Lattice C 5.5

the C Compiler for your Atari ST/STE/TT Computer

Addendum

User Manual

Requires:

✓ Atari 520ST upwards
(1M+ memory required)

✓ Disk drive
(2 floppies or hard disk advised)

✓ Mouse

HiSoft
High Quality Software

Lattice C 5.5 for the Atari ST/STE/TT

By HiSoft and Lattice, Inc.

Program:
designed and programmed by HiSoft and Lattice, Inc.

Manual:
written by Alex Kiernan and David Nutkins.

HiSoft
High Quality Software

Published by HiSoft

The Old School, Greenfield, Bedford MK45 5DE UK

First Edition, March 1992 - ISBN 0 948517 56 5

Table of Contents

Lattice C 5.5 The Editor

The editor supplied with Lattice C is fully integrated with the system which means that you can develop programs in an intuitive and interactive manner, creating and editing your programs in the same environment as running and debugging your finished masterpiece.

Moreover, those of you with strong preferences for your own editor can dispense with the HiSoft editor and use your own favourite package along with the TTP version of Lattice C; although you will lose the benefits of interactive development.

The editor for Lattice C is a multi-window screen editor which allows you to enter and edit text and save and load from disk, as you would expect. It also lets you print some or all of your text, search and replace text patterns and use any of your computer's desk-accessories. It is GEM-based, which means it uses all the user-friendly features of GEM programs that you have become familiar with such as windows, menus and mice. However, if you're a die-hard used to the hostile world of computers before the advent of WIMPs, you'll be pleased to know you can do practically everything you'll want to do from the keyboard without having to touch a mouse.

The editor is 'RAM-based', which means that the file you are editing stays in memory for the whole time, so you don't have to wait while your disk grinds away loading different sections of the file as you edit. As the ST/TT range of computers have so much memory, the size limitations often found in older computer editors do not exist with Lattice C. As all editing operations, including things like searching, are RAM-based they act extremely quickly.

When you have typed in your programs it is not much use if you are unable to save them to disk, so the editor has a comprehensive range of save and load options, allowing you to save all or part of the text and to load other files into the middle of the current one, for example.

To get things to happen in the editor, there are various methods available to you. Features may be accessed in one or more of the following ways:

- Using a single key, such as a Function or cursor key;
- Clicking on a menu item, such as Save;
- Using a menu shortcut, by pressing the Alternate key (subsequently referred to as Alt) in conjunction with another, such as Alt-F for *Find*;
- Using the Control key (subsequently referred to as Ctrl) in conjunction with another, such as Ctrl-A for *cursor word left*;
- Clicking on the screen, such as in a scroll bar.

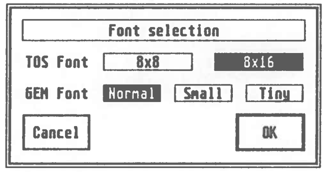
The menu shortcuts have been chosen to be, hopefully, easy to remember.

A word about pop-up menus and dialogs

The editor makes extensive use of dialog boxes and pop-up menus, so it is worth recalling how to use them, particularly for entering text. The editor's dialog boxes contain buttons, radio buttons, and editable text.

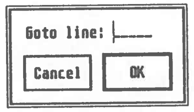
Exit buttons may be clicked on with the mouse and cause the dialog box to go away. Usually there is a default button, shown by having a wider border than the others. Pressing Return on the keyboard is equivalent to clicking on the default button. Where there are non-default buttons, the editor allows these to be selected from the keyboard using the sequence Alt-first letter of the button name; obviously where several buttons have the same first letter only one may be selected!

Radio buttons are groups of buttons of which only one may be selected at a time - clicking on one automatically de-selects all the others.



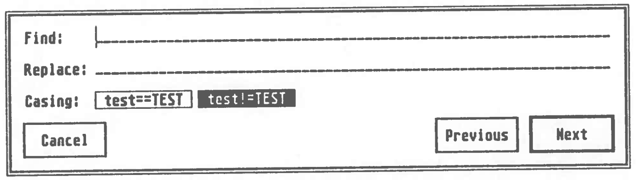
A dialog with buttons (OK, Cancel) and radio buttons (Normal, Small etc.)

Editable text is shown with a dotted line, and a vertical bar marks the cursor position.



Editable text

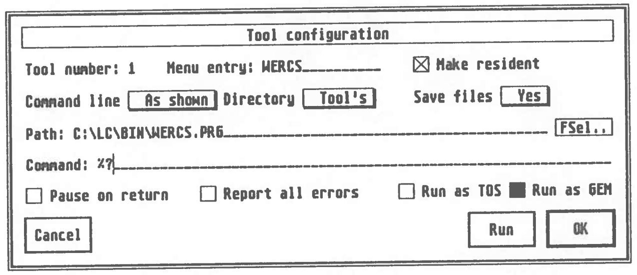
Characters may be typed in and corrected using the Backspace, Delete and cursor keys. You can clear the whole edit field by pressing the Esc key. If there is more than one editable text field in a dialog box, you can move between them using the Tab key or the ↓ and ↑ keys or by clicking near them with the mouse.



More than one editable text field

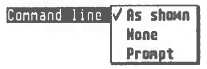
Some dialog boxes allow only a limited range of characters to be typed into them - for example the Goto... dialog box only allows numeric characters (digits) to be entered.

As well as the conventional GEM user interface facilities, the editor also uses some extensions. To illustrate these, consider the dialog box shown below:


Tool configuration
Tool number: 1 Menu entry: WERCS ☒ Make resident
Command line: As shown Directory: Tool's Save files: Yes
Path: C:\LC\BIN\WERCS.PRG FSel..
Command: %?
☐ Pause on return ☐ Report all errors ☐ Run as TOS ■ Run as GEM

The Tool Configuration dialog box

Some options are accessed via 'pop-up' menus similar to those used by Atari's new control panel. Thus if you move the mouse over the As shown selection (by Command line) and press down on the left mouse button, a menu like this will pop up:



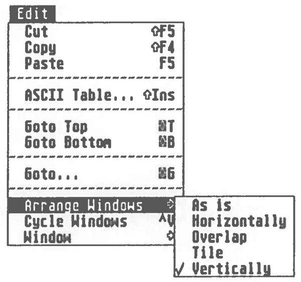
A pop-up menu

This indicates that the current setting for this option is As shown. The mouse will highlight the current selection that you are making and when you let go of the mouse this indicates that you have made your selection. If you let go outside the pop-up menu then this is taken as cancelling the selection.

The box beside Make resident has a cross in it, indicating that this option is selected; similarly Report all errors is *not* selected. Clicking in one of these boxes, or the associated text, will cause that option to be toggled on and off.

Run as TOS and Run as GEM are a pair of 'radio options'; the solid box indicates the currently selected item: clicking on Run as TOS will change both boxes.

Some of the menu items on the main 'drop-down' menus now have sub-menus; these are indicated by a ⋄ symbol. For example:

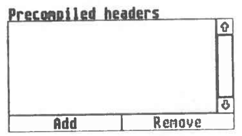


A sub menu

When you highlight a menu item (like Arrange Windows in the example above), the corresponding sub-menu will appear after a short delay. You can then move the mouse to the right to select the particular item that you want. To cancel the operation just let go of the mouse without selecting a sub-item or move to another item from the main menu.

If the editor doesn't have enough room to display the sub-menu to the right of the main menu, it will do so on the left; the items are selected in the same way.

The editor also uses a number of list boxes; these allow a number of selections to be entered (e.g. multiple `#include` directories, `#define` symbols etc.).



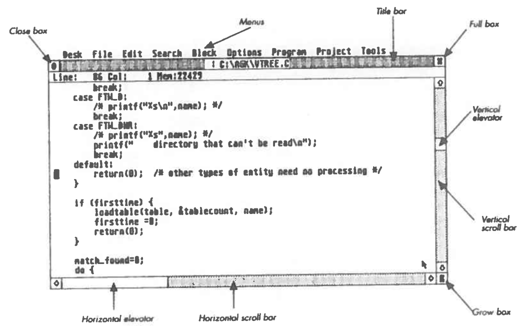
A list box

To add a new element to the list, click on the Add button, whilst an existing element may be removed by clicking on the item (which will become highlighted) and then clicking Remove. To edit an existing item, double-click on it.

The Editor's windows

Having loaded Lattice C, you will be presented with an empty window with a status line at the top and a flashing black block, which is the *cursor*, in the top left-hand corner.

