

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.

Please see [Release\\_and\\_Developer\\_Notes](#) for version info.

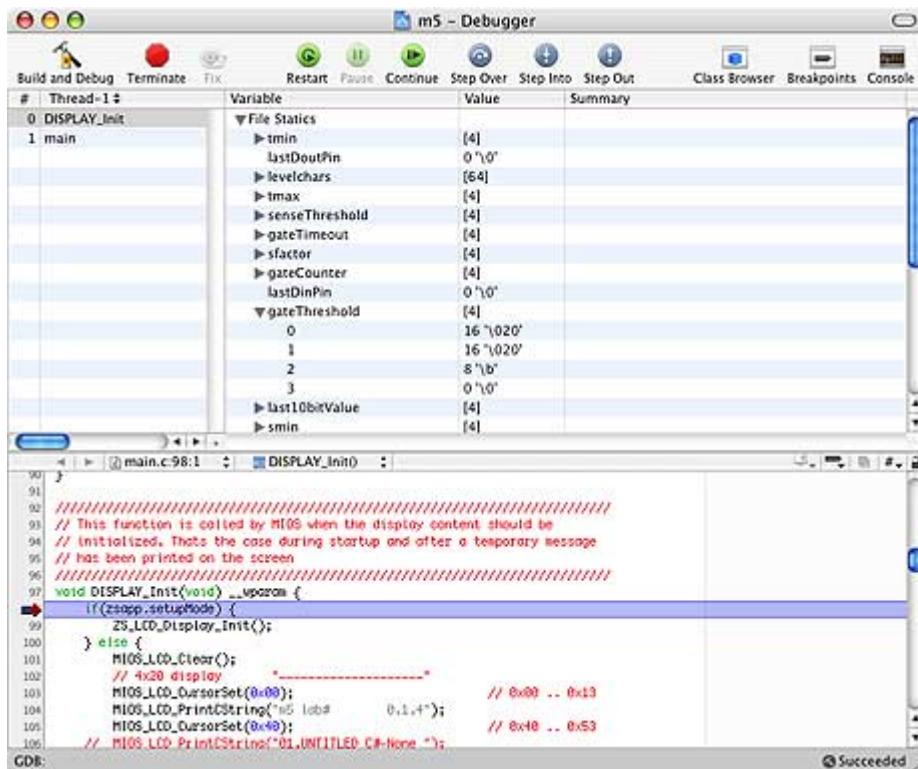
If you add something make sure, you note it here, otherwise your submissions might get overwritten!

# 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):



The screenshot shows the Xcode interface with the "m5 - Debugger" window open. The variable inspector is displayed, showing the "File Statistics" section for the "main" thread. It lists several variables with their current values:

| Variable         | Value     |
|------------------|-----------|
| ► tmin           | [4]       |
| ► lastDoutPin    | 0 '\0'    |
| ► levelchars     | [64]      |
| ► tmax           | [4]       |
| ► senseThreshold | [4]       |
| ► gateTimeout    | [4]       |
| ► sfactor        | [4]       |
| ► gazeCounter    | [4]       |
| ► lastDinPin     | 0 '\0'    |
| ► gateThreshold  | [4]       |
| 0                | 16 '\020' |
| 1                | 16 '\020' |
| 2                | 8 '\b'    |
| 3                | 0 '\0'    |
| ► last10bitValue | [4]       |
| ► smin           | [4]       |

The source code for the `DISPLAY_Init` function is shown in the editor pane:

```

90
91
92 // This function is called by MIOS when the display content should be
93 // initialized. That's the case during startup and after a temporary message
94 // has been printed on the screen
95
96 void DISPLAY_Init(void) __attribute__ ((noinline))
97 {
98     if (ZS_LCD_SetupMode) {
99         ZS_LCD_Display_Init();
100    } else {
101        MIOS_LCD_Clear();
102        // 4x20 display
103        MIOS_LCD_CursorSet(0x00);           // 0x00 ... 0x13
104        MIOS_LCD_Printf("m5 lab# 0.1.4"); // 0x14 ... 0x53
105        MIOS_LCD_CursorSet(0x40);
106        // MIOS_LCD_Printf("01.UNITLED.C#-None ");
107    }
108 }
```

The status bar at the bottom right of the debugger window shows "Succeeded".

