Lattice C 5.5

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

Installation Guide

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

Second Edition, June 1992

Table of Contents

Lattice C 5.5	***1***
Backups	***1***
The Installation Program	***2***
The Disks	***5***
1. Integrated Tools	***5***
2. WERCS & Command Line Tools	***5***
3 Header files, Example, Extras	***7***
4. Base-relative C and GEM Libraries	***10***
5 Non-base-relative C & GEM Libraries	***11***
6 Software/Auto-detecting Maths Libraries	***11***
7 Co-processor Maths Libraries	***11***
Hints for upgraders	***11***
A Lattice C 5.5 Tutorial	***13***

Lattice C 5.5 Installation Guide

Welcome to Lattice C Version 5.5, one of the most powerful and flexible programming environments available for the Atari 680x0 range of computers.

The package contains 7 double-sided, double density disks but the system is usable on any machine with at least a megabyte of RAM and a double-sided disk drive.

The contents of the disks are arranged logically (we hope) so that you can get to the files you need quickly and easily.

Backups

The first thing you should do is make a backup. You have several ways of making a backup of the Lattice C 5.5 disks. You can use the disk copying function of the Desktop to duplicate the disks, or you can use one of the many disk copiers available. *Note however, you must ensure that the backup program preserves the volume name of the original disk as otherwise the installation program will complain that you have inserted the wrong disk.*

However you do it, *please make a backup* and then store the master disks in a safe place, away from moisture, extreme heat or cold, magnetic fields (televisions, telephones etc. give off radiation harmful to disks), strong light, coffee and, above all, children and dogs! If you damage your master disks we will charge you a handling fee for re-copying them.

Now, before we go any further… please read the READ.ME file from your backup of Disk 1. You can do this by double-clicking on it from the desktop. This file will detail any last minute changes that we may have made.

The Installation Program

The GEM-based installation program is designed to ease the building of various standard configurations for the Lattice C 5 system.

For hard disk owners, the installation program will copy the files that you need to your hard disk. If you are the type of person who doesn't like installation programs that write things to your hard disk, you can view the files that would be copied, and copy them yourself. The installation takes note of your hardware configuration and only copies files that could be of use to you.

By default, the installation program won't copy the compressed headers since you will normally get better performance with the standard ones. You also don't get the tools for using the GST format as most people don't need this. The installation program for hard disk users deliberately does not automatically install a ramdisk. This is because many users will already have their own preferred ramdisk. The recommended ramdisk size for 1Mb hard disk users is 100Kb, which is supplied ready to go on disk 3. We strongly recommend that you use your ramdisk for the compiler's intermediate (quad) files. These are set up using the QUAD environment variable. See your user manual addendum for more information.

For users of floppy disk based systems, the installation program will produce two floppies that you can use to develop your programs; one boot disk and one work disk. The boot disk will just contain program files; the work disk will be used for the header files, libraries and your own source and program files. If you have two drives you can keep the boot disk in drive A and the work disk in drive B. On single drive systems we recommend that you use the same scheme, letting the Atari's operating system prompt you to change disks; if you keep all your files on your work disk you shouldn't need to do this often!

If you are using a non-hard disk system and wish to use larger than normal capacity (e.g. 800K) floppy disks, you should format two floppies using your favourite extended formatter *prior* to running the installer. Use the volume names LC5BOOT and LC5WORK. If you intend to use standard floppies then the installation program will format these for you.

We strongly suggest that you start by using the setup recommended by the installation program until you are sufficiently familiar with the package to re-configure it to meet your unique requirements.

If you have a pre-Rainbow version of TOS, you *must* run the Atari program FOLDRnnn.PRG before running the installation program; otherwise the program will terminate with an out of memory error due to the '40 folder' bug in your version of TOS. If you have a floppy-based system just boot from your backup of disk 1. If you have a hard disk and don't have FOLDRnnn.PRG in your AUTO folder copy it from the AUTO folder of disk 1.

To run the installation program, double-click on LCINST.PRG from your backup of disk 1 in Drive A. Note that the installer expects to find its subsidiary files in the current directory.

