General Frequently Asked Questions (FAQ)

This page summarizes the most frequently asked questions concerning the projects found on the http://www.uCApps.de website. It will be enhanced with every new interesting questions and will be modified with every new awareness. Feel free to ask in questions in the MIDIbox forum http://forum.midibox.org or edit this document by yourself if you are missing an important question, don't understand an answer or if you found a better solution in order to improve this “living document”.

About the MIDIbox Hardware Platform

What does µCApps mean?

µCApps stands for “Microcontroller Applications”, and is the name of Thorsten Klose (TK)'s MIDI DIY home page.

Which skills do I need to build a MIDIbox?

There are a lot of people who built a MIDIbox without having any knowledge about electronics before starting. However, if you have never learned to solder, you could start on the soldering page.

Can I control my sequencer with the MIDIbox?

Yes, of course! :) Most modern sequencers like Cakewalk, Sonar, Cubase, Logic offer a MIDI Remote Function. You can control the faders and trigger functions like mute/play/reward/record/... with the buttons. Refer to the manual which comes with your sequencer.

There are also special MIOS applications available like MIDIbox LC (Logic/Mackie Control Emulation) or MIDIbox MM (Motormix Emulation) which are communicating bidirectional with the host application over a specified protocol, they are the most flexible solution for displaying track/instrument/parameter names, since these messages are sent by the host itself. But they also need to be supported by the host software, so check this before.

I posted a message in the Forum/I wrote TK an e-mail three hours ago, why I haven't received a reply yet?

TK doesn't reply every day, but keep cool - your message won't get lost. Note also that he is sometimes on vacation - in Germany they get 30 days and more a year :)

I have posted a detailed concept for a MIDI controller to TK but haven't got an extensive answer

Since TK receives such complex concepts for dedicated solutions nearly every week and noted that reading, understanding and answering consumes a lot of spare time, he decided not to read “concept requests” which are longer than 20 lines of text anymore. Inspiration is always welcome, but he's not coding for your private wishes.

Will you make a synth based on (insert name of chip) for me?

The first question has an obvious answer of: “No”. People (including TK, he's human too! really!) contribute to the MIDIbox project in their spare time. And as such, they want to do things which are interesting to them - not to someone else. Would you like to spend your leisure time doing slave work for someone else? :)

How hard would it be to make a synth based on (insert name of chip)?

Generally, if you can't answer this one yourself - too hard for you. Remember - if you want to start a project based on the MIDIbox, you have to do it all on your own! There are some ready, digitally controlled tone generators out there, which you can buy with complete documentation - many of them are easy to work with, the companies give you all the documents you need, many chips have application notes and guides, the companies also have active communities and even support specialists which you can waste the time of ;) Try tinkering with those first, if you want to. When you're taking a chip out of some vintage computer, you're always dealing with zero support, zero documents, zero manuals, zero guides, zero application notes, and you don't even know if you fried your chip while unsoldering it :) So, if you've never hacked into a microchip - don't expect success at the first try.

Having said that, if you feel you're up to the challenge, you're encouraged to tinker around :)

Building Your MIDIbox

Where can I buy the components?

See this page: where_to_order_components

Which errors can I avoid?

Before you order the components, check the prices of different distributors. Mail order companies with a large variety are the most expensive, smaller companies are cheap and fast. Check also the electronic shops near by you, especially for the chassis (visual control). With luck, you will find special offers, i.e. some good pots or buttons.

When buying cheap components like resistors, capacitors, diodes, pots, buttons... take at least one more to have it on stock when anything goes bust.

Don't forget fitting sockets for every IC because you have to check the voltage pins with a multimeter **before** mounting the ICs. Furthermore a socket for the PIC is very helpful to be sure it won't get damaged while soldering.

Microchip provides several package options for the PIC18F452, make sure that you are buying the PDIP version PIC18F452-I/P or PIC18LF452-I/P

Do I need to buy a LCD for running the MIDIbox?

If this is your first MIDIbox take the answer to be yes.

It really depends on the application you are planning to use; most do not need an LCD for normal operation. But keep in mind that a LCD simplifies the debugging, and you are not able to use the menus which are mostly provided by the MIDIboxes, therefore customization has to be done directly in the source code or via SysEx dumps (if supported). It is very useful to have an LCD that can be plugged into the Core when needed.

