Sierra On-Line Common Install

Product Specification v. 2.x

Authors: Peter Sarrett
Willie Eide
Jay Lee (2.X)
September 15, 1995

Help File: Susan Frischer

Contents

Sierra On-Line Common Install

This document describes the common Install/Setup program(s) being written for use with all products in the Sierra family. This version of the spec covers the specification for a Windows only Setup program.

2.X Note

Setup will run under Windows 3.x as well as Windows 95. Some features have been added that are Win95 only, notably the ability to write to the Win95 registry and support for the DirectX facilities of the Microsoft Game SDK.

1.0 Overview

Sierra is pushing for brand recognition across its product line, uniting all of our subsidiaries under the banner logo of Sierra. A big step in this effort is the creation of a common install program to be used by all products in the Sierra family. Aside from eliminating redundant development work, such an install would provide a familiar "face" to all Sierra products. Users who successfully install one product will find it much easier to install their next Sierra product. This will reduce tech support calls. Indeed, the install program is being designed to help reduce tech support calls from novice users by testing various aspects of the user's system and suggesting ways that common problems can be fixed. By simplifying, streamlining, and standardizing the Sierra installation process, we hope to reduce support costs and gain a competitive advantage in the marketplace.

The Sierra Windows Setup (hereafter referred to as Setup) uses standard Windows controls which will be familiar to Windows users. It uses a script to allow Sierra developers to customize the process to suit an application's needs, yet is smart enough to handle the most common tasks without being directed to do so by the script. Setup allows the user to enter registration information on-line, then dump it to a printer or register directly via modem. Setup will allow a user to easily uninstall products from his system.

2.0 Start Up

Setup actually consists of multiple parts: Presetup, Setup, Setup32 and a DLL full of language-specific resources. Presetup is the uncompressed SETUP.EXE which must reside in the root directory of the first distribution disk. Setup is called _SETUP.EXE and Setup32 is called SETUP32.EXE and they are compressed, along with MIDITEST.MID, one or more language DLLs and one or more help files in a file called SETUP.SOL. This file must be in the same directory as SETUP.EXE. The compressed language DLLs are named SOL_XXX.DLL where XXX is one of ENG, FRE, GER, SPA, or ITA for English, French, German, Spanish, or Italian, respectively. Similarly, the help files are SOL_XXX.HLP (these help files are the appropriate language version of what would be called _SETUP.HLP).

When Presetup is run, it gives the user an error message and exits if there's not enough room in the user's temporary directory (or C:\ if no TEMP dir is defined) for the uncompressed _SETUP.EXE, SETUP32.EXE, MIDITEST.MID, and appropriate SOL_XXX.DLL and SOL_XXX.HLP (as determined by the sLanguage entry in the [intl] section of the user's WINDOWS.INI file). Otherwise, it uncompresses these files to the user's temporary directory, runs _SETUP.EXE, and waits for _SETUP.EXE to terminate. At that time, it deletes all the temporary files and exits. SOL_XXX.DLL is uncompressed as SETUPL.DLL. SOL_XXX.HLP is uncompressed as _SETUP.HLP.

Presetup check in WIN.INI to determine what language the user is running and copies the appropriate DLL and HLP for that language. Setup then proceeds to run in the language that the user is running.

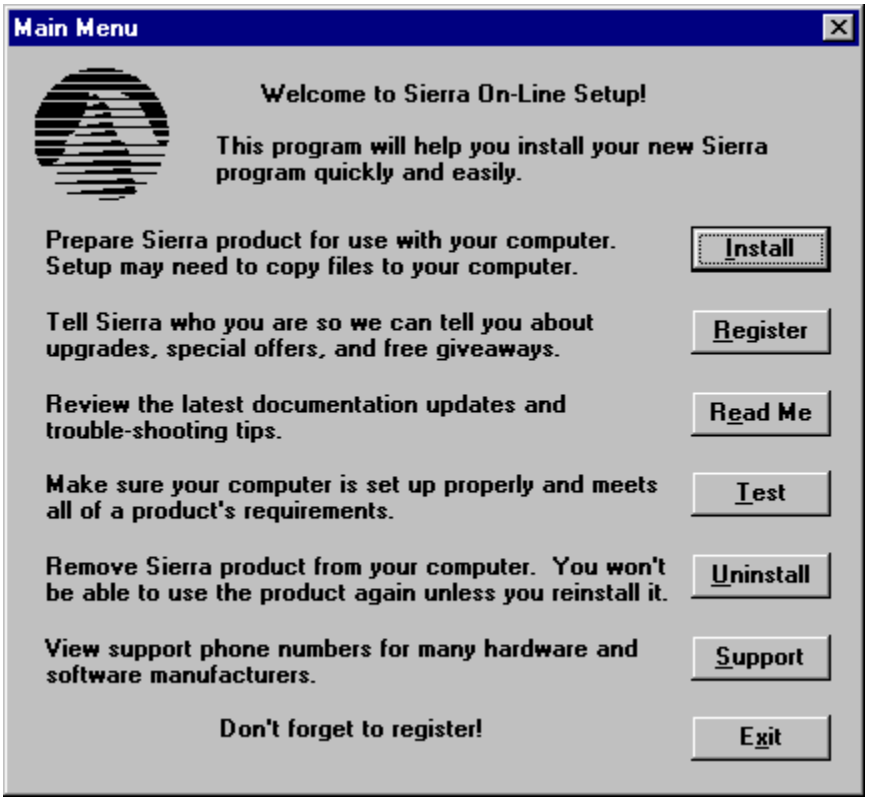
_SETUP.EXE will be copied to a permanent location as SETUP.EXE when the user installs a Sierra product. Thereafter, Setup can be run directly from the user's hard drive should the user wish to test his hardware, uninstall, or perform any of the other functions Setup allows.

Setup plays a .WAV fanfare and displays a basic animated Sierra logo before presenting the user with the main menu.

2.X Note
Setup has been updated to look for a file called LOGO.BMP from the directory that it was launched from. If found, it will use this as the logo instead of the default Sierra Half Dome logo. The use of this feature is discouraged as it runs counter to the desire to have a single corporate identity. As of this writing, the only approved use of this is for educational products for sale in Europe.

3.0 Main Menu

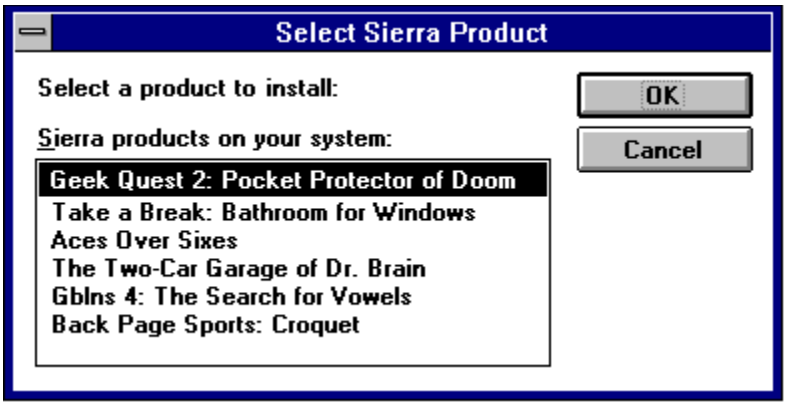
The main menu looks like this:



This dialog disappears or goes to the background when any of the right-hand buttons are pressed, reappearing when the function indicated by that button has finished. Setup will never exit completely without the user pressing the Exit button from this dialog first, unless the product allows the user to reboot his machine or restart Windows as part of that product's installation process.

