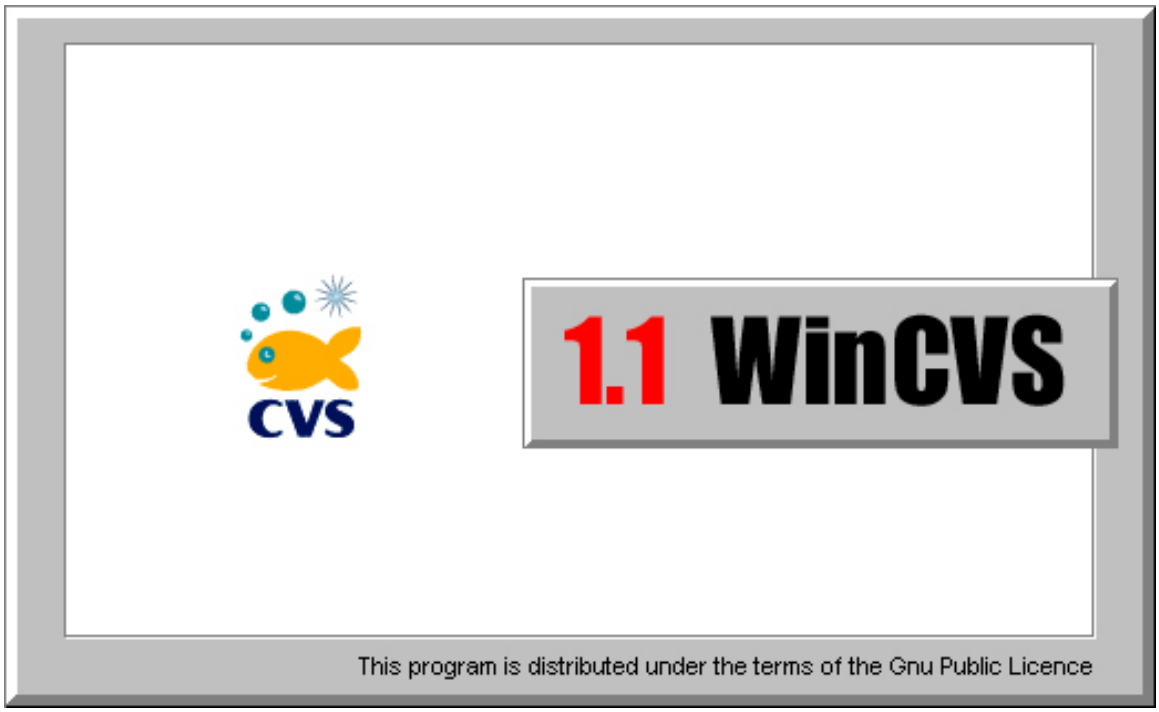
WinCvs Version 1.1

Users Guide



Don Harper

June 1, 1999

Section 1 – Introduction

CVS stands for Concurrent Versions System. It is a version control system that has been developed in the public domain by many people beginning in 1986. Currently, CVS is maintained by Cyclic Software in Washington, DC and there is lots of information on their web site at www.cyclic.com.

The biggest "problem" with CVS is that it uses a command line interface. Since most developers prefer a nice graphical user interface, several groups around the world have developed graphic front ends to the CVS core. The best one available for Windows NT/98 is WinCvs developed and maintained by a group of people around the world. Information about WinCvs is available on the web site at www.wincvs.org.

Note that the versions of cvs and WinCvs documented here differ slightly from the standard releases. The most visible difference can be seen in the examples of the import command. The unofficial patch called Main Branch Import has been applied to allow importing files directly to the trunk instead of a vendor branch. The patch can be found at http://www.cyclic.com/cvs/dev-trunk-import.txt.

This document describes the installation procedure for WinCvs and explains some of the basics required to get started using CVS. Please address all questions or problems with this manual to Don Harper (use the newsgroup for questions about cvs or WinCvs itself. [mailto:donharper@earthlink.net].

Note that this document assumes installation on a Windows NT computer but Windows 98 users should find it to be accurate with a few minor exceptions.

Section 2 – Installation Instructions

The current release of WinCvs 1.1 is still in beta test and is available from the WinCvs website at www.wincvs.org. To use the built-in command line interface Tcl/Tk 8.1.1 must also be obtained from www.scriptics.com.

To begin installing WinCvs, unzip the downloaded zip file to your favorite temporary folder. Locate and run the setup program (Setup.exe). Install the software to the folder Program Files\GNU\WinCvs 1.1 on a local drive (preferably C:).

To install Tck/Tk, run the file tcl811.exe downloaded from scriptics.com. Install to the default installation folder.

If you want a really good editor, install EmEditor (eme200ei.exe). It is shareware so you will need to pay $25 if you want to keep it. More information about EmEditor and licensing is available on the website at http://www.nifty.ne.jp/forum/femsoft/index-e.htm. You can also use any other editor that you prefer.

If you want a free file compare utility, install ExamDiff (examdf16.zip). If you want slightly more features install the shareware version ExamDiff Pro (edpro21c.zip) instead of ExamDiff. ExamDiff Pro is requires a $25 license fee. More information about ExamDiff and ExamDiff Pro is available on the website at http://www.nisnevich.com.

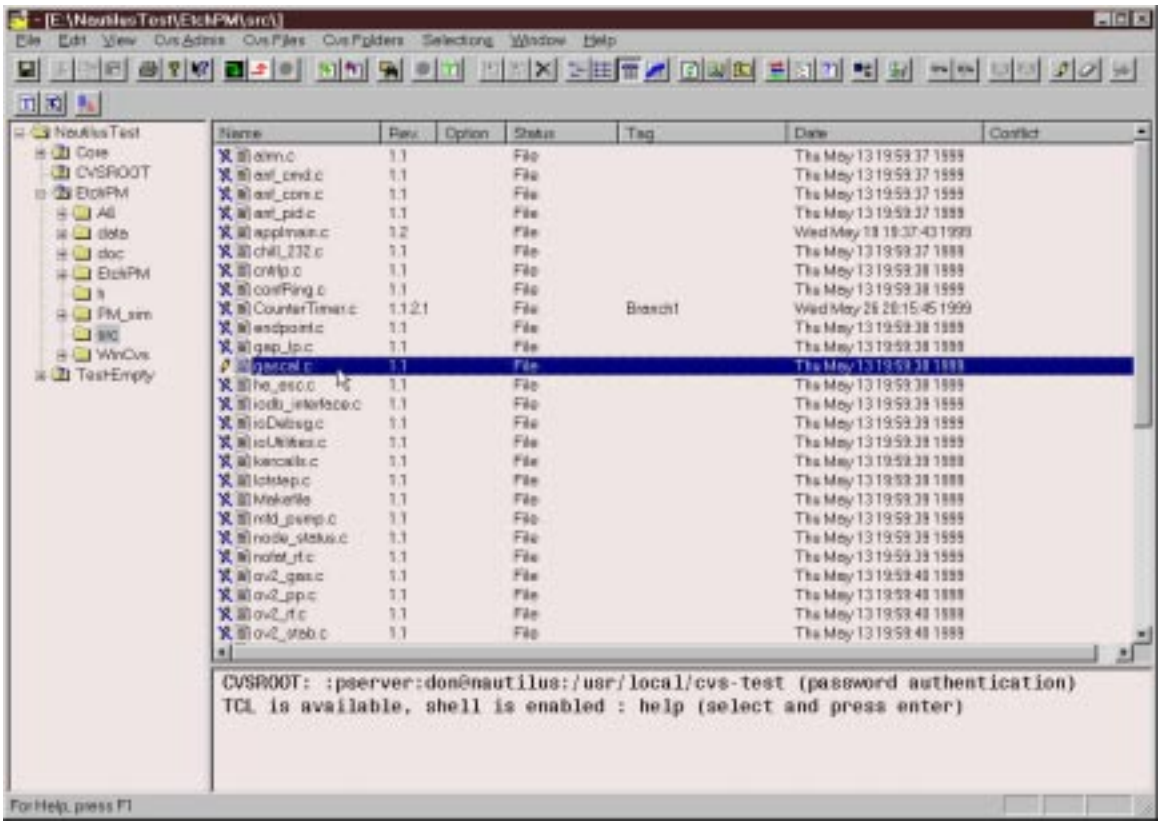
WinCvs will add itself to your start menu but you will probably want a shortcut to Program Files/GNU/WinCvs 1.1/wincvs.exe on your desktop.

Section 3 – Beginners Guide to WinCvs

3.1 Setting the Work Area Root Folder

Before running WinCvs, create a folder that you will use as the root of your local work area. Of course you can manage multiple work areas with WinCvs but it will make these instructions easier if you create your work area root first.

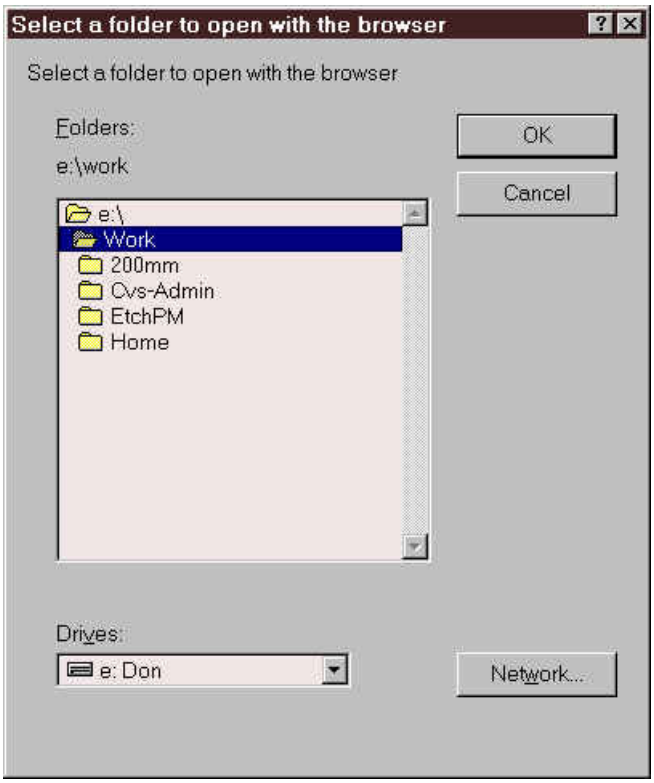
When you first run WinCvs, you will see the main browser window. In this example, an actual work area is displayed so the contents of the browser differ from what you will see when you first run WinCvs.



The first thing you should do is to change the browser's root folder to point to the folder you will be using as your work area root. The browser root can be changed either from the menu:
Cvs Folders->Macros folder->Change Root or from the "binoculars" icon on the toolbar:

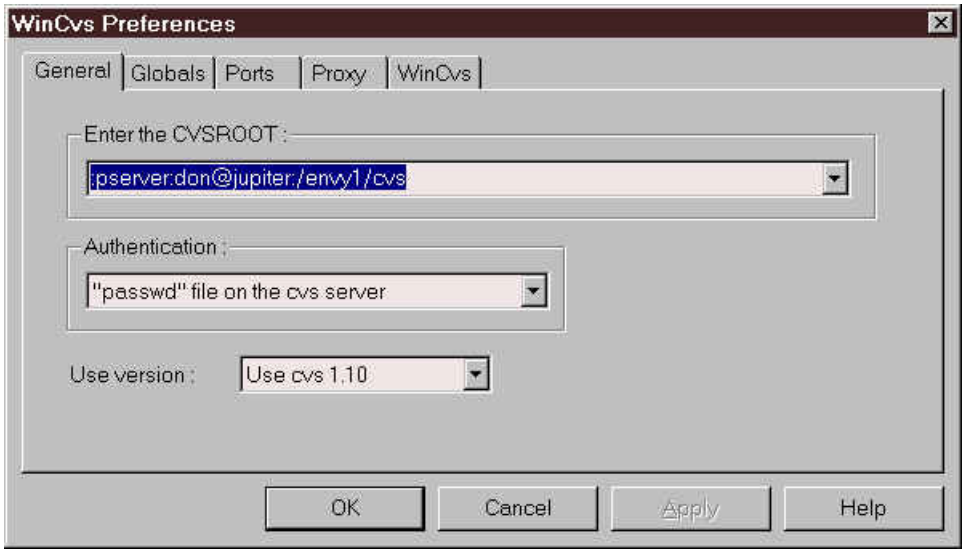
In either case, a panel will open where you can select the folder to open with the browser. Locate the folder you want to use as your work area root and double-click on the folder so the folder icon is open. If you just hilite the folder with a single mouse click, the parent folder will be selected instead of the desired folder. This is a bad feature (bug) all WinCvs commands that use this type of folder selection.

In the following example, the work area root will be set to E:\Work. Notice that the folder “Work” is hilited and its folder icon is open.



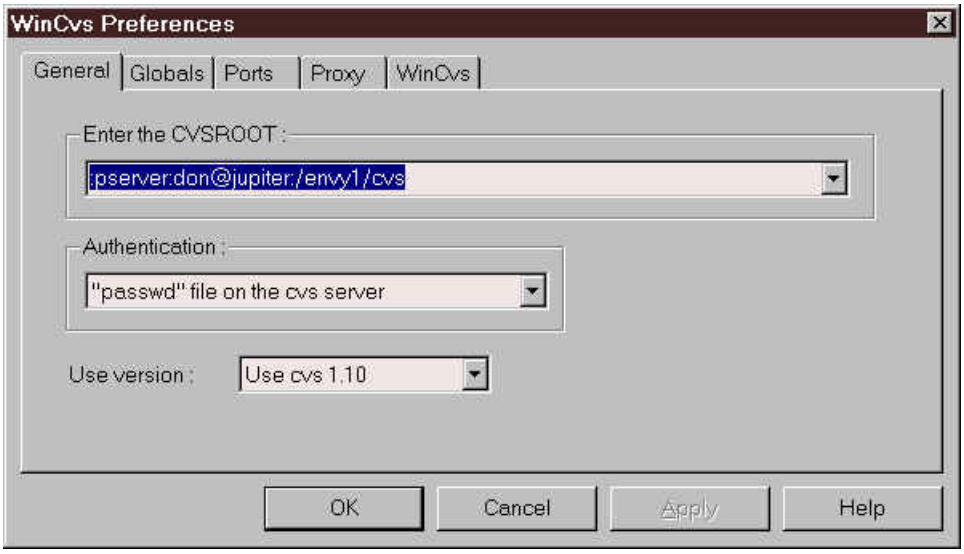
3.2 Setting the WinCvs Preferences

Open the WinCvs Preferences panel by selecting Cvs Admin->Preferences from the menu. As shown, there are five preference panels in WinCvs. The General panel is the most important and most often used.



3.2.1 General Preferences Panel

There are three fields on the General Preferences panel:



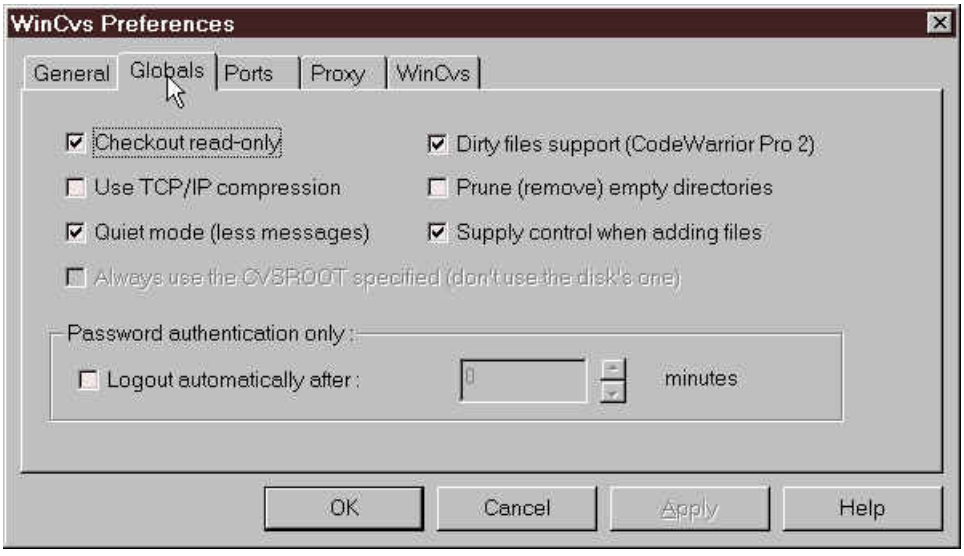
Set the CVSROOT to something similar to the above example: :pserver:don@jupiter:/envy1/cvs

In this example, don is the local domain username that will be used to access the repository /envy1/cvs on the server jupiter. NOTE: there must be an entry in C:\WINNT\system32\drivers\etc\Hosts defining the IP address for the server (jupiter in this case): nnn.nnn.nnn.nnn jupiter

Set the Authentication field to "passwd" file on the cvs server. Set the Use version to Use cvs 1.10.

