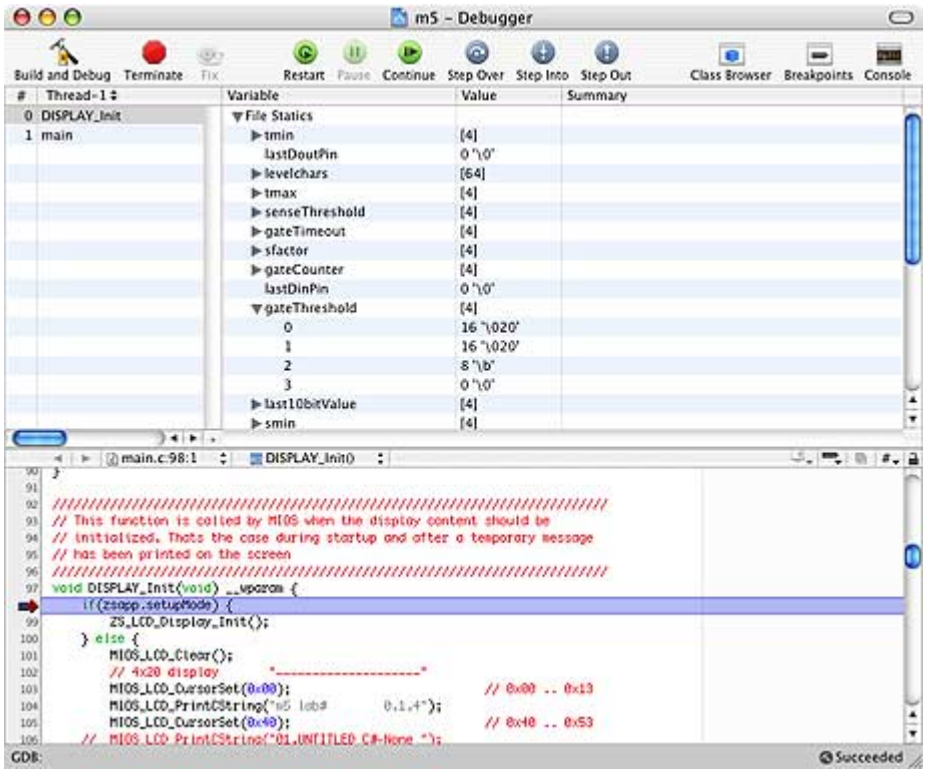
AC-Sim: AudioCommander's Simulator for MIOS Applications. This has been named after it's creator but hopefully all MIDIBoxers will feel free to contribute to and share their code.

Concept

AC-Sim allows you to compile your MIDIBox Application as a command-line application on your Mac or PC (should also run fine with Linux). Input such as knobs, sliders, buttons (ie AIN and DIN) and MIDI Input can be simulated by entering commands, and the output of the LCD and MIDI Out will be displayed in the console. Please refer to the section “Usage (Manual)” to see what inputs are currently accepted and how to use it.

But you'll discover the real strength of the simulator once you are going to use the graphical debugging interface of your IDE → you can inspect all the variables and their contents!

It reduced my number of application uploads dramatically and to be able to use a graphical debugger to watch some variables is quite helpful... Look at that (Pic shows Xcode's GDB-Debugger window):



Isn't that nice?

Overview

The Simulator currently consists of these files:

Source	Contents	Notes
ACSim_console		Adaption required!
ACSim_console.h	Hardware related defines	Change #defines to reflect your settings
ACSim_console.c	Main runloop	Add additional c. source-files here
ACSim_toolbox		*No changes required*
ACSim_toolbox.h	Hexview config	*No changes required*
ACSim_toolbox.c	Helpers like random generators and hex-view	*No changes required*
ACSim_mios		*No changes required*
ACSim_mios.h	pic18f452.h typedefs and global vars	*No changes required*
ACSim_mios.c	MIOS functions for simulation	*No changes required*

To implement the ACSim files into your applicaton, you just have to follow the setup guide to:

- Modify **main.c** and **main.h**
- Adapt your settings (#defines, #includes) in **ACSim_console.h** as required for your debugging purposes.
- If you have multiple sources, also #include your .c-sources in **ACSim_console.c**
- Configure your IDE:
  - Set up a new target (command-line application)
  - Add *ACSim_console.c*, *ACSim_mios.c* and *ACSim_toolbox.c* to the new target, but don't add any other .c-file!
- Build and run

When running/debugging, Main() calls:

- MIOS_Init()
- MIOS_LCD_Init()
- MIOS_Timer() is polled each runloop()

For detailed step-by-step instructions see *Setup Guide* below!