When the Install, Register, Read Me, Test, Uninstall or Support buttons are pushed, Setup will try to determine which product the user wants to operate on. For Install and Test, Setup will search the directory from which it is run for a SIERRA.INF file. This file contains important information about the product being installed (size, system requirements, etc.). If a SIERRA.INF file is found, Setup will assume that the user wants to install the corresponding product. Otherwise, Setup will search the user's system for a SIERRA directory or subdirectory. If found, Setup then searches all subdirectories off the SIERRA directory for SIERRA.INF files. Each such file corresponds to a Sierra product the user has previously installed. For Install and Test, Setup will also check the user's floppy drives and CD-ROM drive for SIERRA.INF files. If all these searches fail, Setup will politely inform the user that something is wrong and will gracefully exit.

When Setup finds SIERRA.INF files, it brings up a dialog listing all the corresponding products in a list box:



The user chooses a product, which lets Setup know which .INF file to use for system requirements and other information.

Note that this dialog never appears if Setup finds a SIERRA.INF file in the directory from which it was run.

3.1 Install

Setup's philosophy is inspired by the Disney theme parks where rides take you along predetermined pathways from which you may not deviate. Setup isn't quite as restrictive as Walt, allowing developers some customization options to tailor Setup's behavior to an application's needs. Setup uses a limited scripting language and acquires information from the SIERRA.INF file. The script language allows the use of user-defined dialogs and the ability to launch a separate module if your application requires special handling not easily supported within Setup itself (for example, if you wanted to use Microsoft's Setup to install Video for Windows). Setup uses the PKWare libraries for file compression. Why? It's fast, compresses well, is easy to use, and works under Windows, 16 and 32 bit. If the compression algorithm is deemed insufficient, effort can be directed towards changing the algorithm to a method accepted by all of Sierra. Note that the PKWare compression libraries do **not** use the ZIP format. Thus files for use with Setup need to be compressed with a custom program (PKCOMP) which we provide.

3.1.1 The SIERRA.INF File

The SIERRA.INF file provides most of the information Setup needs to install a product (the remainder is specified in a LANGUAGE.INF file (see Section 3.1.2)). Each product must have a SIERRA.INF file in the same directory as SETUP.EXE on the installation disk. The SIERRA.INF gets copied to the user's hard drive during the installation process. Setup modifies this copy of SIERRA.INF to allow Setup to later determine which drive the product was installed from. Most aspects of Setup, including the hardware tests, will not work correctly until Setup has found a SIERRA.INF file to refer to.

The SIERRA.INF file looks similar to a Windows .INI file. Not surprising, since Setup uses the Windows PrivateProfile interface to read and write to the .INF file. The sections of the .INF may appear in any order, but **must** be separated by a single blank line consisting solely of a carriage return. Setup may interpret other blank lines as indicating the end of a section, so blank lines should **only** be used to separate sections. Some keys in some sections are optional. If such a key is missing, Setup will use the default value indicated in curly braces ({}).

Any line beginning with a semicolon (;) is a comment and will be ignored by Setup. The examples given in each section use semicolons to separate explanatory comments from the examples themselves-- these comments, which appear on the same line as actual .INF entries, are not legal .INF comments. A line must begin with a semicolon for it to be a legal comment.

3.1.1.1 Requirements

The Requirements section contains information about the application's hardware and software requirements. Any key in this section which has a default value is optional.

[Requirements]

Colors=*colors*

ScreenWidth=*xres*

ScreenHeight=*yres*

VideoSpeed=*Kpixels*

Wave=0 | 1 | 2

MIDI=0 | 1 | 2

CDROM=0 | 1 | 2 | 3 | 4

MemKB=*memory*

PhysicalMem=*memory*

WinVer=*winver*

CPU=386 | 486 | Pentium-*speed*

Joystick=0 | 1 | 2

SetupVer=*setupver*

Colors

colors = Number of colors required by the application (2, 16, 256, etc.). {16}

ScreenWidth

xres = Minimum screen width. {640}

ScreenHeight

yres = Minimum screen height. {480}

VideoSpeed

Kpixels = Thousands of pixels per second that video card must support.

Wave

0 = Product doesn't support .WAVs.
1 = Product supports .WAVs but doesn't require them.
2 = Product requires .WAV support. {0}

MIDI

0 = Product doesn't support MIDI.
1 = Product supports MIDI but doesn't require MIDI.
2 = Product requires MIDI. {0}

CDROM

0 = CD-ROM not required.
1 = Requires at least a single-speed.
2 = Requires at least a double-speed.
3 = Requires at least a triple-speed.
4 = Requires at least a quad-speed. {0}

MemKB

memory = KB of free memory required by your application. {0}

PhysicalMem

memory = KB of physical memory required by your application. {0}

WinVer

winver = Windows version required.

3.0 = 300, 3.1 = 310, Win95 = 395, etc. {300}

CPU

Minimum CPU and clock speed (e.g., 66 for 66 mHz) required.
Ex: 386-40

Pentium-90

Joystick

0 = Joystick not supported.
1 = Joystick supported but not required.
2 = Joystick required. {0}

SetupVer

setupver = Version of Sierra Setup for which this .INF file was designed. Must be of the form i.j.k.l (e.g., 1.0.0.0). {1.0.0.0}

3.1.1.2 Identification

The Ident section contains application identification information.

[Ident]

Version=*version*

ProductID=*product number*
ReadMe=*filename*

Version

version = Product version. Must be an integer.

ProductID

product number = Product number. Must be an integer. This is used for electronic registration.

ReadMe

filename = Filename for readme file. Looks in the same location as Language.INF.

3.1.1.3 Archives

The Archives section contains a list of all archives (files containing compressed files to be installed) on the distribution disks, along with the archive's size, the disk number on which the archive resides or begins, and whether or not this archive should be installed by default. This information is used for two purposes: to make it easy to locate files for installation, and to enable accurate disk space verification. If files exist which Setup will not be installing itself (for example, Video for Windows files which will be installed by Microsoft's Setup program), those files (or the compressed file(s) they are in, even if compressed with a program other than the one provided by the Setup team) should be listed here. The disk number for these archives that may be installed by other external programs should be set to 0. This notifies Setup NOT to install this file. This is so Setup can correctly determine the amount of space all desired files will require on the user's system. Note that the entries in this section do not follow the standard Windows .INI format.

Subdirectories that contain uncompressed files are also specified here. The declaration of the subdirectory is the same as a compressed file, but with a \. For example, if you had a subdirectory called FOO that contained uncompressed files, the specification for the archive name would be FOO\. By using the rules used for regular compressed archives, subdirectories can also be turned on or off using flags.

Care should be taken with the installation flag. Setting an archive's flag to 1 will force all files in that archive to be installed regardless of whether or not the file's installation flag is set to 1. However, setting an archive's flag to 0 will **not** force all files in that archive to remain uninstalled. When an archive's flag is 0, only the member files which are set to 1 will be installed.

