Installing and Running XXICC

For the latest XXICC release and documentation, please visit `xxicc.org`.

This document describes how to install and run XXICC on your computer, either by compiling XXICC source code or installing binaries. These are procedures the author uses, but he may be using a different version of the C compiler or its environment. If you have problems and cannot resolve them yourself, please use the Google Groups mailing list to ask for help.

The current version of XXICC is written in both C and GalaxC. XXICC source code uses the following file types:

`.c` ANSI C source code.
`.h` C header file.
`.w` C source code containing PDF character width information, #included in `pdfwidth.c`.
`.gal` GalaxC source code containing ASCII characters and perhaps format control characters as described in *Programming in the GalaxC Language*, section "Format Control". You may be able to edit `.gal` files using your favorite text editor. Otherwise, use the XXICC Object Editor (XOE).
`.xoe` GalaxC source code containing text and graphics. Edit these files using XOE.

1. Compiling XXICC using GNU/Linux or Cygwin

The author usually compiles XXICC on an Intel architecture PC running the Ubuntu distribution of GNU/Linux or using Cygwin (http://en.wikipedia.org/wiki/Cygwin, www.cygwin.com) on a PC running Windows 2000 or later. Cygwin provides a GNU/Linux look and feel on a Windows machine. It provides the GNU `make` and `gcc` tools for compiling the C modules of XXICC for either Win32 or X Windows, and provides an X Windows implementation for testing X11 executables. It also provides useful GNU tools like `grep`, `diff`, and `sed`.

At the present time, you can compile XXICC for the following targets:

- GNU/Linux, compiled on a GNU/Linux PC. The resulting executable has been run on Ubuntu 11.10 PC and on a Raspberry Pi using Debian "Squeeze" 6-19-04-2012. It should be easy to compile and run on other GNU/Linux distributions. The GNU/Linux version uses X11 as its window manager and needs Xft (http://en.wikipedia.org/wiki/Xft) to provide FreeType scalable fonts, so both `libX11` and `libXft` must be available. Many GNU/Linux distributions (including Ubuntu) already have the fonts needed by XXICC. Otherwise you may need to get the fonts used by XXICC and copy them to a directory where Xft can find them. The Windows versions of these fonts are `cour*.ttf`, `times*.ttf`, `arial*.ttf`, and `symbol.ttf`. Except for `symbol`, these are all Core Fonts for the Web (http://en.wikipedia.org/wiki/ Core_fonts_for_the_Web) which are gratis for non-commercial use. They are all standard PDF fonts and should be present on any system that has a PDF reader. Otherwise you may need to purchase `symbol.ttf`.

- Microsoft Windows, compiled on a Windows PC using Cygwin or BorlandC. The resulting executable has been run on Windows 98, 2000, XP, Vista, and 7. It uses the standard Win32

libraries `kernel32.dll`, `gdi32.dll`, and `user32.dll`. For C library functions, XXICC uses either `cygwin1.dll` (if compiled using Cygwin) or `cw32.dll` (if compiled using BorlandC -- see §2). Windows should have all the font files needed by XXICC.

- Cygwin with X Windows, compiled on a Windows PC using Cygwin. The resulting executable runs on Cygwin's X Windows implementation. This is primarily for testing XXICC's X11 version of G-SWIM (GalaxC Simplified Windows Manager) on a Windows computer. Like other GNU versions, it needs Xft and font files where Xft can find them.

- BeagleBoard and BeagleBone (http://beagleboard.org) under the Ångström GNU/Linux distribution: this version demonstrates XXICC on an ARM processor. It also uses X11 and Xft. For best results, use the Windows fonts. You may need to purchase `symbol.ttf`, but the Core Fonts for the Web are available for non-commercial use.

  BeagleBoard/Bone is cross-compiled using CodeSourcery's free software version of `gcc` running on GNU/Linux or Windows under Cygwin.

Before compiling XXICC, you may need to install software and/or libraries to prepare your environment.

1.1 Preparing to compile the GNU/Linux version

Install Xft library `libXft.so` if it is not already present. In the author's version of Ubuntu, the file is in `/usr/lib/i386-linux-gnu`. I downloaded Xft from `ubuntu.com` using Synaptic by selecting both `libxft2` and `libxft-dev`. I believe the former is for running Xft by iself, while the latter is for compiling and linking programs (such as XXICC) which use Xft. Synaptic automatically took care of loading other libraries needed by Xft.