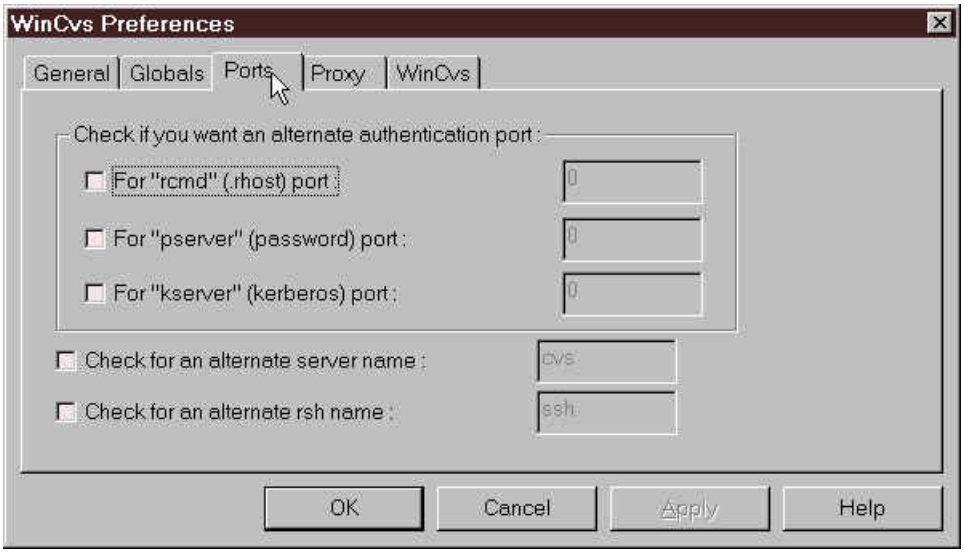
3.2.2 Globals Preferences Panel

Set the Global Preferences as shown below unless you really know what you are doing.



3.2.3 Ports Preferences Panel

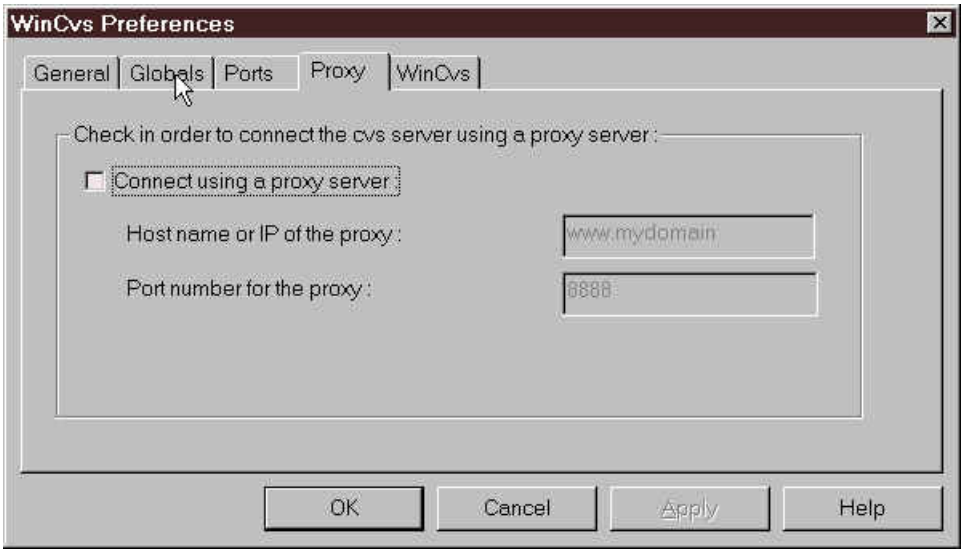
There are five fields on the Ports Preferences panel:



Normally the default values for these fields are used.

3.2.4 Proxy Preferences Panel

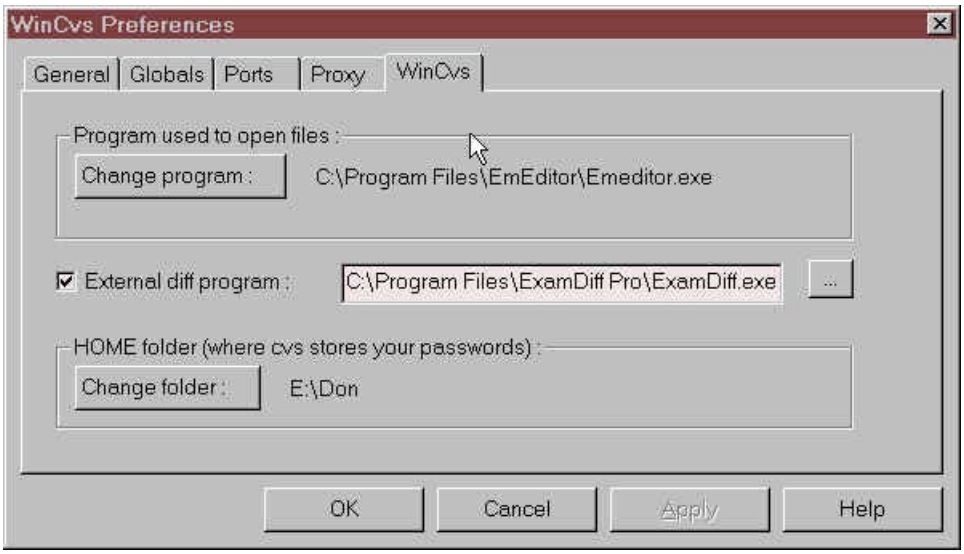
There are two fields on the Proxy Preferences panel:



This panel is required to access cvs through a proxy server.

3.2.5 WinCvs Preferences Panel

There are three settings on the WinCvs Preferences panel:



Set the program used to open files to your favorite editor. WinCvs will run this file when you double-click on a file from the WinCvs browser that has no default Windows filetype association.

If you want to use a graphical diff program, check the box and use the browse button to locate your favorite graphical diff program. Note that for some reason, WinCvs only runs the selected diff program when you run diff from "graph" mode. Graph mode will be explained in a later section.

Set up a HOME folder for CVS to save username and encrypted password information. This can be the CVS installation folder itself (Program Files\GNU\WinCvs 1.1) or your "home" folder if you have one in the Windows NT environment. Specifying this directory separately from the work area allows you add, modify, or delete work areas with only a single login command.

3.3 Logging in to the server

Before any cvs files operations can be performed, the user must log in to cvs. Log in to cvs by selecting Cvs Admin->Login… from the menu. You will be prompted for your password (the password to your software domain usercode as specified in CVSROOT - Section 3.2.1):

Enter your password and click the OK button. If your login was successful, you will see text similar to the following in the status window:

```
cvs -q login
(Logging in to don@nautilus)

*****CVS exited normally with code 0*****
```

If your login fails, you will see text similar to the following in the status window:

```
cvs -q login
(Logging in to don@nautilus)
cvs [login aborted]: authorization failed: server nautilus rejected access

*****CVS exited normally with code 1*****
```

If you try to use other CVS commands prior to logging in, the commans will fail and you will see text similar to the following in the status window:

```
cvs import: could not open E:\Don/.cvspass: No such file or directory
cvs [import aborted]: use "cvs login" to log in first
```

3.4 Checking Out a Module

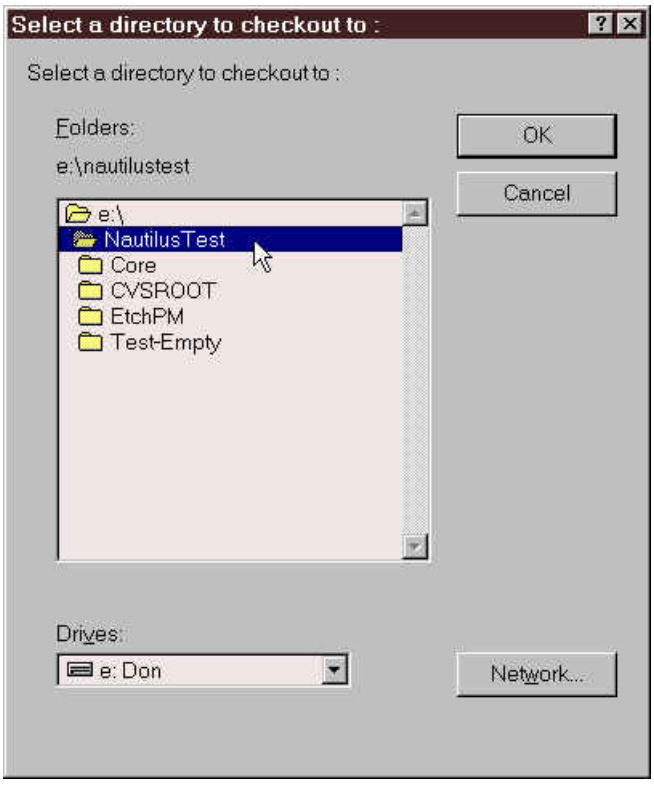
CVS organizes repositories into modules. A module is a hierarchy of folders and files beginning at any folder in the hierarchy of the repository. Every repository has a module called CVSROOT that stores administrative files. It is probably good practice to have a repository administrator maintain these files. Only the administrator should add new modules to the repository.

To checkout a module from the repository, the name of the module must be known. If the name of the module is not known, select Cvs Admin->Macros admin->List the modules on the server from the menu. This command will show the list of modules in the WinCvs status window:

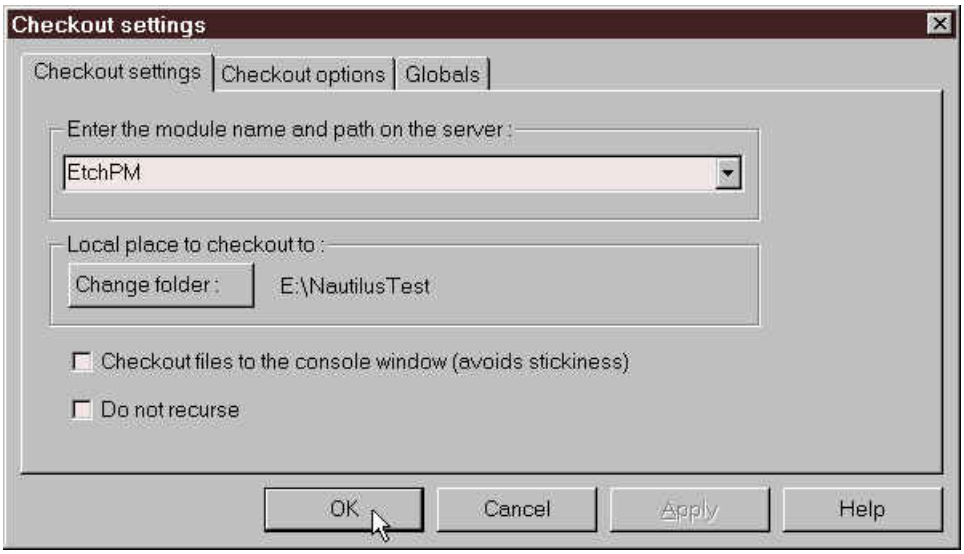
```
*****CVS exited normally with code 0*****

Core          Core
EtchPM        EtchPM
```

The first entry on each line is the module name and the second entry is the name of the root directory of the module relative to the repository root.

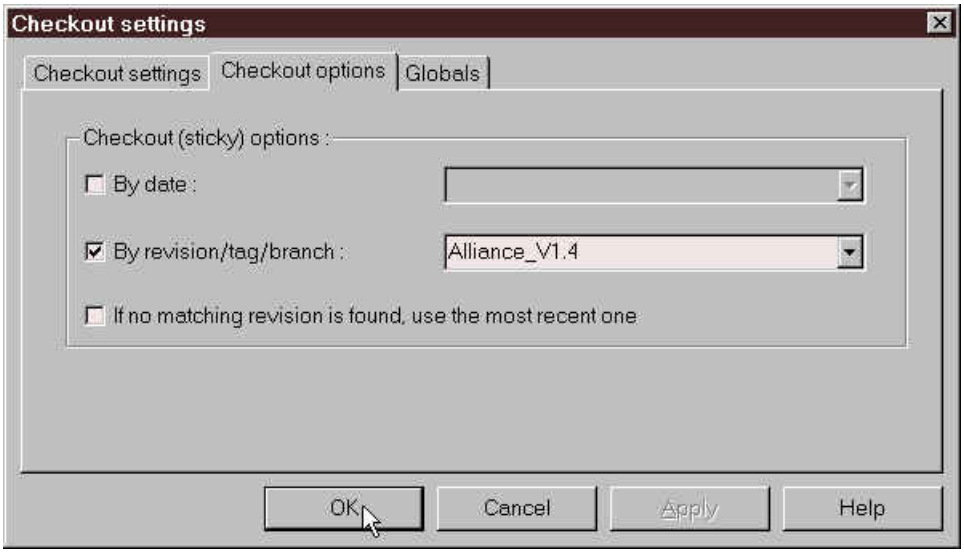
Once the name of the module is known, it can be checked out into the local work area by selecting Cvs Admin->Checkout module… from the menu. A panel will be displayed to allow selection of the destination folder (local work area). Browse to the desired folder and double-click on the name so that the folder name is hilited and the folder icon is open. If you accidentally hilite the folder but do not double-click, the module will be checked out to the parent folder. In the following example, the module will be checked out to the Nautilus Test folder.



Once you have selected your destination folder and clicked OK, the Checkout settings panel will be displayed. The desired module name is entered in the field provided. The destination folder can also be verified and modified if necessary on this panel. Selecting OK at this point will check out the latest version of the module from the root branch (cvs refers to this as the trunk).



If a branched or tagged version of a module is desired, the name of the branch or tag can be specified on the Checkout options panel:



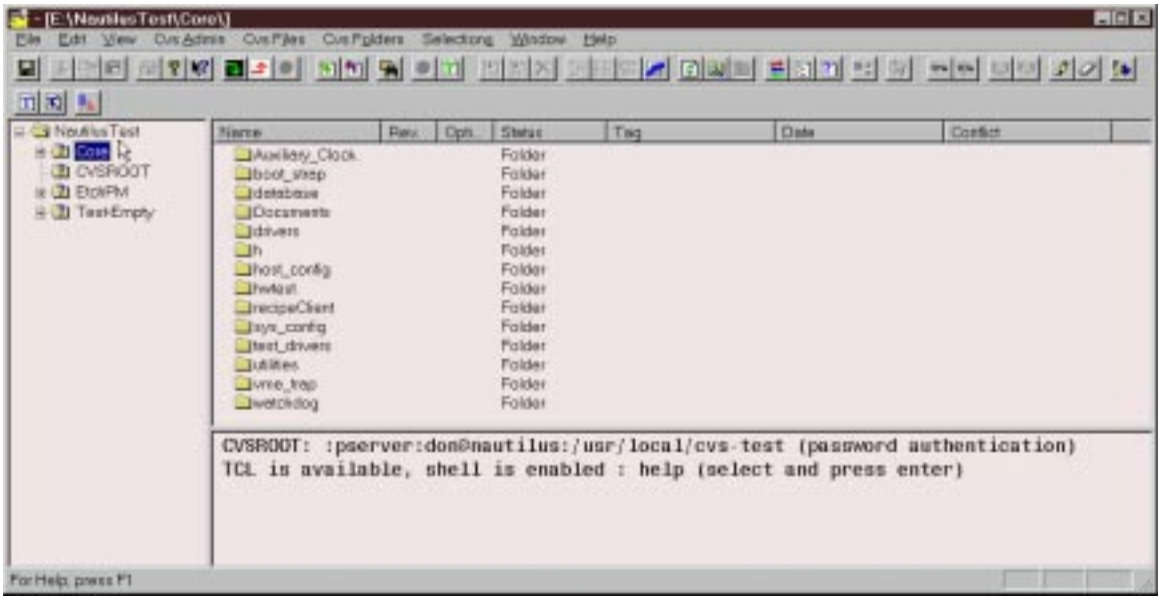
In the above example, the latest version of all files in the EtchPM module for the Alliance_V1.4 branch will be checked out instead of the latest trunk version. Note that it is also possible to further restrict the version selection by date.