The window used by the editor works like all other GEM windows, so you can move it around by using the *Title* bar on the top of it, you can change its size by dragging on the *Grow* box, and make it full size (and back again) by clicking on the *Full* box.



```
Line:   86 Col:    1 Mem:22429
        break;
    case FTW_D:
        /* printf("%s\n",name); */
        break;
    case FTW_DNR:
        /* printf("%s",name); */
        printf("    directory that can't be read\n");
        break;
    default:
        return(0);  /* other types of entity need no processing */
    }

    if (firsttime) {
        loadtable(table, &tablecount, name);
        firsttime =0;
        return(0);
    }

    match_found=0;
    do {
```

A GEM window

The status line contains information about the cursor position in the form of Line and Column offsets as well as the number of bytes of memory which are free to store your text. Initially this is displayed as 29980, as the default text size is 30000 bytes. You may change this default if you wish, together with various other options, by selecting Preferences, described later. The 'missing' 20 bytes are used by the editor for internal information. The rest of the status line area is used for error messages, which will usually be accompanied by a 'ping' noise to alert you. Any message that is printed will be removed when subsequently you press a key.

Switching Windows

The editor has support for up to seven windows, which can be selected by pressing Alt-1 to Alt-7 (on the top row of numbers, *not* on the numeric pad). The windows can be organised in a number of ways and you can select this using Arrange Windows on the Options menu. Try this out for yourself to get the idea of how the different arrangements work.

If you have a preferred window arrangement, you can get the editor to remember your preference by holding down Ctrl whilst selecting the layout. The layout will then become permanent and the editor will rearrange the windows as necessary to conform to your preference.

You can cycle through the open windows using the Cycle Windows command from the Edit menu (or use Ctrl-V), by clicking on the appropriate window with the mouse or by selecting the appropriate sub-item from the Window item on the Edit menu.

To close a window and thus free the memory used by it, click on its close box or use the Ctrl-W key combination.

To cut and paste between windows is just as simple as copying blocks in a single window, i.e. mark the block and then use the Cut command, switch windows (as described above) and then Paste. See below for more detail on cut and paste.

Entering text and moving the cursor

To enter text, simply type on the keyboard and at the end of each line press the Return key (or the Enter key on the numeric pad) to start the next line. You can correct your mistakes by pressing the Backspace key, which deletes the character to the left of the cursor, or the Delete key, which removes the character on the cursor.

Cursor keys

To move the cursor around the text to correct errors or enter new characters, you can use the cursor keys, labelled ← → ↑ and ↓ or the mouse; move the cursor to a specific position on the screen with the mouse pointer and click. If you position the cursor past the right-hand end of the line and type some text at that point the editor will automatically add the text to the real end of the line. If you type in long lines the window display will scroll sideways if required.

When you cursor up at the top of a window the display will either scroll down if there is a previous line, or print the message `Top of file` in the status line. Similarly if you cursor down off the bottom of the window the display will either scroll up if there is a following line, or print the message `End of file`.

You can move the cursor on a character basis by clicking on the arrow boxes at the end of the horizontal and vertical scroll bars.

To move immediately to the start of the current line, press Ctrl ←, and to move to the end of the current line press Ctrl →.

To move the cursor a word to the left, press Shift ← and to move a word to the right press Shift →. You cannot move past the end of a line with Shift →. A word is defined as anything surrounded by a space, a tab or a start or end of line. The keys Ctrl-A and Ctrl-F also move the cursor left and right on a word basis.

To move the cursor a page up, you can click on the upper grey part of the vertical scroll bar, or press Shift ↑. To move the cursor a page down, you can click on the lower grey part of the scroll bar, or press Shift ↓.

Tab key

The Tab key inserts a special character (ASCII code 9) into the buffer, which on the screen looks like a number of spaces, but is rather different. Pressing Tab aligns the cursor onto the next 'multiple of 4' column, so if you press it at the start of a line (column 1) the cursor moves to the next multiple of 4, +1, which is column 5. Tabs are very useful indeed for making items line up vertically and its main use in Lattice C is for such things as indenting structured program lines. When you delete a tab the line closes up as if a number of spaces had been removed. The advantage of tabs is that they take up only 1 byte of memory, but can show on screen as many more.

You can change the tab size before or after loading Lattice C; to change the default use the Preferences command described shortly.

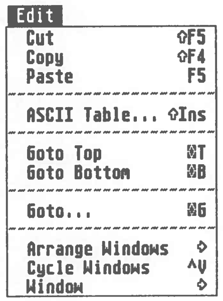
Backspace key

The Backspace key removes the character to the left of the cursor. If you backspace at the very beginning of a line it will remove the 'invisible' carriage return and join the line to the end of the previous line. Backspacing when the cursor is past the end of the line will delete the last character on the line, unless the line is empty in which case it will re-position the cursor on the left of the screen.

Delete key

The Delete key removes the character under the cursor and has no effect if the cursor is past the end of the current line.

The Edit menu



The commands on the top of the Edit menu may be used to perform the conventional Cut, Copy and Paste operations on marked blocks.

These are described under *Block commands*, below.

Go to top of file

To move to the top of the text, click on Goto Top from the Edit menu, or press Alt-T. The screen will be re-drawn if necessary starting from line 1.

Go to end of file

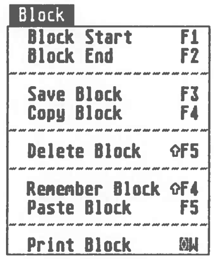
To move the cursor to the start of the very last line of the text, click on Goto Bottom, or press Alt-B.

Goto line

To move the cursor to a specific line in the text, click on Goto... from the Edit menu, or press Alt-G. A dialog box will appear, allowing you to enter the required line number. Press Return or click on the OK button to go to the line or click on Cancel to abort the operation. After clicking on OK the cursor will move to the specified line, re-displaying if necessary, or give the error End of file if the line doesn't exist.

Another fast way of moving around the file is by dragging the slider on the vertical scroll bar, which works in the usual GEM fashion.

Block Commands



```
Block
Block Start      F1
Block End        F2
Save Block       F3
Copy Block       F4
Delete Block    ⇧F5
Remember Block  ⇧F4
Paste Block      F5
Print Block      ⌘W
```

A *block* is a marked section of text which may be copied to another section, deleted, printed or saved onto disk. Blocks may be marked using the mouse, via menu items or with function keys.

A marked block is highlighted by showing the text in reverse. While you are editing a line that is within a block this highlighting will not be shown but will be re-displayed when you leave that line or choose a command.

Marking a block

The simplest way to mark a block is to click on the first character in the block and drag the mouse to the end of the block. The block will be highlighted by showing the text in reverse as you drag the mouse. When you move the mouse to the bottom of the window, the window will scroll. Conversely, moving the mouse to the top of the window, will cause the window to scroll in the opposite direction. You may start marking a block, by clicking at the end if you wish.

Double-clicking will cause the word 'under' the mouse to be marked as the block. If you double-click and then drag, text will be highlighted a word at a time. Clicking in the the left hand margin of the window causes dragging to occur a line at a time.

The start of a block may also be marked by moving the cursor to the required place and selecting Block Start or pressing key F1. The end of a block can be marked by moving the cursor and selecting Block End or pressing key F2. The start and end of a block do not have to be marked in a specific order - if it is more convenient you may mark the end of the block first.

The Clipboard: Copy, Cut & Paste

Lattice C provides conventional clipboard facilities, as popularised by the Apple Macintosh. Once you have marked a block you may copy it to the clipboard by selecting Copy from the Edit menu. The main text will remain as it is. The contents of the clipboard may then be inserted at another position by moving the cursor there and selecting Paste.

The current block may be deleted using Cut from the Edit menu; selecting Paste will then insert the block that was cut (unless you have used Copy in the mean time). Thus to move a block with this method, Cut the block from its original position and then Paste it into its new one.

The block menu also gives you the flexibility of the following commands.

Saving a block

Once a block has been marked, it can be saved by clicking on Save Block from the Block menu or by pressing key F3. If no block is marked, the message `What blocks!` will appear. If the start of the block is textually after its end the message `Invalid block!` will appear. Both errors abort the command. Assuming a valid block has been marked, the GEM file selector will appear, allowing you to select a suitable disk and filename. If you save the block with a name that already exists the old version will be overwritten - no backups are made with this command.

Copying a block

A marked block may be copied, memory permitting, to another part of the text by moving the cursor to where you want the block copied and clicking on Copy Block or by pressing key F4. If you try to copy a block into a part of itself, the message `Invalid block!` will appear and the copy will be aborted.

Deleting a block

A marked block may be deleted from the text by clicking on Delete Block or by pressing Shift-F5. The shift key is deliberately required to prevent it being used accidentally. A deleted block is remembered, memory permitting, in the clipboard, for later use. This is equivalent to Cut on the Edit menu.

Copy block to block buffer

The current marked block may be copied to the block buffer, memory permitting, using Remember Block or by pressing Shift-F4. This can be very useful for moving blocks of text between different files by loading the first, marking a block, copying it to the block buffer then switching to another window or loading the other file and pasting the block buffer into it. This is equivalent to Copy on the Edit menu.

Pasting a block

A block in the clipboard may be pasted at the current cursor position by clicking on Paste Block or by pressing F5. This is equivalent to Paste on the Edit menu.

NOTE

The contents of the clipboard is lost if the edit buffer size is changed and after a compilation.