Usage (Manual)

These input commands are currently available:

- (**q**)quit
- (**SPACE**)OK
- (**r**)andom
- (**++**) (**-**)
- (**e**)ncoder(**++**)/(**- - -**)
- (**a**)(*pin,value*)
- (**j**)umper(*pin*)
- (**p**)rogramChange(*PRG,{store}*)

- (**m**)idiMessage(*byte0,byte1,byte2*)
- (**n**)oteOn(*channel,value,velocity*)

Examples:

- Type “q” to exit the runloop
- Type “a2, 560” to call AIN_NotifyChange(pin 2, value 560)
- Type “r” to call AIN_NotifyChange(pin random, value random)
- Type “d4,1” to simulate a button-press on pin 4 (note that 1 means ON in contrast to MIOS_DIN_PinGet(pin) which returns 0 if the circuit is closed!) → calls DIN_NotifyChange(pin, ON/OFF), where 1 (ON) submits a 0 (0V, closed button) and a 0 (OFF) submits a 1 (5V, open button)
- Type “j10” to simulate a button press on pin RC5 of J10 → sets PORTXbits.Rxx to 1 and shortly after back to 0
- Type the Space-Key to simulate the BUTTON_OK has been pressed → calls DIN_NotifyChange(OK, 0)
- Type “+” to increment the Encoder by 1; enter “- - -” (no spaces) to decrement by 3 → calls ENC_NotifyChange(1, in/decrement)
- Type “e2+++” to increment Encoder 2 three turns → calls ENC_NotifyChange(enc, in/decrement)
- Type “m176,20,100” to send a Midi Controller Change #20 on CH1 (176) with a value of 100 → calls MPROC_NotifyReceivedEvent
- Type “m192,10” to send a Program Change request for PRG 10 → calls MPROC_NotifyReceivedEvent(..)

Setup Guide

If you haven't setup your IDE, please follow these links for your system first:

Setup Guide for XCode on Mac
Setup Guide for Code::Blocks on PC

General Development Info

Then proceed by configuring your main.h and main.c files:

main.h

- you need to add some lines at the bottom of your main.h (or, if you don't have one, main.c) - this is because debug_mios.c calls “DISPLAY_Init()” as MIOS would. Because of the #ifdef statement this does not change your syx-project code!

```
#ifdef _DEBUG_C
  // export functions that are called from within debug_mios.c
  // (e.g. to trigger DISPLAY_Init after sending...)
  extern void DISPLAY_Init(void);
#endif
```

main.c

Minor modifications are required to your MIOS Application's main.c and main.h files.

- You need to add some lines at the top of your main.c - this is to avoid including headers for the PIC/MIOS Core module, when compiling a console app for debugging. Because of the #ifndef statement this does not change your syx-project code, so your app will compile as normal for the PIC!

Before:

```
#include "cmios.h"
#include "pic18f452.h"
```

After:

```
#ifndef _DEBUG_C
    #include "cmios.h"
    #include "pic18f452.h"
#endif
```

ACSim_console.h

- Choose your OS
- Select the LCD-Size
- Set the number of AIN-Lines
- Set the number of ENCoders
- If needed set DIN-Buttons like “OK”

ACSim_console.c

If you have more than just main.c, you have to add the source to these .c files (c only, no headers, else you will get a bunch of compile errors. See the comments in the ACSim_console.c file for details.

These files don't need to be changed, just add them to your project.

Done! :)

Release & Developer Notes

This entry is still in work. It is very incomplete! Especially beginners should be aware of that it's not yet easy to use. This code is mainly intended to simulate and debug new applications. However, if you know C just a bit, you are welcome to try it!

Everyone is welcome to add lines and code! Change anything! Don't be afraid; I don't see this as _my_ work, I hope the code gets completed piece by piece until we finally have a nice simulator/debugging environment built by the mb-community! *audiocommander*

Update: Updated to v0.0.4, now works on MS Windows with GCC (and probably other compilers too) *stryd_one*

Update: Updated to v0.0.5, splitted code to seperate wiki-documents, code cleanup, cleaned namespaces, added MIOS_ICC and MIOS_HLP functions *audiocommander*

Update: Splitted ACSim_toolbox into header and source *audiocommander*

Update: 2007 January 17 improved MIOS_IIC for SpeakJet debugging *audiocommander*

Notes to developers

Some MIOS_function parameters have slightly been changed due to compile errors. If you have questions or would like to try it, you can ask in the forum! I'll give my best to help you using ACSim. It's a bit troublesome in the first three hours, but once you got it, it really helps developing MIOS-apps dramatically!!!

Please 🔧 Fix Me!

- Lots of hardcoded return values (eg. MIOS_ENC_SpeedGet)
- Bankstick could be written to and read from a file
- Many functions are not yet fully implemented. If you add something valueable, please update the Wiki-Sources too or PM me and I'll put it in for you!

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

Permanent link:
https://wiki.midibox.org/doku.php?id=mios_c_simulator_-_debugger&rev=1169038530

Last update: **2007/01/21 21:33**