Once the module name and any desired options are set, the module will be checked out when the OK button is pressed. The progress and results of the checkout can be seen in the browser status window:

```
cvs -q checkout EtchPM (in directory E:\NautilusTest)
U EtchPM/Makefile
U EtchPM/EtchPM/Makefile
U EtchPM/EtchPM/data/Makefile
U EtchPM/EtchPM/data/A2/EtchPM.script
U EtchPM/EtchPM/data/A2/Makefile
U EtchPM/EtchPM/data/A2/Alarms/EtchPM.almdef
.
.
.
U EtchPM/src/tasks.c
U EtchPM/src/tempctrl.c
U EtchPM/src/wat_cntrl.c

*****CVS exited normally with code 0*****
```

Once the module has been checked out it will not, unfortunately appear in the folder panel on the left side of the browser. Use either the F5 key or the right mouse button (Reload View) to update the display or select the Refresh View icon from the tool bar:

In the following example, 4 modules have been checked out into the Nautilus Test work area and Core is selected in the folder panel on the left. The sub-folders of Core are displayed in the file status panel on the right.



3.5 Updating a Work Area

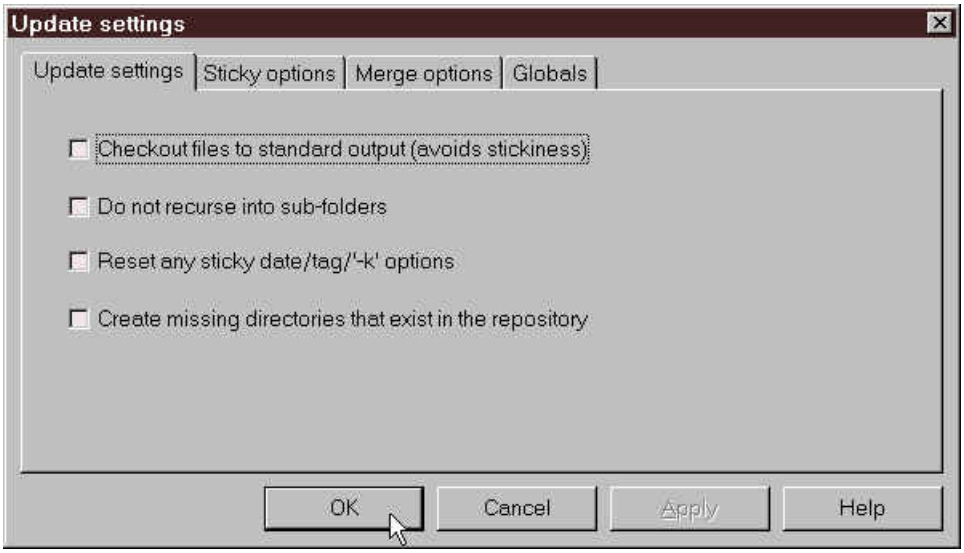
Once a work area has been created, files may become out of date as other developers check in modifications from their own work areas. The CVS update command is provided to allow a work area to be updated to the latest checked in sources. Another common use of the update command is to switch between branches or tagged versions.

The update command (like many commands in WinCvs) can be invoked either on a folder, a file, or on a selection (whatever files or folders are currently hilited). The update command can therefore be found in the Cvs Files, Cvs Folders, and Selections menus. Possibly the most convenient way to update a file or folder is to select it with right mouse button which selects the item and displays the Selections menu all in one mouse click. If you like to use the tool bar, there are also icons on the tool bar for two operations:

Update Folder:

and Update Selection:

No matter how the update command is invoked, the Update settings panel will be displayed:

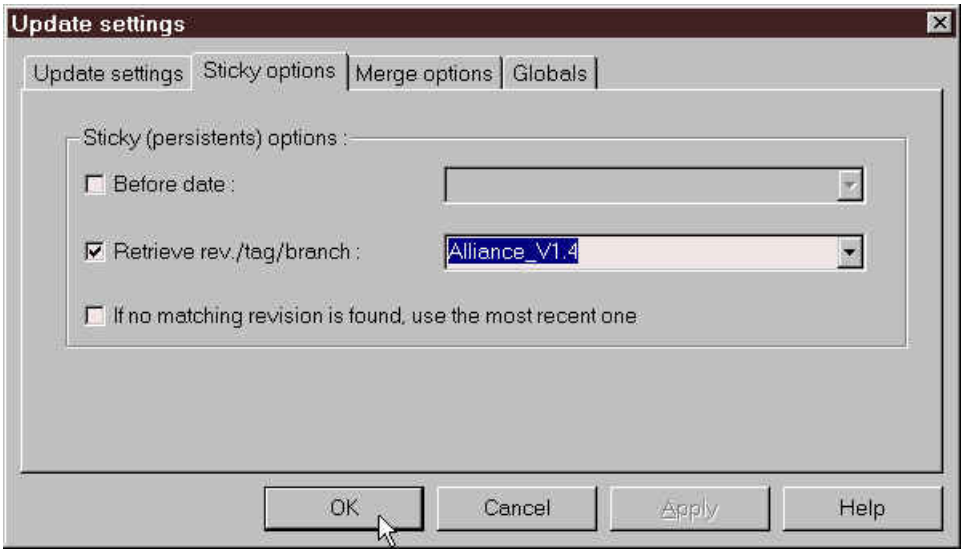


The Do not recurse into sub-folders option may be useful to update only a selected folder instead of all subfolders (recursion is always the default for cvs).

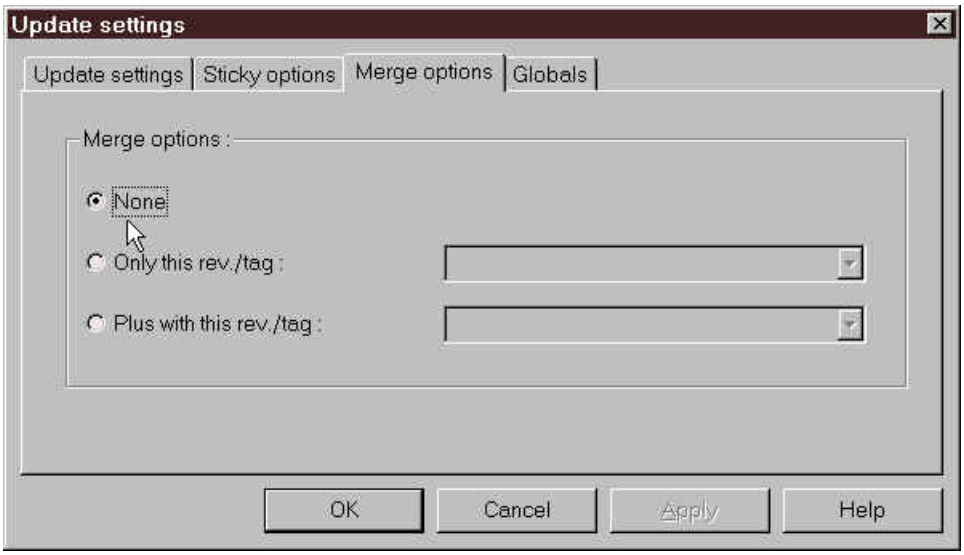
The Reset any sticky date/tag/-k options option is used to update back to the latest version of all files in the main trunk. During development, you may update to a branch, tag, or by date and later wish to get a clean work area updated to the trunk. This option will do that for you.

The Create missing directories that exist in the repository is useful to pick up new directories that have been added to the repository since the module was originally checked out.

To update to a branch other than the trunk, update to a tagged version, or update by date, the Sticky options panel must be used. In the following example, an existing work area would be updated to the latest version of all files in the Alliance_V1.4 branch:



There are some sophisticated options related to merging available on the Merge options panel. Consult the cvs reference manual for more information on this panel:



3.6 Editing a File

By default, files are checked out as non-editable when checking out a module or updating a work area. To modify a file, the edit command must first be used to make the file writable. In WinCvs, an icon to the left of the filename in the browser file window indicates the read/write status of a file.

Read-only files are indicated with:

Writable files are indicated with:

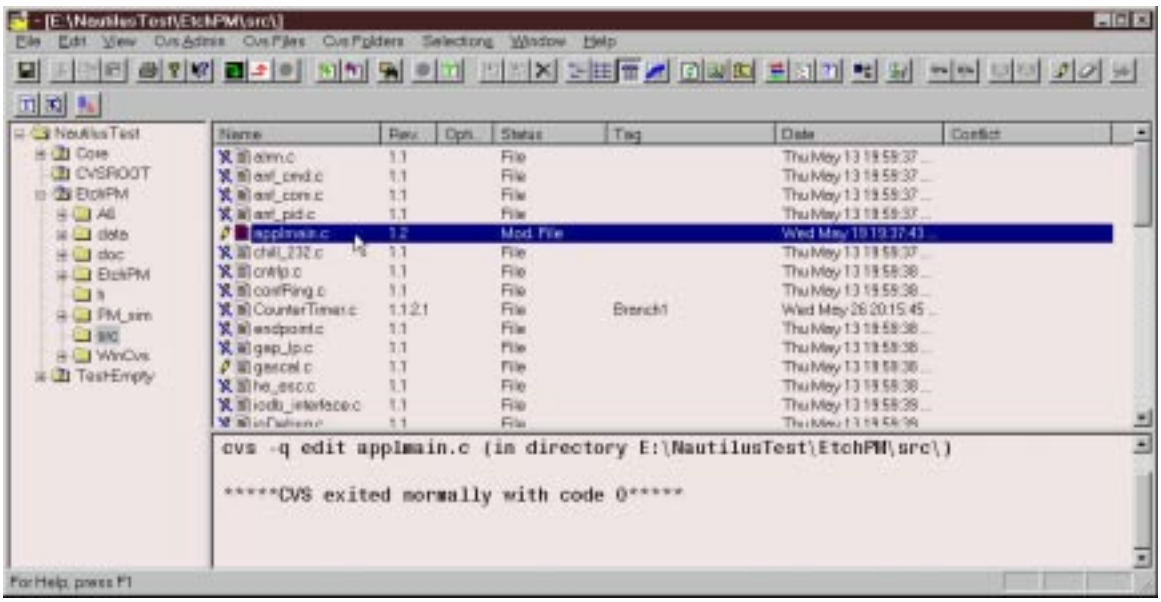
To invoke the edit command and make a file writable, select the file and then press the edit icon on the right side of the toolbar:

The edit command is also available from the right mouse button Monitors selection->Edit selection menu or from the Selections->Monitors selection->Edit selection menu but the tool bar is more convenient.

After the file has been made writable, it can be edited simply by double-clicking on the file in the WinCvs browser window.

To undo the edit action (make the file read-only), the unedit command availble from the menu and tool bar.

After modifying a file, the files ICON in the WinCvs file window will turn red and the Status will change to Mod. File as shown in the following example where applmain.c was modified:



3.7 Checking Diffs Prior to Commit (Text Diff)

It is often useful to run diff as a precaution prior to committing a file to the repository. This section describes the steps required to compare the version of a file in your work area to the current version in the repository. The differences can either be displayed as a standard text diff in the WinCvs status window or graphically (if you have set up an external diff program as described in Section 3.2.5). This section deals with the standard text diff. See the following section (Section 3.8) for an example of a graphical diff.

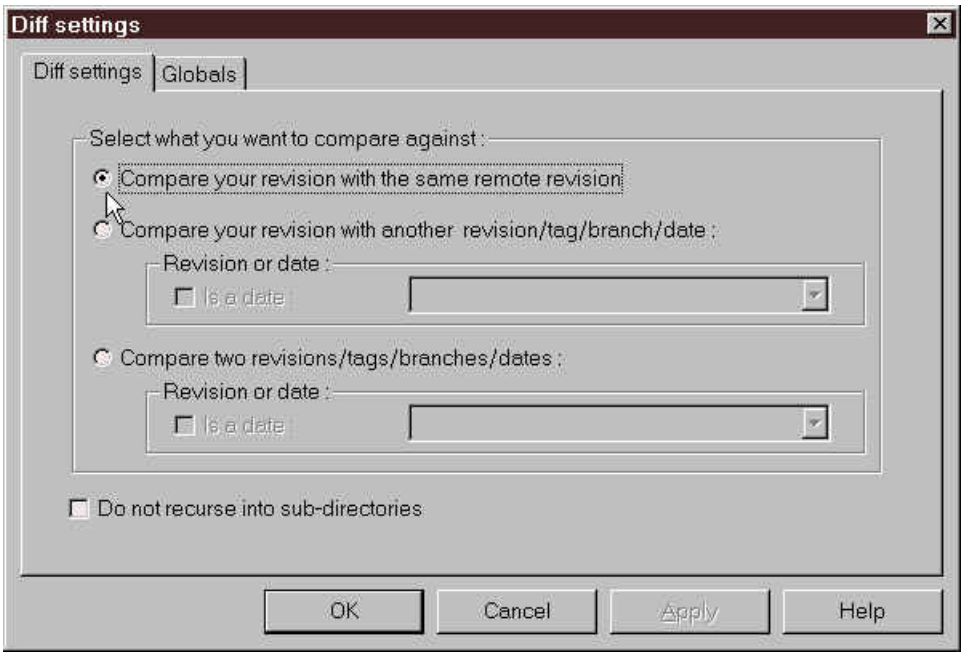
The diff can be run in one of three different ways:

1) Select the file using the left mouse button and use the menu: Selections->Diff selection

2) Select the file using the right mouse button and use Diff selection

3) Select the file using the left mouse button and use the tool bar:

Once diff is invoked, the Diff settings panel will be displayed:



The Diff settings panel has several options but only the first option makes sense for this example (since we just want to compare our revision to the repository). Make sure the first option is selection as shown above and then click the OK button.

The text version of the file differences (if any) will now be displayed in the WinCvs status window as shown here:

```
cvs -q diff applmain.c (in directory E:\NautilusTest\EtchPM\src)
Index: applmain.c
===================================================================
RCS file: /usr/local/cvs-test/EtchPM/src/applmain.c,v
retrieving revision 1.1
diff -r1.1 applmain.c
988,993c988,994
< int             Metal960OPTX_installed = FALSE;
< int             single_plane_transfer = FALSE;
< int             chill_232_present = FALSE;
< int             pump_232_present = FALSE;
< int             lonwork_present = FALSE;
< int             scrubber_232_present = FALSE;
---
> int             Metal960OPTX_installed = TRUE;
> int             single_plane_transfer = TRUE;
> int             chill_232_present = TRUE;
> int             pump_232_present = TRUE;
> int             lonwork_present = TRUE;
> int             scrubber_232_present = TRUE;
> int             advanced_endpoint_installed = TRUE;

*****CVS exited normally with code 1*****
```

The text in the WinCvs status window can be selected, saved to a file, or printed by making the status window active (click anywhere in the log window) and using the Edit and File menus. The log can also be cleared using Edit->Select All and Edit->Cut.

3.8 Checking Diffs Prior to Commit (Graphical Diff)

As mentioned in Section 3.7, it is often useful to run diff as a precaution prior to committing a file to the repository. This section describes the steps required to compare the version of a file in your work area to the current version in the repository. The differences can either be displayed as a standard text diff in the WinCvs status window or graphically (if you have set up an external diff program as described in Section 3.2.5). This section deals with the graphical diff. See the previous section (Section 3.7) for an example of a text diff.

