PIC18F4620 - one of the Future Microcontrollers for memory intensive Applications

A special variant of MIOS is available for the PIC18F4620. The usage of this processor is (currently) only required for the next major step of MIDIbox SEQ (V3), most other projects are running fine on a PIC18F452 *(MIDIbox SID V2 will run on a PIC18F4685)*

Biggest advantages of this microcontroller: 64k internal flash (2 times more), 3968 bytes RAM (2.6 times more), 1024 bytes internal EEPROM (4 times more), and hardware compatibility to the PIC18F452 - therefore the same MBHP_CORE module can be used.

But Rev A3, A4 and B4 of the chip contains a silicon bug within the EUSART peripheral, which makes it nearly useless for MIDI applications: zero bytes can be sporadically inserted into the MIDI Out stream (bug has been found during the development of the MBHP_USB_PIC module; meanwhile - after more than one year - Microchip has documented it in the errata sheet). Note that Microchip recently released the B4 Silicon, where this Bug is not listed in the errata sheet - but it turned out that it still exists. It's a documentation error!

As a workaround for this issue, a MBHP_IIC_MIDI module can be used with minimal configuration (MIDI Out only). This adds acceptable costs of 5 EUR to the project.

There is a small application available at the MIOS Download page which allows you to determine the revision ID (search for “revision_id”)

The PIC18F4620 is almost software compatible to existing MIOS applications. There is a special MIOS version which needs to be uploaded, the major differences are different memory limit checks (e.g. code upload to addresses >= 0x8000 are possible).

For information on developing or converting applications to use the PIC18F4620, please see the page on Application Development

FAQ

Do I need a special Core Module for the PIC18F4620?

No, the PIC18F4620 is pin-, hard- and softwarecompatible to the PIC18F452

Can I run an old PIC18F452 Application on a PIC18F4620?

Yes, all applications are binary compatible, which means, that you even don't need to build a new .hex file, just upload the precompiled code. Only the Bootloader and MIOS itself got some small adaptions in order to allow to program the upper flash area, and the extented EEPROM range. Therefore you will find special versions in the MIOS update package within the pic18f4620/ directory

Do I really need to build a MBHP_IIC_MIDI module as workaround for the EUSART bug?

It depends on the application. The EUSART based MIDI Out port of the core module will mostly work, you are even able to upload code without failures. The error (inserted 0x00 bytes which violate the MIDI protocol) only happens very rarely under special circumstances. It can only happen, when MIDI data is also received at the MIDI In port at the same time at the wrong moment (in simple words, details are documented in the errata sheet). As a result, you could notice a hanging note, or a wrong controller once or two times a hour when you are using the MIDIbox.

Accordingly, so long the MIDIbox application only receives MIDI data, or so long it only sends MIDI data while nothing is received, the bug will never appear, and therefore a MBHP_IIC_MIDI module is not required in this case.

(removed part about MIDIbox SID examples, as it has been decided to use a PIC18F4685)

Remaining Example: MIDIbox SEQ: when used as standalone sequencer, no external MIDI keyboard or MIDI clock source connected to the MIDI In, a MBHP_IIC_MIDI module is not required. In all other configurations one or up to 4 MBHP_IIC_MIDI modules are recommented. Note that MIDIbox SEQ allows to output MIDI events on different MBHP_IIC_MIDI ports, this is nice for reducing the MIDI latency

Do I need to built a complete MBHP_IIC_MIDI module with MIDI In and Out port?

No, you only need the reduced version with MIDI Out port only. The MIDI In port is *not* natively supported by most applications, currently it is only used by the MIDI router project. Adding support for multiple MIDI Ins is not so trivial, since it affects the performance of the application - therefore it cannot be expected that it will be supported by many applications in future, only by special ones.

Info for Newbies: if you are unsure, how a “MIDI Out only” version looks like when you are using the MBHP_IIC_MIDI PCB, just stuff all components, and left out the optocoupler and the MIDI In LED.

How do I upload MIOS and application code via the MBHP_IIC_MIDI interface?

Proposal for Beginners/Electronic Newbies:

- buy a preprogrammed PIC18F4620 from SmashTV or Mike with PIC ID 0000000000000000
- build the core module and test the MIDI In/Out of the board
- upload the PIC18F4620 version of MIOS via the MIDI Out port of the Core module
- build a MBHP_IIC_MIDI module and test it like explained at the MBHP_IIC_MIDI page
- select the ID 10 (both jumpers stuffed)
- upload the iic_midi_10.hex binary of the change_id package, this will change the PIC ID to 0000000000001000
- now the MIDI out stream is redirected to the MIDI Out port of the MBHP_IIC_MIDI module, you don't need to use the MIDI Out port of the core module anymore (even for code uploads it's not required anymore)

Proposal for Experts:

- buy a preprogrammed PIC18F4620 from SmashTV or Mike with PIC ID 0000000000001000, or burn the bootloader by yourself into the chip with this ID
- build the Core + MBHP_IIC_MIDI module

- the MBHP_IIC_MIDI module should send the upload request each 2 seconds so long MIOS is not installed
- upload the PIC18F4620 version of MIOS via the MIDI Out of the MBHP_IIC_MIDI module

Note that it is possible to change the PIC ID, or to upload MIOS without the MIDI Out Port, when the “no feedback from core” option is selected in MIOS Studio. This method is not recommented, as erroneous uploads won't be notified, but it helps in “emergency cases”, e.g. when you wrongly uploaded the main.hex file of the change_id application and you are not able to open the case of your MIDIbox quickly in order to get access to the core based MIDI Out port.

Can I use one or more BankSticks together with the MBHP_IIC_MIDI module(s)?

Yes, on an IIC bus all SCL/SDA (clock/data) lines are connected together, each device must have an unique address, so that the IIC Master (the core module) can perform read/write transactions on a specific IIC Slave. On the MBHP_IIC_MIDI module the address 10/12/14/16 (hexadecimal) is selected via jumpers, on IIC EEPROMs (BankSticks) the address 50/52/54/56/58/5A/5C/5E (hexadecimal) is selected with the Chip Select inputs Pin 1, 2 and 3. As you can see, there is no address conflict between MBHP_IIC_MIDI modules and BankSticks

Can I use my old JDM burner to burn the bootloader onto a PIC18F4620?

Yes. Try JDM with Winpic800

Can I use my old JDM burner to burn the IIC MIDI firmware onto a PIC16F88?

Yes. Try JDM with Winpic800

From:
https://www.midibox.org/dokuwiki/ - **MIDIbox**

Permanent link:
https://www.midibox.org/dokuwiki/doku.php?id=mios_pic18f4620&rev=1168822511

Last update: **2007/03/22 19:48**