension for the file.

**NOTE:** You cannot reference items in a Project level script in Global scripts, nor items in a Global level script in Project scripts.

## Getting Help

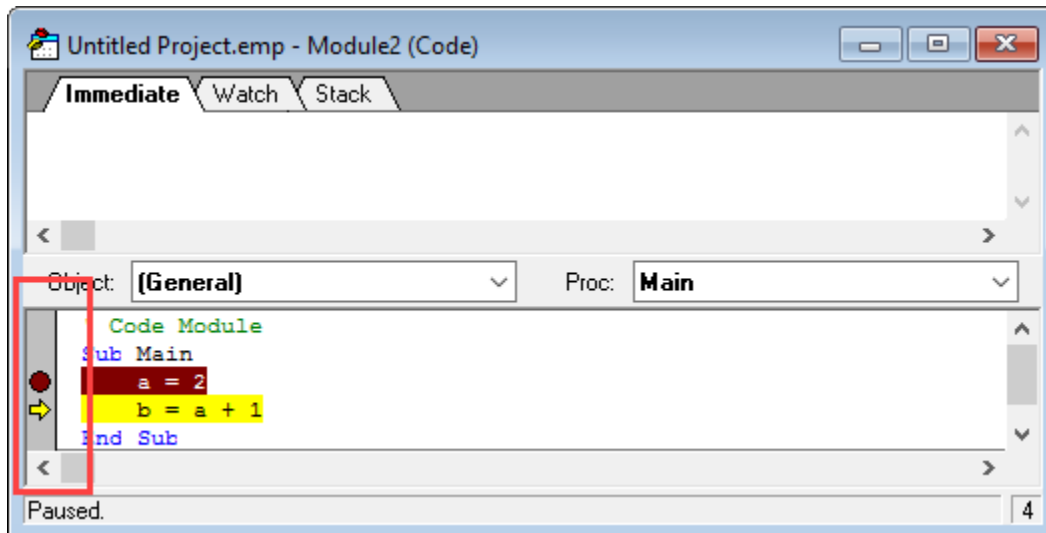
When typing code, press the **F1** key to display Help for the specific word the cursor is over. If the cursor is over a component that belongs to the AWR API, the AWRDE Help opens to the appropriate section. Newer Microsoft operating systems do not support the help system build into the current SDE. We recommend you use the [PDF documentation](#) to understand the SAX basic specific commands.

## Debugging Scripts

There are various techniques to debug scripts that will be described below.

### Break Bar

The break bar shows which lines have breakpoints. It also shows which line is next to execute. Clicking on the break bar toggles the breakpoint for that line. Alternately, the **Debug > Toggle Breakpoint** will toggle the current line of the cursor to have a breakpoint or not.



## Stepping Through Code

When your code has stopped at a breakpoint, you have various options; the options are available from the **Debug** menu. See the menu listings below for the hotkeys.

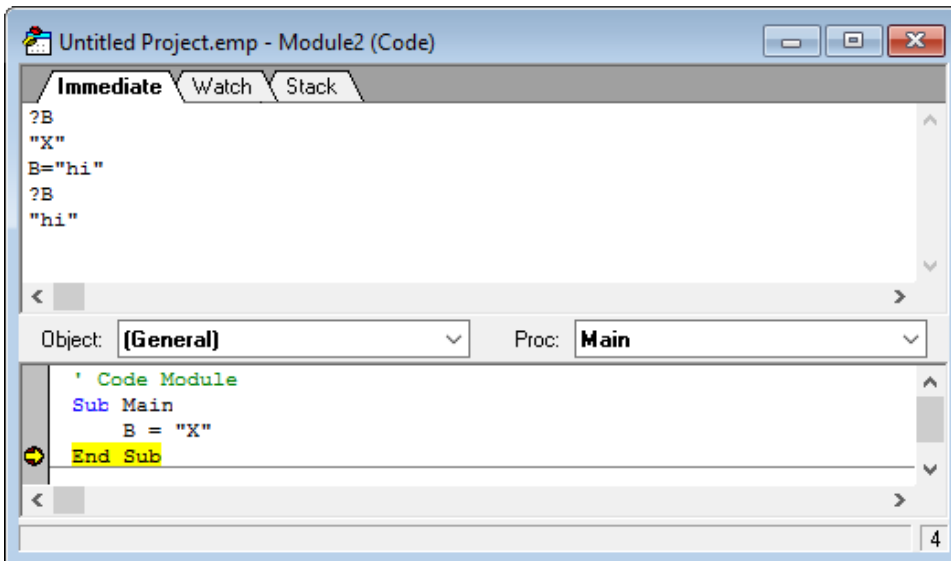
1. **Step Into** - Steps to the next line of code and into any subroutines or functions.
2. **Step Over** - Steps to the next line of code and over any subroutines or functions.
3. **Step Out** - If in a subroutine or function, step out of the procedure to the next line of code.
4. **Run To Cursor** - Run to the current location of the cursor.