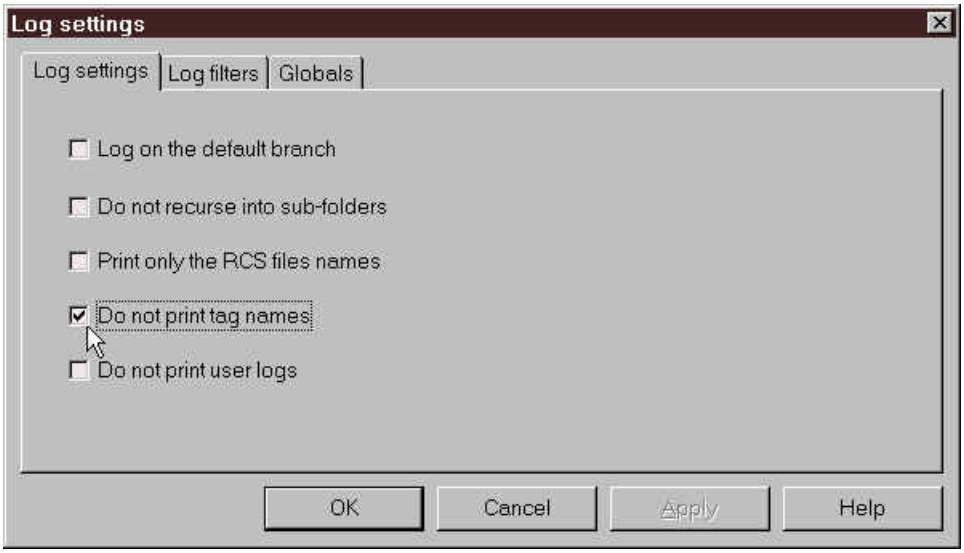
In WinCvs, the graphical diff program can only be used in graph mode. Graph mode shows all revisions of a file in graphical format. The graph can be generate in one of three different ways:

1) Select the file using the left mouse button and use the menu: Selections->Graph selection

2) Select the file using the right mouse button and use Graph selection

3) Select the file using the left mouse button and use the tool bar:

Once the graph is invoked, the Log settings panel will be displayed:



There are many options in the Log setting panels which the user may explore as necessary. In this example, the Do not print tag names option has been selected since this may be a commonly used option to prevent too much information on the graph. Tag names in cvs include branch names (which are useful to see) but also include “tagged versions” which are probably not interesting to developers (these are equivalent to frozen maps most likely used by release managers).

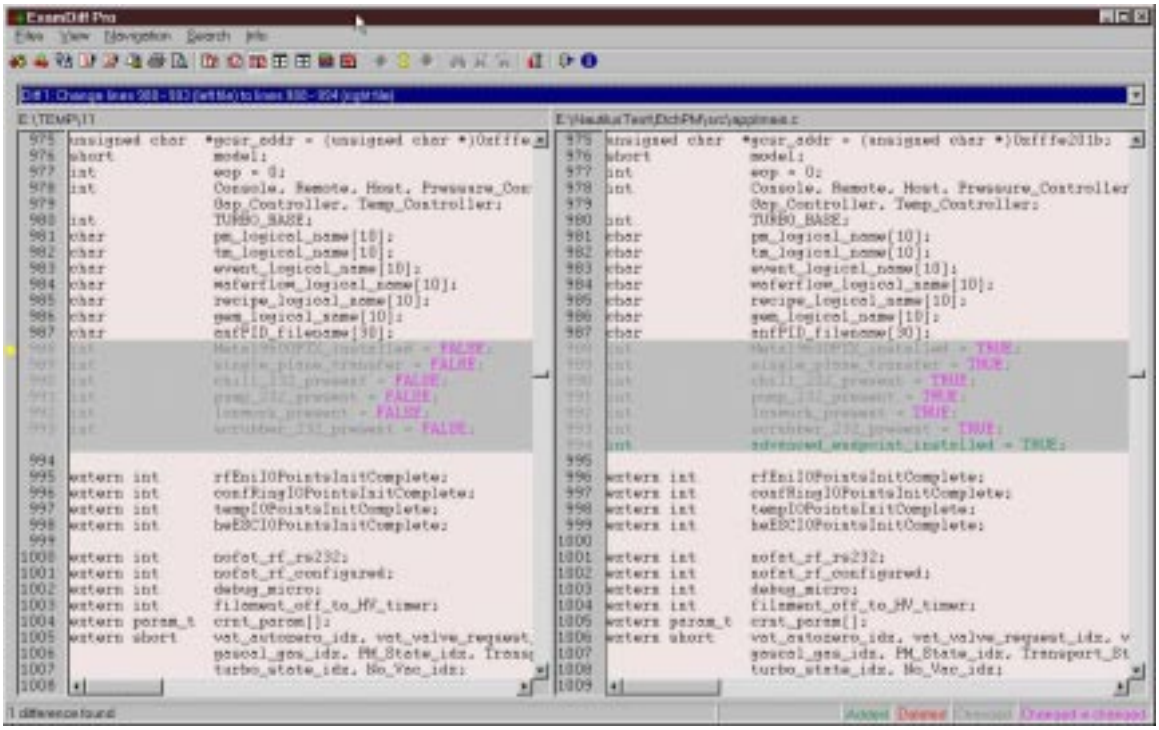
Once all the desired options are selected, click the OK button to display the graph.

The graph in this example for applmain.c shows that there are two versions currently in the repository:

To display the diffs between the work area version of the file and the repository version (1.2), first click on the most recent repository version as shown above. Notice that the version log is displayed in the lower half of the graph window. The version log shows useful information such as the author, modification date/time, and log message for the selected version of the file.

The graphical diff can now be displayed either from the graph menu using Graph->Diff selected or from the graph tool bar:

The external diff program (see section 3.2.5 for setup information) will now be run as shown in the following example:



Notice that the diff program shown in the above example (ExamDiff Pro) has many options including the ability to customize colors, ignore white space, etc (from the View->Options… menu). There is also an option to save a UNIX style text diff (from the File->Save Unix Diff File… menu).

Be careful when closing the graph window after viewing the diffs or you may end up closing WinCvs entirely. A more convenient way to run WinCvs may be in Tile mode which can be selected from the Window->Tile menu in WinCvs.

3.9 Committing a File or Folder

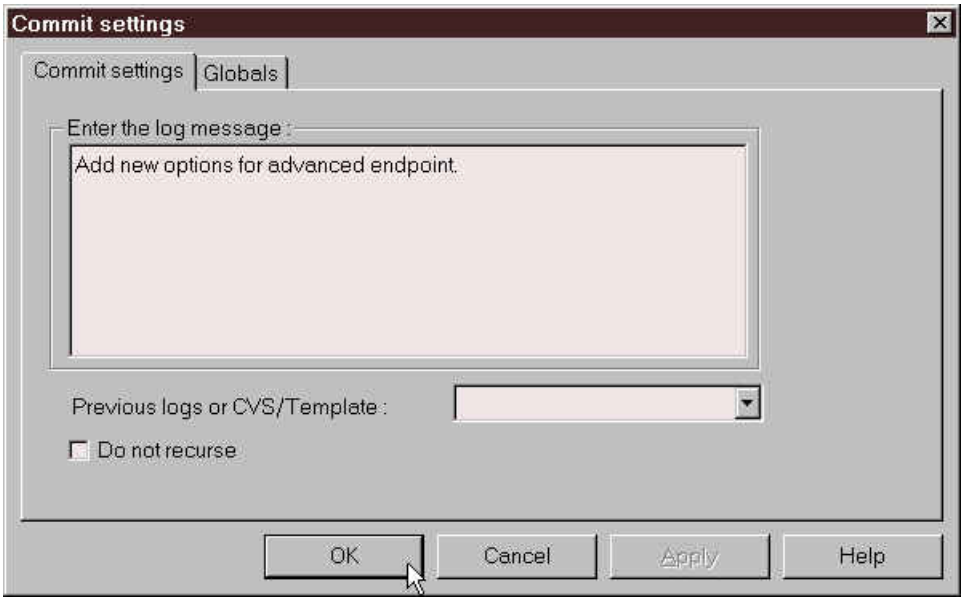
After one or more files have been edited, individual files or an entire tree of folders can be committed to the repository using the commit command. The commit command can be invoked in several ways:

1) Select a file or folder using the left mouse button and use the menu: Selections->Commit selection

2) Select a file or folder using the right mouse button and use Commit selection

3) Select a file or folder using the left mouse button and use the tool bar:

Once commit is invoked, the Commit settings panel will be displayed:



Enter an appropriate log message and hit the OK button to commit your changes. If you have selected a folder to commit and do not wish any sub-folders to be committed, check the Do not recurse option. The results of the commit operation will be displayed in the WinCvs status window as shown here:

```
cvs -q commit -m "Add new options for advanced endpoint." applmain.c (in
directory E:\NautilusTest\EtchPM\src)
Checking in applmain.c;
/usr/local/cvs-test/EtchPM/src/applmain.c,v  <--  applmain.c
new revision: 1.2; previous revision: 1.1
done

*****CVS exited normally with code 0*****
```

3.10 Adding Files or Folders To the Repository

Adding files or folders to a cvs repository can be done using either the cvs add or import command. Developers should normally use the add command to add files or a small number of folders. There are only two situations where the import command should be used:

- If the files to be created require the creation of a new module (because, for example, no appropriate module exists), the import command must be used. To maintain a consistent naming convention for modules, it is a good idea to leave module creation to a cvs administrator. The number of modules in a repository should also be controlled to avoid confusion.

- If a multi-level hierarchy is being added to an existing module, it can be tedious to add each level manually. WinCvs allows addition of all files and sub-folders of a single folder to be added in one operation. The same operation would, however, have to be repeated manually within each of the sub-folders down to the leaves of the tree. In addition, binary files need to be added in a separate operation from text files making the task even more tedious. In this situation, it would be much simpler to use the import command to import the entire hierarchy to a sub-folder of an existing module. It still might be a good idea to get help from a cvs administrator when attempting file imports in this way.

If the add command sounds more appropriate (the normal case), read Section 3.10.1 for documentation on using the add command. If the update command is required, read Section 3.10.2 or contact your cvs administrator.

3.10.1 Adding Files or Folders Using Add (to an existing module)

Before files or folders can be added to a module, a work area must exist with the module already checked out. Files and folders can then be created within the existing work area using any appropriate method such as:

- create an empty folder from the Windows NT Explorer
- copy files from another location
- copy an existing hierarchy from another location (consider using import, Section 3.10.2)
- create files using a text editor, Microsoft Word, or other application

There are three important facts that need to be mentioned regarding the add command:

1) The add command is NEVER recursive in cvs. This means that adding an existing hierarchy to an existing module is a manual process. You must manually add the files and folders at each level of the tree. If you are adding more than one level to the existing repository, consider using the import command (Section 3.10.2).

2) Cvs needs to be told which files to treat as binary and which files to treat as text files. WinCvs will usually warn you if you try to add a binary file as a text file but it is safer to add text files with the Add versions of WinCvs commands, and binary files with the Add Binary versions.

3) The add command just adds the file to the local work area. The commit command (Section 3.9) must be used to permanently add the files to the repository.

Once all the files and folders are in place in the work area, you are ready to add them to the repository. As with other commands, WinCvs provides several ways to invoke the add command:

1) Select the file(s) or folder(s) using the left mouse button and use the appropriate menu entry:

   Selections->Add selection for text files or folders
   Selections->Add selection binary for binary files

2) Select the file(s) or folder using the left mouse button and use the right mouse button to open the Selections menu. Choose the appropriate menu entry:

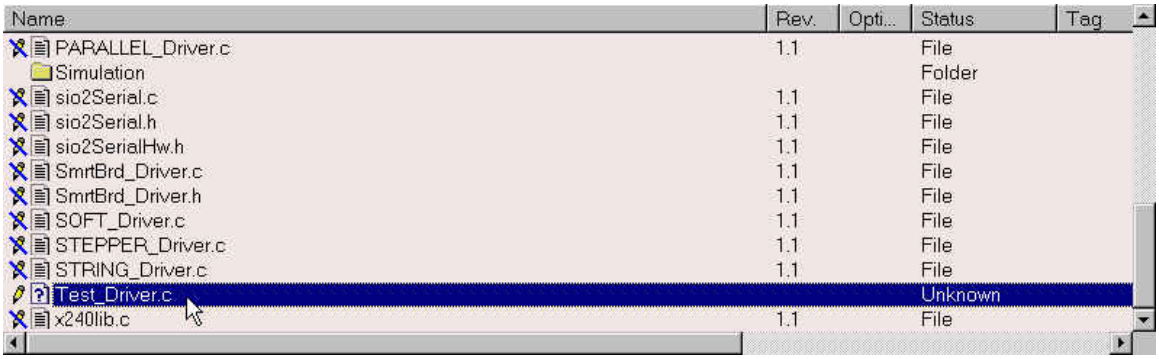
   Add selection for text files or folders
   Add selection binary for binary files

3) Select the file(s) or folder using the left mouse button and use the appropriate tool bar icon:

   Add selected (for folders or text files):

   Add selected binary (for binary files):

Note that the selected file or folder should be displayed with a "?" icon and a status of Unknown as shown:

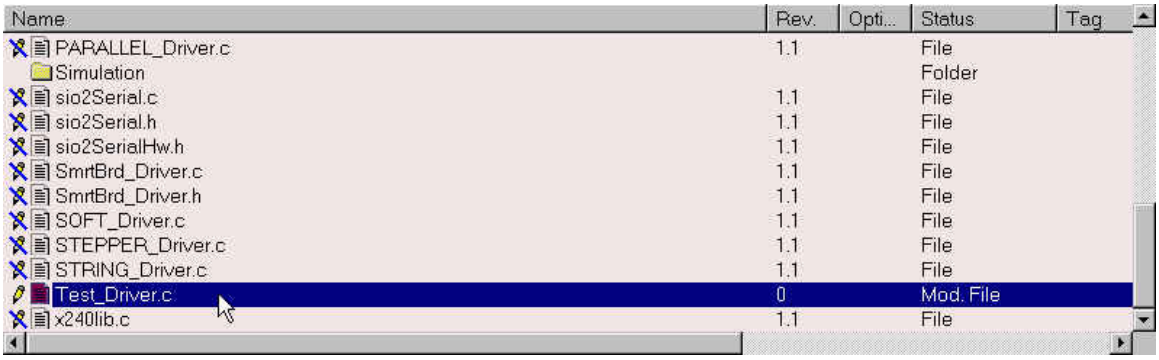


Once the add (or add binary) is invoked, the file or folder will be added to the work area. The status of the add operation can be seen in the WinCvs status window as shown here:

```
cvs -q add Test_Driver.c (in directory E:\NautilusTest\Core\drivers\)
root server: use 'root commit' to add this file permanently

*****CVS exited normally with code 0*****
```

As indicated, the file or folder still requires a commit operation to actually add the file to the repository.

After a file is added to the work area, the status should change to Mod. File and the revision should be shown as zero (0) as shown here:



The added file(s) may now be edited further or committed to the repository (Section 3.9).

3.10.2 Adding Files or Folders Using Import

This cvs import command provides a way to add an existing hierarchy of files and folders to the repository by either creating a new module or adding to an existing module. As discussed in Section 3.10, there are several reasons this command should be used primarily by a cvs administrator.

