Programmer's File Editor Help

Welcome to Programmer's File Editor, a programmer-oriented text editor for Windows 3.1 and later.

PFE is a standard Windows Multiple Document Interface (MDI) application, which means that you can operate it in the same way as any other standard Windows application. It features many powerful facilities, though, which you will find described in the sections in this help file.

What is Programmer's File Editor?

Programmer's File Editor (PFE) is a large-capacity text file editor designed to be used with Windows 3.1 and later releases. It is oriented towards those who use Windows as their program development environment, and so incorporates many features that make it a convenient work management system. For example, the editor has some awareness of C syntax; and there are menu items to make running DOS-mode compilers and Windows-based development tools easy and quick.

PFE's capacity is essentially limited only by the total amount of memory available on your system. There are no editor-imposed limits on the number of files that you can edit simultaneously, nor on the number of edit windows that you may have open. There is no limit on the size of file that can be handled, and none on the number of lines that a file may contain. One arbitrary limit *has* been built in: no file line may exceed 1024 characters. This value may be increased by changing a few definitions and recompiling from the source.

Many of the features in PFE have been developed from techniques used in MicroEMACS, a public domain DOS mode editor (although no MicroEMACS code, and no internal design, has been used).Thus many functions can be invoked from the keyboard by pressing two keys in sequence: for example **Ctrl-X** followed by **2** duplicates a window. If you disagree with the command mappings that are built in, you change change them: PFE allows you to change the keys associated with almost all of its functions.

Apart from this optional use of two-key command sequences, PFE adheres strictly to the Windows MDI conventions. You can invoke most commands and facilities from menus; you can move around with a mouse or with the standard keyboard shortcuts; you can cut and paste from the clipboard, and so on.

PFE is also able to run DOS commands, such as compilers, and to capture their output into windows for inspection. This lets you use it as an integrated development environment, cutting down the amount of work you need to do to build and test your applications. You can also quickly launch the application you're developing, and you can configure details of the development tools that you use, so that these too can be launched with only a few mouse clicks.

There are also some features that let you build files more easily. You can define sets of **templates** - standard lines of text - that you can insert into the file you're editing with just a few mouse clicks. You can group the templates you work with into distinct files, and load them for use automatically.

Why should you use PFE? If you already have a favourite editor, fine. PFE is not sold for money, so the author has no need to persuade you to do something you don't want. However, you might like it; and since it's supplied in source form, you can also do something to change those bits that you *don't* like. You are also very welcome to contribute ideas for improvements, and (especially) actual code for new features. Especially in the beta releases (which have version numbers like 0.xxx.xxx) there is a great deal of scope for changing and improving things, so let the author know what you think.

Distributing and contributing

For the purposes of this section, "Programme's File Editor" is used to mean the editor itself, and all the various support programs that are needed to build it and to operate it.

Programmer's File Editor may be freely used by any individual or non-commercial organisation for any purpose. You may distribute it to anyone, and you may place it on any archive or bulletin board system for wider access. You may not charge anyone for it other than a reasonable fee to cover your distribution costs.

Normally, you should distribute PFE in the form as supplied by the author; however, you may repackage it to suit the conventions and needs of an archive or bulletin board system if you wish.

Versions of PFE from 1.00.000 onwards will be issued with the full C language source. You should normally include the sources as part of what you distribute; but if you do not do so you must either undertake to provide them to the recipient if they are required, or indicate how he may obtain them.

You are free to modify the sources in any way to customise your own version of PFE. You may distribute the executable program so derived, but you must:

- Retain the original About dialog box with the author's credits (although you may add to it)

- Retain the original source file headers and all the original copyright attributions in all places

- Change the version resource and identification strings to make it clear that the program is not from the main distribution. You can see how to do this from the documentation on development

You must not distribute sources containing your own modifications.

You may utilise all or part of the sources for any non-commercial purpose, but you must not include any part of the code or any part of the editor's design obtained from the code in any commercial product. You should include an acknowledgement in your documentation if you use any substantial portion of PFE.

If you discover any faults in PFE, or if you have developed your own fixes or enhancements, you are welcome to pass them back to the author for inclusion (with full credit) in a future release. You might like to look at the list of known problems and enhancements that the author and others are known to be working on before you begin work.

The author can be contacted by e-mail at

alan@lancaster.ac.uk (Internet)
alan@uk.ac.lancaster (JANET)

New versions of PFE will be uploaded to the HENSA/micros archive in the UK, and the SIMTEL20 and CICA archives in the United States by the author.

Handling files

PFE lets you open existing files, create new ones, save changes, and insert one file into another, as you would expect.

Opening an existing file

To start editing an existing file, select the **Open File** item from the **File** menu or press the key sequence **CTRL-F O**. You can also click the right mouse button anywhere within the PFE desktop to do this, or use the tool bar

This starts a standard dialog that allows you to move around your directories to find the file you want to edit. You can either browse till you find the one you want, or you can type the name into the edit control.

If you want the file to be marked as *read only* when it is loaded, click the **Read Only** box. (You can change the *read only* state of a file at any time using the **Settings** menu). You can also use the **File View** option in the **File** menu to open a file in *read only* mode.

The **List Files of Types** box at the lower left lets you restrict the files shown in the list to those of specific file types. By default, the list selects all files, but you can configure this as you wish.

When the file is loaded, PFE looks at the *file type* to determine such things as the tab size to use, whether to treat the file as C language source, and so on. You can configure the settings it uses for different file types with the **Configure Defaults** item in the **Settings Menu**.

If you select a file that is already open, PFE will not load a new copy from disk. Instead, it will activate one of the windows that are currently showing the file.

PFE will examine the contents of the file specified to see if it is a template file. If it is, it will be attached rather than opened for editing.

You can also open files for editing by dragging-and-dropping their icons from File Manager; by specifying their names in a command line invocation; and over a DDE link from another application

Dragging and dropping from File Manager

You can instruct PFE to open one or more files for editing from the File Manager. To do this:

1. Select the files you want to edit in the File Manager window
2. Click on one of the files and hold the left mouse button down
3. Move the mouse to position the file icon over PFE's window and release the left mouse button

You can drag and drop as many files as you wish in one operation.

If you drag and drop files into PFE when it is an icon, it will automatically restore itself into a window.

Specifying files on the command line

PFE can be invoked with one or more files named on the command line, and will open them in turn for editing. You can associate files of specific types with PFE through the File Manager, allowing you to edit them by double-clicking on their names in the File Manager window.

Creating a new file

To create a window in which you can edit a new file, select the **New** item in the **File** menu, or press the key sequence **CTRL-F N**. You can also hold down the **CONTROL** key and click the right mouse button anywhere in the PFE desktop, or use the tool bar

PFE will open a new window, which will show the caption, for example,

```
Un-Named 3
```

The tab size and other settings will be the default values you set up with the **Configure Defaults** item in the **Settings** menu.

Inserting an existing file

To insert an existing file into the window that you're editing, place the caret to the point at which you want the insertion to be made, then select the **Insert** option in the **File** menu or press the key sequence **CTRL-F I**.

This starts a standard dialog that allows you to move around your directories to find the file you want to insert. You can either browse till you find the one you want, or you can type the name into the edit control.

PFE will insert the file into the window, leaving the caret positioned after the last character inserted.

If the file to be inserted is a template file, PFE will reject the operation.

Saving a file

To save a file to disk, select the **Save** option in the **File** menu, press the key sequence **CTRL-F S**, or use the tool bar. PFE will write the file's contents back to disk; anything the file contained previously will be lost.

If you haven't altered the file, you can't use the **Save** option. Nor can you do it if you created the window with the **New** option in the **File** menu and haven't yet saved it to a named file on disk. In these cases, you should select the **Save As** option from the **File** menu, or press the key sequence **Ctrl-F A**, which will enable you to specify the name of the file to be written to; or use the **Write** option from the **File** menu.

For this option, PFE will show you a standard dialog. As with opening a file, you can either type in the name, or click on one in the list. For saving, PFE will ask you for confirmation if you give the name of a file that already exists.

If the current window contains a template, you won't be able to use the **Save** or **Save As** options. Instead, use the **Store** or **Store As** options in the **Templates** menu.

Writing a file

PFE will not allow you to use the **Save** option in the **File** menu if the file you're editing has not changed, or if you haven't yet established a name for a newly-created file. Nor can you use this option if you're editing a template. However, you can always write the contents of the current window to disk by selecting the **Write** option in the **File** menu, or by pressing the key sequence **Ctrl-F W**.

