Wildcards

Wildcards let you specify a file or group of files by typing a partial filename. The appropriate directory is scanned to find all of the files that match the partial name you have specified.

Wildcards are usually used to specify which files **should** be processed by a command. If you need to specify which files should **not** be processed see File Exclusion Ranges (for internal commands), or EXCEPT (for external commands).

Most internal commands accept filenames with wildcards anywhere that a full filename can be used. There are two wildcard characters, the asterisk [*] and the question mark [?], plus a special method of specifying a range of permissible characters.

An asterisk [*] in a filename means “any zero or more characters in this position.” For example, this command will display a list of all files in the current directory:

```
[c:\] dir *.*
```

If you want to see all of the files with a *.TXT* extension, you could type this:

```
[c:\] dir *.txt
```

If you know that the file you are looking for has a base name that begins with ST and an extension that begins with *.D*, you can find it this way. Filenames such as *STATE.DAT*, *STEVEN.DOC*, and *ST.D* will all be displayed:

```
[c:\] dir st*.d*
```

With **CMD.EXE**, you can also use the asterisk to match filenames with specific letters somewhere inside the name. The following example will display any file with a *.TXT* extension that has the letters AM together anywhere inside its base name. It will, for example, display *AMPLE.TXT*, *STAMP.TXT*, *CLAM.TXT*, and *AM.TXT* :

```
[c:\] dir *am*.txt
```

A question mark [?] matches any single filename character. You can put the question mark anywhere in a filename and use as many question marks as you need. The following example will display files with names like *LETTER.DOC* and *LATTER.DAT*, and *LITTER.DU* :

```
[c:\] dir l?tter.d??
```

The use of an asterisk wildcard before other characters, and of the character ranges discussed below, are enhancements to the standard wildcard syntax, and may not work properly with software other than **CMD.EXE** and Take Command.

“Extra” question marks in your wildcard specification are ignored if the file name is shorter than the wildcard specification. For example, if you have files called *LETTER.DOC*, *LETTER1.DOC*, and *LETTERA.DOC*, this command will display all three names:

```
[c:\] dir letter?.doc
```

The file *LETTER.DOC* is included in the display because the “extra” question mark at the end of *“LETTER? ”* is ignored when matching the shorter name *LETTER*.

In some cases, the question mark wildcard may be too general. You can also specify what characters you want to accept (or exclude) in a particular position in the filename by using square brackets. Inside the brackets, you can put the individual acceptable characters or ranges of characters. For example, if you wanted to match *LETTER0.DOC* through *LETTER9.DOC*, you could use this command:

```
[c:\] dir letter[0-9].doc
```

You could find all files that have a vowel as the second letter in their name this way. This example also demonstrates how to mix the wildcard characters:

```
[c:\] dir ?[aeiouy]*.*
```

You can exclude a group of characters or a range of characters by using an exclamation mark [**!**] as the first character inside the brackets. This example displays all filenames that are at least 2 characters long **except** those which have a vowel as the second letter in their names:

```
[c:\] dir ?[!aeiouy]*.*
```

The next example, which selects files such as *AIP*, *BIP*, and *TIP* but not *NIP*, demonstrates how you can use multiple ranges inside the brackets. It will accept a file that begins with an **A**, **B**, **C**, **D**, **T**, **U**, or **V**:

```
[c:\] dir [a-dt-v]ip
```

You may use a question mark character inside the brackets, but its meaning is slightly different than a normal (unbracketed) question mark wildcard. A normal question mark wildcard matches any character, but will be ignored when matching a name shorter than the wildcard specification, as described above. A question mark inside brackets will match any character, but will not be discarded when matching shorter filenames. For example:

```
[c:\] dir letter[?].doc
```

will display *LETTER1.DOC* and *LETTERA.DOC*, but not *LETTER.DOC*.

A pair of brackets with no characters between them [], or an exclamation point and question mark together [**!?**], will match only if there is no character in that position. For example,

```
[c:\] dir letter[].doc
```

will not display *LETTER1.DOC* or *LETTERA.DOC*, but will display *LETTER.DOC*. This is most useful for commands like

```
[c:\] dir /I"[]" *.btm
```

which will display a list of all .BTM files which don't have a description, because the empty brackets match only an empty description string (DIR /I selects files to display based on their descriptions).

You can repeat any of the wildcard characters in any combination you desire within a single file name. For example, the following command lists all files which have an **A**, **B**, or **C** as the third character, followed by zero or more additional characters, followed by a **D**, **E**, or **F**, followed optionally by some additional characters, and with an extension beginning with **P** or **Q**. You probably won't need to do anything this complex, but we've included it to show you the flexibility of extended wildcards:

```
[c:\] dir ??[abc]*[def]*.[pq]*
```

You can also use the square bracket wildcard syntax to work around a conflict between long filenames containing semicolons [;], and the use of a semicolon to indicate an include list. For example, if you have a file on an HPFS drive named *C:\DATA\LETTER1;V2* and you enter this command:

```
[c:\] del \data\letter1;v2
```

you will not get the results you expect. Instead of deleting the named file, **CMD.EXE** will attempt to delete *LETTER1* and then *V2*, because the semicolon indicates an include list. However if you use square brackets around the semicolon it will be interpreted as a filename character, and not as an include list separator. For example, this command would delete the file named above:

```
[c:\] del \data\letter1[;]v2
```