

# Installing GPUTILS & SDCC

While you probably need a PC to burn the MBHP-Loader into the PIC, you can stick to your beloved Macintosh while coding your C-Applications!

And more than that: you are able to use Apple's powerful Xcode IDE and don't leave that except for sending the .syx file! You can even debug your MIOS C-Applications with the [MIOS C Simulator - Debugger](#) - although this is in a very early development state, please contribute code! *audiocommander*.

We're starting with SDCC-Installation, then we'll setup a new Xcode2 project:

The first step in this process is to install SDCC. If you have already SDCC running and calling "sdcc -v" in terminal gives you the info you're running 2.5. or higher, you have the most complicated step already completed!

## How to install a library from the terminal

If you've never installed \*nix libs or codes, don't worry: Either you can fetch a binary (that's easy if available for your system, just copy it to usr/local/bin) or you have to fetch the source and compile by yourself. In most cases this should be also ok, but if there are missing libraries it can be a bit tricky...

The "normal" way to install and compile packages you loaded (eg from sourceforge) is this:

Open the Terminal and type "cd" (change directory). Now drag the folder containing the unpacked source directory onto the terminal window to save yourself from typing the path. Close this step with pressing enter. You should now be in the right directory (hopefully!)

Now you have to type three commands:

- ./config
- make
- make install

sometimes you have to call these with "sudo", esp. "sudo make install".

If you have problems compiling and installing SDCC, you might lack a library.

Here is what you should have installed (in this particularly order)

## 1. GPUTILS

[Ben](#) has released a [precompiled GPUTILS installer](#). Thanks for that!


Check out the rest of his site, he has some useful infos about MicroChip Dev on OSX

## 2. SDCC

The SDCC sourcecode is available here: <http://sdcc.sourceforge.net/index.php#Download> (latest snapshot – don't use the installer for SDCC 2.4.0 for Mac, it won't work; you'll need 2.5.x or higher)

If you have trouble compiling the sdcc source (and you will have trouble, believe me), you might better try a binary precompiled release. I've built a binary installer:

<http://www.audiocommander.de/downloads/midibox/sdcc254-installer-osx.mpkg.tgz> **(MAC OS X**

**Only!)** Now, is that cool? 

A test in the terminal with “sdcc -v” should now show the proper version:

```
mymac:~ user$ sdcc -v
SDCC : mcs51/gbz80/z80/avr/ds390/pic16/pic14/TININative/xa51/ds400/hc08
2.5.4 #1189 (Dec 30 2005) (UNIX)
```

If you get instead a return “command not found”, you have to add usr/local/bin to your path environment:

```
mymac:~ user$ echo $PATH
```

now you should something like this: “/bin:/sbin:/usr/bin:/usr/sbin”

copy this line (the output from your terminal window, not the one above). now type & paste:

```
mymac:~ user$ PATH=:<paste here>:/usr/local/bin
```

### 3. If there are errors

You might need additional libraries. Go and load [Fink](#) and [FinkCommander](#)!

Then open FinkCommander and install these libraries: → gettext-0.14.5, gputils-0.13.2, libiconv-1.9.2\_1, libtool-1.5.22\_1 And don't blame me or anyone else if this breaks up something. It's your own responsibility!

If you have trouble installing SDCC on your mac, post your questions/problems here:

<http://www.midibox.org/forum/index.php?topic=6527.0>

## Let's open Xcode2

That was pretty easy, wasn't it?

Now, let's go on to real cool stuff and switch to Xcode.

I presume, you have downloaded the MIOS-C-Skeleton, unpacked it in an appropriate folder and you're ready to start coding your own cheezy MIOS-App.

## Creating a new project

Create a new, empty project from the File-Menu. Select your MIOS-application folder.

Now drag your files into the project. This should be mainly main.h and main.c, but it won't hurt if you add all the files in your directory.

Edit the file "MAKEFILE.SPEC" to your needs. That means, if you have additional classes, you should add them to "MK\_ADD\_OBJ myfile.c"

## Setting up the make target

Because things change a lot, we're making it easy: we set up a "makefile"-target, so with a press of button, your new makefile is generated!

Note, that you can skip this step if your makefile will never change. Nevertheless, the procedure is the same as setting up your main target, so you should read it anyway :)

- go to "Project > New Target" and choose "Shell Script Target". Name the target "Makefile"
- open the Target-Tree, open the target "Makefile" and double-click on "Run Script"
- you'll find now:

```
# shell script goes here  
exit 0
```

type this:

```
# shell script goes here  
  
PATH=/usr/local/bin:$PATH  
  
cd $SRCROOT/tools  
./mkmk.pl $SRCROOT/MAKEFILE.SPEC  
  
rm Makefile.bat  
echo Makefile.bat deleted \ (blame Windows®\ ).  
mv -f Makefile ../  
echo Makefile replaced.  
echo Success.  
  
exit 0
```

Don't forget to close the panel or switch forth and back to the next window item ("Comments"), because sometimes the changes are not recognized.

You're done!

Once you select "makefile" as your primary target and press "build", the perl tool ./mkmk.pl is invoked, generates your Makefile, deletes the unnecessary "makefile.bat" and moves the new makefile in your directory root.

If you want to, you can now add “\$(SRCROOT)/MAKEFILE.SPEC” to the “Input Files” and “\$(SRCROOT)/Makefile” to the “Output Files”. Then Xcode2 checks if any of the input files is newer than the output files and will start the build only if necessary.

## Setting up the primary application target

Now this is basically the same as before. Repeat the steps above, but call your target like your application. The bash script would look like this in the “Run Script”:

```
# shell script goes here

PATH=/usr/local/bin:$PATH

cd $SRCROOT
make

exit 0
```

As you see, the only trick is, to invoke the makefile.

Don't forget to choose your new current target, your application. Otherwise only your makefile is generated again :)

Easy, isn't it?

## Debugging with Xcode

go to the [mios\\_c\\_simulator\\_-\\_debugger](#) page and grab the debug sources. Note, that these are far from being complete. If you add functions in your own project, please update the files here in the wiki!

Add the debug-files to your Xcode project, and adapt what needs to be adapted (defines, includes etc...).

Now add a new target, choose “Carbon Shell Tool” and name it “MIOS\_Debug” or whatever you like.

Now you have to add the Carbon Framework (Right Mouse on the Folder “Libraries”, Add Framework, Search for Carbon).

At this point you just have to make sure, that “debug.c” is the only file that is included for this target. If you open the debug target, you see “Compile Sources (1)”. If there are more files than “debug.c”, delete them from this target.

**Important:** If you debug your C-Application, you need to make inputs from the command-line, which can't be done from the debugger-console. Launch the “Standard I/O” Window from the “Debug” Menu. This console-window is only available in debug-mode!

If you have questions, feel free to post them here:

<http://www.midibox.org/forum/index.php?topic=6527.0>

From:

<http://www.midibox.org/dokuwiki/> - **MIDIbox**

Permanent link:

[http://www.midibox.org/dokuwiki/doku.php?id=how\\_to\\_use\\_xcode2\\_as\\_ide\\_on\\_a\\_mac&rev=1152818790](http://www.midibox.org/dokuwiki/doku.php?id=how_to_use_xcode2_as_ide_on_a_mac&rev=1152818790)

Last update: **2006/10/15 09:35**