More info on compatible LCDs here: lcd

What's the difference between potentiometers, sliders, motor faders and encoders?

Potentiometers, aka pots, are also known as variable resistors. This name best describes the function of the part. As you turn the knob (or slide the fader) from minimum to maximum, you decrease the amount of resistance that the pot introduces to the circuit. This reduced resistance means that more electricity flows through the circuit. The amount of this voltage is an analog signal, so it can be measured by an AIN (analog in) module, which uses the ADC (Analog to Digital Converter) of the PIC in the Core Module, to convert the amount of voltage coming out of the pot, into a numerical value. Then, you can use that value to control part of your MIOS application, for example to send a MIDI CC that controls the filter cutoff when you turn the knob :)

A slider is another type of pot, only it is a straight line. Instead of turning left-to-right, you slider from bottom to top. A motor fader is a slider which has a motor attached to it, so not only can you move the fader to adjust the voltage sent to the Core, but the core can set the position of the slider.

Encoders, aka Rotary Encoders, are often misunderstood. Encoders are not analog, like the pots are. They don't have a minimum and a maximum setting, and they don't put out an analog signal which has a varying voltage to indicate their position. One way to explain them is using the analogy of two buttons. With an encoder, when you turn it to the right, it's like pushing the “up” button, and when you turn it to the left, it's like pushing a “down” button. Buttons are just switches, like a light switch, they allow power to flow through the circuit when they are on. This is a Digital signal, because it is measured as ON or OFF, not as a range of voltages. This pair of digital signals, the “up” and “down”, can be read by a DIN (Digital In) Module, which sends messages in MIOS that your application can use to change a parameter up and down. Note that an encoder needs two pins on the DIN Module, one for “up” and one for “down”. As you continue to turn turn the wheel to the left, it's like you are pressing the “down” button over and over again. As you continue to turn turn the wheel to the right, it's like

you are pressing the “up” button over and over again. This signal could be used in a similar manner to a pot.
First, the position of this 'virtual pot' is stored as a value in the PIC's memory.
When you turn the encoder to the right, the application senses the 'up' signal, and increases that value. Now, that value is sent as the value of a MIDI CC
Similarly, when you turn the encoder to the left, the application senses the 'down' signal, so it decreases that value. Now, that value is sent as the value of athe same MIDI CC

Wikipedia has a decent article about rotary encoders. We use the 'relative' type.

Should I buy linear or logarithmic pots/faders?

Linear pots or faders are required for MIDIbox. Logarithmic are nice for analog mixers, but not for digital processing. In catalogs you will mostly find a “lin.” or “log.” index for every pot type - if not you should call your distributor and ask him before you buy the wrong pots. The pots itself are mostly marked with a letter.

Steve, a frustrated MIDIboxer, wrote some words regarding pot markings: “I read that the European standard for linear pots is to mark the letter “A” on the pot body. I know that some manufacturers use this standard, because I have tested a pot with the letter A on it, and it was linear. But I have also tested a pot made by another manufacturer, with the letter A marked on it, and this one was Logarithmic! So, some manufacturers mark linear “A” and Log “B” and others mark linear “B” and log “A” a bit confusing……. so to make sure you have the right pots, you can test them with a multimeter. At the half way point, a 10K linear pot will measure 5K ohms. At the half way point, a logarithmic pot will either measure 6.5K ohms, (approx) or 3.5K ohms (depending on which side of the pot you are testing). When you put one probe on the center pin, and the other on one of the outer pins. This may be obvious for some people, but it had me confused for ages!”

Where can I find datasheets for the parts you used?

Try Google, type the part number and keyword “datasheet”. If you purchase parts from SmashTV's MIDIbox store the parts list includes Mouser part numbers. You search the mouser distributor site to find datasheets for all of those parts.

Do I have to make a Printed Circuit Board (PCB)?

PCBs are popular, because they give you something like a success guarantee that every connection between the components does exist and the MIDIBox will work with the first power-on. But creating a PCB may take a long time. However, dblevine wrote a nice walkthrough which could help if your are interested: [HowToCreatePCB]. Here is another link to a tutorial:
http://www.piclist.com/techref/piclist/biketut/chapter3.html

If you don't want do DIY and don't want to do without a PCB, you should look for a company which is specialized for prototype boards. I.e. http://www.expresspcb.com in Europe or http://www.4pcb.com/ in the US.