When you run the script after a breakpoint, the code will run to the next breakpoint or to completion.

## Immediate Window

The **immediate** window is automatically displayed above your code. Use your mouse to drag the divider between the code and the **Immediate** window to change how big each one is relative to the window size. The **Immediate** window is used to evaluate an expression, assign a variable or call a subroutine.

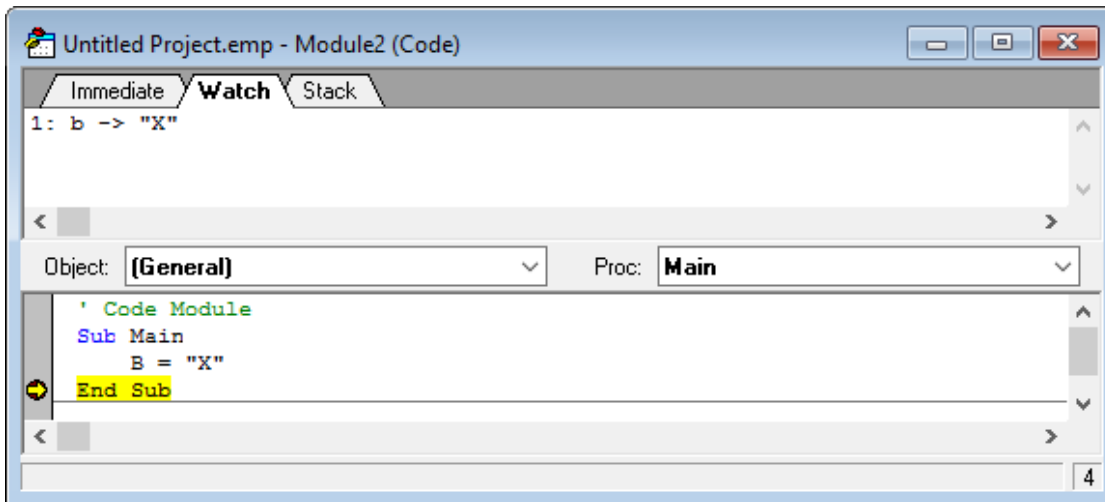
- Type "?expr" to show the value of "expr".
- Type "var = expr" to change the value of "var".
- Type "Set var = expr" to change the reference of "var".
- Type "subname args" to call a subroutine or built-in instruction.
- Type "Trace" to toggle trace mode. Trace mode prints each statement in the immediate window when a macro/module is running.



## Watch Window

The **Watch** window is automatically displayed above your code when the code paused during debugging. The **Watch** window is used to list the variables, functions, and expressions that are calculated and displayed.