Printing a block

A marked block may be sent to the printer by clicking on Print Block or by pressing Alt-W. An alert box will appear confirming the operation and clicking on OK will print the block. The printer port used will depend on the port chosen with the Control Panel, or will default to the parallel port. Tab characters are sent to the printer as a suitable number of spaces, so the net result will normally look better than if you print the file from the Desktop.

NOTE

If you try to print when no block is marked at all then the whole file will be printed.

Block markers remain during all editing commands, moving where necessary, and are only reset by the commands Delete block and Load.

Deleting text

Delete line

The current line can be deleted from the text by pressing Ctrl-Y.

Delete to end of line

The text from the cursor position to the end of the current line can be deleted by pressing Ctrl-Q

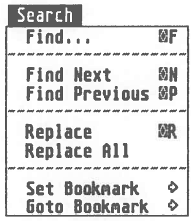
UnDelete Line

When a line is deleted using either of the above commands it is preserved in an internal buffer, and can be re-inserted into the text by pressing Ctrl-U, or the Undo key. This can be done as many times as required, particularly useful for repeating similar lines or swapping individual lines over.

Delete block

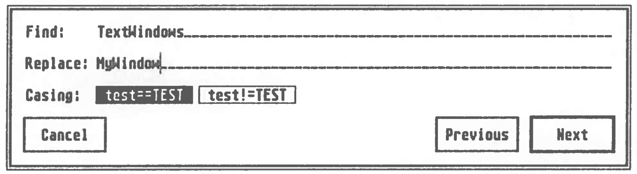
A marked block may be deleted from the text by clicking on Delete Block or by pressing Shift-F5. The shift key is deliberately required to prevent it being used accidentally. A deleted block is remembered, memory permitting, in the clipboard, for later use. This is equivalent to Cut on the Edit menu.

Searching and Replacing Text



The commands on the Search menu may be used for finding and perhaps replacing existing text. The strings involved are set up by selecting Find or press Alt-F.

This allows you to enter the find and replace strings as shown in the following dialog box:



Find: TextWindows

Replace: MyWindow

Casing: test==TEST test!=TEST

In the example above TextWindows has been entered as the find string and MyWindow as the replace string.

If you click on Cancel, no action will be taken; if you click Next (or press Return) the search will start forwards, while clicking on Previous will start the search backwards. If you do not wish to replace, leave the replace string empty.

If the search is successful, the screen will be re-drawn with the cursor positioned at the start of the string. If the string could not be found, the message Not found will appear in the status area and the cursor will remain unmoved.

Whether `test` is treated as the same as `TEST` or `Test` etc. depends on which Casing button is selected. In the example above the search would stop if `TEXTWINDOWS` was found; if test!=Test was selected then the search would not find `TEXTWINDOWS` .

To find the next occurrence of the string click on Find Next from the Edit menu, or press Alt-N. The search starts at the position just past the cursor.

To search for the previous occurrence of the string click on Find Previous from the Search menu, or press Alt-P. The search starts at the position just before the cursor.

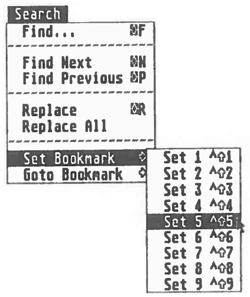
Having found an occurrence of the required text, it can be replaced with the replace string by clicking on Replace from the Search menu, or by pressing Alt-R. Having replaced it, the editor will then search for the next occurrence.

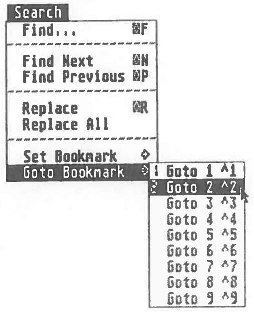
If you wish to replace every occurrence of the find string with the replace string from the cursor position onwards, click on Replace All from the Search menu. During the global replace the Esc key can be used to abort when the status area will show how many replacements were made. There is deliberately no keyboard equivalent for Replace All to prevent it being chosen accidentally.

To search and replace Tab characters press Ctrl-I when typing in the dialog box. Other control characters may be searched for in a similar manner except for the CR (Ctrl-M) and LF (Ctrl-J) characters. Alternatively, press Shift-Ins and this will display the character set from which you may pick the required character with the mouse.

Bookmarks

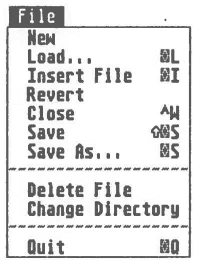
A further way to navigate your source text is via the use of bookmarks. A bookmark is set by selecting the appropriate Set Bookmark item from the Search menu or by using Ctrl-Shift and a digit key (not the numeric keypad). When you set a bookmark the corresponding item on the Goto Bookmark menu will become enabled. Then, selecting this item, or by pressing Ctrl and the digit, will return you to the original position.





When you set a bookmark, the window number to which it refers is displayed in the menu. Going to a bookmark may cause you to switch windows. Note that bookmarks that are set in a given window are lost when you close that window.

Disk Operations



The File menu contains many operations that involve using the disk system; you can save and load your source file, insert text into your source, delete a file from a disk and more.

New

Select New to open an empty window, assuming that there is one available - you are allowed up to seven windows at once in Lattice C.

Assuming that there are no more than six windows open, New will create a window which is empty and has no title.

Loading Text

To load in a new text file, click on Load from the File menu, or press Alt-L. This will open a new window (and warn you if no more windows are available) or select an unused window and then a file selector will appear, allowing you to specify the disk and filename. Assuming you do not Cancel, the editor will attempt to load the file. If it will fit, the file is loaded into memory and the new window is re-drawn. If it will not fit an alert box will appear warning you, and you should use Preferences to make the edit buffer size larger, then try to load it again.

If the file can't be found a dialog box will appear, asking you if you wish to create that file. You may do so, or alternatively modify the filename and try again.

When loading Lattice C from a CLI, you may include up to seven filenames. The corresponding files will then be loaded automatically. If a file cannot be found you will be asked if you wish to create it or may change the filename if you wish. If you use the desktop to install Lattice C as a *GEM takes parameters* (GTP) program then you may also enter up to seven file names to be loaded.

Revert

Revert will warn you that you are about to lose the text in the selected window and, assuming that you choose to continue, it will then re-load the last saved version of the file that you were editing in this window.

Revert will do nothing if you try to use it on a file that has not been saved previously.

Save As...

To save the text you are editing, click on Save As... from the File menu, or press Alt-S. The File Selector will appear, allowing you to select a suitable disk and filename. Clicking OK or pressing Return will then save the file onto the disk.

If you click on Cancel the text will not be saved. Normally if a file exists with the same name it will be deleted and replaced with the new version, but if Make backups is selected from Preferences then any existing file will be renamed with the extension .BAK (deleting any existing .BAK file) before the new version is saved.

Save

If you have already done a Save As (or a Load), the editor will remember the name of the file and display it in the title bar of the window. If you want to save it without having to bother with the file selector, you can click on Save on the File menu, or press Shift-Alt-S, and it will use the old name and save it as above. If you try to Save without having previously specified a filename you will be presented with the File Selector, as in Save As.

Inserting Text

To read a file from disk and insert it at the current position in your text, click on Insert File from the File menu, or press Alt-I. The File Selector will appear and assuming that you do not cancel, the file will be read from the disk and inserted, memory permitting.

Delete File

You may want to delete a file from disk (if for instance you have run out of disk space whilst trying to save); click on Delete File. The File Selector will appear, allowing you to select a suitable disk and filename. Clicking OK or pressing Return will then delete the file from the disk. If you click on Cancel the file will *not* be deleted.

Close

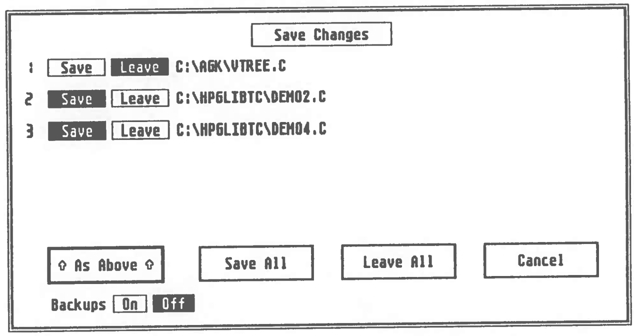
This is the same as pressing Ctrl-W and will close the currently selected window. If the file that is being edited in this window has been changed since it was loaded or is a new file, you will be warned before the window is closed. You can choose to continue and lose your changes, cancel the action or save the changes.

Change Directory

This option allows you to move the current directory path; this can be useful when running programs which expect all of their files to be in the same place as the program itself. After clicking on Change Directory the File Selector will appear, allowing you to select a suitable disk and folder name. Clicking OK or pressing Return will then change the directory. If you click on Cancel the directory path will not be changed.

Quitting Lattice C

To leave Lattice C, click on Quit from the File menu, or press Alt-Q. If changes have been made to the text which have not been saved to disk, an alert box will appear asking for confirmation.


