kiMisc.dll Kenneth Ives kenaso@tx.rr.com

I am open to ways to improve this application, please email me.

The VB 6.0 Service Pack 6 runtime files are required for this software to run.

VBRun60sp6.exe installs Visual Basic 6.0 SP6 run-time files
http://support.microsoft.com/kb/290887

To obtain the VBRun60sp6.exe file, visit the following Microsoft Web site:
http://www.microsoft.com/downloads/details.aspx?FamilyId=7B9BA261-7A9C-43E7-9117-F673077FFB3C

This software has been tested on Windows XP through Windows 7.
Windows 9x, 2000 and NT4 are no longer supported.

References:

Randomize Statement Doesn't Re-initialize Rnd Function
http://support.microsoft.com/default.aspx?scid=kb;en-us;120587

"To re-initialize the random-number generator, use the Rnd function with a value of -1 and then use the Randomize statement with the value you want to use as the seed value for the Rnd function."

VBA's Pseudo Random Number Generator
http://www.noesis.net.au/prng.php

Mark Hutchinson article about the Microsoft Visual BASIC random number generator.
An Examination of Visual Basic's Random Number Generation
http://www.15seconds.com/issue/051110.htm

INFO: How Visual Basic Generates Pseudo-Random Numbers for the RND Function
http://support.microsoft.com/kb/231847/en-us

RND and RANDOMIZE Alternatives for Generating Random Numbers
http://support.microsoft.com/kb/28150/EN-US/

```
================================================================================
Available in cPRNG (clsRandom)
A cryptographically random number generator using Microsoft's CryptoAPI.
================================================================================

' ************************************************************************
' Enumerations
' ************************************************************************
  Public Enum enumPRNG_ReturnFormat
      ePRNG_ASCII           ' 0
      ePRNG_HEX             ' 1
      ePRNG_HEX_ARRAY       ' 2
      ePRNG_BYTE_ARRAY      ' 3
      ePRNG_LONG_ARRAY      ' 4
      ePRNG_DBL_ARRAY       ' 5
  End Enum

  Public Enum enumPRNG_HashAlgorithm
      ePRNG_MD2             ' 0
      ePRNG_MD4             ' 1
      ePRNG_MD5             ' 2
      ePRNG_SHA1            ' 3
      ePRNG_SHA256          ' 4
      ePRNG_SHA384          ' 5
      ePRNG_SHA512          ' 6
  End Enum

  Public Enum enumPRNG_Compare
      ePRNG_CaseSensitive   ' 0 - Exact byte match
      ePRNG_IgnoreCase      ' 1 - Uppercase/Lowercase considered same
  End Enum

' ************************************************************************
' ****                       Properties                               ****
' ************************************************************************

  StopProcessing - Input/Output - Boolean - True if user wants to stop processing

  AES_Ready - Output - Boolean - True if operating system can use SHA2 functionality

  CompareMethod - Input - Long Integer - Designates type of data comparison to be used

' ************************************************************************
' ****                       Methods                                  ****
' ************************************************************************
' Build random data using ASCII values 0-255.
Function BuildRndData(ByVal lngDataLength As Long, _
             Optional ByVal lngReturnFormat As enumPRNG_ReturnFormat =
ePRNG_BYTE_ARRAY, _
             Optional ByVal blnCreateExtraSeed As Boolean = True) As Variant

' Build random data that falls between two ASCII values, inclusive.
Function BuildWithinRange(ByVal lngDataLength As Long, _
                 Optional ByVal lngLowValue As Long = 0, _
                 Optional ByVal lngHighValue As Long = 255, _
                 Optional ByVal lngRetDataType As enumPRNG_ReturnFormat =
enuByteArray, _
                 Optional ByVal blnCreateExtraSeed As Boolean = True) As Variant

' The data will be SORTED.  This routine removes all duplicates based on
' user selection of case sensitivity. The number of duplicates removed
' are returned.
Function RemoveDupes(ByRef avntData As Variant, _
```

```
            Optional ByRef lngDupeCnt As Long = 0, _
            Optional ByVal blnReturnMixed As Boolean = False) As Boolean

' An array of data passed to this routine will be rearranged.
Sub ReshuffleData(ByRef avntData As Variant, _
         Optional ByVal lngMixCount As Long = 25)

' With this routine you can generate a series of non-repeating numbers.
' An array will be loaded starting with the base number (lngMinValue)
' requested up to the maximum value requested (lngMaxValue).  You can
' also enter the incremental step between the minimum and maximum value.
' This array is then passed to another routine ReshuffleData() to be
' throughly rearranged.  When it is returned, the requested number of
' elements (lngReturnQty) from the mixed array are transferred
' sequentially to the return array (alngMixed()).
'
' Syntax:  x = NonRepeatingNbrs(100, 0, 9999, 5)
'          Return 100 numbers, lowest = 0, highest = 9999,
'          incremental step = 5, Sort return data in
'          Ascending order (default)
Function NonRepeatingNbrs(ByVal lngReturnQty As Long, _
                          ByVal lngMinValue As Long, _
                          ByVal lngMaxValue As Long, _
                 Optional ByVal lngStep As Long = 1, _
                 Optional ByVal blnSortData As Boolean = True) As Long()

' CombSort is faster than all but QuickSort and close to it.  On the
' other hand, the code is much simpler than QuickSort and can be easily
' customized for any array type.  The CombSort was first published by
' Richard Box and Stephen Lacey in the April 1991 issue of Byte magazine.
Function CombSort(ByRef avntData As Variant, _
         Optional ByVal blnAscending As Boolean = True) As Boolean

' Generate a one-way hash string from a string of data. These are the
' algorithms to use:   MD2  MD4  MD5  SHA-1  SHA-256  SHA-384  SHA-512
'
' Special note:  SHA-224, SHA-512/224 and SHA-512/256 have not yet been
' implemented into the Microsoft crypto suite of hashes.
Function CreateHash(ByVal strInput As String, _
           Optional ByVal lngHashAlgo As enumPRNG_HashAlgorithm = ePRNG_SHA512, _
           Optional ByVal blnReturnAsHex As Boolean = True) As String

' Generate a random long integer between two input values.
Function GetRndValue(ByVal sngLow As Single, _
                     ByVal sngHigh As Single) As Long

' Convert a long integer to a double precision number.  Returns a decimal
' position of 14 places.
Function LongToDouble(ByVal lngValue As Long) As Double

' This is an ArrPtr function that determines if the passed array is
' initialized, and if so will return the pointer to the safearray header.
' If the array is not initialized, it will return zero.
' Syntax:   If CBool(IsArrayInitialized(array_being_tested)) Then ...
Function IsArrayInitialized(ByVal avntData As Variant) As Long

' Properly empty and deactivate a collection
Sub EmptyCollection(ByRef colData As Collection)

' This little code snippet returns a truly random value.
Function RndSeed() As Double

' Swap data with each other.  Wrote this function since BASIC stopped
```

```
' having its own SWAP function.  Use this for swapping strings, type
' structures, numbers with decimal values, etc.
Sub SwapData(ByRef vntData1 As Variant, _
             ByRef vntData2 As Variant)

' Swap numeric data (byte, integer, or long) with each other
' without using a temporary holding variable.
Sub SwapLong(ByRef AA As Long, _
             ByRef BB As Long)

Sub SwapInt(ByRef AA As Integer, _
            ByRef BB As Integer)

Sub SwapBytes(ByRef AA As Byte, _
              ByRef BB As Byte)

' Converts a byte array to string data.
Function ByteArrayToString(ByRef abytData() As Byte) As String

' Converts string data to a byte array.
Function StringToByteArray(ByVal strData As String) As Byte()

' Creates a unique string of hex data using CryptoAPI hash functions.  Also,
' randomly select a starting position in hashed data string to capture two
' eight byte strings of data.  These will be converted into long integers
' for new carryover values.
Function CreateExtraSeed(Optional ByVal lngRetLength As Long = 0) As String
```

```
================================================================================
Available in cDiskInfo (clsDiskInfo)
================================================================================

' ************************************************************************
' Enumerations
' ************************************************************************
  Public Enum enumDiskSpace
      eFreespace         ' 0
      eTotalSize         ' 1
      eUsedSpace         ' 2
      eAvailSpace        ' 3
  End Enum

  Public Enum enumDriveType
      eUnknown           ' 0 Unknown drive type
      eBadRoot           ' 1 No root directory
      eRemovable         ' 2 Floppy or Jaz drive
      eFixed             ' 3 Local hard drive
      eNetwork           ' 4 Shared Network drive
      eCDRom             ' 5 CD-Rom drive (CD or DVD)
      eRamdisk           ' 6 Virtual memory disk
  End Enum

  Public Enum enumIDE_DRIVE_NUMBER
      ePrimaryMaster     ' 0
      ePrimarySlave      ' 1
      eSecondaryMaster   ' 2
      eSecondarySlave    ' 3
      eTertiaryMaster    ' 4
      eTertiarySlave     ' 5
      eQuartiaryMaster   ' 6
      eQuartiarySlave    ' 7
  End Enum

  ' Status Flags Values
  Public Enum enumSTATUS_FLAGS
      ePrefailureWarranty = &H1
      eOnLineCollection = &H2
      ePerformanceAttribute = &H4
      eErrorRateAttribute = &H8
      eEventCouontAttribute = &H10
      eSELF_PRESERVING_ATTRIBUTE = &H20
  End Enum


' ************************************************************************
' ****                        Properties                              ****
' ************************************************************************
  StopProcessing - Input/Output - Boolean - True if user wants to stop processing

  DriveType - Output - String - Type of drive (Ex:  Physical Hard Drive)

  DriveTypeExtra - Output - String - Type of drive extras (Ex:  Fixed Hard Drive)

  FormattedSize - Output - String - Formatted drive size (Ex:  70.1 GB)

  Partition - Output - String - Partition information (Ex:  Disk #0, Partition #1)

  VolumeName - Output - String - Name of partition.  Changes with each format.
                                 (Ex:  HP_PAVILION)

  VolumeSerial - Output - String - Serial number assigned to partition after each
```

format. (Ex: 4FC0-112D)

FileSysType - Output - String - Type file system assigned during format (Ex: NTFS)

MfgHDSerial - Output - String - Manufacturer hard drive serial number. Cannot be changed by user.

