http://www.instructables.com/id/Getting-started-with-VMUSIC2/

[](http://www.instructables.com/file/F9UKCU4FIM95VAA/)The VMUSIC2 is a complete MP3 player module from FTDI, Inc. which makes it easy to integrate MP3 functionality in to your next microcontroller project. It has two interfaces: SPI or UART (serial)   
  
Some example applications:   
1. Make your robot talk and play sound effects based on some condition or trigger   
2. Add sound to your models and dioramas; Provide some interactivity or history to your piece   
3. Interactive art installations   
4. Spice up your Pinball or other arcade machine   
5. Use the module to load a script from a flash drive defining an animation, such as playing back a sound file while performing some mechanical routine as used in animatronics. The play time is output in seconds which can be used for (crude) syncing (HINT: This last one is what I have planned for a later instructable)   
  
The documentation from FTDI describing how to use this module is not laid out very well and I had some trouble getting started with mine. While looking for information, I have seen many posts in various forums from people also having trouble getting started, so after finally getting mine running, I decided to create this instructable to help others.   
  
This instructable will cover getting connected and controlling your VMUSIC2 module from hyperterminal. I recommend starting out this way because it will allow you to get familiar with the commands and get a feel for how it works before connecting it to your microcontroller. I will follow up with another instructable on controlling it from a microcontroller.

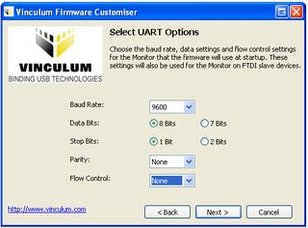
**Step 1: Introduction to the VMUSIC2 module**

[](http://www.instructables.com/file/F662I4UFIJ3YDOF/)[](http://www.instructables.com/file/F9UKCU4FIM95VAA/)

The VMUSIC2 module is based on the Vinculum VNC1L USB Host controller from FTDI and the VS1003 MP3 encoder/decoder from VLSI. The VMUSIC2 allows you to plug in a USB flash drive filled with MP3s and play them from your microcontroller.  
  
This module is inexpensive and comes in an enclosure that can easily be mounted in a panel cutout to give a professional look to you project.  
  
The main advantage to using this module is that it only requires a few pins to control it, and the software overhead is very small. This allows your microcontroller to focus on the rest of your project (monitoring switches, etc...) while the MP3 is playing. It also means you can get away with using microcontrollers with less program memory and fewer I/O. Another advantage is that you don't need to know anything about programming for FAT32, USB host, or MP3s.  
  
While the VMUSIC2 module is designed for playing back MP3s, the USB host interface can also be used to access (both read and write) any files on the flash drive. When I get to the instructable where I connect this to a microcontroller, it will be apparent how useful this is.  
  
You are going to need the following items to complete this instructable:  
\*VMUSIC2 Module from FTDI.  
\*RS232 to RS232-TTL-3.3v converter **(if you connect this module directly to your PC serial port, you risk damaging it. See step 3 for a suggestion)**  
\*5V power supply  
\*A computer with hyperterminal (included with windows) or some other terminal program  
  
Got everything? Good let's take a look at the module:

**Step 2: Programming the firmware**

[](http://www.instructables.com/file/F8FWW1MFIJ3YDQI/)[](http://www.instructables.com/file/F1RTM7PFIJ3YDQD/)

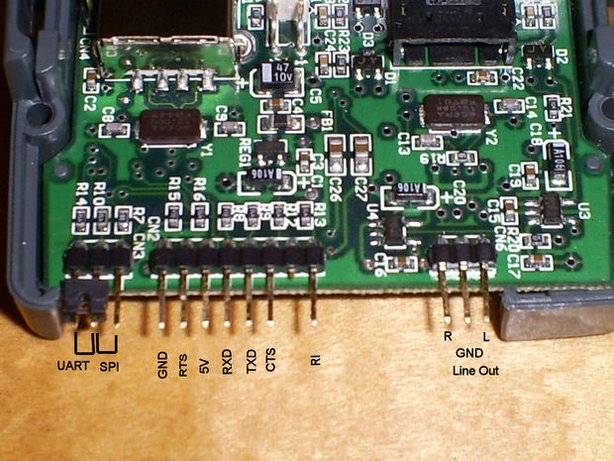
[](http://www.instructables.com/file/F6UDNUYFIJ3YDQJ/)[](http://www.instructables.com/file/F8R6TXKFIJ3YDQF/)

[](http://www.instructables.com/file/FBIJRIYFIJ3YDQG/)

[](http://www.instructables.com/file/FG5EWJ9FIJ3YDQH/)

I ordered my VMUSIC2 module directly from FTDI. The biggest source of my frustration was that I found that my module had no firmware installed. Of course, I thought it was a problem with my connections, so I went on a wild goose chase trying to figure out what I was doing wrong! And I didn't realize that I could upgrade the firmware from the USB drive, so I ended up doing it the hard way. This instructable should spare you that.  
  