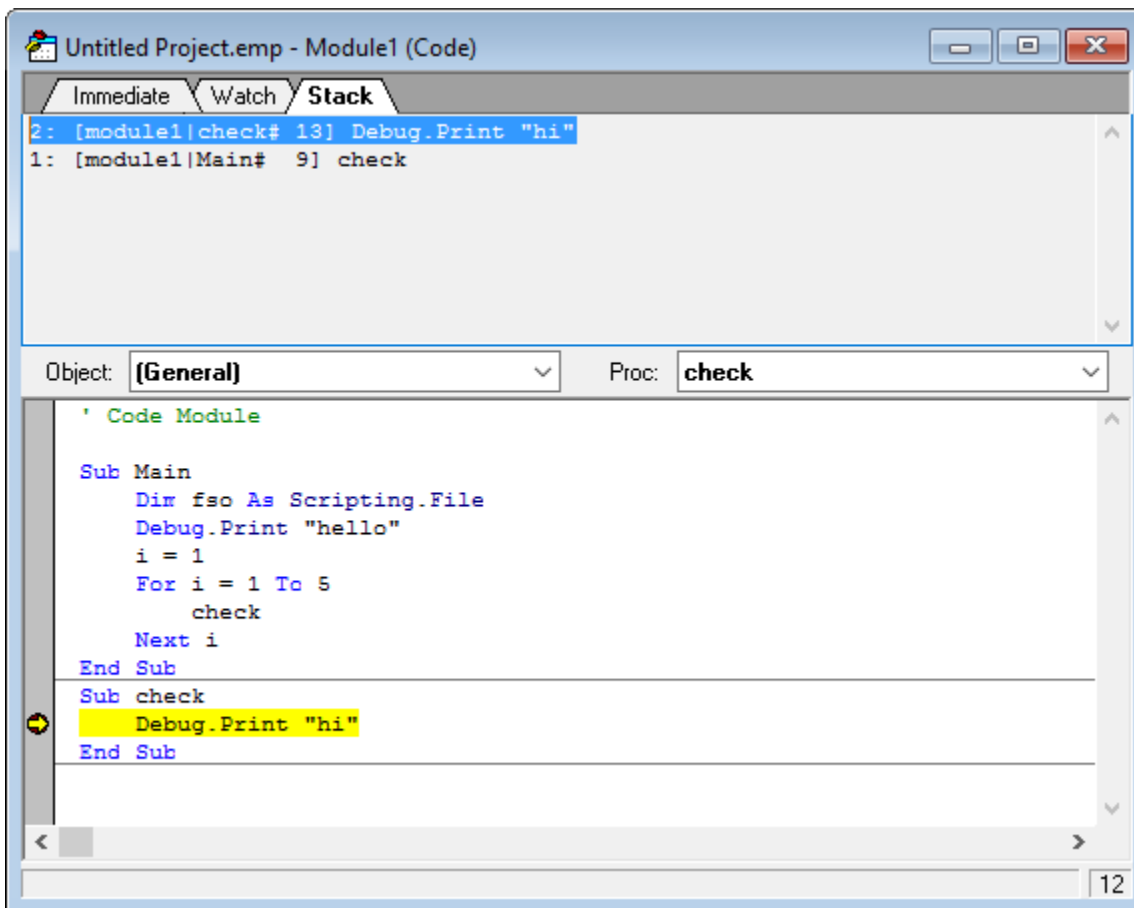
- Each time execution pauses the value of each line in the window is updated.
- The expression to the left of "->" may be edited.
- Pressing Enter updates all the values immediately.
- Pressing Ctrl-Y deletes the line.
- Add variables to the **Watch** window typing the name and typing **Enter**. Alternately, when the code is paused, put your cursor over a variable and select **Debug > Add Watch**.



## Stack Window

The **Stack** window is automatically displayed above your code when the code paused during debugging. The **Watch** window is used to list the lines which called the current statement.

- The first line is the current statement. The second line is the one that called the first. And so on.
- Clicking on a line brings that macro/module into a sheet and highlights the line in the edit window.



## Debug.Print Statement

You can use the 'Debug.Print' statement to print values of variables. The values will be printed in the **Immediate** window for you to view.