In the author's version of Debian "Squeeze" on the Raspberry Pi, `libXft.so` was already present in `/usr/lib`. However, the include files for X11 and Xft were not, so he had to give the command "`sudo apt-get install libxft-dev`" to get those files from the Debian repository. You may need to give the command "`sudo apt-get update`" first to get "`apt-get install`" to work.

If you're not sure whether Xft is loaded, try compiling and/or running XXICC and see if error messages indicate that Xft include and/or library files are missing.

1.2 Preparing to compile using Cygwin

Install Cygwin from www.cygwin.com. When you download it, be sure to select the programs and files needed for compiling C programs (`gcc`, `make`, etc.) and the library files for X windows software development. These should include Xft.

1.3 Preparing to compile the BeagleBoard version

Install CodeSourcery's free software version of `gcc` for ARM. There are several versions: for Ångström you need the GNU/Linux version `arm-none-linux-gnueabi`. XXICC 0.0d - 0.0f compiled successfully using "Sourcery G++ Lite 2011.03-41 for ARM GNU/Linux".

The BeagleBoard version of XXICC (`xxibea`) uses shared object (`.so`) libraries for the C library, X11, and Xft. The `.so` files are not in `xxibea` itself: they are in the Ångström file system on the BeagleBoard. However, CodeSourcery needs to read them when it links `xxibea`. For the C library, I have found that it

works to let CodeSourcery use its own copies of the C libraries for linking. When you run it on the BeagleBoard, `xxibea` finds the equivalent Ångström libraries and it's happy. However, CodeSourcery does not have include and library files for X11 and Xft. For include files, I added symbolic links to CodeSourcery's `include` directory on the PC. For `libX11` and `libXft`, I have a copy of Ångström's file system on my PC and I define `RTdir` in `Makefile` to point to the appropriate `/usr/lib` directory in the file system copy. For details, search for `Beagle` in `Makefile`. You will probably have to make some changes, e.g., changing `RTdir` to point to where you loaded a copy of the Ångström libraries.

1.4 Procedure for compiling XXICC

You are now ready to compile XXICC. Here is the procedure, which is the same for GNU/Linux PCs and Cygwin on a Windows machine.

1. Open a terminal window.

2. Create a directory for XXICC, which we'll refer to as `XXICC`.

3. Copy the desired release file -- e.g., `code00e.zip` -- to `XXICC` and expand it.

4. Give the command "`make T=`*xxx*" in the `XXICC` directory. This will compile XXICC for target *xxx*, which can be:

   - `T=XGL`: compile XXICC for X11/GNU/Linux. The resulting executable `xxicc` can then be run from a terminal window. You must create subdirectory `XXICC/xxicc.od` for object modules.

     You may need to add the "`-waa`" and/or "`-wad`" options to work around problems with arcs/circles and/or diagonal lines. Leave them out for now and see §4 for more information.

   - `T=Win32`: compile XXICC for Win32, using `cygwin1.dll` for C library functions. The resulting executable `xxicw.exe` can be run from a Cygwin window or from anywhere else in Windows provided that `PATH` can find `cygwin1.dll`. You must create subdirectory `XXICC/cygw32.od` for object modules.

   - `T=CygX11`: compile XXICC for Cygwin's X Windows. The resulting executable `xxicx.exe` can be run from a Cygwin `xterm` window. You must create subdirectory `XXICC/cygx11.od` for object modules.

   - `T=Beagle`: cross-compile XXICC for the ARM-based BeagleBoard under Ångström GNU/Linux. The resulting executable `xxibea` can be copied to a BeagleBoard and run under Ångström. You must create a subdirectory `XXICC/beagle.od` for object modules. You will probably need to make minor changes to `Makefile` to tell `gcc` where to look for `.so` files.

   "`make T=`*xxx*" compiles all the C modules and links them into one of the above executables. We have tried to make `Makefile` as clear as possible so that you can adapt it to other environments.

5. You now have an executable ready to compile GalaxC code. The next step is to compile the XXICC Object Editor in `XXICC` so that you can edit and compile your own GalaxC programs. Do this using the command for your environment:

`xxicc -force xoe`	Execute in a GNU/Linux terminal window.
`xxicw -force xoe`	Execute in a Windows command window.
`xxicx -force xoe`	Execute in a Cygwin or other X11 `xterm` window.
`xxibea -force xoe`	Execute in `XXICC` directory on your BeagleBoard in a terminal window.

These commands assume `XXICC` or the current directory '`.`' is in `PATH`. If it's not, enter `./xxicc` or similar.

