Vinculum VMusic2 Module Compilation of Notes

This document contains my working notes for using the Vinculum VMusic2 module. Quoting from the VMusic2 datasheet (spelling fixed, however):

"The VMusic2 module is a product which not only lets you add USB Flash disk interfacing to your product, but also allows you to play back MP3 and other popular digital music formats directly from a USB flash disk. Only four signal lines plus 5V supply and ground are required to be connected. Extensions to the Vinculum VDAP command set allow you to play a selected file, as well as control the volume, balance, etc. of the sound channel, and monitor the status of the file being played. The Vinculum VDAP firmware also allows the VNC1L's I/O interface to be selected between the serial UART or SPI using on-board jumper pins. The VMusic2 is ideal for adding MP3 playback from USB drive capability to home entertainment and in-car audio systems, as well as other appliances requiring audio playback capability from USB Flash disks. Not only is the VDrive2 ideal for evaluation and development of VNC1L designs, but also its neat enclosure and attractive quantity discount structure makes this module suitable for incorporation into finished product designs.
The Vinculum VNC1L is the first of F.T.D.I.'s Vinculum family of Embedded SoC USB host controller integrated circuit devices. Not only is it able to handle the USB Host Interface, and data transfer functions but owing to the inbuilt MCU and embedded Flash memory, Vinculum can encapsulate the USB device classes as well. When interfacing to mass storage devices such as USB Flash drives, Vinculum also transparently handles the FAT file structure communicating via UART, SPI or parallel FIFO interfaces via a simple to implement command set. Vinculum provides a new cost effective solution for providing USB Host capability into products that previously did not have the hardware resources available.

The English is pretty good, but you can tell that it is not written by someone who has English as their primary language. The rest of the document suffers in clarity in some areas as a result.

Most of the information presented here is from the Vinculum and FTDI websites and documents, and I simply have consolidated the info here and cleaned it up for easier reference.

See the following Internet references for more information, programming files, programs, cool products, etc:

www.ftdichip.com/Products/EvaluationKits/TTL-232R-3V3.htm - USB to 3.3V TTL compatible RS232 cable. Useful for talking to the VMusic2 module from a PC and re-programming the firmware in the Flash memory.

http://www.vinculum.com/documents.html - List of all documents for Vinculum products, including the VMusic2 module.

http://www.vinculum.com/documents/datasheets/DS_VMUSIC2.pdf - Latest datasheet for the VMusic2 module.

http://www.vinculum.com/documents/schematics/VMUSIC2%20Schematic%20Prints.pdf – Schematics of the VMusic2 module.

http://www.vinculum.com/documents/fwspecs/Vinculum%20Firmware%20User%20Manual%20V2.1%20Rev%202.pdf – Latest firmware documentation.

http://www.vinculum.com/downloads/vprog_com.zip - ZIP file of com port firmware programmer for VMusic2 module.

http://www.vinculum.com/documents/appnotes/ANVNC1L-01-VinculumBootloader.pdf - Documentation on the bootloader software for re-loading the firmware to the latest version.

http://www.vinculum.com/downloads/firmware/VMSC1FUL_V3_56.ROM - Latest firmware code executable. Must be put into Flash memory in module using firmware loader software such as the VPROG_COM program.