The Disks

The disks and their contents are:

1. Integrated Tools

`bin\lc5.prg` -----------	The editor and integrated shell
`bin\hisofted.inf` ------	The editor configuration file
`bin\default.prj` -------	Default project file for the integrated environment
`bin\lc1.lc` ------------	English error messages
`bin\lc1.ttp` -----------	Cut down first phase of the compiler
`bin\lc1b.ttp` ----------	First phase of the compiler
`bin\lc2.ttp` -----------	Compiler code generator
`bin\clink.ttp` ---------	The Lattice linker
`bin\monstc.prg` --------	Debugger for 68000 machines
`bin\monttc.prg` --------	Debugger for 68030 machines
`lcinst.prg` ------------	The installation program
`lcinst.rsc` ------------	File used by the installation program
`lcinst.inf` ------------	File used by the installation program
`lcdisks.dir` -----------	File used by the installation program

2. WERCS & Command Line Tools

`bin\asm.ttp` -----------	Lattice macro assembler
`bin\batcher.prg` -------	MS-DOS style shell
`bin\cpxbuild.ttp` ------	CPX builder
`bin\dercs.ttp` ---------	Resource embedder
`bin\go.ttp` ------------	Global optimiser
`bin\gstlib.ttp` --------	The GST format librarian

`bin\lc2gst.ttp`	Lattice to GST format converter
`bin\lcompact.ttp`	Header file compacter
`bin\linkst.ttp`	GST format linker
`bin\omd.ttp`	Object module disassembler
`bin\oml.ttp`	Object module librarian
`bin\strip.ttp`	Symbol removing utility

WERCS Program Files

`wercs\wconvert.prg`	Resource file name converter
`wercs\wconvert.ttp`	Resource file name converter
`wercs\wercs.inf`	WERCS settings file
`wercs\wercs.lng`	WERCS language description file
`wercs\wercs.prg`	WERCS
`wercs\wercs.rsc`	Resource file
`wercs\wimage.prg`	Image converter
`wercs\wimage.rsc`	Resource file

Non-C language WERCS Test programs

```
wercs\wtest\wrsc.mod    wercs\wtest\wtest.bas
wercs\wtest\wtest.mod   wercs\wtest\wtest.hsp
wercs\wtest\wtest.pas   wercs\wtest\wtest.s
```

3 Header files, Example, Extras

Un-compressed Header Files

The headers directory contains the pure ASCII versions of the header files and is useful when checking the prototypes or definitions of external objects.

```
headers\acc.h           headers\aes.h            headers\assert.h
headers\basepage.h      headers\conio.h          headers\cpx.h
headers\ctype.h         headers\dirent.h         headers\dos.h
headers\errno.h         headers\error.h          headers\ext.h
headers\fcntl.h         headers\fctype.h         headers\float.h
headers\fsm.h           headers\ftw.h            headers\gemextra.h
headers\gemlib.h        headers\gemout.h         headers\ieeefp.h
headears\io.h           headers\ios1.h           headers\limits.h
headers\linea.h         headers\locale.h         headers\memory.h
headers\m68881.h        headers\math.h           headers\osbind.h
headers\oserr.h         headers\portab.h         headers\process.h
headers\setjmp.h        headers\signal.h         headers\stdarg.h
headers\stddef.h        headers\stdio.h          headers\stdlib.h
headers\string.h        headers\strings.h        headers\taddr.h
headers\time.h          headers\tos.h            headers\utime.h
headers\unistd.h        headers\varargs.h        headers\vdi.h
headers\sys\file.h      headers\sys\except.h     headers\sys\param.h
headers\sys\time.h      headers\sys\dir.h        headers\sys\stat.h
headers\sys\types.h
```

Compressed Header Files

The headers in the h directory are the normal ASCII header files which have been processed by lcompact, resulting in space savings and time savings on floppy based systems. We do not recommend their use on hard disk systems.