[Archives]

FILENAME.EXT, disk number, file size, flag

Examples:

[Archives]
STUFF.SOL, 1, 0, 0

[Archives]
FOO\, 1, 0, 1

Note: Use of the compression utility, PKCOMP.EXE, will generate a PKCOMP.INF file in the root directory where the compression is performed. This file will contain the above-mentioned section of Archives which can then be cut and pasted into the products' SIERRA.INF file.

FILENAME.EXT

FILENAME.EXT = Archive file name
disk number **=** Number of the disk on which this file resides. The first disk is #1. In the case of archives which span multiple disks, this value must be the number of the disk on which the archive begins.

file size = Size of uncompressed archive in bytes. This value should be 0 for all archives which will be uncompressed by Setup. This value should be non-zero for all archives which will be uncompressed by an external program.

flag = 0 if some or all files in the archive should **not** be installed by default.
1 if all files in the archive should be installed by default.

STUFF.SOL

STUFF.SOL on disk 1 is a Sierra archive and should not be installed by default.

FOO\

Subdirectory FOO is on disk 1. It should be installed by default.

3.1.1.4 Files

The Files section contains a list of all files which Setup might need to copy to the user's hard drive. Also included with each file is the name of the archive that file resides in, the uncompressed size of that file, and a flag representing whether or not the file should be installed by default. Setup uses the location of the archive to determine which disk a file is on. If the file is uncompressed (and thus uses the NOARCHIVE keyword), you must also specify the disk where Setup can find the file. A file must not have the same name as an archive.

When NOARCHIVE is specified, Setup looks for that file in the *SOURCEDIR directory. When identifying files that reside in a subdirectory, the subdirectory name must be indicated in the *archive* key name (with the backslash). This directs Setup to look into that subdirectory for that file. The specified subdirectory must be in the list of Archives (see Section 3.1.1.3).

A more advanced use of the Files section is to use *FILENAME.EXE* as the destination (path or subdirectory included). This can be accomplished by using Special Values (see Section 3.1.1.7). Also the *archive* key name can be specified as a path or subdirectory.

Note: The only time the *disk* key name is used is when specifying NOARCHIVE.

[Files]

FILENAME.EXE, archive, [disk,] size, flag, [groupId]

Example:
[Files]
FIRST.EXE, STUFF.ZIP, 143000, 1
README.TXT, NOARCHIVE, 2, 23143, 0
*SYSTEMDIR\MYFILE.DLL, SYSTEM\, 23143, 1

Note: Use of the compression utility, PKCOMP.EXE, will generate a PKCOMP.INF file in the root directory where the compression is performed. This file will contain the above-mentioned section of Files which can then be cut and pasted into the products' SIERRA.INF file. Non-archived files (files needing no compression) can still take advantage of PKCOMP.EXE by providing a /X parameter. The /X parameter bypasses all compression activity and only produces the PKCOMP.INF file. This feature saves installers time by listing all non-archived files with their size.

FILENAME.EXE

FILENAME.EXE = file name
archive = Filename of the archive in which this file resides. For uncompressed (files which are not in an archive) use the keyword NOARCHIVE.

disk = Number of the disk on which this file resides. **This must only be used with an uncompressed file using the NOARCHIVE keyword!**

size = Size of uncompressed file, in bytes.

flag = 0 if file should not be installed by default, 1 if it should (**Important:** see note in Archives section above).

groupId = Optional group identifier used to group files together for use with TOGGLEGROUPON command.

FIRST.EXE

FIRST.EXE in STUFF.ZIP is 143,000 bytes long and should be installed.

README.TXT

README.TXT, an uncompressed 23,143 byte file on disk 2, shouldn't be installed by default.

*SYSTEMDIR

MYFILE.DLL, an uncompressed 23,143 byte DLL in SYSTEM\ should be installed by default and into the system directory on the machine.

3.1.1.5 Dialogs

The Dialogs section contains pseudo-templates for custom dialog boxes. Execution of these dialogs is controlled via the Script section. Entries in this section do not conform to the Windows .INI standard.

Setup allows the use of static text controls, check boxes, radio buttons, push buttons, and edit controls within a dialog. Combo boxes, list boxes, and other controls are not currently supported. These controls will be arranged automatically into a suitably pleasing configuration. Within a dialog definition, controls should be listed in the order in which they should appear in the dialog, top to bottom, with push buttons listed last. Grouped radio buttons must be listed consecutively. If you want more than one group of radio buttons in the same dialog, each group must be separated in the definition by a control which is not a radio button. Push buttons will always appear in the upper-right corner of the dialog box, stacked top to bottom in the order in which they appear in the definition. To ensure the dialog's layout is as pleasing as possible, push buttons must be listed last to allow them to be horizontally aligned properly.

You should make sure to have at least an OK or YES button in every dialog. If the OK button is pushed (or the YES button, which is treated separately but usually results in similar actions), the user's selections from the dialog are acted on. The NO button causes the dialog to end without altering Setup's behavior. **Important**: **No files or flags are ever set as a result of the NO button being pushed, regardless of whether or not you specify them in the dialog definition. The NO button essentially causes Setup to continue as if that dialog had never appeared in the first place.** The CANCEL button aborts the Setup process completely.

The installation team can associate a group of files or archives with a given control. If that control is selected, the associated files/archives are toggled "on," meaning they will be installed to the user's system. Additionally, a flag (see Script section) can be associated with a control. The flag will be set if the control is selected.

No text is specified in dialog definitions. To allow for language independence, keynames are used instead. The actual text represented by these keys is defined in a LANGUAGE.INF file (see Section 3.1.2).

[Dialogs]

BEGIN *template, label*

titlekey
type1, textkey1[*, (files1)*[*, flag1*]]files1_flag1
or

type1, textkey1, edit1
type2, textkey2[*, (files2)*[*, flag2*]]
...

typeN, textkeyN[*, (filesN)* [*, flagN*]]

END

Example 1
Example 2

Note: If you are not toggling any files, but want to set a flag, specify a blank file list with () - left parenthesis followed immediately by right parenthesis.

BEGIN

template = Integer ID # of Setup dialog template to use. Currently this is read but not used.
label = Dialog's name. Unseen by user, needed to invoke dialog in the Script section.

titlekey

titlekey = Keyname for text to appear in title bar.

type1, textkey1[, (files1)[, flag1]]

This first form is used for all controls other than Edit (6).
type1 = Type of the first control.

1: Static text.
2: Check box.
3: Radio button.
4: Single-selection list box (not currently supported).
5: Multi-selection list box (not currently supported).
6: Edit control (must use second form, without *files*).
10: OK button.
11: CANCEL button.
12: YES button.
13: NO button.
14: HELP button.
15: RETRY button.

textkey1 = Keyname for text to appear in control.

files1 = Comma-separated list of filenames representing which files and archives will be toggled on if this control is selected. Filenames should be specified identically to their specifications in the Archives or Files section.

flag1 = Flag to set if this control is selected. Flags MUST be of the form FLAGn, where n is a value from 0-9.

type1, textkey1, edit1

This second, shorter form is used for edit controls.

type1 = Type of the first control.
6: Edit control (in this case, without *files*).