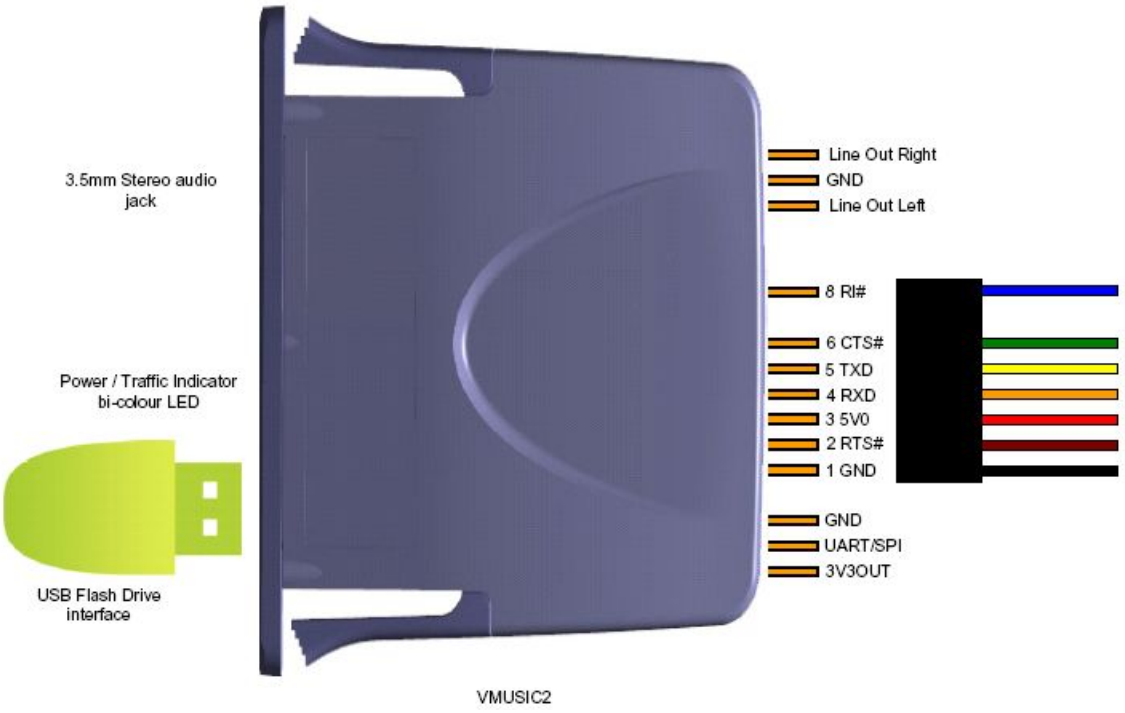
http://www.vlsi.fi/vs1003/vs1003.shtml - Documentation for the VLSI VS1003 MP3 player chip used in the VMusic2 module.

http://www.vlsi.fi/datasheets/vs1003.pdf - Datasheet for the VLSI VS1003 MP3 player chip used in the VMusic2 module.

http://www.maxstream.net/support/xctu/setup_x-ctu.exe - Freeware program used for terminal to talk to VMusic2 module.

Hooking up the VMusic2 Module to a PC for testing

Below is a picture from the VMusic2 datasheet:



It shows the top view of the VMusic2 module. Of note is the pinout of the module. I have only used the module in UART mode, so there needs to be a jumper installed from 3V3OUT pin to UART/SPI pin to work with the information I have supplied here. The module should have been shipped with this jumper already in place. Also of note is that these pins are on 2mm centers, not the ubiquitous 0.1 inch standard. That means DON'T LOSE THE JUMPER! All the rest of my electronics stuff uses 0.1 inch jumpers, so this is the only 2mm jumper I have (not including the other one inside the module that we will see further in these notes).

From the datasheet, these are the pin definitions:

Pin No.	Name	Type	Description
1	GND	PWR	Signal ground
2	RTS#	Output	Request To Send Control Output / Handshake signal
3	5V0	PWR	+5VDC supply input
4	RXD	Input	TTL-level RS232 Receive asynchronous data input
5	TXD	Output	TTL-level RS232 Transmit asynchronous data output
6	CTS#	Input	Clear To Send Control Input / Handshake signal
7	NC		No Connect
8	RI#	Input	Ring Indicator Control Input. Used to resume the Vinculum from suspend.

The module communicates using TTL-compatible RS232, +3.3V or +5V levels. DO NOT HOOK UP THE MODULE DIRECTLY TO A P.C.! A cable I have been using is from the makers of the chips in this module, FTDI. Their TTL-232R-3V3 USB to TTL RS232 converter cable is convenient to use, but you must figure out how to transfer from the 0.1 inch pitch cable to the 2mm cable supplied with the VMusic2 module. I simply cut off the end of the supplied VMusic2 cable and tinned the ends of the wires and stuffed the two serial communications pins into the TTL-232R-3V3 cable appropriately (TXD to RXD, RXD to TXD, GND to GND). The default baud rate is 9600 baud, although this can be changed (explained later). The data format is 8 data bits, 1 start bit, 1 stop bit, and no parity with RTS/CTS hardware handshaking enabled. However, I simply tied CTS# (pin 6) to ground (pin 1) and not use hardware control. This works as long as you do not send too much data (which a bit-banging microcontroller that is in sync with the module is not going to do ;).