This option will start a dialog asking you to give the name of the file you want to save the current window's contents into. If the file already exists, you'll be asked to confirm that this is really what you mean to do.

Once the save is complete, you can carry on editing in the window. The name of the file that PFE associates with the winfow - shown in the window's caption bar - is not changed by this operation.

Closing a file

When you've finished editing a file, you can close it by selecting the **Close** option in the **File** menu, or by pressing the key sequence **Ctrl-F C**. PFE will close all the windows that are showing the same file as the current window.

If you haven't yet saved some changes to the file, PFE will give you the opportunity to either write the file to disk before it is closed, or to abandon the operation.

If you want to close *all* the files that you are working on, select the **Close All** option in the **File** menu, or press the key sequence **Ctrl-F X**.

Configuring the File Type list

Whenever you use the **Open**, **Save As** or **Write** options in the **File** menu, PFE will show you a standard Windows dialog that allows you choose the filename to use, moving around all your directories.

The possible file names in the current directory are shown in the list on the left; the files shown in this list are determined by your selection from the **List Files of Type** list at bottom left.

By default, there will be only one item in the list, which selects all files, using a pattern of "***.***". If you wish, you can configure this list to let you select a smaller group of files: for example, if you program in C, you might wish to see only files with types of "**,c**" and "**.h**" in one group, and files of type "**.dlg**" in another.

The contents of the list are determined by the **[fileopen]** section in the file **pfe.ini**, which is held in your Windows directory. For example, you could edit the file to include the section

[fileopen]
Source Files (c,h)=*.c,*.h
Linker files (def,lnk)=*.def,*.lnk
Make files=makefile
All Files=*.*

This example sets up four groupings of files.

The text to the left of the "**=**" on each line is shown in the list box; you can set this to whatever you wish. To the right of the "**=**" sign you should place a comma-separated list of wildcard file names, such as "***.c**". You can use any of the standard DOS wildcard characters in the names; however, they must not include any drive letters or directory names.

Any changes you make to the **[fileopen]** section do not take effect until the next time you start PFE.

Editing techniques

You edit with Programmer's File Editor in the same way that you do with NotePad or any other standard Windows application. You'll find that the mouse works in the same way as with any other Windows program; you can scroll windows as in any other application; and you can use the Windows clipboard to communicate with other programs.

The sections here give you details of the various editing techniques and facilities of PFE. Normal Windows operations are briefly noted, and there are full details of specific PFE features that you will find helpful in your work.

Undoing edit changes

This beta release of PFE does not allow you to undo edit changes. The facility will be added for the first full version.

Manipulating windows

PFE uses the standard Windows Multiple Document Interface to allow you to edit as many files as you wish simultaneously. Click an item in the list below for more details:

Duplicating a window

PFE allows you to have as many windows as you want open for the *same* file. Selecting the **Duplicate Window** item on the **Windows** menu, or typing the key sequence **CTRL-X 2**, will make an *exact* duplicate of the current window. PFE will create a second window that is the same size as the original, with all the same window modes, showing the same text.

Having the ability to have multiple view of a file can be extremely useful if you're moving text around a great deal, if you constantly need to refer to a part of the file a long way from where you are currently editing.

You can edit the file concerned in any of the windows that are showing it, but there will be a small performance penalty, as PFE needs to synchronise a change made in one window with all the others.

Although you can select text freely when you have multiple windows on a file, only *one* of the windows can have an active selection at any time. If you select some text and then activate another window on the *same* file, the selection will be cleared.

Closing a window

To close a window, you can select the **Close** item on the **Windows** menu, or type the key sequence **CTRL-X 0**. If the window is the only one currently showing a file, the file is closed also; if you have made any changes to the file you'll be prompted to see if you want to save them, discard them, or cancel the operation.

Selecting a window

You have several ways of moving from one edit window to another.

On the keyboard, you can press the key sequence **Ctrl-X N** to move between them. However, this only moves you between windows that are not currently icons - any iconic windows are not restored by this.

With a mouse, you can click on any window to activate it. To prevent you accidentally moving the caret when you activate a window with the mouse cursor within the edit area, PFE will not change the caret position until you click a second time.

The **Windows** menu contains one item for each of the first ten edit windows; clicking the item that identifies the window will activate it (if the window is an icon, it will be restored). If you have more than ten windows open, you'll see a **More Windows** item, which will run a dialog that will list all the existing windows.

Moving and sizing a window

You can iconize any edit window by clicking on its minimize box, or via its system menu. You can also iconize *all* the edit windows that exist in one operation by selecting the **Iconize All** item on the **Windows** menu, or pressing the key sequence **Ctrl-X I**.

You can maximize a window so that it fills the full extent of the main window by clicking its maximize box, or via its system menu. You can also double-click anywhere on the window's caption bar to do this.

You can move a window around or resize it with the mouse, or via the window's system menu.

Widening a window

Often you may want to make a window as wide as possible, so that you can see the maximum possible amount of text in each file line. You can achieve this in the normal way that Windows allows you to resize a window; but an easier way is to select the **Widen** option in the **Windows** menu, or press the key sequence **CTRL-X W**.

PFE will automatically make the current window as wide as possible, moving it to the far left of the main window. The window's depth and vertical position are not affected.

The status bar

PFE normally shows you a **status bar** at the bottom of its window. This gives you several items of useful information:

- The line number and column number where the caret is
- The total number of lines in the file
- Whether the file has changed : you will see a **C** if it has
- Whether the file is *read only* : the characters **RO** will show if it is
- The command sequence prefix key - that is, whether you've typed one of **ESC, CTRL-B, CTRL-F** or **CTRL-X** and PFE is waiting for you to supply the second key of the sequence
- Whether your keystrokes and menu selections are being recorded in a keyboard macro : the characters **REC** will show if they are
- Whether typing is to be *inserted* into what's there already, or will *overwrite* it : you see **INS** or **OVR** respectively
- The state of the **Num Lock** and **Shift Lock** keys

You can turn off the status bar and make a little more room for showing the files you're editing, or turn it on again, by checking and unchecking the **Status Bar** item in the **Settings** menu. On slower machines you will probably see a noticeable performance increase when you have the status bar turned off.

You can also turn the status bar off by double-clicking the left mouse button anywhere within it.

Using keyboard macros

PFE is able to record key strokes and menu selections in a **keyboard macro**, which you can replay as many times as you wish to make repetitive operations easier.

Selecting the **Record Keyboard Macro** item from the **Execute** menu, typing **Shift-F7** or clicking the appropriate button on the tool bar will start the recorder. PFE will record anything you type into an edit window, and any menu items that you select into the keyboard macro buffer; recording will continue until you select the **End Keyboard Macro** item, click the button on the tool bar again, or press **Ctrl-F7**.

Once you've recorded a keyboard macro, you can replay it by selecting the **Replay Keyboard Macro** item, or type **F7**. PFE will re-issue all the keystrokes and menu selections that have been recorded, just as if you were performing them yourself.

It's up to you to ensure that it's sensible to replay a keyboard macro. For example, if you don't have a file open, it's not possible for PFE to insert any characters. So that you don't accidentally ask PFE to perform some impossible operation, replay of the keyboard macro will be automatically halted if any command or keystroke fails. This can be very useful if you're performing searches: macro replay will stop if the target string is not found.

Although it records menu selections made by both keyboard and mouse, PFE does *not* record other mouse operations. You should, therefore, take care to use only the keyboard equivalents of mouse operations when you're recording a macro.

PFE also does not record what you type into dialog boxes. Thus, if you record an **F2** key for a search, you'll see the dialog box every time you replay the key, and will have to fill in the details again. In many cases, this is how you would want the editor to operate, but sometimes you actually want to repeat a search operation exactly every time. To achieve this, you can type **F2** to perform a search *before* you start recording; then, within the macro you can use the **F2** key to repeat the search exactly without seeing the dialog box.

PFE provides only *one* keyboard macro; starting a recording will automatically delete any previous macro. In a suture release you'll be able to select from multiple keyboard macros, and to record them to files for use in subsequent edit sessions.

Setting window and file modes

PFE allows you to change several aspects of how it operates in any particular window. You can alter these aspects, called **modes**, at any time you wish; and you can also configure PFE to use specific sets of modes depending on the file type of the file you are editing.