Isn't that nice?

# Overview

The Simulator currently consists of these files:

| Source   | Contents                                    | Notes                                    |
|--|---|--|
| <b>ACSim_console</b> (Console based input, main runloop) |   | Adaption required!                       |
| <a href="#">ACSim_console.h</a>                          | Hardware related defines                    | Change #defines to reflect your settings |
| <a href="#">ACSim_console.c</a>                          | Main runloop                                | Add additional c. source-files here      |
| <b>ACSim_mios</b> (MIOS Emulation)                       |   | <i>No changes required</i>               |
| <a href="#">ACSim_mios.h</a>                             | pic18f452.h typedefs and global vars        | <i>No changes required</i>               |
| <a href="#">ACSim_mios.c</a>                             | MIOS functions for simulation               | <i>No changes required</i>               |
| <b>ACSim_tools</b> (Tools & Helpers)                     |   | <i>No changes required</i>               |
| <a href="#">ACSim_toolbox.h</a>                          | Hexview config                              | <i>No changes required</i>               |
| <a href="#">ACSim_toolbox.c</a>                          | Helpers like random generators and hex-view | <i>No changes required</i>               |
| <a href="#">ACMidiDefines.h</a>                          | Midi Defintion listing                      | <i>No changes required</i>               |

You can download the files in a [.zip file here](#). If you make any updates, please send them to sstryd\_one and he will update the zip for you.

<box 75% left bluelight|SCM Code News>I created a SCM page for ACSim, but as the current SCM-implementation of Xcode is quite lousy, I will wait for the Leopard version to complete this step. That means the next update 0.0.8 will follow in the near future and will contain some major changes. I am happy for anyone joining the dev team! <http://code.google.com/p/acsim/> audiocommander 2007/10/26 09:39 </box>

---

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\_Tick() is polled each runloop()
- MIOS\_Timer() is polled each runloop() when a timer has been set

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)**
- **(b)yte(system realtime number)**
- **(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(..)
- Type “b248” to send a single System Realtime byte (248 is MIDI\_CLOCK)

# Setup Guide

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

[Setup Guide for XCode on Mac](#)

[Extend your toolchain with Code::Blocks and GDB \(Don't forget \*\*Part 1 & 2!\*\*\)](#)

[General Development Info](#)

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

## main.c

Minor modifications are required to your MIOS Application's main.c file.

- You need to edit 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
```

## 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
```

**ACSim\_console.h**

- Choose your OS by enusering that only one of these lines is remarked (the one you DON'T want!)
  - `#define _DEBUG_OS_WIN * #define _DEBUG_OS_MAC`
- 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.

**ACSim\_mios & ACSim\_Toolbox incl. ACMidiDefines**

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 0.0.4: now works on MS Windows with GCC (and probably other compilers too) *stryd\_one*

Update 0.0.5: splitted code to seperate wiki-documents, code cleanup, cleaned namespaces, added MIOS\_ICC and MIOS\_HLP functions *audiocommander*

Update 0.0.5-r1: Splitted ACSim\_toolbox into header and source *audiocommander*

Update 0.0.5-r2: 2007 January 17 improved MIOS\_IIC for SpeakJet debugging *audiocommander*

Update 0.0.6: 2007 January 21 added MIDI-Merger, INTCONbits (MIDI-Clock support) *audiocommander*

Update 0.0.7: 2007 January 26 added MIDI-Debug Messages, improved Timer() calls, added Tick() calls *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!!!



- 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:

<http://wiki.midibox.org/> - **MIDIbox**



Permanent link:

[http://wiki.midibox.org/doku.php?id=mios\\_c\\_simulator\\_-\\_debugger](http://wiki.midibox.org/doku.php?id=mios_c_simulator_-_debugger)

Last update: **2008/05/21 02:13**