Save Changes
1 Save Leave C:\AGK\VTREE.C
2 Save Leave C:\HPGLIBTC\DEMO2.C
3 Save Leave C:\HPGLIBTC\DEMO4.C
⇧ As Above ⇧
Save All
Leave All
Cancel
Backups On Off

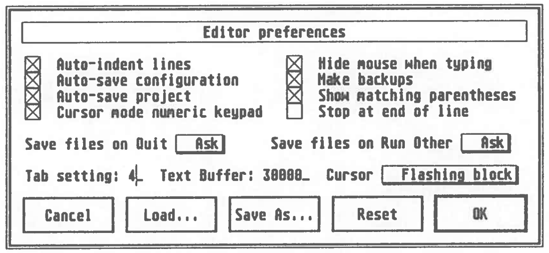
This example shows that two files have changed. Clicking on Save All, As Above or pressing Return will exit the editor saving the changes. Clicking on Cancel will return to the editor. Leave All will ignore all the changes you have made.

If you wish to save some files but not others click on the appropriate Leave buttons. For example if you clicked on the Leave button by VTREE.C in the above example and then pressed Return, only DEMO2.C and DEMO4.C would be saved.

You can also enable and disable backups from this dialog box. This is useful if you normally use backups, but decide that you don't require a backup of a one line change.

Configuring the editor

Selecting Preferences... from the Options menu will produce a dialog box like this:



Editor preferences

- [✓] Auto-indent lines
- [✓] Auto-save configuration
- [✓] Auto-save project
- [✓] Cursor mode numeric keypad
- [✓] Hide mouse when typing
- [✓] Make backups
- [✓] Show matching parentheses
- [ ] Stop at end of line

Save files on Quit: Ask   Save files on Run Other: Ask

Tab setting: 4   Text Buffer: 30000   Cursor: Flashing block

The editor preferences box

This box allows you to set up the editor as you would like to use it; you can then save your customisation to disk so that the editor will always behave the same way. Here are the different settings that you can change.

Auto-indent lines

Selecting this option sets auto-indent mode. When active, an indent is added to the start of each new line created when you press Return. The contents of the indent of the new line is taken from the white space (i.e. tabs and/or spaces) at the start of the previous line. This allows you to lay out your program neatly, by simply pressing Return.

Auto-save configuration

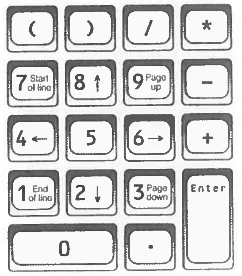
When this option is selected, the current preferences will automatically be saved when you exit the editor. So when you load the editor again, the preferences will be just the same as when you last used it.

Auto-save project

When this option is selected, the current project will automatically be saved when you exit the editor. So when you load the editor again, the project (including the compiler's options) will be just the same as when you last used it.

Cursor Mode Numeric pad

The Cursor Mode Numeric Pad option allows the use of the numeric keypad in an IBM-PC-like way allowing single key presses for cursor functions, and defaults to Cursor pad mode. The keypad works as shown in diagram below:



When this option is not selected the keyboard reverts to returning the digits etc.

Hide mouse when typing

Selecting Hide mouse when typing causes the mouse pointer to disappear when you start entering text with the keyboard. As soon as you move the mouse, or use a command that displays a dialog box, the mouse will re-appear. This option may be disabled if you prefer to always see the mouse on the screen.

Make Backups

Selecting this option causes the editor to make a backup (with the extension .BAK) when saving files.

Show matching parentheses

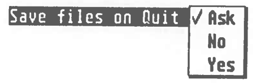
This facility lets you check that your parentheses match. With this option enables, when you press) the cursor will quickly move to any matching (character and then back to the current position, thus you can ensure that you have closed the correct number of brackets in a complex expression. If you find this cursor movement distracting then disable the option.

Stop at End of Line

When this option is selected, if you press cursor left at the beginning of a line or cursor right at the end of line, the cursor does not move. Disabling this option, causes the cursor to move to the previous line if you press cursor left at the beginning, and to the next line if you press cursor right at the end.

The best way to find out which you prefer is to try using each setting.

Save files on Quit


Save files on Quit
✓ Ask
No
Yes

By default the editor will prompt you, if you are about to quit without having saved all the files, you have changed.

The saving of these files can be made automatic by selecting Yes or disabled by selecting No (but don't blame us if you forget to save your files!).

Save files on run other

This enables you to choose whether files are saved before using the Run Other and Run with Shell commands, in the same way as that for Save files on Quit.

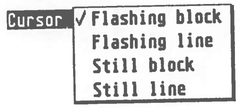
Tab setting

By default, the tab setting is 4, but this may be changed to any value from 2 to 16.

Text Buffer

By default the text buffer size is 10000 bytes, but this can be changed from 4000 to 999000 bytes. This determines the largest file size that can be loaded and edited. This amount of memory is allocated for each window in use. Care should be taken to leave sufficient room in memory for compilations - pressing the Help key displays free system memory, and for compilations this should always be at least 100k bytes. Changing the editor workspace size will cause any text you are currently editing to be lost, so a confirmation is required if it has not been saved.

Cursor


Cursor
✓ Flashing block
Flashing line
Still block
Still line

By default the editor cursor is a flashing block, but this can be changed as required.

Load...

This button lets you load a settings file. The editor settings are normally stored in a file called HISOFTED.INF in the current directory, but the editor will 'look down' both the AES and GEMDOS paths. If you want to use more than one set of preferences, then you can explicitly load a settings file.

Saving preferences

To save the settings file you can either choose Save as... from the Preferences box or choose Save preferences from the Options menu.

This latter command, on the Options menu, saves the current editor, compilation and Tools menu preferences under the name HISOFTED.INF. If you want to call your settings file a different name you should use Save as... in the Preferences... box, as described below.

When the editor is loaded, it looks for the HISOFTED.INF configuration file firstly in the current directory (which is the folder where you double-clicked on the data file), then using the system path. Saving the editor preferences this way will put the .INF file in the same place it was loaded from or, if it was not found, it will be placed in the current directory path.

In addition to saving the editor configuration the current program buffer size, from within the compilation options dialog box, is also saved.

Use Save as... from the Preferences box to save a settings file with a name other than HISOFTED.INF; an extension of .INF is still usual.

With this option you can save a number of different settings files under different names; however the editor always loads the settings file called HISOFTED.INF when it starts up so that, if you want to make a particular settings file the default, you will need to re-name it to HISOFTED.INF.

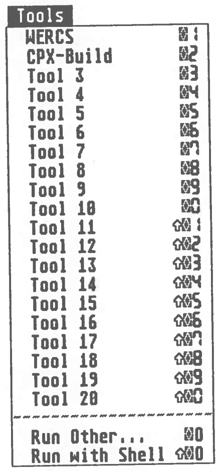
Reset

Clicking on this box causes the settings to be reset to their default values; useful if you have made a complete mess of your options.

Running other programs

There are three ways that you can execute other programs from within the editor; Run Other..., Run with Shell... and by a selection from the Tools menu. These different methods will now be described.

Tools Menu



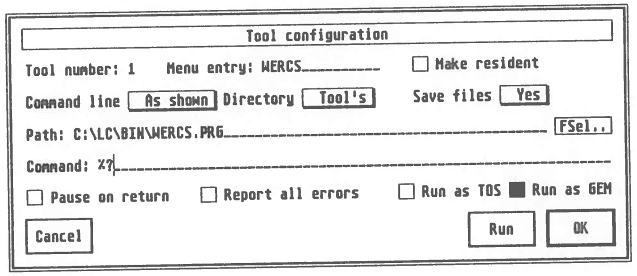
The Tools menu lets you run programs of your choice from within the editor using a single keystroke or click of the mouse.

The configuration can be saved in the preferences file, ensuring that the same facilities can be used again, the next time that you run the editor.

The preferences file that we supply is already set up to run the tools supplied with Lattice C.

Before you can use this facility you will need to configure each tool so that the editor can find the appropriate file. To configure a tool, hold down the Ctrl key and select the appropriate menu item or press Ctrl-Alt and the appropriate key *on the numeric keypad*.

This will produce a dialog box like this:


Tool configuration

Tool number: 1 Menu entry: WERCS ☐ Make resident

Command line As shown Directory Tool's Save files Yes

Path: C:\LC\BIN\WERCS.PRG FSel..

Command: %?

☐ Pause on return ☐ Report all errors ☐ Run as TOS ■ Run as GEM

Cancel Run OK

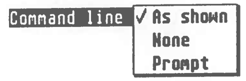
If you just want to use the default settings, you need only change the Path item so that the file can be found; either amend this item or click on FSel and use the file selector to select the appropriate file.

Once you have made the required changes you should press Return (or click on OK) to make your changes permanent; alternatively pressing Cancel will ignore any changes you have made. The other options in this box are:

Menu entry

The name typed in this field gives the name of the tool as placed on the Tools menu. Hence in the above example the name WERCS appears on the menu.