In Germany I found a guy who practise a private PCB service with very good prices for hobbyists:

http://www.mikes-elektronikseite.de/midiseite.htm. All the private boards have been made by him within three days for about 5 to 8 EUR for each PCB. Compared to big companies like http://www.pcbpool.de, where you have to pay at least 40-50 EUR for a board, this is a really good offer. Michael Klein delivers worldwide.

Since the year 2003, Tim (known as SmashTV in the MIDIbox forum) offers a private PCBs service for MBHP boards from the USA. His online shop can be found under http://mbhp.avishowtech.com. He sells also pre-burned PIC18F452 with the MIOS Bootstrap Loader which saves you from having to build a PIC burner like the JDM.

Since 2010 novski develops special PCBs for more experienced DIY makers and offers a opensource download on Github as well as PCBs and Kits over his webshop www.vlrlab.com / https://github.com/novski/Midibox.

Alternatively you can do it like me in the early days - I developed my circuits on vectorboard (sometimes also called breadboard or veroboard, in german “Lochrasterplatine”). These are boards with a lot of pre-made holes ordered in a matrix which save you from drilling holes for every component. My suggestion are vectorboards with pads around the holes (in german: “Lochrasterplatine mit Loetpunkten”) which simplifies the soldering. They cost around 2.50 Euros (…you see the difference? 🙂 Find an example for a vectorboard circuit under http://www.ucapps.de/midibox_ext/lfo/breadboard.jpg

How Do I get the firmware into the PIC microcontroller?

You only need to burn the MIOS bootstrap loader into the PIC with a PIC programmer, thereafter MIOS itself and MIOS applications can be uploaded via MIDI. The details are explained at the MIOS Bootstrap Loader page. If you get the PIC from one of the suppliers of MidiBox boards, the MIOS bootstrap loader may already be burned into the PIC for you and you only need to load MIOS.

MIOS Bootstrap Loader for Newbies is a detailed explanation of the process for first time MidiBoxers.

How do I test my circuit?

Before you plug the ICs into the sockets, you should measure the voltage levels at the power pins (Vss and Vdd - Vss means: ground (0V), Vdd means: +5V, you will find the corresponding pins in the schematics). Disconnect the power, plug the ICs into the sockets, turn on the power. Now quickly measure the voltage between any ground and 5V pin again. If you don't see any change (already +5V) you've made the most critical step. Otherwise you have a shorted in your circuit and should disconnect the power as fast as possible to avoid damages. However, the PIC is hard to destroy… I know that from my experience 😉

Continue with the MIDI Out port tests like descriped at the MIDI Troubleshooting troubleshooting page.

How do I up/download a SysEx Dump to the MIDIbox?

MIOS and application code has to be uploaded with the MIOS Bootstrap Loader

If the application provides an additional configuration mechanism, then follow this link http://www.ucapps.de/howto_tools_mbsyx.html

Can I use a switching power supply (known from PCs) for my MIDIbox?

No, because switching power supplies require a minimal load before they start to operate correctly, and if the circuit is not drawing enough power, or only sometimes like the MIDIbox circuits, the power supply will just switch off and/or reset the circuit reiteratively. Possible Workaround: use of a permanent consumer load like the backlight of a LCD or a lamp. Permanent use of this solution has not been tried yet. Better and cheaper: just buy a good transformer (for about 1-2 EUR) or wall adapter (for about 3-5 EUR), and use a rectifier, a big capacitor, a little capacitor and a 7805 (for Motorfader Driver: 7812) to regulate the voltage. All these components (apart from the transformer or wall adapter) are already part of the MBHP_CORE module.

Update: TK: “I made very good experiences with a switching PSU from Pollin (available for 4.95EUR) on my MIDIbox NG - schematic can be found under http://www.ucapps.de/midibox_lc.html

What is the meaning of a device ID?

In common MIOS applications there are two different IDs: the MIOS ID which addresses the core for code down- and uploads, for debug commands, for remote messages - and the application specific ID which addresses the SysEx handler within the main program.