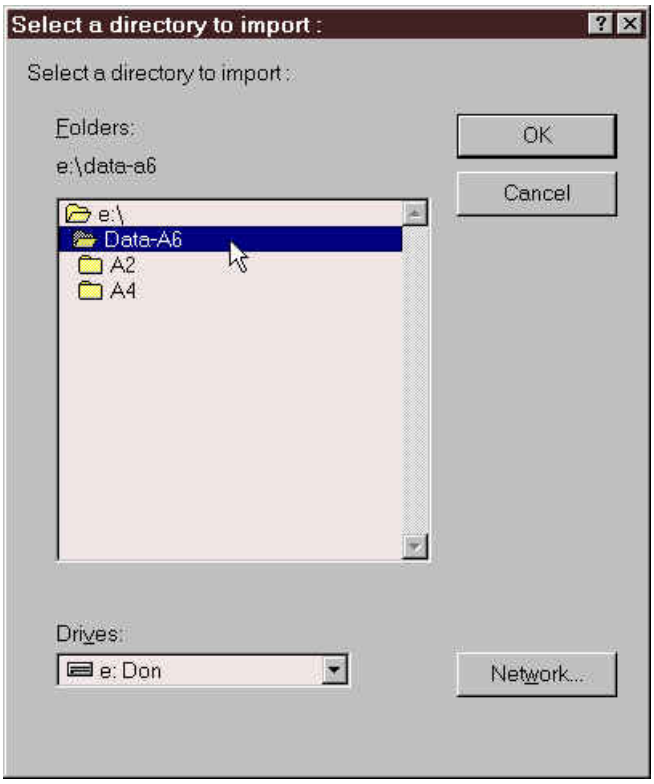
Unlike the add command, the hierarchy to be imported should NOT be in the local work area prior to running the import command. The entire hierarchy can later be checked out to the local work area after the import is complete.

There are two examples in this section. In the first example (Section 3.10.2.1), an existing hierarchy of text and binary files will be imported to an existing module. In the second example (Section 3.10.2.2), an existing hierarchy of text and binary files will be imported as a new module.

3.10.2.1 Import File Hierarchy To Existing Module

Unlike other commands, WinCvs provides only one way to invoke the import command:

Select Cvs Admin->Import module… from the WinCvs menu. The Select panel will be displayed:



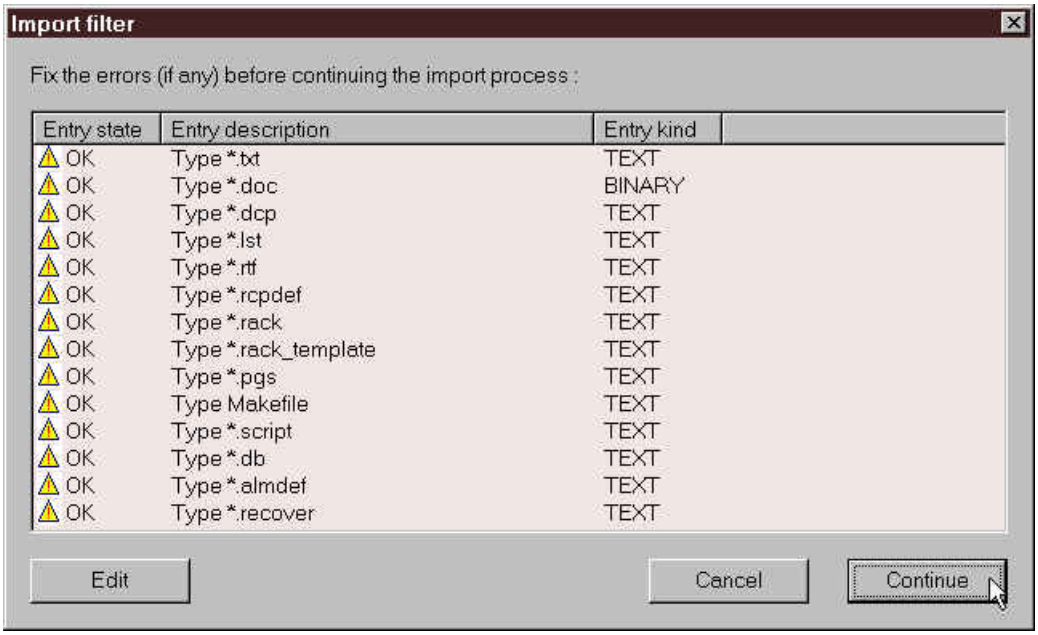
Locate the folder you want to import and double-click on the folder so the folder icon is open. If you just hilite the folder with a single mouse click, the parent folder (and all of its subfolders) will be selected instead of just the desired folder. This is a bad feature (bug) all WinCvs commands that use this type of folder selection. In the above example, the Data-A6 folder is selected and its folder icon is open.

After double-clicking on the folder to import, select OK. At this point, WinCvs will examine the hierarchy to determine how many files are being imported and determine the most likely type for each file (text or binary). This process is called filtering and may take some time on a large hierarchy. During this time, the WinCvs status window will display a line for each folder examined as shown here:

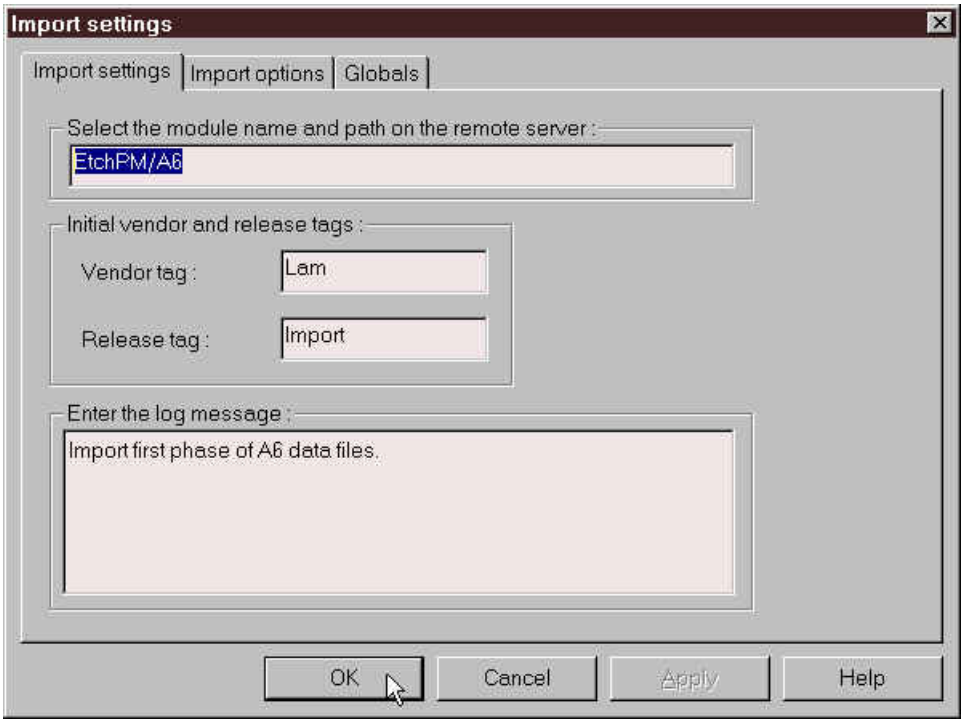
```
Filtering 'E:\Data-A6'...
Filtering 'E:\Data-A6\A2'...
Filtering 'E:\Data-A6\A2\Alarms'...
Filtering 'E:\Data-A6\A2\db'...
Filtering 'E:\Data-A6\A2\pg'...
Filtering 'E:\Data-A6\A2\Rack'...
Filtering 'E:\Data-A6\A2\Recipe'...
Filtering 'E:\Data-A6\A4'...
Filtering 'E:\Data-A6\A4\Alarms'...
Filtering 'E:\Data-A6\A4\db'...
Filtering 'E:\Data-A6\A4\dcp'...
Filtering 'E:\Data-A6\A4\pg'...
Filtering 'E:\Data-A6\A4\Rack'...
Filtering 'E:\Data-A6\A4\Recipe'...
```

When the filtering process is complete, the Import filter results panel will be displayed. This panel lists the file extensions found in the import hierarchy and the type cvs will use to add the file (TEXT or BINARY). In the following example there were no warnings or errors. In some cases, however, WinCvs may find a file with an extension that normally indicates a text file but the file appears to be a binary file. The errors can either be ignored (WinCvs usually does the right thing anyway) or the Edit button may be used to override the choices made by WinCvs.

After the Import filter panel has been reviewed, select Continue to begin the import process:



After the Continue button is pressed, the Import settings panel will be displayed:



In this example, we are creating a new folder A6 under the existing EtchPM module. Note that the imported folder Data-A6 will not appear in the repository. All the files and subfolders of Data-A6 will be imported to the repository under the A6 folder. This is example was done to show that a folder can be imported with a different name. Four fields need to be entered prior to initiating the import:

- Enter the module name name and path as shown above. Notice that cvs REQUIRES forward slashes "/" to separate directory names. If you specify a backslash, you will actually create a new module called EtchPM\A6 in the repository root instead of a new directory A6 under the exsiting EtchPM directory.

- Enter something for the vendor and release tags. These tags are often useless and can be deleted later. In that case, use tags like VTAG, and RTAG, and delete them after the import is complete. CVS requires them, unfortunately due to the history of the import command. They can be deleted after the import by selecting the top level folder and using Delete tag which is available under the Selections->Tag selection menu.

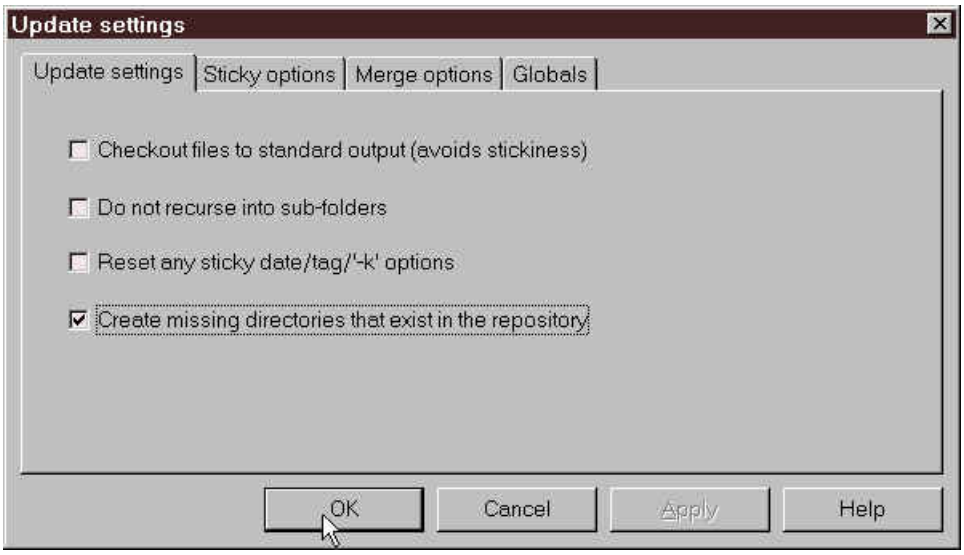
- Enter an appropriate log message (this message will be part of the files log information for the initial import version 1.1) which will appear in the file if the $Log keyword is specified in the file (text files only).

After all the fields are entered, hit the OK button to initiate the import.

During the import, the actual cvs command executed and any output will be displayed in the WinCvs status window as shown in this example:

```
cvs -q import -I ! -I CVS -W "*.doc -k 'b'" -m "Import first phase of A6
data files." EtchPM/A6 AVendor Import (in directory E:\Data-A6)
N EtchPM/A6/Endpoint.doc
N EtchPM/A6/EtchPM.doc
N EtchPM/A6/Makefile
N EtchPM/A6/README.txt
N EtchPM/A6/A2/EtchPM.script
N EtchPM/A6/A2/Alarms/EtchPM.almdef
.
.
.
N EtchPM/A6/A4/Rack/SILYLATION.rack
N EtchPM/A6/A4/Recipe/EtchPM.rcpdef
N EtchPM/A6/A4/Recipe/SilyPM.rcpdef

No conflicts created by this import


*****CVS exited normally with code 0*****
```

Now that the files are imported to the repository, they may be checked out to the work area by using the update command on the parent folder. In this example, select the EtchPM directory from your WinCvs browser window and select update with the right mouse button. The Update settings panel is displayed:



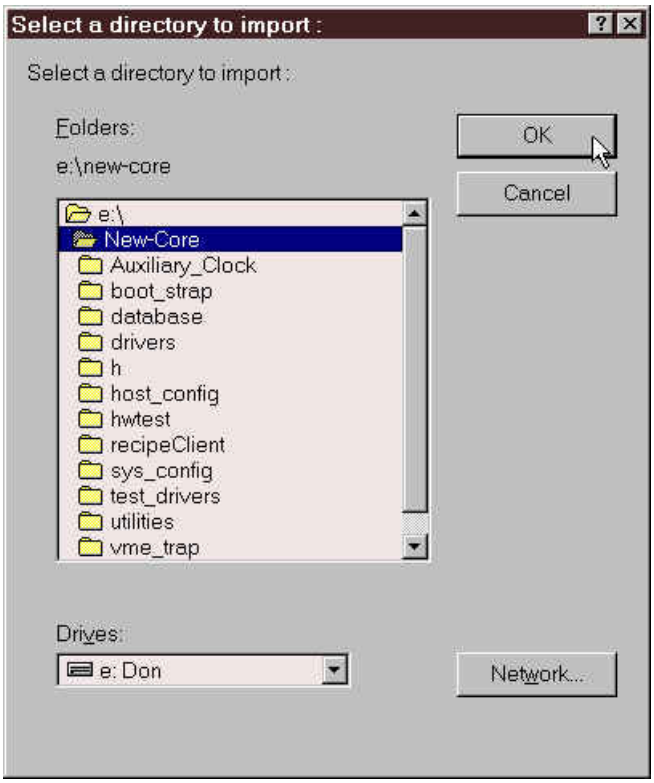
Check the Create missing directories that exist in the repository option and hit the OK button to retrieve a working copy of the imported files to the local work area.

If you want your newly imported files to be accessible from the repository as a separate module, see Section 4.1 (Updating the modules file). This can be useful to allow a user to check out part of a module tree instead of requiring checkout of the entire tree.

3.10.2.2 Import File Hierarchy To New Module

Unlike other commands, WinCvs provides only one way to invoke the import command:

Select Cvs Admin->Import module… from the WinCvs menu. The Select panel will be displayed:



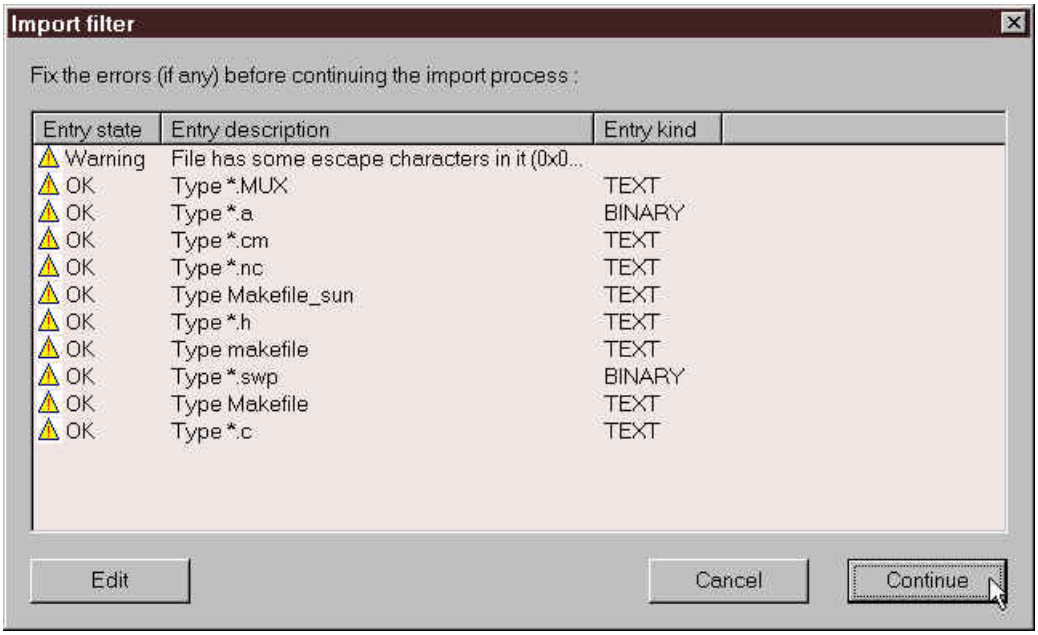
Locate the folder you want to import and double-click on the folder so the folder icon is open. If you just hilite the folder with a single mouse click, the parent folder (and all of its subfolders) will be selected instead of just the desired folder. This is a bad feature (bug) all WinCvs commands that use this type of folder selection. In the above example, the New-Core folder is selected and its folder icon is open.