*textke*y = The edit control's default value in this case.

edit1 = Used only for edit controls, specifies an identifier for that edit control. This is of the form EDITn where n is a value from 0-4.

Dialogs - Example 1

```
BEGIN 1, Test1
TESTDIALOG
1, CHOOSETYPE
3, SMALL, (FOO.CMP, BAR.EXE, MYFILE.TXT)
3, MEDIUM, (FOO.CMP, BAR.EXE, MYFILE.TXT, MORE.CMP)
3, LARGE, (FOO.CMP, BAR.CMP, MORE.CMP, YETMORE.CMP), FLAG3
6, DEFAULT, EDIT0
10, OK
11, CANCEL
END
```

The above template creates a dialog called Test1. The title bar text will be whatever TESTDIALOG is defined as in the appropriate LANGUAGE.INF (see Section 3.1.2). The first control is a static text control. The next three controls are radio buttons causing the indicated files and archives to be toggled on (and later installed) if the button is selected. If the third radio button is checked, FLAG3 is also set. The fourth control is an edit control which begins with whatever DEFAULT is defined in LANGUAGE.INF as its contents.

Dialogs - Example 2

```
BEGIN 1, MBox
VFW
1, PROMPT
12, YES, (VFW.ZIP), FLAG1
13, NO
11, CANCEL
END
```

The above example brings up a message box with YES NO CANCEL buttons. If YES is pushed, VFW.ZIP is toggled on. Since this is a ZIP file, it won't be uncompressed by Setup. Toggling VFW.ZIP on merely allows Setup to correctly determine the amount of space required on the user's system. FLAG1 will probably be used later in the script to run an external program which will install the files contained in VFW.ZIP.

3.1.1.6 Setup

The Setup section contains information needed by Setup or Presetup to run correctly. Both entries that address the size of Setup and its Billboard files are mandatory. This ensures that Setup has enough room on the users hard drive to run. Included in this section is AnimateDLL. This keyword specifies the filename that will be used to perform animation during the copying process.

2.X Note
To make it easier to determine what has happened in a script gone wrong, a simple debugging facility has been added. In addition, this section is used to retrieve information needed to update the Win95 registry in conjunction with the DOWIN95REGISTRY command.

[Setup]

SetupSize=*size*

BillboardSize=*size of billboards*

AnimateDLL=*filename.DLL*

CanInstallDOS=No

Debug=True

RegProductKey=*key*

RegKey*n*=*keyname*

RegValue*n*=*value*

InstallType=UninstallOnly
SkipReplaceWarning=True

Example

SetupSize

size = Uncompressed size, in KB, of maximal set of files contained in SETUP.SOL which could get installed to the user's system (_SETUP.EXE + SETUP32.EXE + MIDITEST.MID + the largest SOL_XXX.DLL and SOL_XXX.HLP files).

BillboardSize

size of billboards = Size of uncompressed billboards, in KB.

AnimateDLL

filename = Name of the animation DLL used during the copying process.

CanInstallDOS

Specify this entry if you do not have a DOS version of the game available on the same media. This prevents setup from recommending that the user install DOS version of the game if the users machine fails to meet the minimum requirements set forth in the Requirements section.

Debug

Specify this entry if you want a dialog to show up as each command in your script is parsed. This is to aid in debugging your script and should not be left in the shipping version. Note that clicking on the cancel button in the dialog stops it from appearing again through the remainder of the script.

RegProductKey

key = The name under which all other registry items will be placed. The convention is that a key called **HKEY_LOCAL_MACHINE\\SOFTWARE\\SierraOnLine*key*** will be written to the Win95 registry. Any other entries placed under this key, and this is where the game should search for entries.

RegKey*n*

n = Integer counting up from 0.

keyname = Name of the key that will be created under **KEY_LOCAL_MACHINE\\SOFTWARE\\SierraOnLine*key*** (see above). To support pairs of Keys and Values that will be written into the Win95 registry by setup.

RegValue*n*

n = Integer counting up from 0.

value = Value that will be entered for associated key **HKEY_LOCAL_MACHINE\\SOFTWARE\\SierraOnLine*key*** (see above). To support pairs of Keys and Values that will be written into the Win95 registry by setup. The special values *SIERRADIR, *DESTDIR, and *SOURCEDIR are currently supported here. If the value evaluates to a numeric value, it will be written out as such. If not, it is written as a string.

InstallType

Specify this entry if you want setup to ignore the SIERRA.INF file when it is searching for installed products to install or register. This will only be considered a product if uninstall is requested. This feature was specifically created to support Sierras collection series and should be used with extreme care.

SkipReplaceWarning

Set to True if you want setup to bypass the warning issued when a previous installation is found. Note that installing a game over a previous installation could cause the loss of saved games, etc.

Setup - Example 1

2.X Note
Since registry support is new, an example is provided. Given the following entries:

```
[Setup]
RegProductKey=Thexder
RegKey0=InstallDir
RegValue0=*DESTDIR
RegKey1=String
RegValue1=XYZ
RegKey2=Numeric
RegValue2=777
```

and the presence of the DOWIN95REGISTRY command in the Script section, the following will be written to the registry:

1) A key called HKEY_LOCAL_MACHINE\\SOFTWARE\\SierraOnLine\\Thexder
2) A key under 1) called InstallDir with a value like C:\SIERRA\THEXDER
3) A key under 1) called String with the value XYZ
4) A key under 1) called Numeric with the value 777

3.1.1.7 Script

The Script section (it must start with a capital S) is what drives Setup. Commands will be executed in the order in which they appear in the script. When an action label is invoked, the corresponding action will only be executed if that action has previously been toggled on by a dialog.

[Script]

:label

ADDTOINI *file, section, key, value*

ADDPROGMANGROUP *titlekey*

ADDPROGMANITEM [/F] [/D] [/R] *command, titlekey [,[icon][,workdir]]*

APPEND *filename text*

COLORS_NEQ *num* FLAGx

COPY

DATECHECK *filepath date* FLAGn

DIALOG *label*

DISKSPACE_LT *num* FLAGx

DOWIN95REGISTRY

END

EXIST *fileSpec* FLAGx

FLAG*n command*

GOTO *scriptlabel*

INSTALLDIRECTX

LANGUAGE_EQ *lang* FLAG*n*

NOTWINNT FLAG*n*

PHYSICALMEM_LT *memK* FLAG*n*

PICKDEST [*scriptlabel*]

REBOOTSYSTEM

REGISTER

RESETFLAGS

RESTARTWINDOWS

RUN [NOWAIT]

TESTMIDIEX FLAG*n*

TOGGLEGROUPON *groupid*

TOGGLEON (*file1* [*, file2, ... filen*])

VERSIONCHECK *filepath version* FLAG*n*

WIN32CHECK *version (file1, file2... file2),* FLAG*n*

WINGPROFILE

WINDISKSPACE_LT *num* FLAG*n*

WRITE *filename text*

Special Values

Example

:label

label = Name of label preceded by a colon. Defines a script label. Other commands can refer to this label to jump script execution to the label.

