VBBACKUP.VBX for Visual Basic 3.0

RUW-Computer
Robchestraße 5
58540 Meinerzhagen, Germany
Telephone: 049-2358-7760
Fax: 049-2358-8709
Compuserve: 100116,1532

Contents:

1. Overview

The VBBACKUP.VBX is a Visual-Basic Custom-Control for saving and restoring data from your VB-Application. The VBX has a built in data-compression and error-check. You only have to add the VBX to your project-file and then you can use it in a very easy way. All you have to do then is to place it somewhere on your form, and make the following somewhere in your application:

- add files with the AddItem-Method
 (e.g. Backup1.AddItem "C:\WINDOWS\WIN.INI")

- start the backup with the Action-Property
 (e.g. Backup1.Action = 1)

When you want to restore files somewhere in your application, you start the restore
either by setting the action value to 2 (which will trigger a sequential restore) or to 3 (which will trigger the random restore).

The compression-rate of the built in data-compression depends on the type of files to save (Databases up to 30% of the original size, already compressed files not much).

2. Two ways of restoring data

The sequential restore (action property set to 2) starts restoring with the first file on the first disk. All the files of your backup-set will be restored sequentially (you can also skip a file when you set skip to True in the GetDestination-Event, see chapter 4 for more information).

The random restore (action property set to 3) at first wants the last disk of your backup-set. There the catalog of the backup-set is stored. After VBBACKUP has read this catalog the EditRestoreList-Event is triggered where you can process the restorelist and select the files you want to be restored

3. The Properties

Here is a list of all the properties of the VBBACKUP.VBX
Properties with an * are Standard-Properties, look in Visual-Basic Help for a description.

- **Caption ***
- **Index ***
- **Left ***
- **Name ***
- **Tag ***
- **Top ***

- **Action**
  - starts the backup or restore
  - integer
  - runtime read/write

  when writing, this property is used as a pseudo-method, where the following methods could be invoked :

  1 starts the backup
  2 starts a sequential restore
  3 starts a random restore
  4 starts a reading of the restorelist from disk

  when reading, this property indicates the state of the custom-control, where :

  0 indicates that nothing is running and the custom control is ready to start one of the above pseudo-methods
  1 indicates a backup is running
  2 indicates a sequential restore is running
  3 indicates a random restore is running
  4 indicates a reading of the restorelist from disk is running

  The following constants are defined in the RUWCONST.TXT file :

  VBBACKUP_ACTION_NONE 0
  VBBACKUP_ACTION_BACKUP 1
  VBBACKUP_ACTION_RESTORESEQ 2
  VBBACKUP_ACTION_RESTORERANDOM 3
  VBBACKUP_ACTION_READFILELIST 4

- **BackupStartPath**
  - contains the path (this means also the name) to which you want to backup the files (or from which you want to restore the files)
  - string
  - runtime read/write / designtime read/write

  When you backup to a floppy this path must be a root path, e.g. "A:\BACKUP"
  When you backup to a harddisk you can also enter a directory, e.g. "C:\TEMP\BACKUP"

  There should be no extension, because it is automatically set from the program.

- **BackupPath**
  - contains the full path of the backup (or restore)
  - string

- runtime read

  E.g., "A:\BACKUP.001", when backup to the first disk

- **BackupDrive**
  - contains the drive of the backup (or restore)
  - string
  - runtime read

  E.g., "A:", when backup to drive a:

- **BackupDir**
  - contains the directory of the backup (or restore)
  - string
  - runtime read

  E.g., "", when backup to drive a:

- **BackupName**
  - contains the name of the backup (or restore)
  - string
  - runtime read

  E.g., "BACKUP"

- **BackupExt**
  - contains the extension of the backup (or restore)
  - string
  - runtime read

  E.g., ".002", when disk two is the current disk

- **BackupDate**
  - contains the date of the backup (or restore)
  - string
  - runtime read

  E.g., "06/13/95" when language property is set to english or "13.06.95", when set to german

- **BackupTime**
  - contains the time of the backup (or restore)
  - string
  - runtime read

  E.g., "11:55:22"

- **BackupSize**
  - contains the original size of all backup-files
  - long integer
  - runtime read

- **BackupRead**
  - while backup is running it contains the total number of original bytes read so far from the source drive (normally the hard disk),
    while restore is running it contains the total number of compressed bytes read so far from source drive (normally the floppy-drive)
  - long integer
  - runtime read

  you can calculate the percentage of how much of the backup-work is done by the following:
  perc% = Backup1.BackupRead / Backup1.BackupSize * 100

- **BackupWritten**
  - while backup is running it contains the total number of compressed bytes written to the destination drive (normally the floppy-disk) so far, while restore is running it contains the total number of original bytes written to the destination drive normally the hard disk) so far
  - long integer
  - runtime read

  you can calculate the percentage of how much of the restore-work is done by the following:
  perc% = Backup1.BackupWritten / Backup1.BackupSize * 100

- **DefaultEvents**
  - enables/disables the use of default events
  - Boolean
  - runtime read/write / designtime read/write

  When set to False (default) no default events are processed and you must supply the messages in the events of the VBBACKUP.VBX by yourself. When set to True the VBX internally manages the events GetNextDisk, GetLastDisk, ErrorQuery, so that those events are not called anymore (Code, standing in those events will not be processed).