Hook up the GND and +5VDC to an appropriate power supply. I use a PC supply that I have connected into the +5VDC and GND, with a load resistor on +5VDC of 5 ohms, 5 watt to keep the supply in regulation.

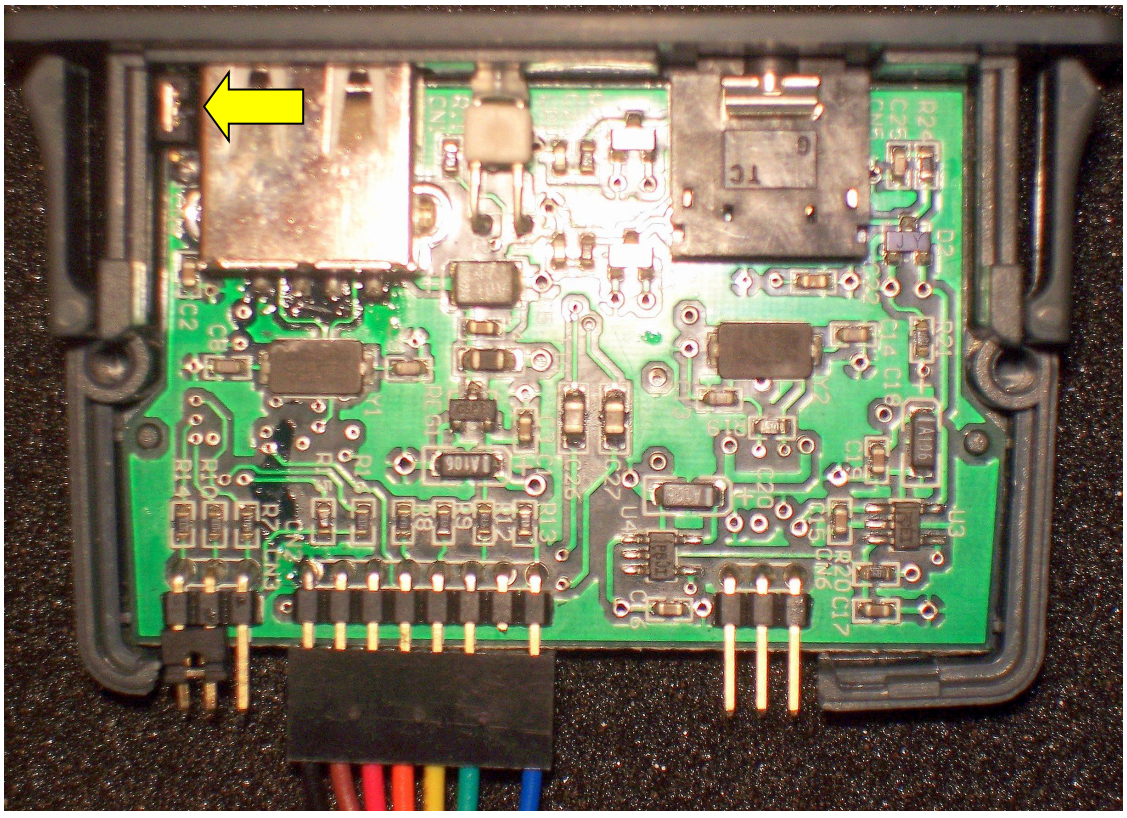
I use a program called X-CTU on a Windows XP machine to communicate to the VMusic2 module. It has a terminal tab that also allows “assembly of packets” and showing hex values. The VMusic2 doesn’t support backspace, so you must enter the command correctly the first time. The assembly of packets feature of the X-CTU program is convenient. See the references section for a link to this program (it is designed for X-Bee modules, but works just fine for a free terminal program ;).

Shipping version of VMusic2 Module vs. latest firmware:

It seems that the shipping modules have firmware version 1.12. The latest documentation is Version 2.1. My very first command (IPA) that I sent was not supported in V1.12.