In a single core environment the MIOS device ID should be zero (0x00). It has to be configured when burning the MIOS Bootstrap Loader into the PIC, but it can also be changed later by using the change_id application. The ID can be verified during the boot phase of MIOS, the “upload request” SysEx string contains it (F0 00 00 7E 40 _the id_ 01 F7). *Note:* once the MIOS device ID has been changed, all .syx files have to be converted so that they are matching with the new request header. Just run the mk_syx.pl script again with the -device_id option. Example: “mk_syx.pl main.hex -device_id 0x01”

It depends on the application how to setup and verify the application specific ID. If it isn't print on LCD (for example within any menu), you could send some pings with different device IDs to the box - once the application returns a ping, you are sure that you've selected the correct ID. It's mostly zero by default, and you should set it to the same value like the MIOS device ID.

I've burned the bootstrap loader into the PIC, but my LCD doesn't show any message?

See lcd page for more information incl. troubleshooting of LCDs

The PIC Microcontroller

Why are you using the PIC and not a more modern Microcontroller?

TK: “I started my first MIDI projects in the 80's with a C64, later I worked with 8051 derivatives. Sometime along the way I decided to use a PIC controller because it *was* modern during many years - cheap, easy to program, almost non-destroyable, no SMD package, high availability for hobbyists. Although I have worked with a lot of 16 and 32 bit controllers in the meantime (job-related...), I stick to the lovely PICs in order to reduce my support effort for these sparetime projects.

Do I need a deep knowledge of PIC microcontrollers in order to build your applications?

No! The software is ready made, normaly you don't need to assemble code, you just only have to burn the application (.syx file) into the PIC via the MIOS Bootstrap Loader http://www.ucapps.de/mios_bootstrap.html. If the application provides software/hardware options, which are not part of the preassembled binary, you have to edit them in the main.asm file of the application package (therefore it always make sense to take a look into this file) and rebuild the application like described under http://www.ucapps.de/howto_tools_gpasm.html

Where can I learn more about PIC microcontrollers?

Get all the interesting datasheets and application notes from the http://www.microchip.com website and check the famous http://www.piclist.com site.

How do you develop your applications?

Update: Please see the Application Development page for new info!

TK's answer, edited to remove references to MPLab which is now obsolete:
Mostly I develop under Linux. I'm using XEmacs http://www.xemacs.org to edit the source, GPUtils to assemble the code and the MIOS bootstrap loader http://www.ucapps.de/mios_bootstrap.html to upload the application into the PIC.

If I have to work under Windows (i.e. for making music with Logic http://www.emagic.de and Reaktor http://www.native-instruments.de, I'm using the Windows version of Emacs http://www.gnu.org/software/emacs/windows/ntemacs.html and GPUtils to assemble the code. Find a snapshot of my development environment under Linux here: http://www.ucapps.de/images/devl_env.gif

It is also possible to develop on the mac, see how to use Xcode2 as IDE on a Mac

Where can I download the source code?

It's part of every application package: http://www.ucapps.de/mios_download.html

Buying and Selling MIDIbox stuff

Where can I buy your stuff?

TK: “All the applications of http://www.uCApps.de are non-profit projects which are not for sale. The purpose is to follow the spirit of Open Source in order to allow people to rebuild, modify, improve or just learn from my projects. Many features that can be found here are not my inventions, but suggested by users from all of the world. On this way *we* are creating what *we* have ever searched for and I myself can enhance my experience with electronics and MIDI (makes really fun! :)”

Would you build a MIDIbox for me if I give you some money?

TK: “I neither have the time, nor the motivation to build MIDIboxes for other peoples. These are DIY projects, that means: Do-It-Yourself. Just ask a friend or an expert in your neighbourhood for help.”

stryd_one: “If you don't have time to solder a little, you sure won't have time to use a synth to make music.”

As everything is free, am I allowed to bring the stuff to market?

Only under special circumstances. See also this forum article.

MIDIbox in Use.

How can I connect a MIDIbox to my MIDI device/computer?

For a direct connection to a common MIDI device you need to use the MIDI In and Out of the CORE module. The optocoupler behind the MIDI In port decouples the circuit from the other device to avoid grounding loops. The MIDI device “at the other side” has normaly also an optocoupler at the MIDI In.

