Toolchain Code::Blocks IDE Setup

```
for MIOS application development on Windows
```

The technical jargon:
This is the *Code::Blocks IDE* section in a walkthrough on the process of developing MIOS Applications, on Microsoft Windows XP, primarily in C, utilising the MIOS C Wrapper but also in ASM. The IDE platform used will be Code::Blocks IDE or NotePad++ and the applications will be built for the MIDIBox Hardware Platform (MBHP) stuffed with a Microchip PIC18F. C code will be compiled with SDCC, and assembled with GPUtils, using MSYS to generate and run DOS-Console-based makefile scripts, and MIOS Studio will be used for debugging on MBHP. For PC emulation for debugging purposes, you may also use AC-Sim (AudioCommander's CSimulator) compiled with GCC from MinGW.

Enough of that.
You should have come here after the Core Tools setup. This part of the tutorial can actually be skipped completely if you wish. It contains setup instructions for an optional component of the toolchain. If you are going to be coding your apps only in C, then you only need to install this one last application, and your toolchain is complete. If you are not sure if this is you, then it isn't.... Read on... You can always come back to this step later if you need to.

- Code::Blocks - It's an IDE. If you don't know what that entails, then just trust me that you will like it ;)
- GDB - The GNU De-Bugger - If you want to make use of AC-Sim (you will have decided this previously in this tutorial) you should install the GDB to assist you in the debugging process. If you are not going to use Code::Blocks, your IDE may use a different debugger, so you won't need this.

If you are not sure, and you have a few meg of diskspace to spare, I would recommend that you install this now, as Code::Blocks will automatically detect it's presence, which makes setup simple.

GDB (GNU DeBugger)

GDB is used to debug apps and is a must-have for MIOS development.

- Download the installer. This Link is for GDB 6.3.2, You may find a newer version on the SourceForge page
- Install, taking the same default install path as MinGW (see Part 2).

Code::Blocks IDE

Code::Blocks is a free yet powerful IDE. Highly recommended on Windows.

Install

The full Code::Blocks installer package is *not recommended* . We need to manually install and configure GDB (GNU DeBugger) and MinGW, which contains GCC - the GNU C Compiler. **It is strongly recommended that you install these PRIOR to installing Code::Blocks** If you've been following the three-part tutorial, then you're about right.

Code::Blocks

For our purposes, CodeBlocks latest nightly build version should be installed.

If there is an existing version installed, then delete the share\ dir, for example C:\Program Files\CodeBlocks\share.* before upgrading*

- Get link for Nightly Build from the CB Forum
- Unzip downloaded file to Program Directory, for example C:\Program Files\CodeBlocks\, and overwrite where prompted.

 Note that these links sometimes change! They are usually (always!) in the first post of the thread for the Nightly Build. Get them from there!!

- Download a patched wxwidgets dll  and extract to the CodeBlocks Program Directory, for example C:\Program Files\CodeBlocks\
- Start Code:Blocks. If it generates an error about mingwm10.dll missing, ensure the above path variables are set. If you are SURE that they are and the file is missing, download it from here  and copy the .dll to your program directory
- Code:Blocks will detect and configure installed compilers. It should find 'GNU GCC Compiler' if you installed MingW, and 'SDCC Compiler' if you installed SDCC (WHICH YOU SHOULD HAVE!). Accept defaults.

Configure

In order to use the GDB (GNU DeBugger) debugging features of Code::Blocks with AC-Sim (AudioCommander's C Simulator) you need to configure GCC to produce debugging symbols when compiling as follows:

- Select the menu 'Settings... Compiler and Debugger...'
- Select 'Global Compiler Settings' in the pane on the left.
- In the box labelled 'Selected compiler', select 'GNU GCC Compiler'
- Select the tab labelled 'Compiler Settings' and then the tab labelled 'Compiler Flags'
- From the dropdown box labelled 'Categories:', select <All categories>.
- Tick the box labelled 'Produce debugging symbols [-g]' (Sometimes there's no box, just click in the space to the left of the text and you'll see the tick.
- Click OK.

- Again, select the menu 'Settings... Compiler and Debugger...'
- Select 'Global Compiler Settings' in the pane on the left.
- Select the 'Toolchain executables' tab.
- Make sure that 'Linker for dynamic libs' is set to 'mingw32-gcc.exe' (and **not** 'mingw32-g++.exe').
- Click OK.

This should be all that is needed to have the application itself ready to go!

Project Setup

Full instructions are below, but you may jump to the required section based on your requirements:

Project Setup - New Application

If you are creating a new application, you can simply download the MIOS SDCC Skeleton Application Template, extract it to your template directory, and start with that, and none of the below steps are necessary. This zip file does not change any of the functionality of the normal SDCC Skeleton App, so you can use it as normal, as well as with C::B. Here is a walkthrough:

- Download the MIOS SDCC Skeleton Application Template
- Extract the contents of the zip to your user template directory. It will be C:\Documents and Settings\<your username>\Application Data\codeblocks\UserTemplates
- In Code::Blocks, Select 'File... New Project'
- Select the tab labelled 'User Templates', and double-click the project named 'CB SDCC Skeleton'
- Writing your code is up to you but once you're done, jump to the section on Compiling and Debugging for either AC-SIM or MBHP

You may want to visit the AC-Sim page to update your Simulator files!