This lets you, for example, tell PFE that all files with type "**.c**" and "**.h**" should be edited with a tab size of 4 and auto-indenting, while all other files need a tab size of 8 and no auto-indenting.

Modes are set by options in the **Settings** menu. The **Current Settings** option affects only the current window, and **Default Settings** lets you associate a set of modes with a particular file type.

Setting default modes
Setting current modes

Setting default modes

In order to associate a set of modes with a given file type you should select the **Default Settings**option in the **Settings** menu. This starts up a dialog box, divided into several sections.

The area at the top tells PFE which of three default settings you're configuring.

The **New Files** button sets up the modes to be used for any window you create with the **New** option in the **File Menu**.

The **File Type** button sets up modes for specific file types, such as "**.c**" or "**.asm**". Whenever you use the **Open** option in the **File** menu to open a file with one of the configured types, PFE will automatically use the corresponding modes in the window it uses. If you select this button you can choose a file type from the list box beside it, or type in a new file type. You must always include the leading "**.**" when giving the type. You can use the **Delete** button to remove items from this list.

The **Other Types** button sets modes for all other files - these settings are used when you open an esiting file whose file type has not been specified.

Once you've chosen one of the buttons in the area at the top, you can set the modes in the rest of the dialog box to meet your needs.

Within the **Display/Input Modes** area, the boxes do this:

Auto Indenting tells PFE that whenever you press **ENTER** to start a new line, it should automatically insert the same leading space as in the line above.

Strip Trailing Spaces tells PFE that whenever you press **ENTER** to end a line, it should remove any trailing space or tab characters.

Show Line Numbers tells PFE to display the number of each line in the window. You can also turn line number display on or off with the key sequence **CTRL-X N**, or by using the tool bar

Overwrite tells PFE whether what you type is to be inserted into existing text, or to overwrite what is already there. You can also switch between *insert* and *overwrite* modes by pressing the **INS** key.

The **Language** box lets you select whether PFE is to perform some language related operations automatically, and what language it is to assume when doing so. Currently only the C programming language is supported.

Within the **Screen Formatting** area, the boxes do this:

The **Tab Size** box lets you specify how tab characters are to be displayed. Whenever you type a tab PFE will always store it as one single character in the file, but you can configure how wide a tab stop is to be when it is displayed.

Within the **Printing** area are these boxes:

The **Page Headers** box tells PFE whether you want it to head each output page with a title showing the name of the file, the date and time, and the page number

The **Wrap Long Lines** box tels PFE what to with lines that are too long to fit across the

printed page. If the box is checked, an overlong line in the file will be folded to the next printed line; if it isn't checked, the text will be truncated at the right hand side of the page.

Once you've set the modes you want, you can apply them by clicking the **Apply** button. The settings you establish will have effect only for the current session of PFE. If you want to record the settings permanently, click the **Save** button.

*In this beta version of PFE, the **Wrapping Enabled** checkbox has no effect.*

Setting current modes

To change the modes associated with the *current window* only, and the *current file*, select the **Current Settings** option in the **Settings** menu. This starts up a dialog box, which is divided into several sections

The dialog box reminds you of the name of the current file at the top. You should set the modes that you want by checking or unchecking the boxes, then press **OK** to action them,

Within the **Display/Input Modes** area, the boxes do this:

Auto Indenting tells PFE that whenever you press **ENTER** to start a new line, it should automatically insert the same leading space as in the line above.

Strip Trailing Spaces tells PFE that whenever you press **ENTER** to end a line, it should remove any trailing space or tab characters.

Show Line Numbers tells PFE to display the number of each line in the window. You can also turn line number display on or off by typing **Ctrl-N**, or by using the tool bar

Overwrite tells PFE whether what you type is to be inserted into existing text, or to overwrite what is already there. You can also switch between *insert* and *overwrite* modes by pressing the **INS** key.

The **Language** box lets you select whether PFE is to perform some language related operations automatically, and what language it is to assume when doing so. Currently only the C programming language is supported.

The **Tab Size** box lets you specify how tab characters are to be displayed. Whenever you type a tab PFE will always store it as one single character in the file, but you can configure how wide a tab stop is to be when it is displayed.

Within the **Printing** area are these boxes:

The **Page Headers** box tells PFE whether you want it to head each output page with a title showing the name of the file, the date and time, and the page number

The **Wrap Long Lines** box tels PFE what to with lines that are too long to fit across the printed page. If the box is checked, an overlong line in the file will be folded to the next printed line; if it isn't checked, the text will be truncated at the right hand side of the page.

The windows modes that you change here affect only the one current window; any other window showing the same file is not changed. The file settings, though, apply to all windows showing the file.

*In this beta version of PFE, the **Wrapping Enabled** checkbox has no effect.*

Language awareness

PFE contains some awareness of the format of programming languages, to help you when writing program sources. Currently only the C language is supported.

To select language awareness, you need to specify the language in the modes for the window you are editing in. You can specify that any file with a given file type should use a particular language automatically, or you can change the current window's language at any time.

When editing files that have the **C language** mode set, PFE provides these facilities:

- A **#** character typed in an otherwise empty line is always moved to column 1

- A closing **}** brace typed in an otherwise empty line is moved to the same column as the immediately preceding opening brace **{**, providing that this is the only character in its line

- When the caret is positioned on any of the brace characters **(**, **)**, **[**, **]**, **{** and **}**, typing the key sequence **ESC CTRL-F** will move it to the corresponding opening or closing brace.

Running other programs

PFE allows you to run DOS commands, or start DOS command shell sessions freely. If you use a DOS mode compiler, for example, you can arrange to capture the output into a PFE window and browse it to correct your sources.

You can also start a DOS program or Windows application running as an independent task.

Starting a DOS session

Selecting the **DOS Prompt** item from the **Execute** menu, or typing the key sequence **CTRL-X !,** will start up a DOS session, in which you can issue any DOS commands you wish. Normally PFE will start up the DOS command interpreter defined in the **COMSPEC** environment variable; if this is not defined, it will start **COMMAND.COM**.

You can also use the tool bar to start a DOS session.

The starting directory for the DOS session will be PFE's own current directory. This can be changed through the standard file opening dialogs run by items in the **File** menu.

You can start as many DOS sessions as you wish from the **Execute** menu, and you can continue to use PFE while a session is running.

Running DOS commands

Programmer's File Editor allows you to issue a DOS command, such as a compiler driver, and capture the output into a new edit window for inspection.

To do this, select the **DOS Command** item from the **Execute** menu, type the key sequence **CTRL-X @**, or use the tool bar. This starts a dialog that lets you specify various things about the command you want to run

The dialog allows you to set:

- The command line to be executed. This can be any legal DOS command, including built-in commands such as **DIR.** If you are using the **4DOS** command processor, you can also use aliases and command sequences separated by the current delimiter character.

- The working directory to be used. PFE will make this the current directory for the command process before the command is issued (PFE's own current directory setting is not affected)

- Whether PFE is to minimize itself into an icon once the command has been issued

- Whether PFE is to sound a beep once the DOS command has completed.

The values you set in the dialog box are remembered throughout the editing session, enabling you quickly to re-issue the same command with the minimum of effort.

Once you click the **OK** button in the dialog, PFE will execute the DOS command in the specified working directory; when the command completes the captured output will appear in an edit window with the caption **Command Output**. This window will be created as a read-only window with no associated file, but you can change this, or use the **Save As** option from the **File** menu, if you wish.

While the DOS command is running, you will not be able to perform any editing. You will need to interrupt the execution of the command with **CTRL-C** or **CTRL-BREAK** to return control to PFE. The command's DOS window will show the command that is being run, but you will not see the output it produces until the command terminates.

Should the DOS command ask for keyboard input from the *stdin* channel, it will receive an "end of file" response from DOS and will probably terminate.

In order to run a DOS command, PFE requires you to have its helper module **pfedos.exe**, and the associated **pfedos.pif** file, in a directory on the path. The helper module is responsible for actually running the DOS command and redirecting its output. The PIF file as supplied runs the command in full-screen and exclusive modes; if you are working in 386 enhanced mode, you may wish to use the PIF editor to modify this.

(Note that only output to the *stdout* and *stderr* channels are captured in the PFE edit window. If the command that you run writes directly to the screen hardware, its output will not be captured).