After double-clicking on the folder to import, select OK. At this point, WinCvs will examine the hierarchy to determine how many files are being imported and determine the most likely type for each file (text or binary). This process is called filtering and may take some time on a large hierarchy. During this time, the WinCvs status window will display a line for each folder examined as shown here:

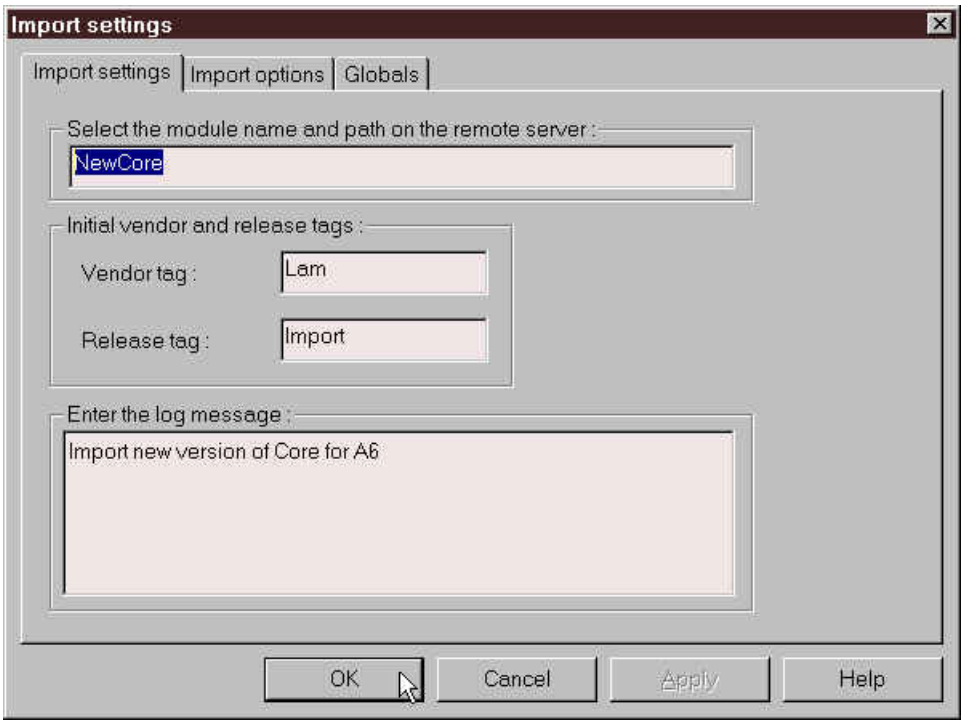
```
Filtering 'E:\New-Core'...
Filtering 'E:\New-Core\boot_strap'...
Filtering 'E:\New-Core\database'...
Filtering 'E:\New-Core\drivers'...
.
.
.
Filtering 'E:\New-Core\h'...
Filtering 'E:\New-Core\utilities'...
Filtering 'E:\New-Core\vme_trap'...
```

When the filtering process is complete, the Import filter results panel will be displayed. This panel lists the file extensions found in the import hierarchy and the type cvs will use to add the file (TEXT or BINARY). In the following example there was 1 warning and no errors. The warning indicates that a file with an extension that normally indicates a text file contains some non-text characters. The errors can either be ignored (WinCvs usually does the right thing anyway) or the Edit button may be used to override the choices made by WinCvs.

After the Import filter panel has been reviewed, select Continue to begin the import process:



After the Continue button is pressed, the Import settings panel will be displayed:



In this example, we are creating a new folder A6 under the existing EtchPM module. Note that the imported folder Data-A6 will not appear in the repository. All the files and subfolders of Data-A6 will be imported to the repository under the A6 folder. This is example was done to show that a folder can be imported with a different name. Four fields need to be entered prior to initiating the import:

- Enter the module name name and path as shown above. Notice that cvs REQUIRES forward slashes "/" to separate directory names. If you specify a backslash, you will actually create a new module called EtchPM\A6 in the repository root instead of a new directory A6 under the exsiting EtchPM directory.

- Enter something for the vendor and release tags. These tags are often useless and can be deleted later. In that case, use tags like VTAG, and RTAG, and delete them after the import is complete. CVS requires them, unfortunately due to the history of the import command. They can be deleted after the import by selecting the top level folder and using Delete tag which is available under the Selections->Tag selection menu.

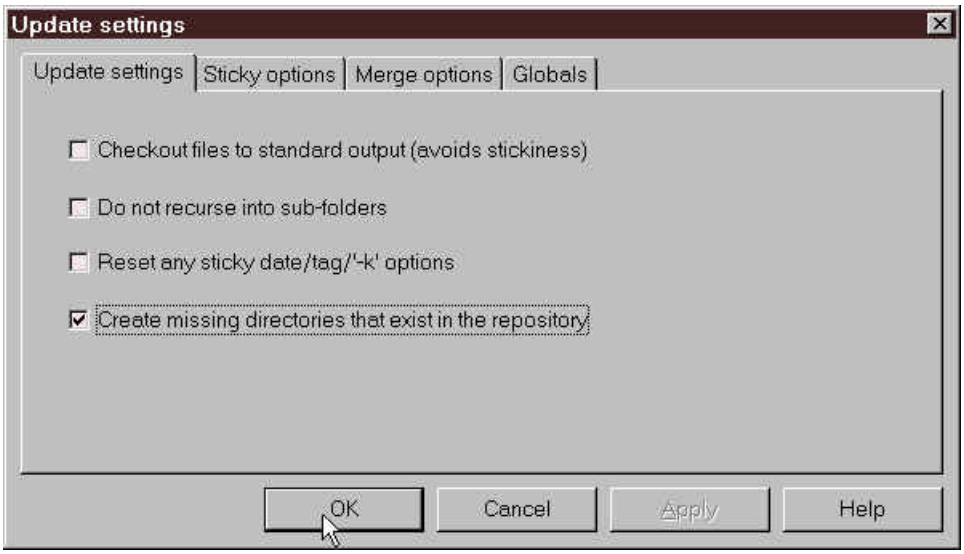
- Enter an appropriate log message (this message will be part of the files log information for the initial import version 1.1) which will appear in the file if the $Log keyword is specified in the file (text files only).

After all the fields are entered, hit the OK button to initiate the import.

During the import, the actual cvs command executed and any output will be displayed in the WinCvs status window as shown in this example:

```
cvs -q import -I ! -I CVS -W "*.a -k 'b'" -W "*.swp -k 'b'" -m "Import new
version of Core for A6" NewCore AVendor Import (in directory E:\New-Core)
N NewCore/Auxiliary_Clock/auxClock.c
N NewCore/Auxiliary_Clock/Makefile
N NewCore/boot_strap/.readConfig.c.swp
N NewCore/boot_strap/makefile
.
.
.
N NewCore/watchdog/Makefile
N NewCore/watchdog/watchdog.c

No conflicts created by this import


*****CVS exited normally with code 0*****
```

Now that the files are imported to the repository, they may be checked out to the work area by using the update command on the parent folder. In this example, select the EtchPM directory from your WinCvs browser window and select update with the right mouse button. The Update settings panel is displayed:



Check the Create missing directories that exist in the repository option and hit the OK button to retrieve a working copy of the imported files to the local work area.

In this example of the import command, a new module was created. The module will not appear in the list of available modules, however, until the module is added to an administrative file called modules. See Section 4.1 for help on adding modules to the modules file.

3.11 Multiple Developer Coordination

The default model in cvs for managing multiple developers is known as *unreserved checkouts*. In this model, multiple developers can edit a *working* copy of a file simultaneously. The first developer to com a merge will be required if another developer commits changes to a file while you are editing the same file. The merge feature is handled automatically by cvs and only requires manual merging when cvs finds conflicting modifications that it cannot resolve.

Cvs also provides limited support for a *reserved checkout* model. The support is limited in that cvs does not require a lock before allowing a file to be edited although it does not allow commit to a file that is locked by another developer.

3.11.1 Understanding Merging and the Unreserved Checkout Model

When developers use the *unreserved checkout* model in cvs, any number of developers may be working on the same file simultaneously. As explained in Section 3.6, the edit command should be used to make a file writable before making any modifications to the file. At any time, the list of developers currently editing a file can be listed by selecting the file with the right mouse button and using the Monitors selection->Editors of selection command from the menu.

When a developer tries to commit a file, one of two things will happen:

- If no newer version of the file is found in the repository, a new version will be created and the file will be committed to the repository directly as described in Section 3.9. This is the normal case where either no other developers are working on the same file or you are the first one to commit your changes.

- If a newer version of the file is found in the repository, cvs will issue a warning and abort the commit operation.

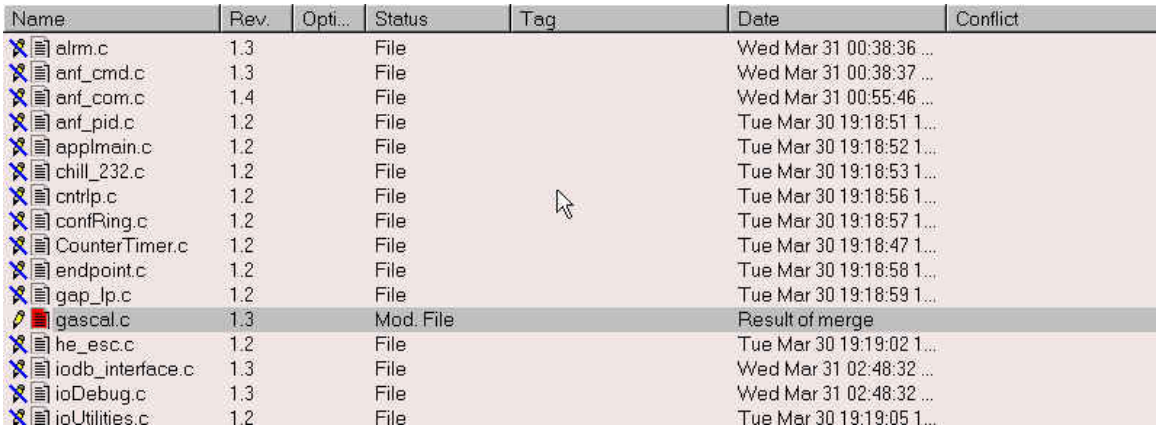
In the case where cvs aborts the commit operation due to a newer version found in the repository, messages like the following will be displayed in the WinCvs status window:

```
cvs -q commit -m "This is the change from user 1" gascal.c (in directory
E:\WorkAreaOne\EtchPM\src)
cvs server: Up-to-date check failed for `gascal.c'
cvs [server aborted]: correct above errors first!

*****CVS exited normally with code 1*****
```

In this case, the update command must first be used to merge in the missing changes. If the merge succeeds, messages similar to the following will be displayed:

```
cvs -q update gascal.c (in directory E:\WorkAreaOne\EtchPM\src)
RCS file: /Store/200mm/EtchPM/src/gascal.c,v
retrieving revision 1.2
retrieving revision 1.3
Merging differences between 1.2 and 1.3 into gascal.c
M gascal.c


*****CVS exited normally with code 0*****
```

In this case, the status of the file will remain Mod. File but the modification date will change to the text: Result of Merge as shown here:



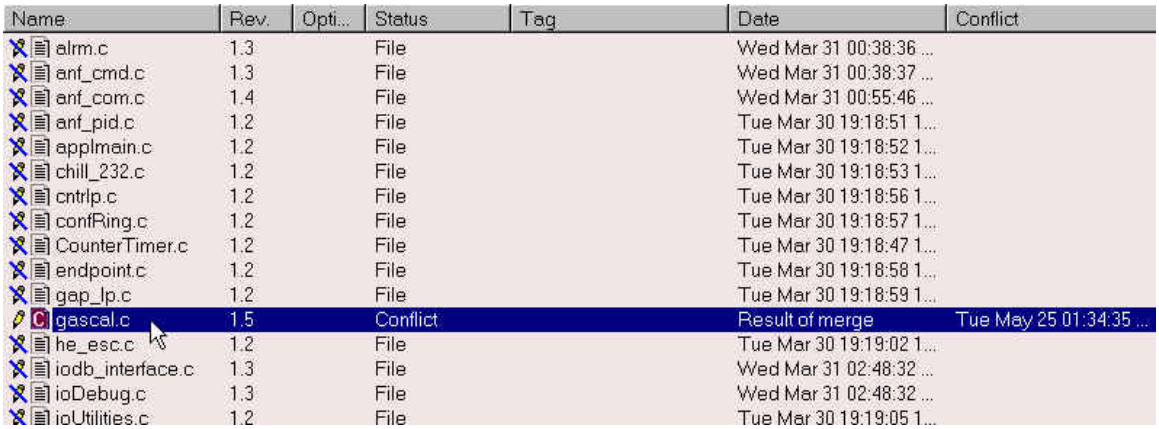
Name	Rev.	Opti...	Status	Tag	Date	Conflict
alrm.c	1.3		File		Wed Mar 31 00:38:36 ...	
anf_cmd.c	1.3		File		Wed Mar 31 00:38:37 ...	
anf_com.c	1.4		File		Wed Mar 31 00:55:46 ...	
anf_pid.c	1.2		File		Tue Mar 30 19:18:51 1...	
applmain.c	1.2		File		Tue Mar 30 19:18:52 1...	
chill_232.c	1.2		File		Tue Mar 30 19:18:53 1...	
cntrlp.c	1.2		File		Tue Mar 30 19:18:56 1...	
confRing.c	1.2		File		Tue Mar 30 19:18:57 1...	
CounterTimer.c	1.2		File		Tue Mar 30 19:18:47 1...	
endpoint.c	1.2		File		Tue Mar 30 19:18:58 1...	
gap_lp.c	1.2		File		Tue Mar 30 19:18:59 1...	
gascal.c	1.3		Mod. File		Result of merge	
he_esc.c	1.2		File		Tue Mar 30 19:19:02 1...	
iodb_interface.c	1.3		File		Wed Mar 31 02:48:32 ...	
ioDebug.c	1.3		File		Wed Mar 31 02:48:32 ...	
ioUtilities.c	1.2		File		Tue Mar 30 19:19:05 1...	

At this point the commit command can be run again causing the merged file to be copied to the repository.

In the case where the update command fails because cvs is unable to merge your changes with the repository version, messages similar to the following will be displayed:

```
cvs -q update gascal.c (in directory E:\WorkAreaOne\EtchPM\src)
RCS file: /Store/200mm/EtchPM/src/gascal.c,v
retrieving revision 1.4
retrieving revision 1.5
Merging differences between 1.4 and 1.5 into gascal.c
rcsmerge: warning: conflicts during merge
cvs server: conflicts found in gascal.c
C gascal.c

*****CVS exited normally with code 0*****
```

Notice the “C” indicating conflicts were found. In this case the status of the file will change to Conflict and the Date will change to Result of Merge as shown here:



Name	Rev.	Opti...	Status	Tag	Date	Conflict
alrm.c	1.3		File		Wed Mar 31 00:38:36 ...	
anf_cmd.c	1.3		File		Wed Mar 31 00:38:37 ...	
anf_com.c	1.4		File		Wed Mar 31 00:55:46 ...	
anf_pid.c	1.2		File		Tue Mar 30 19:18:51 1...	
applmain.c	1.2		File		Tue Mar 30 19:18:52 1...	
chill_232.c	1.2		File		Tue Mar 30 19:18:53 1...	
cntrlp.c	1.2		File		Tue Mar 30 19:18:56 1...	
confRing.c	1.2		File		Tue Mar 30 19:18:57 1...	
CounterTimer.c	1.2		File		Tue Mar 30 19:18:47 1...	
endpoint.c	1.2		File		Tue Mar 30 19:18:58 1...	
gap_lp.c	1.2		File		Tue Mar 30 19:18:59 1...	
gascal.c	1.5		Conflict		Result of merge	Tue May 25 01:34:35 ...
he_esc.c	1.2		File		Tue Mar 30 19:19:02 1...	
iodb_interface.c	1.3		File		Wed Mar 31 02:48:32 ...	
ioDebug.c	1.3		File		Wed Mar 31 02:48:32 ...	
ioUtilities.c	1.2		File		Tue Mar 30 19:19:05 1...	