Project Setup - Existing Application or Skeleton Creation

A C::B Project can be setup from either an existing MIOS C Application or the SDCC_Skeleton App as follows. You should have your application/skeleton in a dedicated folder.

Create Empty Project

- Copy all files from either the C Skeleton, or your app, to a directory where you would like to store the project. *
- Start Code::Blocks
- Open a new project ('File... New Project')
- Click 'Console Application' from the list. Click 'Go'.
- In the box labelled 'Project Title', give your app a name. For 'Project Path', browse to the directory containing your MIOS Application or Skeleton *, and add the name of your Project. Click 'Next'.
- Ensure 'GNU GCC Compiler' is selected. Click 'Next'
- If you are prompted to select a language to use, select 'C' (not C++). Click 'Finish'.

Add Application files to project

- When you first create the project, C::B will automatically create a 'Hello World' application. This main.c needs to be deleted. Right-Click it, and choose 'Remove from project'.
- Now open up Widows Explorer (aka 'My Computer'), browse to your project directory* and delete that file for good. It would get in the way when you...
- Move all of your skeleton files into the C::B Project directory* (Where you see a *.CBP file you specified earlier).
- Create an empty text file in that directory, named “main.h”. You will need this file for the AC-Sim setup later on, and probably in your own application anyway. It won't hurt.
- Select 'Project... Add Files', Select All of the source code files in your new Skeleton directory*, Click 'Open'.
- If you are prompted to select which build target should be used, click 'Select All' and click OK until the files are all added.
- Double-click the tree branch in the far left pane labelled “Sources”. you should see your main.c file in there. Double-click it to open it for editing.
- Immediately after the copyright notice, insert this code:

```
#include "main.h"
```

- Right Click the file's name in the tab bar, and select 'Close'. When you are prompted to save the file, save it.

Configure Build Targets

AC-SIM Build Target

You may skip this section if you do not need to use the simulator

- Ensure AC-SIM files are added to the project. Follow the instructions at the AC-Sim WIKI page Setup Guide
- Select 'Project... Properties', Select the 'Build Targets' tab, you should see two targets preconfigured ('Debug' and 'Release'. Debug will be used for AC-Sim, and Release for MBHP (PIC))
- Select the target 'Debug'
- In the box labelled 'Selected build target options', select 'Console Application' from the drop-down list labelled 'Type'
- In the box labelled 'Selected build target files', Click 'Toggle Checkmarks' to uncheck all of the files. Now, Tick ONLY 'ACSim_Console.c', 'ACSim_mios.c' and 'ACSim_Toolbox.c'.

MBHP Build Target

You should always perform the following sections to build the application for MBHP(your MIDIbox!)

- Returning to 'Project... Properties', 'Build Targets' tab, select the build target 'Release'
- In the box labelled 'Selected build target options', select 'Commands Only' from the drop-down list labelled 'Type'
- In the box labelled 'Selected build target files', Click 'Toggle Checkmarks' to uncheck all of the files. Now, Tick ONLY 'main.c' (even if you have other files to compile. That's done by the make script)
- Click 'OK' to close project properties

- Select 'Project... Properties', Select the 'Targets' tab (Yes, again. It is necessary to apply the previous settings first)
- Select target 'Release', click 'Build Options'
- In the box labelled 'Selected Compiler' Select 'SDCC Compiler' from the dropdown list (until this point both targets will be using C::B's default of GCC), Click 'OK' and return to the 'Targets' tab. If you receive a warning recommending you to “fully rebuild...”, that is normal. Click OK.
- In the box labelled 'Selected build target files' (bottom right), select file 'main.c' and then click 'Selected File Properties'
- Select the 'Advanced' tab, in the box labelled 'Custom Build', Select 'SDCC Compiler' from the dropdown list labelled 'For this compiler'
- Tick 'Use Custom Command to build this file'
- Type into the textbox “make”
- Click OK
- Click OK

At this point, I highly recommend selecting 'File... Save Project as user template', then you can use it as a quick starting point for all your awesome projects :)

AC-Sim Simulator

Setup

Follow the instructions at the AC-Sim WIKI page

Compile

Select the 'Debug' target, Click 'Build', and you're done!

Debug

Follow the instructions at the AC-Sim WIKI page

MBHP

Compile

Select the 'Release' target, Click 'Build', and you're done!

Debug

Well, that's up to you!

IDE Setup Complete

If you made it this far, you have done all you need to edit C based apps, including using AC-Sim. If you're going to code in ASM, or you're unsure, you may like to install another IDE like NotePad++. You won't break anything. :)