Command line

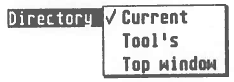


These options configure the way the command line is obtained for a program which is about to be run.

If None is selected then a program will be run as a plain GEM or TOS program with no command line. If Prompt has been selected you will be prompted for a command line in the same way as occurs when using Run Other.

Finally As shown allows the command line on the line below to be used. This command line is specified in the same way as that used by Run with Shell and may have the same meta-characters in it, as in the example above.

Directory



This sets up which directory will be the current one when the tool is run. Current will leave the directory as that of the editor itself.

Tool's switches to the directory of the tool being run, whereas Top window switches to where the file in the current window is stored on disk.

Save Files

This option changes which files will be saved before running the tool. If you select No then no files will be saved, selecting Yes (the default) will save *all* files (not just the current window), whilst Ask... will prompt you using the Save/Leave dialog described under Quitting Lattice C.

Path

This option specifies which program is actually to be run. If you give a full pathname, or select one by clicking on the FSel.. button then that specific file is run. If you just use a name then this will be treated as if you had used it as an argument to the Run with Shell command described above.

Pause on return

This option controls whether the editor pauses after running the tool. Typically you will select this when running a TOS program but disable it when running a GEM program.

Report all errors

This option allows you to specify which errors the editor will bring to your attention when returning. If this option is not selected then you will only be alerted to negative return codes from programs, i.e. those normally indicating GEMDOS errors. Selecting it will also force positive program error returns to be flagged.

Run as TOS & Run as GEM

These buttons select how the program is run, either as a GEM program or as a TOS program; note that the same warnings about GEM/TOS mode made under Run with GEM apply here also.

Make Resident

If this item is selected then when the editor next loads it will attempt to load this tool into memory and make it resident, i.e. merely execute the tool from memory rather than load it from disk each time. This is particularly useful with substantial programs like WERCS.

As well as the obvious disadvantage of permanently tying up your memory, not all programs can be made resident. In general your own Lattice C programs can be made resident if compiled using the Resident startup option.

We do not recommend running third party programs in this way. They may crash immediately, or the second time they are run or may simple not quite work correctly possibly destroying your valuable files in the process.

Running Tools

Running a configured tool is simple, just select the appropriate menu item or press Alt and the appropriate key *on the numeric keypad* and the program will be run using the settings described above.

Run Other...

This command, on the Tools menu (also reached by Alt-0), lets you run other programs from within the editor, then return to it when they finish.

When you select Run Other... you will first be warned if you have not saved your source code (unless you have modified the setting of the Save files on Run Other option in Preferences). Then the GEM File Selector will appear, from which you should select the program you wish to run. If it is a .TOS or .TTP program you will be prompted for a command line, and then the screen will be initialised suitably.

This is the command to use for 'one-off' execution of a program within the editor. If you are likely to want to run the same program a number of times, then use the facilities of the Tools menu. If you would prefer to specify the program to run via a command line, rather than using the File Selector then use the Run with Shell command described below.

If you include the character sequence `%.` (i.e. per cent followed by full stop) in the command line (remember, you are prompted for a command line) these characters will be replaced by the full name of the file that you are currently editing. To pass the name without its extension, use `%?`.

If you need a true `%` to be passed type `%%`.

Run with Shell...

This command also lets you run other programs from within the editor, then return to it when they finish. The keyboard shortcut for this command is Shift-Alt-0.

It differs from Run Other in that you enter the file to run as a command line. If the editor finds that the `_shell_p` vector has been set up then this will be called to execute the command. This works well with the Craft, PKS and Gulam shells as the shell can be used to run batch files and expand file wildcards etc.

If the `_shell_p` vector has not been set up then the editor will look for the file to run using the PATH environment variable, which can be set using the Environment command from the Options menu.

The same expansion of the current filename as used by Run Other can be used by this command. If you wish to use the same command more than once you will probably save time by using the Tools menu.

Setting the Path

The editor maintains a number of directory paths to make the operation of the integrated environment natural and seamless.

Paths are routes to files. Normally you keep all files of a similar type or usage in one folder or you may have a number of related folders all within one outer folder. For example you probably have an LC folder containing the Lattice C program, its tools and its libraries.

In order that a program that uses these files can find them without having to ask the user for help, both the ST/TT operating system and the Lattice C editor maintain a number of directory paths, some of which you can alter.

Here is a summary of the paths used by the integrated environment, how they are set and what uses them:

Current directory - this is a path that is set up (initially) by the program which ran the current program. For example, for the Lattice C editor this path will have been set up by the Desktop, assuming of course you ran Lattice C from the Desktop. However, since the editor allows this to be changed (via the Change Directory command on the File menu), it is normally reset to whatever was last stored in the HISOFTED.INF file, to save you having to change it every time you run the editor.

Most of the disk-related functions within the editor will search this path first.

GEMDOS path - this path is that contained in the PATH environment variable. It is used by shells (e.g. Craft, PKS Shell, Gulam) to locate programs to run. It is specified as a list of , or ; separated folder names, each of which specify a folder which should be searched when trying to locate a file.

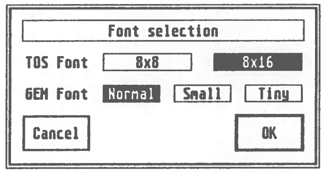
Within the editor it is used by Run with Shell, to locate the named program, and initially when attempting to find the LC1.LC file. Other tools, like WERCS, may use it for locating subsidiary files, such as WERCS.RSC and WERCS.INF.

AES path - this is the path used by the AES when the user calls one of the AES routines which search for a file (`shel_find` and `rsrc_load`). Internally the format of this variable is identical to the GEMDOS path (in fact it *is* the GEMDOS PATH for the AES program!), although the AES provides no way of altering it and merely sets it to `A:\` for a floppy based machine or `C:\` for a hard disk machine.

Miscellaneous Commands

Fonts...

The Fonts command is used to select different GEM or TOS fonts for use in the editor; it can be selected either by clicking on Fonts... from the Options menu, or by pressing Ctrl-G. It displays a dialog box like this:



The GEM Font is the font that will be used by the editor to display text. In ST high resolution and the TT resolutions, there are three fonts available as above. Changing to Small will double the number of line displayed on the screen. With the Tiny font the characters are only 6 pixels by 6 pixels wide but this does mean that even in ST high resolution, there are over 100 characters per line and 54 lines!

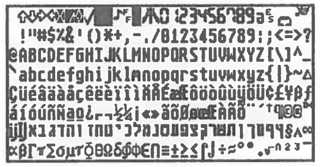
In ST medium resolution, there are only two fonts; Normal and Small. Small is 6 by 6 pixels and thus the characters are difficult to read but this does give an extra 7 lines of text and over 100 characters per line.

TOS font is used by non-GEM programs. In TT medium resolution, using 8x8 will give 60 lines instead of 30.

You should be aware that any change of font that you make here will also be effective outside the editor, after you leave it.

ASCII Table...

To be found on the Edit menu, this displays a pop-up dialog box at the current mouse position, showing all the ASCII characters:

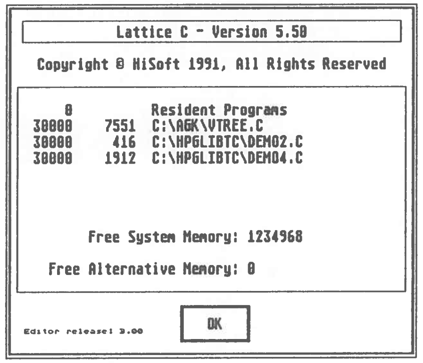


You may click on an individual character and it will be added to the text that you are editing at the current cursor position. You can bring up this display from the keyboard using Shift-Insert. This short cut can also be used in the editor's dialog boxes.

Note that the characters that would confuse the editor are 'greyed out' and may not be selected. Remember that characters other than those in the standard 7 bit ASCII set are not necessarily the same on other computers.

About Lattice C

It you select About Lattice C... from the Desk menu, a dialog box will appear giving various details about Lattice C, including its version number. You will also be told the amount of free memory that is available to you and how much is used by the resident programs including the text in the open windows.



```
Lattice C - Version 5.50
Copyright © HiSoft 1991, All Rights Reserved

    0          Resident Programs
30000   7551   C:\AGK\VTREE.C
30000    416   C:\HPGLIBTC\DEMO2.C
30000   1912   C:\HPGLIBTC\DEMO4.C

        Free System Memory: 1234968
   Free Alternative Memory: 0

Editor release: 3.00        OK
```

Pressing Return or clicking on OK will return you to the editor.

Help Screen

The key equivalents for the commands not found in menus can be seen by pressing the Help key, or Alt-H. A dialog box will appear showing the cursor and function keys, as well as the free memory left for the system.

Desk Accessories

If your system has any desk accessories, you will find them in the Desk menu. If they use their own window, as Control Panel does, you will find that you can control which window is at the front by clicking on the one you require.

