HEXANET

Setup Studio 2.4 - Manual

Contents

PART 1: Using Setup Studio

Overview...

Why Setup Studio?

Setup Studio builds high quality Setup programs for (and under) Windows 3.x. Thanks to Setup Studio, you use CSETUP.DLL functions with your favourite language (it' s easier). No more than 4 functions required and we provide templates for Visual Basic and Visual C++. This generic samples let you build a professional quality program in less than one hour! Also, the Setup Studio Wizard, thanks to its built-in Zip engine, compresses all the files, builds the diskettes and creates all the required files lists. Setup Studio handles up to 1200 files, 40 files groups, 30 diskettes, files larger than a media, backups, automatic renamed files and so on... Also, it is MS Setup Toolkit compatible. This new version adds 2 customizable languages, a WRI manual, the VB sample and so on... As a conclusion, Setup Studio 2.4 is incredibly cheap: No more than $50-, Free upgrade and Royalties free!

Also, Setup Studio was one of the first setup toolkits available as ShareWares. We use it with most of our other Studio toolkits, that' s why more than 10000 developers tried it! With the previous releases, we found about 30 difficult cases (...), and since we improved all the setup components, we hope we will find no more than 1 problem for 1000 installations...

Required hardware and software

- 100% compatible PC, SVGA, Hard disk.
- DOS 3.3 or higher.
- Windows 3.10 or higher,
- A Windows 3.10 development software which creates executable programs and can also works with external DLLs.

Features

Setup Studio 2.4 doesn' t require any more external tool: Now it integrates a ZIP file compiler and the wizard handles multiple Setup projects and, includes a powerful files manager (to select group or options for the files to install) and, an optimized files compiler to create diskettes images. Main features are:

- Microsoft SDK Setup Toolkit compatible,
- Up to 1200 The files , 30 diskettes and 40 groups,
- Unlimited files sizes,
- Splitted files management, system shared files and VBX support,
- Only 4 functions required to create a complete setup program,
- Over than 80 low and high levels functions ,
- It' s possible to use the DLL functions without the Setup engine,
- Files date checking, automatic and dynamic backup,
- New 'Never overwrite' mode,
- Automatically renames files with conflictual filenames,
- Multiple Setup projects support,
- WinFile, DOS 6.0, files , paths... support,
- Automatic trace file management,
- About 200 Kb. required on diskette for a medium setup.
- Setup Studio comes with 2 templates: One for Microsoft Visual Basic 2.0 and another for Visual C++ 1.5.
- 2 customizable languages available,
- Generated installation programs are bilingual,
- The assistant allows files management, create diskettes images and helps you to design setup programs.
- Also, we do not distribute the Setup Studio C++ source code.

Unregistered Version

The evaluation version allows you to create installation programs as the registered version do with 3 differences:

- You are not allowed to distribute your installation programs made with this release,
- Files will be installed into the 'TEST' directory. For example, if you specify C:\PROGRAM as the destination directory, files will be copied into C:\TEST\PROGRAM. This may causes a problem if you plan to copy some file by yourself, since the destination directory is not the destination directory!
- You can not modify dialog boxes title.

To register, have a look at Registration and use the Order form.

What' s new?

If you register Setup Studio 1.2, 1.3 or 2.x, you don' t have to pay for this new release, just give your personal passwords to the initialisation functions as you do with the previous version.

New from 2.x to 2.4:

Customizable languages...
SETUP.EXE parameters changed...
VB Template...
VB Functions...
Built-in 'Browse' dialog box...
<Never Overwrite> mode...

New from 1.x to 2.x:

ZIP has replaced LZEXPAND...
Mutliple Setup projects...
The new setup Wizard...
New Functions...
Technical support...

ZIP has replaced LZEXPAND...

The problem

Previous Setup Studio releases use COMPRESS.EXE and LZEXPAND.DLL to compress and to expand files. Many users told us they don' t own COMPRESS.EXE, so they can' t use Setup Studio... Also, COMPRESS.EXE is quite slower and builds bigger files than PKZIP utility...

Setup Studio answer

Now, Setup Studio use ZIP files: Since Setup Studio uses its own (Zip Studio 2.5) Zip engine, you don' t have to use PKZIP program! Also, you can use PKWARE' s PKZIP. However, LZEXPAND.DLL must be inside the WINDOWS\SYSTEM directory to run Setup Studio setup programs. With ZIP files, you can' t use direct functions to copy a compressed file from a Setup Diskette to the hard drive. The <ExpandAndCopyFile> function has been deleted: If you need to performs some special actions outside Setup process, there are two solutions: You can use not compressed files or you can register Zip Studio API.

Tips

1/ Previous diskettes images.
You need to rebuild diskettes images with the ZIP format (since it' s not LZEXPAND compatible).

2/ MS SDK Setup Toolkit.
This tool doesn' t support ZIP files, so if you want to use the SDK toolkit with Setup Studio you need to compile your files with the DSKLAYT2.EXE utility.

Multiple Setup projects...

The problem

Previous Setup Studio release was using the last used configuration each time you call the wizard. For example, you create your project 'A' paths, you compile and so on... and you make this again for another Setup program. When you recall the Wizard, the project 'B' will be on the screen, if you want to use the first project you need to

configure all the project paths. We find another problem regarding MAIN.EXE program: Each time we compile our MAIN.EXE, we have to put this file in the SOURCE directory...

Setup Studio answers

In fact, we don' t think Setup Studio is really interesting for a single Setup program. Now Setup Studio is able to build new Setup project: Just call the [New Project] command and the Setup Studio Wizard builds all the necessary directories and, also it writes a configuration file (SETUP.CFG) for this project. When done, everything is ready. Obviously when you will launch the wizard, the last opened project will be used. You can recall another project with the [Load configuration] command. Setup Studio 2.4 also uses a specific directory for your MAIN.EXE. You should build this program (thanks to your current development system) in the 'MAIN' directory, so that, the Wizard can copy the latest version each time you use the [Select files] command.

Tips

1/ Old version to 2.x version
Yes, you must create projects for all your setup programs made with an old Release. Don' t forget to put your MAIN.EXE program inside the 'MAIN' directory (to avoid warnings).

2/ Save as...
The Setup Studio WIZARD includes a [Save as...] command: It' s a good idea to save SETUP.CFG with its original file name in a different directory (to reload it, since Setup Studio uses SETUP.CFG and not *.CFG).

3/ SETUP.CFG...
When you are on the Files Manager, just double click on a SETUP.CFG project configuration file to load the Wizard with this project configuration.

The new Setup Wizard...

The problems

With the old releases, you had to use the MS SDK Setup Toolkit. This tool allow you to set options for the files you wish to layout. Unfortunately, many users don' t have this tool, so, if they wish to set files options, they need to edit the SETUP.INF system file. It' s not really easy for medium and big setups. Another advantage of the MS SDK Setup Toolkit is the files compiler (to generate diskettes images): Now, we include a built-in files manager and an 'Optimized Files Compiler' to do these jobs.

Select files...

Now Setup Studio doesn' t use any more the MS SDK Setup Toolkit but builds compatible scripts files (SETUP.LYT, SETUP.LST and SETUP.INF). Our product includes a files manager to allow you to set many options, sections, files date... for the files you would like to layout. This is actually a powerful feature and you can use up to 1200 files! When you use this files manager, SETUP.LYT information file is written so that the Optimized Files Compiler can use it.

Optimized Files Compiler...

The old files compiler can' t use files options, nor splitted files and sections management... This new release handles everything you need at a very high speed thanks to the ZIP format. All options you specified with the Files Manager are used, it always splits the files to fill the diskettes directories, it manages SETUP.LST system file, automatically renames duplicated files name... and lots more. You can also use the built in ZIP compiler (Zip Studio 2.5 API based) or an external ZIP program.

Tips

1/ Old version to Setup Studio 2.x
Generally you must call [Select files] to build SETUP.LYT and, you have to use the [Build diskettes images] command to build ZIP compatible diskettes.

2/ Internal or External ZIP?

If you will zip all your diskettes images into a ZIP file (to put your product on BBS for example), the internal ZIP is recommended for its speed. If you want the best compression rate you will use the external mode (in such case the <Cancel> button will be disabled and you can' t do something else during the ZIP process).

3/ SDK Setup Toolkit compatible?
If you wish to use Setup Studio with this tool, you will use SETUP.LYT instead of the other files, because, this toolkit doesn' t support ZIP files. SETUP.LYT contains your Setup description (see higher). If you use the BACKUP option, you will have to rename 'BACKUP' and to use a filename (as specified in this tool documentation). The toolkit doesn' t support directory specification, so you will have warning when you will open the LYT file with DSKLAYT utility.

4/ When do we need to compile?
You need to use the [Select files] command each time you add or you delete a file inside your 'SOURCE' directory. You need to use the [Build diskettes images] command each time you call [Select files] or in case you change your MAIN.EXE program.

New functions...

Global modification

Previous releases use LPSTR (char far*) as arguments, now we use a LPCSTR (const char far*) which is more easy to use with C/C++.

Renamed

SetupInitForFunctions	replace InitForFunctions
SetupSetTitle	replace SetTitle
SetupSetBeepMode	replace SetBeepMode
SetupSetLanguage	replace SetLanguage

Changed

GetBackupName	replace GetTempName
CopyFile	replace ExpandAndCopyFile

Added

SetTextDetail
CenterDialogBox
GiveTheHand
IsFileNewer
BackupFile

Technical Support...

We provide a free technical support to registered users on CIS and Internet. Our CIS ID is 100333,27 and, our net address is hnet@dialup.francenet.fr. We check the mail every day. Please, understand this service is only for bug report and real problems. Also, make sure you studied enough this documentation before any mail...

Customizable languages...

Overview

As you know, Setup Studio uses two languages defined as LANGUAGE_ENGLISH and LANGUAGE_FRENCH. You can dynamically change the language thanks to SetupSetLanguage. However, maybe you would like to build a setup program for spanish and german users. To do this you will use

LANGUAGE_ENGLISH (default) as the spanish version, and, LANGUAGE_FRENCH for the german version.

How must I do?

First of all, backup CSETUP.DLL, SETUP.EXE and FINISH.EXE. Now, you have to open these 3 files with a resources compiler. Common Resources compiler are Microsoft APSTUDIO.EXE and Borland resources workshop: These tools allow you to open a binary file, to change the resources (in our case, the strings and dialog boxes resources), then to save the changes. Now you will simply replace the french strings with your LANGUAGE_FRENCH language string (in previous case, they are german strings), and the others with your LANGUAGE_ENGLISH version. The ENGLISH_LANGUAGES strings are under the french versions. You must not replace '%1', nor '%2' since these values are used with Setup Studio and they will be replaced during the Setup.

SETUP.EXE parameters changed...

With previous versions, MAIN.EXE was using 2 command line' s parameters: The first one was the original path (the source directory), and the next one was the temporary path (WINDOWS\~SSETUP). As we can guess the temporary path, SETUP.EXE doesn' t send it any more: The only one parameter MAIN.EXE receive is the source path. The temporary path is the WINDOWS directory and "~SSETUP\".

VB Template...

Warning!

We are not VB programmers! So, maybe we made some big mistakes with the VB sample we distribute. In such case, let us now. That' s why you should have a look at all the code we give for this sample before you choose to distribute it...

VB problems

The first problem is VBRUNx00.DLL. If the user may not have this file, you should distribute it with your Setup pack. To do it, add VBRUNx00.DLL into your SOURCE directory, and add the following line in SETUP.LST in order to copy the file to the temporary setup path before SETUP.EXE call your Visual Basic MAIN.EXE:

```
VBRUNx00.DL_ = VBRUNx00.DLL
```

As usual, the file must be compressed. Next problem: Visual Basic doesn' t find the DLL when they are not in the WINDOWS\SYSTEM directory. To work around this problem, we used SDK LoadLibrary/FreeLibrary to preload the DLLs. Also, CSETUP.DLL functions which return a string are not available with VB and they were replaced with VB... functions. Have a look at VB Functions for details.