Such a “common” MIDI connection via optocouplers is the best solution if the MIDIbox should be connected to a computer. This avoids jitter on the analog inputs, and buzzing noise on the audio outputs (relevant for MIDIbox SID and FM)

PC MIDI interfaces:

- Mostly a MIDI Interface is integrated into the gameport of your motherboard or soundcard. You only need to buy a special adapter with MIDI plugs. Alternatively you can build an adapter yourself, see http://www.tarigon.de/tramp/midibox.html, http://www.borg.com/~jglatt/hardware/pc_intfc.htm,

http://www.siliconchip.com.au/cms/A_101116/article.html

I made bad experiences with CNY17 based circuits (it's an optocoupler without internal amplifier), the 6N138 or PC900 optocoupler should be prefered

- a dedicated (commercial) ISA/PCI/USB based interface with 2 or more IO ports is propably most preferable. Hint: if you are buying a new one, pay attention for “multiclient capability”. This is a very important feature, which allows you to use the MIDI IO ports with different applications (e.g. Cubase, MIOS Studio and JSynthLib Editor) at the same time!
- MBHP_USB provides two MIDI In and two MIDI Out ports, but due to M$ driver quirks it is not multiclient capable
- add other recommendations here...

Alternative solutions:

- you can plug the MIDIbox directly into the Gameport like shown here: http://www.ucapps.de/mbhp/mbhp_midi_gameport.gif
- you can build the MIDIbox-to-COM interface.

Both alternative solutions are not recommended for MIDIbox SID or FM due to the missing optocoupler.

What is the meaning of this funny hexadecimal numbers?

That are the MIDI messages which will be send over the MIDI interface.

The MIDI Specification describes the most important (standardized) messages, a hexadecimal to decimal conversion table can be found here: http://www.ascii.cl/conversion.htm.

A complete official overview of MIDI Messages can also be found on the midi.org site:

- http://www.midi.org/about-midi/table1.shtml
- http://www.midi.org/about-midi/table2.shtml
- http://www.midi.org/about-midi/table3.shtml

Which MIDI events can I send with the MIDIbox?

In fact MIOS can send any MIDI data, it only depends on the application which events can be configured without additional programming effort. E.g., with MIDIbox64/MIDIbox64E/MIDIO128 you are able to send Note/Controller/Aftertouch/Pitch Bender/Program Change Events by default. The events can be configured via the MIDI Learn function or via MIDI dumps. SysEx Streams, NRPN events, multiple events or whatever can be implemented via Meta Events.

How high is the latency of MIOS?

First of all it should be noted, that the latency of a common MIOS application is negligible compared to the latency of a computer. MIOS is a real-time system which runs on a dedicated microcontroller.

The reaction time of the low-level drivers is deterministic, calculable and ensured. The design focus was on MIDI controllers, therefore the processor load caused by tasks running in background has to be measured against the initial latency caused by the MIDI transfer protocol.

It can be assumed that the latency of MIDI itself is about 1 mS: the baudrate is 31250 bps and a common Note or CC event consists of 3 bytes (the running status which makes the transfer faster is not taken into account here), this results into a transfer delay of 960 uS. Additional delays are caused by the UART, the optocoupler, the buffer logic (MIDI Thru port...), but I don't want to get lost in details here... it's more important to know, that this latency increases with more complex events like SysEx or NRPN (at least ca. 2-3 mS) and the number of events which should normally happen at the same time, but must be pipelined through the serial interface.

With these values in mind, the so called system tick of MIOS was designed to be 1 mS. This is the minimum sampling/update frequency of the DIN and DOUT chains. The approximative task is running in background and interrupts the main program every millisecond with high priority - only the priority for receiving incoming and transferring outgoing MIDI bytes is higher. The MIDI transfer itself don't use so many system time, since it's buffered with 64 (incoming) and 48 (outgoing) bytes. This means that when a short MIDI stream i sent, the program will continue to run without waiting until the transfer finishes. Of course, big SysEx dumps will stall the main program, but they won't stall the background tasks! Incoming events: from my observations mostly only 2-3 bytes of the 64 byte MIDI-In Buffer are allocated. This also gives an impression of the overall system performance: 3 bytes == ca. 1 mS maximum waiting time until an incoming event has been processed. But nothing is without exception: if incoming data is received while the main program saves data into EEPROM or flash memory, the buffer could be filled rapidly due to the long writing times (5-10 mS for a page). Worst case scenario: the CPU will be suspended when writing data into the flash memory, during this time all events (DIN triggers, AIN changes, MIDI data) will be delayed or lost (among other things, this is the reason why you must specify a transfer delay in MIDI-OX).