XXICC quickly compiles and links all the `.gal` and `.xoe` files needed for XOE. GalaxC object code files have a `.gi` extension.

When XXICC is finished compiling, it displays a file selection dialog so you can open a XOE window to create or edit GalaxC programs or other documents. On Windows, you can also press `ctl-O` in the Output Window to get a file selection dialog.

Once the `.gi` files are created, you can run XXICC with the `xxicc`, `xxicw`, `xxicx`, or `xxibea` command without argument. Currently you must run XXICC in the directory that contains the `.gi` files.

2. Compiling XXICC using Borland C

You can also compile XXICC using Borland C. This uses Borland `make` file `xxibor.mak` instead of `Makefile`, since the syntax is somewhat different.

1. Create a directory for XXICC, which we'll refer to as `XXICC`.

2. Copy the desired release file -- e.g., `code00e.zip` -- to `XXICC` and expand it.

3. Give the command "`make -f xxibor.mak`" in the `XXICC` directory, assuming Borland C executables are found in `PATH` before any other version of `make.exe`. This will compile XXICC into target `xxibor.exe` for Win32, using Borland's `cw32.dll` for C library functions. `xxibor.exe` can be run from anywhere Windows provided that `PATH` can find `cw32.dll`. You must create a subdirectory `XXICC/borland.od` for object modules.

4. You now have an executable ready to compile GalaxC code. The next step is to compile the XXICC Object Editor in `XXICC` so that you can edit and compile your own GalaxC programs. Use the command "`xxibor -force xoe`". XXICC quickly compiles and links all the `.gal` and `.xoe` files needed for XOE. They are listed in `main.c`. GalaxC object code files have a `.gi` extension.

   When XXICC is finished compiling, it displays a file selection dialog so you can open a XOE window to create or edit GalaxC programs or other documents. You can also press `ctl-O` in the Output Window to get a file selection dialog.

Once the `.gi` files are created, you can run XXICC with the `xxibor` command by itself. Currently you must run XXICC in the directory that contains the `.gi` files.

3. Installing XXICC in Binary Form

One of the huge advantages of FLOSS is that you can see the source code and compile it with your own

compiler, minimizing the possibility that you are bringing malware into your system. However, as each compiler has its own quirks, it can be time-consuming to do this. So we may provide pre-compiled versions of `xxicc`, `xxicw`, `xxibea`, and others as appropriate.

XXICC executables rely on library files, as described in §1. Some of these are already present in your operating system, e.g., all tested versions of Windows have `kernel32.dll`. Other binaries such as Xft may not be present by default and you'll have to get them from your OS distributor or elsewhere on the Internet. GPL does not allow redistribution of binaries without their source code, so we don't include library binaries.

If you want to run `xxicw.exe`, you'll need `cygwin1.dll`. You can get this by installing Cygwin on your machine using the instructions at www.cygwin.com. If you don't want to install Cygwin, you can extract just `cygwin1.dll` from a release file at one of the many HTTP mirror sites listed at http://cygwin.com/mirrors.html. Look in release/cygwin (e.g., http://mirrors.kernel.org/sourceware/cygwin/release/cygwin) for a file like `cygwin-1.7.9-1.tar.bz2`. This is release 1.7.9 in the form of a compressed `.tar` (tape archive) file. `cygwin1.dll` is buried in this file in subdirectory `/usr/bin`. You can extract just this file using `tar` or `7-Zip` and put `cygwin1.dll` at some convenient place where `PATH` can find it, or just put it in your `XXICC` directory.

Here is the procedure for installing XXICC in binary form:

1. Create a directory for XXICC, which we'll refer to it as `XXICC`.

2. Copy the desired release file -- e.g., `code00e.zip` -- to `XXICC` and expand it. Even though you don't need the `.c` and `.h` files, you do need the `.gal` and `.xoe` files.

3. Copy the desired XXICC binary executable -- e.g., `exe00e.zip` -- to `XXICC` and expand it. This will expand all released binaries, e.g., `xxicc` (for x86 PCs), `xxicc.raspi`, `xxicw.exe`, and `xxibea`. Delete the ones you don't need. For Raspberry Pi, rename `xxicc.raspi` to `xxicc`.

4. Make sure you have the library files (`cygwin1.dll`, `libXft.so`) and fonts you need for your executable.

5. Run your executable to compile the XXICC Object Editor as in §1.4 step 5 or §2 step 4.

This document will be updated to handle other environments as XXICC evolves. At some point it may also make sense to use the package installation procedures offered by some distributions. At least with our procedures you know what's going on and have a clue as to how to fix problems if they occur.