Improvements...

The VB sample we provide is not as powerful as the VC sample: It uses 8 sections (groups). Also, we do not change the 'Options' form size as we do with VC. If you would like to share your VB template with other VB users, send us your works so that we can give it with our future releases.

WARNING!: Unregistered Version

Files will be installed into the 'TEST' directory. For example, if you specify C:\PROGRAM as the destination directory, files will be copied into C:\TEST\PROGRAM. This may causes a problem if you plan to copy some file by yourself, since the destination directory is not the destination directory! Also, you can' t install the program icons for the same reason.

VB Functions...

VB Strings management...

As you know Visual Basic doesn' t handle strings as C/C++ does: It doesn' t add a 0x00 character to the end of the string. That' s why you must build your buffer in this way (see CHOICE1.FRM for details):

```
Dim sTemp As String * 256
sTemp$ = String$(256, 0)
Mid$(sTemp$, 1, Len(Texte1.Text)) = Texte1.Text
Call SetupBrowseDir(Form3.hWnd, sTemp$)
iZero% = InStr(sTemp$, Chr$(0))
szTemp$ = Left$(sTemp$, iZero% - 1)
Texte1.Text = szTemp$
```

There is no problem when you use this method. However, you can' t call CSETUP functions which return a string. That' s why we add some new functions to let you use those functions.

VB Functions

VB... functions use a buffer (szBuffer$) and are 'Sub' instead of functions. This buffer is the original function result (for details, have a look at the corresponding CSETUP.DLL function).

VBFormatFileNameValid(szFileNameIn$, bWithFinalSlash%, szBuffer$)
VBGetAbsoluteFileName(szFileName$, szBuffer$)
VBGetPathFromPATH(iPath%, szBuffer$)
VBGetEnvVariable(szVariableName$, szBuffer$)
VBGetWinIniString (szSection$, szKey$, szDefault$, szBuffer$)
VBGetIniString (szFile$, szSection$, szKey$, szDefault$, szBuffer$)
VBGetWinFileAssociation(szFileExtName$, szBuffer$)
VBGetStringFromFile(szFileName$, lIndex&, szBuffer$)

<Never Overwrite> mode...

Description

Select this Setup Wizard, [Select files to layout] dialog box checkbox if you wish the setup process do not install the selected file if this one is already available on the computer. This option may be useful for system DLLs like CTL3D.DLL, VBRUNx00.DLL or LZEXPAND.DLL, to avoid GPF whith shared files.

Your first Setup program...

As you may know, there are 2 components to build a nice setup program: The Setup program itself (in our case, this one is MAIN.EXE), and the 'Layout', ie. the files lists and the compressed files. This chapter studies the MAIN.EXE part.

Setup Studio Concepts

3 required programs

Setup Studio setup programs use 3 EXE files instead of a single one: The first program is SETUP.EXE. As you guess, this file must not be compressed since SETUP.EXE launch the setup process. The next one, in logical order, is your actual setup program and must be named MAIN.EXE. This program let the user choose the destination directory, the files to install, then it launch the files transfer and it installs icons and so on... Your MAIN.EXE file use CSETUP.DLL and VBRUNx00.DLL in case of a VB applixcation. MAIN.EXE must be compressed. The last one EXE module is FINISH.EXE: This little program will simply destroy the temporary setup directory (WINDOWS\~SSETUP) as soon as possible. FINISH.EXE must not be compressed. You must distribute this 3 programs with all your Setup Studio programs. There is no royalties for these files.

2 required files lists

Also, Setup Studio uses 2 files lists: They are useful to keep the installation files group and options, and, they allow the setup program to install the required files. SETUP.LST is the system files list. Its main section is '[Files]'. This section is the files list of all the necessary files to run your MAIN.EXE file. The user doesn' t see these files. Basically, SETUP.LST files list contains CSETUP.DLL, UZSETUP.XXX, SGAUGE.DLL and CTL3D.DLL. The setup wizard uses a default template to build this files, its name is DEFAULT.LST and this file should be into your Setup Studio Wizard directory. The next one files list is SETUP.INF. This is the actual files list with groups and all the required informations to copy the files to layout. The Setup Studio Wizard builds this file when you [compile] your files (when you build the diskettes images). These files are MS SDK Setup Toolkit compatible.

Your files

You probably whish to install something...You files should be copied into the SOURCE directory (you create with the Setup Studio Wizard [New project] command). You can put up to 1200 files in this directory. It' s better to use files with extension and to compress this files. The files will be zipped when you choose <Compress>. To change installation options, use [Select files] Wizard command, and, when this is done, use [Build diskettes...] to compress the files and to build SETUP.INF.

The other setup modules

CSETUP.DLL is the Setup Studio library: All setup functions comes from this DLL, and, you must distribute it, and, you must compress this file. See VB and VC templates to use this library. Also, CSETUP.DLL need 2 additional modules: UZSETUP.XXX and SGAUGE.DLL. UZSETUP is the Unzip engine and SGAUGE is the graphical gauge. Do not forget to put these files into your SOURCE directory.

Your setup program

Now, it' s up to you! Once you called Setup Studio Wizard [New project] command, put the files you want to layout into the SOURCE directory, then call [Select files...] to create the files groups. Then call your development system and build your new project into the setup 'MAIN' directory.

Your MAIN.EXE program...

if you didn' t read Setup Studio concepts..., please do it now!

Who calls your program?

The user clicks on SETUP.EXE to run your (MAIN.EXE) installation program. SETUP.EXE will expand MAIN.EX_ into the temporary setup path (WINDOWS\~SSETUP\). Then when all SETUP.LST files specifications ares copied and expanded, SETUP.EXE will run MAIN.EXE whith one argument: the source directory. As you see, the end user doesn' t see MAIN.EXE.

MAIN.EXE tasks

For a minimal installation program, there are 3 compulsary process: MAIN.EXE must initialize CSETUP.DLL (with SetupInitialize), then it must add some sections (files groups) to install (with AddSectionToCopyList), then it will launch the setup process thanks to the GoSetup function, and when everything is performed, your setup program will delete the temporary setup path with DeleteSystemDir.

Step 1: Building MAIN.EXE project

The installation program you must create must be named MAIN.EXE and must be Windows 3.x compatible. This MAIN.EXE project or program must be placed in the MAIN directory the [New Project] command created. Depending on your compiler, you maybe have to link with CSETUP.DLL and CTL3D.DLL, and, you may have to preload some setup modules (like we did with VB). The LIB file we provide is the import libary for Microsoft Visual C++ 1.5. The first thing you must do in your 'WinMain' (or main initialization function), is to read the command line argument and to store it as the 'Setup source directory' (i.e. the directory in which the user launched the setup). Then, you will call the CSETUP.DLL SetupInitialize function to initialize the setup library. Also, you will use CTL3D.DLL <Ctl3dRegister> and <Ctl3dAutoSubclass> for 3D look.

Step 2: Dialog boxes

In many case you must use, at least, 3 dialog boxes: The first one is the main dialog box. It must be borderless, without any controls, since we will use this dialog as the setup background. To do this, you must use the ShowMainWindow function with WM_PAINT and WM_NCPAINT messages. Another dialog box will be used to ask the destination directory. To do this you can also use the new SetupBrowseDir function. The last one dialog box is the 'Options' dialog box, to let the user chooses the items he would like to install. To add a section (a files group), you will use the AddSectionToCopyList function.

Step 3: Setup process

Once every thing is OK, you will call GoSetup. Depending on the result, you may also install the icons, winfile associations and you can also run a demonstration program. WARNING: Because the unregistered version doesn' t use the path your give (but TEST\...), you can' t do it with this release. Also, before you close MAIN.EXE, you must call DeleteSystemDir, to delete the setup temporary directory.

Step 4: Build diskettes images

If you think your MAIN.EXE program is valid, it' s high time to build your setup files list and to build the diskettes. If you are building your first program, please use just 1 diskette whith very few options! To build the diskettes you will use the Setup Studio Wizard [Build diskettes...] command.

Template creation...

When your first MAIN program is done, it' s usual to create a template which you will use to create the MAIN.EXE file for your different projects. Demonstration program dialog boxes are typical dialog boxes for a medium complexity setup program. To create more complex setup programs, you can add, for example, a generic button for each kind of files to install. Also these buttons can call a second generic dialog box, like the first dialog box, with check boxes for the files to install.

Using a template just required the sections and check boxes names modification. The data files (SETUP.INF) and the (MAIN.EXE) program, because they are independent allow you to create and to use a template. Obviously, it' s more difficult to add low level functions to a template.

With Visual C++ and MFC 2.0 and a correct template you can develop simple installation programs and test it in about one hour. Try the VB and VC++ samples provided with Setup Studio 2.4.

Building the diskettes...

First of all, make sure all the necessary files are inside the SOURCE directory. If not, simply drag' n drop the files thanks to WinFile. You don' t have to put MAIN.EXE into the source directory. Now, call the Wizard [Select files...] command.

Setting the layout...

Each time you add one or more files to the SOURCE directory, you must run the [Select files...] command. When you do this, the Wizard will try to add SETUP.EXE, CSETUP.DLL, SGAUGE.DLL and UZSETUP.XXX, if they are in your Wizard directory. Also, the Wizard will look for MAIN.EXE in your 'MAIN' directory. Also, if the SETUP.LST file exist, this file entries are checked and a Warning is generated if a system file is not into your SOURCE directory. The dialog box displays the files list, the installation mode (checkboxes), the section (group) name and the files date and sub-directories. The most important field is the Section name: Default is 'Files'. You must give the sections names you use in MAIN.EXE.

Building the diskette(s)...

If your layout is OK, you can build SETUP.INF and compress your files: To do this call the [Create diskettes images] wizard command. The dialog box let you change the paths, the diskettes sizes and the compression engine. To build your first program, set <Diskettes size> to 'None' and choose the 'Internal compression' mode which is faster. To launch the process, click on <OK>. When done, the wizard displays a warning if there is something wrong, or, it will ask you to launch SETUP.EXE to test your setup. FINISH.EXE will be added to all the diskettes, so that the user doesn' t have to provide the first diskette to close SETUP.EXE. Also, if you have enough knowledge about SETUP.INF and SETUP.LST, you can edit these file to enhance the setup program. If you need more informations about the method to build the diskettes, have a look at [Build diskettes images].

Editing SETUP.INF

In some cases you will have to edit the SETUP.INF file to optimize diskettes sizes, to change the installation order and so on... You must have enough knowledge about Setup Studio to do this! You will call the [Edit SETUP.INF] command to do this: This function let you change the DEST\DISK1 SETUP.INF version. You must build the diskettes before you can use this functionality. Also, don' t forget that you will loose changes when you will build the diskettes. You can do the same with the SETUP.LST system file.

Setup Studio Wizard...

Setup Studio 2.4 includes this 'Setup Studio Wizard' useful assistant. This assistant allows you to manage your Setup projects, to create files lists, to select options, to make diskettes images and so on...The only thing this tool doesn' t handle is your MAIN.EXE setup program!

Using the Setup Wizard (configuration)...

To use the wizard, you must install it thanks to the setup program provided with this package.
2.4 CHANGE: You don' t have to put your PIF files in the WINDOWS directory.

The first thing to do is to edit Setup Studio directory PIF files to adapt them to your system. Maybe you will have to change the programs paths (PKZIP, ...). You must create a backup before any PIF file modification.

Then, you just have to call the [New project] command to create your first Setup project. As a conclusion, you must edit DEFAULT.LST file with a text editor (NOTEPAD.EXE for example) to adapt your configuration to yours default installation programs. For more informations, see SETUP.LST.

Overview...

This assistant allows you to manage your Setup projects, to specify files options, to create files lists, to make diskettes images and so on... To launch the wizard, double click on the "WIZARD" icon or, in the files manager, on a SETUP.CFG file (in this case, this project configuration will be used).

To use the Wizard, you must build a Setup project: To do this, simply call the [New Project] command. Afterwise, run WinFile and put all the files you wish to install (but MAIN.EXE) in your newly created project 'SOURCE' directory. Now, you can set the files installation options and section thanks to the [Select files..] command, or, you may prefer to build your MAIN.EXE file. For details about MAIN.EXE, see 'Your first setup program...' chapter. Thanks to the [Select files..] dialog box, give sections (groups) name for the files and make sure they correspond to the names you are using with your MAIN.EXE setup program! When everything is OK, just call [Build diskettes images] to buid the diskettes and to test your setup program.

SETUP.LST

SETUP.LST is the system files list to install. If there is no SETUP.LST file in your source directory, the DEFAULT.LST (in the main Setup Studio directory) will be used. It is important your DEFAULT.LST file is correctly configured. The wizard allows you to edit the SETUP.LST file of your first diskette image directory. The SETUP.LST file never will be compressed.

SETUP.LST Format

The SETUP.LST file includes 2 sections:

The first one, [Params] is no used by Setup Studio.
The second [Files] specifies the system files list which will be transferred in the system temporary Setup Studio directory. The files must be present on the first diskette image. The list format is this one:

```
CompressedFile = File
```

where CompressedFile specifies the actual diskette image file and File its original name. We must find in this list:

```
CSETUP.DL_   = CSETUP.DLL
UZSETUP.XXX = UZSETUP.XXX
MAIN.EX_     = MAIN.EXE
SGAUGE.DL_   = SGAUGE.DLL
SETUP.INF    = SETUP.INF
SETUP.LST    = SETUP.LST
```

Remark

With this new version, you must put UZSETUP.XXX into your SOURCE directory. DLLs and EXE system files must be compressed, otherwise Setup Studio can load these files in memory and after a diskette removing the system would be stopped. The FINISH.EXE file must not be specified in SETUP.LST. You can use up to 60 system files. Also, VB users may add VBx00RUN.DLL to SETUP.LST.

VBX and shared files

The VBX files specified in SETUP.LST are always copied in the Windows System directory(a backup copy is automatically performed). The VBX files and shared DLL must not be loaded in memory during this copy. If you use this kind of files , you must add some informations in the READ.ME file, regarding possible installation troubles, for example:

```
"Before you launch the installation program, please, close applications
that use XXX.VBX and YYY.DLL. "
```

SETUP.INF

SETUP.INF is created during the [Build diskettes images] process and this is the files list.

Description

After Compilation, you can edit this file thanks to the Wizard. The Wizard can edit the DEST\DISK1 SETUP.INF version and, in case we wish to use this file again, we must make a backup to keep changes which will be destroyed during the next files compilation.

SETUP.INF format

The file SETUP.INF includes, at least, two sections:
[Source Media Description] is the required diskettes list with this format:

```
"Diskette number", "title", "", ""
```

For example:

```
"1", "Disk1", "", ""
"2", "Disk2", "", ""
```

Next sections are groups of files named SECTIONS. During the installation process, we select or not these sections for the installation (with AddSectionToCopyList). These sections format is:

```
[SectionX]
c1,c2,,,,c3,,,c4,c5,,,,,,c6,c7,,,,
```

With:

c1 = Diskette number for this file.
c2 = Original name of the file(not compressed). This field can specify sub directories which will be added to the specified directory with the AddSectionToCopyList.function. The length of these sub directories must not be longer than 17 characters.
c3 = The file date YYYY-MM-DD formatted. If this field is empty , the original file date is preserved.
c4 = It can be blank, NEVER (overwrite) or OLDER (install if newer).

c5 = The file will be in read only mode if this field is READONLY.
c6 = Original file size in bytes.
c7 = If this field specifies SYSTEM, the file will be treated as a shared file.

Examples :

```
[Section1]
1, FILE1.TXT, , , , , , , ,  , , , , , , , , 150875 , , , ,
```

The first diskette FILE1.TXT file will be copied in the section Section1 associated directory. If the files checking mode is on, the program will check the copied file size is 150.875 bytes.

```
[Section2]
2, DIR1\DIR2\DIR3\FILE2.DLL, , , , 1994-01-30, , , , , , , , , ,
15879, SYSTEM , , , ,
```

The second diskette FILE2.DLL file will be, first, copied in the temporary shared files directory specified with the SetSharedBatchFile function because this is a shared file. The file date will be 1994-01-30. When the "normal" files treatment will be done, you need to call the DosTempExec function to launch the system files copy. Then the FILE2.DLL file will be copied in the Section2 associated directory plus DIR1, DIR2 and DIR3 sub directories which will be created if they don' t exist.

Limits

Each time you create a new setup program, just test it to be sure your program is OK. There are some SETUP.INF file incompatibilities, you can avoid them if you are a few logical. We can' t specify a SYSTEM READONLY file because the system transfert batch would have problems to delete the shared files temporary directory with such file, for instance. The real copy of the system files never will be checked. Maximum number of files is 1200. Maximum number of diskettes is 30. Maximum number of sections is 40.

READ.ME

The READ.ME file is a text file you generally add to your first diskette image. In case you don' t have a READ.ME file into your 'SOURCE' directory, when you build the diskettes, this one will be automatically generated. This text file won' t be compressed and give basic informations to avoid troubles and to let the user makes a manual installation (for example, in case you use VBX or shared DLLs with your installation program).

Tips...

How to layout the files?

Using the Wizard...

As you know, you must use the Setup Studio Wizard, [Select files] to layout the files. The dialog box lets you change installation options, sections, files date and sub-directories. You need to do this before you can compress the files and build the diskettes.

Overwriting a file...

There are 2 ways to overwrite a file: Thanks to the <Install if newer> option, the file will be installed if the file is missing on the computer, and, if this file already exists, Setup Studio will compare the date and will install your version if this one is the newest. This method rely on the programmer: Bad programmers may change some DLLs date (like CTL3D.DLL), so it' s better to use this method with your files but not for the others! The second option is <Never overwrite>. You maybe use this method with shared files (like VBRUNx00.DLL), to avoid the <Shared> mode and possible GPF. You may also use the <Backup> option to make a backup of the file if this one already exist: Use this method carefully because you can get 30 backup files for the same file!!! These 3 options are available with splitted files.

Using 'replaced' files...

When the Wizard compresses a file, it change its extension: The new file name is *.*_. Also, if you own 2 files named MAIN.RC and MAIN.RC2, there names would be MAIN.RC_ and MAIN.RC_... That' s why Setup Studio renames MAIN.RC2 into REPLACE.x_. If you have such a file, most layout options are not available and you may get a GPF if you use sub-directories or some options!

Using files larger than a media or more than one diskette...

When you use more than 1 diskette, you will find 1 or 2 splitted files on all these one. Splitted files names extensions are *.x_. <Install if newer>, <Never overwrite> and <Backup> options are available with splitted files. Setup Studio doesn' t allow more than 10 parts for these files. Splitted files must be compressed and original names must include a file extension.

Using shared system files...

If possible, do not use the <shared> option because many other options are not available and it needs an ugly DOS screen... However, if your software includes a new COMMDLG.DLL version for instance, as you now, you can' t replace it when the user runs Windows. To install the file, use the SetSharedBatchFile function. GoSetup copies these into a special temporary directory and writes in the *.BAT all necessary commands for a DOS transfer. When MAIN.EXE is about to end, you must launch this batch file thanks to the DosTempExec function which will copy these files under DOS. If you wish to delete the system temporary directory, you must manually add the corresponding command in the *.BAT file with the AddString.function for instance.

Using VBXs...

If you would like to install some VBX into the WINDOWS\SYSTEM directory, put these files into SETUP.LST and not into SETUP.INF. However, many users don' t like VBX in this shared directory!

Using encrypted ZIP files...

If you wish to install an encrypted ZIP file, there is nothing special, but this file must be compressed (one more time).

Additional ZIP treatments...
You may need to access the ZIP files during the setup process. To do this, you should download Zip Studio API (ZS25A.ZIP and ZS25B.ZIP). Do not forget you can save $35- when you register Setup Studio with Zip Studio API!

In case of a GPF...

Delete the WINDOWS\~SSETUP dir.
First of all, you should delete this directory. If you have a big crash, you probably can' t do it. In this case you must close window and restart the GUI. You can get a GPF if your Setup layout is too complex for Setup Studio, specially in case you are using splitted files (ie. more than 1 diskette) with other settings (shared, sub-dir, system...). To avoid this problem, delete some files into your SOURCE directory and rebuild the diskettes, and have a look at SETUP.INF. You must test your setup program...

Also...
If your layout is quite simple and you have a GPF, and you are using one of the template we provide, send us your system files and your SOURCE directory contents, thanks to a diskette.

Using Internal or External comp. mode?

Internal mode
The Internal compression mode uses Zip Studio 2.5 engine: This mode is faster than external mode so, it' s a good choice when you test your setup program. Also, produced ZIP files are a little bigger than PKWARE' s PKZIP, but it' s should not be a problem if you zip all the files to put them on a BBS for instance!

External mode
The 'External compression mode' uses PKZIP.PIF. PKZIP.PIF uses PKWARE' s PKZIP (if available on the computer). You should configure the PIF file for your paths before you can use it! PKZIP builds smaller files than the internal ZIP engine, but it is a little slower (specially when you have many little files). Also, you can' t <Cancel> the ZIP process with this mode and, the compression directory must be set to 'COMP' (default value for new projects).

How to exchange Setup Studio scripts...

When you are working in a developers team, maybe you need to exchange your Setup scripts and layout. (Please notice that you must buy a Setup Studio License for all the computers or developers).

SETUP.CFG
These files are Setup Studio Projects files. SETUP.CFG is on the LYT directory. They used INI files format. You may have to change the path values. As you know, when you double click on these CFG files inside Windows WinFile, the corresponding project is loaded inside the Setup Studio Wizard.

SETUP.LYT
As SETUP.CFG, this file should be inside the LYT directory. SETUP.LYT is built when you call the [Select files] command, in order to build SETUP.INF during the diskettes images built. This file is MS SDK Setup Toolkit (DSKLAYT) compatible as long as you don' t use the Setup Studio subdirectory functionality. DSKLAYT warning has no matter, but you should have a look at the configuration when you use the SDK tool. The only field you may change is the second line ('SRC=...'): You must give your SOURCE directory path (without any

final '\'). Also, don' t forget to put all the files to layout in this directory.

SETUP.INF

This file is available inside 'DEST\DISK1' directory when the [Build diskettes images] is successfully completed. There is nothing to change.

SETUP.LST

This file is available inside 'DEST\DISK1' directory when the [Build diskettes images] is successfully completed. There is nothing to change.

About MS SDK Setup Toolkit...

If you wish to use Setup Studio with this tool, you will use SETUP.LYT instead of the other files, because, this toolkit doesn' t support ZIP files. SETUP.LYT contains your Setup description (see higher). If you use the BACKUP option, you will have to rename BACKUP and to use a filename (as specified in this tool documentation). The toolkit doesn' t support directory specification, so you will have warning when you will open the LYT file with DSKLAYT utility.

Terms

Used terms are C and SDK language type. If you do not program with these languages, perhaps you will have some difficulties to understand them. These are some explanations:

int	Integer value.
long	Long integer value.
UINT	Signed long integer value.
BOOL	Integer value, 0 for FALSE, anything else for TRUE.
char	Character.
char far*	String.
LPSTR	String.
LPCSTR	Constant String.
HWND	Window' s Handle(Integer value).
HINSTANCE	Application' s Handle(Integer value).
void	Null type (empty).
WINAPI	FAR PASCAL.
COLORREF	Long integer value (Colour).

Shareware?

Setup Studio is ShareWare:
You can try it as you want before you decide or not to register. The only one way we have to sell enough licenses to support this software, is to distribute it all over the world. So, if you think this product will help one of your friends, don' t hesitate to give him a copy of this product: This is your interest! When you distribute Setup Studio, don' t change any file and give the complete pack. If you upload Setup Studio on a BBS, please name it SSETUP24.ZIP or SSETUP24.EXE. Obviously, if you register, you can' t distribute the registered version! Distributing programs made with Setup Studio unregistered version is forbidden!

Registration...

Versions
Two versions are available: The registered version and the evaluation version. The evaluation version can be freely distributed if you give the full Setup Studio pack and you don' t modify any file. You can find the latest sharewares versions:

- On COMPUSERVE (go WUGNET, MSBASIC or WINSHARE),
- On main Internet FTP sites, like CICA (winftp.cica.indinana.edu),
- On ShareWares CD-ROMs,
- For French users, thanks to DP Tool Club (BP 745, 59657 VILLENEUVE D' ASCQ, TEL: (16) 20.05.35.66 (14-18H))

With the unregistered version, you can fully test Setup Studio. You will see Shareware principles reminders when you initialize the DLL. You cannot specify directory in which you want to put your files: All installed files will be placed in the "TEST" directory. Distributing programs made with Setup Studio unregistered version is forbidden! The registered version required the evaluation version because you will change the old version thanks to your two personal passwords.

WARNING!: Unregistered Version
Files will be installed into the 'TEST' directory. For example, if you specify C:\PROGRAM as the destination directory, files will be copied into C:\TEST\PROGRAM. This may causes a problem if you plan to copy some file by yourself, since the destination directory is not the destination directory! Also, you can' t install the program icons for the same reason.

How to order?
You can order by mail: See Order form for details. We receive US mail in about one week.

Also, you can use Compuserve SWREG registration service. To use it just 'go SWREG' then give one of the following ID and your credit card number. SWREG IDs are: 2455 for Setup Studio alone ($50-), 2667 for Setup Studio + Zip Studio ($85-) and, 3337 for Setup Studio + VBX Studio ($85-). You may pay a little additional fee for SWREG registrations.

If you have an US credit card (or an international credit card), you may also register through PsL registration service: You can order with MC, Visa, or American Express from the Public (software) Library (PsL) by calling 1-800-2424-PSL or FAXing 1-713-524-6398. These numbers are for orders only. For questions about credit card orders, call PsL at 713-524-6394. You can also mail credit card orders to PsL at P.O.Box 35705, Houston, TX 77235-5705. Setup Studio PSL ID is #11364 ($50-).

Prices?
When you register Setup Studio, you can save up to $35-! We propose 2 bundles for Setup Studio: The first one includes Zip Studio and Setup Studio, it costs $85- ($35- savings). SWREG ID is 2667. The next one includes

VBX Studio and Setup Studio, it costs $85- ($20- savings). SWREG ID is 3337. PsL doesn' t propose these special offers.

Currency	SETUP STUDIO	SETUP STUDIO + ZIP STUDIO
Francs français:	250-	450-
US Dollars:	50-	85-
Deutsch Mark:	90-	145-
GB Pounds:	38-	60-

(Others not allowed)
These prices include shipping and French VAT.
If you' ve got a COMPUSERVE ID or an EMail address, we will mail you your passwords before we send you your invoice, in order to avoid Postal delays. We always send out a printed invoice.

Upgrades

If you' ve registered the Setup Studio 1.2, 1.3 or 2.x release, you don' t have to pay for this new version. Just give your personal passwords to initialisation functions as you do with previous releases.

License

The license is an using authorization but not a product cession. Programs made with the unregistered version never will be distributed. To distribute your installation programs, you must register Setup Studio. The registered version kills Copyright warning and product limitations. You must register one license for one user and one computer. If you develop in a team, you must register Setup Studio for all developers.

Technical support

You can join us at CIS 100333,27 and Internet hnet@dialup.francenet.fr. We check the mail every day. We do not provide any thecnical support to unregistered users. Please, understand that this support is only for bug report and real problems you may find with our toolkit and not with other development tools (like Microsoft Visual Basic for instance). You should carefuly read the WRI file before any questions!

Other products

- Zip Studio API 2.5
Our famous Zip\Unzip API for Windows 3.x. It includes the DLLs, a VBX and the Zip Studio Shell. Setup Studio built-in Zip engine use Zips Studio DLLs. You can register this toolkit with Setup Studio to save som $$$!

- Zip Studio Shell 2.5
Our Archives Shell for Windows 3.x: Built-in Zip engine and up to 25 Archives format. Available for $25- SWREG ID is 3832. Licenses packs available from $3.8/License!

- String Studio 2.02
A MFC CString extension for MFC (VC++) users. Now it includes over 110 routines and a DOS version is also provided. With the new CStr string class you can handle CTime objects, files, numbers, strings items and much more. Useful with VBX Studio. Source code is available on Compuserve for just $35- (SWREG id is 3003) or $70 with VBX Studio 1.2 (SWREG id is 3521).

- VBX Studio 1.2b/2.x
15 general purpose VBXs for Visual programers. SWREG ID is 3336, PSL ID is 11593. Just $55-!

Order form

To register you can print this help page and manually fill it, or use ORDER.TXT.
Unreadable orders will be ignored!

- Setup Studio 2.4 (SW) Order Form -

Name

```
COMPANY:_ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _
NAME:_ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _
FIRST NAME:_ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _
ATTENTION TO:_ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _
ADDRESS:_ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _
_ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _
ZIP:_ _ _ _ _ _ _ _ _ _
TOWN:_ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _
COUNTRY:_ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _
COMPUSERVE ID: _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _
EMAIL: _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _
WHERE DID YOU FIND SETUP STUDIO?:_ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _
WHICH DEVELOPMENT DO YOU USE?_ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _
```

Product(s)

```
QUANTITY:_ _ _
CURRENCY:_ _ _ _ _ _ _ _
PRODUCT(S):_ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _

PRICE:
```

Currency	SETUP STUDIO	SETUP STUDIO + ZIP STUDIO
Francs français:	250-	450-
US Dollars:	50-	85-
Deutsch Mark:	90-	145-
GB Pounds:	38-	60-

(Others not allowed)
These prices include shipping and French VAT.
If you' ve got a COMPUSERVE ID or an EMail address, we will mail you your passwords before we send you your invoice, in order to avoid Postal delays. We always send out a printed invoice.

Send this form with your check to:

HEXANET

HEXANET
BP 385.16
75768 PARIS CEDEX 16
FRANCE

PART 2: CSETUP.DLL Functions

CSETUP.DLL Functions

To initialize CSETUP.DLL without using the installation engine (ie. to use low level functions), you must use the SetupInitForFunctions. initialization function.

Misc. functions...

bDetect

BOOL bDetect(int CVALUE)

Description
Returns TRUE in case the specified item is available. Constants <CVALUE> are the following:

BDETECT_WIN386	386 mode.
BDETECT_VGA640	VGA 640 pixels at least.
BDETECT_VGA800	VGA 800 pixels at least.
BDETECT_VGA1024	VGA 1024 pixels at least.
BDETECT_16COLORS	16 colours at least.
BDETECT_256COLORS	256 colours at least.
BDETECT_COPRO	Math. coprocessor.
BDETECT_286	80286 CPU.
BDETECT_386	80386 CPU.
BDETECT_486	80486 CPU.
BDETECT_MOUSE	Mouse available.

Example
```
if (!bDetect( BDETECT_MOUSE))
      MessageBox( NULL, "it is more easy to use Windows with a mouse!" ,
"", MB_OK);
```

lDetect

long lDetect(int CVALUE)

Description
Returns the long value for the specified item. <CVALUE> is one of the following constants:

LDETECT_BUFFERS	Buffers specified in CONFIG.SYS.
LDETECT_FILES	FILES specified in CONFIG.SYS.
LDETECT_DRIVES	Number of available drives.
LDETECT_FREERES	Rate of free system resources.
LDETECT_FREERAM	Available memory in Kb.

Example
```
if ( lDetect( LDETECT_FILES) < 5L)
      MessageBox( NULL, "There is no enough FILES specified in FILES!" ,
"", MB_OK);
```

Drives informations...

AskForDriveType

UINT AskForDriveType(UINT Drive)

Description
Returns the specified drive type depending on the <Drive> number. Returned value is one of these constants:

DRIVETYPE_REMOTE	Network unit.
DRIVETYPE_FIXED	Fixed unit (hard disk).
DRIVETYPE_REMOVABLE	Not fixed unit (floppy disk).
DRIVETYPE_UNKNOWN	Unknown unit type.

Example
```
if ( GetDriveType( GetDriveNumber("A")) == DRIVETYPE_REMOVABLE)
   TRACE( "The unit A:\\ is a floppy diskette.");
```

See also...
AskForDriveSpace
IsDriveValid
GetDriveNumber

AskForDriveSpace

long AskForDriveSpace(UINT drive)

Description
Returns the available free space in Kb. for the specified <drive> drive.

Example
```
if ( AskForDriveSpace( GetDriveNumber("C")) < 500L)
   TRACE( "There is no enough space on C: to continue.");
```

IsDriveValid

BOOL IsDriveValid(LPSTR szDriveLetter)

Description
Returns TRUE in case the drive is available. <szDriveLetter> must begin with the drive character. Also, you can provide a fully qualified path.

Example
```
if ( IsDriveValid ("C:\\WINDOWS\\"))
   TRACE( "The drive C:\\ is available.")
```

GetDriveNumber

int GetDriveNumber(LCPSTR szPath)

Description

Returns the corresponding drive number for <szPath>.<szPath> must begin with the drive character . You can give a fully qualified path. The returned number is used with many DLL and C Language functions.

Example

```
if ( AskForDriveSpace( GetDriveNumber("C:\\WINDOWS\\")) < 500L)
   TRACE( "There is no enough space on C: to continue.");
```

Directories Informations...

AskForWinDir

[Deleted]
We use SDK functions!

AskForWinSysDir

[Deleted]
We use SDK functions!

DoesDirExist

BOOL DoesDirExist(LPCSTR szDirName)

Description
Returns TRUE in case the <szDirName> specified directory exists. the "\" final is not compulsory for <szDirName>. You can use the function FormatFileNameValid to format <szDirName>. Also, if you wish to perform special treatments with files names, maybe our String Studio toolkit can help.

Example
```
if (!DoesDirExist( AskForWinDir()))
  TRACE( "Windows is no available on this station!")
```

Files informations...

DoesFileExist

BOOL DoesFileExist(LPCSTR szFileName, BOOL FormalSearch)

Description
Returns TRUE if the file exists. Also, if you wish to perform special treatments with files names, maybe our String Studio toolkit can help. If <FormalSearch> is TRUE, the search process is performed in this directory, otherwise, it occurs in the following directories:

- In the specified directory,
- In the current directory,
- In Windows and Windows\System directories,
- In PATH.

Example
```
if (!DoesFileExist( "WIN.INI", FALSE))
      TRACE( "Can' t find WIN.INI");
```

IsFileNewer

BOOL IsFileNewer(LPCSTR szFileIn, LPCSTR szFileOut);

Description
This new function returns TRUE if <szFileIn> is newer than <szFileOut>, FALSE if not. Files can be fully qualified. This function doesn' t check the files times, just the files dates are processed.

Example
```
if ( IsFileNewer( "A:\\READ.ME", "C:\\SETUP\\READ.ME"))
 {
 BackupFile( "C:\\SETUP\\READ.ME");
 CopyFile( "A:\\READ.ME", "C:\\SETUP\\READ.ME");
 }
```

GetAbsoluteFileName

char far* GetAbsoluteFileName(LPCSTR szFileName)

Description
Returns the full name of an existing file. If the file doesn' t exist , the function returns an empty string. Also, if you wish to perform special treatments with files names, maybe our String Studio toolkit can help.The search process occurs:

- In the specified directory,
- In the current directory,
- In Windows and Windows\System directory,
- In PATH.

Example

```
if ( lstrcmp( GetAbsoluteFileName("WIN.INI"), "") == 0)
      TRACE( "Can' t find WIN.INI");
```

Modules informations...

CountModuleUsage

UINT CountModuleUsage(LPCSTR szModule)

Description
Returns the number of instances for a module. The module can be a *.EXE or *.DLL file. The number of instances corresponds to the number of loads for a file. <szModule> must not specify any directory but, you can give a file name extension (EXE or DLL)

Example
```
if ( CountModuleUsage( "MAIN") > 0)
  TRACE( "The SETUP program is running.")
```

IsDLLLoaded

BOOL IsDLLLoaded(LPCSTR szModule)

Description
Returns TRUE if the module is currently loaded. The module can be a *.EXE or *.DLL file. <szModule> must not specify any directory, but you can give a file name extension (EXE or DLL).

Example
```
if ( IsDLLLoaded( "CSETUP") )
  TRACE( "The CSETUP.DLL DLL is loaded.")
```

Environment variables...

GetEnvVariable

char far* GetEnvVariable(LPCSTR szVariableName)

Description
Returns an environment variable value. This environment variable must not include the "=" character. If the variable is not available, the function returns an empty string.

Example
```
TRACE( GetEnvVariable("PATH"))
```

CountPathsInPATH

int CountPathsInPATH(void)

Description
Returns the number of paths in the environment variable PATH. Returns 0 in case the PATH is not defined in AUTOEXEC.BAT.

Example
```
if ( CountPathsInPATH() < 1)
      TRACE( "No PATH variable")
```

GetPathFromPATH

char far* GetPathFromPATH(int iPath)

Description
Returns a directory from the environment variable PATH. The first directory is 1. We can use CountPathsInPATH with this function. If an error occurs, this function returns an empty string.

Example
```
int NbPaths = 0;
NbPaths = CountPathsInPATH();
for ( int i=1 ; i < NbPaths +1 ; i++)
  TRACE( GetPathFromPATH( i))
```

Binary Files...

RenameFile

BOOL RenameFile(LPCSTR szOldFile, LPCSTR szNewFile)

Description
Renames a file. The file can specify paths but the drives must be the same. Returns TRUE in case this operation is successful, FALSE if not.

Example
```
RenameFile( "C:\\WINDOWS\\OLD.INI", "C:\\NEW.INI");
```

GetTempName

[DELETED]

GetBackupName

void GetBackupName(LPCSTR szFileIn, LPSTR szNewName);

Description
This new function has replaced GetTempName. This function fills <szNewName> with a not used backup name depending on the <szFileIn> filename value. If you give a path, this one will be saved. <szNewName> is formatted like "NAME.nnn" where 'nnn' is a not used number.

Example
```
LPSTR OldFile = "C:\\WINDOWS\\WIN.INI";
char BackupName[180];

GetBackupName( OldFile, BackupName);
...
```

DeleteFile

BOOL DeleteFile(LPCSTR szFile)

Description
Deletes a file. The file name may include its path without any wildcard (* or ?). Returns TRUE in case of success, FALSE if not.

Example
```
DeleteFile( "C:\\WINDOWS\\WIN.OLD");
```

TouchFile

BOOL TouchFile(LPCSTR szFileName, LPCSTR szNewDate)

Description
Changes a file date. If szNewDate is an empty string , the file will be stamped with the current system date. Also, if you wish to perform special treatments with files dates, maybe our String Studio toolkit can help. The szNewDate format is YYYY-MM-DD. Returns TRUE if successful, FALSE in the other case.

Example
```
TouchFile( "C:\\WINDOWS\\WIN.INI", "1994-01-30");
```

SetReadOnly

BOOL SetReadOnly(LPCSTR szFileName, BOOL bRO)

Description
Puts a file in read only mode or in normal mode. Also, if you wish to perform special treatments with files attributes, maybe our String Studio toolkit can help. If bRO is TRUE, the file attributes will be SYSTEM, HIDDEN and READONLY, otherwise the file attribute will be set to NORMAL. szFileName can specifies paths. Returns TRUE if successful, FALSE in the other case.

Example
```
SetReadOnly( "C:\\WINDOWS\\SPECIAL.EXE", TRUE);
```

GetFileLength

long GetFileLength(LPCSTR szFileName)

Description
Returns the file size in bytes or 0L in case the file doesn' t exist. szFileName can specify paths.

Example
```
lSize = GetFileLength( "C:\\WINDOWS\\NOTEPAD.EXE");
```

ExpandAndCopyFile

Deleted: Use CopyFile

CopyFile

BOOL CopyFile(LPCSTR szSrcFile, LPCSTR szDestFile);

Description
This new function has replaced <ExpandAndCopyFile>. Call this function to copy <szSrcFile> to <szDestFile>. This function doesn' t handle ZIP files format, so, if you wish to perform special ZIP treatment during the setup, you will need our Zip Studio API (see registration informations for details). CopyFile runs in background. Returns TRUE if the file is successfully copied, FALSE if not.

Example

```
LPSTR szFileIn = "A:\\SRCFILE.EXE";
LPSTR szFileOut = "C:\\SETUP\\DESTFILE.EXE";
if (!CopyFile( szFileIn, szFileOut))
 MessageBox( hWnd, "Can' t copy this file","", MB_OK);
```

BackupFile

BOOL BackupFile(LPCSTR szFileIn);

Description

This new function backup <szFileIn> file. This file will be backuped in the same directory with a not used name like "NAME.nnn" where <nnn> is a number. This function returns TRUE if backup is successful, FALSE if not.

Example

```
if ( DoesFileExist( "C:\\AUTOEXEC.BAT", TRUE))
   BackupFile( "C:\\AUTOEXEC.BAT");
```

ReCreateFile

BOOL ReCreateFile(LPCSTR szOriginalName, LPCSTR szPieceFileName)

Description

This function recreates a file from pieces of file. To use this function, you must call this one for each Piece you want to add (see the sample below). Returns TRUE if successful, FALSE in the other case.

Example

```
ReCreateFile( "C:\\PROG.EXE", "C:\\PROG.001");
ReCreateFile( "C:\\PROG.EXE", "C:\\PROG.002");
```

ASCII Files...

CreateAsciiFile

BOOL CreateAsciiFile(LPCSTR szFileName)

Description
This function creates an ASCII file (text). szFileName can specify paths. Returns TRUE if successful, FALSE in the other case.

Example
```
CreateAsciiFile( "C:\\WINDOWS\\SETUP.LOG");
```

CountFileLines

long CountFileLines(LPCSTR szFileName)

Description
This function returns number of lines of a text file.szFileName can specify paths. Returns 0L in case of error.

Example
```
lNbLines = CountFileLines( "C:\\WINDOWS\\\WIN.INI");
```

GetStringFromFile

char far* GetStringFromFile(LPCSTR szFileName, long lIndex x)

Description
This function returns a line from a file (without the final New Line character). First lIndex value is 1.szFileName can specify paths.This function returns an empty string in case of error.

Example
```
TRACE( GetStringFromFile( "C:\\WINNDOWS\\WIN.INI", 1L));
```

GetLineFromString

long GetLineFromString(LPCSTR szFileName, LPCSTR szString, long lStart)

Description
This function returns number of lines beginning with szString.The search starts to lStart. This function is not case sensitive. A space, a tab or an End of line character must end the string to look for.The function returns 0L in case of error.

Example

```
if ( GetLineFromString( "C:\\WINDOWS\\WIN.INI", "[WINDOWS]", 0L) == 0L)
   TRACE( "The string [WINDOWS] is not found in WIN.INI");
```

FindStringInTextFile

UINT FindStringInTextFile(LPCSTR szFileName, LPCSTR szString)

Description
Counts number of strings beginning with szString in a text file. This function is not case sensitive. A space, a tab or a End of line character must end the string to find. The function returns 0 in case of error.

Example
```
if ( FindStringInTextFile( "C:\\WINDOWS\\WIN.INI", "[WINDOWS]") == 0)
   TRACE( "The string [WINDOWS] is not found in WIN.INI");
```

DeleteString

BOOL DeleteString(LPCSTR szFile, int iIndex)

Description
This function deletes the specified line with iIndex in the text file szFile. First iIndex is 1. Returns TRUE if successful, FALSE in the other case.

Example
```
DeleteString( "SETUP.LOG", 1);
```

AddString

BOOL AddString(LPCSTR szFile, LPCSTR szValue)

Description
This function adds a string at the end of the szFile text file. Returns TRUE if successful, FALSE in the other case.

Example
```
AddString( "WIN.INI", "New line");
```

SetLineInFile

BOOL SetLineInFile(LPCSTR szFileName, LPCSTR szString, long lIndex)

Description
This function changes a text file line. lIndex specifies the line to change, first lIndex is 1. Returns TRUE if successful, FALSE in the other case.

Example
```
SetLineInFile( "WIN.INI", "[WINDOWS]", 1L);
```

Directories management...

CreateDir

BOOL CreateDir(LPCSTR szDirName)

Description
This function creates the specified directory. Also, if you wish to perform special treatments with files, maybe our String Studio toolkit can help. szDirName can specify sub directories which will be created if they don' t exist. szDirName can be ended with a "\" final character. Returns TRUE if successful, FALSE in the other case(in case the directory already exists).

Example
```
CreateDir( "C\\WINDOWS\\TEST1\\TEST2");
```

DeleteDir

BOOL DeleteDir(LPCSTR szDirName)

--- WARNING: Use with care! ---

Description
This function deletes the specified <szDirName> directory. Also, if you wish to perform special treatments with files, maybe our String Studio toolkit can help. If this directory contains files, these one will be erased (including system files). If this directory includes empty sub directories, they will be deleted. (This function can' t delete not empty sub directories). <szDirName> can end with '\'. Returns TRUE if successful, FALSE in the other case.

Example
```
DeleteDir( "C:\\TEMP\\TEST");
```

INI Files...

GetWinIniInt

int GetWinIniInt(LPCSTR szSection, LPCSTR szKey, int iDefault)

Description
Returns an integer value from WIN.INI. Also, if you wish to perform special treatments with INI files, maybe our String Studio toolkit can help.

Example
```
Value = GetWinIniInt( "WINDOWS", "KeyBoardDelay", 2);
```

GetWinIniString

char far* GetWinIniString(LPCSTR szSection, LPCSTR szKey, LPCSTR szDefault)

Description
Returns a string from WIN.INI. Also, if you wish to perform special treatments with INI files, maybe our String Studio toolkit can help.

Example
```
TRACE( GetWinIniString("WINDOWS", "spooler", "no"));
```

GetIniInt

int GetIniInt(LPCSTR szFile, LPCSTR szSection, LPCSTR szKey, int iDefault)

Description
Returns an integer value from an INI file. Also, if you wish to perform special treatments with INI files, maybe our String Studio toolkit can help. The INI file name must be fully qualified if this INI file is not in the 'Windows' directory.

Example
```
Value = GetIniInt( "C:\\SETUP\\SSTUDIO.INI", "GENERAL", "XPosition", 100);
```

GetIniString

char far* GetIniString(LPCSTR szFile, LPCSTR szSection, LPCSTR szKey, LPCSTR szDefault)

Description
Returns a string from an INI file. Also, if you wish to perform special treatments with INI files, maybe our String Studio toolkit can help. The INI file name must be fully qualified if this INI file is not in the 'Windows' directory.

Example

```
TRACE( GetIniString( "C:\\SETUP\\SSTUDIO.INI", "GENERAL", "XPosition", "100"));
```

SetWinnIniString

BOOL SetWinnIniString(LPCSTR szSection, LPCSTR szKey, LPCSTR szValue)

Description
Writes a string value to WIN.INI. Also, if you wish to perform special treatments with INI files, maybe our String Studio toolkit can help. Returns TRUE if successful, FALSE if not.

Example
```
SetWinIniString( "WINDOWS", "Spooler", "no");
```

SetIniString

BOOL SetIniString(LPCSTR szFile, LPCSTR szSection, LPCSTR szKey, LPCSTR szValue)

Description
Writes a string value to an INI file. Also, if you wish to perform special treatments with INI files, maybe our String Studio toolkit can help. The INI file name must be fully qualified if this INI file is not in the 'Windows' directory. Returns TRUE if successful, FALSE if not.

Example
```
SetIniString( "C:\\SETUP\\SSTUDIO.INI", "GENERAL", "XPosition", "100");
```

Programs execution...

QuitWindows

void QuitWindows(UINT cMode)

Description
Exits Windows with one of these <cMode> method:

RESTART_QUIT	Simple exit.
RESTART_REBOOT	Reboot.
RESTART_RESTART	Quitt and restart Windows.

Example
```
QuitWindows( RESTART_RESTART);
```

AppExecute

BOOL AppExecute(LPCSTR szAppName, UINT cShowMode)

Description
Launches a DOS or Windows application with the one of the following <cShowMode> value :

APPEXECUTE_SNORMAL	Show and activate the application.
APPEXECUTE_SICON	Iconize and activate the application.
APPEXECUTE_SMAX	Show in full screen and activate the application.
APPEXECUTE_SNOACTIVATE	Show and do not activate the application.

If the program is not in the Windows directory, you must give the application directory.
Returns TRUE if successful, FALSE if not.

Example
```
AppExecute( "C:\\SETUP\\SSTUDIO.EXE", APPEXECUTE_SNORMAL);
```

DosTempExec

BOOL DosTempExec(LPCSTR szAppName , LPCSTR szParams)

Description
This function allows you to quit Windows, to execute a DOS program and to comes back on Windows. This function must be used, with installation programs which use system shared files (COMMDLG.DLL for instance). <szParam> is the arguments used with the application command line. In case of Setup Studio shared files installation, <szAppName> will be set to the BAT file name you used for the copy process, and, <szParams> will be an empty string. Returns TRUE if successful, FALSE if not.

Example
```
DosTempExec( "C:\\SYSTEMP\\SETUP.BAT", "");
```

Sections informations...

GetSectionSize

long GetSectionSize(LPCSTR szSectionName)

Description
Returns the <szSectionName> section size in Kb.., or, 0 if the section doesn' t exist in SETUP.INF. The section name must be specified without "[ ]" and this name is case sensitive.

Example
```
lSectionSize = GetSectionSize( "Files ");
```

GetConfigurationSize

long GetConfigurationSize(void)

Description
Returns the user' s configuration size in Kb.. This is the size of all the selected sections to install.

Example
```
lConfigurationSize = GetConfigurationSize();
```

Files names functions...

IsFileName

BOOL IsFileName(LPCSTR szFileName, UINT IsFileType , UINT iCheckMode)

Description
Returns TRUE if the file name and\or the directory name are valid. Also, if you wish to perform additional treatments with files name, maybe our VBX Studio and String Studio toolkits can help. You can choose one of the following <iFileType>:

FILENAME_SINGLEFILE	The file doesn' t include any directory.
FILENAME_DIRWITHSLASH	Only paths with "\" are allowed.
FILENAME_DIRWITHOUTSLASH	Only paths without "\" are allowed.
FILENAME_DIRANDFILE	The file must be fully qualified.

The checking mode depends on one of the following <iCheckMode>:

CHECK_NOCHECK	No test.
CHECK_FILEEXIST	The file must exist.
CHECK_ROOTDIRSEXIST	Sub directories (but last) must exist.
CHECK_ALLDIREXIST	All specified dirs must exist.
CHECK_DIRANDFILE	All dirs and the file must exist.

Example
```
if ( IsFileName( szUserFile, FILENAME_DIRANDFILE, CHECK_DIRANDFILE))
      FileOpen( szUserFile, ....
```

FormatFileNameValid

char far* FormatFileNameValid(LPCSTR szFileNameIn, BOOL bWithFinalSlash)

Description
Reformats a file or a directory name. Also, if you wish to perform additional treatments with files name, maybe our VBX Studio and String Studio toolkits can help. If <bWithFinalSlash> is set to TRUE and, in case <szFileNameIn> doesn' t end with "\", this last character will be added. This function must be used with IsFileName.If <szFileNameIn> doesn' t specify any unit,the Windows drive will be used. The returned value is new <szFileNameIn> or an empty string in case of error.

Example
```
if (!IsFileName( szUserPath, FILENAME_DIRWITHSLASH, CHECK_ALLDIREXIST))
      lstrcpy( szUserPath, FormatFileNameValid( szInput, TRUE));
```

Setup directories...

GetOriginalDir

void GetOriginalDir(LPSTR szBuffer)

Description
Fills <szBuffer> with the directory from which the installation program (SETUP.EXE) was launched. This directory ends with "\".

Example
```
char far szBuffer[160];
GetOriginalDir( szBuffer);
TRACE( szBuffer);
```

GetSetupTempDir

void GetSetupTempDir(LPSTR szBuffer)

Description
Fills <szBuffer> with the system temporary WINDOWS\~SSETUP\ directory.This directory ends with "\".

Example
```
char far szBuffer[160];
GetSetupTempDir( szBuffer);
TRACE( szBuffer);
```

Initialization functions...

SetupInitialize

BOOL SetupInitialize(LPCSTR szYourName, LPCSTR szPassword, int cLanguage , LPCSTR szSrcDir, LPCSTR szDestDir)

Description
This is the main initialization function. This function performs the following tasks:

- Checking license :
  If you are not using the registered version, a warning will be shown and sub directories and files to install will always be copied into the "TEST" directory. If you are registered, this limitation doesn' t apply, but don' t forget to give the correct values to this function, otherwise the Setup will be stopped. Unregistered users must give "TEST" as <szYourName> and an empty string as <szPassWord>. Registered users will use their passwords. To initialize the CSETUP.DLL DLL without launching the installation engine (In order to use low level functions), you must use the InitForFunctions.function.

- Language choice:
  Specifies LANGUAGE_FRENCH or LANGUAGE_ENGLISH. It' s possible to change the language during execution thanks to the SetupSetLanguage. function.

- Directories:
  You will receive, in your MAIN.EXE program the source directory to use with this function. This directory is the directory from which SETUP.EXE has been launched, you must give this one as <szSrcDir>. <szDestDir> is the temporary 'WINDOWS\~SSETUP\' directory: You should build this directory name in your MAIN program. (See the samples programs for details)..

If parameters are OK, this function reads SETUP.LST and SETUP.INF. This function returns TRUE in case the initialization is successful, FALSE if not.

Example
```
SetupInitialize( "TEST", "", LANGUAGE_ENGLISH, szSrcDir, szDestDir);
```

SetupInitForFunctions

void SetupInitForFunctions(LPCSTR szYourName, LPCSTR szPassword)

Description
This function indicates to CSETUP.DLL the license you have and you wish to use low level functions without the files installation engine. You must call this function at the beginning of your program. If you are not registered, specifies "TEST" as szYourName and an empty string as szPassword.

Example
```
WinMain( ...)
{
SetupInitForFunctions( "TEST", "");
...
```

SetLogFile

BOOL SetLogFile(LPCSTR szLogFileName)

Description
This function sets the LOG file to use during the installation process. If you don' t use this function, no LOG file will be used during the Setup. <szLogFileName> must be a fully qualified file name. If sub directories do not exist, they will be built during the installation process. Setup Studio automatically fills this file. To disable this trace mode, you must specify an empty string as <szLogFileName>. Remark: Do not put the trace file in the system temporary Setup directory because this file would be destroyed when Setup is complete. The function returns TRUE if the file is correct or in case the mode trace is disabled, FALSE if not.

Example
```
SetLogFile( lstrcat( szUserPathChoice , "SETUP.LOG");
```

SetSharedBatchFile

BOOL SetSharedBatchFile(LPCSTR szBatchFileName, LPCSTR szSysDir)

Description
This function sets the 'shared' temporary directory and the name of the BAT file to use to copy its content to its TRUE destination directory(ies). <szBatchFileName> must include a directory specification and, in case the this one doesn' t exist, it will be built during the installation process. In the same order, the temporary directory will be built by CSETUP.DLL. When MAIN.EXE program is about to terminate, the function DosTempExec must be used with <szBatchFileName> in order to launch the (DOS) system copy. If we wish to delete the system temporary CSETUP.DLL (C:\WINDOWS\~SSETUP) directory, we must add a line in <szBatchFileName> to do this. The function returns FALSE in case of error, TRUE if not.

Example
```
char szUserPath[160];
char szSysPath[160];
char szSysBatch[160];
...
lstrcpy( szSysPath, szUserPath);
lstrcat( szSysPath, "TMPSYS\\");
lstrcpy( szSysBatch, szSysPath);
lstrcat( szSysBatch, "SYSCOPY.BAT");

SetSharedBatchFile( szSysBatch, szSysPath);
...
GoSetup...
... // Use a variable not C:\\WINDOWS...
AddString( szSysBatch, "DEL C:\\WINDOWS\\~SSETUP\\MAIN.EXE");
AddString( szSysBatch, "DEL C:\\WINDOWS\\~SSETUP\\CSETUP.DLL");
...
AddString( szSysBatch, "RD C:\\WINDOWS\\~SSETUP");
...
DosTempExec( szSysBatch, "");
```

ProgMan functions...

There are two ways to use the following functions: You can add icons with an unique function which shows a dialog box with the icon name, or, you can use low level functions which do not show dialog boxes. Remark: If the Program manager is shown as an icon, adding groups and items won' t be visible, and, if the Windows Shell is not Windows ProgMan, icons installation may fail.

AddItemToProgman

void AddItemToProgman(LPCSTR szGroupName, LPCSTR szExeName, LPCSTR szTitle, LPCSTR szTextInfo)

Description
High level function to add a group and an icon in Program Manager and to display a dialog box. <szGroupName> is the group name in which we will add the icon. If this group doesn' t exist, it will be created. <szExeName> is the program name which contains the icon or, the name of the data file. The file must be fully qualified (with drive and path). <szTitle> is the icon title. <szTextInfo> is an overview of the icon which will be shown in the dialog box. Afterwise you must use <ShowWindow> with MAIN.EXE to re-activate MAIN.EXE.

Example
```
AddItemToProgman("Group1", "NOTEPAD.EXE", "Simple Text Editor", "Installed
by Setup Studio!");
```

AddProgmanGroup

BOOL AddProgmanGroup(LPCSTR szGroupName)

Description
Create a new group of icons in Program Manager.

Example
```
AddProgmanGroup( "New group");
```

ShowProgmanGroup

BOOL ShowProgmanGroup(LPCSTR szGroupName)

Description
Display a Program Manager group.

Example
```
ShowProgmanGroup( "Main Applications");
```

DeleteProgmanGroup

BOOL DeleteProgmanGroup(LPCSTR szGroupName)

Description
Delete a Program Manager group.

Example
```
DeleteProgmanGroup( "Games");
```

AddProgmanItem

BOOL AddProgmanItem(LPCSTR szProgName, LPCSTR szTitle)

Description
Adds an icon (an item) to Windows Program Manager. The program is added to the active group. szProgName specifies the name of executable file with its full path and szTitle specifies the icon name to create.

Example
```
AddProgManItem( "C:\\WINDOWS\\NOTEPAD.EXE", "Simple Editor");
```

QuitProgman

BOOL QuitProgman(void)

Description
Tells Program Manager we do not want to continue. To come back to MAIN.EXE, we must use SDK <ShowWindow> function or a similar function to re-activate MAIN.EXE.