Starting an independent task

PFE allows you to start any other Windows application, or a DOS application, with the **Launch Application** item on the **Execute** menu, by pressing **F11**, or by using the tool bar. This gives you a convenient way of launching the application that you editing and compiling with PFE. The menu item starts a dialog that lets you specify various things about the application you want to run

The dialog allows you to specify

- The command line to be executed.
- The working directory to be used. PFE will make this the current directory for the command process before the command is issued (PFE's own current directory setting is not affected)
- Whether PFE is to minimize itself into an icon once the command has been issued

The values you set in the dialog box are remembered throughout the editing session, enabling you quickly to start the same application again with the minimum of effort.

PFE starts the application running independently; it will not capture any output.

You can launch as many independent applications as you wish.

Working with development tools

PFE allows you easily to launch development tools by name from a list that you can configure. The applications need not be related to program development - you can have an entry to start a word processor or a spreadsheet; in fact any Windows or DOS application can be defined.

When you add a development tool to the list you assign it a *tool name*, which can be any string up to 32 characters long. You associate with this name the command string you want to execute and the initial directory; then, when you want to start the tool, you simply pick the name from the list.

Adding, changing and deleting tool entries

Changing the list of development tools that you can launch is done via a dialog.

To add an entry for a new development tool, you should do this:

1. Select the **Configure Development Tools** option from the **Execute** menu, or press **Shift-F12**. This starts the tool configuration dialog.

2. Either type in a new tool name in the **Tool** edit control, or select an existing tool name from the list.

3. Type in the command line you want executed when you launch the tool. If you don't give an absolute pathname, the tool's executable module must be either in the working directory you specify, or in a directory on the path. You can include command arguments after the command itself.

   The program you run can be either a Windows application or a DOS application.

   Clicking the **Browse** button will enable you to browse all the executable files on your disks to find the one you want.

4. If you want PFE to make itself an icon when the tool is launched, check the **Minimize Editor** box.

5. Click the **Add** button to add a new entry, or the **Change** button to alter an existing entry.

To delete an entry, you need to do this:

1. Select the **Configure Development Tools** option from the **Execute** menu, or press **Shift-F12** This starts the tool configuration dialog.

2. Select the existing tool name from the list.

3. Click the **Delete** button

Any changes you make affect only the copy of the tools list the PFE maintains in memory. In order to save the amended list permanently, you need to click the **Save** button.

If you've made changes to the list and haven't saved them, PFE will warn you when you click the **Save** button. You can if you like choose to carry on; the changes you've made will be usable in your current session, but will be lost when you exit PFE. If you later decide to make them permanent, just reselect the dialog as in step 1 above and click **Save**.

At any time, clicking the **Reset** button will return the list that PFE is using in memory to the state the last time you saved it.

Launching a development tool

To launch a tool that you've defined in the tools list, select the **Launch Development Tool** item in the **Execute** menu, or press **F11**.

This starts a dialog that lets you select one of the tools you've configured:

When you select one of the tool names from the list, PFE will show you the command line, the working directory name and the options that you chose the last time you launched the tool. If you haven't used this tool before, the working directory field will be empty, and the command line and options will be as you defined them in the table.

Set the working directory and options as you want them. If you want, you can alter the command line, either to run a different executable program, or to add or change the command line arguments. Clicking the **Browse** button lets you browse all the executable programs on your disks.

When you've set all the fields as you want them, click the **OK** button to launch the application.

PFE will remember the settings you used, and the next time you launch the application, the fields will show you those values. You can change them as you wish; clicking the **Reset Command Line** button will reset the command line field to the value defined in the tool table.

Running Control Panel and File Manager

The **Control Panel** and **File Manager** items on the **Execute** menu start the standard Windows control panel and file manager applications.

Using templates

Templates are pre-built sections of text that you can include into your files, saving you from repeatedly typing the same things.

For example, you might like to document your C procedures by preceding each one with an explanatory section like this:

```
/************************************************

   start_editor
   ------------

   This procedure starts the system in edit
   mode

************************************************/
```

Rather than type the lines of asterisks every time, you could set up a *template* containing the text:

```
/************************************************

    -------------

************************************************/
```

and insert it automatically in a simple operation. You can also place *template marks* within templates to show where you have to type in extra items (such as the name of the procedure in this example). PFE lets you move between these template marks with a single keypress, so that editing is very simple and convenient.

Templates are held in *template files*, which you can regard as a sort of library. You can have as many template files ready for use, or *attached* as you want, and you can insert templates from any of them at will. Template files are not ordinary text files and can't be edited directly, but the **Templates** menu provides you with options for creating template files, attaching and detaching them, creating, editing and deleting templates.

You can attach template files manually at any time as you need them. However, you'll probably find that you create a set of templates that you use all the time: you can arrange for PFE to attach these automatically when it starts up. You can also have a set of templates that are specific to a particular project, and this also can be attached automatically.

Creating a template file

Before you can store a template you must have a *template file* to place it in. Selecting the **Create File** item in the **Template** menu starts a standard dialog that lets you specify the name of the file you want to create. PFE will create the file you specified and write a small amount of binary control information into it.

You should use the file type **.tpl** when you create a template file. You should not try to template files directly, as they do not contain normal text.

When you create a template file it is automatically attached and ready to use.

Attaching and detaching template files

Before you can access any of the templates contained in a template file, you must first *attach* it. PFE loads the file into memory and constructs an index to the templates it contains, after which you can manipulate it. The **Attach File** item on the **Templates** menu starts a dialog that allows you to select the file you want to attach.

Having a template file attached uses some memory: PFE uses global memory to hold the data within the template file, and some control blocks. You can recover the memory used by an unwanted template file by *detaching* it: the **Detach File** item on the **Templates** menu starts a dialog that enables you to detach template files.

If you like, you can have PFE attach some template files automatically when it starts up. You can specify the files in two ways:

- As it starts, PFE scans the current directory for a file called **auto.tpl** and attaches it if it is a valid template file

- If you have specified a template directory in your **PFE.INI** file, with a line of the form

  directory=path

  in the **[templates]** section, PFE will look for a directory with the specified pathname, and will attach *all* the files it contains with a file type of **.tpl**

If you use the **Attach File** menu item, you can attach a template file in *read only* mode. You will be able to insert templates contained in this file into files that you are editing, but you will not be able to add new templates to it or edit existing ones within it.

Creating a template

In order to create a template, you should select the **New** item on the **Templates** menu. This opens up an edit window that is exactly the same as any other; the sole difference is that the window caption will show that you're editing a template.

You can edit inside the window exactly as in any other. You have an additional facility, though: you can use the **Insert Mark** item on the **Templates** menu (or type **SHIFT-F4**) to insert a *template mark*. This is a character sequence (actually the characters **<???>**) that are automatically searched for by the **Find Mark** item on the template menu (or by pressing the key **F4**). You can use template marks to hold the position for things to be filled in once you've inserted a template into your text file.

In order to use the template once you've finished editing it, you must store it in a template file. You can create a template at any time, but to store it you must have a template file attached. The **Store As** item in the **Template** menu starts a dialog that shows you all the template files that you've attached. You should select the file you want to store the template in, then type the template's name. This can be any sequence of up to 16 characters, used to identify this particular template.

PFE will write the template information into its memory copy of the template file, and you can then insert the template into files at will. However, the template will *not* be saved to disk until you use the **Save File** option of the **Templates** menu.

Inserting a template

At any time when you're editing, you can insert a template from any attached template file at the position of the caret.

Selecting the **Insert Template** item on the **Templates** menu, pressing the **F9** key or using the tool bar, shows you a dialog that gives you a list of all the attached template files. When you select one of the files you'll see a list of all the templates that it contains; you can then select the particular template that you want to insert.

PFE will read the template data from the template file and insert it into the current window at the position of the caret. Unlike the case when you're inserting another file or pasting from the clipboard, the caret remains at the *start* of the inserted data. This allows you to use the **F4** key (or the **Find Mark** item on the **Templates** menu) to move automatically to any template marks that you've put in the template to show where extra text needs to be inserted.

Editing a template

In order to change a template that you've stored in a template file you need first to attach the file, then use the **Edit** item on the **Templates** menu, or press **Shift-F9**. Since template files do not contain plain text, you shouldn't attempt the edit them directly.

The menu item shows you a dialog that lists all the attached template files. Selecting one of the files shows you a list of all the templates it contains, and then you can select the template you want to edit.