In this case, the resulting merged file must be re-edited to resolve the conflicts manually. Conflicts can be found in the file by searching for the <<<<<<<, ========, and >>>>>>> character strings as shown in this example:

```
/*************************************************************
*
* new_function_from_user_1
*
*/
<<<<<<< gascal.c
void new_function_1( d, e, f )
=======
void new_function_1( a, b, c )
>>>>>>> 1.5
{
}
```

The developer must manually remove the lines containing <<<<<<<, =======, and >>>>>>> and resolve the conflicts on the indicated lines. After the conflicts have been resolved, the commit command may be re-issued.

If the developer attempts to commit the file without resolving the conflicts, the following error message will be printed by cvs and the commit will be aborted:

```
cvs -q commit -m "This is the change from user 1\n\n" gascal.c (in
directory E:\WorkAreaOne\EtchPM\src)
cvs server: file `gascal.c' had a conflict and has not been modified
cvs [server aborted]: correct above errors first!

*****CVS exited normally with code 1*****
```

If the developer resolves some of the conflicts but leaves some unresolved conflicts in the file, cvs will print the following warning message when the file is committed:

```
cvs -q commit -m "This is the change from user 1\n\n" gascal.c (in
directory E:\WorkAreaOne\EtchPM\src)
cvs server: warning: file `gascal.c' seems to still contain conflict
indicators
Checking in gascal.c;
/Store/200mm/EtchPM/src/gascal.c,v  <--  gascal.c
new revision: 1.6; previous revision: 1.5
done

*****CVS exited normally with code 0*****
```

Note that cvs will still commit the file even though it detected some of the conflict indicators <<<<<<<, =======, or >>>>>>>. This is probably not a good feature of cvs but this is how it works. If a file is accidentally committed with remaining conflict indicators, the file must be edited, re-modified, and commited again to remove the conflict indicators and resolve any remaining conflicts.

3.11.2 Understanding Locking and the Reserved Checkout Model

As stated in Section 3.11, cvs does provide a mechanism to support the *reserved checkout* model. The lock command is the basis of this model. Locking a file in cvs prevents another developer from either locking or committing a new version of the file. This is safe in that it does prevent the other developer from checking in a new version of the file but it does allow another developer to edit the file. This could lead to frustration, for example, if a developer forgets to get a file lock but just edits the file. When the developer attempts to commit the file, the operation will fail as shown in the following example:

```
cvs -q commit -m "This is the change from user 2\n\n" gascal.c (in
directory E:\WorkAreaTwo\EtchPM\src)
cvs [server aborted]: Revision 1.7 is already locked by apples

*****CVS exited normally with code 1*****
```

If all developers cooperate and ensure that files are locked prior to editing, this situation need not arise. Alternatively, the *unreserved checkout* model could be used.

As with other commands, WinCvs provides several ways to invoke the lock command:

1) Select the file(s) using the left mouse button and use Selections menu:

   Selections->Monitors selection->Lock selection

2) Select a file using the right mouse button and choose Monitors selection->Lock selection

3) Select the file(s) using the left mouse button and use the tool bar icon:

Use the unlock command to unlock the file in one of several ways:

1) Select the file(s) using the left mouse button and use Selections menu:

   Selections->Monitors selection->Unlock selection

2) Select a file using the right mouse button and choose Monitors selection->Unlock selection

3) Select the file(s) using the left mouse button and use the tool bar icon:

If a lock request succeeds, the following messages will be displayed on the status window:

```
cvs -q admin -l gap_lp.c (in directory E:\WorkAreaTwo\EtchPM\src)
RCS file: /Store/200mm/EtchPM/src/gap_lp.c,v
1.2 locked
done

*****CVS exited normally with code 0*****
```

If a lock request fails, information including the owner of the lock will be displayed:

```
cvs -q admin -l gascal.c (in directory E:\WorkAreaTwo\EtchPM\src)
RCS file: /Store/200mm/EtchPM/src/gascal.c,v
cvs [server aborted]: Revision 1.7 is already locked by apples

*****CVS exited normally with code 1*****
```

The Log command is available from all menus and the tool bar to determine the locking status of a file. The output is displayed in the status window as shown in this example where user don has revision 1.2 of gap_lp.c locked:

```
Rcs file : '/Store/200mm/EtchPM/src/gap_lp.c,v'
Working file : 'gap_lp.c'
Head revision : 1.2
Branch revision :
Locks : strict
      1.2 : 'don'
```

There are some cases where you may end up with locks on multiple revisions of a file. If you try to use the unlock command in this case you will get the following error:

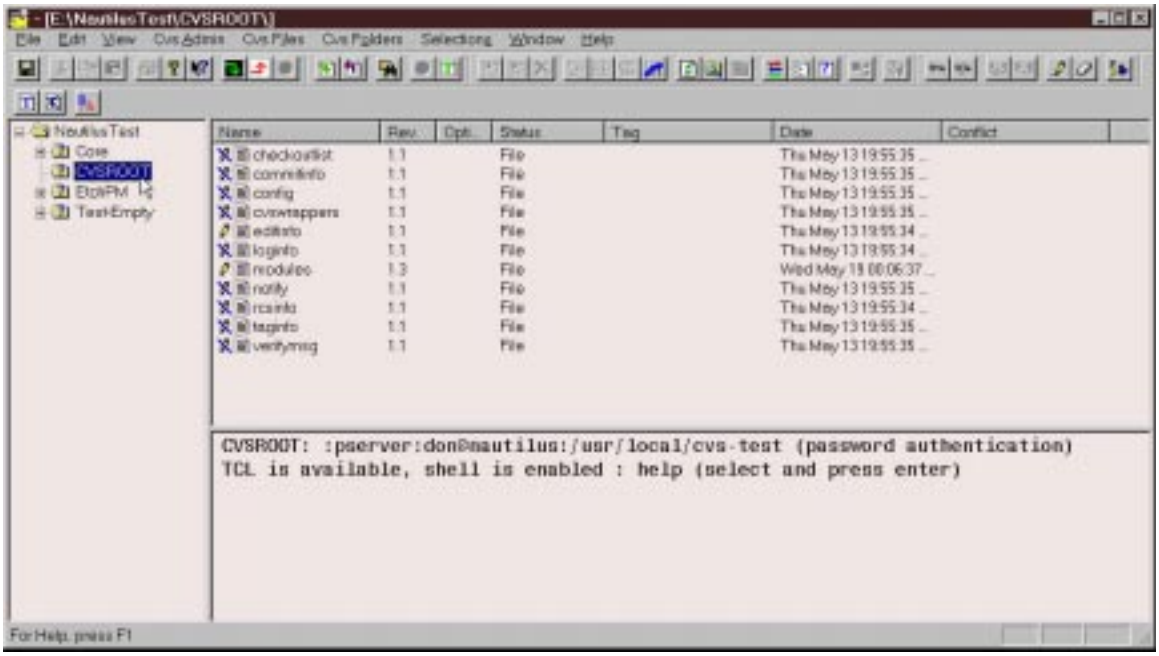
```
cvs -q admin -u gap_lp.c (in directory E:\WorkAreaTwo\EtchPM\src)
RCS file: /Store/200mm/EtchPM/src/gap_lp.c,v
cvs server: /Store/200mm/EtchPM/src/gap_lp.c,v: multiple revisions locked
by don; please specify one
cvs server: cannot modify RCS file for `gap_lp.c'

*****CVS exited normally with code 1*****
```

In this case, you must unlock a specific version of the file using the cvs admin command by manually typing it in the WinCvs status window. In this example, version 1.1 of gap_lp.c is specifically unlocked (the cvs –q admin –l1.1 gap_lp.c command is typed by the developer here):

```
cvs -q admin -l1.1 gap_lp.c

*****CVS exited normally with code 0*****

RCS file: /Store/200mm/EtchPM/src/gap_lp.c,v
done
```

Section 4 – Administrative Commands

This sections covers commands that should normally be used only by a cvs administrator. To edit any administrative files, the CVSROOT module must be checked out to a local work area. See Section 3.4 for information on the checkout command. A work area with a checked out version of CVSROOT is shown here:



Notice that there are 11 administrative files in the CVSROOT module that provide a way to manage many advanced features of cvs. See appendix C (Page 125) of the cvs reference manual for a complete description of all these files and features including the optional passwd file.

In this section we will desribe the use of the modules file only.

4.1 Editing the Modules Administrative File

The modules file in cvs lists names for folder hierarchies that can be checked out from the repository using the Cvs Admin->Checkout module… command from the WinCvs menu. Using the modules file, any part of a single module hierarchy or multiple existing modules can be aliases with a single module name.

If a module is imported to cvs and not listed in the modules file, the module may still be checked out but it will not be listed in response to the Cvs Admin->Macros admin->List the modules on the server command in the WinCvs menu.

In order to edit the modules file, the CVSROOT module but first be checked out as mentioned in the introduction to Section 4. The file must also be made editable as discussed in Section 3.6.

A sample modules file is shown here:

```
# Three different line formats are valid:
#   key -a    aliases...
#   key [options] directory
#   key [options] directory files...
#
# Where "options" are composed of:
#   -i prog     Run "prog" on "cvs commit" from top-level of module.
#   -o prog     Run "prog" on "cvs checkout" of module.
#   -e prog     Run "prog" on "cvs export" of module.
#   -t prog     Run "prog" on "cvs rtag" of module.
#   -u prog     Run "prog" on "cvs update" of module.
#   -d dir      Place module in directory "dir" instead of module name.
#   -l      Top-level directory only -- do not recurse.
#
# NOTE:  If you change any of the "Run" options above, you'll have to
# release and re-checkout any working directories of these modules.
#
# And "directory" is a path to a directory relative to $CVSROOT.
#
# The "-a" option specifies an alias.  An alias is interpreted as if
# everything on the right of the "-a" had been typed on the command line.
#
# You can encode a module within a module by using the special '&'
# character to interpose another module into the current module.  This
# can be useful for creating a module that consists of many directories
# spread out over the entire source repository.
#
EtchPM EtchPM
Core   Core
EtchPM-Source EtchPM/src
```

Section C.1 (page 125) of the cvs reference manual describes the modules file in detail. As with all modifications, the modules file must be committed to the respository (Section 3.9) before the changes will take effect.

4.2 Recovering from Locked Repository

Internally, cvs uses a lock in each repository to prevent simultaneous access by multiple users. The lock is actually a directory in the repository named `#cvs.lock`. If a cvs operation aborts, it is possible that this lock will remain in the repository preventing any further operations from proceeding. In this case, the message "waiting for user's lock" will be printed in the status window as shown in the following example where user R2D2 is the user holding the lock:

```
cvs admin -bLockTest2 -u applmain.c

*****CVS exited normally with code 0*****

cvs server: [13:46:22] waiting for R2D2's lock in /Store/200mm/EtchPM/src
```

Normally, this message means that user R2D2 is in the process of updating the repository. This should normally not take very long and the lock will be available as shown here:

```
cvs server: [13:46:52] waiting for R2D2's lock in /Store/200mm/EtchPM/src
cvs server: [13:47:22] obtained lock in /Store/200mm/EtchPM/src
RCS file: /Store/200mm/EtchPM/src/applmain.c,v
1.2.6.2 unlocked
done
```

In cases where cvs aborted for some reason (such as a network timeout or machine reboot), the lock may need to be removed manually. The `#cvs.lock` directory should be located in the indicated repository and removed using the UNIX rmdir command.

4.3 Release Management

The basic mechanism for maintaining multiple releases of a product using cvs is the tag. Tags are described in Section 4.4 (page 32) of the manual Version Management with CVS. When a product has been tested and ready for release with a given set of file revisions, the files can then be tagged with a symbolic name. Files may be modified after the release but the tagged version can always be retrieved.

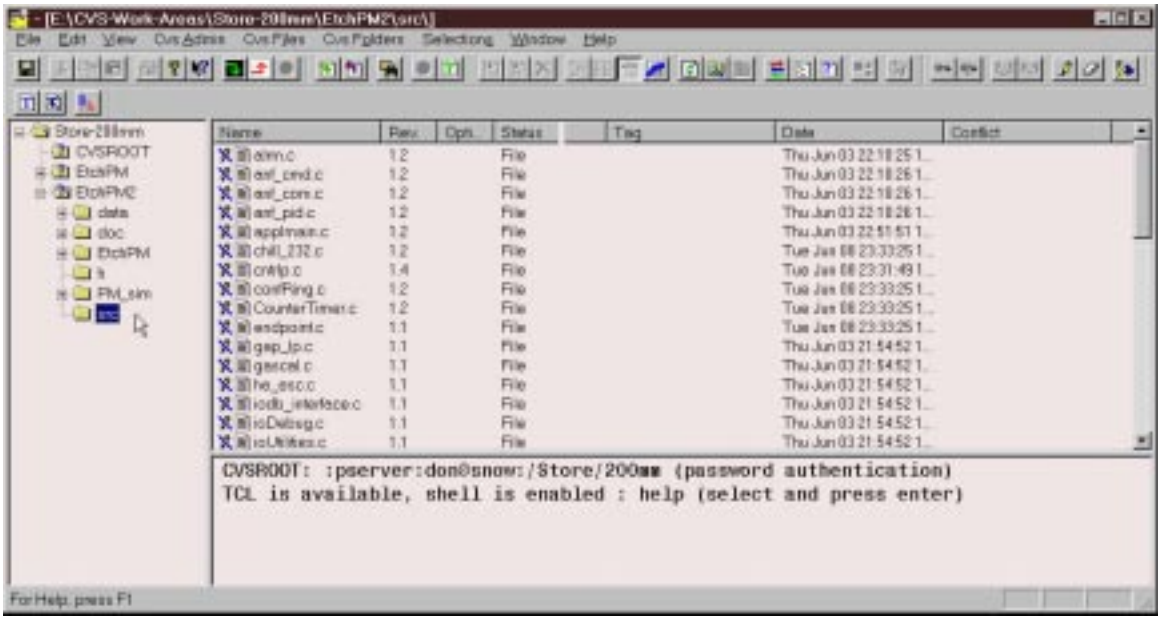
The problem that always comes up in release management is what to do when a bug is found after a product version has been released. Presumably, the files in the repository have been modified since the release and are not likely to be stable enough to be included in a bug fix release.

The solution to this problem is to create a branch from the version of the repository tagged at release time. Files can then be modified in the branch without interfering with continuing development on the trunk. After the fixes have been verified, the original release tag can be moved to the new file revisions or a new release tag can be created.

The following two sections show examples for both moving a tag to a new revision and creating a new release tag.

4.3.1 Tagging a Product Release

When a given set of files has been tested and ready for release, the work area from which the release was built should be tagged. In the following example, a work area for the EtchPM2 product is displayed. A partial list of files for the src folder is shown to illustrate a set of revision numbers (1.1, 1.2, and 1.4).



Assuming the EtchPM2 work area represents a tested set of files, the files can be tagged for release using the tag command.