MfgHDModel - Output - String - Manufacturer hard drive model number. Cannot be changed by user.

MfgHDFirmware - Output - String - Manufacturer hard drive firmware number. Cannot be changed by user.

BytesPerSector - Output - Long - Bytes per sector as defined when disk is formatted.

BytesPerCluster - Output - Long - Bytes per sector as defined when disk is formatted.

SectorsPerTrack - Output - Long - Sectors assigned to each track.

TotalCylinders - Output - Currency - Total number of cylinders per hard drive.

TracksPerCylinder - Output - Long - Number of tracks per cylinder.

TotalSectors - Output - Long - Total number of sectors per hard drive.

TotalClusters - Output - Long - Total number of clusters per hard drive.

FreeClusters - Output - Long - Total number of free clusters per hard drive.

TotalBytesFree - Output - Currency - Total number of free bytes per hard drive.

AvailableBytes - Output - Currency - Total number of available bytes per hard drive.

UsedSpace - Output - Currency - Amount of used space on hard drive.

TotalDiskSpace - Output - Currency - Total amount of space on hard drive.

PartitionSpace - Output - Currency - Amount of space on hard drive after formatting.

```
' ************************************************************************
' ****                        Methods                                 ****
' ************************************************************************

' Captures information about a specific drive into a type data structure.
Sub GetDriveInfo(ByVal strDrive As String)

' Make an API call to a specific drive and capture the volume information.
' Returns a brief desription of drive (ex: "Remote (network) drive").
Function GetDriveDescription(ByVal strDrive As String) As String

' Determines if the drive is a CD-Rom drive and returns its handle.
Function IsCDRomDrive(ByVal strDrive As String) As Long

' Make an API call to a specific drive and capture the volume information
Function GetClusterSize(ByVal strDrive As String) As Long

' Capture a list of all available drive letters  (Ex: A:\, C:\, D:\, etc.)
Function GetDriveLetters() As String()

' Make an API call to a specific drive and capture the volume information
Sub GetVolumeInfo(ByVal strDrive As String, _
```

```
         Optional ByRef strVolName As String = "", _
         Optional ByRef strVolSerialNo As String = "", _
         Optional ByRef strFileSysType As String = "")

' Used to determine if a drive is of a specific type.  True - if there is
' a match else FALSE is returned.
Function SpecificTypeOfDrive(ByVal strDrive As String, _
                             ByVal lngTypeNeeded As enumDriveType) As Boolean

' Capture the specific space information about a selected drive.
Function GetDiskSpaceInfo(ByVal strDrive As String, _
                 Optional ByVal lngChoice As enumDiskSpace = eFreeSpace) As Currency

' Determine if a device is functioning.
Function IsDeviceReady(ByVal strDrive As String) As Boolean

' Lock a device.  True - Lock the device.  FALSE - Unlock the device.
Sub DeviceLock(ByVal strDrive As String, _
               ByVal blnLockDevice As Boolean)

' Use WMI (Windows Management Instrumentation) to obtain a drive's partition
' information and the disk number as assigned by the BIOS.
Function GetDiskNumber(ByVal strDrive As String) As Long

' Get hard disk manufacturer information.  Default is the first physical drive.
' EX:  strModel = "QUANTUM FIREBALLP AS20.5"
Sub GetMfrInfo(Optional ByVal lngDrvNumber As enumIDE_DRIVE_NUMBER = ePrimaryMaster, _
               Optional ByRef strSerial As String = "", _
               Optional ByRef strModel As String = "", _
               Optional ByRef strFirmware As String = "")

' Return a string representing the value in string format to requested number
' of decimal positions.   (Ex:  2,530,096 bytes --> 2.4 MB)
Function DisplayNumber(ByVal dblCapacity As Double, _
                       Optional ByVal lngDecimals As Long = 1) As String

' Create a nested directory structure.
' EX:  strPath = "C:\Program Files\MyDir\Sub_1\Sub_2"
Function CreateDirStructure(ByVal strPath As String) As Boolean

' See if CD drive is physically mounted or just a drive letter (Virtual)
Function IsCDMounted(ByVal strDrive As String) As Boolean
```

```
================================================================================
Available in cOperSystem (clsOperSystem)
================================================================================

' ************************************************************************
' ****                       Properties                                 ****
' ************************************************************************
  VersionName - Output - String data - the operating system version name.
                Example: Windows 2000

  VersionNumber - Output - String data - the operating system version number.
                  Example: 5.00

  BuildNumber - Output - String data - the operating system build number.
                Example: 2195

  VersionData - Output - String data - the full operating system version name.
                Example: Windows 2000 5.00.2195 Service Pack 4

  ServicePack - Output - String data - In Win9x, this can be any arbitrary
                string provided by the manufacturer. In NT based operating systems,
                this is the service pack.

  WinPlatformID -  Output - Long integer - Represents the operating system platform ID

  bWindowsNT - Output - Boolean - True if operating system is Windows NT based

  bWinNT4orNewer - Output - Boolean - True if operating system is Windows NT4 or newer

  bWin2000orNewer - Output - Boolean - True if operating system is Windows 2000 or
newer

  bWinXPorNewer - Output - Boolean - True if operating system is Windows Xp or newer

  bWinVistaOrNewer - Output - Boolean - True if operating system is Windows Vista or
newer

  bWin2000 - Output - Boolean - True if operating system is Windows 2000

  bWin2000Pro - Output - Boolean - True if operating system is Windows 2000
Professional

  bWin2000Workstation - Output - Boolean - True if operating system is Windows 2000
Workstation

  bWin2000Server - Output - Boolean - True if operating system is Windows 2000 Server

  bWin2000DatacenterSvr - Output - Boolean - True if operating system is Windows 2000
Datacenter Server

  bWin2000AdvancedSvr - Output - Boolean - True if operating system is Windows 2000
Advanced Server

  bWinXP - Output - Boolean - True if operating system is Windows XP

  bWinXPSP2 - Output - Boolean - True if operating system is Windows XP with SP2

  bWinXPHomeEdition - Output - Boolean - True if operating system is Windows XP Home
Edition

  bWinXPPro - Output - Boolean - True if operating system is Windows XP Professional

  bWinXPMediaCenter - Output - Boolean - True if operating system is Windows XP Media
```

Center

bWinXPStarter - Output - Boolean - True if operating system is Windows XP Starter OS

bWinXPTabletPC - Output - Boolean - True if operating system is Windows XP Tablet PC

bWinXPEmbedded - Output - Boolean - True if operating system is Windows XP Embedded

bWinVista - Output - Boolean - True if operating system is Windows Vista

bWinVistaSP1 - Output - Boolean - True if operating system is Windows Vista with Service Pack 1

bWinVistaHomeBasic - Output - Boolean - True if operating system is Windows Vista Home Basic

bWinVistaHomeEdition - Output - Boolean - True if operating system is Windows Vista Home Edition

bWinVistaHomePremium - Output - Boolean - True if operating system is Windows Vista Home Premium

bWinVistaHomeServer - Output - Boolean - True if operating system is Windows Vista Home Server

bWinVistaUltimate - Output - Boolean - True if operating system is Windows Vista Ultimate

bWinVistaBusiness - Output - Boolean - True if operating system is Windows Vista Business

bWinVistaEnterprise - Output - Boolean - True if operating system is Windows Vista Enterprise

bWinVistaWorkstation - Output - Boolean - True if operating system is Windows Vista Workstation

bWinVistaStarter - Output - Boolean - True if operating system is Windows Vista Starter edition

bWindows7 - Output - Boolean - True if operating system is Windows 7

bWin2003 - Output - Boolean - True if operating system is Windows 2003

bWin2003Server - Output - Boolean - True if operating system is Windows 2003 Server

bWin2003ServerR2 - Output - Boolean - True if operating system is Windows 2003 Server Rel 2