For example, if you have selected the Control Panel it will appear in the middle of the screen, on top of the editor window. You can then move it around and, if you wish it to lie 'behind' the editor window, you can do it by clicking on an editor window, which brings the editor window to the front; you can then re-size it so you can see some part of the control panel's window behind it. When you want to bring the control panel back to the front just click on it and the editor window will go behind. The editor's cursor only flashes and the menus only work when an editor window is at the front.

Automatic Launching

You may configure Lattice C to be loaded automatically whenever a source file is double-clicked from the Desktop, using the *Install Application* option.

To do this you first have to decide on the extension you are going to use for your files, which we recommend to be .C for C files. Having done this, go to the Desktop, and click once on LC5.PRG to highlight it. Next click on Install Application from the `Options` menu and a dialog box will appear. You should set the Document Type to be `C` (or whatever you require), and leave the GEM radio button selected. Finally click on the OK button (if you press `Return` it will be taken as `Cancel`).

Having done this, you will return to the Desktop. To test the installation, double-click on a file with the chosen extension which must be on the same disk and in the same folder as Lattice C and the Desktop will load Lattice C, which will in turn load in the file of your choice ready for editing or compilation.

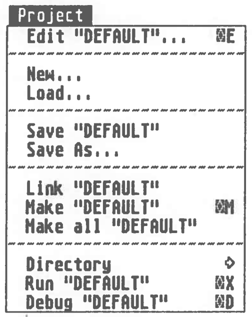
Note: To make the configuration permanent, you have to use the Save Desktop option.

Compiling Programs

Having produced your C program and saved it to disk using the editor you can then compile it. Normally this is a two stage process; first the compiler turns the C source file into an object file and then the linker links this object (together with any other object and/or libraries required) to produce an executable program.

Because most larger programs consist of more than one C source file, the integrated compiler provides a *Project manager* to aid in the maintenance of larger projects. For small, single file projects, the project manager is still employed using a *default* project.

The Project menu



The Project menu allows the entire structure of an application to be set up and managed from within the integrated compiler.

A project file (.PRJ) contains a list of all the files (and which files they include) and options which are required to rebuild an application.

Associated with each project is a project directory, the directory in which the project file is located. This directory is used as the current directory for both compiling an running the program, thus ensuring that subsidiary files (such as resource files) can be easily located.

New...

The New... option is used to create a new project, and presents a file selector to allow you select the project directory, and the name of the project file. Note that you should decide on a name for your project at this time to ensure that output file names are chosen appropriately.

Load...

The Load... option reloads a previously saved project. On loading the project the current in-memory project definition is completely replaced by the new one, together with all its options.

Save "..."

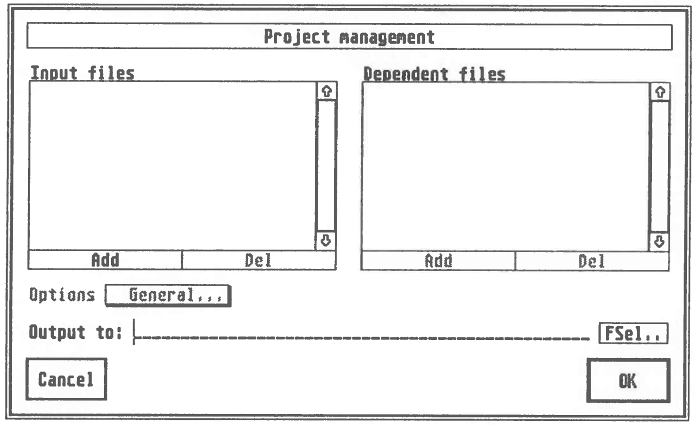
This saves the current project under the name originally given to the project.

Save As ...

Save As... is used to save the current project under a new name; this may be useful if you are editing an old project to create a new project.

Edit "..."...

The Edit... item is the central part of the project interface. It allows the files which form part of a project to be set up, and the final output file to be set. The following large dialog box is used:



The main part of the Project management dialog consists of two list boxes. The left hand one is used to enter the source files which make up the project; the files may be added, deleted or changed in the normal list box manner. In addition it is possible to change the 'build order' of the files (i.e. the order in which they are built & then linked) by clicking on an entry and dragging it to a new position.

Within a project any number of input files may be specified in the Input files list; the extension of the file is used to decide how the project manager will process it:

Extension	Action
C ‘ None	Compile named file
S	Assemble named file
LIB	Include named library file when linking
LNK	Include file as WITH file during linking
O Other	Include named object file when linking
PRJ	Make sub-project substituting target name in source file list

For each source file (.C/.S) within a project dependent files may be specified; these are the files which are *included* by the main source file. To specify dependent files for a particular source file, selecting the source file will activate the Dependent files list box allowing any number of dependents to be entered.

In addition to allowing dependent files a source file may have specific options set. Note that this is only required where a particular source file should have specific options, normally the options for the whole project should be set via the normal options menu. To set a set of file specific options first select the file and then select the appropriate options box which you wish to change via the Options popup menu.

The output filename may be specified in the Output to line or via a file selector after clicking on FSel. The extension of the output file name is used by the project manager to decide what sort of object file is to be built:

Extension	Action
LIB	Pass input files to librarian
O	Pass input files to linker for pre-linking (PRELINK keyword)
Other	Pass input files to linker and any other automatic files (startup stubs, math libraries, gem libraries & C libraries) for linking into an executable.

Note that the project manager recognises any non .LIB or .O extension as a normal executable file. It is often easiest to give the output file no extension and instead allow one to be invented based upon the setting of the *Executable options - Application type* setting.

Make "..."

Make "..." is used to start the process of rebuilding a project. The project manager examines the time and dates of all the input source files with respect to their object files and rebuilds only those that have changed.

Note that you *must* ensure that the real time clock in your machine is set; *if you do not the project manager will not function correctly.*

Make all "..."

Make all "..." is used to unconditionally rebuild a project. This can be useful if you have changed a global option which would affect the 'code model' and so all source files need rebuilding. This is obviously much slower than the Make "..." option!

Link "..."

Link "..." is used to unconditionally rebuild the output file; note that all the input files must be available, the project manager will not attempt to build any of them. Link "..." is useful in situations where you have simply changed one of the linker options (e.g. *Linker options - Add exported symbols*) and want the project manager to build a new output file.

Note that the name Link "..." is something of a misnomer; if the output file is a library (*output*.lib) then the librarian would be run instead!

Run "..."

Run "..." runs the output file for the current project. Note that the output file must be executable in order to select this option; if the output file ends in `.O` or `.LIB` you will not be able to run it directly.

Debug "..."

Debug "..." passes the output file for the current project to the debugger. The options to the debugger may be set using the *Options - Debugger...* option.

Note that the output file must be executable in order to select this option; if the output file ends in `.O` or `.LIB` you will not be able to run it directly.

Directory

Directory is used to select which directory is current when a program is run.

Because many GEM programs have difficulty finding subsidiary files when run from remote directories the default setting of *Directory - Project* ensures that the current directory is that of the project. For TOS or more robust GEM programs the *Directory - Current* is useful to ensure that a program will function correctly when run from a remote directory.

Problems

When issuing a Run command from the editor the machine may seem to 'hang up' and not run the program. This occurs if the mouse is in the menu bar area of the screen and can be corrected by moving the mouse. Similarly when a program has finished running, the machine may not return to the editor. Again, moving the mouse will cure the problem. This is due to a feature of GEM beyond our control.

Compilation Errors

When the compiler detects an error or something that may be an error (a warning) it generates a message; these errors are remembered, and can be recalled from the editor.

When you return to be editor you can use Alt-J to move to the next error with the error message displayed in the status line. If you have a large number of errors the editor may not be able to remember them all. Alt-J goes to the next error regardless of the position of the cursor; it will switch windows if required. To go to a previous error use Ctrl-J. The editor takes account of any insertions or deletions automatically so that unless one error (like a mistake in a definition) has caused multiple errors you should only need to compile once.

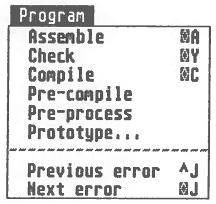
There's also the Shift-Alt-J command which finds the next error after the cursor in the current window. It is the appropriate one to use if you have got a number of include files and want to fix all the errors in one file before going on to the next one. You can also use it to find the first error in a file by typing Alt-T (to go to the top) and then Shift-Alt-J.

Occasionally the compiler will spot errors somewhat later than you might expect. This is usually because the text up to the point it has read is allowed in a certain context. If you have missed something out at the end of a line, then the error may be detected at the beginning of the next line.

On occasions the compiler will generate more than one error message as a result of a single error in your program; do not be put off by this. If you get confused, just re-compile.

Incidentally, if you start a compilation of a large program you can break out and returned to the editor using the key combination Left-Right-Shift when using the integrated compiler.

The Program menu



The Program menu provides facilities for compiling single source files in a 'one-shot' manner. Although this menu allows compilation and assembly of source files, normally you will not use these functions, preferring instead the facilities of the Project menu.