When you've selected a template, PFE will place it in an edit window for you. The window is exactly the same as one for editing a normal text file, but the caption will show you that it's actually a template you're editing. You can edit the template in any way you like.

Once you've finished your edit, you can update the template file by selecting the **Store** item on the **Templates** menu. This will delete the old version of the template from the template file and replace it with the new one. If you want to save the contents of the edit window as a template with a different name, or into a different template file, use the **Store As** item on the **Templates** menu. The template *file* is *not* updated on disk when you finish editing a template. To preserve your changes you need to explicitly save the file using the **Save File** option in the **Templates** menu.

Because a template is not regarded as an ordinary file, you can't use the **File** menu's **Save** and **Save As** options. If you need to save the template into an ordinary file, use the **Copy** item on the **Edit** menu to copy the data to the clipboard and paste it into a new window.

When you're editing a template, you can use the **Insert Mark** item on the **Templates** menu (or type **SHIFT-F4**) to insert a *template mark*. This is a character sequence (actually the characters **<???>**) that are automatically searched for by the **Find Mark** item on the template menu (or by pressing the key **F4**). You can use template marks to hold the position for things to be filled in once you've inserted a template into your text file.

Deleting a template

You can delete unwanted templates from a template file by using the **Delete** item in the **Templates** menu. You must have already attached the template file concerned.

The menu item starts a dialog that allows you to specify one or more templates in a template file and delete them. The template *file* is *not* updated on disk when you delete a template: to preserve your changes you need to explicitly save the file using the **Save File** option in the **Templates** menu.

Saving a changed template file

When you edit a template, create a new one, or delete one, PFE makes the changes only to the copy of the template data that it keeps in memory. The template file on disk is not altered in any way.

To make your changes permanent, you need to use the **Save File** item on the **Templates** menu. This starts a dialog that shows you the names of all the template files that have been altered, allowing you to select those you want to write back to your disk.

Moving And selecting

PFE allows you to move around a file that you're editing as you would in any other Windows application. Click an item from the list below for more details:

Moving using the mouse

PFE handles a mouse just as any other standard Windows application does. You can position the caret at any point in a file by clicking on that point; and you can scroll up or down, or side to side, by clicking the scroll bars.

If when you click the left button the mouse pointer is past the end of a line, or past the end of the file, or in the space occupied by the expansion of a tab character, PFE will set the caret to the rightmost character possible.

When you use the horizontal scroll bar to scroll the screnn, or scroll it vertically using the **up** and **down** arrows on the vertical scroll bar, the position of the caret *does not change* in the file. This means that, depending on how far you scroll, the caret may no longer be visible in the window.

When this happens, typing any character will always cause PFE to reposition the window so that the caret is roughly centred. You can also force PFE to reposition the window to show you the caret by typing **Shift-F5**.

Moving the thumbtack in the vertical scrollbar, or clicking in the bar above or below it, will always move the caret to the top left of the window.

Moving using the keyboard

As with other standard Windows applications, PFE lets you move around a file using the cursor keys and with the scroll bars.

The **cursor keys** will move the caret up and down in the current window. Normally the caret moves one character or line at a time; holding the **CONTROL** key down at the same time as the left or right cursor key will move the caret by one word at a time.

The **PgUp** and **PgDn** keys will scroll the window by one window's worth of lines, and will postion the caret at the top left corner of the window. Holding the **CONTROL** key down at the same time will move you to the start or end of the file respectively.

The **Home** and **End** keys will move the caret to the start or end of the current line respectively.

Moving to a specific line

You have several ways of moving to a specific line in a file. You can use the normal movement keys and watch the file position in the status bar at the bottom of the main window, or you can turn on file line numbers and use those to guide you.

Alternatively, you can use the **Goto Line** item on the **Edit** menu, or press **F5**. This starts a dialog that lets you enter the line number directly, and will adjust the window so that the caret is on the first character of the specified line.

Selecting using the mouse

If you want to **select** some text with the mouse, simply hold down the left button and drag the cursor. PFE will scroll the window in the appropriate direction when you move outside the window area. All the text that you select will be shown on your display as white characters on a black background.

To select a single word, you can place the mouse pointer anywhere within it and double-click the left mouse button.

Remember that, although you can have an area of selected text in each of the *files* you have open, only *one* window from any particular file can have a selection.

Selecting using the keyboard

As with other Windows applications, PFE allows you to select text using the keyboard.

Holding down the **SHIFT** key while using the **cursor keys** will move the caret up and down in the current window, selecting all the text that you pass over. Normally the caret moves one character or line at a time; holding the **CONTROL** key down at the same time as the left or right cursor key will move the caret by one word at a time.

Holding down the **SHIFT** key while pressing the **PgUp** or **PgDn** keys will scroll the window by one window's worth of lines, selecting all the lines that are passed over. Holding the **CONTROL** key down at the same time will move you to the start or end of the file respectively.

Holding down the **SHIFT** key while pressing the **Home** and **End** keys will move the caret to the start or end of the current line respectively, selecting all the text passed over.

To select a complete word, move the caret to anywhere within it and select the **Text Select Word** item in the **Edit** menu.

Cancelling a selection

To cancel a selected area of text and return it to unselected state, you should either click the left mouse button once anywhere in the window concerned; or press the **5** key on the numeric keyboard with Num Lock off. You can also click the **Cancel Selection** item in the **Edit** menu.

Cutting, pasting and copying

PFE uses the standard Windows *clipboard* for cutting, copying and pasting text. You can move text around in one edit window; move it from one window to another; or transfer text to and from other applications.

You can perform all these operations from items in the **Edit** menu or from keystrokes; you can also use the tool bar.

Cutting text to the clipboard

When you have some text selected in an edit window you can *cut* it to the clipboard by using the **Cut** item in the **Edit** menu; by pressing **Shift-DEL**; or by using the tool bar.

PFE will *delete* the selected text from the edit window after making a copy of it in the Windows clipboard. Anything that was previously held in the clipboard will be lost.

Copying text to the clipboard

When you have some text selected in an edit window you can *copy* it to the clipboard by using the **Copy** item in the **Edit** menu; by pressing the **Ctrl-C** key; or by using the tool bar.

Pasting text trom the clipboard

When the Windows clipboard contains some ASCII text (either put there from PFE or some other application) you can *paste* it into the file you're editing at the position of the caret by using the **Paste** item in the **Edit** menu; by pressing **Shift-Ins**; or by using the tool bar.

PFE will *delete* the selected text from the edit window after making a copy of it in the Windows clipboard. Anything that was previously held in the clipboard will be lost.

Searching and replacing

PFE allows you to search for text strings in the current file, and also to replace occurrences of one string with another.

To perform a search operation, select the **Search** operation in the **Edit** menu, press the **F2** key, or use the tool bar. PFE will start a dialog that lets you specify the text that you're looking for, and various other details.

> The **String** edit area is where you type the string to look for. You can type any characters here; to include a newline character in the string type "**^n**"; to include a tab character type "**^t**", and to include a "^" character type "**^^**"
>
> The **Match Case** box lets you specify whether case is significant or not in the search.
>
> The **Up** and **Down** directtion buttons specify whether the search is to proceed *up* from the caret to the start of the file, or *down* from the caret to the end of the file.

Once you've filled in the boxes, pressing **OK** will perform the search. PFE will locate the next occurrence of the desired string, and will position the file so that it is visible. The whole of the string that matched what you specified will be highlighted.

To repeat a search operation *exactly*, simply select the **Repeat Search** item in the **Edit** menu, or press **Shift-F2**. PFE will use all the details that you set in the dialog box the last time you selected it.

Replacing text is done in a very similar way. Selecting the **Replace** item in the **Edit** menu, pressing **F3**, or using the tool bar will start a dialog that lets you specify the text that you're looking for and what to replace it with.

> The **Replace** edit control lets you specify the string that PFE is to replace. You can type in any characters here; to include a newline character in the string type "**^n**"; to include a tab character type "**^t**", and to include a "^" character type "**^^**"
>
> The **With** edit control specifies what each occurrence of the target string is to be replaced with. Again you can type any character, and represent newline and tab as described above. If you leave this edit control empty, each occurrence of the target string will be deleted and nothing put in its place.
>
> The **Match Case** button lets you specify whether PFE is to pay regard to case when searching for the target string.
>
> The **Query Before Change**button lets you sepcify whether or not all occurrences are to be replaced automatically. If it is *unchecked*, PFE replaces every matching string without asking you. If it is *checked*, PFE will reposition the file so that you can see each match, and give you the chance to make the replacement or ignore it.
>
> The **Up** and **Down** directtion buttons specify whether the search is to proceed *up* from the caret to the start of the file, or *down* from the caret to the end of the file.