Example
```
QuitProgman();
ShowWindow( hwnd, SW_SHOW);
```

DeleteProgmanItem

BOOL DeleteProgmanItem(LPCSTR szItemName)

Description
Deletes a Program Manager icon.

Example
```
ShowProgmanGroup( "Games");
DeleteProgManItem( "Alone");
```

Specific functions...

AddSectionToCopyList

BOOL AddSectionToCopyList(LPCSTR szSection, LPCSTR szDestDir)

Description
This function lets you select a section for installation. <szSection> is the section name to select. This function is case sensitive. Do not give the "[]" characters. <szDestDir> is the destination directory for this section (only this section). <szDestDir> must end with "\". If a section you try to select is already selected with an other destination directory, a warning will be displayed. The function returns TRUE in case of success, FALSE if not.

Example
```
char szUserMainPath[160];
char szDestDirHelpPath [160];
...
lstrcpy( szDestDirHelpPath , szUserMainPath);
lstrcat( szDestDirHelpPath , "HELP\\");
AddSectionToCopyList( "Help", szDestDirHelpPath );
```

GoSetup

UINT GoSetup(HWND hwnd, BOOL bUseFinish)

Description
GoSetup is the main installation function. This function launches the files installation process and displays the transfer dialog box. <hwnd> is the window HWND to use to send notification messages and the parent window.Generally, this is the window you use with the ShowMainWindow.function.

If <bUseFinish> is TRUE, the installation program will try to destroy the setup temporary system directory. In this case, GoSetup searches for FINISH.EXE on the current diskette. If this file is not on the current diskette, GoSetup asks the user to provide the first diskette which must contain this program. Then, in your program MAIN.EXE, you must call DeleteSystemDir to delete the directory thanks to FINSH.EXE. Also, if you use shared system files, you can delete this directory thanks to the BAT shared file that' s why . you can set <bUseFinish> to FALSE.You will never give FINISH.EXE as a system file in SETUP.LST.

The GoSetup function sends some notification messages to its parent window.If you are using Visual Basic, you can' t use these message unless you have a special VBX to do this task! You can use these messages to display some special dialog boxes, to launch a special treatment in case of error or, to change the text of the background screen during the installation. These messages allow you to customize your users interface.

These are the messages GoSetup sends out to <hwnd>:

SN_FILENOTFOUND
: File not found - lParam is a LPSTR on the not found file name (The user can retry).

SN_SRCDIRCHANGE
: User changes source directory - You can get the new directory name with the GetOriginalDir.function.

SN_FILEINSTALLED
: A file was successfully installed- lParam is the fully specified file name.

SN_NEWDISKREQUIRED
A new diskette is required - lParam is the diskette label or "0" in case GoSetup searches FINISH.EXE.

GoSetup returns one of the following constants:

SETUP_SUCCESSFUL The installation is OK.
SETUP_USERBYPASS User bypassed a file.
SETUP_FAILINCOPYING Error during copying (fatal).
SETUP_USERABORT User abort (fatal).
SETUP_CANTCREATEDIR Can' t create a directory (fatal).
SETUP_WRONGSIZE File format error (fatal in VERIFY mode).
SETUP_INTERNALERROR Error in your program (fatal).

Example

```
if ( GoSetup( hWnd, TRUE)== SETUP_SUCCESSFUL)
      AddProgmanGroup ....
```

DeleteSystemDir

BOOL DeleteSystemDir(void)

Description

This function deletes the system temporary directory as soon as possible (when MAIN.EXE is unloaded). This function requires FINISH.EXE which must be in the directory from which we launch the installation program.This function returns TRUE in case FINISH.EXE is available, otherwise a warning is displayed and the function returns FALSE.

Example

```
DeleteSystemDir();
```

AddPath

BOOL AddPath(LPCSTR szNewPath)

Description

This function adds the <szNewPath> directory to the PATH line in AUTOEXEC.BAT. <szNewPath> must start with ";". In case of a MS DOS 6.x multiple boot, this function adds <szNewPath> to all the lines beginning with PATH. You must check this function result value and, in case this one is set to FALSE, you must alert the user that a manual operation is required. This function returns TRUE in case of success, FALSE if not.

Example

```
char szUserPath[160];
char szNewPath[160];
...
lstrcpy( szNewPath, ";");
lstrcat( szNewPath, szUserPath);
if (!AddPath( szUserPath))
 MessageBox( hWnd, "You must add the path manually...
```

GetWinFileAssociation

char far* GetWinFileAssociation(LPCSTR szExtName)

Description
This function returns the full name of an EXE program associated with a data type. This name is the full name of the associated program. <szExtName> must be a file extension (like 'TXT'), or a filename (like 'MYFILE.TXT'). This function returns an empty string in case of error or, in case there is no associated program for this kind of files. This function is also available in our String Studio toolkit.

Example
```
TRACE( GetWinFileAssociation("README.TXT"));
TRACE( GetWinFileAssociation("TXT"));
```

CreateWinFileAssociation

BOOL CreateWinFileAssociation(LPCSTR szExtName, LPCSTR szExeFile)

Description
This function builds a new association for the <szExtName> kind of files. After you use this function, you can launch a file when you double click on it with Windows WinFile. <szExtName> is the files extension (like 'TXT'), and <szExeFile> is the program name with its full path. You must restart WinFile to activate this new association. The function returns TRUE in case of success and FALSE if not.This function is also available in our String Studio toolkit.

Example
```
CreateWinFileAssociation( "TXT", "C:\\WINDOWS\\NOTEPAD.EXE");
```

ShowWaitCursor

BOOL ShowWaitCursor(void);

Description
This function displays the Wait cursor. This function must be used only with SETUP programs which call the SetupInitialize. function (In normal case, there is no need to call this function because the Wait cursor management is automatically performed).

Example
```
ShowWaitCursor();
...
HideWaitCursor();
```

HideWaitCursor

BOOL HideWaitCursor(void)

Description
This function hides the Wait cursor. This function must be used only with SETUP programs which call the

SetupInitialize. function (In normal case, you don't need to call this function because the Wait cursor management is automatically performed).

Example

```
ShowWaitCursor();
...
HideWaitCursor();
```

GiveTheHand

void GiveTheHand(void);

Description

This new function must be called when you launch a long treatment. With this function, Windows messages are processed during your long treatment. If you use this function, you must take care that the user can close your application or dialog box during the treatment: To avoid this, add a variable like <bCanClose> and check this one in your WM_CLOSE function. Visual Basic programmers will use the standard 'DoEvents' instruction instead of our function.

Example

```
case PM_TREATMENT:
  bCanClose = FALSE;
  ...
  GiveTheHand();
  ...
  bCanClose = TRUE;
  break;

case WM_CLOSE:
  if ( bCanClose) ...
```

ShowMainWindow

BOOL ShowMainWindow(HWND hOriginalWnd)

Description

This is the main display function. You must call this function in your program, to reply to the WM_PAINT and WM_NCPAINT Windows messages (ie. when a draw operation is required). This function shows <hOriginalWnd> in full screen and shows, in this window, the background bitmap. the window <hOriginalWnd> can be a dialog box without border (see the provided samples for details). <hOriginalWnd> is the window which will receive notification messages sent out thanks to the GoSetup.function. This function returns TRUE if is possible to show the background bitmap, FALSE if not.

Example

```
case WM_PAINT:
case WM_NCPAINT:
      ShowMainWindow( hWnd);
      break;
```

Interface functions...

SetupSetLanguage

BOOL SetupSetLanguage(UINT cSetupLanguage)

Description
This function sets the language to use. <cSetupLanguage> can be LANGUAGE_FRENCH or LANGUAGE_ENGLISH. It is possible to change the language during the Setup. This function returns TRUE in case the language is correctly set, FALSE if not.

Example
```
SetupSetLanguage( LANGUAGE_ENGLISH);
GoSetup(...);
...
CASE SN_NEWDISKREQUIRED:
  SetupSetLanguage( LANGUAGE_FRENCH);
  ...
```

SetupSetBeepMode

BOOL SetupSetBeepMode (BOOL bDiskBeepMode)

Description
This function sets the beep mode when a new diskette is required. If <bDiskBeepMode> is TRUE the beep mode is available, The function returns TRUE in case the beep mode is set, FALSE if not.

Example
```
SetupSetBeepMode ( TRUE);
```

SetVerifyMode

BOOL SetVerifyMode (BOOL bVerify)

Description
This function sets the files checking mode .If <bVerify> is TRUE, a file format checking will be performed. This checking compares files sizes as specified in SETUP.INF to the actual files sizes. If they are different and this mode is selected, the setup program will be stopped. The checking mode allows the user to bypass a file and, in case you wish to do a full installation or nothing, you must make some actions thanks to the GoSetup. returned value.The function returns TRUE in case the verification mode is set, FALSE if not.

Example
```
SetVerifyMode ( TRUE);
```

SetupSetTitle

BOOL SetupSetTitle(LPCSTR szDialogTitle)

Description
This function sets the dialog boxes title to use with GoSetup. This function is available to registered users only. You can call this function during the installation process. The function returns TRUE in case the title is set, FALSE if not.

Example
```
SetupSetTitle( "First title");
GoSetup( ...);
...
case SN_NEWDISKREQUIRED:
   SetupSetTitle( "New title");
   ...
```

SetPatternBrush

BOOL SetPatternBrush(HINSTANCE hi, UINT iBitmap)

Description
This function sets the BITMAP picture to use to build the background brush. <hi> is the instance HANDLE of your application and, <iBitmap> is the number of a 8 x 8 pixels and 16 colours, at best, BITMAP . This function must be called before GoSetup and you can' t call it again during the installation process. The function returns TRUE in case the bitmap is set, FALSE if not.

Example
```
#define IDB_BRUSHBITMAP  362
HINSTANCE hInst;
...
WinMain( hInstance, ...)
hInst = hInstance;
...
SetPatternBrush( hInst, IDB_BRUSHBITMAP);
GoSetup( ...);
```

SetPatternBrushStandard

BOOL SetPatternBrushStandard(UINT iModel)

Description
This function sets the background brush to use. <iModel> is one of the following constants:

BROSSE_PBLANC	white.
BROSSE_PNOIR	black.
BROSSE_GRIS100	very light grey.
BROSSE_PGRISC	light grey.
BROSSE_GRIS150	grey.
BROSSE_PGRISF	dark grey.
BROSSE_CYAN25	very light cyan.
BROSSE_CYAN50	light cyan.
BROSSE_CYAN100	light cyan.
BROSSE_PCYANC	cyan.
BROSSE_PCYANF	dark cyan.

BROSSE_CYAN200	very dark cyan.
BROSSE_BLEU50	very light blue.
BROSSE_BLEUVERT	blue green.
BROSSE_BLEU150	medium blue.
BROSSE_PBLEUC	blue.
BROSSE_BLEU200	dark blue.
BROSSE_PBLEUF	blue .
BROSSE_BLEU250	blue-black.
BROSSE_VERT25	very light green.
BROSSE_VERT50	light green.
BROSSE_VERT100	light green.
BROSSE_PVERTC	green.
BROSSE_VERT200	dark green.
BROSSE_VERT250	very dark green.
BROSSE_PVERTF	dark green.
BROSSE_KAKI	khaki.
BROSSE_JAUNE50	very light yellow.
BROSSE_JAUNE100	yellow.
BROSSE_PJAUNEC	yellow.
BROSSE_JAUNE150	dark yellow.
BROSSE_ORANGE	orange.
BROSSE_BEIGE	beige.
BROSSE_PJAUNEF	very dark yellow.
BROSSE_MARRON	brown.
BROSSE_ROSE100	pink.
BROSSE_ROUGE100	very light red.
BROSSE_PROUGEC	red.
BROSSE_PROUGEF	dark red.
BROSSE_FUSH50	very light fuchsia.
BROSSE_FUSH100	light fuchsia.
BROSSE_PFUSHC	fuchsia.
BROSSE_PFUSHF	dark fuchsia.
BROSSE_3DRECT	rectang 3D.
BROSSE_3DCIRCLE	circle 3D.
BROSSE_3DLOZANGE	lozenges 3D.

