MIOS Studio

MIOS Studio was started by Wilba in November 2003, and later taken over by Adam King. The original documentation can be found at http://miosstudio.midibox.org/.

It is a java-based, platform-independent MIDI processing environment, which not only provides upload and debug functions for MIOS, but also other useful features like MIDI port routing, data filtering, and a virtual keyboard. It's written in a modular way, and will be published under GPL later so that other programmers can make their contributions.

Adam hasn't finished all of his plans yet, therefore only a precompiled binary package is available. It can be downloaded from the MIOS Studio beta 8 forum thread. Please **read the entire thread**, especially if you encounter problems. He is interested in your input: how is it working, are there any problems, does the user interface need improvements, which features could be useful?

Finally don't forget to edit this page if you discover compatibility issues, workarounds, or have anything to add.

Installation Instructions

Windows

To run MIOS Studio you will need the Java Runtime, at least version 1.5 installed. If you do not have it, you will need to go to the Sun site http://java.sun.com/j2se/1.5.0/download.jsp first, download the JRE 5.0 setup file and install it.

Once Java is installed, all you will need to do is download MIOS Studio from the MIOS Studio beta7 forum thread and save it to your PC. Always use the latest version! Extract the zip file, and put the .jar file somewhere useful.

In Windows, you should be able to just double click on the .jar file to open the program. To start from a command line (or setup a shortcut), you can use the command

```
java -jar <path_to_jar_file>/MIOSStudio_beta7_5.jar
```

to open MIOS Studio.

As an added utility, a cutdown version with just a keyboard controller, MIDI routing/filtering and keyboard zone mapping is available. This can be started with the command

```
java -cp <path_to_jar_file>/MIOSStudio_beta7_5.jar
org.midibox.apps.virtualkeyboard.~VirtualKeyboard
```

Mac OS X

MIOS Studio functions correctly under Mac OS X 10.4 (and, likely, onward).

For versions of Mac OS X since 10.4.8, Apple no longer supports the com.apple.audio.midi java package, which allowed java applications to access the CoreMIDI system. Because MIOS Studio is a java application, you will need something to bridge the gap between your midi hardware and java.

Free (ppc/intel): *mmj* "mmj - Mac OS X universal binary java Midi subsystem" http://www.humatic.de/htools/mmj.htm (This software requires a manual install, so be sure to follow included instructions). Also, if you use Mandolane and then change to mmj be sure to clean the older files of mandolane in your system (speacialy the .lic mandolane file). Please support mmj by donating!

Free (ppc): *Plumstone* "Plumstone MIDI SPI for PPC Macs only" http://www.mandolane.co.uk/software.html (This software requires a manual install, so be sure to follow included instructions)

Demo (time-limited, ppc/intel): *Mandolane.* "Mandolane MIDI SPI for PPC/Intel Macs (£5)" http://www.mandolane.co.uk/software.html (This software requires a manual install, so be sure to follow included instructions)

After you have installed one of the Java/MIDI enablers above, simply download MIOS Studio, copy the "jar" file to your applications folder, and double-click it to run!

Linux

This is now confirmed to work on various distros and MIDI interfaces:

For Debian and distributions based on it, such as Ubuntu; you need to install three packages.

```
sudo apt-get install sun-java6-bin sun-java6-jre libgcj8-awt
```

Once this is done, you can start MIOS Studio by typing:

```
java -jar MIOSStudio(version).jar
```