Speaking about analog inputs: they are scanned periodically; each input takes 200 uS, so 64 inputs take 12.8 mS. Higher scan rates make no sense, since the signals have to be “settled” before the conversion – to avoid jittering values which are acceptable for audio applications, but not for a MIDI controller. But wait: 12.8 mS is not the real latency of the AIN driver because of the Dynamic Priority handler ™ which scans the last two changed AIN pins more often than the others. This means: if you tweak two pots at the same time (with your two hands — or do you have more?), the resulting latency of those two will be much less than the others (about 1-2 mS).

High performance inputs/outputs: yes, they do exist  The DIN and DOUT chains are intended for common digital IO, but sometimes it is required to interact with an external peripheral much faster than 1 mS. Examples: IIC devices like the BankStick, an LCD module or the MBHP_SID module. These devices are connected to dedicated PIC pins which are normally controlled from the main program. It's very hard to measure the latency to these modules, since they are normally serviced with lowest priority. More deterministic IO can be realized by using the timer ISR (USER_Timer).

This leads us to the USER_Tick, an endless iteration loop which will be processed when nothing else is to do. One iteration can take from 10 uS to 1 mS and more, it just depends on the system load. In normal applications it rarely exceeds 500 uS.

Last words to the MIOS programmers: take care that periodically called interrupt service routines (ISR) are running as fast as possible so that other ISR tasks are not blocked. Especially take care that they never exceed the 640 uS threshold, since this is the time for which the PIC UART can buffer two

incoming MIDI bytes before the double-buffer overruns (worst case, back-to-back transfer). Time consuming calculation routines should be split into several pieces and executed in rotation with every USER_Tick iteration so that every low-priority task gets the chance to do its stuff as often as possible (a simple implementation of preemptive multitasking). And a last tip: don't print text on the LCD immediately after an AIN or DIN event has been notified, but just request an update (as demonstrated in all example MIOS applications). This saves the system from unnecessary lags caused by rapid display changes which will be overwritten immediately and therefore not visible for the user.

Last words to the users: common latency of a PC MIDI interface ca. 2-10 mS, common latency of a soundcard ca. 3-20 mS, common latency of yourself: ca. 1-2 s 

MIOS MIDI Benchmark: http://www.midibox.org/forum/index.php?topic=2342.0

What the hell are Double-Note-Events?

Programs like Cakewalk, Cubase, Logic, ... support MIDI remote, which allows the user to control the software with his MIDI keyboard or something similar. Normaly MIDI remote works in a manner, that one note has to be played to enable MIDI remote (→“Shift” key), and another note at the same time to trigger a function like Play, Record, Pause, Forward, Rewind and much more.

The MIDIbox64/MIDIbox64E simplifies the use of MIDI remote with Double-Note Events. You only have to press _one_ button in order to play _both_ notes which are necessary to trigger a function of your software. Double Note Events can be assigned to the buttons with the meta codes _F0 08_ to _F0 0F_ (see also mb64_meta.inc, resp. mb64e_meta.inc).

These codes have to be defined in your MIDIbox initialization file, see also http://www.ucapps.de/midibox/mk_syx.zip

Customize your MIDIbox

How many pots/ faders/ buttons/ encoders/ screens/ LED's/ etc is the maximum that I can connect?

There are two parts to this answer. The first relates to the standard hardware, the second is related to the implementation of that hardware.

Hardware

AIN: Up to 64 analog inputs (pots/sliders/etc) are available by chaining two AINx4 modules
DIN: Up to 128 digital inputs (switches/encoders/etc) are available by chaining four DINx4 modules
DOUT: Up to 128 digital outputs (LED's/7-segment LED digits/etc) are available by chaining four DOUTx4 modules
LCD's: Up to 2 40×2 Character LCD displays (IE HD44780), 1 Graphical LCD (IE KS0108), 3x Nokia GLCD (IE PCD8544)

Implementations

Now, connecting the hardware physically, and actually being able to use it, are two different things.