To repeat a replace operation *exactly*, simply select the **Repeat Replace** item in the **Edit** menu, or press **Shift-F3**. PFE will use all the details that you set in the dialog box the last time you selected it.

Printing a file

You can print the file that is being shown in an edit window by selecting the **Print** item in the **File** menu, pressing the key sequence **CTRL-F P**, or using the tool bar. PFE will show you a dialog box that lets you choose what part of the file you want to print: you can select to print either all of it, or only the part that you've selected, or the part between two line numbers.

You can specify the last line of the file as the word **end** in the rightmost of the line range edit controls.

The dialog box shows you the name of the printer that PFE is currently using - if you haven't specified otherwise, this will be your system default printer.

Clicking the **Setup** button starts PFE's **Printer Setup** dialog.

This lets you,

- Change the printer you'll be using
- Change the font that is to be used
- Configure whether you want margins on the page edges
- Run the printer's own setup program

The file is printed using the same tab width that is being used in the window, and with line numbers if you have line numbering turned on. You can configure both these settings with the **Current Settings** item in the **Settings** menu.

You can quickly turn line numbering on or off by typing the key sequence **CTRL-X N**, or clicking the line number toggle button on the tool bar.

Setting up a printer

To configure the printer that PFE is to use, select the **Print Setup** item in the **File** menu, click the **Setup** button from the **Print File** dialog, or press the key sequence **ESC P**.

PFE starts a dialog that shows you various printer-related topics.

The list box at the top left shows you all the devices that are configured on your system, and you can choose whichever you want.

The four **margin** check boxes in the **Print Options** area let you select whether PFE is to leave half-inch margins at each edge of the page.

The **Setup** button runs the setup program specific to whichever printer you've selected. This lets you, for example, select landscape or portrait mode, or choose the print quality you want.

Clicking the **Fonts** button starts a further dialog that lets you choose what font is to be used.

All the settings that you make in this setup dialog are remembered by PFE. The items defined in the **Print Options** area are recorded on a per-printer basis, so you can have different settings for different printers,

Selecting a printer font

Clicking the **Fonts** button in PFE's **Printer Setup** dialog starts a sub-dialog that lets you select the font to be used with the selected printer device.

The printer that's being used is shown at the top of the dialog. The boxes below let you choose the font you want by name, and the typeface size to use. Because PFE is a text editor rather than a word processor, you're restricted to choosing fixed pitch fonts, where the characters are all the same width.

PFE gives you a list of suggested typeface sizes, and you can select whichever you want. If the font you've chosen is a *continuous scaling font*, such as a TrueType font or one supplied by Adobe Type Manager, you can type in any font size you like, as well as choosing those from the list.

In some cases, though, the printer may not be able to honour the size requested. In that case, PFE uses the closest available size.

The **type style** boxes at the bottom left of the dialog let you select the style of text used.

PFE records the values you set in this dialog with the selected printer: the next time you choose this device, it will use the same settings.

The tool bar

Besides controlling PFE from the keyboard, or by using the menu, you can also use the *tool bar* to make access to the most common operations simple.

The tool bar is a collection of coloured push buttons. By default it appears at the top of the screen below the menu, but you can place it at any of the main window's edges, or have it floating as a separate window.

From left to right, the buttons in the tool bar have the following functions:

File New	Creates a new, empty file edit window
File Open	Opens an existing file
File Save	Saves the current file to disk
Cut	Cuts the selection to the clipboard
Copy	Copies the selection to the clipboard
Paste	Pastes the clipboard contents into the current file
Search	Searches for a string
Replace	Replaces one string with another
Undo	This button currently has no function
Insert Template	Inserts a template into the current file
Toggle Recorder	Toggles the keyboard macro recorder on and off
Start DOS Shell	Starts a DOS command line session
Run DOS Command	Starts a DOS command, capturing the output in an edit window
Launch Application	Lets you launch any Windows or DOS application as an independent task
Toggle Numbers	Turns line numbering on or off in the current window
Print	Prints the current file

Not all of the tool bar buttons will be usable at all times: for example, the **Print** button can't work if you don't actually have a file open. The buttons that you *can* use will display icons in colour; those that aren't currently available will show gray shadow images.

Moving the tool bar

By default, the tool bar appears as a row of button at the top of the screen below the menu. You can choose to place it at any of the main window's edges, or you can have it appear as a small floating window that you can place anywhere at all in the edit window. You can also, of course, choose to turn it off completely.

One way of moving the tool bar is to use the menu. The **Toolbar** item in the **Settings** menu gives you a list of possible options. You can *hide* the tool bar or *show* it, or specify where you want it to be placed.

If you have a mouse, you can also *drag* the tool bar to where you want it. You need to do this to start:

- If the tool bar is at one of the window edges, place the mouse pointer within it but *not* over one of the buttons. Press the left mouse button and hold it down; the cursor changes to a four-pointed arrow.

- If the tool bar is a floating window, put the mouse pointer over the caption bar at the top, press the left button and hold it down.

Now *drag* the tool bar to where you want it to be placed and release the left mouse button. If the mouse pointer is close to one of the main window's edges, the tool bar will be placed on that edge; if the mouse pointer is nearer the middle of the main window, the tool bar will turn into a floating window.

PFE will remember where you've placed the tool bar, and will put it there the next time you start the program.

Key mappings

PFE allows you to change the details of what keys you press to perform most of its operations. For instance, you could set the **CTRL-A** key so that pressing it brought up the **Attach Template File** dialog if you found that convenient.

The operations that PFE perfoms are called *functions*, and these have fairly descriptive names such as **FileOpen** and **TransposeCharacters**. Most of the functions are similar in name to the commands used to control PFE over a DDE link, and to those in the forthcoming script language.

The relationship between a key on the keyboard and the function that is performed by pressing it is known as a *key mapping*.

You can store your personalised key mappings so that they are automatically applied every time you start PFE. You can also save them in key mapping files, so that different people using your system could load their own mappings; or so that you could have different mappings for different purposes

What keys can be mapped

PFE is controlled from the keyboard using two different types of key sequences. You can't choose arbitrary keys, but the available collection is very large (some 400 keys), so you should be able to find combinations that suit your way of working.

Firstly, it uses *single-character* key presses such as **Ctrl-A** and **F7**, in a similar way to a standard Windows application. If you wished, you could set up PFE to operate entirely with such characters.

The second mode uses *two-characters sequences*, where you first press a *prefix key*, and then press a second key. This form provides a very large number of available keys for you to choose from, and is similar in approach to the **MicroEMACS** DOS-mode editor. The prefix keys are **ESC**, **Ctrl-B**, **Ctrl-F** and **Ctrl-X**; and you can follow each of these with a wide range of other keys.

When you run the dialog that lets you change the key mappings, you'll see a list of all the keys and key sequences that you can choose from.

PFE does not allow you to select keys that have standard functions within Windows. Thus, you can't map any key that is pressed with **Alt**, since this is the standard way of selecting menu items from the keyboard. Nor can you map, for example, the **F10** key, since this is the standard way of accessing the system menu of an MDI child window.

Also, since the four *prefix keys* are always taken as introducing another character, you can't map any of these to to functions either.

Changing the key mappings

To change the mappings of keys to functions to be as you want, select the **Edit Key Mappings** item in the **Settings** menu. This starts a dialog box from which you can choose your requirements.

The list box at the top contains a list of all the functions to which you can map keys. When you select a command from the list, the **Now Mapped From** list shows you which, if any, of the keyboard keys invoke that command. You can map as many keys as wish to any command.

To map keys to the function, select those you want from the **Available Keys** list on the right, then press the **Map** button. The key names will move to the left hand list. Similarly, to unmap keys, select them in the left-hand box and press the **Unmap** button.

If the key you want to map to the selected function is already mapped to a different one, it will not appear in the **Available Keys** list. You will need to select the key name in the **Keys In Use** box at lower right and press the **Free Key** button to unmap the key and return its name to the **Available Keys** list, then select it from there and map it. If you wish, you can use the **Free All** button to unmap *all* the currently mapped keys.

