Quick Install 1.1 for Windows

This distribution disk contains an executable and documentation that you can use to build an application installation utility to run under Microsoft® Windows™. The code used to write the program is a subset of the source code used to build the Windows retail Setup application.

This information consists of the following:

- This document, which provides information on how to use Quick Install to install your own applications.

The files are installed in directories as follows:

\QINSTALL (root)
The files in the root directory are the files you will want to look at for info on how to setup Quick Install.

\QINSTALL\DEMODSKS
The files in the DEMODSKS subdirectory will create for you a quick two disk demonstration of how the setup program works. Use MAKE.BAT to create the demonstration.

What is Quick Install??
Quick Install is a fast and easy install program for Windows 3.xx.
It has many powerful and useful features:

1) Quick Install is free...

2) Quick Install can install from many disks.

3) Quick Install can uncompress files compressed with COMPRESS.EXE

4) Quick Install allows users to choose what parts of the application to install.

5) Quick Install can create Program Manager groups for your application.

Trying the Sample Installation Utility

To get an idea of what a basic Quick Install-based installation utility looks like, run INSTALL.EXE.

The information file INSTALL.INF as provided on this distribution disk is designed to install the components of this distribution disk onto your system. It also includes comments that describe the various sections, and their formats. Another file, INSTALL.INF in the DEMODSKS directory is used to create the demonstration installation that you find in the DEMODSKS subdirectory.

Follow these steps to compile the sample installation utility and create a set of distribution disks for it to work with:

If you have not already done so, run the INSTALL program from the distribution disk to install this product onto your system. You can specify any destination location for this installation, by default, it is C:\QINSTALL.

You have already seen what the final product can do, because the installation of this distribution disk uses it to install itself. If you would like to create a 'demo' disk set that shows you another example, then go to the DEMODSKS directory and use MAKE.BAT to create a two disk demo of the install program. This will copy the installation program (INSTALL.EXE) and an information file (INSTALL.INF) and several dummy files to the floppy disks.

To test the example Install program, choose Run from the File menu in Program Manager, or your shell application. Specify A:\INSTALL (where A: is your floppy drive) for the name of the program you want to run.

To build a simple installation utility, follow these steps:

- Determine how much disk space your application requires.

- In INSTALL.INF, use the minspace statement to tell Install the minimum amount of space your application needs.

- Determine the number of distribution disks your application will use.

- In INSTALL.INF, change the information in the [disks] section so that it defines each distribution disk.

- Determine which files will be shipped on each disk, and the destination directory for each file (relative to the main installation directory).

- In INSTALL.INF, change the information in the [app.copy.full] section so that it defines the information for your application's files and destination directories.

- In INSTALL.INF, change the information in the [progman.grp] section so that it specifies your application's command line and icon.

- Copy the files INSTALL.EXE, INSTALL.INF, INSTALL.TXT and LZEXPAND.DLL to the root directory of your application's first distribution disk.

- Test Install thoroughly by running it in conjunction with your INSTALL.INF file and your application's distribution disks.

- Be sure to test Install in each of the two Windows operating modes (standard and 386 enhanced). Also, be sure to test all portions of the application and code.

Simplifying the Installation Process
As you work on the installation of your application, try to keep it as simple as possible. The simpler the installation process, the less likely the user will make mistakes that require him or her to repeat the entire process.

Design Disk Contents to Avoid Unnecessary Disk Swapping
If your application files come on several distribution disks, you might want to design the disk contents carefully so that the user does not have to swap disks repeatedly. The best approach allows the user to insert each disk once.

Keep the User Informed About What is Happening
The INSTALL.INF file lets you define strings that the utility will display while copying a particular file. You can use these strings to display either the filename or a description of that file.

Sections of INSTALL.INF
This section guides you through the creation of a simple INSTALL.INF file. In this case the example is for Resta 2.2, a windows screen saver written by Stefan Olson.

The file, INSTALL.INF, contains information that Install uses to install applications. This file must be in the same directory as the program file INSTALL.EXE. The two files can be on a floppy disk, a hard disk, or a network disk drive.

The information in this file determines

* The names of the disks and directories from which, and to which, Install copies files

* The name of the group that Install creates in Program Manager's window, and the names of program items that Install adds to that group

You can create your own Install program by changing some of the information in this file.

INSTALL.INF is similar in format to a Windows initialization (.INI) file.

Sections: INSTALL.INF is divided into sections. Each section is identified by an alphanumeric name enclosed in square brackets.

