How to set up the toolchain for coding MIOS32 apps with OS X

steps:

0- install Xcode
1- about the terminal
2- get the mios32 files from the repository server
3- install the MIOS32 toolchain
4- configure the paths
5- other derivatives

1- install Xcode

Xcode is a toolchain from Apple with a powerful IDE, which can be downloaded for free in the App Store.

Even if you don't use the source code editor of this IDE, you have to install Xcode in order get access to the command line tools, such as “make”

1- about the terminal

Most of the things now will be done with the Terminal, if you don't know anything about it here are the basic commands you need:
Launch it from your “utilities” folder.
After each command, press “enter” to validate it.

Type “pwd” to show the actual path of where you are.
Type “ls” to list all the files and directories in your current directory
Type “cd aDirectoryName” to go inside that directory
Type “cd aPath” to go to a particular path
Type “cd ..” to go to the parent directory
Type “cd ~” to go to your home directory

A nice shortcut if you want to go to a particular folder without having to type the full path is to type “cd ” in the terminal, then open the parent directory of the one you want to go in with the finder, and drag and drop the folder icon inside the terminal. It's path will appear by magic, press “enter”.

2- get the files from the repository server

TK proposes to use the “svn” command inside a Terminal. This command is already part of the Xcode toolchain, so that no further installations are required. A nice installation and usage guide can be found under http://www.rubyrobot.org/tutorial/subversion-with-mac-os-x

After the installation just create a svn directory:

```
mkdir svn
```

thereafter checkout the repository with:

```
svn co svn://svnmios.midibox.org/mios32
```

done! Applications will be located under svn/mios32/trunk/apps

If you prefer a GUI (but TK states that this will be more confusing), install a client like “svnX”, launch it, in the “repositories” window click on the “+” sign, give a name to the repository, put the repository path in the path field: svn://svnmios.midibox.org/mios32 and double clic on the repository name to get a connexion.
Then you are able to drag and drop the “trunk” folder from the server to wherever you want on your computer. I suggest that you create an “snv” folder in your home directory, then a “mios32” folder and to put the dragged “trunk” folder inside that folder. Now the Unix path of your mios32 files is “~/svn/mios32/trunk” (“~” stands for “home directory”).

You can also do this from Xcode itself:
In the “SCM” menu choose “configure SCM repositories”.
Under the repositories list clic on the “+” button to add a repository. Give it a name (mios32 for example), and change the “SCM system” to “Subversion”.
Then in the “path” field put the repository server path: svn://svnmios.midibox.org/mios32 and clic “OK”.
In the “SCM” menu choose “repositories”, you now have the repository window where you can access to the mios32 files.

3- install the MIOS32 toolchain

The MIDIbox community have created a modified GNU Compiler Collection, ready for use with the ARM Cortex M3 platform. The toolchain contains all of the tools required to build MIOS32 applications including GCC and NEWLIB.

The Toolchain has currently been compiled for Windows (2000+), Mac OS X (Leopard or newer) and Linux x86 (built on Ubuntu).

The download directory for the toolchain is http://www.midibox.org/mios32_toolchain

Download the latest “macos” file and unzip into your favorite directory, If you unzip into /usr/local the toolchain will create /usr/local/mios32_toolchain and various sub-directories containing the toolchain files.

You now need to set the path variable for the toolchain bin directory (/usr/local/mios32_toolchain/bin), see the next section about this

4- set up the paths variables

For those who don't know anything about Unix like me a few days ago, here's a quick explanation:
In Unix we call a shell the piece of software that provides an interface for users. For example, when you use the terminal in OS X, you are talking to the computer through the “bash” shell.
Your compiler also uses the shell to access to the different files involved. Thus, it has to know where to find these files. For this, we set up variables in the shell. For example, we will put the path of your

mios32 folder in the variable “MIOS32_PATH” so that when the compiler will want to check inside your “mios32” folder, it will just call this variable.

Setting up path variables in the shell is quite easy: in a terminal window just type
```
export VARIABLE_NAME=something
```
and if you want to check to what the variable has been assigned, type:
```
echo $VARIABLE_NAME
```
this will print what's inside the variable.
For example, type
```
export MIOS32_PATH=~/svn/mios32/trunk
```
to set up the mios32 folder path, then type
```
echo $MIOS32_PATH
```
to check that the path variable has been assigned.