Also, the default firmware is not set up to be friendly with hyperterminal. So, we will be modifying and updating the firmware.  
  
First, go to [the Vinculum website](http://www.vinculum.com/downloads.html) and download the latest **Reflash(FTD)** file there. Also get the **VPROG Re-flash Utility** (not used in this instructable, but will be helpful if you have trouble) and the **Firmware Customizer** from the Applications and Utilities section further down the page.  
  
Start up the Firmware customizer and load the Reflash(FTD) file that you downloaded. You will see a summary of the programmed options in the firmware. Click next to begin editing the firmware.  
  
(I have included screenshots of the sections that I have changed)  
  
This is a summary of the options that I chose:  
UART: 9600 baud, 8 bits, No Parity, 1 Stop bit, and **No flow control** (this one is important)  
IPA Mode and Extended Command Set  
  
Now, just click next through the rest of the options. You will be asked for a unique 3 character identifier for your custom firmware. I created two different firmwares and used these IDs; COM for hyperterminal friendly (this one) and MCU for the microcontroller friendly version. We are only concerned with the hyperterminal friendly version for this instructable.  
  
Now, save your new image somewhere and name it **FTRFB.FTD**. You might need to change the file type in the drop down menu to save an FTD file. A ROM file will not work for this instructable as it cannot be flashed from the USB drive.  
  
Now, copy the FTRFB.FTD file that you just created over to the **root folder** of the flash drive you are planning to use.  
  
Alternately, you can name the original file something that will help you identify it later, and then just rename it to FTRFB.FTD after copying it to the flash drive. It MUST have this filename or the bootloader will not recognize it and reprogram itself.

**Step 3: Getting Connected**

[](http://www.instructables.com/file/FCY40GMFIJ3YDRB/)

The first step is to connect the VMUSIC module to 5V and ground. Refer to the picture below for the pinout.  
  
After powering up the VMUSIC2, insert your USB drive with the FTRB.FTD file on it.  
  
You should see the lights blinking. It will take about 20-30 seconds to program the new firmware. **Do not remove power while it is flashing** or you will have to go the more expensive and complicated route I went through (or worse -- you loose the bootloader completely.) The light should glow RED for a second or two **and then stay a steady green once flashing is complete.** Don't be impatient. It will be pretty obvious while it is programming. Normally, there is no activity light on the USB flash drive upon start up. The lights will be going crazy while it is programming and verifying.  
  
Disconnect the 5V from the VMUSIC module.  
  
You are now ready to hook the rest of the signals up and start talking to it with hyperterminal.  
  
This is where a bunch of information is unclear in the documentation. First of all, the VMUSIC2 UART interface is 3.3V logic. You will need some type of converter for your connection. I am not going to go into too much detail on how to do this, because there are many options. One of the easiest is to purchase a Nokia DKU-5 cellular phone data cable and modify it by cutting off the Nokia connector. This cable is a USB to RS232-TTL3.3v cable based on the FT232R. There is no CTS or RTS handshaking signals in this cable, which is one of the reasons that we modified the firmware for no flow control.  
  
There are several articles on the internet that tell you how to modify this cable and it is useful for many purposes. **It doesn't matter what converter you use as long as it is designed for 3.3V TTL, not 5V.** Prepare for the next step by making sure that you have installed the drivers for your converter (if USB.) It should be showing up in device manager. Make a note of the COM Port assigned to it.  
  
If enough people have a problem with this step, I will create a separate instructable based on the nokia cable. But this cable is extremely popular in many modding communities and the mod is pretty well documented. I am using a different (more expensive) converter right now and do not have access to my nokia cable ATM, otherwise I would add it here.  
  
**UPDATE: Found a good article explaining different options and sources to cheap cables. I don't know how I missed it, but the TTL-232R-3V3 from FTDI is practically made for this purpose. Check out [this](http://www.nslu2-linux.org/wiki/HowTo/AddASerialPort)this link for other options (you might even have something lying around)**  
  
Before connecting anything else to the VMUSIC2, again, make sure the power is not connected.  
  
Connect the RXD on your converter to the TXD on the VMUSIC2  
Connect the TXD on your converter to the RXD on the VMUSIC2  
Connect the GROUND on your converter to the GROUND on the VMUSIC2  
Connect CTS on the VMUSIC2 to GROUND (This is not optional. You will be able to send commands, but you will receive no responses if you do not connect this signal to ground.)  
RTS can be left disconnected  
RI is left disconnected  
  
Now would be a good time to load the flash drive up with some MP3s (I renamed mine 1.mp3, 2.mp3, etc... to make it easier) and also delete the FTRFB.FTD file from the flash drive. The firmware can only read DOS filenames. This means 8 characters for the filename and 3 for the extension. I believe you can leave off the extension and just use all eleven characters for a filename, but that might break in future firmware, so I don't advise it.  
  