You can not change the brush value during the setup process. This function returns TRUE in case the brush is set, FALSE if not.

Example

```
SetPatternBrushStandard( BROSSE_3DCIRCLE);
```

SetLogo

BOOL SetLogo(HINSTANCE hi, UINT iSetupLogo)

Description

This function changes the BITMAP to use as logo. <hi> is the instance's HANDLE of your application (MAIN.EXE) and <iSetupLogo> is the number of the bitmap you would like to use. The BITMAP is transparent type: The fuchsia light colour will not be displayed and the main background brush will not be hidden by this colour. It is possible to change the logo during the program execution (This allows you, for instance, to tell the users to must register...) and you can also switch from text mode to BITMAP mode. The function returns TRUE in case the picture is set, FALSE if not.

Example

```
#define IDB_LOGOBITMAP  363
HINSTANCE hInst;
...
WinMain( hInstance, ...)
hInst = hInstance;
...
SetLogo( hInst, IDB_LOGOBITMAP);
SetLogoType( FALSE);
GoSetup( ...);
```

SetTextLogo

BOOL SetTextLogo(LPCSTR szLogoText, LPCSTR szFont, UINT iLogoSize, COLORREF clrLogo, BOOL b3D, BOOL bItalic)

Description
This function sets the text to use as a logo (a title). <szLogoText> is the logo's text, <szFont> is the font (name) to use, <iLogoSize>, the height in pixels, <clrLogo>, the text colour. If <b3D> is set to TRUE the text will be displayed with a 3D effect and, in case <bItalic> is set to TRUE, the text will be displayed in italic mode. You can change this logo during the setup program (This allows you, for instance, to tell the users they must register...) and, you can also switch from text mode to BITMAP mode. The function returns TRUE in case the text is set, FALSE if not.

Example
```
SetTextLogo( "Setup Studio", "Arial", 48, RGB(192,192,192), TRUE, TRUE);
SetTextDetail( "To create powerful\nWindows Setup programs", "Arial", 20,
RGB(128,0,0), TRUE, TRUE);
SetLogoType( TRUE);
```

SetLogoType

BOOL SetLogoType(BOOL bTextType)

Description
You must use this function if you would like to use the text or the BITMAPlogo. If <bTextType> is set to TRUE, the text logo will be used and conversely. The function returns TRUE in case the logo mode is set, FALSE if not.

Example
```
SetTextLogo( "Setup Studio", "Arial", 48, RGB(192,192,192), TRUE, TRUE);
SetLogoType( TRUE);
```

CenterDialogBox

void CenterDialogBox(HWND m_hWnd);

Description
This new function centers the specified <m_hWnd> dialog box. In most cases, you must call this function during the dialog box intialisation (WM_INITDIALOG). However, this function doesn' t work with MS Visual Basic... This function is also available in our String Studio toolkit.

Example

```
case WM_INITDIALOG: CenterDialogBox( hWnd);....
```

SetTextDetail

BOOL SetTextDetail(LPCSTR szLogoText, LPCSTR szFont, UINT iLogoSize, COLORREF clrLogo, BOOL b3D, BOOL bItalic)

Description

Same as SetTextLogo but this function can display a (long) text under the text title. To use this function, you must call SetTextLogo before. The text length can be up to 1Ko. and you can use '\n' to add a newline.

Example

```
SetTextLogo( "Setup Studio", "Arial", 48, RGB(192,192,192), TRUE, TRUE);
SetTextDetail( "To create powerful\nWindows Setup programs", "Arial", 20,
RGB(128,0,0), TRUE, TRUE);
SetLogoType( TRUE);
```

SetupBrowseDir

void WINAPI SetupBrowseDir(HWND hWnd, LPSTR szBuffer); [2.4]

Description

This new function displays a 'Browse' dialog box. This box use COMMDLG.DLL 'Open' template. <hWnd> is the parent HWND for the dialog box (it must be set to your dialog box HWND) and, <szBuffer> is the current destination directory. <szBuffer> must be initialized to the user choice. <szBuffer> is also filled on output, and, it ends with a final '\'.

Example

```
    'VB Sample (see CHOICE1.FRM)
    Dim sTemp As String * 256
    sTemp$ = String$(256, 0)
    Mid$(sTemp$, 1, Len(Texte1.Text)) = Texte1.Text
    Call SetupBrowseDir(Form3.hWnd, sTemp$)
    iZero% = InStr(sTemp$, Chr$(0))
    szTemp$ = Left$(sTemp$, iZero% - 1)
    Texte1.Text = szTemp$
```

PART 3: Dialog Boxes explanations

Configuration...

This dialog box allows you to configure the Setup Studio Wizard. However, when you used the [New project] command, you don' t need to change the configuration.

Paths

"Source dir" is the source directory in which you put all the files to layout.
"Main.exe dir" is the directory in which you will build the MAIN.EXE setup program.
"Comp. dir" is the temporary compressed files directory. (This dir is 'COMP' when you use the 'External compression mode').
"Dest. dir" is the diskettes images root directory(sub directories 'DISKx' will be built inside this directory).
"Script dir" is the SETUP.LYT directory.

Remark

Directories names must end with "\" and all the used directories must be on the same drive.

New project...

You must call this function each time you want to build a new Setup project.

Current Main Setup Directory

This is the root directory in which the new project will be built. You can use the browse button to change this directory.

New project main directory

Just give your new project name here. It can be "SETUP", or "PROJECT1" for instance. The name you give must be a valid DOS directory name.

Validate the project name

When you click on the 'OK' button, this new project will be built. First of all, the main project directory will be built (this is the Setup Directory plus the Project Name). Afterwise, 5 subdirectories will be added: 'SOURCE', for your sources files to install, 'MAIN' for your MAIN.EXE setup program, 'LYT' for SETUP.CFG and SETUP.LYT required files, 'COMP' the temporary compressed files directory, and, 'DEST' which is the root directory for your diskettes images ('DISK1', 'DISK2'...). Then, a new setup configuration file (SETUP.CFG) will be added in the LYT directory. Also, you can build another project and use the [Load configuration] command to reload your previous project. Now, you are ready to put the files to layout in the 'SOURCE' directory.

Select files...

This Dialog Box lets you set the files installation options. You must call this dialog box each time you add or you delete a file in the SOURCE directory. See the 'You to layout the files' chapter for details.

Files list

This list box contains all the 'SOURCE' directory files. They are sorted depending on the order you use to put them in this directory, also, if your Setup need more than one diskette, you must take care of this order. If you don't respect this order, your user may have to give the diskette 3, then the diskette 1 and the diskette 3 once...In

most cases the users don' t like it! Also, you can edit SETUP.INF to make the installation process more 'natural'! You can select one or more files in this list to set its (or their) installation attributes. If you select more than one file, the last file
attributes will be used and you must reset all attributes to be sure all the files will be correctly configured.

Files options

Check or uncheck CheckBoxes to set the following installation options:
COMPRESS: To compress the file (Default),
SYSTEM: To put this file on the first diskette and to add this file in SETUP.LST: If you don' t wish to copy this file to the temporary setup directory, you must delete the corresponding line in SETUP.LST (thanks to the [Edit SETUP.LST] menu command for instance),
SHARED: To install this file on the DOS command line,
READ ONLY: Read Only, Hidden and System file attributes will be set,
BACKUP: During the installation, if this file is found, a backup will be performed with an unused file name,
INSTALL IF NEWER: If your file is newer or not found, it will be installed.
NEVER OVERWRITE: Install the file if not found.

Section (Compulsory!)

You must give the section name for the selected file(s). Section name can be as long as 20 characters and default is "Files". This name is case sensitive. You can also use the browse button to pick a section name in the list of already defined section names. These names must be used in your setup program thanks to the AddSectionToCopyList function.

SubDir

This is a specific Setup Studio functionality (not MS SDK Setup Toolkit compatible): You can give a directory for a file. For instance, you use one section labelled "Files", you associate "MYSETUP" as the destination directory for this section thanks to the AddSectionToCopyList function. Then, all the files from the 'Files' group, will be installed in this directory. But, if you would like to copy a specfic file from this section in a subdirectory: There are two methods to do it: You can build a new section, or, you simply give a value to this 'SubDir' field. For instance, if this special file is 'READ.ME' and you would like to put this file in the 'READ' directory, simply use the 'Files' section, but add the 'READ' subdir. During the setup, this file will copied into the 'MYSETUP\READ' directory.You can also use the browse button to pick a subdir in the list of already defined subdirs.

Date

You can change the date of the selected file(s). Write in this Edit Control the file date (YYYY-MM-DD format) or use the [>>] button to retrieve the actual file date. If you use this field, the file(s) time will be set to 00:00. You can use file(s) date with the <INSTALL IF NEWER> installation option.

View

Select one or more files and click on the VIEW button to edit them with their associated program.

File Info

Select a single file then click on the FILE INFO button to get its names, sizes and dates.

Update

Click on this button when you add or delete some files in the 'SOURCE' directory while this dialog box is displayed.

About and Help

Displays the 'about' dialog box and call this help. You can also use the 'F1' key to get help.

Now, if you would like to save your work, click on 'OK' or, use the 'CANCEL' button if you don' t want to save it. The Next step is [Build diskettes images].

Build diskettes images...

This dialog box lets you build the diskettes thanks to the Setup Studio Wizard Optimized Compiler.

Diskette Size

Is the diskettes sizes you would like to use. Choose 'NONE' if you do not use diskettes (when you use a CD-ROM for instance).

Paths

"Source dir" is the source directory in which you put all the files to layout.
"Main.exe dir" is the directory in which you will build the MAIN.EXE setup program.
"Comp. dir" is the temporary compressed files directory. (This dir is 'COMP' when you use the 'External compression mode').
"Dest. dir" is the diskettes images root directory(sub directories 'DISKx' will be built inside this directory).
"Script dir" is the SETUP.LYT directory.

Compression

Choose 'Internal' if you wish to use the built in ZIP compiler or External if you prefer to use an external ZIP utility. In case you choose the external mode, you must specify a PIF file, and, this PIF file must be in your main Setup Studio directory, and, this PIF window title must be the same as the PIF name ("PKZIP" for PKZIP.PIF" for example). If you don' t respect this, a GPF may occur! Also, if you use the external compression mode, you must set the compression directory to 'COMP' (as the [New Project] command does).

Zip to ZIP...

This dialog box allows you to zip all your diskettes images into a ZIP file you can easily distribute on BBS for instance.

Paths

"Source dir" is the main directory name in which your diskettes images sub directories were built.
"Destination" is the full name of the ZIP file to create.

Keep sub directories

Click on this checkbox if you wish to keep the sub directories (in this case, the diskettes images sub directories) names in your ZIP. file. Sub directories will be rebuilt when the user unzips this file.

Remark

Directories names must end with "\" and all the used directories must be on the same drive. If the ZIP file already exists, it will be deleted.

Zip to EXE...

This dialog box lets you zip all your diskettes images into a self-extractible ZIP file.

Paths

"Source dir" is the main directory name in which you built the diskettes images .

"Destination" is the full name of the ZIP (*.EXE) file to create.

Remark
Directories names must end with "\" and all the used directories must be on the same drive. If the file already exists, it will be deleted.

Manual operation...

This dialog box lets you compress or expand a file with PKZIP.PIF or PKUNZIP.PIF.

Paths
"Source file" is the full source file name.
"Destination" is the full destination file name.

Action
We choose "Compress" to compress the source file to the "Destination" file or, "Expand" to expand the source file to the "Destination" file.

Delete source file
If this check box is selected, the "Source file" will be deleted once the process is performed.

Remark
Directories names must end with "\" and all the used directories must be on the same drive. If the destination file already exists, it will be deleted.