4. Running XXICC

We have already seen examples of running XXICC, both for compiling XOE and for running XOE. This section describes additional options. The general form for running XXICC is:

`xxicc` *options target*

where `xxicc` is one of the XXICC executables listed in Sections 1 and 2.

File *target* is a source or object file that XXICC should either compile or load. If it has a "`.gi`" extension, XXICC loads it, automatically performing Make. Otherwise, XXICC assumes it's a "`.gal`" or "`.xoe`"

source file and compiles it, stopping after each stage for the user to press `RET` or `ENTER`. This was used early on when debugging the compiler and is not generally useful.

If *target* is "`xoe`", XXICC recompiles all of XOE except for figure editing using a Make procedure. The `-force` option forces recompilation of all XOE files instead of just checking time stamps. It is equivalent to deleting all the `.gi` files before compiling.

If *target* is "`fig`", XXICC recompiles all of XOE plus the files needed for figure editing. As with compiling "`xoe`", XXICC uses a Make procedure and considers the `-force` option.

If there is no target, XXICC loads and runs XOE without checking any time stamps. Use this if you haven't changed XOE and do not want figure editing.

Here are the possible *options*, which can be used in most combinations. They are separated by spaces:

`-l`*`lib`*`.dll`
`-l`*`lib`*`.so`

Include a Windows Dynamic Load Library *`lib`*`.dll` or its GNU/Linux equivalent Shared Object library *`lib`*`.so` when XXICC starts up. XXICC assumes that any `PATH` or equivalent environment variable has been set up correctly to find the library. The default libraries needed for Windows or X11 have already been included and do not need to be listed on the `xxicc` command line.

XXICC does not need to know about *`lib`* when it is compiled and linked at the C level. It loads *`lib`* and looks up function names during GalaxC compilation as it compiles `cdecl` and `stdcall inline` function calls. In fact, it does not check whether *`lib`* even exists until it compiles `inlines`. This is truly dynamic linking.

`-force`

This forces recompilation of all GalaxC code that is needed to compile or load *target*. Otherwise XXICC uses a Make procedure by checking source and object file time stamps. For details, see *Compiling and Running GalaxC Programs.*

`-SC`
`-SE`
`-SU`

These specify the character coding for the Symbol font under X11. This is needed because different Symbol font files may encode characters differently. Usually the correct coding is selected by `x11gswim.c`, but if it guesses incorrectly you can change it with one of these commands. `-SC` selects the `CP1252` Latin-1 encoding used by the other fonts, which just maps 8-bit Symbol characters to themselves. `-SU` converts Symbol characters to Unicode and works on the author's Ubuntu GNU/Linux PC. `-SE` is an empirical encoding which seems to work for `symbol.ttf` with Cygwin/X11 and BeagleBoard.

You can check the coding using `showfont.gal` to display all the characters of the Symbol or other font.

`-nocal`
`-nogm`

These are special Make options primarily for the Borland C version. For details, see *Compiling and*

Running GalaxC Programs.

`-rtcw`

XOE allows two text orientations: horizontal and rotated 90°. To minimize the amount of head tilting needed to read rotated text, XOE rotates all text either clockwise or counter-clockwise. This also minimizes the number of extra fonts that need to be created. The default rotation is counter-clockwise; for clockwise, include the `-rtcw` option when starting XXICC.

`-waa`
`-wad`

The Ubuntu 11.10 version of X Windows sometimes has trouble with arc/circles and diagonal lines. The actual misbehavior depends on whether you're running with a GNOME, Ubuntu (Unity), or Ubuntu 2D (Unity 2D) desktop. Debian "Squeeze" on the Raspberry Pi works fine, so I suspect the problem is an interaction between the X11 server and GNOME or Unity.

0.0f and later include work-arounds for the arc/circle (`-waa`) and diagonal line (`-wad`) problems. First try running XXICC without the work-arounds. If you see graphics behaving strangely add one or both options. The work-arounds only affect the compilation of `x11api.gal`, so you only need to recompile that file. For example, to incorporate both work-arounds:

```
touch x11api.gal
xxicc -waa -wad xoe
```

After you have recompiled `x11api.gal`, you don't need to specify the `-waa` or `-wad` options. However, it doesn't hurt anything to include them since they do not affect any files other than `x11api.gi`.