bWin2003StorageServer - Output - Boolean - True if operating system is Windows 2003 Storage Server

bBladeServer - Output - Boolean - True if this PC is a Blade Server

bWebServer - Output - Boolean - True if this PC is a Web Server

bWinHomeServer - Output - Boolean - True if this PC is a Home Server

bClusterServer - Output - Boolean - True if this PC is a Cluster Server

bComputeClusterServer - Output - Boolean - True if this PC is a Compute Cluster Server

bWinServer2008 - Output - Boolean - True if this PC is a Windows Server 2008

bWinServer2008R2 - Output - Boolean - True if this PC is a Windows Server 2008 R2

bDataCenterServer - Output - Boolean - True if operating system is Windows XP Datacenter

bDataCenterServerCore - Output - Boolean - True if operating system is Windows XP Datacenter Core

bBackOfficeServer - Output - Boolean - True if this PC is a Backoffice Server

bDomainController - Output - Boolean - True if this PC is a Domain Controller

bEnterpriseServer - Output - Boolean - True if this PC is a Enterprise Server

bEnterpriseServerCore - Output - Boolean - True if this PC is a Enterprise Server Core

bTerminalServer - Output - Boolean - True if this PC is a Terminal Server

bSmallBusinessServer - Output - Boolean - True if this PC is a Small Business Server

bSmallBusinessServerPremium - Output - Boolean - True if this PC is a Small Business Server Premium

bSmallBusinessRestrictedServer - Output - Boolean - True if this PC is a Small Business Restricted Server

bStandardServer - Output - Boolean - True if this PC is a Standard Server

bStandardServerCore - Output - Boolean - True if this PC is a Standard Server Core

bWinVista64 - Output - Boolean - True if operating system is Windows Vista 64-bit

bWinXPPro64 - Output - Boolean - True if operating system is Windows XP 64-bit Professional

bDatacenterItanium64 - Output - Boolean - True if operating system is Windows XP 64-bit Datacenter Itanium

bEnterpriseItanium64 - Output - Boolean - True if operating system is Windows XP 64-bit Enterprise Itanium

bDatacenter64 - Output - Boolean - True if operating system is Windows XP 64-bit Datacenter

bEnterprise64 - Output - Boolean - True if operating system is Windows XP 64-bit Enterprise

bStandard64 - Output - Boolean - True if operating system is Windows XP 64-bit Standard

bComputeServer64 - Output - Boolean - True if operating system is Windows Compute Server 64-bit

bDatacenterServer64 - Output - Boolean - True if operating system is Windows 2000 Datacenter Server 64-bit

bEnterpriseServer64 - Output - Boolean - True if operating system is Windows Enterprise Server 64-bit

bWebBladeServer64 - Output - Boolean - True if operating system is Windows Blade

Server 64-bit

  bStandardServer64 - Output - Boolean - True if operating system is Windows Standard Server 64-bit