Many commands have equivalent menu entries, and you can choose to have the name of *one* of the keys mapped to it appear on the menu entry. Select the key name in the **Mapped Keys** list on the left and set the check mark in the **Show On Menu** box below it.

The **Save** button saves your key mappings in the default key mapping file **pfe.key**, which is held in your Windows directory. PFE will automatically load these mappings the next time you start it. If you wish, you can save the mappings to a different file by pressing the **Save As** button; you can then use the **Load** button to load any key mapping file that you wish to use.

Loading different key mappings

PFE allows you to save as many different sets of key mappings as you wish, so that different people using your system can each have their own favourite settings; or you can have different sets of your own for different purposes.

Key mappings are stored in files with type "**.key**". When it starts up, PFE will automatically load the key mappings in a file called **pfe.key** in your Windows directory if this file exists.

To load a set of key mappings from another file, select the **Edit Key Mappings** item in the **Settings** menu. This starts up the dialog that lets you configure key mappings: clicking the **Load** button will cause PFE to ask you the name of the file of mappings that you want to load.

The default key mappings

As supplied, PFE contains a set of key mappings that enable you to perform the most common operations quickly. You may use these as they stand, or may modify hem to suit your preferences.

The function keys have the following default mappings:

Key	Used Alone	With SHIFT	With CTRL
F1	Help		
F2	Search	RepeatSearch	
F3	Replace	RepeatReplace	
F4	FindTemplateMarker	InsertTemplateMarker	
F5	GotoLine	CaretCentre	
F6			
F7	RecorderPlay	RecorderStart	RecorderStop
F8			
F9	TemplateInsert		
F11	RunDOSCommand	LaunchApplication	RunDOSPrompt
F12	LaunchTool	ConfigureTools	

Other keys have these default mappings:

Key	Function	Key	Function
Ctrl-F A	FileSaveAs	**Ctrl-F C**	FileClose
Ctrl-F I	FileInsert	**Ctrl-F N**	FileNew
Ctrl-F O	FileOpen	**Ctrl-F P**	FilePrint
Ctrl-F S	FileSave	**Ctrl-F V**	FileView
Ctrl-F W	FileWrite	**Ctrl-F X**	FileCloseAll
Ctrl-X A	WindowArrangeIcons	**Ctrl-X C**	WindowCascade
Ctrl-X I	WindowIconizeAll	**Ctrl-X N**	WindowNext
Ctrl-X W	WindowWiden	**Ctrl-X 0**	WindowClose
Ctrl-X 2	WindowDuplicate		
ESC W	SelectWord	**ESC ~**	MarkFileUnchanged
Ctrl-C	Copy		
Ctrl-K	DeleteToEndOfLine	**Shift-Ctrl-K**	DeleteLine
Ctrl-N	ToggleLineNumbers		
Shift-DEL	Cut	**Shift-INS**	Paste

Dictionary Of functions

PFE allows you to map keyboard keys or key sequences to the following functions:

CMatchBrace	When used with a window that has the C language type associated with it, moves the caret to a matching brace character

ChangeCurrentSettings	Starts a dialog to change the current window and file modes
ChangeDefaultSettings	Starts a dialog to change the window modes for specific file types
CancelSelection	Removes any highlight from selected text in the current window
ConfigureTools	Starts a dialog to configure details of program development tools
Copy	Copies the selected text to the clipboard
Cut	Copied the selected text to the clipboard and deletes it
DeleteSelection	Deletes the selected text
DeleteCharBackwards	Deletes the character to the left of the caret
DeleteCharForwards	Deletes the character to the right of the caret
DeleteLine	Deletes the whole of the line that the caret is in
DeleteToEndOfLine	Deletes from the position of the caret to the end of the current line
EditKeyMappings	Starts a dialog to change the mapping of keys to functions
Exit	Terminates the PFE editing session
FileClose	Closes the file being edited in the current window
FileCloseAll	Closes all the files being edited
FileInsert	Inserts a file into the current one at the position of the caret
FileNew	Creates a new empty edit window for an unnamed file
FileOpen	Opens an existing file for editing
FilePrint	Prints the current file
FileSave	Saves the current file to disk, provided that it has been altered
FileSaveAs	Saves the current file to disk, allowing the output file name to be specified
FileView	Opens an existing file in *read-only* mode
FindTemplateMarker	Searches in a forward direction from the position of the caret for the next occurrence of a *template marker*
GotoLine	Starts a dialog that prompts for the line number to go to in the current file
Help	Starts the Windows help engine to give help on PFE

HelpOnHelp	Starts the Windows help engine to give help about itself
HelpAbout	Shows PFE's *About* box
HelpOnSDK	Starts the Windows help engine, using the help file configured as the *SDK Help File*
InsertTemplateMarker	Inserts a *template marker* into a template
LaunchApplication	Starts a Windows or DOS application to run independently of PFE
LaunchTool	Starts a program development tool
LowerCase	Converts all the letters in the current selection to lower case
MarkFileUnchanged	Removes PFE's indication that the current file has been altered
NewLine	Inserts a new line character at the position of the caret
Paste	Pastes the contents of the clipboard into the current file at the position of the caret
PrintSetup	Runs a dialog that allows printer information to be configured
RecorderPlay	Replays the last recorded keyboard macro
RecorderStart	Starts recording keystrokes and menu selections
RecorderStop	Stops recording keystrokes and menu selections
RepeatReplace	Repeats the last replace operation
RepeatSearch	Repeats the last search operation
Replace	Replaces occurrences of one string with another string
RunControlPanel	Starts the Windows Control Panel
RunDOSCommand	Runs a DOS command, capturing output in an edit window
RunDOSPrompt	Runs a DOS command-line shell
RunFileManager	Runs the Windows File Manager
Search	Searches for a string in the current file
SelectWord	Selects the whole word that the caret is currently within
SplitLine	Splits the current line at the position of the caret, leaving the caret unmoved
TemplateFileAttach	Attaches a template file so the templates within it can be used
TemplateFileCreate	Creates a new, empty template file

TemplateFileDetach	Detaches an attached template file
TemplateFileSave	Saves an altered template file to disk
TemplateDelete	Deletes a template from a template file
TemplateEdit	Edits a template from within a template file
TemplateInsert	Inserts a template from a template file into the current file at the position of the caret
TemplateNew	Creates a new empty window suitable for editing a template
TemplateStore	Stores a template in a template file
TemplateStoreAs	Stores a template in a template file, allowing its name to be changed
ToggleInsertMode	Switches between *insert* and *overwrite* modes
ToggleLineNumbers	Turns line numbering on or off in the current window
ToggleStatusBar	Makes the status bar visible or invisible
ToolBarBottom	Positions the tool bar at the bottom of the screen
ToolBarFloat	Makes the tool bar into a detached floating window
ToolBarHide	Makes the tool bar invisible
ToolBarLeft	Positions the tool bar at the left of the screen
ToolBarRight	Positions the tool bar at the right of the screen
ToolBarShow	Makes an invisible tool bar visible at its previous location
ToolBarTop	Positions the tool bar at the top of the screen
TransposeCharacters	Exchanges the character under the caret with the one to its left
UpperCase	Changes all the text that is currently selected to be upper case
WindowArrangeIcons	Arranges all iconized windows neatly
WindowCascade	Arrannges all non-iconic windows in a cascade
WindowClose	Closes the current window
WindowDuplicate	Makes an exact duplicate of the current window
WindowIconizeAll	Makes all existing windows iconic
WindowMaximize	Maximizes the current window
WindowNext	Switches control to the next non-iconic window

WindowRestore	Restores the current window from iconized or maximized state
WindowTile	Arranges all non-iconic windows in a tile pattern

Controlling PFE over a DDE link

PFE has the ability to act as a **DDE Server**, allowing you to control its actions from some other program. This program will need to have the ability to act as a **DDE Client**; many standard applications already support this.

If you are using an existing Windows application as a DDE client you will need to consult its documentation for how to perform the necessary operations. You can easily write your own client application in Visual Basic (the DDETester application supplied with PFE demonstrates many of the techniques you will need); or you can write at API level using the Microsoft **DDEML** support library described in the SDK documentation and third party books.

In this beta release of PFE, the DDE intefarfaces are still under development, and there may be considerable changes from what is described here in the final released version.