  Setting this property to True gives you the easiest way of realizing a backup/restore, because you only have to add the items with AddItem and start the show with the Action-Property.

- **DiskNumber**
  - contains the current backup disk number
  - integer
  - runtime read

- **FilePath**
  - contains the full path of the file currently processed from a backup (or restore)
  - string
  - runtime read

  E.g., "C:\WINDOWS\WIN.INI".
  It's always the original file name.

- **FileDrive**
  - contains drive of the file currently processed
  - string
  - runtime read

E.g., "D:", when backuping or restoring a file with the original path "D:\TEMP\TEST.DOC"

- **FileDir**
  - contains the directory of the file currently processed
  - string
  - runtime read

  E.g., "\TEMP\", when backuping or restoring a file with the original path "D:\TEMP\TEST.DOC"

- **FileName**
  - contains the name of the file currently processed
  - string
  - runtime read

  E.g., "TEST", when backuping or restoring a file with the original path "D:\TEMP\TEST.DOC"

- **FileExt**
  - contains the extension of the file currently processed
  - string
  - runtime read

  E.g., ".DOC", when backuping or restoring a file with the original path "D:\TEMP\TEST.DOC"

- **FileDate**
  - contains the date of the file currently processed
  - string
  - runtime read

  E.g., "06/13/95" when language property is set to english or "13.06.95", when set to german

- **FileTime**
  - contains the time of the file currently processed
  - string
  - runtime read

  E.g., "11:55:22"

- **FileAttr**
  - contains the Attribut of the file currently processed
  - integer
  - runtime read

- **FileSize**
  - contains the original size of the current file
  - long integer
  - runtime read

- **FileRead**
  - while backup is running it contains the number of original bytes read so far from the current file, while restore is running it contains the number of compressed bytes read so far from the source drive (normally the floppy-drive) which belong to the current file
  - long integer

- runtime read

you can calculate the percentage of how much of the current file is already backuped by the following:
perc% = Backup1.FileRead / Backup1.FileSize *100

- **FileWritten**
  - while backup is running it contains the number of compressed bytes of the current file written so far to the destination drive, while restore is running it contains the number of original bytes of the current file written so far to to the destination drive (normally the hard disk)
  - long integer
  - runtime read

you can calculate the percentage of how much of the current file is already restored by the following:
perc% = Backup1.FileWritten / Backup1.FileSize *100

- **FileNums**
  - contains the total number of files of a backup-set
  - integer
  - runtime read

normally the value of this property is identical to the FileListCount. Only when a sequential restore is running this property contains the total number of files of the backup-set where else FileListCount contains the number of files restored so far, because at a sequential restore the filelist is build dynamically while the restore progresses

- **FileListCount**
  - contains the number of files in the FileList
  - integer
  - runtime read

The number of files in the filelist is increased when a file is added with the AddItem method, before a backup is started or it is increased dynamically while a sequential restore is running. When filelist is read from disk, this property is set to the total number of files of the backup-set.

- **FileListIndex**
  - contains the index of the filelist-item currently accessed
  - integer
  - runtime read

The value of this property may range from 0 to FileListCount - 1. If no item is accessed, e.g., when you added files with AddItem, but havn't yet started the backup, the value is -1.

- **FileListPath(index)**
  - contains the path of the filelist-item specified by index
  - string
  - runtime read

Use this property to access the path of the files in the filelist. This is a property-array, therefor you must specify the index of the item you want to access :

[form.]backup1.FileListPath(index)

The index of the first item is 0 and the index of the last item is FileListCount - 1.

- **FileListDate(index)**
  - contains the date of the filelist-item specified by index
  - string
  - runtime read

  Use this property to access the date of the files in the filelist. This is a property-array, therefor you must specify the index of the item you want to access :

  [form.]backup1.FileListDate(index)

  The index of the first item is 0 and the index of the last item is FileListCount - 1.

- **FileListTime(index)**
  - contains the time of the filelist-item specified by index
  - string
  - runtime read

  Use this property to access the time of the files in the filelist. This is a property-array, therefor you must specify the index of the item you want to access :

  [form.]backup1.FileListTime(index)

  The index of the first item is 0 and the index of the last item is FileListCount - 1.