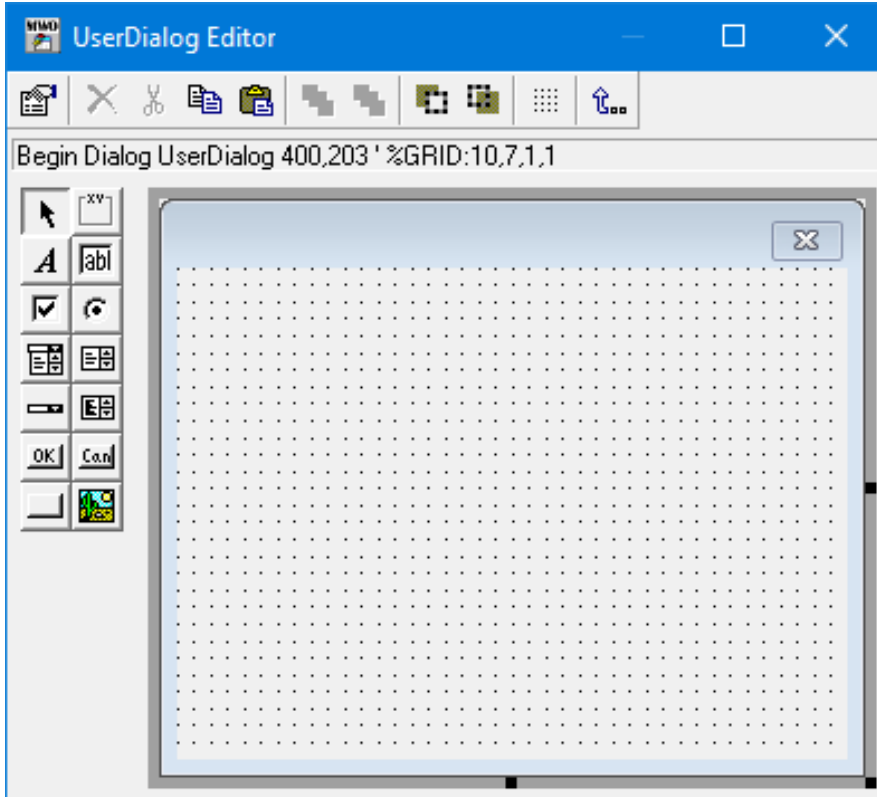
You can also use the 'Debug.Clear' statement to clear out the contents of the **Immediate** window.

**Note:** you can copy and paste from the **Immediate** window

## User Forms

UserDialog is described by a Begin Dialog...End Dialog block. To graphically add a UserDialog place the current selection in the code where you want the dialog and select Insert > UserForm from the Scripting Development Environment.

The following will display:



Use the controls on the left to add items to design your form. Hover your mouse over each item to understand what they are for. When you are done, close the dialog, and the code for the form will be inserted into your script.

To graphically edit a UserDialog place the current selection in a UserDialog block and select Insert > UserForm from the Scripting Development Environment. The dialog will open to edit the previously created form.

The following code is an **OK** and **Cancel** button.

```
Begin Dialog UserDialog 400,203 ' %GRID:10,7,1,1
    OKButton 50,161,90,21
    CancelButton 210,161,90,21
End Dialog
Dim dlg As UserDialog
Dialog dlg
```

When you have a cancel button, you need to return a value from the dialog to make decisions. Below is the code changed to handle a cancel button press.

```

' Code Module
Sub Main

    Begin Dialog UserDialog 400,203 ' %GRID:10,7,1,1
        OKButton 50,161,90,21
        CancelButton 210,161,90,21
    End Dialog
    Dim dlg As UserDialog
    rtn =Dialog(dlg)
    If rtn = 0 Then 'cancel pressed
        End
    End If

```

After adding a control, double-click to edit properties for each control. The example below is a text field in a dialog.

**Edit Text Properties**

Left

Top

Width

Height

Caption  ☒ Quoted

Field

Comment

There are options for the dialog itself. You double click anywhere in the editor, not on a control item.

**Edit UserDialog Properties**

Left  ☒ Centered

Top

Width

Height

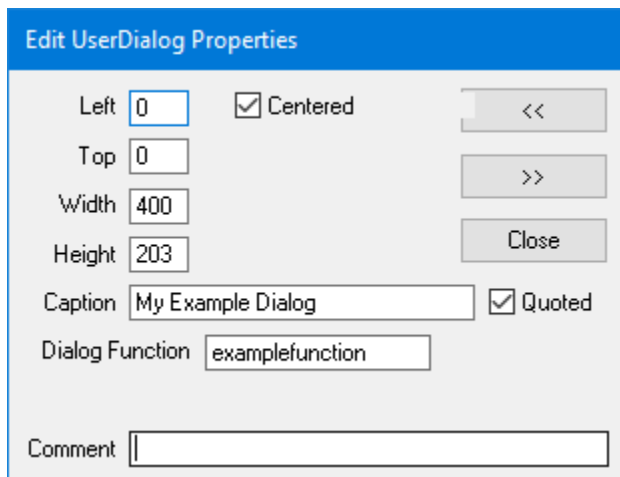
Caption  ☒ Quoted

Dialog Function

Comment

The **Caption** field will be the name of the dialog, and the **Dialog Function** name is to reference the function to do advanced dialog controls.

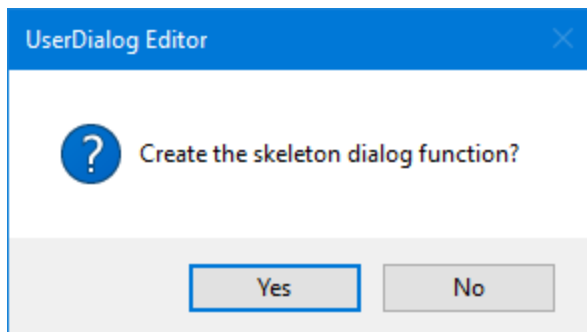
The example below shows both fields entered.



The 'Edit UserDialog Properties' dialog box features a blue title bar. It contains several input fields and checkboxes: 'Left' (0), 'Top' (0), 'Width' (400), 'Height' (203), 'Caption' (My Example Dialog), 'Dialog Function' (examplefunction), and 'Comment' (empty). There are checkboxes for 'Centered' and 'Quoted', both of which are checked. Navigation buttons '<<' and '>>' are located to the right of the position and size fields. A 'Close' button is positioned to the right of the 'Height' field.

Left	0	<input checked="" type="checkbox"/> Centered	<<
Top	0		>>
Width	400		
Height	203		Close
Caption	My Example Dialog		
Dialog Function	examplefunction		
Comment			

When closing the dialog with the **Dialog Function** field filled, you will get the prompt below. You usually would click next, and then the shell of the dialog function code will be added.

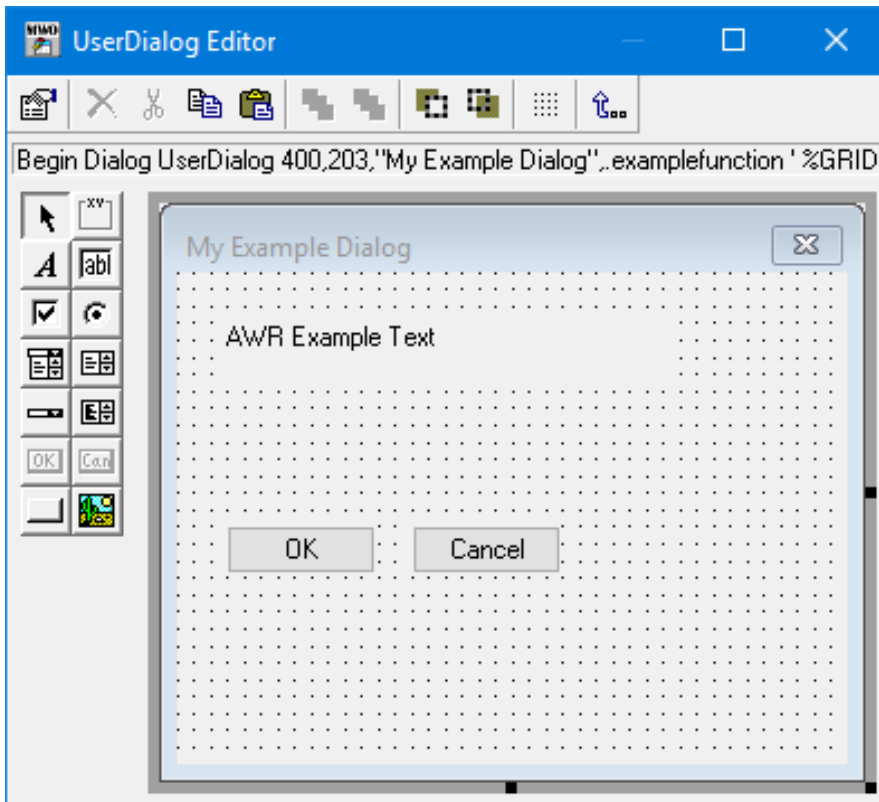


The 'UserDialog Editor' dialog box has a blue title bar with a close button. It displays a question mark icon and the text 'Create the skeleton dialog function?'. At the bottom, there are two buttons: 'Yes' and 'No'.