Examples

`examples\bell\cpx\*.*`	Source, resource and project files for `bell.cpx`
`examples\bell\h\*.*`	Header files used by the bell example
`examples\bell\tsr\*.*`	Source and project files for `belltsr.prg`
`examples\clock\*.*`	Desk accessory example
`examples\cpx\master.*`	CPX resource file template
`examples\gstlib\*.*`	GST format librarian source and project files
`examples\hc\*.*`	Example of use of the HiSoft C Interpreter library
`examples\hc\lib\*.*`	HiSoft C GEM Toolbox library and project files
`examples\prog2.c`	Lesson source file
`examples\wtest\*.*`	WERCS example program

Source Files

The source code supplied in the src directory form assorted parts of the Lattice C runtime library. They are supplied so that the expert user may customise them as required.

`src\c\_assert.c`	`assert()` failure code
`src\c\_cxovf.c`	C library stack overflow code
`src\c\_cxovf.c`	GEM library stack overflow code
`src\c\_iob.c`	ANSI level I/O blocks
`src\c\_main.c`	`_main()` startup code
`src\c\_setargv.c`	`_setargv()` argument parser
`src\c\_stack.c`	Default stack size module
`src\c\_stub.c`	`_stub()` undefined linker default
`src\c\basepage.i`	Include file used by `c.s`
`src\c\c.s`	Initial program startup source

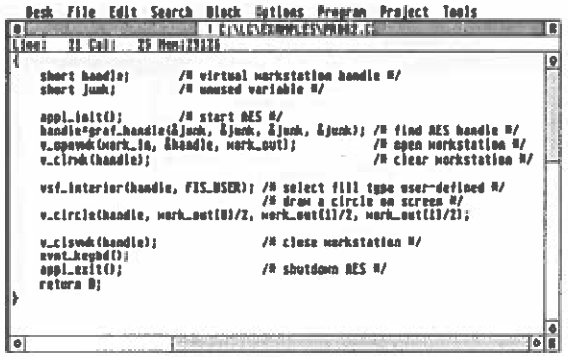
`src\c\gemdos.i`	Include file used by `c.s`
`src\c\oserr.c`	_OSERR messages
`src\c\syserr.c`	`errno` messages
`src\c\sysvar.i`	Include file used by `c.s`
`src\g\_aesaddr.c`	AES control array
`src\g\_aesglob.c`	AES control array
`src\g\_aesinti.c`	AES control array
`src\g\_aesinto.c`	AES control array
`src\g\_aespb.c`	AES control array
`src\g\_vdictrl.c`	VDI control array
`src\g\_vdiinti.c`	VDI control array
`src\g\_vdiinto.c`	VDI control array
`src\g\_vdipb.c`	VDI control array
`src\g\_vdiptsi.c`	VDI control array
`src\g\_vdiptso.c`	VDI control array
`src\m\cxferr.c`	Floating point exception handler
`src\m\cxfpe.c`	Floating point exception handler
`src\m\fpcfpcr.s`	Floating point co-processor configuration
`src\m\matherr.c`	Maths error handler

HiSoft Ramdisk Tools

The ramdisk directory contains the reset proof ramdisk and its installation program - see the Lattice C 5 Tools, hramdsk chapter in Volume I - The User Manual.

`ramdisk\hramdsk.prg`	HiSoft ramdisk program
`ramdisk\raminst.prg`	HiSoft ramdisk installation program
`ramdisk\raminst.rsc`	Resource file

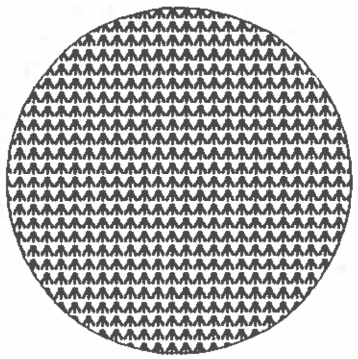
Now compile the program again, remembering to save it first. It should now compile and link successfully, to PROG2.PRG, and you can try running it.