- **FileListSize(index)**
  - contains the size of the filelist-item specified by index
  - long integer
  - runtime read

  Use this property to access the size of the files in the filelist. This is a property-array, therefor you must specify the index of the item you want to access :

  [form.]backup1.FileListSize(index)

  The index of the first item is 0 and the index of the last item is FileListCount - 1.

- **FileListAttribut(index)**
  - contains the atrribut of the filelist-item specified by index
  - integer
  - runtime read

  Use this property to access the attribut of the files in the filelist. This is a property-array, therefor you must specify the index of the item you want to access :

  [form.]backup1.FileListAttribut(index)

  The index of the first item is 0 and the index of the last item is FileListCount - 1.

•FileListCompressedSize(index)

- •contains the compressed size of the filelist-item specified by index
- •long integer
- •runtime read

Use this property to access the compressed size of the files in the filelist. This is a property-array, therefor you must specify the index of the item you want to access:

[form.]backup1.FileListCompressedSize(index)

The index of the first item is 0 and the index of the last item is FileListCount - 1.

•FileListDiskNumber(index)

- •contains the disk number on which the filelist-item specified by index is stored
- •integer
- •runtime read

Use this property if you want to know on which disk of your backup-set the storing of the files in the filelist starts. This is a property-array, therefor you must specify the index of the item you want to access :

[form.]backup1.FileListDiskNumber(index)

The index of the first item is 0 and the index of the last item is FileListCount - 1.

•FileListRestore(index)

- • this flag indicates if an item of the filelist should be restored in a random restore or not
- •boolean
- •runtime read/write

Use this property to determine if a file should be restored or not. This property only has a function within the EditRestoreList event when a random restore is running. By default this flag is set true for all files in the restorelist. In the EditRestoreList event you can toggle this value for each file, e.g. by displaying a listbox and let the user select the files he want to restore.
This is a property-array, therefor you must specify the index of the item you want to access :

[form.]backup1.FileListRestore(index)

The index of the first item is 0 and the index of the last item is FileListCount - 1.

•FreeSpace

- •contains the number of free bytes on the backup disk
- •long integer
- •runtime read

•IgnoreTimestamp

- •sets the flag if the timestamp should be restored or not
- •boolean
- •runtime read/write / designtime read/write

when set to False (default), the timestamp of an restored file will be restored to its original

values.
when set to True those original values will be ignored and the timestamp will contain the time and date at which the restored file is created on destination drive.

- **Language**
  - used to set the language of the DefaultEvents messages
  - integer
  - runtime read/write / designtime read/write

  This property determines the language used for messages, when the DefaultEvents property is set to true. Also it determines the way the date is displayed.
  Currently the following languages are supported :

  english and german

  The following constants are defined in the RUWCONST.TXT file :

  VBBACKUP_LANG_ENGLISH0
  VBBACKUP_LANG_GERMAN1

- **QueryOverwrite**
  - sets the flag if there should be a query before overwritting a file
  - boolean
  - runtime read/write / designtime read/write

  when set to True (default), there will be a query before overwriting a file on the destination drive. when set to False there will be no asking if a file that is to be restored already exists on the destination drive. It will be overwritten without query.

- **Result**
  - contains the result of the last operation invoked with the Action property
  - boolean
  - runtime read

  After a backup or restore has finished you should use this property to check if everything went well.

  True indicates that the last action was successful

  False indicates that the backup or restore hasn't been finished successfully.

- **SelectedFiles**
  - contains the number of files selected files of the filelist
  - integer
  - runtime read

  The number of SelectedFiles is by default the same as the FileListCount. In the EditRestoreList event you can toggle the FileListRestore(index) property of the files in the filelist. The SelectedFiles property contains the number of files selected to be restored.

- **WrongDiskNumber**
  - contains the number the disk currently in the drive

•integer
•runtime read

This property is contains the number of the restore disk currently in the drive, while the DiskNumber property contains the disk number which should be in the drive. This property is useful when you want to display a message, when a user has inserted a wrong disk. E.g., the user should insert disk 3 of a backup-set. So the DiskNumber property is 3. If the user instead inserts disk 2, the WrongDiskNumber property is set to 2 and you can display a message (in the ErroqQuery event) that the user has inserted disk 2 instead of disk 3.

3. The Events

Here is a list of all the events triggered from the VBBACKUP.VBX

•**ErrorQuery**(ErrIndex as Integer, ErrMsg as String, DosError as Integer, MsgBoxChoice as Integer)

This event is triggered everytime that an error occurs. There are several errors that may occur, e.g. that no disk is inserted, or an inserted disk contains data, and so on. See the RUWCONST.TXT file for a list of all error constants. Note, that this event isn't triggered when you have set the DefaultEvents property to True, because in that case the default message box is opened from the custom control.