Standard Implementations

One or more, but not necessarily all, of the previous options are available in standard MIOS Applications. For more specific information, see the page related to your application on uCApps All options will be explained there. If you are using a standard application, most of the time only the main.c/main.asm and/or configuration files for the application will need to be altered, to tell the application which hardware configuration you have built.

Custom Implementations

Although the MBHP/MIOS system has suggested standard configurations, it is extremely flexible, so those who are capable programmers/hardware engineers may be able to push the system to further capabilities, however this becomes complex and is certainly not for newbies.... but hang around a few months, build a simpler project and learn some C or ASM in the meantime, and you'll pick it up no worries, so long as you are keen enough :)

Some examples of such customisations are as follows:

- Core modules can be cascaded using MB-Link or MIDI, to allow for devices to be combined. (MIDIBox of the Year link)
- Digital devices such as switches and LED's may be connected in a matrix fashion, which allows for a greater number of connected devices and/or easier handling. (Scanning Matrix examples, SID, LED-Rings, Keyboards (both alphabetical and musical))
- Pins can be reallocated where not required, to give more digital In and Outs (Eg: J5)
- Additional control lines such as CS RE and WE lines can also be custom driven to allow connection of extra devices such as LCD's
- LCD's can be connected serially
- Additional RAM can be driven through LCD pins
- IIC buss can be used to connect other IC's to the Core

Can I use PWM to control individual LED brightness

The short answer is that using PWM on DOUT to do individual brightness control on a large number of LEDs is not possible as it would overload the Core.

e.g. see http://www.midibox.org/forum/index.php/topic,10278.msg77510.html#msg77510 "[...] controlling the brightness of individual LEDs is not possible with the MIOS concept, as it would load the CPU so much, that no other tasks could be executed in parallel. Problem is, that you need a PWM period of 10 kHz or more if brightness is to be controlled with a resolution higher than 3bit. That's the

reason, why dedicated chips (or microcontrollers with special integrated PWM peripherals) are normally used - they keep the load of the main CPU low.”

The longer answer, as mentioned above, is that this is possible using specialised LED driver chips but you would need to interface these to the Core yourself.

Can I connect other parts aside from pots to the analog inputs?

Yes, you can connect anything which delivers a voltage between 0 and +5V like additional buttons, CV outputs of antique synthesizer equipment, or just external effect units like LFOs, see the MIDIBox Extensions page.

How do I add a MIDI Thru Port?

Find the circuit under http://www.ucapps.de/midibox/midi_thru.gif - the 74HC00 is used to amplify the received MIDI signal. The MIDI Thru Port will not work without this amplifier.

Can I power a PIC from the MIDI line without additional power supply?

Thats very problematic and mostly doesn't work properly - the MIDI specification doesn't take this into account. The MIDI line is specified as current loop, and the maximum voltage level is not defined. Most interfaces deliver a logic signal with TTL level between 0V and 5V, a 7805 regulator between MIDI Out and the PIC power pins would reduce the voltage to <4V, so it should not be used in this case. Instead the ground pin and one of the outer pins which always supplies 5V (has to be checked with a measuring instrument) should be connected directly to J2 (behind the 7805). A diode has to be added in serial to prevent back-currents. An additional cap (100 uF) stabilizes the voltage level if a large amount of MIDI data (low/high pulses) is received.

Note: some people who tried this with the MIDI merger and the MIDI processor reported that this is working, others not. Somebody tried it with a very small PIC (16F628) at low frequency but without success on his Yamaha CS1x MIDI keyboard. I don't provide a schematic here since I cannot guarantee that your MIDI interface won't be damaged during the experiments. So, the information above is only relevant for experienced electronic experts.

Forum / Wiki / Web

I can't log into the forum

Possibly you are using the M$ Internet Explorer which prevents YaBB from saving “cookies” within a frame which contains content from another domain (midibox.org). In this case click on the “Forum (unframed)” link instead, or (better) change your internet browser - see also http://www.heise.de/security/dienste/browsercheck/demos/ie/

I can't log into the Wiki

Make sure you have cookies turned on in your browser. As of May 13, 2006, at least one browser will not log in to the Wiki: Camino 1.0.1 for Mac. (Please add to this list if you find a browser that won't log in to the Wiki)

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

Permanent link:
https://wiki.midibox.org/doku.php?id=frequently_asked_questions_faq&rev=1465568429

Last update: **2016/06/10 14:20**