  bOperSystem64 - Output - Boolean - True if operating system is Windows 64-bit

```
================================================================================
Available in cMath32 (cls32BitMath)
================================================================================

' ************************************************************************
' Enumerations
' ************************************************************************
  Private Enum enumBITS
      eBits4  ' 0
      eBits8  ' 1
  End Enum

  Public Enum enumDataType
      eLong   ' 0
      eShort  ' 1
      eByte   ' 2
  End Enum

' ************************************************************************
' ****                       Properties                               ****
' ************************************************************************
  StopProcessing - Input/Output - Boolean - True if user wants to stop
                   processing.  Sets the value of a global variable.

' ************************************************************************
' ****                       Methods                                  ****
' ************************************************************************

' Convert a 32-bit binary value to its hex equivalent.
Function BinaryToHex(ByVal strBinary As String) As String

' Converts a 32-bit binary string to a long integer.
Function BinaryToNumber(ByVal strBinary As String, _
               Optional ByVal lngDataType As enumDataType = eLong) As Long

' Converts a long integer to a 32-bit binary string.
Function NumberToBinary(ByVal lngValue As Long, _
               Optional ByVal lngDataType As enumDataType = eLong) As String

' Transfers contents of one byte array to another
Function ByteArrayToByteArray(ByRef abytData() As Byte) As Byte()

' Convert data from a byte array into a long integer. This routine
' assumes that the byte array will have at least four elements.
Function ByteArrayToLong(ByRef abytData() As Byte, _
                Optional ByVal lngIdx As Long = 0) As Long

' Convert data from a byte array into a long integer array.  This routine
' assumes that the byte array will have at least four elements.
Function ByteArrayToLongArray(ByRef abytData() As Byte, _
                           Optional ByVal lngPointer As Long = 0, _
                           Optional ByVal lngReturnSize As Long = 1) As Long()

' Return a string representing the value in string format to requested number
' of decimal positions.
Function DisplayNumber(ByVal dblCapacity As Double, _
              Optional ByVal lngDecimals As Long = 1) As String

' Converts Convert a double precision number to a long integer.
Function DoubleToLong(ByVal dblValue As Double) As Long

' This function creates a new array from a selected section of an array.
' The original array is unaffected.  This function only works with an
```

```
' array of long integers.
Function ExtractFromLongArray(ByRef alngSource() As Long, _
                              ByRef alngTarget() As Long, _
                      Optional ByVal lngStart As Long = 0, _
                      Optional ByVal lngCount As Long = -1, _
                      Optional ByVal blnExclude As Boolean = False) As Boolean

' Get the complement (inverse) of a byte (0-255)
Function GetComplement(ByVal bytData as Byte) As Byte

' Get the low and high word of a long integer.
Function GetHiLoWord(ByVal lngValue As Long, _
                      ByRef LOWORD As Long, _
                      ByRef HIWORD As Long) As Boolean

' Obtain the low and high byte of an integer.
Function GetHiLoByte(ByVal intValue As Integer, _
                      ByRef LOBYTE As Integer, _
                      ByRef HIBYTE As Integer) As Boolean

' Convert a hex value to its 32-bit binary equivalent
Function HexToBinary(ByVal strHex As String, _
            Optional ByVal lngDataType As enumDataType = eLong) As String

' Convert a Hex string to a byte array
Function HexToByteArray(ByVal strHex As String) As Byte()

' Convert a Hex string to a long integer
Function HexToLong(ByVal strHex As String) As Long

' Convert a hex string, stored in a byte array, into a normal string
' of data, also stored in a byte array.  One character per byte.
Sub HexArrayToByteArray(ByRef abytData() As Byte)

' Convert a normal string, stored in a byte array, into a hex array.
' Separate hex characters prior to making a byte value.
Function ByteArrayToHex1CharArray(ByRef abytData() As Byte) As Byte()

' Convert a normal string, stored in a byte array, into a hex array.
' Two hex characters are converted to asingle byte.
Function ByteArrayToHex2CharArray(ByRef abytData() As Byte) As String()

' Parses a string of data to determine if it is in hex format.
Function IsHexData(ByVal strData As String) As Boolean

' Parses a string of data to determine if it is in binary format.
Function IsBinaryData(ByVal strData As String) As Boolean

' Determine if an array has been properly initialized.
' Syntax:    If CBool(IsArrayInitialized(array_being_tested)) Then ...
Function IsArrayInitialized(ByVal avntData As Variant) As Long

' Convert a Long array to a byte array.
Function LongArrayToByteArray(ByRef alngData() As Long) As Byte()

' Convert a Long array to a string.
Function LongArrayToString(ByRef alngData() As Long) As String

' Convert a Long integer to a byte array.
Function LongToByteArray(ByVal lngValue As Long) As Byte()

' Converts a long integer to a double precision number.
Function LongToDouble(ByVal lngValue As Long) As Double
```

```
' Converts a long integer to a hex string.
Function LongToHex(ByVal lngValue As Long) As String

' Returns the reversed hexadecimal representation of a
' specified Long (4 bytes = 8 hex chars, most significant
' byte right).  Zeroes are right-padded.
Function LongToHexRev(ByVal lngValue As Long) As String

' Convert an integer to an unsigned long integer.
Function IntegerToUnsigned(intValue As Integer) As Long

' Converts a double precision value to an integer.
Function UnsignedToInteger(ByVal lngValue As Long) As Integer

' Convert a long integer to a double.
Function LongToUnsigned(lngValue As Long) As Double

' This routine will work out the higher 32 bits. This code looks like
' it could be done with a simple division, but you have the problem of
' the IDE using longs.
Sub CurrencyToLongs(ByVal curValue As Currency, _
                    ByRef lngLowOrder As Long, _
                    ByRef lngHighOrder As Long)

' This routine will convert two Long values into one Currency value.
Function LongsToCurrency(ByVal lngLowOrder As Long, _
                         ByVal lngHighOrder As Long) As Currency

' Function takes a Double precision value and returns a long integer.
Function UnsignedToLong(ByVal dblValue As Double) As Long

' Combines two one-byte values to one 2-byte Word (aka Integer).
Function MakeWord(ByVal LoByte As Byte, _
                  ByVal HiByte As Byte) As Integer

' Creates a Long Integer value from two Integers.
Function MakeDWord(ByVal LOWORD As Long, _
                   ByVal HIWORD As Long) As Long

' Copy data from its memory location, by pointer, and place into
' a byte array.
Sub PutWord(ByVal lngValue As Long, _
            ByRef abytCrypt() As Byte, _
   Optional ByVal lngPointer As Long = 0)

' Converts a string of data to a binary string.
Function StringToBinary(ByVal strData As String) As String

' Convert a normal string, stored in a byte array, into a hex string
' of data, also stored in a byte array.
Function StringToHex(ByVal lngReturnLength As Long, _
                     ByVal strIncomingData As String, _
             Optional ByVal blnReturnString As Boolean = True) As Variant

' Converts a byte array to string data.
Function ByteArrayToString(ByRef abytData() As Byte) As String

' Converts string data to a byte array.
Function StringToByteArray(ByVal strData As String) As Byte()

' The message is converted from a hex string to a byte array.
Function HexStringToByteArray(ByVal strData As String) As Byte()
```

```
' Converts hex data from Big-Endian to Little-Endian or Little-Endian
' to Big_Endian format. Used with certain hash algorithms.
Function SwapEndianHex(ByVal strHex As String, _
             Optional ByVal lngRetLength As Long = 16) As String

' Returns a Long with reversed byte order.
Function SwapEndianLong(ByVal lngValue As Long) As Long

' Swap data with each other.
Sub SwapData(ByRef vntData1 As Variant, _
             ByRef vntData2 As Variant)

' Swap data (byte, integer, or long) with each other without using a temp.
' Using math to do this takes a substantial amount more processing than Logic
' Gates. The logic gates foundation is in all processors. Which you could
' argue that so is Math, but running a math process with a temp variable
' in fact uses a considerable amount more processor cycles than logic gates.
'
' Using 3 Xor's over 3 Basic Equations greatly speeds up the application.
' Especially when dealing with millions of equations/Logic Gates.
Sub SwapLong(ByRef AA As Long, _
             ByRef BB As Long)

Sub SwapInt(ByRef AA As Integer, _
            ByRef BB As Integer)

Sub SwapBytes(ByRef AA As Byte, _
              ByRef BB As Byte)

' Function to add two unsigned numbers together as in C.  Overflows are ignored.
Function UnsignedAdd(ByVal lngValue1 As Long, _
                     ByVal lngValue2 As Long) As Long

' Function to divide two unsigned numbers together as in C.  Overflows are ignored.
Function UnsignedDivide(ByVal lngValue1 As Long, _
                        ByVal lngValue2 As Long) As Long

' Divides a double value by a (signed) Long divisor, treated as unsigned long,
' and returns the result as a Double of long integer value.
Function UnsignedDivideDbl(ByVal dblDividend As Double, _
                           ByVal lngDivisor As Long) As Double

' Function to multiply two unsigned numbers together as in C.  Overflows are ignored.
Function UnsignedMultiply(ByVal lngValue1 As Long, _
                          ByVal lngValue2 As Long) As Long

' Function to subtract two unsigned numbers together as in C.  Overflows are ignored.
Function UnsignedSubtract(ByVal lngValue1 As Long, _
                          ByVal lngValue2 As Long) As Long

' Shifts the bits to the right or left the specified number of positions and
' returns the new value.  Bits "falling off" the edge do not wrap around on
' the opposite side.  Some common languages like C/C++ or Java have an operator
' for this job:  ">>" or "<<".
Function w8Shift(ByVal bytValue As Byte, _
                 ByVal lngBitShift As Long) As Byte

' Shifts the bits to the right or left the specified number of positions and
' returns the new value.  Bits "falling off" the edge will wrap around on the
' opposite side.  Some common languages like C/C++ or Java have an operator
' for this job:  ">>>" or "<<<".
Function w8Rotate(ByVal bytValue As Byte, _
```

```
                  ByVal lngBitShift As Long) As Byte

' Shifts the bits of a short integer value either right or left the specified
' number of positions and returns the new value.  Bits "falling off" the edge
' do not wrap around.  Fill bits on the opposite side are zeroes.  Some common
' languages like C/C++ or Java have an operator:  ">>" or "<<".
Function w16Shift(ByVal lngValue As Long, _
                  ByVal intBitShift As Integer) As Long

' Performs a circular shift of a short integer value either right or left the
' specified number of positions and returns the new value.  Bits "falling off"
' the edge wrap around and are attached to the opposite side.  Some common
' languages like C/C++ or Java have an operator:  ">>>" or "<<<".
Function w16Rotate(ByVal lngValue As Long, _
                   ByVal intBitShift As Integer) As Long

' Shifts the bits of a long integer value either right or left the specified
' number of positions and returns the new value.  Bits "falling off" the edge
' do not wrap around.  Fill bits on the opposite side are zeroes.  Some common
' languages like C/C++ or Java have an operator:  ">>" or "<<".
Function w32Shift(ByVal lngValue As Long, _
                  ByVal intBitShift As Integer) As Long

' Performs a circular shift of a long integer value either right or left the
' specified number of positions and returns the new value.  Bits "falling off"
' the edge wrap around and are attached to the opposite side.  Some common
' languages like C/C++ or Java have an operator:  ">>>" or "<<<".
Function w32Rotate(ByVal lngValue As Long, _
                   ByVal intBitShift As Integer) As Long

' Compute cube root of (vntInput), by iteration, to up to 29
' digits.  The iteration continues until the limit of precision is reached.
Function CubeRoot(ByVal vntInput As Variant) As Variant

' Compute square root of (vntInput), by iteration, to up to 29
' digits.  The iteration continues until the limit of precision is reached.
Function SquareRoot(ByVal vntInput As Variant) As Variant

' The ExamineBit function will return True or False depending on the value
' of the nth bit (lngBitPosition ) of a long integer (lngValue). The sign
' bit is not used because if no other bits have been set to "1" then the
' sign bit would be ignored.
Function ExamineBit(ByVal lngValue As Long, _
                    ByVal lngBitPosition As Long) As Boolean

' The ClearBit Sub will change the state of the nth bit (lngBitPosition) of
' a long integer (lngValue).  The sign bit is not used because if no other
' bits have been set to "1" then the sign bit would be ignored.
Sub ClearBit(ByRef lngValue As Long, _
             ByVal lngBitPosition As Long)

' The SetBit Sub will set the nth bit (lngBitPosition) of a long integer
' (lngValue).  The sign bit is not used because if no other bits have been
' set to "1" then the sign bit would be ignored.
Sub SetBit(ByRef lngValue As Long, _
           ByVal lngBitPosition As Long)

' The ToggleBit Sub will change the state of the nth bit (lngBitPosition) of
' a long integer (lngValue).  The sign bit is not used because if no other
' bits have been set to "1" then the sign bit would be ignored.
Sub ToggleBit(ByRef lngValue As Long, _
              ByVal lngBitPosition As Long)
```

```
' This is where the prime number array is filled with long integers only.