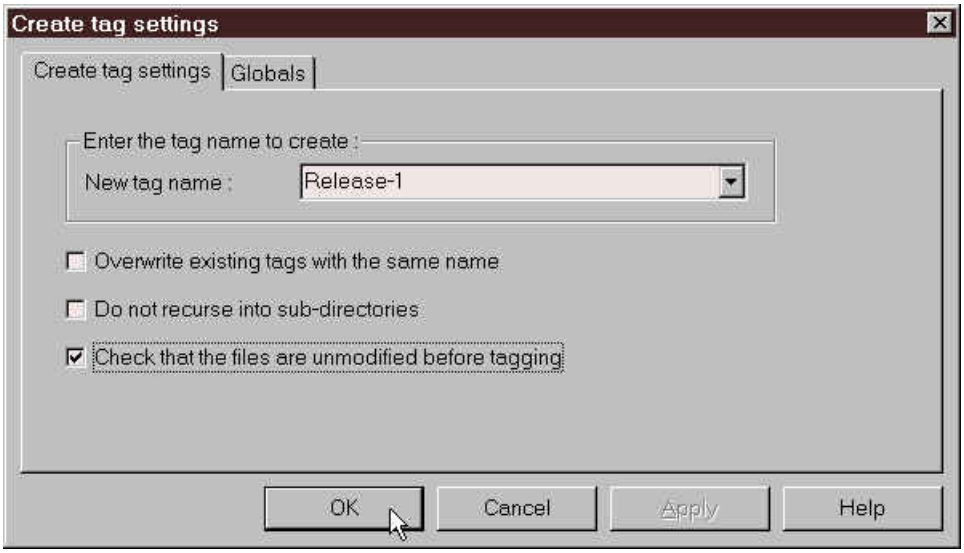
The tag command can be invoked in one of several ways:

1) Select the highest level folder (EtchPM2 in this case) using the left mouse button and use the menu: Selections->Tag selection->Create a tag…

2) Select the folder using the right mouse button and select from the pop-up menu: Tag selection->Create a tag…

3) Select the folder using the left mouse button and use the tool bar:

After the tag command is invoked the Create tag settings panel will be displayed as shown here:



The Create tag settings panel provides a field to enter the tag name and three options. The tag name may not contain any of the following characters: `$ , . : ; @`.

The Overwrite existing tags with the same name option specifies that any existing tags found with the same name should be moved to the version in the current work area.

The Do not recurse into sub-directories option may be useful in cases where a snapshot of a single folder is desired.

It is probably a good idea to use the Check that the files are unmodified before tagging option since it does not make much sense to have files in your release work area that are modified but not committed.

After the desired tag name and options are selected, click OK to begin the tagging process. During the tagging operation, cvs will display the files being tagged as shown here:

```
cvs -q tag -c Release-1 (in directory E:\CVS-Work-Areas\Store-
200mm\EtchPM2\)
T Makefile
.
.
*****CVS exited normally with code 0*****
```

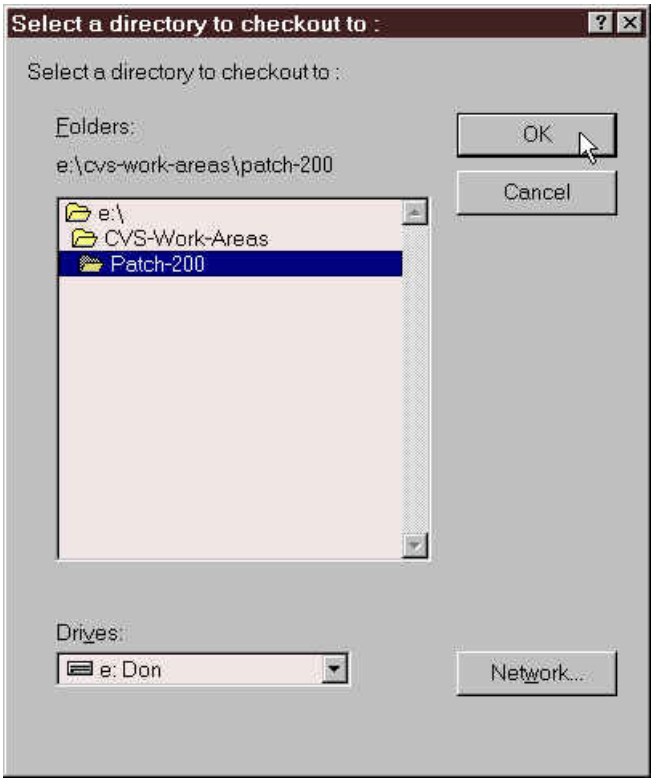
4.3.2 Fixing Bugs after Product Release

When a bug is discovered in a released product, the usual goal is to release an incremental release with the minimum set of changes required to the bug. The first step is to create a work area that can be used to build the incremental release. Depending on the product, this may require as little as a single file or as much as the entire repository tree. The work area will be set up to contain the revision of all files tagged at the time of the release.

4.3.2.1 Creating the Work Area

If a work area for the product or module already exists, the work area can be updated to the tagged version using the update command (Section 3.5). This method can easily lead to undesired results such as when modified files exist in the work area or when the folder hierarchy is different than the original tagged revision.

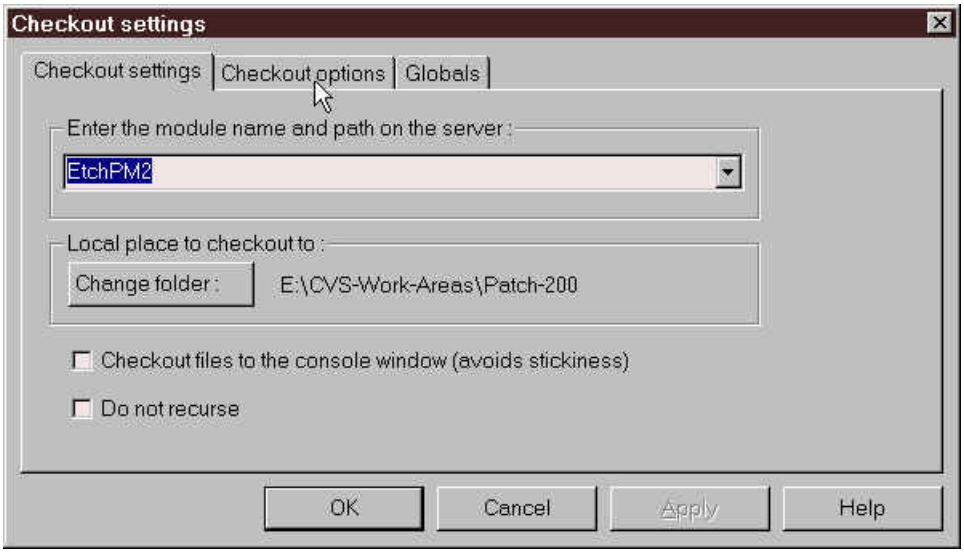
The preferred method to create a work area from a tagged release is to use the checkout command (Section 3.4) to check out a clean copy of the release to a new work area. When the checkout command is invoked, the following panel will be displayed:



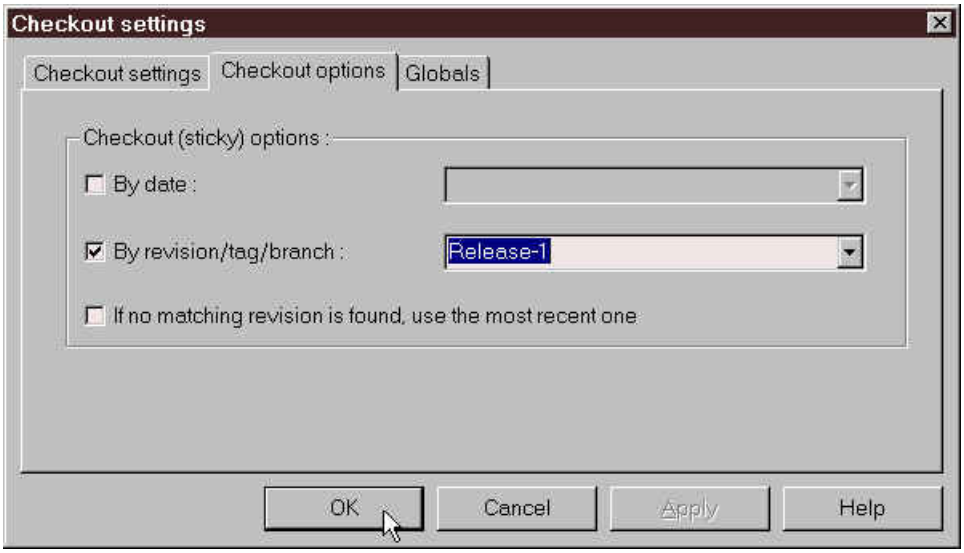
Select the folder where the work area will be created. Note that you must double-click on the desired folder so the folder icon is open as shown above.

After the desired folder is selected, click OK to continue with the checkout process. The Checkout settings panel will then be displayed.

The Checkout settings panel is shown below. Enter the desired module name and then proceed to the Checkout options panel by clicking on the tab as shown here:



The Checkout options panel is shown below. Check the By revision/tag/branch option and enter the desired tag name as shown here:

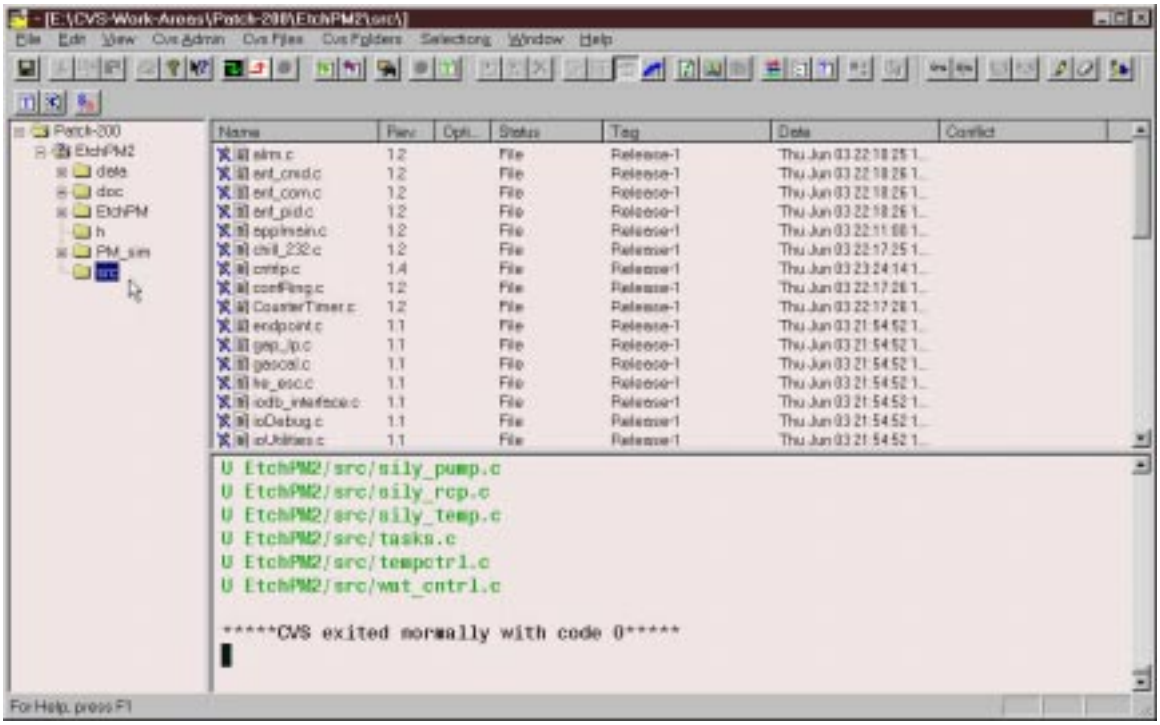


After the tag name is entered, click OK to initiate the checkout operation.

During the checkout operation, cvs will display each file as it is checked out as shown here:

```
cvs -q checkout -r Release-1 EtchPM2 (in directory E:\CVS-Work-
Areas\Patch-200)
U EtchPM2/Makefile
U EtchPM2/EtchPM/Makefile
.
.
.
U EtchPM2/h/gascal.h


*****CVS exited normally with code 0*****
```

After the checkout is complete, use the View->Change browser location command from the menu or tool bar to point to the newly created work area. An example of a work area created from the Release-1 tagged version is shown here:



Now that a work area has been created, a branch must be created to allow modification of the files necessary to fix the reported bug. The following section explains the way to create a branch from the tagged revisions in the work area just created.

4.3.2.2 Creating a Branch

The files in a work area updated to a tagged release cannot be modified and committed back to the repository. To make modifications to files, you need to be updated to a branch. If an appropriate branch already exists, use the update command (Section 3.5) to get the last branch revision into your work area. If a branch does not exist, create a branch as described in this section.

As with other commands, WinCvs provides several ways to invoke the branch (fork) command:

1) Select the file(s) or folder(s) using the left mouse button and use the menu entry:

   Selections->Tag selection->Create a branch…

2) Select the file(s) or folder(s) using the left mouse button and use the right mouse button to open the Selections menu. Choose the menu entry:

   Tag selection->Create a branch…

3) Select the file(s) or folder(s) using the left mouse button and use the tool bar icon:

When the branch (fork) command is invoked, the Create branch settings panel will be displayed as shown here:

Enter the desired name for the branch and hit the OK button. It is a good idea to give the branch a name that clearly indicates its purpose. In this case, we are doing further development off the Release-1 tagged release so Release-1-Development seems appropriate.

The Create branch settings panel has two options that may be appropriate in certain cases. It is normally a good idea to set the option to check for modified files since it doesn't make sense to branch from a directory of files that haven't been committed. It is often useful to set the Do not recurse option also since it is common to branch a single folder and not need branches in the sub-folders (the default).

The status window will display the progress of the branch creation as shown here:

```
cvs -q tag -b -c Release-1-Development (in directory E:\CVS-Work-
Areas\Patch-200\EtchPM2\src\)
T CounterTimer.c
T Makefile
.
.
.
T PM_WaferFlow.c

*****CVS exited normally with code 0*****
```

The branch will be created with the current version of files checked out in your working copy. Notice that WinCvs uses the tag command with the –b option to create the branch. This command can also be entered manually from the WinCvs status window.

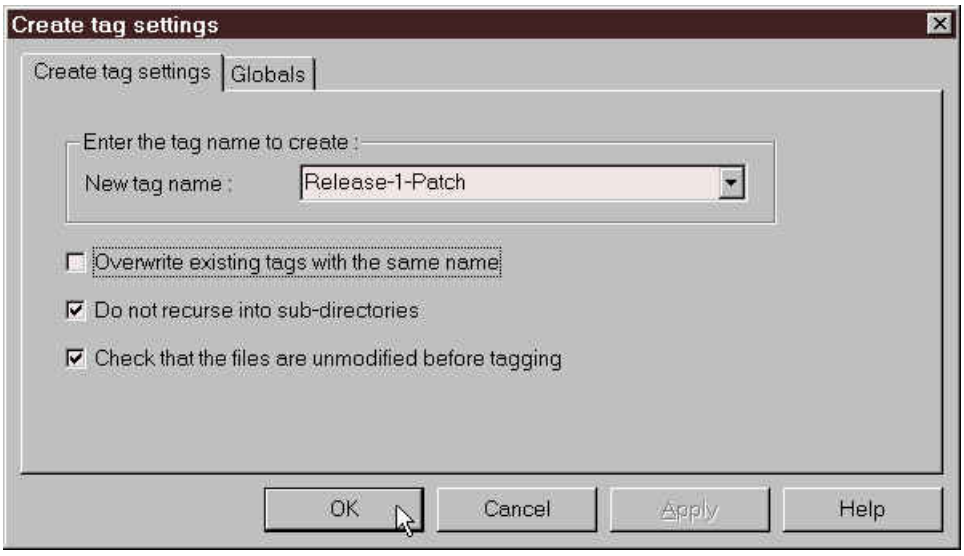
Note that simply creating a branch does not update the work area to the branched version. It is necessary to use the update command (Section 3.5) to update to the newly created branch.

Once the branch is created and the work area is updated, files can be modified and committed to the branch as described in Sections 3.6 through 3.9. When all modifications are complete and tested. There are two possibilities for creating a new tagged release described in the following section.

4.3.2.3 Creating the New Tagged Release

In the previous example, a tagged release called Release-1 was checked out and some files were modified by creating a branch called Release-1-Development. At this point, the release manager has two options: the modified files can be re-tagged with the original tag name Release-1 or a new tagged release can be created. In cases where Release-1 has already been widely distributed to internal or external customers, the original Release-1 tagged release should most likely not be modified requiring a tagged release with a new name. If Release-1 was not distributed it may be reasonable to re-tag the modified files with the original Release-1 tag. This is also referred to as moving tags since the tags are effectively moved from one revision to another for the modified files.

In either case, the tag command needs to be invoked as described in Section 4.3.1. The following settings would be used to create a new tagged release called Release-1-Patch:



The following settings would be used to re-tag the new revisions with the original Release-1 tag: