

## VBA Tip – Arrays

### Efficient Array Looping

VBA provides the `LBound` and `UBound` functions for retrieving the lower and upper bounds of an array. When looping through an array many developers will conserve lines of code by putting the `LBound` and/or `UBound` functions in the loop test statement. This can decrease the performance of your application as these functions are called on every iteration through the loop.

To improve the performance of your application you should assign the `LBound` and `UBound` values to variables prior to entering the loop. Then use the variables in the loop test statement. Using this method will only make a single call to these functions instead of one call on every iteration.

The subroutine that follows illustrates the two methods.

```
'  
' Efficient array looping  
'  
Sub ArrayExample1()  
    Dim myArray() As Long  
    Dim numItems As Long  
    Dim indexCount As Long  
    Dim indexStart As Long  
    Dim indexEnd As Long  
  
    numItems = 9999999  
    ReDim myArray(numItems)  
  
    '  
' Commonly used method for looping through an array  
' The UBound function is called on every iteration  
'  
    For indexCount = LBound(myArray) To UBound(myArray)  
        '  
' Do stuff here  
'  
    Next  
  
    '  
' For better performance do this  
' The UBound function is only called once  
'  
    indexStart = LBound(myArray)  
    indexEnd = UBound(myArray)  
    For indexCount = indexStart To indexEnd  
        '  
' Do stuff here  
'
```

```
Next
End Sub
```

## Determining if an array is dimensioned

The above example brings up a common question: "How can you determine if an array has been dimensioned?" Using the `LBound` or `UBound` functions on arrays that have not been dimensioned will raise an error. Use this to your advantage to create your own function to make this determination. The custom function `isArrayDimd` listed below illustrates how to accomplish this.

```
'
' Determine if array is dimensioned
'
Sub ArrayExample2()
    Dim myArray() As Long

    '
    ' Array is not dimensioned, will display "False"
    '
    MsgBox CStr(IsArrayDimd(myArray)), vbInformation, "IsArrayDimd"

    ReDim myArray(10)

    '
    ' Now that array is dimensioned, will display "True"
    '
    MsgBox CStr(IsArrayDimd(myArray)), vbInformation, "IsArrayDimd"
End Sub

'
' Custom function to do the test
'
Public Function IsArrayDimd(inArray() As Long) As Boolean
    On Error GoTo NOT_DIMD

    '
    ' The following line will raise an error if the array
    ' has not been dimensioned. It will always return "True"
    ' for any array that has been dimensioned.
    '
    IsArrayDimd = UBound(inArray) >= 0
    Exit Function

NOT_DIMD:
    IsArrayDimd = False
End Function
```

## Putting it all together

Now that you can efficiently loop through an array, and determine if an array has been dimensioned, you can put it all together as is illustrated in the following subroutine.

```
'  
' Combining Examples 1 and 2  
'  
Sub ArrayExample3()  
  Dim myArray() As Long  
  Dim numItems As Long  
  Dim indexCount As Long  
  Dim indexStart As Long  
  Dim indexEnd As Long  
  
  numItems = 9999999  
  
  '  
  ' If array is not yet dimensioned, dimension it  
  '  
  If Not IsArrayDimd(myArray) Then  
    ReDim myArray(numItems)  
    '  
    ' You could also raise your own custom error here  
    ' and pass it off to your error handler  
    '  
  End If  
  
  indexStart = LBound(myArray)  
  indexEnd = UBound(myArray)  
  For indexCount = indexStart To indexEnd  
    '  
    ' Do stuff here  
    '  
  Next  
End Sub
```

[The Envision Group, Inc.](http://www.TheEnvisionGroup.net) is a small business corporation with an 11-year history of providing CADD training and consulting, IT programming and support, and 3-D virtual modeling for government and private organizations both nationally and internationally.