Function GetPrimeNumbers(Optional ByVal lngQtyReq As Long = 1) As Long()

' Determines whether a long integer number is a prime.
Function IsPrime(ByVal lngNumber As Long) As Boolean
```

```
================================================================================
Available in cMath64 (cls64BitMath)
================================================================================

' ************************************************************************
' Enumerations
' ************************************************************************
  Public Enum enumHEX_REPRESENTATION
      eCubeRoots    ' 0
      eSquareRoots  ' 1
  End Enum

  Public Enum enumHEX_RETURN_FORMAT
      e16Chars    ' 0
      eLeft8      ' 1
      eRight8     ' 2
  End Enum

' ************************************************************************
' ****                       Properties                               ****
' ************************************************************************
    StopProcessing - Input/Output - Boolean - True if user wants to stop processing.
                  Sets the value of a global variable.

' ************************************************************************
' ****                       Methods                                  ****
' ************************************************************************
' Convert a 16 character hex string to a big number in string format.
Function w64HexToNumber(ByVal strHex As String) As string

' Convert a 16 character hex string to a binary string.
Function w64HexToBinary(ByVal strHex As String) As string

' Convert a big number in string format to a hex string.
Function w64NumberToHex(ByVal strNumber As String) As string

' Convert a big number in string format to a binary string.
Function w64NumberToBinary(ByVal strNumber As String) As string

' Convert a binary string format to a hex string.
Function w64BinaryToHex(ByVal strBinary As String) As string

' Convert a binary string format to a big number in string format.
Function w64BinaryToNumber(ByVal strBinary As String) As string

' Perform addition on two 16 char hex strings.
Function w64HexAdd(ByVal strHex1 As String, _
                   ByVal strHex2 As String) As String

' Perform subtraction on two 16 char hex strings.
Function w64HexSubtract(ByVal strHex1 As String, _
                        ByVal strHex2 As String) As String

' Perform multiplication on two 16 char hex strings.
Function w64HexMultiply(ByVal strHex1 As String, _
                        ByVal strHex2 As String) As String

' Perform division on two 16 char hex strings.
Function w64HexDivide(ByVal strHex1 As String, _
                      ByVal strHex2 As String) As String

' Perform addition on two big numbers in string format.
Function BigAdd(ByVal strNumber1 As String, _
```

```
                ByVal strNumber2 As String) As String

' Perform subtraction on two big numbers in string format.
Function BigSubtract(ByVal strNumber1 As String, _
                     ByVal strNumber2 As String) As String

' Perform multiplication on two big numbers in string format.
Function BigMultiply(ByVal strNumber1 As String, _
                     ByVal strNumber2 As String) As String

' Perform division on two big numbers in string format.
Function BigDivide(ByVal strNumber1 As String, _
                   ByVal strNumber2 As String) As String

' Parses a string of data to determine if it is in binary format.
Function IsBinaryData(ByVal strData As String) As Boolean

' Parses a string of data to determine if it is in hex format.
Function IsHexData(ByVal strData As String) As Boolean

' This is where the hex representation of an array of numbers is
' calculated and returned in a string array.  If parameter lngNumber
' is greater than zero then only a single value is processed.  If the
' parameter lngNumber equal zero then an array of prime numbers are
' determined in sequence and their hex representation is calculated
' accordingly.
'
' These types of hex values are generally used in the work and
' constant arrays for the SHA2 family of hash algorithms.
Function HexRepresentation(ByVal lngQtyReq As Long, _
                  Optional ByVal lngRetFmt As enumRETURN_FORMAT = e16Chars, _
                  Optional ByVal lngHexRep As enumHEX_REPRESENTATION = enuCubeRoots, _
                  Optional ByVal lngNumber As Long = 0) As String()

' Determine if an array has been properly initialized.
' Syntax:    If CBool(IsArrayInitialized(array_being_tested)) Then ...
Function IsArrayInitialized(ByVal avntData As Variant) As Long

' Shifts the bits to the right/Left the specified number of positions and
' returns the new value.  Bits "falling off" the direction of the shift
' do not wrap around.  Fill bits coming in from the opposite side are zeros.
' Some common languages like C/C++ or Java have an operator for this job:
' ">>" or "<<"
Function w64Shift(ByVal strHexData As String, _
                  ByVal intShiftCount As Integer, _
         Optional ByVal blnShiftLeft As Boolean = True, _
         Optional ByVal blnReturnAsHex As Boolean = True) As String

' Shifts the bits to the right/Left the specified number of positions and
' returns the new value.  Bits falling off the direction of the shift wrap
' around to the opposite end. Some common languages like C/C++ or Java have
' an operator for this job:   ">>>" or "<<<"
Function w64Rotate(ByVal strHexData As String, _
                   ByVal lngShiftCount As Long, _
          Optional ByVal blnRotateLeft As Boolean = True, _
          Optional ByVal blnReturnAsHex As Boolean = True) As String

' Perform a NOT bit comparison on a 16 character hex string. The NOT expression
' used for bit comparison does not exist in Visual Basic.  This is the same as
' performing the Compliment of a number.
Function w64Hex_NOT(ByVal strHex As String) As String

' Perform an AND bit comparison between two 16 character hex strings. The AND
```

```
' operator also performs a bitwise comparison of identically positioned bits
' in two numeric expressions and sets the corresponding bit.
Function w64Hex_AND(ByVal strHex1 As String, _
                    ByVal strHex2 As String) As String

' Perform an XoR bit comparison between two 16 character hex strings. The XoR
'  operator performs as both a logical and bitwise operator. A bit-wise
' comparison of two expressions using exclusive-or logic to form the result.
Function w64Hex_XoR(ByVal strHex1 As String, _
                    ByVal strHex2 As String) As String

' Perform an OR bit comparison between two 16 character hex strings. The OR
' operator also performs a bitwise comparison of identically positioned bits
' in two numeric expressions and sets the corresponding bit.
Function w64Hex_OR(ByVal strHex1 As String, _
                   ByVal strHex2 As String) As String

' Converts hex data from Big-Endian to Little-Endian or Little-Endian
' to Big_Endian format. Used with certain hash algorithms.
Function SwapEndianHex(ByVal strHex As String, _
              Optional ByVal lngRetLength As Long = 16) As String

' This is where the prime number array is filled with long integers only.
Function GetPrimeNumbers(Optional ByVal lngQtyReq As Long = 1) As Long()

' Determines whether a long integer number is a prime.
Function IsPrime(ByVal lngNumber As Long) As Boolean
```

```
================================================================================
Available in cSort (clsSort)
================================================================================

' ************************************************************************
' Enumerations
' ************************************************************************
  Public Enum enumSortMethod
      eShellSort           ' 0
      eCombSort            ' 1
      eQuickSort           ' 2
  End Enum

  Public Enum enumSortDirection
      eSort_Ascending      ' 0
      eSort_Descending     ' 1
  End Enum

  Public Enum enumSortTypeOfData
      eSort_Numeric        ' 0 - 12345
      eSort_String         ' 1 - "abc"
      eSort_Dates          ' 2 - "3/18/2006  6:18:46 AM"
  End Enum

  Public Enum enumSortCompare
      eSort_CaseSensitive  ' 0 - Exact byte match
      eSort_IgnoreCase     ' 1 - Uppercase/Lowercase considered same
  End Enum

' ************************************************************************
' ****                        Properties                              ****
' ************************************************************************
  StopProcessing - Input/Output - Boolean - True if user wants to stop processing.
                   Sets the value of a global variable.

  SortDirection - Input - Boolean - True = Ascending order (Default)
                                    False = Descending order
                   Based on enumSortDirection

  SortMethod - Input - Long Int - Sort algorithm to be used.
                   Based on enumSortMethod

  CompareMethod - Input - Long Int - Designates type of data comparison to
                   be used.  Based on enumSortCompare

  TypeOfData - Input - Long Int - Designates type of data to be sorted.
                   Based on enumSortTypeOfData

  DateFormat - Input - Long Int - Designates date format to be processed.
                   Based on enumDateFormat.  This is ignored unless
                   TypeOfData = eCDT_Dates.

  TimeFormat - Input - Long Int - Designates time format to be processed.
                   Based on enumTimeFormat.  This is ignored unless
                   TypeOfData = eCDT_Dates.

  ProcessTime - Input - Boolean - Designates if time is part of the input
                   data string to be evaluated and sorted.  This is ignored
                   unless TypeOfData = eCDT_Dates.
                       True - Time is part of the input data (Default)
                       False - Time is not part of input data

' ************************************************************************
```

```
' ****                        Methods                                      ****
' ****************************************************************************
' Various sort routines using the three basic types of data (numeric,
' string, dates).  The sorts include ShellSort, CombSort
' and QuickSort.  This is the main routine to call.  My main concern
' is sorting the indices and using them to correctly identify the
' original data.
Sub SortData(ByRef astrData() As String, _
    Optional ByRef strElapsedTime As String)

' The data must be SORTED.  This routine removes all duplicates based
' on user selection of case sensitivity or not.  The number of duplicates
' removed are also returned.
Function RemoveDupes(ByRef avntData As Variant, _
            Optional ByRef lngDupeCnt As Long = 0) As Boolean
```

```
================================================================================
Available in cConvertDateTime (clsConvertDateTime)
================================================================================

' ************************************************************************
' Enumerations
' ************************************************************************
  Public Enum enumDateFormat
      eCDT_Date_0    ' MMM dd, yyyy
      eCDT_Date_1    ' MMM d, yyyy
      eCDT_Date_2    ' MMMM dd, yyyy
      eCDT_Date_3    ' MMMM d, yyyy

      eCDT_Date_4    ' dd-MMM-yyyy
      eCDT_Date_5    ' d-MMM-yyyy
      eCDT_Date_6    ' dd MMM yyyy
      eCDT_Date_7    ' d MMM yyyy
      eCDT_Date_8    ' dd.MMM.yyyy
      eCDT_Date_9    ' d.MMM.yyyy

      eCDT_Date_10   ' yyyy-MMM-dd
      eCDT_Date_11   ' yyyy-MMM-d
      eCDT_Date_12   ' yyyy MMM dd
      eCDT_Date_13   ' yyyy MMM d
      eCDT_Date_14   ' yyyy.MMM.dd
      eCDT_Date_15   ' yyyy.MMM.d

      eCDT_Date_16   ' mm/dd/yyyy
      eCDT_Date_17   ' m/d/yyyy
      eCDT_Date_18   ' mm-dd-yyyy
      eCDT_Date_19   ' m-d-yyyy
      eCDT_Date_20   ' mm.dd.yyyy
      eCDT_Date_21   ' m.d.yyyy

      eCDT_Date_22   ' dd/mm/yyyy
      eCDT_Date_23   ' d/m/yyyy
      eCDT_Date_24   ' dd-mm-yyyy
      eCDT_Date_25   ' d-m-yyyy
      eCDT_Date_26   ' dd.mm.yyyy
      eCDT_Date_27   ' d.m.yyyy

      eCDT_Date_28   ' yyyy/mm/dd
      eCDT_Date_29   ' yyyy/m/d
      eCDT_Date_30   ' yyyy-mm-dd
      eCDT_Date_31   ' yyyy-m-d
      eCDT_Date_32   ' yyyy.mm.dd
      eCDT_Date_33   ' yyyy.m.d

      eCDT_Date_34   ' yyyy/dd/mm
      eCDT_Date_35   ' yyyy/d/m
      eCDT_Date_36   ' yyyy-dd-mm
      eCDT_Date_37   ' yyyy-d-m
      eCDT_Date_38   ' yyyy.dd.mm
      eCDT_Date_39   ' yyyy.d.m
  End Enum

  Public Enum enumTimeFormat
      eCDT_Time_0    ' h:nn
      eCDT_Time_1    ' hh:nn
      eCDT_Time_2    ' hh:nn:ss
      eCDT_Time_3    ' hh:nn:ss AM/PM
      eCDT_Time_4    ' h:nna/p
      eCDT_Time_5    ' hh:nnam/pm
```

```
      eCDT_Time_6    ' hh:nn:ss A.M./P.M.
      eCDT_Time_7    ' hh:nn:ss.ttt
      eCDT_Time_8    ' hh:nn:ss:ttt
      eCDT_Time_9    ' hh:nn:ss.ttt AM/PM
      eCDT_Time_10   ' hh:nn:ss:ttt AM/PM
      eCDT_Time_11   ' hh:nn:ss.ttt A.M./P.M.
      eCDT_Time_12   ' hh:nn:ss:ttt A.M./P.M.

      eCDT_Time_13   ' h.nn
      eCDT_Time_14   ' hh.nn
      eCDT_Time_15   ' hh.nn.ss
      eCDT_Time_16   ' hh.nn.ss AM/PM
      eCDT_Time_17   ' h.nna/p
      eCDT_Time_18   ' hh.nnam/pm

      eCDT_Time_19   ' hh.nn.ss A.M./P.M.
      eCDT_Time_20   ' hh.nn.ss.ttt
      eCDT_Time_21   ' hh.nn.ss.ttt AM/PM
      eCDT_Time_22   ' hh.nn.ss.ttt A.M./P.M.

      eCDT_Time_23   ' hh:nn:ss.tttt
      eCDT_Time_24   ' hh:nn:ss:tttt
      eCDT_Time_25   ' hh.nn.ss.tttt
      eCDT_Time_26   ' hh:nn:ss.tttt AM/PM
      eCDT_Time_27   ' hh:nn:ss:tttt AM/PM
      eCDT_Time_28   ' hh.nn.ss.tttt AM/PM
      eCDT_Time_29   ' hh:nn:ss.tttt A.M./P.M.
      eCDT_Time_30   ' hh:nn:ss:tttt A.M./P.M.
      eCDT_Time_31   ' hh.nn.ss.tttt A.M./P.M.
  End Enum

' ************************************************************************
' ****                       Properties                               ****
' ************************************************************************

  ReturnNumeric - Input - Boolean - Designates if data is to be return
                  in numeric or string format.
                      True - Numeric
                      False - String

  DateFormat - Input - Long Int - Designates date format to be processed.
                  Based on enumDateFormat.  This is ignored unless
                  TypeOfData = eCDT_Dates.

  TimeFormat - Input - Long Int - Designates time format to be processed.
                  Based on enumTimeFormat.  This is ignored unless
                  TypeOfData = eCDT_Dates.

  ProcessTime - Input - Boolean - Designates if time is part of the input
                  data string to be evaluated and sorted.  This is ignored
                  unless TypeOfData = eCDT_Dates.
                      True - Time is part of the input data (Default)
                      False - Time is not part of input data

' ************************************************************************
' ****                       Methods                                  ****
' ************************************************************************
' Converts date and/or time into a numeric or string equivalent to be
' used for comparison within a sort routine.
Function ConvertDateTime(ByVal strInputData As String) As Variant
```

```
================================================================================
Available in cDates (clsDates)
================================================================================

' ************************************************************************
' Enumerations
' ************************************************************************
  Public Enum enumDayOfWeek
      eSystemDate ' 0
      eSunday     ' 1
      eMonday     ' 2
      eTuesday    ' 3
      eWednesday  ' 4
      eThursday   ' 5
      eFriday     ' 6
      eSaturday   ' 7
  End Enum

  Public Enum enumDateInterval
      eYears      ' 0
      eMonths     ' 1
      eDays       ' 2
      eHours      ' 3
      eMinutes    ' 4
      eSeconds    ' 5
  End Enum

  Public Enum enumYear
      e1800 = 1800
      e1900 = 1900
      e2000 = 2000   ' Default
      e2100 = 2100
  End Enum

' ************************************************************************
' ****                        Properties                              ****
' ************************************************************************
  StopProcessing - Input/Output - Boolean - True if user wants to stop processing.
                   Sets the value of a global variable.

  ShortDateFormat - Output only - String - Display format of date for this locale.
                Ex:  M/d/yyyy

  LongDateFormat - Output only - String - Display format of date for this locale.
                Ex:  dddd, MMMM dd, yyyy

  DateSeparator - Output only - String - Symbol used as date separator for this
locale.
                Ex:  /

  TimeFormat - Output only - String - Display format of time for this locale.
                Ex:  h:mm:ss AMPM

  DateSeparator - Output only - String - Symbol used as time separator for this
locale.
                Ex:  :


' ************************************************************************
' ****                        Methods                                 ****
' ************************************************************************
' Determine a future or previous date.
' intNumber - Number of iterations.  Positive number calculates a future date.
```

```
'             Negative number calculates a date in the past.
' lngDateInterval - Segment to use in time spanning. See enumDateInterval above.
Function CalcDate(ByVal datDate As Date, _
                  ByVal lngNumber As Long, _
         Optional ByVal lngDateInterval As enumDateInterval = eDays) As Date

' This procedure takes a normal date format (that is, 1/1/94) and converts it
' to the appropriate Julian date.
Function DateToJulian(ByVal datDate As Date) As String

' Most government agencies and contractors require the use of Julian dates.
' A Julian date starts with a two-digit year, and then counts the number of
' days from January 1.  this routine converts the Julian date to a formatted
' date.
Function JulianToDate(ByVal strJulianDate As String, _
             Optional ByVal lngYear As enumYear = e2000) As Date

' This procedure takes a Julian date (yyddd) and converts it to the appropriate
' serial date.
Function JulianToSerial(ByVal strJulianDate As String, _
               Optional ByVal lngYear As enumYear = e2000) As Long

' This procedure takes a serial date number and converts it to the appropriate
' Julian date and returns it as a string.
Function SerialToJulian(ByVal lngSerial As Long) As String

' This procedure takes a normal date format (that is, 1/1/94) and converts it
' to serial number.
Function DateToSerial(ByVal datDate As Date) As Long

' Most government agencies and contractors require the use of Julian dates.
' A Julian date starts with a two-digit year, and then counts the number of
' days from January 1.  This routine converts a serial date to a formatted
' date.
Function SerialToDate(ByVal lngSerial As Long, _
             Optional ByVal strDateFmt As String = "") As Date

' Calculate the number of days left in a year.
Function DaysLeftInYear(ByVal datDate As Date) As Long

' This procedure takes the time and returns the time in words.
' Works great for party invitations.
' Ex: 12:25:12 AM -> Twenty-five minutes past Midnight
Function TimeToWords(ByVal strTime As String) As String

' This procedure takes a normal date and returns the date in words.
' Works great for party invitations.
' Ex: 4/17/2008 -> Tuesday, Seventeenth day of April in the year Two Thousand Eight
Function DateToWords(ByVal datDate As Date) As String

' This procedure will return the number of days that have passed since
' January 1 of a given year.
Function DaysPassed(ByVal datDate As Date) As Long

' Determine how many days there are in a year.  Leap years have 366 days
' and all others have 365.
Function DaysInYear(ByVal datDate As Date) As Long

' Determine if a 4-digit year is a leap year.  Prior to the 1500's, there
' was no such thing as a leap year.
Function IsLeapYear(ByVal lngYear As Long) As Boolean

' Convert a week number to a specific date.
```

```
Function WeekNumToDate(ByVal lngWeekNbr As Long, _
                       ByVal lngYear As Long, _
              Optional ByVal bln2000 As Boolean = True, _
              Optional ByVal lngFirstDayOfTheWeek As enumDayOfWeek = eSunday) As Date

' This function check how many days there are between two certain dates.
Function DaysBetween(ByVal datStartDate As Date, _
                     ByVal datEndDate As Date, _
             Optional ByVal blnIncludeStartDay As Boolean = False, _
             Optional ByVal lngFirstDayOfTheWeek As enumDayOfWeek = eSunday) As Long

' Returns the number of days in a specified month
Function DaysInMonth(ByVal datDate As Date) As Long

' Returns the day of the week of a certain date in numeric format.
Function GetWeekday(ByVal datDate As Date, _
           Optional ByVal lngFirstDayOfTheWeek As enumDayOfWeek = eSunday) As Long

' Find last day of a month
Function LastDayOfMonth(ByVal datDate As Date) As Long

' Find last day of previous month
Function LastDayPrevMonth(ByVal datDate As Date) As Long

' Find the week of the year as a number (1 - 54).
Function GetWeekNumber(ByVal datDate As Date) As Long

' Get the name of the week day
' Ex:   GetWeekdayName(3, False, eSunday) will return Tuesday
Function GetWeekdayName(ByVal lngDayRequested As Long, _
               Optional ByVal blnAbbreviate As Boolean = True, _
               Optional ByVal lngFirstDayOfWeek As enumDayOfWeek = eSunday) As String

' Get name of the month
' Ex:   GetMonthName(2, False) will return February
Function GetMonthName(ByVal lngMonthRequested As Long, _
             Optional ByVal blnAbbreviate As Boolean = True) As String

' Determine day number of a specific month of a specific year.  Takes
' in consideration if this is a leap year or not.
' Ex:   Thanksgiving Day in USA (Fourth Thursday in November 2012)
'       Parameters:
'           lngNthDay = 4    4th
'           lngWeekday = 5   Thursday (Sunday first day of week)
'           lngMonth = 11    November
'           lngYear = 2012   Prefer 4-digit year
'
'       Returns:  NthDayOfMonth = 22
'
Function DayOfMonth(ByVal lngNthDay As Long, _
                    ByVal lngWeekday As Long, _
                    ByVal lngMonth As Long, _
                    ByVal lngYear As Long, _
           Optional ByVal lngFirstDayOfWeek As enumDayOfWeek = eSunday) As Long

' In Western Christianity, Easter always falls on a Sunday from March 22 to
' April 25 inclusive. The following day, Easter Monday, is a legal holiday
' in many countries with predominantly Christian traditions.
'
' Returns:  Full date.  Optional - Month and day (ex: "Apr 15")
Function EasterSunday(ByVal lngYear As Long, _
             Optional ByRef strMonthDay As String = vbNullString) As Date
```

```
' Determine if current locale is observing standard time, or is on fixed
' time or Daylight Savings time.
Function IsDaylightSavings() As Boolean

' Calculate current Coordinated Universal Time (UTC) for this locale.
' Coordinated Universal Time (UTC) is the primary time standard by which
' the world regulates clocks and time. It is one of several closely related
' successors to Greenwich Mean Time (GMT) which is also known as Zulu Time.
' Greenwich, England is longitude 0 and latitude 0 degrees.
Function CurrentUTC(Optional ByRef strHoursMinutes As String = vbNullString) As Date

' Calculate when Daylight Savings time begins for this locale.
Function DaylightSavingsBegins(ByVal lngYear As Long) As Date

' Calculate when Daylight Savings time ends for this locale.
Function DaylightSavingsEnds(ByVal lngYear As Long) As Date

' Format system date and time to be used in a log file.  If no format
' is passed, this routine will use the current system short date
' and time format.
Function GetTimeStamp(Optional ByVal strDateFmt As String = "", _
                      Optional ByVal strTimeFmt As String = "") As String

' The next Y2K event in programming is referred to as Y2K38. Ever
' heard the phrase "Unix time"?  That's the most often-used
' description of the method of marking time by the number of seconds
' that have passed since Jan 1, 1970. A question arose about this, and
' how negative values were starting to creep into VB calculations,
' ostensibly because of the way VB uses the high-bit to indicate sign.
' The Clock rolls over on 1/19/2038 at 3:14:07 AM, if the base date is
' the commonly used midnight on 1/1/1970.
'
' Converts a date to seconds.
Function Y2K38_DateToSeconds(ByVal datDate As Date) As Double

' The next Y2K event in programming is referred to as Y2K38. Ever
' heard the phrase "Unix time"?  That's the most often-used
' description of the method of marking time by the number of seconds
' that have passed since Jan 1, 1970. A question arose about this, and
' how negative values were starting to creep into VB calculations,
' ostensibly because of the way VB uses the high-bit to indicate sign.
' The Clock rolls over on 1/19/2038 at 3:14:07 AM, if the base date is
' the commonly used midnight on 1/1/1970.
'
' Converts seconds to a date.
Function Y2K38_SecondsToDate(ByVal dblNetDate As Double) As Date
```

```
================================================================================
Available in cFileDate (clsFileDate)
================================================================================

' ************************************************************************
' Enumerations
' ************************************************************************
  ' 01-Nov-2008 Reset order to match API SetFileTime()
  Public Enum enumDateProperties
      eCreateDate      ' 0
      eLastAccessed    ' 1
      eLastModified    ' 2
      eTwoDates        ' 3  Last Accessed, Last Modified (most common)
      eThreeDates      ' 4  Create Date, Last Accessed, Last Modified
  End Enum

' ************************************************************************
' ****                       Properties                               ****
' ************************************************************************
    StopProcessing - Input/Output - Boolean - True if user wants to stop processing

    SelectedDateField - Input only - Long Integer - Based on enumDateField values

    PathFileName - Input/Output - String - Full path and file name

    TimeStamp - Input/Output - Date - Date value of the time stamp (Created,
                Modified, last Accessed)

    LongDateFormat - Output only - String - Designates what format to use to
                display the date in long format in this users locale

    ShortDateFormat - Output only - String - Designates what format to use to
                display the date in short format in this users locale

    TimeFormat - Output only - String - Designates what format to use to
                display the time in this users locale

    DateSeparator - Output only - String - Designates which symbol to use to
                separate the short date display in this users locale

    TimeSeparator - Output only - String - Designates which symbol to use to
                separate the time display in this users locale

    CreateDate - Output only - Date - Folder or file creation date

    LastAccessed - Output only - Date - Folder or file last accessed date

    LastModified - Output only - Date - Folder or file last modified date

' ************************************************************************
' ****                       Methods                                  ****
' ************************************************************************
Format a date/time string based on the format desired by the user obtained from
the current system date/time settings.
Function SystemDateInfo(ByRef strDate As String, _
                        ByRef strTime As String, _
                Optional ByVal blnUseShortdate As Boolean = True) As Boolean

Format a date/time string based on the format desired by the user obtained from
the passed filename date/time settings.
Function FileDateInfo(ByRef strDate As String, _
                      ByRef strTime As String, _
              Optional ByVal blnUseShortdate As Boolean = True, _
```

```
            Optional ByVal lngDateField As enumDateProperties = eLastModified) As
Boolean
```

Change a folder or file date/time stamp based on date stored in property TimeStamp().

```
Function SetDateProperty() As Boolean
```

```
================================================================================
Available in cBigFiles (clsBigFiles)
================================================================================

' ************************************************************************
' ****                        Properties                              ****
' ************************************************************************
    StopProcessing - Input/Output - Boolean - True if user wants to stop processing


' ************************************************************************
' ****                        Methods                                 ****
' ************************************************************************
' Open a file to be used as input.  The file must already exist.
' If the file does not exist, an error will occur.
Function OpenReadOnly(ByVal strFileName As String, _
                      ByRef hFile As Long) As Boolean

' Open a file to update.  If the file exist, if will be opened.  If the
' file does not exist, it will be created.  Use carefully.  If you open
' an existing file and something goes wrong, the file may become a zero
' byte file.  There is no recovery of the data available. I use this to
' access a temporary work file only.
Function OpenReadWrite(ByVal strFileName As String, _
                       ByRef hFile As Long) As Boolean

' This routine is used to open a file as read only and calculate it's size.
Sub CalcFileSize(ByVal strFileName As String, _
                 ByRef curFilesize As Currency, _
         Optional ByRef strBitsInHex As String = "")

' This routine is used to read data from an opened file.
Function API_ReadFile(ByVal hFile As Long, _
                      ByVal curPosition As Currency, _
                      ByRef abytData() As Byte) As Boolean

' This routine is used to write data to the file.
Function API_WriteFile(ByVal hFile As Long, _
                       ByVal curPosition As Currency, _
                       ByRef abytData() As Byte) As Boolean

' Sets the pointer to the end of the file designating that we are now
' finished with this file.
Sub API_SetEndOfFile(ByVal hFile As Long, _
                     ByVal curPosition As Currency)

' Closes an open file.
Sub API_CloseFile(ByRef hFile As Long)

' Creates a file from 1 byte to greater than 2gb filled with null values.
' I have created files greater than 5gb without any problems.
Function CreateBigFile(ByVal strFileName As String, _
                       ByVal curFilesize As Currency) As Boolean

' Updates a file from 1 byte to greater than 2gb with null values.
Function LoadWithNullValues(ByVal hFile As Long, _
                            ByVal curFilesize As Currency) As Boolean
```

```
================================================================
License                                          Kenneth Ives
                                                 kenaso@tx.rr.com
================================================================
```

Preamble

This License governs Your use of the Work. This License is intended to allow developers to use the Source Code and Executable Files provided as part of the Work in any application in any form.

The main points subject to the terms of the License are:

- Source Code and Executable Files can be used in commercial applications;
- Source Code and Executable Files can be redistributed; and
- Source Code can be modified to create derivative works.

No claim of suitability, guarantee, or any warranty whatsoever is provided. The software is provided "as-is".

This License is entered between You, the individual or other entity reading or otherwise making use of the Work licensed pursuant to this License and the individual or other entity which offers the Work under the terms of this License ("Author").

License

THE WORK (AS DEFINED BELOW) IS PROVIDED UNDER THE TERMS OF THIS CODE PROJECT OPEN LICENSE ("LICENSE"). THE WORK IS PROTECTED BY COPYRIGHT AND/OR OTHER APPLICABLE LAW. ANY USE OF THE WORK OTHER THAN AS AUTHORIZED UNDER THIS LICENSE OR COPYRIGHT LAW IS PROHIBITED.

BY EXERCISING ANY RIGHTS TO THE WORK PROVIDED HEREIN, YOU ACCEPT AND AGREE TO BE BOUND BY THE TERMS OF THIS LICENSE. THE AUTHOR GRANTS YOU THE RIGHTS CONTAINED HEREIN IN CONSIDERATION OF YOUR ACCEPTANCE OF SUCH TERMS AND CONDITIONS. IF YOU DO NOT AGREE TO ACCEPT AND BE BOUND BY THE TERMS OF THIS LICENSE, YOU CANNOT MAKE ANY USE OF THE WORK.

Definitions.

"Articles" means, collectively, all articles written by Author which describes how the Source Code and Executable Files for the Work may be used by a user.

"Author" means the individual or entity that offers the Work under the terms of this License.

"Derivative Work" means a work based upon the Work or upon the Work and other pre-existing works.

"Executable Files" refer to the executables, binary files, configuration and any required data files included in the Work.

"Publisher" means the provider of the website, magazine, CD-ROM, DVD or other medium from or by which the Work is obtained by You.

"Source Code" refers to the collection of source code and configuration files used to create the Executable Files.

"Standard Version" refers to such a Work if it has not been modified, or has been modified in accordance with the consent of the Author, such consent being in the full discretion of the Author.

"Work" refers to the collection of files distributed by the Publisher, including the Source Code, Executable Files, binaries, data files, documentation, whitepapers and the Articles.

"You" is you, an individual or entity wishing to use the Work and exercise your rights under this License.

Fair Use/Fair Use Rights. Nothing in this License is intended to reduce, limit, or restrict any rights arising from fair use, fair dealing, first sale or other limitations on the exclusive rights of the copyright owner under copyright law or other applicable laws.

License Grant. Subject to the terms and conditions of this License, the Author hereby grants You a worldwide, royalty-free, non-exclusive, perpetual (for the duration of the applicable copyright) license to exercise the rights in the Work as stated below:

You may use the standard version of the Source Code or Executable Files in Your own applications.

You may apply bug fixes, portability fixes and other modifications obtained from the Public Domain or from the Author. A Work modified in such a way shall still be considered the standard version and will be subject to this License.

You may otherwise modify Your copy of this Work (excluding the Articles) in any way to create a Derivative Work, provided that You insert a prominent notice in each changed file stating how, when and where You changed that file.

You may distribute the standard version of the Executable Files and Source Code or Derivative Work in aggregate with other (possibly commercial) programs as part of a larger (possibly commercial) software distribution.

The Articles discussing the Work published in any form by the author may not be distributed or republished without the Author's consent. The author retains copyright to any such Articles. You may use the Executable Files and Source Code pursuant to this License but you may not repost or republish or otherwise distribute or make available the Articles, without the prior written consent of the Author.

Any subroutines or modules supplied by You and linked into the Source Code or Executable Files this Work shall not be considered part of this Work and will not be subject to the terms of this License.

Patent License. Subject to the terms and conditions of this License, each Author hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable (except as stated in this section) patent license to make, have made, use, import, and otherwise transfer the Work.

Restrictions. The license granted in Section 3 above is expressly

made subject to and limited by the following restrictions:

You agree not to remove any of the original copyright, patent, trademark, and attribution notices and associated disclaimers that may appear in the Source Code or Executable Files.

You agree not to advertise or in any way imply that this Work is a product of Your own.

The name of the Author may not be used to endorse or promote products derived from the Work without the prior written consent of the Author.

You agree not to sell, lease, or rent any part of the Work.

You may distribute the Executable Files and Source Code only under the terms of this License, and You must include a copy of, or the Uniform Resource Identifier for, this License with every copy of the Executable Files or Source Code You distribute and ensure that anyone receiving such Executable Files and Source Code agrees that the terms of this License apply to such Executable Files and/or Source Code. You may not offer or impose any terms on the Work that alter or restrict the terms of this License or the recipients' exercise of the rights granted hereunder. You may not sublicense the Work. You must keep intact all notices that refer to this License and to the disclaimer of warranties. You may not distribute the Executable Files or Source Code with any technological measures that control access or use of the Work in a manner inconsistent with the terms of this License.

You agree not to use the Work for illegal, immoral or improper purposes, or on pages containing illegal, immoral or improper material. The Work is subject to applicable export laws. You agree to comply with all such laws and regulations that may apply to the Work after Your receipt of the Work.

Representations, Warranties and Disclaimer. THIS WORK IS PROVIDED "AS IS", "WHERE IS" AND "AS AVAILABLE", WITHOUT ANY EXPRESS OR IMPLIED WARRANTIES OR CONDITIONS OR GUARANTEES. YOU, THE USER, ASSUME ALL RISK IN ITS USE, INCLUDING COPYRIGHT INFRINGEMENT, PATENT INFRINGEMENT, SUITABILITY, ETC. AUTHOR EXPRESSLY DISCLAIMS ALL EXPRESS, IMPLIED OR STATUTORY WARRANTIES OR CONDITIONS, INCLUDING WITHOUT LIMITATION, WARRANTIES OR CONDITIONS OF MERCHANTABILITY, MERCHANTABLE QUALITY OR FITNESS FOR A PARTICULAR PURPOSE, OR ANY WARRANTY OF TITLE OR NON-INFRINGEMENT, OR THAT THE WORK (OR ANY PORTION THEREOF) IS CORRECT, USEFUL, BUG-FREE OR FREE OF VIRUSES. YOU MUST PASS THIS DISCLAIMER ON WHENEVER YOU DISTRIBUTE THE WORK OR DERIVATIVE WORKS.

Indemnity. You agree to defend, indemnify and hold harmless the Author and the Publisher from and against any claims, suits, losses, damages, liabilities, costs, and expenses (including reasonable legal or attorneys' fees) resulting from or relating to any use of the Work by You.

Limitation on Liability. EXCEPT TO THE EXTENT REQUIRED BY APPLICABLE LAW, IN NO EVENT WILL THE AUTHOR OR THE PUBLISHER BE LIABLE TO YOU ON ANY LEGAL THEORY FOR ANY SPECIAL, INCIDENTAL, CONSEQUENTIAL, PUNITIVE OR EXEMPLARY DAMAGES ARISING OUT OF THIS LICENSE OR THE USE OF THE WORK OR OTHERWISE, EVEN IF THE AUTHOR OR THE PUBLISHER HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

Termination.

This License and the rights granted hereunder will terminate automatically upon any breach by You of any term of this License. Individualss or entities who have received Derivative Works from You under this License, however, will not have their licenses terminated provided such individuals or entities remain in full compliance with those licenses. Sections 1, 2, 6, 7, 8, 9, 10 and 11 will survive any termination of this License.

If You bring a copyright, trademark, patent or any other infringement claim against any contributor over infringements You claim are made by the Work, your License from such contributor to the Work ends automatically.

Subject to the above terms and conditions, this License is perpetual (for the duration of the applicable copyright in the Work). Notwithstanding the above, the Author reserves the right to release the Work under different license terms or to stop distributing the Work at any time; provided, however that any such election will not serve to withdraw this License (or any other license that has been, or is required to be, granted under the terms of this License), and this License will continue in full force and effect unless terminated as stated above.

Publisher. The parties hereby confirm that the Publisher shall not, under any circumstances, be responsible for and shall not have any liability in respect of the subject matter of this License. The Publisher makes no warranty whatsoever in connection with the Work and shall not be liable to You or any party on any legal theory for any damages whatsoever, including without limitation any general, special, incidental or consequential damages arising in connection to this license. The Publisher reserves the right to cease making the Work available to You at any time without notice

Miscellaneous.

This License shall be governed by the laws of the location of the head office of the Author or if the Author is an individual, the laws of location of the principal place of residence of the Author.

If any provision of this License is invalid or unenforceable under applicable law, it shall not affect the validity or enforceability of the remainder of the terms of this License, and without further action by the parties to this License, such provision shall be reformed to the minimum extent necessary to make such provision valid and enforceable.

No term or provision of this License shall be deemed waived and no breach consented to unless such waiver or consent shall be in writing and signed by the party to be charged with such waiver or consent.

This License constitutes the entire agreement between the parties with respect to the Work licensed herein. There are no understandings, agreements or representations with respect to the Work not specified herein. The Author shall not be bound by any additional provisions that may appear in any communication from You. This License may not be modified without the mutual written agreement of the Author and You.

Kenneth Ives kenaso@tx.rr.com