Most section names are "hard-coded": Install is written to search for in INSTALL.INF for that specific section name. However, some section names are defined within other sections, and are therefore easily customizable.

Comments: A comment begins with a semi-colon. You can include a comment on the same line as syntax, as long as it comes after the syntax.

Spaces: Spaces are ignored, except when between double quotes. Blank lines are also ignored.

[dialog]
caption = "Resta Install"

The [dialog] section contains information pertaining to Install's dialog box.

The Caption statement determines the text that appears in the title bar of Install's dialog box.

The Caption statement specifies the caption "Resta Install".
To specify your own caption, replace the text "Resta Install" with the title you want (enclosed in double quotes).

[data]

```
  defdir      = C:\WINDOWS\RESTA
  background  = blue
  fullinst    = 570k
  mininst     = 90k
  custinst1   = 90k
  custinst2   = 570k
```

The defdir specifies the default directory to install the program into. This will come up when the user chooses the directory. From there the user can change the directory to install the program into.
The background option can be blue, or anything else. Blue means the background fades like the Microsoft® Setup applications. Anything else, or nothing means there will be no background.
The fullinst option tells the install program how much disk space, in whatever format you like, a full install option will take up.
The mininst option tells the install program how much disk space, in whatever format you like, a minimum install option will take up.
The custinst1 option tells the install program how much disk space, at the least, in whatever format you like, a custom install will take.
The custinst2 option tells the install program how much disk space, at the most, in whatever format you like, a custom install will take.

[disks]

```
  1 =.,       "Resta Install Disk 1"
```

The [disks] section defines the distribution disks that contain the application
files. Install uses this information to tell the user to insert the correct disk.

Elsewhere within this .INF file, the distribution disks are normally referred to
by a single-character disk ID. This section defines those disk IDs, and includes

information about the disk to which each disk ID refers.

The disk ID '0' is reserved; it represents the installation directory -- the directory in which the user is installing the application.

The format of each disk definition is:

```
   n = path, title
```

where

n is the disk ID (a single character from 1-9 or A-Z).

path the path of the source directory from which Install should copy the files to the disk The path can be relative to the source directory (see examples below).

title is a descriptive name for the disk. The title should match the disk's printed or written label exactly.

The following is another example [disk] section:

```
1 =.,        "Demo Application Disk 1"
2 =.\files,  "Demo Application Disk 2"
```

The first statement tells Install to refer to Disk 1 as "Demo Application Disk 1". Because the period (.) denotes the current directory, the files on that disk will be copied from the root directory of the distribution disk.

The second statement tells Install to refer to Disk 2 as "Demo Application Disk 2"; the files that Disk 2 contains will be copied from the \FILES directory of the distribution disk.

You can include as many disk-definition statements as necessary. Every distribution disk should have a corresponding disk-definition statement; otherwise, Install cannot tell the user to insert the appropriate disk.

[needed.space]

```
  minspace  = 90000
  fullspace   = 600000
```

This shows the needed space in bytes for a full install and a minimum install. Bytes = kilobytes * 1000.

If the specified amount of space is not available,
Install will ask the user to specify a different hard disk, or exit Install.

[app.copy.full]

```
#app.main,  0:
#app.icons, 0:
#app.help,  0:
```

The [app.copy.full] section contains section-definition statements. Each statement defines a section that lists application files to be copied as part of installation. The sections are organized by file destination; you should define a separate section for each destination directory.

Each section definition has the following form:

```
#section_name, 0:dest_pathname
```

where

#section_name defines the name of the .INF section that lists the files to be copied.

0 is the disk ID that represents the installation directory. (0 is a reserved disk ID, and always represents the installation directory -- the directory the user specified when asked where to install the application.)

dest_pathname is the pathname of the destination directory, relative to the installation directory. For example, "0:FILES" represents the FILES subdirectory of the installation directory.

The previous section-definition statements define three sections, [app.main], [app.icons]
and [app.help]. All files listed in each section will be copied into the main installation directory.

[app.copy.min]

```
#app.main,  0:
```

The [app.copy.min] section contains section-definition statements. Each statement defines a section that lists application files to be copied as part of installation. The sections are organized by file destination; you should define a separate section for each destination directory.

See the preceding section for a description of the syntax for listing files.

The previous section-definition statements define one section, [app.main]. All files listed in each the section will be copied into the main installation directory.

[app.copy.cust]

```
#app.main,  0:, "Main files (90k)"
#app.icons, 0:, "Demonstration icons (3k)"
#app.help,  0:, "Help file and viewer (475k)"
```

The [app.copy.cust] section contains section-definition statements. Each statement defines a section that lists application files to be copied as part of installation. The sections are organized by file destination; you should define a separate section for each destination directory.

