Developing MIOS Applications with Code::Blocks

This is a walkthrough on the process of developing MIOS Applications primarily in C but also in ASM utilising the MIOS C Wrapper. The platform used will be Code::Blocks IDE onMicrosoft Windows XP, and the applications will be built for both AC-Sim (AudioCommander's C Simulator) compiled with GCC for MinGW, and for the MIDIBox Hardware Platform (MBHP) compiled for PIC18F with SDCC, assembled with GPUtils, using ActiveState ActivePerl to generate, and parse files for, the DOS-Console-based make batch file scripts, and MIOS Studio will be used for debugging on MBHP.

Please don't be put off by all that mumbo jumbo. If you're totally newbie then this document is also for you :) I've included everything you need to do, step-by-step, for everything except for actually coding the app. I hope this will not only make it easier for newbies to get started, but for ASM coders like myself to transfer their skills into C development (which seems to be the future of MIOS) or as a reference point for the experienced. In order to do this, I've included as much information as possible, but I have created links to jump through the document for those who don't need all that detail.

I hope you find this document helpful and that you will feel free to make suggestions or criticism or corrections or any kind of modifications as you see fit :)

Install and Configure Applications

GNU PIC Utilities (GPUtils)

- Download gputils-win32
- Install the complete application using the 'Complete' option.
- When the installer asks you, click 'Yes' to add the installation directory to the PATH environment variable.

ActiveState ActivePerl

- Download ActivePerl
- Install using defaults, I recommend changing the path to c:\program files\activestate\perl
- If you are prompted to restart your computer, then do so.

Small Device C Compiler (SDCC)

- Download SDCC
- Install using defaults

Environment Variables

Normally, the necessary directories are added to you PATH environment variable during the above installations. Sometimes, they aren't. Here's how to check it:

- Minimise any open windows so that you can see the desktop.
- Right-click on 'My Computer', select 'Properties'.
- Click on the 'Advanced' tab, then click 'Environment Variables'
- Under either User or System variables, select the variable 'PATH' and click 'Edit'
- Add these entries if they don't exist:
  - ;C:\Program Files\ActiveState\PERL\bin\
  - ;C:\Program Files\SDCC\bin
  - ;C:\Program Files\gputils\bin

Sun Java J2SE

- Download J2SE
- Install with defaults or whatever meets your requirements. I recommend disabling automatic updates.

Code::Blocks IDE

Install

- Download Code::Blocks RC2 Full Installer package [See below]
- Install using defaults

Full Installer package is *highly* recommended unless you're an expert. This will also install and configure GDB (GNU DeBugger) and MinGW, which contains GCC - the GNU C Compiler. GDB is used to debug your application in the simulator, which is compiled with GCC.

<trivia> GCC actually stands for GNU Compiler Collection. It supports heaps of other stuff so the name changed.</trivia>

Upgrade

For our purposes, CodeBlocks should be updated to the latest nightly build. Features required to build the applications are unavailable in version 1.0RC2 which has been installed above.

- Download a newer wxwidgets dll and extract to the CodeBlocks Program Directory, for example C:\Program Files\CodeBlocks\
- Get link for Nightly Build from the CB Forum
- Delete share\ dir, for example C:\Program Files\CodeBlocks\share*.*
- Unzip downloaded file to Program Directory, for example C:\Program Files\CodeBlocks\, and

overwrite where prompted.
- Start Code:Blocks, it will detect and configure installed compilers. 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' and then the tab labelled 'Compiler Flags'
- From the dropdown box labelled 'Categories:', select <All categories>
- Tick the box labelled 'Produce debugging symbols [-g]'

This should be all that is needed to have all the applications 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 'MB-ACSim'
- 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

Project Setup - Existing Application

I plan to write a small application which will automatically convert a MIOS application (as downloaded from UcApps) into a Code::Blocks Project complete with AC-Sim sources for pre-MBHP debugging. In the meantime, the 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

- Start Code::Blocks
- Open a new project ('File... New Project')
- Tick the checkbox 'Show File-based Templates'
- Click 'SDCC Application' from the list. You may need to scroll down to see it. If it is not there, ensure you have performed the previous step
- In the box labelled 'Project Path and Name', browse to the directory containing your MIOS Application or Skeleton, and add the name of your Project
- In the box labelled 'Options', tick the box labelled 'Do not create any files'
- Click 'OK'
- Click 'Yes' on dialog(s) stating that the directory already exists

Add Application files to project

- Select 'Project... Add Files', Select All of the files in your Project directory, Click 'Add'

Configure Build Targets

AC-SIM Build Target

Skip this section if you do not need to use the simulator

- Select 'Project... Properties', Select the 'Targets' tab, Click 'Add'
- Name the target 'MB-ACSim', and Click OK
- 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', Tick ONLY 'debug.c'

MBHP Build Target

Perform the following sections to build the application for MBHP

- Returning to 'Project... Properties', 'Targets' tab, Click 'Add'
- Name the target 'MBHP', and Click OK
- 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', Tick ONLY 'main.c'
- 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 'Default', click 'Delete', Click 'Yes' to confirm deletion
- select target 'MBHP', click 'Build Options'
- In the box labelled 'Selected Compiler' Select 'SDCC Compiler' from the dropdown list, Click 'OK' (until this point both targets will be using C::B's default of GCC)
- In the box labelled 'Selected build target files', 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.bat”
- Click OK
- Click OK

MIOS-Specific SDCC Libraries

TK says: If multiplications, divisions, pointer operations, etc. are used in the .c code, the linker may fail due to missing functions, which are part of the “libsdcc.lib” library. The error message will be:

```
Linking project
lib/libsdcc.lib: No such file or directory
ERROR!
```

The common library for pic16 derivatives is not compatible to MIOS, therefore I've created a special one. Click here for more information

- Download the MIOS-Specific SDCC Libraries
- Create a 'lib' directory within the root directory of your project
- Copy the 'libsdcc.lib' file from the above zip file into this directory
- Open the “project.lkr” file of your project and add following line below the “LIBPATH” entry:

```
FILES lib/libsdcc.lib
```

AC-Sim Simulator

Setup

Follow the instructions at the AC-Sim WIKI page

Compile

Debug

MBHP

Compile

Debug

The following is yet to be documented by me.... upload http://www.midibox.org/mios_studio/ done! Make music!

From:
https://wiki.midibox.org/ - **MIDIbox**

Permanent link:
https://wiki.midibox.org/doku.php?id=stryd_one_codeblocks&rev=1151173394

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