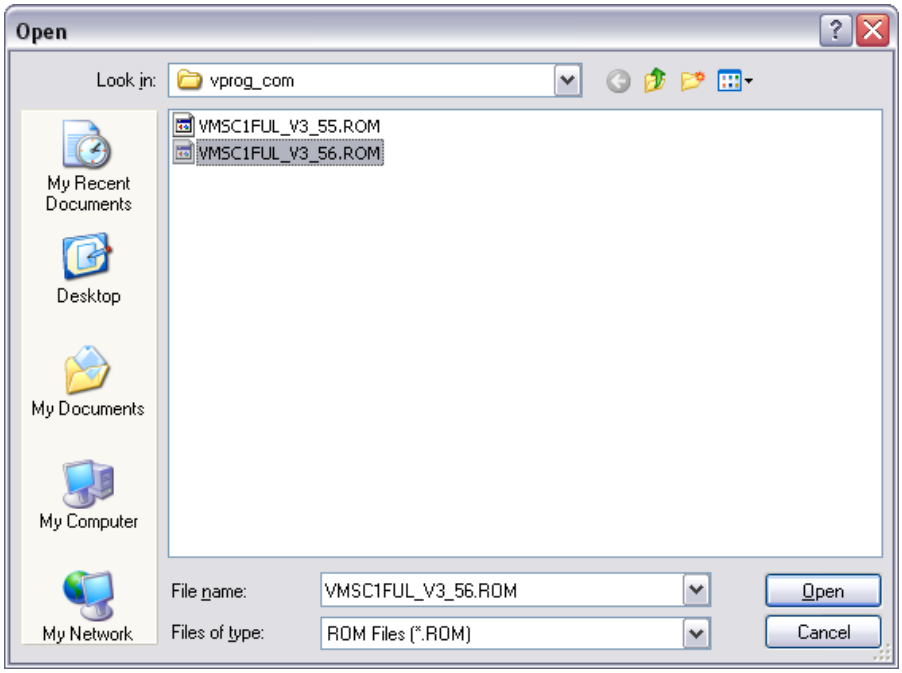
So it is best to download the latest firmware version from Vinculum's website (3.56 s latest when I did this), along with their VPROG_COM program for programming using FTDI's TTL-232R-3V3 USB to +3.3V compatible RS232 converter cable (VERY handy cable!). See the list of links for the latest firmware and the firmware programmer program VPROG_COM as well as link to the cable I used to talk to the VMusic2 module.

The module must have a jumper moved to re-program the firmware. Simply pop the lid off the module (friction fit, just hold the two halves, top and bottom, of the module and pull...) and find the jumper right beside the USB connector (see photo). Pull the jumper off the front and middle pins and place it on the middle and back pins. This now puts the module in a bootloader mode to re-load (program the Flash) the firmware.



Connect the VMusic2 module to the com port via a compatible (TTL-level!) RS-232 cable and run (left click twice) the program VPROG_COM.exe. Select the COM port that you have attached the VMusic2. The screen should look as below:

Now press the ROM File button and browse to the directory where you unzipped the latest firmware. I placed mine in the vprog_com directory:



Select Open, then press Program button. The VMusic2 module should now be programming, DO NOT TURN OFF POWER OR EXIT…. Wait until the programming completes. A success message should appear. If not, then try and figure out what is not hooked up right and that the jumper is in the right place.

After successful programming, change the jumper back so that the module will work in command mode again.

VMusic2 Module UART Monitor Commands

The VMusic2 module communicates using their firmware, which is called VDAP. The VMusic2 module uses the VMSC1FUL version of their VDAP firmware that supports the VS1003 MP3 player chip in the VMusic2 module.

The firmware is in "command" mode. If a suitable FTDI USB slave device has been found on USB Port 1 it can also be sent configuration commands. In this mode, a USB flash disk on USB Port 2 can also be accessed. The VMusic2 module doesn't have access to the other mode (called data mode).

The default conditions for the firmware after a reset are:

- Input numbers in HEX mode, and
- Extended Command Set entry

There are 2 commands to switch the way numbers are sent or received by the module for Version 2.0 and later firmware (NOT supported in the shipping Version 1.12 version). These are IPA (InPut ASCII) and IPH (InPut Hex). The default starting condition is HEX. If the command IPA is used then numbers can be entered as ASCII characters.