ADDTOINI *file, section, key, value*

file = Name of file to add to (see Special Values).

section = Name of section within INI file.

key = Name of key.

value = Value of key (see Special Values). Adds the given *key* and *value* to the given *section* of the indicated .INI file. If *key* already exists in *section,* its value is set to *value* instead.

ADDPROGMANGROUP *titlekey*

titlekey = Keyname of group title. Creates a program group of the given *title* value if none already exists with that name. This command **must** come before the first ADDPROGMANITEM command.

ADDPROGMANITEM [/F] [/D] [/R] *command, titlekey [, [icon] [,workdir]]*

/F = FORCE. Add item regardless of whether or not the path is valid.

/D = DELETE. Removes item at uninstall even if it is in SIERRA directory. Default is not to delete.

/R = RETAIN. Retain item at uninstall unconditionally.

command = Command to give the program item (see Special Values).

titlekey = Keyname of title to give program item.

icon = Icon name which may be an icon not embedded in the executable.

workdir = Working directory to be specified in the properties section of the program item. Creates a program item with the given *command* and *title* in the program group created by a previous ADDPROGMANGROUP command, provided the file and path specified in *command* exist. Must be preceded by an ADDPROGMANGROUP command earlier in the script.

APPEND *filename text*

Appends the specified *text* to the *filename.* Special Values may be used in *filename.*

COLORS_NEQ *num* FLAG*n*

Tests whether user is running a display with *num* colors.
num = Number of colors to test for.
FLAG*n* = Flag to turn on if number of colors does not equal *num.*

COPY

Copies all "on" files to the user's system.

DATECHECK *filepath date* FLAG*n*

Checks date in specified file.
filepath = Path of file to check. May include Special Values.
date = Date threshold. Must be in the following format: MM/DD/YY.
FLAG*n* = Flag to turn on if *filepath*'s date is less than *date*.

DIALOG *label*

label = Name of dialog box as defined in the Dialogs section. Invokes the named dialog.

DISKSPACE_LT *num* FLAG*n*

Tests that the destination drive has *num* free space available.
num = Amount of memory in K.
FLAG*n* = Flag to turn on if space on destination drive is less than *num.*

DOWIN95REGISTRY

WIN95 Only. Process the registry info under [Setup] in Sierra.INF (see Requirements section for details).

END

Terminate Setup. An End command **must** appear at the end of the script.

EXIST *fileSpec* FLAG*n*

Tests for a file matching *fileSpec.*
fileSpec = File specification, wildcards (*, ?) allowed.
FLAG*n* = Flag to turn on if *fileSpec* is found.

FLAG*n command*

n = 0-99

command = Any legal script command. If given flag was set by an earlier dialog, *command* is executed.

GOTO *scriptlabel*

scriptlabel = Name of label defined elsewhere in the script, without the colon. Script execution jumps to *scriptlabel*. This will usually be used with a FLAG.

INSTALLDIRECTX

WIN95 Only. Runs the Game SDK for Win95 install program. Note: dsetup.dll, dsetup16.dll and the DirectX folder must be on the source media. These are the contents of the Redist folder on the Game SDK.

LANGUAGE_EQ *lang* FLAG*n*

Tests if the machine is running the *lang* specified.
lang = Language - one of [ENGLISH|FRENCH|GERMAN|SPANISH|ITALIAN].
FLAG*n* = Flag to turn on if *lang* is language on machine.

NOTWINNT FLAG*n*

FLAG*n* = Toggles the flag on if not running under WinNT or Win95.

PHYSICALMEM_LT *memK* FLAG*n*

Tests the physical memory available.
memK = Amount of memory to test for in K.
FLAG*n* = Flag to turn on if physical memory is < *memK.*

PICKDEST [*scriptlabel*]

scriptlabel = Name of label defined elsewhere in the script, without the colon. Asks user to select a destination directory. If there isn't enough disk space on that drive, the script jumps to *scriptlabel*. **Important**: If your script doesn't allow users to choose a subset of files to install before it calls PICKDEST, *scriptlabel* should be a label appearing at the end of the script. Otherwise, an endless loop could result. If your script uses a dialog to allow the user to choose a file subset, that dialog must always be invoked **before** the PICKDEST command, and *scriptlabel* should be a label appearing before that dialog (see example).

REBOOTSYSTEM

Reboot user's computer.

REGISTER

Calls up the on-line registration portion of setup.

RESETFLAGS

Resets all flags to 0.

RESTARTWINDOWS

Restart Windows.

RUN [NOWAIT] *disk searchfile commandline*

Conditionally executes the specified command line.

[NOWAIT] = Optional parameter that causes setup to continue executing without waiting for executed file.

disk = Disk number on which *file* is expected.

searchfile = File which much exist in order to run command.

commandline = Command line to execute. If *searchfile* is found, executes the given command, suspending Setup until the launched application terminates. If *searchfile* is not found, prompts the user to insert disk *#disk*. If *disk* is 0, *commandline* is executed without first looking for *searchfile*.

TESTMIDIEX FLAG*n*

FLAG*n* = Flag to set if the user has extended MIDI. Plays an extended MIDI file and asks user if he heard it. If so, sets FLAG*n*. If not, plays a base MIDI file and asks user if he heard it. If not, tells user what may be wrong.

TOGGLEGROUPON *groupid*

Toggles the file in *groupid* on.
groupid = ID specified as last param of file in [Files] section.

TOGGLEON (*file1 [, file2, ...filen*])

Toggles the listed files "on" for installation. Special Values should **NOT** be used in file names.

VERSIONCHECK *filepath version* FLAG*n*

filepath = Path of file to check. May include Special Values.

version = Version threshold. Must be in the following format: w.x.y.z (e.g., 1.0.3.4) as used in Microsoft VERSIONINFO resources.

FLAG*n* = Flag to turn on if *filepath*'s version is less than *version*.

WIN32CHECK *version (file1, file2...filex),* FLAG*n*

version = Version of Win32s to check for.
(file1, file2... filex) = List of files to toggle on if correct version - use () if none.
FLAG*n* = Flag to turn on if Win32s version is >= version.

WINGPROFILE

Run the WinG profile utility.

WINDISKSPACE_LT *num* FLAG*n*

Tests that the Windows drive has *num* free space available.
num = amount of memory in K.
FLAG*n* = Flag to turn on if space on destination drive is less than *num*.

WRITE *filename text*

Writes the specified *text* to the *file.* Special Values may be in *filename*.

Special Values

Special values have been defined which, when used in conjunction with *file* or *value* parameters of ADDTOINI or the *command* parameter of ADDPROGMANITEM, are replaced with other values. These replacements occur in place, leaving the rest of the command line untouched. All values begin with an asterisk (*). These values are:

*DESTDIR

*SOURCEDIR

*SOURCEDRIVE

*CDROMDRIVE

*EDIT*n*

*SIERRADIR

*WINDOWSDIR

*SYSTEMDIR

*DESTDIR, *SOURCEDIR, *SIERRADIR, *WINDOWSDIR, and *SYSTEMDIR do not include trailing backslashes. *SOURCEDRIVE and *CDROMDRIVE are a single character (the drive letter) without a colon or backslash.

WARNING: Caution should be used when using *SIERRADIR and *DESTDIR. These parameters are substituted on the fly so that if the destination is not yet determined, these values are NULL.