UserDialog Editor	
? Create the skeleton dialog function?	
Yes	No

With the form looking like below,





the code in the editor will be.

```
' Code Module
Sub Main
    Begin Dialog UserDialog 400,203,"My Example Dialog",.examplefunction ' %GRID:10,7,1,1
        Text 30,21,270,28,"AWR Example Text",.Text1
        OKButton 30,105,90,21
        CancelButton 140,105,90,21
    End Dialog
    Dim dlg As UserDialog
    Dialog dlg
End Sub

Rem See DialogFunc help topic for more information.
Private Function examplefunction(DlgItem$, Action%, SuppValue&) As Boolean
    Select Case Action%
        Case 1 ' Dialog box initialization
        Case 2 ' Value changing or button pressed
            Rem examplefunction = True ' Prevent button press from closing the dialog box
        Case 3 ' TextBox or ComboBox text changed
        Case 4 ' Focus changed
        Case 5 ' Idle
            Rem Wait .1 : examplefunction = True ' Continue getting idle actions
        Case 6 ' Function key
    End Select
End Function
```

## Running Scripts from the AWR Design Environment

You can run scripts directly from the AWRDE **Scripts** menu. Scripts display either one of three areas.

1. **Custom Name**
2. **Scripts > Project Scripts**
3. **Scripts > Global Scripts**

The list that displays is in the format<module name>(<sub name>). Any Sub in a given module that takes no input arguments is available in this list. You can hide any such sub by using the **Private** declaration before the name of the subroutine. For example, the following figure shows a macro named "example" that has three subroutines, a,b, and c.

[blocked URL](#)

Notice that the third sub is declared private. If you choose **Scripts > Project Scripts** you can access only the first two subroutines as shown in the following figure.

[blocked URL](#)

You can control how scripts display in the **Scripts** menu. Above any Sub, you can add a line that defines the folder in which to display the script. This line is in the form ' \$MENU=name where name is the desired folder name. The name of the sub displays under that folder. You can use the folder name "hidden" to prevent that sub from displaying. For example, you can add the line " \$MENU=Data" to display this script under the Data sub menu. Only one level of folders is allowed. When you use this syntax the script is only in this folder; it is no longer in the **Scripts > Project Scripts** or **Scripts > Global Scripts** folders. Because of this structure, AWR recommends that you write your scripts with one Sub Main that uses the hidden folder tag, and then give your code a meaningful name that you then call from the Sub Main. This is because you need a Sub Main to run the script from the SDE, but you want a meaningful name when running the script from the Scripts menu. The following example scripting code demonstrates these various concepts.

```
' $MENU=hidden
sub main
    Delete_Unassociated_iNets
end sub
' $MENU=Layout
Sub Delete_Unassociated_iNets
End sub
```

The following figure shows how this script displays in the AWRDE menu.

[blocked URL](#)

Notice that there is a folder named Layout and a script named "Delete\_Unassociated\_iNets" under that folder.

You can use menu, toolbar, or hotkey customizations to run any macro. See “[Customizing Toolbars and Menus](#)” and “[Customizing Hotkeys](#)” for details on customizing these objects. When customizing, look for the **Categories** setting equal to **Macros**. You only should customize global scripts since they are guaranteed to be available to run in each project opened.

## Menus

### Edit Area Speed Menus

When in the edit menu, right click to get the following menu items.

Item	Description
Cut	Move the selected text to the Clipboard. ( <b>Ctrl+X</b> )
Copy	Copy the selected text to the Clipboard. ( <b>Ctrl+Y</b> )
Paste	Paste the Clipboard text over the selected text. ( <b>Ctrl+V</b> )
Run	Run the macro to completion. (If the macro is not active, start it.) ( <b>F5</b> )

Pause	Stop the macro/module. Execution can be continued. ( <b>Esc</b> )
End	Terminate the macro/module. Execution cannot be continued.
Step Into	Execute the current line. If the current line is a subroutine or function call, stop on the first line of that subroutine or function. (If the macro is not active, start it.) ( <b>F8</b> )
Step to Cursor	Execute until the line the cursor is on is the current line. (If the macro is not active, start it.) ( <b>F7</b> )
Toggle Break	Toggle the breakpoint on the current line. ( <b>F9</b> )
Clear All Breaks	Clear all breakpoints. ( <b>Shift+Ctrl+F9</b> )
Quick Watch	Show the value of the expression under of the cursor in the immediate window.
Browse	Open the Object Browser window and show the item at the current cursor location
Show Next Statement	Show the next statement to be executed.