Leave the 5V line disconnected. We will be connecting it again in the next step.  
  
I want to make a few final notes regarding this section just to make it complete, as it will not be relevant in future steps:  
  
If you do have a converter capable of RTS/CTS flow control, this is actually preferred. If you ever need to reload the firmware the hard way, this is absolutely necessary. The bootloader only supports this mode. There are two reasons that we turned this off, however. The first being that most people are going to find cheap cables that do not support this, and programming a microcontroller to handle RTS/CTS flow control is a bit more complex and requires more IO and software overhead. Most libraries do not support it, so just turn it off. If you run into trouble flashing your firmware, you are going to need an RTS/CTS capable serial converter to recover it, though.   
  
To use RTS/CTS flow control, swap the two signals between devices. This means RTS on the converter should be connected to CTS on the VMUSIC2 module, and vice versa.   
  
Finally, RI is Ring Indicator, which can be pinged to wake the VMUSIC2 module out of standby. We will not be using it here and it is not used in firmware recovery. Just leave it unconnected.  
  
The pinout of the VMUSIC2:

**Step 4: Start talking**

Start up hyperterminal or whatever serial communication program you like and create a connection on the com port of your converter with these settings:  
  
9600 Baud Rate  
8 Data Bits  
1 Stop Bit  
No parity  
No Handshake or Flow Control  
  
You might also want to turn on local echo so that you see what you are typing. The VMUSIC module does not provide remote echo.  
  
Once you have a connection established, connect the 5V on the VMUSIC.  
  
You should see a prompt that looks something like this:  
  
*Ver 03.64-COMVMSC1F On-Line:*  
*No Disk*  
*No Upgrade*  
  
Now insert your flash drive, and it should report that a device is detected on Port 2 (P2) and give you a D:\> prompt indicating that it is now ready to accept commands!

**Step 5: TODO: Firmware Commands**

This is going to be rather lengthly, so will get updated as I have time.  
  
Here are a few quick commands to try:  
DIR (Lists the files in the current folder)  
CD <directory name> (changes the current directory)  
VPF <filename> p(lays the MP3 file <filename>)  
VP (pause and resume)  
VST (stops playback)  
V3A (Play all files in current directory)  
VF (Skips to the next track)  
VB (Skips to the previous track)  
VSV <byte> (sets the volume. Valid values are between $00 max volume and $FE min volume - that's hex. That's 0 to 254 decimal. It doesn't say it, but I bet $FF is simply MUTE.)  
  
Now, check out [the Vinculum website](http://www.vinculum.com/documents.html) and download the Vinculum Firmware User Manual. This one manual covers several different firmwares. It is laid out well as a reference if you are already aware of a few things. First of all, most of the firmware varieties are based on the generic VDAP firmware, and special firmwares simply extend upon that. The VMUSIC2 uses the VMSC firmware, which provides commands specific to MP3 playback and communication with the VS1003.   
  
Section 6.8 of the firmware user manual is specific to the VMSC firmware. This is the section you will be most interested in. Sections 6.0 to 6.8 are universal commands, mostly for disk access, file access, and maintenance. There are also some bits of information about using other peripherals in the USB port, such as printers, HID devices, etc.. Table 3.1 lists the various device classes supported by each firmware.  
  
Only port2 is connected on the vinculum, so limit your exploration to that port. It should be noted here that plugging your computer into the USB port on the VMUSIC may result in damage to either the VMUSIC2, your PC, or both. The hardware in the VMUSIC does not support this, even though the VNC1L chip does.  
  
Also, the VMUSIC module should be able to support just about any USB Mass storage device. I have only tried it so far with a USB flash drive, but according to the manual, it should work with USB Hard drives, Cameras (in mass storage mode), card readers, etc... There are some restrictions, however. These are explained in the firmware user manual.  
  
It supports FAT12, FAT16, and FAT32, but FAT32 long filenames are not supported. Sector size must be 512; various cluster sizes seems to work fine. I tested this with an 8GB USB disk with no problems, but some of the commands will not output correct values when the drive is larger than 4GB.  
  
Right now, we are mainly concerned with the VMSC specific commands as well as a few of the VDAP commands (for navigating the file system.)  
  
Again, I will revisit this section to help make some sense of all this. But, you now have enough information to start playing around with your module. I would basically be retyping the user manual, anyway, as it is pretty clear once you know where to look. I have successfully created a text file in the format of an INI file, inserted data into the separate sections, and then read it back one section at a time, all using hyperterminal only. It is very easy and I bet you'll have it all figured out before I even update this!  
  
Have fun!