Desk File Edit Search Block Options Program Project Tools
C:\LC\EXAMPLES\PROG2.C
Line: 21 Col: 25 Mem:[illegible]

```
{
    short handle;       /* virtual workstation handle */
    short junk;         /* unused variable */

    appl_init();        /* start AES */
    handle=graf_handle(&junk, &junk, &junk, &junk); /* find AES handle */
    v_opnvwk(work_in, &handle, work_out);           /* open workstation */
    v_clrwk(handle);                                /* clear workstation */

    vsf_interior(handle, FIS_USER); /* select fill type user-defined */
                                    /* draw a circle on screen */
    v_circle(handle, work_out[0]/2, work_out[1]/2, work_out[1]/2);

    v_clsvwk(handle);               /* close workstation */
    evnt_keybd();
    appl_exit();                    /* shutdown AES */
    return 0;
}
```

The Corrected Program

The program draws a filled circle like this:



The PROG2.PRG program running

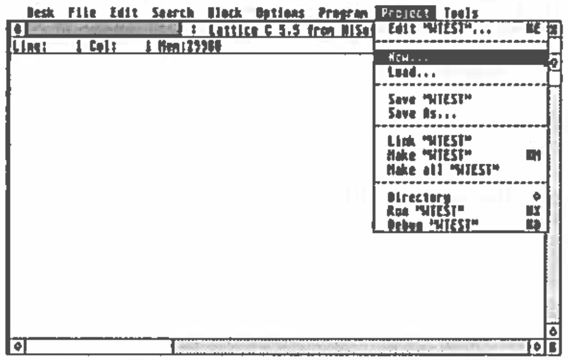
If you are wide awake you may have noticed that the compiler reported one warning and two errors (look back at the output on page 13) and that we only corrected the warning and one of the errors to obtain a successful compilation. This is because the second error was caused by the first error; the compiler became confused as to the meaning of the program and thought that a semi-colon was overdue. This is an example of a spurious error caused by a previous problem - always be on the look out for this type of error.

Lesson 3 - Optionally yours

The program we are going to build is an extended example of using the GEM system and uses structures created with the WERCS resource editor - see the chapter WERCS, The Resource Editor in Volume I for more details.

Our concern here is to show you how to set up the project which we will use to compile it, in addition to using some useful compiler options; these are used to let you modify the way the compiler behaves when compiling your program.

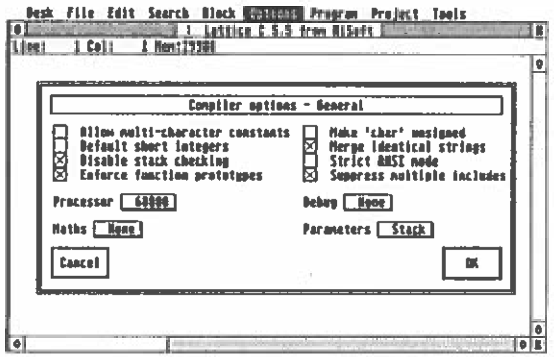
Run LC5.PRG, insert your working disk and then select New... from the Project menu.



Preparing to create a new project

When the file selector appears, locate the WTEST directory inside the EXAMPLES directory on your work disk and then type in the name WTESTEX.PRJ, this is the name of the file we are going to use as our *project file*. A project file is a file which contains all of the compiler options and source files which are need for a particular project.

The Compiler options - General box should now look like this:

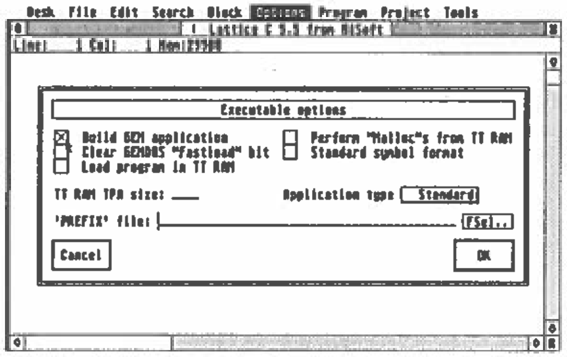

Desk File Edit Search Block Options Program Project Tools
1 Lattice C 5.5 from HiSoft
Line: 1 Col: 1 Mem:79380

```
Compiler options - General

[ ] Allow multi-character constants    [ ] Make 'char' unsigned
[ ] Default short integers             [X] Merge identical strings
[X] Disable stack checking             [ ] Strict ANSI mode
[X] Enforce function prototypes        [X] Suppress multiple includes

Processor [ 68000 ]                    Debug [ None ]

Maths [ None ]                         Parameters [ Stack ]

[Cancel]                                              [OK]
```

Setting the General *compiler options*

Now click on OK to accept the changes we have made.