DDE commands

DDE commands are sent to PFE using the standard DDE methods. Your application should open a DDE link to PFE specifying the server name "**PFE**", and a topic name of either "**Editor**" or "**System**". From Visual Basic for example, you would perform a *LinkExecute* operation.The DDE item name is not used.

Once the link is established, you can send one or more command strings to be executed. The command string should have the general form

[CommandName(argument,...)]

where the arguments you supply withing the parentheses will vary depending on the command. The command name can be written in any mixture of upper- and lower-case letters. A maximum of 128 bytes can be sent in any transaction.

PFE will return a response of **DDE_FACK** immediately if the command syntax is ostensibly correct, or **DDE_FNOTPROCESSED** if there is some obvious error. The command itself is executed asynchronously to the DDE client, which will regain control immediately after sending it. If the last DDE command resulted in a lengthy action - such as loading a large file, or asking the user for input in a dialog box - the server may not be able to accept a new command immediately. In this case it will return a reply **DDE_FBUSY** to the client, which should wait for a suitable interval before attempting the command again.

To determine whether any DDE link command worked or not you need to request the data for the item "**Result**" from the "**Editor**" topic.

The DDE commands currently supported are these:

ActivateWindow(*id*) Activates the window with the specified id value. The id is a numeric string, returned by querying the data item **WindowID** when the desired window is current.

CaretDown(*select*) Moves the caret down by one line. If *select* is 1 the characters it moves over are selected and shown in reverse video; if it is 0 they are not and any outstanding selection is cancelled.

CaretEnd(*select*) Moves the caret to the end of the current line, possibly extending a selection.

CaretHome(*select*) Moves the caret to the start of the current line, possibly extending a selection.

CaretLeft(*select*) Moves the caret left by one character, possibly extending a selection.

CaretRight(*select*) Moves the caret right by one character, possibly extending a selection.

CaretUp(*select*) Moves the caret up by one line, possibly extending a selection.

GotoEndOfFile(*select*) Moves the caret to the end of the file, possibly extending a selection.

GotoStartOfFile(*select*) Moves the caret to the start of the file, possibly extending a selection.

Insert("*string*") Inserts the quoted string into the current file at the position of the caret. Within the string, the characters "**^n**" represent a line break and "**^t**" a tab character. To insert a "^" character, specify it as "^^", to insert a quote character, specify it as two consecutive quote characters.

FileClose() Closes the current file. If the file has been altered, the user will see a dialog box asking him if he wishes to save the data.

FileInsert("*filename*") Inserts the specified file into the current file at the position of the caret. The filename must be an absolute pathname starting with a drive letter; if it is omitted, PFE will show a dialog box to ask the user to supply it.

FileNew() Creates a new window for an un-named file.

FileOpen("*filename*") Opens the specified file for editing. The filename must be an absolute pathname starting with a drive letter; if it is omitted, PFE will show a dialog box to ask the user to supply it.

FileSave() Writes the current file to disk, providing it has been changed since it was last saved. If the file is currently un-named, the user will see a dialog asking him for the file name it is to be saved under.

FileSaveAs("*filename*") Writes the current file to disk, changing its name in the process. The filename must be an absolute pathname starting with a drive letter; if it is omitted, PFE will show a dialog box to ask the user to supply it.

FileView("*filename*") Opens the specified file in read-only mode. The filename must be an absolute pathname starting with a drive letter; if it is omitted, PFE will show a dialog box to ask the user to supply it.

FileWrite("*filename*") Writes the current file to disk with the specified filename, whether or not it has been changed. The name by which the file is known to PFE, as shown in the window caption, is not altered. The filename must be an absolute pathname starting with a drive letter; if it is omitted, PFE will show a dialog box to ask the user to supply it.

Paste() Causes the contents of the Windows clipboard (provided that it is in **CF_TEXT** format) to be pasted into the current file at the position of the caret. You can also use a DDE Poke operation to paste data.

Requesting data from the DDE server

The PFE DDE server allows you to request items of data from your application. For example, you can ask PFE whether the DDE server is busy, or what the result of the last DDE command was; you can also find out about the state of files that are being edited and so on.

To request data, you need to open a DDE link to service "**PFE**" with an appropriate topic. The link item is the name of the information that you require. From Visual Basic, for example, you would perform a *LinkRequest* operation.

The supported items that you can request under the topic "**Editor**" are:

FileChanged Tells you whether the current file has been altered since it was last saved. The reply will be the string "**Yes**" if the file has changed; and "**No**" if either the file has not changed, or there is no current file.

Result Gives you the result of the last command executed over the DDE link. The reply will be one of the strings "**OK**" or "**Error**" if a result is available, and "**Busy**" if the server is currently processing a DDE link command. The results of commands executed from the keyboard, menu or toolbar cannot be read by this method.

Status Gives you the current status of the DDE server. The reply will be one of the strings "**Ready**" and "**Busy**".

WindowID Gives you the unique id number of the current window as a numeric string. This id value can be used to identify the window for the whole of its life; id values are never re-used in any given PFE session. If no window exists, the string "**Error**" is returned.

The supported items under the topic "**System**" are

Formats A list of the clipboard formats supported by PFE's DDE server. This will be the string "**TEXT**"

SysItems A list of items supported by PFE under the "**System**" topic, separated by tab characters

Topics A list of the topics supported by PFE, separated by tab characters

Status The current status of the DDE server. The reply will be one of the strings "**Ready**"and "**Busy**".

Pasting data over a DDE link

PFE gives you several ways of pasting data into the current file over a DDE link.

For small amounts of data, you can send the DDE command **Insert()**, giving the string to be pasted as the argument. This will often be the most convenient technique to use.

For larger amounts of data, you have a choice of two methods:

Using the clipboard

With this method, you should place the data to be pasted into the Windows clipboard in **CF_TEXT** format (i.e. with end of line represented by CR-LF bytes pairs and a null byte marking the end of data) and issue the DDE command **Paste()**.

Using DDE Poke

With this method, you initiate a conversation with the PFE DDE server using service "**PFE**", topic "**Editor**" and item "**Paste**". Your application should place the data to be pasted into a shareable global memory segment in **CF_TEXT** clipboard format, then perform a DDE **Poke** operation passing the handle to the memory segment.

PFE will reply **DDE_FACK** if the operation succeeded, **DDE_FNOTPROCESSED** if it failed, and **DDE_FBUSY** if the server was busy and could not accept the command.

A DDE Poke operation is always performed synchronously, so your DDE client will need to set a suitable timeout value if a large amount of data is being pasted.

Notes on this beta release

This is a beta release of Programmer's File Editor. Although is reliable in operation - the author uses it as his only programming editor, and does all the development work on PFE with it - and able to handle huge files (the largest tested so far has been a 12 megabyte file of over 300,000 lines), not all the planned facilities have yet been added to it, and some facilities may be provided in a different way in the ultimate release.

Facilities that are planned, but not yet implemented, are:

- The ability to undo edit changes
- Automatic location of errors from compiler output
- Automatic wrapping of text at a user-definable column
- Ability to store keyboard macros in files for re-use
- A command language to allow the use of scripts

Support for Windows 3.0 is not planned, but a 32-bit version for Windows/NT is.

The author would welcome feedback from users of the beta releases, and ideas for further facilities. He can be contacted by electronic mail as

alan@uk.ac.lancaster (JANET)
alan@lancaster.ac.uk (Internet)

The C source of PFE will be distributed with the first full release version.

Known Bugs in this Release

Known bugs

There are the following known problems in this beta release:

- When editing a file in more than one window simultaneously, some operations in one will cause the caret to be mis-positioned in others

- A double click in the system box of a maximized edit window does not close the window

- Tool bar button presses are not recorded in keyboard macros

There are also the following known deficiencies:

- File save is not as fast as planned; currently it does make best use of the internal information available

This help file was created for Programmer's File Editor pre-release beta version 0.03.020

Acknowldgements

The author wishes to gratefully acknowledge the help of those who have contributed ideas, code and testing time towards this product.

The beta testers were Marc Goldman, Dave Bleasdale, Gavin Silver, Andy Errington, Martin Kalugin and Dave Ingles, all of whom also contributed important development suggestions. The DDETester Visual Basic application was written by Andy Errington.

Icons and bitmaps were designed by Dave Bleasdale, who helped considerably with the key mappings.

Contributions of actual code for enhancements, bug reports, bug fixes, comments and suggestions are always welcome: see the development documentation supplied in a separate text file.