*DESTDIR

Full path of the directory to which the user is installing.

*SOURCEDIR

Full path of the directory from which the user is installing.

*SOURCEDRIVE

Drive letter from which the user is installing.

*CDROMDRIVE

Drive letter of the first CD-ROM drive on user's system.

EDIT*n

n = 0-4 (text most recently entered in the corresponding edit control). Ex: *EDIT2.

***SIERRADIR**

The Sierra directory (i.e., C:\SIERRA).

*WINDOWSDIR

Full path of the users Windows directory.

*SYSTEMDIR

Full path of the user's Windows system directory.

Script - Example

```
[Script]
:Start
DIALOG PickFiles
PICKDEST Start
COPY
FLAG0 VFWSETUP.EXE
ADDTOINI SIERRA.INI, Config, File, *DESTDIR\RESOURCE.CFG
END
```

The above script asks the user to pick a set of files to install, then gets the destination directory. If there isn't enough space on the destination drive for all selected files, the user is returned to the PickFiles dialog. Once a file set is chosen which fits on the destination drive, files are copied to the user's system. If FLAG0 was turned on during the PickFiles dialog, VFWSETUP.EXE is executed. A line is added to the Config section of the SIERRA.INI file. *DESTDIR is replaced with the actual destination path. If the product installed to C:\SIERRA\FOO, the line would read "File=C:\SIERRA\FOO\RESOURCE.CFG". Finally, the script ends.

3.1.1.8 Preview

Not implemented.

The Preview section in the SIERRA.INF file is used to install (if not already there) an icon to the Sierra program group, and launch a CD-ROM catalog program being built by an outside vendor. This catalog program will preview Sierra's products along with various marketing and sales information. The catalog has not yet been delivered and we have no specs on it, so this section is currently unimplemented.

3.1.1.9 Billboards

The Billboards section in the SIERRA.INF is used to specify data that is displayed to the user when the copying process is going on. Various information can be passed along to the user like "Have you filled out your Warranty Registration card today?" or may provide the user with helpful tips for a successful execution of the newly purchased product. The developer can also specify bitmaps that will be displayed when the copy process is humming along. The bitmaps are read as "device independent bitmaps" (DIBs) or just plain BMP files. The Setup program distinguishes between text files and bitmaps by identifying the extension of the file. These files are located on the installation disk. If the install is from a floppy, the billboard files are located on the first floppy disk compressed into BILLBRD.SOL. If the install is from a CD-ROM drive, then the files are uncompressed and are in a BILLBRD subdirectory on the CD. There will be one BILLBRD.SOL or BILLBRD subdirectory for every language included on the disk, and they will be located in that language's subdirectory (see Section 3.1.2). The billboards are displayed according to their corresponding percentage numbers.

[Billboards]
percentNum=filename.xxx

Example

Billboards - Example

An example of a typical install with billboards is as follows.

[Billboards]
0=WARRANTY.TXT
15=OUTPOST.BMP
25=WARRANTY.TXT
40=UPCOMING.TXT
75=PROMO.DIB
90=WARRANTY.TXT

This should allow the developer flexibility to show screen shots and text while the copying process is running.

3.1.2 The LANGUAGE.INF File

To allow for greater language independence, all language-specific information for a script appears in a separate file. This allows all languages to use identical scripts, but with different text, thus simplifying the internationalization process. Each language provides its own LANGUAGE.INF file in a language-specific subdirectory on the distribution disk. Supported subdirectories are ENGLISH, FRENCH, GERMAN, SPANISH, and ITALIAN. These subdirectories also contain the billboards to be displayed when installing with that language. The language used by Setup is determined by the user's sLanguage setting in the Intl section of the WINDOWS.INI file. For a product consisting of two files (PRODUCT.EXE and PRODDATA.DAT) shipping with English, French, and German versions on the same disk, the disk might look like this:

```
SETUP.EXE
SETUP.SOL
SIERRA.INF
PRODUCT.EXE
PRODDATA.DAT
[ENGLISH]
        LANGUAGE.INF
        [BILLBRD]
                MESSAGE.TXT
                GRAPHIC.BMP
[FRENCH]
        LANGUAGE.INF
        [BILLBRD]
                MESSAGE.TXT
                GRAPHIC.BMP
[GERMAN]
        LANGUAGE.INF
        [BILLBRD]
                MESSAGE.TXT
                GRAPHIC.BMP
```

The LANGUAGE.INF file has two sections.

3.1.2.1 Identification

The [Ident] section of the LANGUAGE.INF contains language-specific identification information.

[Ident]

Title=*application title*

ShortTitle=*short title*

DirName=*directory*

Title

application title = Full title of the application (e.g., King's Quest VI: Heir Today, Gone Tomorrow).

ShortTitle

short title = Shorter version of the title. Can be the same as *application title*, but it's best to use a shorter title if possible (e.g., King's Quest VI).

DirName

directory = Name to use for application's subdirectory (e.g., KQ6).

3.1.2.2 Strings

The Strings section contains definitions for all the string keys used in the SIERRA.INF file. These definitions follow the format for .INI file string definitions:

[Strings]
Keyname=value

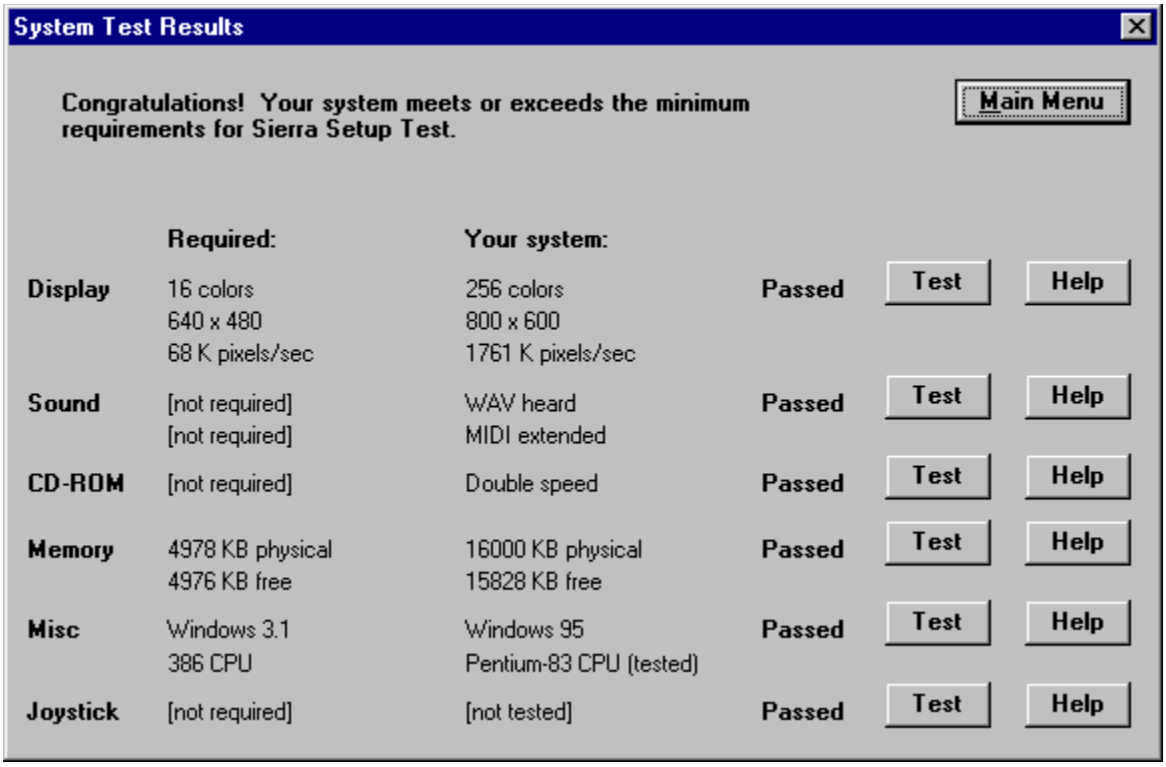
Example:
[Strings]
OK=OK
CANCEL=Cancel
PROMPT=Do you want to install Video for Windows?