In our case, we need to set up the path variable for the MIOS32 toolchain for STM32: type in the terminal
```
export PATH=$PATH:/usr/local/mios32_toolchain/bin
```
and then we need to set up variables for mios32 itself:
```
export MIOS32_PATH=~/svn/mios32/trunk
export MIOS32_BIN_PATH=$MIOS32_PATH/bin
export MIOS32_GCC_PREFIX=arm-none-eabi
export MIOS32_FAMILY=STM32F10x
export MIOS32_PROCESSOR=STM32F103RE
export MIOS32_BOARD=MBHP_CORE_STM32
export MIOS32_LCD=universal
```
Here I considered that you have put all the files downloaded from the svn server in the directory “~/svn/mios32/trunk” but up to you to put them elsewhere if you want and change the “MIOS32_PATH” variable accordingly.

Now everything is set up to compile properly your projects. Test if everything is OK, open a terminal window, type
```
cd $MIOS32_PATH/apps/tutorials/001_forwarding_midi
```
to go to the “forwarding_midi” tutorial folder and type
```
make -s
```
you should have a return looking like this:
```
————————————————————-
Application successfully built for:
Processor: STM32F103RE
Family: STM32F10x
Board: MBHP_CORE_STM32
LCD: clcd
————————————————————-
```
and with the finder, go to this tutorial folder, you should have now a .hex file uploadable to your core32 with MIOS Studio.

The annoying point there is that the variables you set up in the shell just disappear when you shut down or log off your computer. That means you have to set them up again when you power on your computer again. There's a way to avoid this annoying stuff:
The shell will always read particular files before starting up. If you put these variables there they will always be set up. I didn't really understood if the proper file was “~/.bash_profile” “~/.bash_login” “~/.profile” or “/etc/profile” to do that, it looks like they all do the job.
Moreover, all these files are hidden from the finder, so the best way to edit them is to open a terminal

window and edit them with the “pico” editor.
If you don't understand what's going on, here's what I did:

in a terminal window, type:
`pico ~/.profile`
then copy and paste these lines inside the pico editor:

```
export PATH=$PATH:/usr/local/mios32_toolchain/bin
export MIOS32_PATH=~/svn/mios32/trunk
export MIOS32_BIN_PATH=$MIOS32_PATH/bin
export MIOS32_GCC_PREFIX=arm-none-eabi
export MIOS32_FAMILY=STM32F10x
export MIOS32_PROCESSOR=STM32F103RE
export MIOS32_BOARD=MBHP_CORE_STM32
export MIOS32_LCD=universal
echo MIOS32 variables initialized.
```

do “ctrl-O” to save the file
do “ctrl-X” to exit

Once you open a new terminal, the variables will be set automatically, and you should see the “MIOS32 variables initialized” message.
It isn't required to log off or shut down your computer to test this.
Open a terminal window and check if the variables are all set up by typing:
`echo $MIOS32_PATH` for example.
If you don't see the messages (and MIOS32 variables don't exist), that's probably because a .bash_login or .bash_profile already exists (edit only an already existing file - of none exists, use .profile).
Useful info about bash and initialisation files here:
http://johnnywey.wordpress.com/2008/04/17/fixing-bash-profile-in-os-x/
and here:
http://macdevcenter.com/pub/a/mac/2004/02/24/bash.html

5- other derivatives

For STM32F1 running on a MBHP_CORE_STM32 board use following setup:

```
export MIOS32_PATH=~/svn/mios32/trunk
export MIOS32_BIN_PATH=$MIOS32_PATH/bin
export MIOS32_GCC_PREFIX=arm-none-eabi
export MIOS32_FAMILY=STM32F10x
export MIOS32_PROCESSOR=STM32F103RE
export MIOS32_BOARD=MBHP_CORE_STM32
export MIOS32_LCD=universal
```

For LPC17 running on a MBHP_CORE_LPC17 (or LPC1769 based LPCXPRESSO) board use following setup:

```
export MIOS32_PATH=~/svn/mios32/trunk
export MIOS32_BIN_PATH=$MIOS32_PATH/bin
export MIOS32_GCC_PREFIX=arm-none-eabi
```

```
export MIOS32_FAMILY=LPC17xx
export MIOS32_PROCESSOR=LPC1769
export MIOS32_BOARD=MBHP_CORE_LPC17
export MIOS32_LCD=universal
```

For STM32F4 running on a MBHP_CORE_STM32F4 board use following setup:

```
export MIOS32_PATH=~/svn/mios32/trunk
export MIOS32_BIN_PATH=$MIOS_PATH/bin
export MIOS32_GCC_PREFIX=arm-none-eabi
export MIOS32_FAMILY=STM32F4xx
export MIOS32_PROCESSOR=STM32F407VG
export MIOS32_BOARD=MBHP_CORE_STM32F4
export MIOS32_LCD=universal
```