Assemble

The Assemble option is used to assemble the current window, using the global project options (described below). Since an assembly language source file will normally be part of a much larger project it is normally easier to place the description of the file within a project for later reuse.

Check

The Check option, checks the syntax of the source text that is currently being edited without producing any output file. If the compiler is already loaded then this lets you check your program quickly.

Compile

The Compile option is used to compile the current window, using the global project options (described below). Since many C source files will be part of a much larger project it is normally easier to place the description of the file within a project for later reuse.

Pre-compile

The Pre-Compile option is used to generate a pre-compiled header file suitable for inclusion in the Compiler options - Advanced (Precompiled headers) dialog. The current source window is processed together with the global project options (described below) to produce a symbol file *source*.sym.

Note that any files mentioned in the Compiler options - Advanced (Precompiled headers) dialog *will* be loaded as part of the precompilation process and so must be available (i.e. if you are rebuilding an existing pre-compiled header you should remove its declaration from the Precompiled headers list).

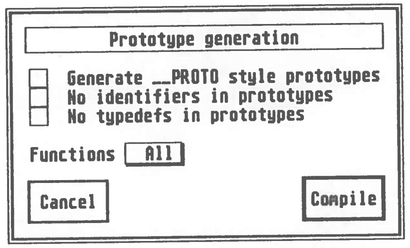
Pre-process

The Pre-process option is used to force the compiler to write the results of preprocessing the current source file into the output file *source*.p.

Prototype...

The Prototype... option is to invoke the compilers prototype generation option on the current window, using the global project options (described below), building a prototype file *source*.i.

This option produces a dialog allowing the options for the prototype generation to be set:



Prototype generation
Generate __PROTO style prototypes
No identifiers in prototypes
No typedefs in prototypes
Functions
All
Cancel
Compile

Generate __PROTO style prototypes -pp

This option forces the compiler to generate prototypes 'protected' by an `__PROTO` macro. Normally on re-reading the prototype file the prototypes will be used, however if the symbol `__NOPROTO` is pre-defined by the user then the prototypes will be ignored. This can be useful to allow portability of source files to older K&R compilers.

No identifiers in prototypes -pi

No identifiers in prototypes (`-pi`) suppresses the compilers generation of formal names for the prototypes. Since these names are not required in the prototypes removing them can save both space and compilation time later.

No typedefs in prototypes -pt

No typedefs in prototypes (`-pt`) suppresses the compilers use of `typedef` names within prototypes. In many cases this is desirable as the `typedef` may not be 'visible' to the prototype file and so any unknown types would cause the compiler to issue an error.

Functions

The *Functions* option allows the user to configure how prototyping deals with `static` functions.

There are three possible options:

All -pr

Causes the compiler to generate a prototype file containing prototypes for all functions defined in the source file.

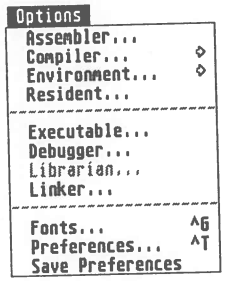
External -pe

Eliminates prototypes for all static functions. Only those functions available externally will have prototypes generated for them.

Static -ps

Generates prototypes only for static functions. Only those functions defined with the static function will be output.

The Options menu



The Options menu provides the main place from which options for both the compiler and editor are set.

The items on this menu relating to the compiler are described in this section; the other options are described elsewhere.

Environment...

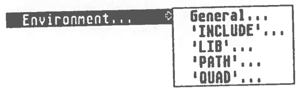
The Environment... option allows the environment variables used by the tools which are run (and other parts of the Latice C system as described above) to be altered.

The 'environment variables' are an array of strings which GEMDOS creates for every program which it runs. This array of strings contains 'variables' of the form *`name=value`* which the program may interrogate to obtain information about the 'environment' in which it is running (hence the name). Some typical environment variables are: `PATH` - specify the directories in which a program should look, `INCLUDE` - specify the directories in which a compiler should look for include files, `EDITOR` - the users preferred text editor which an application should run.

Normally a parent program creates a new environment for the child program; if this is not done explicitly then the child 'inherits' a copy of the parent's environment. Hence every program has its own, unique, copy of the environment.

For programs started normally from the Desktop, only a single environment variable is available, `PATH`.

Lattice C and the environment



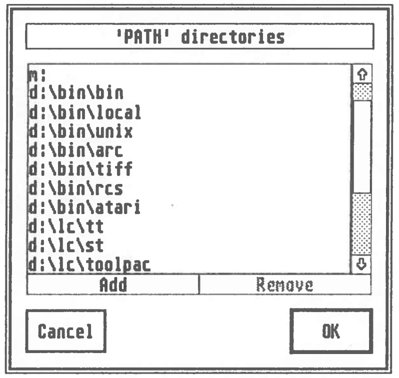
Several environment variables are used by the compiler to locate files it may need.

These are `INCLUDE`, `LIB`, `PATH` and `QUAD`.

PATH — *Executable path*

This variable defines a sequence of directory prefixes where a program (e.g. the editor!) should search for an executable file (e.g. when trying to run LC1, LC2, GO etc.).

On selecting the `Environment-'PATH'` option, the following dialog appears (although without all the entries!):



```
'PATH' directories
m:
d:\bin\bin
d:\bin\local
d:\bin\unix
d:\bin\arc
d:\bin\tiff
d:\bin\rcs
d:\bin\atari
d:\lc\tt
d:\lc\st
d:\lc\toolpac
Add          Remove
Cancel                OK
```

This shows a list of all the directory prefixes specified in the PATH variable. The entries in this list dialog may be manipulated in the manner described in the section *A word about pop-up menus and dialog*.

When a new entry is added, or an existing one edited (by double-clicking), a file selector is presented to allow the entry of a path.

INCLUDE — *Include path*

The INCLUDE variable is similar to the PATH variable except that it is used by the compiler to locate include files referenced by your program.

Manipulating this variable is done in an identical manner to that described for the PATH variable.

LIB — *Library path*

The LIB variable is similar to the PATH variable except that it is used by the linker to locate input and library files referenced by the editor.

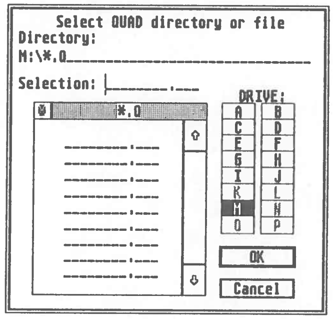
Manipulating this variable is done in an identical manner to that described for the PATH variable.

Note that just because a library file is in the library directory does not mean that the file will be linked in, you must tell the compiler to link it!

QUAD — *Quad file*

The QUAD environment variable specifies the default intermediate (QUAD) *file name* used by the compiler. If the filename has a trailing backslash (\) then the compiler assumes that this is the name of a directory such that it may form a filename by concatenating the source file name to it.

On selecting the `Environment-'QUAD'` option, a standard file selector appears to allow you to select a quad file name, or path. To select a specific name, enter it in the file selector, or to set a directory, leave the filename portion blank:



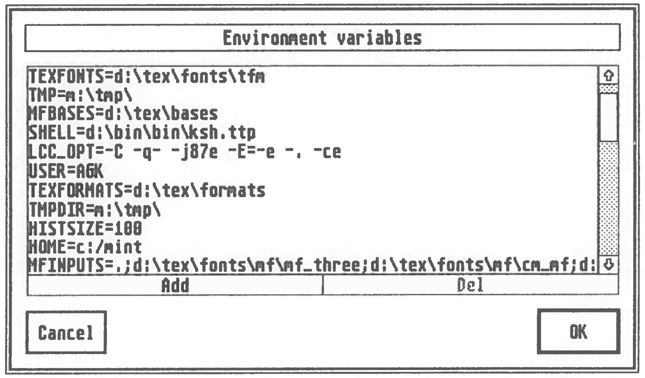
```
Select QUAD directory or file
Directory:
M:\*.Q
Selection:
*.Q
DRIVE:
A B
C D
E F
G H
I J
K L
M N
O P
OK
Cancel
```

If you have a RAM disk installed you can greatly increase compiler performance if you use this as the quad temporary directory.

General...

Because the Lattice C editor provides a full visual shell, in order that other programs which may be run from it (e.g. the Tools, etc.), full support is available for arbitrary environment variables (i.e. other than PATH, INCLUDE, LIB and QUAD).

On selecting the `Environment-General...` option, the following dialog appears (although without all the entries!):

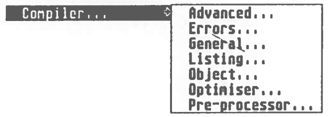


```
Environment variables
TEXFONTS=d:\tex\fonts\tfm
TMP=m:\tmp\
MFBASES=d:\tex\bases
SHELL=d:\bin\bin\ksh.ttp
LCC_OPT=-C -q- -j87e -E=-e -. -ce
USER=AGK
TEXFORMATS=d:\tex\formats
TMPDIR=m:\tmp\
HISTSIZE=100
HOME=c:/mint
MFINPUTS=.;d:\tex\fonts\mf\mf_three;d:\tex\fonts\mf\cm_mf;d:
Add    Del
Cancel    OK
```

This shows a list of all environment variables known to the editor, either because they were inherited from the parent, or because you have previously set them. The entries in this list dialog may be manipulated in the manner described in the section *A word about pop-up menus and dialog*.