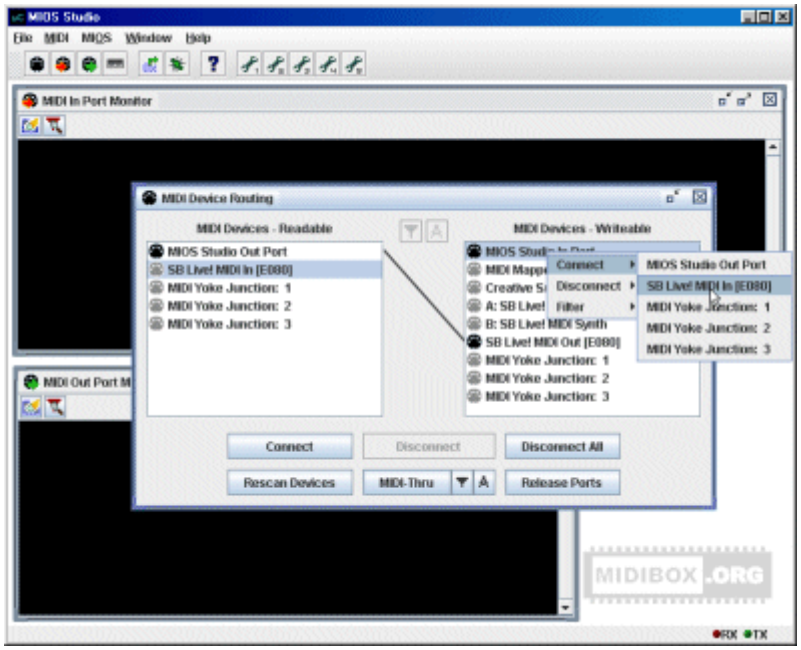
Using MIOS Studio

Below are examples of some of the main features of MIOS Studio. For further details, click on the links to go to the relevent section of the MIOS Studio Help File.

MIDI Routing

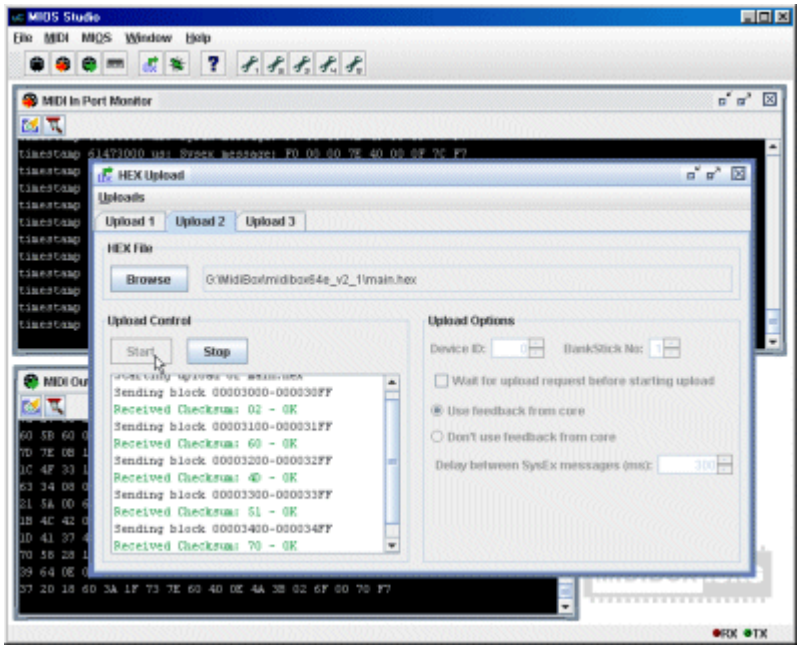
MIDI Device Routing: it allows you to forward and filter MIDI streams to any MIDI port. At the top of the list you will always find the MIOS Studio In and Out port. This is the port to which MIDI data has to be routed when you want to upload code, monitor it, send debug commands, etc...

You must set up the readable port: "MIDI IN" of your interface must be routed to the "MIOS Studio In" Port, and the writable port (MIDI OUT of your interface) has to be routed to the MIOS Studio OUT Port:



Uploading Applications to MIDIbox

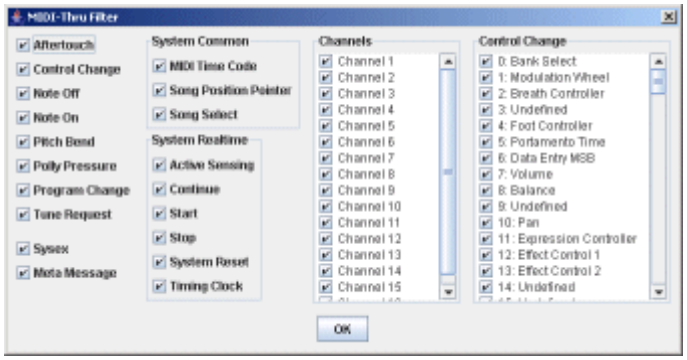
Hex Upload: The MIOS code upload is very comfortable - you don't need to generate a .syx anymore, HEX files can be uploaded directly:



You should **ALWAYS** use 'Smart Mode', and enable 'Wait for Upload Request', and then power on your core after hitting 'Start'. **ALWAYS**. If this doesn't work, there's something wrong, and you should begin troubleshooting as per the uCApps MIDI Troubleshooting page. **ALWAYS**!!! :)

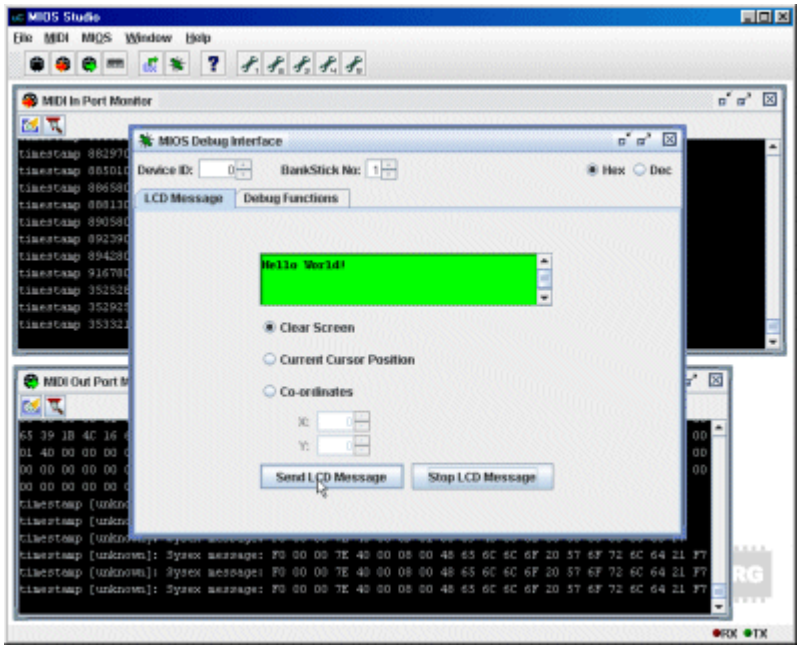
Filtering MIDI events

MIDI Filter: It's possible to filter MIDI events:



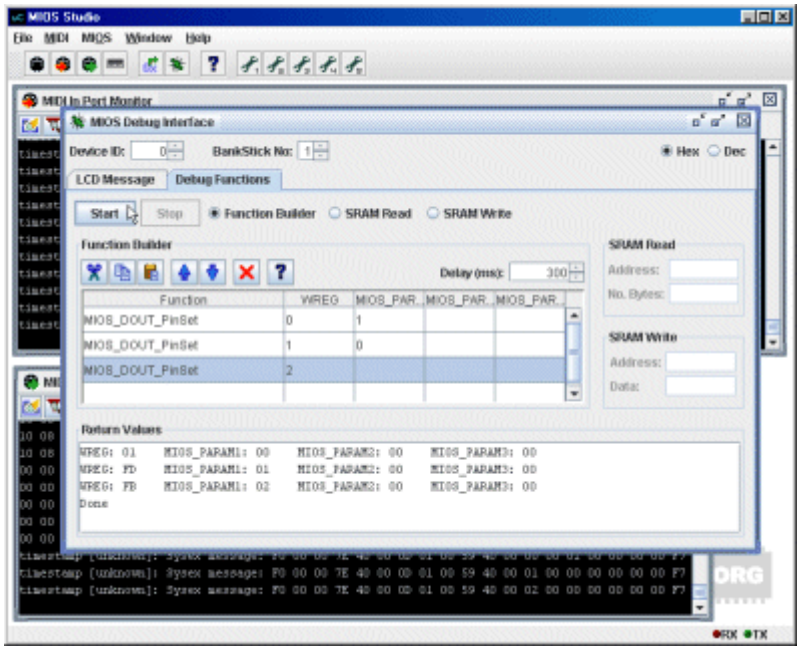
Displaying Messages on Your MIDIbox LCD

MIOS Debug LCD: This is useful for debugging your LCD connection, or just amazing your friends.