Each section definition has the following form:

```
#section_name, 0:dest_pathname, , "description"
```

where

#section_name defines the name of the .INF section that lists the files to be copied.

0 is the disk ID that represents the installation directory. (0 is a reserved disk ID, and always represents the installation directory -- the directory the user specified when asked where to install the application.)

dest_pathname is the pathname of the destination directory, relative to the installation directory. For example, "0:FILES" represents the FILES subdirectory of the installation directory.

description is the description of the option that will be displayed in the listbox where the users get to choose what sections to install. You should give an indication of the dive space each section takes up.

The previous section-definition statements define three sections, [app.main], [app.icons] and [app.help]. All files listed in each section will be copied into the main installation directory. Only the sections chosen by the user will be copied into the directory.

[app.main]

```
  1:RESTA.EX_,    "Resta executable file"
  1:RESTA.BU_,    "Resta Bug Form"
  1:INSTALL.BU_,  "Quick Install Bug Form"
  1:INSTALL.TXT,  "Information on Quick Install"
  1:README.TXT,   "Last minute information"
  1:RESTA.WR_,    "Documentation in Windows Write format"
```

This section is a user-defined section that lists files to be copied to a particular destination directory. The [app.copy.full], [app.copy.min] and [app.copy.cust] sections define the name of this section and the destination
directory of the files.

In each section like this one, you should list all files that you want copied
to the same destination. (For example, all the files in this section, [app.main],
will be copied to the installation directory.)

Install copies the files listed in this section in the order in which they are listed.

The syntax of each file listing is

```
  N:FILENAME, "Description"
```

where

N: is the disk ID of the disk that contains the file. (Disk IDs are defined in the [disks] section.) If the specified disk is not in the disk drive, Install prompts the user to insert it.

FILENAME is the name of the file, including the filename extension, if any.

Description is descriptive text that Install displays as it is copying the file or group of files. If you leave the description blank, Install will continue displaying the descriptive text from the previous file. This lets you use a general name for a group of files.

For example, the first statement below above Install to copy the file RESTA.EX_ from
Disk 1, and to display the descriptive text "Resta executable file" while

copying that file. As
explained above, the destination of the files in this section is determined by a
section-definition statement in the [app.copy.full], [app.copy.min] and [app.copy.cust] sections.

[app.icons]
Like the [app.main] section, this section is user-defined, and lists files to be copied to a particular destination directory. The [app.copy.full], [app.copy.min] and [app.copy.cust] sections define the name of this section and the
destination directory of the files listed in this section.
See the preceding section for a description of the syntax for listing files.

```
1:BUGS.IC_,     "Demonstration 16 colour Bugs Bunny icon file"
1:BUGS2.IC_,    "Demonstration 2 colour Bugs Bunny  icon file"
```

[app.help]
Like the [app.main] section, this section is user-defined, and lists files to be copied to a particular destination directory. The [app.copy.full], [app.copy.min] and [app.copy.cust] sections define the name of this section and the
destination directory of the files listed in this section.
See the preceding section for a description of the syntax for listing files.

```
1:RESTA.HL_,    "Resta help file"
```

[progman.groups]
```
 Utils,   utils.grp,   NO
```

The [progman.groups] section (optional) tells Install to create Program Manager groups for your application. (Quick Install then uses DDE to communicate
with Program Manager.)

The section lists the names of the groups you want to create. You then define additional sections in this file; those sections list the program items you want in each group.

The syntax for each group name is:

groupname, [groupfile.grp], [replace items]

where

groupname is the title you want Program Manager to display under the icon that represents the group. (The groupname will also be the name of the section that defines the contents of the group.).

groupfile.grp is the filename of the file in which Program Manager will save information about the group. (You must include the .GRP filename extension.) This parameter is optional; if you omit it, Install uses a default name for the group file.

replace items is whether or not to replace an item if the specified item already exists.

The previous group-definition statement tells Quick Install to create a group named Utils, store its information in a file named UTILS.GRP. The [Utils] section will contain information about the group's contents.

[Utils]
"Resta 2.2", RESTA.EXE

This section is a user-defined section that define the contents of a Program Manager group file. The [progman.groups] section defines. the name of this section and the group, and the name of the file in which to store information about the group.

Note: If you have spaces in the group name (e.g: Demo Utils) the title for the section needs to be [Demo.Utils]. That is each space needs to be a '.'.