Keyname

Keyname = Keyname used in SIERRA.INF file.
value = Text to display.

3.2 TEST - The System Test Results Dialog Box

Selecting this option brings up the System Test Results Dialog Box:



Options which don't apply to the current product (e.g., for products which don't use sound) will appear with [not required]. Selecting a button engages the corresponding hardware test. As each test is performed, the results are written to a SIERRA.INI file in the user's Windows directory. This information will thus be available to Setup and other Sierra applications using the standard Windows PrivateProfile interface. A context sensitive help button is available for each test which explains the test to the user and may help them troubleshoot why they are not passing a test. Refer to the Requirements section for details on setting the values used when performing these tests.

3.2.1 Display

Testing the display involves checking the user's color depth (number of colors) and resolution against the minimums required by the product, as indicated in the SIERRA.INF file. Setup displays the detected number of colors and resolution. If these are sufficient, Setup so informs the user and continues. If these are insufficient, Setup informs the user of the minimum requirements, suggests how to fix the problem, and returns to the System Test Results dialog.

2.X Note
A video speed test has been added that determines the speed of the video card in K pixels per second.

The user's SIERRA.INI file will receive the following information:

[Config]

ScreenWidth=*screen width*

ScreenHeight=*screen height*

Colors=*number of colors*

VideoSpeed=*speed*

ScreenWidth

screen width = Width of the users display, in pixels.

ScreenHeight

screen height = Height of the user's display, in pixels.

Colors

number of colors = Number of colors supported by the current display (2, 16, 256, 64K, etc.).

VideoSpeed

speed = The tested speed of the video card, in K pixels/second.

3.2.2 Sound

Testing the sound card involves making sure the user can play .WAV and .MID files. Setup scans the user's system for a device capable of playing .WAV files. If none is found, the user is informed and Setup returns to the Test Hardware dialog. If a .WAV device is found, Setup plays a .WAV file and asks the user if he heard it. If he answers NO, Setup suggests some possible causes (volume turned down, speakers off or unplugged, etc.) and returns to the Test Hardware dialog. On success, the above process is repeated for MIDI devices and a MIDI file.

After successful completion of the sound card tests, the user's SIERRA.INI file will contain the following entries:

[Config]

SoundBitSupport=0 | 8 | 16 | -1

SoundChannels=1 | 2

MIDI=0 | -1 | Error code

If a sound test fails, the SIERRA.INI file will contain all information gained up through the failed test. If the user's system didn't hang up, information from the failed test will also appear.

SoundBitSupport

0 = Sound card found.
8 = 8-bit support.
16 = 16-bit support.
-1 = No sound card found.

SoundChannels

1 = Mono.
2 = Stereo.

MIDI

0 = MIDI support.
-1 = No MIDI support.
Error code = MCI error value generated when Setup tried to play a MIDI file.

3.2.3 CDROM

The Setup program tests any/all CD-ROM drives located on the users system. The test determines the data transfer rate of a users drive. Although testing a device driver in Windows via a timer can give varying results, the Setup only determines if the drive is a single, double, triple, or quadruple speed drive. It performs this test by accessing a CD in the drive and attempts to perform data transfer commands while running a timer.

After successful completion of the CD-ROM tests, the user's SIERRA.INI file will contain the following entries:

[Config]

CDROM= 1 | 2 | 3 | 4

CDROM

1 = Single speed drive.
2 = Double speed drive.
3 = Triple speed drive.
4 = Quadruple speed CD-ROM drive.

3.2.4 Memory

Testing memory involves checking the user's current amount of free memory to determine whether or not there is enough to be able to run the program being installed. Setup tells the user how much free memory it found and, if it's not enough, how much is needed and how the user might acquire the additional memory. Setup reports both physical and free memory and the user must have enough of both to pass the test. This means that if the user is running a lot of applications when he tries to install, he may not be able to do so even though his system as a whole has enough memory to run the program at another time.

The user's SIERRA.INI file will receive the following information:

[Config]

FreeMem=*freeKB*

PhysicalMem=*freeK*

FreeMem

freeKB = Amount of free memory on system, in KB.

PhysicalMem

freeK = Amount of physical memory on system, in KB.

3.2.5 Miscellaneous

This section tests whether the user is running the appropriate version of Windows and whether his CPU meets the minimum recommended configuration in terms of CPU and clock speed.

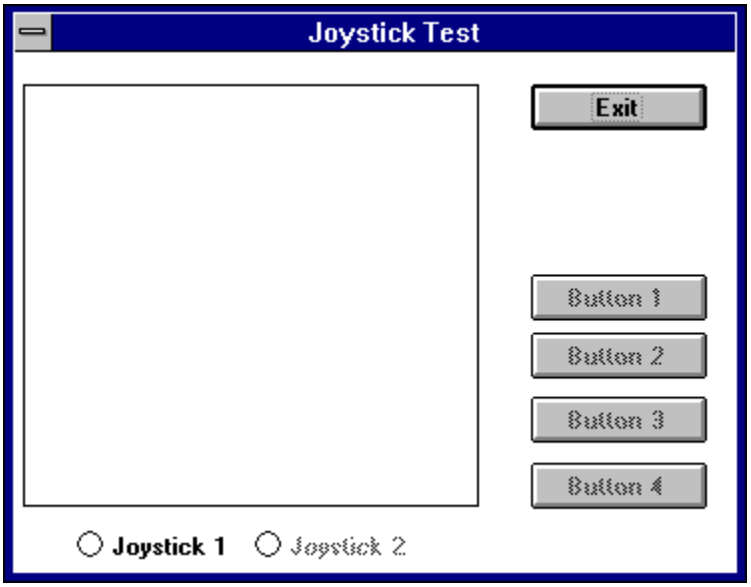
The user's SIERRA.INI file will receive the following information:

[Config]
WinVer=395
CPU=Pentium-90 CPU

3.2.6 Joystick

Testing the joystick is a multi-step process. If any of the detection steps fail, the user is informed and returned to the System Test Results dialog. First, Setup verifies that the joystick driver has been installed, then it detects whether or not a joystick is currently connected to the system.

If a joystick is detected, the Joystick Test dialog will appear to allow the user to test his joystick. There is an opportunity to test two joysticks; the user can choose a joystick and test it by pressing the appropriate button. Currently Setup does not perform any calibration of the joystick. Calibration of the joystick will most likely be different for various products. It will be up to the program to calibrate the joystick for its own purposes.