Note that the editor has no way of knowing which variables are list-type variables (like PATH) and which are single assignment type (like QUAD), so it is up to you to format the entries correctly for the tools you intend to run.

Compiler options

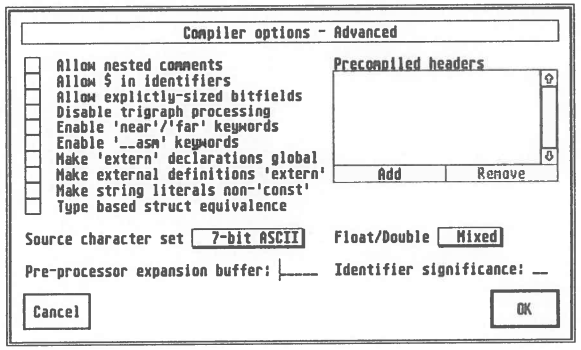


The Compiler options item and its associated sub-menu are the main place (for a project or single file) in which compilation options are set.

On selecting one of these options, one of a number of dialog boxes are used to allow the options to be set. Within the following descriptions the text of the option is shown on the left of the heading, whilst the command line option (for LC.TTP users) is shown on the right.

Compiler options - Advanced

The Advanced dialog includes all the options which are of use to experienced programmers who wish to tailor the translation environment to their needs.



```
Compiler options - Advanced
[ ] Allow nested comments
[ ] Allow $ in identifiers
[ ] Allow explictly-sized bitfields
[ ] Disable trigraph processing
[ ] Enable 'near'/'far' keywords
[ ] Enable '__asm' keywords
[ ] Make 'extern' declarations global
[ ] Make external definitions 'extern'
[ ] Make string literals non-'const'
[ ] Type based struct equivalence
Precompiled headers
Add    Remove
Source character set [7-bit ASCII]    Float/Double [Mixed]
Pre-processor expansion buffer: ___   Identifier significance: __
Cancel    OK
```

Allow nested comments -cc

Allow $ in identifiers -cd

Allow explicitly-sized bitfields -cb

This option allows the compiler to recognise bitfield specifications using types `char`, `short` and `long` rather than just the ANSI types of `int`, `unsigned int` and `signed int`.

Disable trigraph processing -cg

Disable trigraph recognition;. Trigraphs are enabled in *Strict ANSI mode* `(-ca)`; we strongly recommend that you keep trigraphs disabled due to the overhead incurred by their recognition.

Enable 'near'/'far' keywords -ck

Enables the presence of the `near` and `far` keywords even when the *Strict ANSI mode* `(-ca)` has been specified.

Enable '__asm' keywords -cr

Enables the register keywords `__d0`...`__d7`, `__a0`...`__a7` and `__fp0`...`__fp7`, even when *Strict ANSI mode* `(-ca)` has been specified.

Make 'extern' declarations global -cx

Cause all `extern` declarations to have global scope. The compiler deals with implicit and explicit in-block (cf. global) `extern` declarations according to ANSI, i.e. their scope is restricted to that of the block. This option can be used to force their scope to global, for compatibility with older non-ANSI compilers.

Make external definitions 'extern' -x

Cause all global data declarations to be treated as externals. This can be useful if you define data in a header file that is included by multiple source files. This option can be used with all the files except one, in this case, to cause the data items to be defined in one module and referenced as externals in the others.

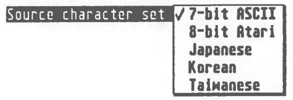
Make string literals non-'const' -ch

When using the *Merge identical strings* (`-cs`) option the compiler gives each literal string the type `const char []`, although this can create many warnings which were not intended. This option disables this behaviour, and makes string literals of type `char []`.

Type based struct equivalence -cq

Relax the aggregate type checker to allow aggregates with common initial subsequences to type check equivalent. This means that `struct A` will be considered equivalent to `struct B` if the types of `struct A`'s members match the types of `struct B`'s members over the length of `struct B`. This option is most useful where a larger structure (`struct A`) has a smaller structure (`struct B`) embedded at the start of it and you wish to pass a pointer to `struct A` to a function expecting a pointer to a `struct B`.

Source character set -e



This option is used to activate the extended character set, either for 8-bit ability or to access codes used in Asian-language applications.

Use of any of these options *disables* compressed header files, and so the full uncompressed versions must be used.

Certain Asian characters are represented by two consecutive bytes, the first byte of which has its high bit set, i.e. a value above 127. With the *Source character set* (`-e`) option, a two-byte Asian character is treated as a unit in string and character constants. This means that when the compiler scans a text string enclosed in double quotes, it will recognise the first byte of an Asian character and suppress lexical analysis of the second byte. So, if the second byte is a backslash or a double quote, it will not receive any special processing.

Also, if you enclose an Asian character in single quotes, the compiler will produce a two-byte constant, and if you have not specified *Allow multi-character constants* (`-cm`), it will warn you that multi-character constants are not allowed.

8-bit Atari -e

8-bit coding is used. This means that characters above `0x80` are treated as normal characters, thus allowing all 8-bit ASCII codes to be embedded in a program.

Japanese -e0

Japanese coding is used. This means that characters with values from `0x81` to `0x9F` and `0xE0` to `0xFC` are treated as the start of a two-byte sequence. The characters from `0xA0` to `0xDF` are single-byte Katakana codes.

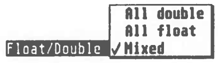
Korean -e2

Korean character coding is used. This means that characters with values from `0x81` to `0xFD` are treated as the start of a two-byte sequence.

Taiwanese -e1

Chinese/Taiwanese coding is used. This means that characters with values from `0x81` to `0xFC` are treated as the start of a two-byte sequence.

Float/double -f



These options allow control over the precision attributed to `float` and `double` declarations used within the user code.

All Double **-fd**

Causes the compiler to treat all declarations as double precision.

All float **-fs**

Causes the compiler to treat all declarations as single precision.

Mixed **-fm**

Causes the compiler to treat `float` as single precision and `double` as double precision; .this is the default option.

Pre-processor expansion buffer: *size* ***-z****size*

Whilst pre-processing the source file the compiler uses several buffers to store the pre-processed line. If this buffer overflows (giving a line buffer overflow or pre-processor symbol loop (macro expansion too long or circular) error) then the size may be increased using this option. The buffer has, by default, a size of 6000 bytes for 'big' compilers, or 3000 bytes for 'small' compilers

Precompiled headers ***-H****file.sym*

This list box specifies the precompiled header files the compiler is to pre-load into its symbol table. Precompiled header files are generated using the Pre-compile menu option.

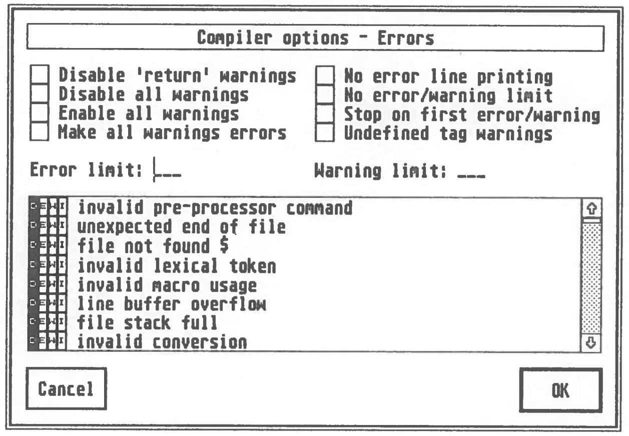
There is no limit to the number of precompiled header files that may be read in.

Identifier significance: *sig* ***-n****sig*

This option specifies the number of characters the compiler is to retain for identifiers. This can be useful if more than the default of 31 is required, or to reduce to 7 or 8 for compatibility with very old programs.

Compiler options - Errors

The Errors dialog provides options for controlling the errors reported by the compiler. This can be useful to enable additional diagnostic messages, or to suppress messages which are of no interest.



```
invalid pre-processor command
unexpected end of file
file not found $
invalid lexical token
invalid macro usage
line buffer overflow
file stack full
invalid conversion
```

Disable 'return' warnings ***-cw***

Shuts off warning messages generated for `return` statements which do not specify a return value within an `int` function. For conformance with the ANSI standard, all such functions should be declared as `void` instead of `int`.

Disable all warnings ***-j*i***

Enable all warnings ***-j*w***

Make all warnings errors ***-j*e***

Promote all warnings to errors; this option can be useful to ensure that a program compiles with no warnings whatsoever.

No error line printing ***-ce***

Suppresses the printing of the error source line in conjunction with any warnings or errors.

No error/warning limit ***-q-***

Never quit on any errors or warnings.