ErrIndex
- is an integer with the error index

ErrMsg
- is a string with the error message corresponding with the ErrIndex. The language of this string depends from the setting of the Language property

DosError
- is an integer with the corresponding dos error (only available for some errors)

MsgBoxChoice
- an integer with the possible calling values for the MessageBox function. This variable also takes the return value of this event, e.g., when MsgBoxChoice has the value MB_OKCANCEL, then you should set MsgBoxChoice to either IDOK or IDCANCEL, depending on what you (or the user of your application) wants to do.

•**CheckUser**(Int1 as Integer, Int2 as Integer, Double1 as Double, Double2 as Double, Str1 as String, Str2 as String, Str3 as String, Str4 as String, Cancel as Integer)

This event is triggered everytime that the header of a backup disk has been read. That occurs while a restore, when the required disk has been inserted. The variables are filled with the information found in the header of the backup disk, those will be either empty or contain the values which have been passed with a corresponding GetUser() while the backup.

Int1 - integer value
Int2 - integer value
Double1 - double value
Double2 - double value

Str1 - string with a len of 30 characters
Str2 - string with a len of 30 characters
Str3 - string with a len of 10 characters
Str4 - string with a len of 10 characters
Cancel - boolean, which you can set to True if you want to cancel the restore

This event shall give you the possibility of adding password checking to your backup/restore. It works in conjunction with the GetUser event.
Note, that this event is only triggered while a restore is running.

•**GetUser**(Int1 as Integer, Int2 as Integer, Double1 as Double, Double2 as Double, Str1 as String, Str2 as String, Str3 as String, Str4 as String, Cancel as Integer)

This event is triggered everytime that the header of a backup disk is to be written to disk. You can fill the variables with any information you like to add to this backup, e.g. a password, a description of the backup-set or whatever else.

Int1 - integer value
Int2 - integer value
Double1 - double value
Double2 - double value
Str1 - string with a len of 30 characters
Str2 - string with a len of 30 characters
Str3 - string with a len of 10 characters
Str4 - string with a len of 10 characters
Cancel - boolean, which you can set to True if you want to cancel the restore

This event shall give you the possibility of adding password checking to your backup/restore. It works in conjunction with the CheckUser event. Note, that this event is only triggered while a backup is running.
You can e.g. let the user enter a password and set Str4 to this one Str4 = password$, and then, while the CheckUser event of a restore, let the user enter his password and compare it with Str4.
Of course it would be better to scramble the passwords.

•**Progress**(Cancel as Integer)

This event is triggered every time when a block of 16 KB has been written to disk. So you can you this event to display a progress bar, show the filename, offer a cancel button or a lot of more.

Cancel - boolean, which you can set to True if you want to cancel the backup/restore

•**GetNextDisk**(DiskNumber as Integer, Cancel as Integer)

This event is triggered everytime when a new disk should be inserted into the disk drive. Note, that this event isn't triggered when you have set the DefaultEvents property to True, because in that case the default message box is opened from the custom control.

In this event you should open a message box which tells the user to insert a disk.

DiskNumber - integer containing the number of the required disk

Cancel - boolean, which you can set to True if you want to cancel the restore

•**GetLastDisk**(Cancel as Integer)

This event is triggered when the last disk of a backup-set is required. This happens when you start a random restore or read the filelist. Note, that this event isn't triggered when you have set the DefaultEvents property to True, because in that case the default message box is opened from the custom control.

In this event you should open a message box which tells the user to insert the last disk of a backup-set.

Cancel - boolean, which you can set to True if you want to cancel the restore

•**GetDestination**(Path as String, Skip as Integer, Cancel as Integer)

This event is triggered when a file is about to be restored. You can change the destinationpath of the file or skip this file.

Path - string, that contains the full path of the file which is to be restored. You can change this variable to force the file to be restored in another path.
Skip - boolean, that you can set to True if you want to skip this file
Cancel - boolean, which you can set to True if you want to cancel the restore

•**EditRestoreList**(Cancel as Integer)

This event is triggered only when a random restore is running, after the last disk has been inserted and the filelist has been read from disk. In this event you have the possibility to select the files of the filelist which should be restored by setting the FileListRestore(index) property to True or to unselect the files by setting this property to False. By default all files are selected to be restored.

Cancel - boolean, which you can set to True if you want to cancel the restore

3. The Methods

Here is a list of all the methods you can use with a VBBACKUP.VBX custom control.

•**AddItem**

used to add an item to the filelist.
Syntax :

backup1.**AddItem** filepath

backup1 is VBBACKUP control
filepath is a string which contains the full filepath of the file to be added to the filellist

All the files you add to the filelist will be backuped when you start the backup with the Action property.

•**Clear**

used to clear the contents of the filelistthe filelist.
Syntax :

backup1.**Clear**

backup1 is VBBACKUP control

The Clear method removes all items from the filelist and sets the FileListCount property to zero.