The big square will be black with a white crosshair. The crosshair moves when the joystick moves. Pressing a button on the joystick causes the corresponding dialog button to depress.

When the joystick test concludes, the SIERRA.INI file will receive the following info:

[Config]

Joystick= -1 | 0 | 1 | 2 | 3

Joystick

-1 = Drivers detected but no joystick found.
 0 = 0 joysticks detected.
 1 = Joystick 1 detected.
 2 = Joystick 2 detected.
 3 = Joysticks 1 & 2 detected.

3.2.7 Printer

No longer supported.

Testing the printer involves detecting whether or not a printer is connected to the user's system and if so, what printer it is and to which port it is connected. The user is informed of Setup's findings and returned to the System Test Results dialog.

The SIERRA.INI file receives the following information:

[Config]

Printer=*printer name*

PrinterPort=*printer port*

Printer

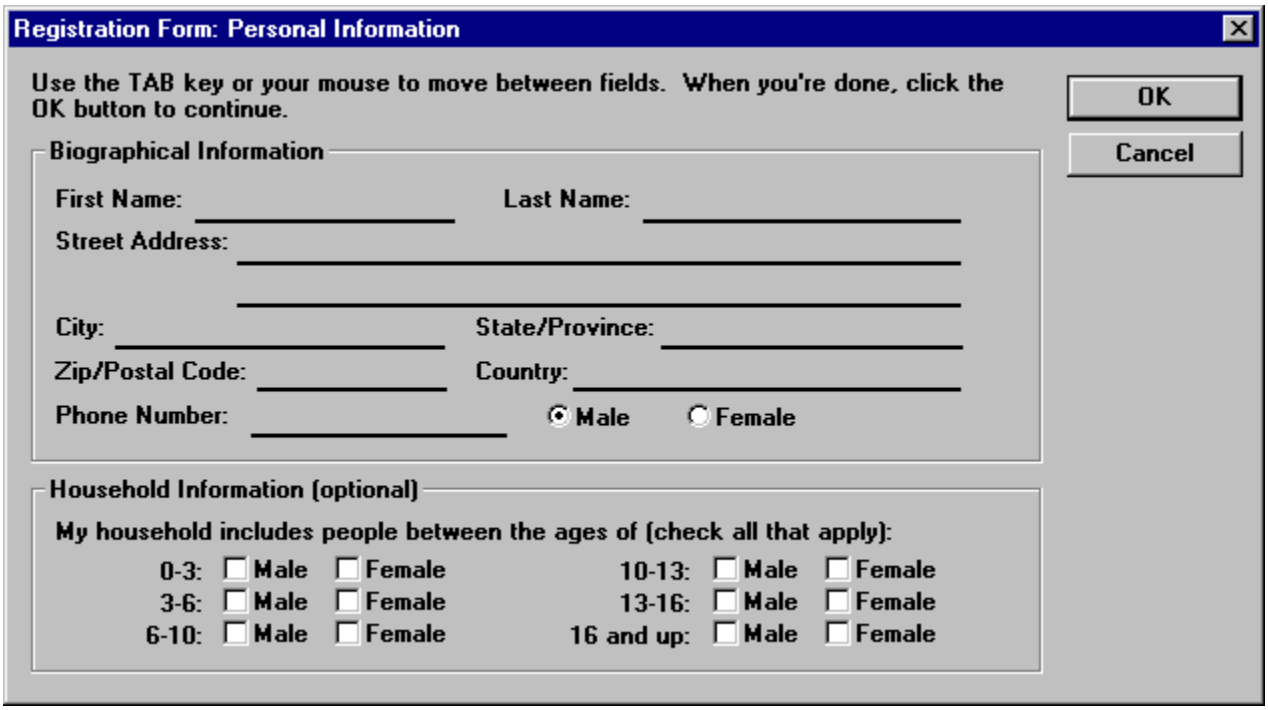
printer name = Name of printer.

PrinterPort

printer port = LPT1, LPT2, etc.

3.3 Register

Another intention of Setup is to encourage users to register their software product. The names, addresses and profiles of our users are very important to our marketing and sales staff. By providing a means for the user to input this information via the keyboard, we can anticipate an increased number of registered owners of our products. The selection of the Register button will invoke a dialog that will allow the user to input his name, address, and other valuable information as requested by marketing/customer service departments. This information is then captured to a text file, **register.txt**. A Print button will be provided on the registration form to allow the user to direct the data to a hardcopy output so the user may be able to mail it in. An example of the registration form is shown below.



2.X Note

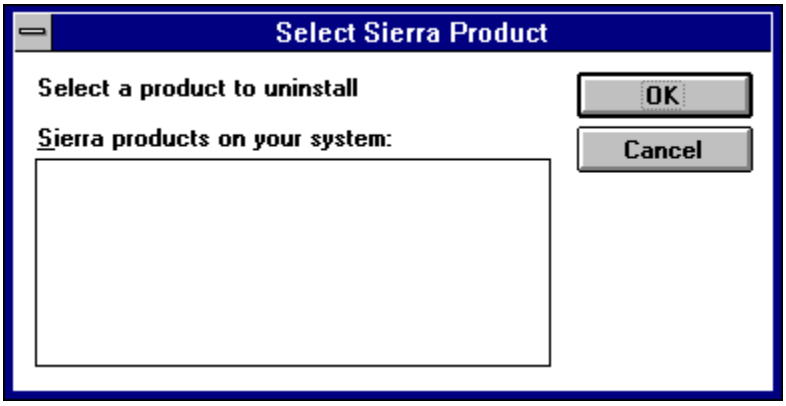
Modem registration is now supported. The user will be asked whether he chooses to register by modem or by printing out the data captured and mailing it in. If by modem, setup will detect the modem and dial up the Sierra BBS and upload the file with the information that was captured.

3.4 Read Me

This option executes the text processor WRITE to display the readme file. By default, looks to open a file called README.TXT in the same location as the LANGUAGE.INF file being used by Setup. The name of the file may be changed via a specification in the Ident section of the script. See Section 3.1.1.2 (Identification) for more details.

3.5 Uninstall

The Setup program will allow the user to uninstall a Sierra game on the users hard disk. This functionality is provided so the user does not have to go into File Manager or DOS and extract the game manually. When the user selects this option, a pop-up dialog will ask the user which of the Sierra products found on the users system he would like to remove. Upon selection of a product and clicking OK, all files from that Sierra directory will be deleted as well as the directory itself.



3.6 Support

The Support button brings up the Support page of the Setup help file. This page has links to pages describing the various means of accessing Sierra tech support, lists of sound card manufacturers and their phone numbers, lists of video card manufacturers and their phone numbers, and lists of other manufacturers with their phone numbers. In this subtle way, we hope to encourage users to contact people other than Sierra when the problem involves a hardware or software product not produced by Sierra.

3.7 Exit

Clicking on this button exits the main menu and ends the execution of Setup.

3.8 About Icon

By clicking on the About Icon (Half Dome Icon) in the Main Menu, it can be determined which version of Setup is being run. This information may be requested by the programmer responsible for setup to aid in trouble shooting.