## File Menu

The File menu provides the normal options.

Item	Description
MWOffice	Switch control to the AWR Design Environment
Save	Save the current AWRDE Project.
Print	Print the current macro/module.
Print Setup	Select the default printer.
Close	Close the SDE

## Edit Menu

The File menu provides the normal options.

Item	Description (hot Key)
Undo	Undo the last edit. ( <b>Ctrl+Z</b> )
Redo	Redo the last edit. ( <b>Ctrl+Y</b> )
Cut	Move the selected text to the Clipboard. ( <b>Ctrl+X</b> )
Copy	Copy the selected text to the Clipboard. ( <b>Ctrl+Y</b> )
Paste	Paste the Clipboard text over the selected text. ( <b>Ctrl+V</b> )
Delete	Delete the selected text. ( <b>Del</b> )
Select All	Select all of the text. ( <b>Ctrl+A</b> )
Find...	Find a string. ( <b>Ctrl+F</b> )
Replace...	Replace a string with another. ( <b>Ctrl+R</b> )
Find Next	Repeat last find or replace. ( <b>F3</b> )
Indent	Move selected lines right. ( <b>Tab</b> )
Outdent	Move selected lines left.
Reference	Edit the macro/module's references.
Parameter Info	Show the parameter information. ( <b>Ctrl+I</b> ). Used on procedures to get input and output information.
Complete Word	Complete the word. ( <b>Tab</b> ) When auto-complete has selected the next object, this will complete the identified word.
Font...	Font setting for Edit Area

## View Menu

The View menu provides the normal options.

Item	Description (Hot Key)
------	-----------------------

Object Browser	Open the Object Browser window and show the item at the current cursor location.
Split Window	Toggle the split on/off for Immediate window and edit area.
Immediate Window	Show the immediate output window. ( <b>Ctrl+E</b> )
Watch Window	Show the watch expressions window. ( <b>Ctrl+W</b> )
Call Stack	Show the call stack window. ( <b>Ctrl+T</b> )
Project Window	Open project window if closed
Toolbars	Toggle the Debug, Edit and Standard toolbars on/off.
Status Bar	Toggle the status bar on/off. Located at the bottom of the SDE.

## Insert Menu

The Insert menu provides the normal options.

Item	Description
User Form	Opens the user interface to design a dialog
Code Module	Adds a new code module either to Local or Global scripts depending on where your cursor was last in the project tree

## Debug Menu

The Debug menu provides the normal options.

Item	Description (Hot Key)
Step Into	Execute the current line. If the current line is a subroutine or function call, stop on the first line of that subroutine or function. (If the macro is not active, start it.) ( <b>F8</b> )
Step Over	Execute to the next line. If the current line is a subroutine or function call, execute that subroutine or function completely. ( <b>Shift+F8</b> )
Step Out	Step out of the current subroutine or function call. ( <b>Ctrl+F8</b> )
Run to Cursor	Execute until the line the cursor is on is the current line. (If the macro is not active, start it.) ( <b>F7</b> )
Quick Watch	Show the value of the expression under of the cursor in the immediate window.
Add Watch	Add the expression under of the cursor in the watch window, only available when a script is paused.
Toggle Breakpoint	Toggle the breakpoint on the current line. ( <b>F9</b> )
Clear All Breakpoints	Clear all breakpoints. ( <b>Shift+Ctrl+F9</b> )
Set Next Statement	Set the next statement to be executed. Only statements in the current subroutine/function can be selected.
Show Next Statement	Show the next statement to be executed.

## Run Menu

The Run menu provides the normal options.

Item	Description (Hot Key)
Run Sub	Run the macro to completion. (If the macro is not active, start it.) ( <b>F5</b> )
Split Window	Stop the macro/module. Execution can be continued. ( <b>Esc</b> )
Immediate Window	Terminate the macro/module. Execution cannot be continued.

## Help Menu

The Help menu provides the normal options.

Item	Description (Hot Key)
About AWR Scripting	Information dialog box.