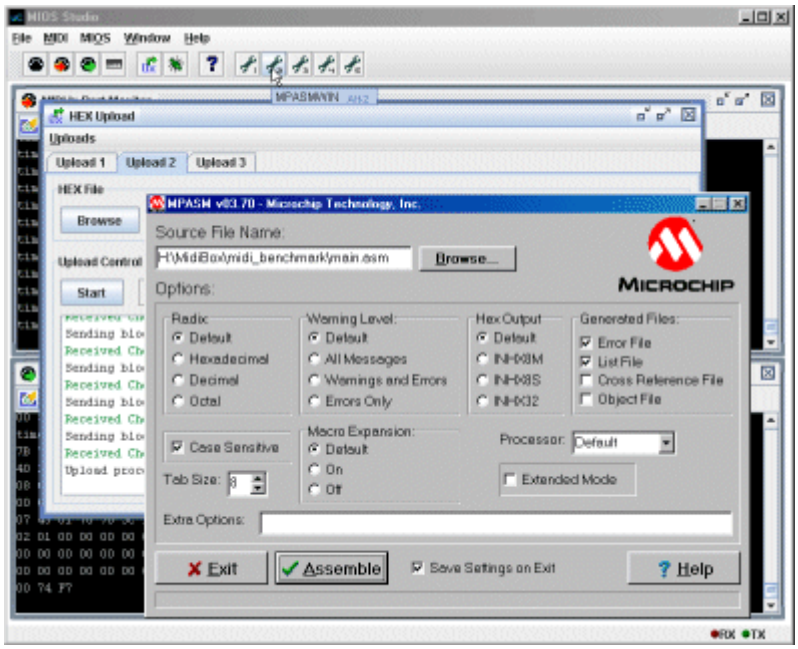
MIOS / Application debugging

MIOS Debug Functions: The MIOS debug window allows to execute MIOS functions via remote:



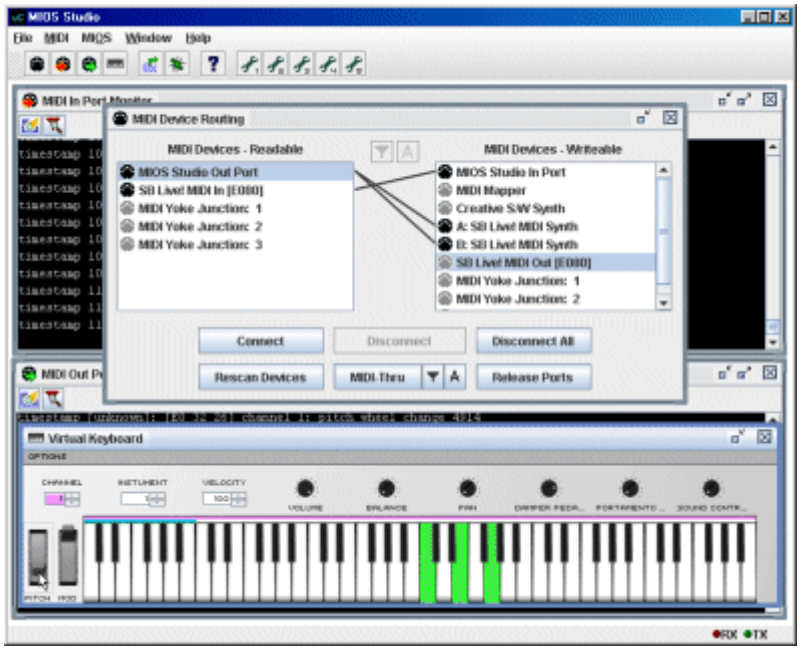
It's also possible to read and write the memory.

Run External Commands: Buttons on the toolbar can be customised to run external commands:



On-Screen Keyboard

MIDI Keyboard Controller: A virtual keyboard controller is also available:



Virtual Keyboard / MIDI Processor

Virtual Keyboard: A smaller standalone program is also available with just the features of the keyboard controller and MIDI Routing/Mapping. This is more suitable for somebody who wants these functions but isn't building a MIDIbox.