In each section like this one, you should list all items that you want Install to add to the group.

The syntax for item-definition statements is:

"Description", APPFILE.EXE, [PARAMS], [ICONFILE.EXE[, N]]

where

Description is the text that will appear below the program icon when displayed in the Program Manager group.

APPFILE.EXE is the executable file that starts the application.

PARAMS is the command line parameters to run the application. This paramter is optional; if you omit it, Install will use a blank command line.

ICONFILE.EXE is the application file that contains the icon you want to represent the application. Typically, this is the executable application file, but it could be a different file. (You can also specify a .ICO file, created using the SDKPaint tool.) This parameter is optional; if you omit it, Install will use the first icon it finds in APPFILE.EXE.

N is the offset of the icon you want to use within the file ICONFILE.EXE. This parameter is optional; if you omit it, Install uses the first icon it finds in ICONFILE.EXE. You must include this parameter if the file you specify contains more than one icon, and you want to use an icon other than the first icon.

To use the Nth icon, specify the number N-1. For example, to use the third icon, specify 2.

For example, the previous item-definition statement tells Quick Install to add an item
titled "Resta 2.2" to Program Manager. The application command line is RESTA.EXE; the file that contains the application icon is the executable file,
RESTA.EXE

Compression:
Quick Install will recognise files compressed with the SDK's COMPRESS.EXE utility.
If a file is compressed with this utility and the file LZEXPAND.DLL is available, the
files will be decompressed and put the directory like any normal file would.

Needed Files:
If you use Quick Install you must ship the INSTALL.EXE file, INSTALL.TXT and
LZEXPAND.DLL, which is in your windows system directory. It has all the compression
and copying routines in it.

NOTE: You must leave LZEXPAND.DLL uncompressed, otherwise Quick Install will
be unable, unless the library is on the user's system to decompress and copy files.

Disclaimer:
Quick Install is distributed free. No warranty exists, either express or implied. No liability is assumed for any damage or loss resulting from the use of this program. No claims are made regarding the accuracy of this program. The author reserves the right to charge for future versions.

Copying info:
This program may be freely used and copied but may not be sold except for a nominal copying charge not to exceed US $10. This file should be included with all copies of Quick Install

Revision History:
1.0 First release version!! 5/11/92
1.1 Second release!! 20/8/93 Command Line parameters & minor bugs fixed

Future Enhancements:
Will allow the programmer to add a DLL to change dialogs, etc...
Will allow the programmer to specify a BMP file to display in any place on the screen.
Will allow the programmer to specify a file to read after installation.
Will allow the INF file to specify changes to INI files.
And anything else users suggest....

About the Author:
Stefan Olson is 16 years old and this is the second windows program he's written. He hopes to get a job as a programmer one day in the future (anyone looking for a young programmer??), and has many other windows programs waiting to be written. These include: a disk cataloguer , a windows unarchiver and archiver, a screen capture program with the ability to save the captured picture to .Bmp (anyone got any code to do this??), and icon viewer and editor... He has written Resta 2.2, a powerful windows screen saver.

How to contact the author:
The author can be contacted through the Internet at (don't type the brackets): (stefan@olson.acme.gen.nz)

The author can be contacted through compuserve at (don't type the brackets): (>Internet:stefan@olson.acme.gen.nz)

I'd appreciate it if you'd send me a mail message if you have any comments or suggestions...

If you find a bug please send me a mail message with a filled in bug report

Programs and books used to write this program:
Quick Install was written in C using Borland Turbo C++ for Windows 3.1.
Quick Install was degbugged using Borland Turbo Debugger for Windows 3.1
Quick Install's icons and dialogs were created using Borland Resource Workshop 1.02(comes with Turbo C++ for Windows)
Quick Install's documentation was written in Lotus Ami Pro for Windows release 2.0 and then saved to Microsoft Write format.

I used the Borland Windows API Reference and the Turbo C++ for Windows on-line help to find the correct commands.

I used Charles Petzold's "Programming Windows" (Microsoft Press, 1990) to learn windows programming.

Trademarks:
Turbo C++ for Windows, Turbo Debugger for Windows and the Resource Workshop are trademarks of Borland International.
Ami Pro is a trademark of Lotus Development Corporation.
Windows and Write are trademarks of Microsoft Corporation.
Microsoft is a registered trademark.

Acknowledgements:
Thanks to Microsoft Corporation for supplying the basic source to the program.
Thanks to Mum and Dad for letting me use the computer, even when they wanted to.
Thanks to Hamish Mackenzie for the invaluable help he has given me.