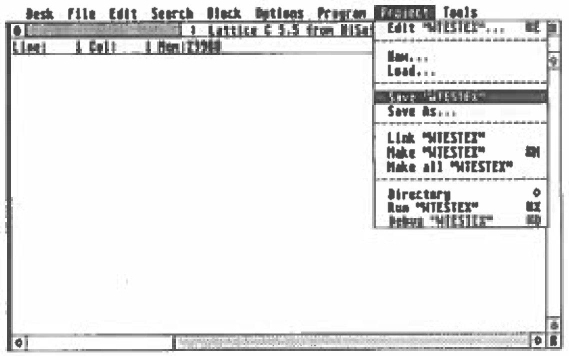
Because this is a GEM program we also need to tell the compiler to build a GEM application; we do this, as in lesson two, using the Executable options dialog:


Desk File Edit Search Block Options Program Project Tools
1 Lattice C 5.5 from HiSoft
Line: 1 Col: 1 Mem:79380

```
Executable options

[X] Build GEM application              [ ] Perform "Malloc"s from TT RAM
[ ] Clear GEMDOS "fastload" bit        [ ] Standard symbol format
[ ] Load program in TT RAM

TT RAM TPA size: ___                   Application type [ Standard ]

'PREFIX' file: |________________________________________ [FSel..]

[Cancel]                                              [OK]
```

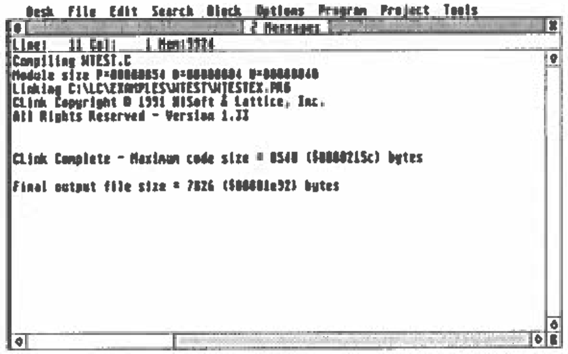
Setting a GEM type program

Having finished setting up our project we should now save the project file for future use, so select the Save "WTESTEX" option from the Project menu:



Saving the completed project file

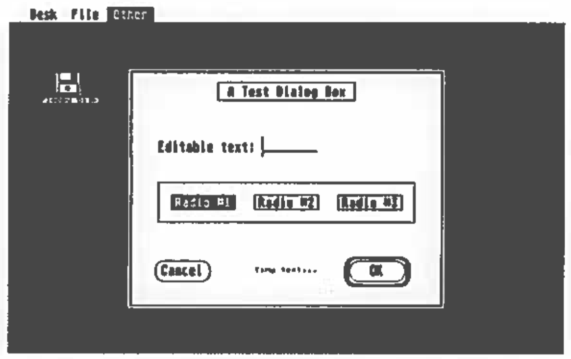
We are now ready to build the project in the normal way, by selecting Make "WTESTEX" from the Project menu; you should see a compilation report something like this:



```
Compiling WTEST.C
Module size P=00000854 D=00000004 U=00000040
Linking C:\LC\EXAMPLES\WTEST\WTESTEX.PRG
CLink Copyright © 1991 HiSoft & Lattice, Inc.
All Rights Reserved - Version 1.33


CLink Complete - Maximum code size = 8540 ($0000215c) bytes

Final output file size = 7826 ($00001e92) bytes
```

A successful 'Make'

The WTESTEX.PRG program will now be in the WTEST directory of your Work disk, and we can run it using the Run "WTESTEX" from the Project menu; the running program looks like:



WTESTEX.PRG *running*

That completes our brief but, we hope, useful introduction to using the Lattice C 5.5 system. We now encourage you to get on and use the package, referring to this and the other volumes as and when you need to.