In HEX mode, the exact number of bytes specified for the command must be entered. In ASCII mode leading zeros are removed and the number is terminated by the carriage return (***<CR>*** = $0D). In this way, for ASCII mode:
12 == $C == $000000C == 0xC == 0Xc == 0x000C

For example, to read twelve bytes from an open file in IPH mode (HEX default mode):
'rdf' <sp> $00 $00 $00 $0C ***<CR>***

in terms of actual bytes sent to the module this would be:
$72 $64 $66 $20 $00 $00 $00 $0C $0D

To enter the same command in IPA mode (ASCII mode):
'rdf' <sp> 12 ***<CR>***

in terms of actual bytes sent to the module this would be:
$72 $64 $66 $20 $31 $32 $0D

Or, using the Hex format character '$' preceding the hex digits:
'rdf' <sp> $C ***<CR>***

in terms of actual bytes sent to the module this would be:
$72 $64 $66 $20 $24 $43 $0D

Or, using the Hex format '0x' characters preceding the hex digits:
'rdf' <sp> 0xC ***<CR>***

in terms of actual bytes sent to the chip this would be:
$72 $64 $66 $20 $30 $78 $43 $0D

VMusic2 Module's VS1003 MP3 Player Chip

The VMusic2 module contains VLSI's VS1003 MP3 player chip for playback of music files. It communicates to the chip through an SPI serial interface bus called SCI in the VS1003 literature. From the VS1003 datasheet (with spelling and grammar corrected, and some clarity added ;):

The VS1003 is a single-chip MP3/WMA/MIDI audio decoder and ADPCM encoder. It contains a high performance, proprietary, low-power DSP processor core (called VS DSP4), working data memory, 5 Kbytes of instruction RAM and 0.5 Kbytes of data RAM for user applications, serial control and input data interfaces, 4 general purpose I/O pins, a UART, as well as a high-quality variable-sample-rate mono ADC and stereo DAC, followed by an earphone amplifier and a common buffer. The VS1003 receives its input bitstream through a serial input bus, which it listens to as a system slave. The input stream is decoded and passed through a digital volume control to an 18-bit oversampling, multi-bit, sigma-delta DAC. The decoding is controlled via a serial control interface (SCI). In addition to basic decoding, it is possible to add application specific features, like DSP effects, to the user RAM memory.

The VS1003's serial control interface (SCI) is compatible with the SPI bus specification. Data transfers (reads and writes) are always 16 bits. The VS1003 is controlled by writing and reading of its registers across this interface.

The main controls accomplished across the SCI are:

- control of the operational modes, clock, and built-in effects
- access to status information and header data
- access to encoded digital data
- uploading user programs

There are 15 registers that can be accessed in the VS1003 through the UART interface of the VMusic2 module:

Reg Addr	Type	Reset Value	Abbrev	Description
0x0	R/W	0x800	MODE	Mode control of VS1003
0x1	R/W	0x3C3	STATUS	Status of VS1003
0x2	R/W	0	BASS	Built-in bass/treble enhancer
0x3	R/W	0	CLOCKF	Clock freq + multiplier
0x4	R/W	0	DECODE TIME	Decode time in seconds
0x5	R/W	0	AUDATA	Misc. audio data
0x6	R/W	0	WRAM	RAM write/read data
0x7	R/W	0	WRAMADDR	Base address for RAM Write/read
0x8	R only	0	HDAT0	Stream header data 0
0x9	R only	0	HDAT1	Stream header data 1
0xA	R/W	0	AIADDR	Start address of application
0xB	R/W	0	VOL	Volume control
0xC	R/W	0	AICTRL0	Application control register 0
0xD	R/W	0	AICTRL1	Application control register 1
0xE	R/W	0	AICTRL2	Application control register 2
0xF	R/W	0	AICTRL3	Application control register 3

Of these registers, I have only used two of them. I have listed them here for reference from the datasheet, again, with some edits:

DECODE TIME (RW)
When decoding correct data, the current decoded time is shown in this register in full seconds. The user can change the value of this register. But, when doing so, the new value should be written twice to guarantee the new value is accepted, if the player is playing. DECODE TIME is reset at every software reset and when WAV (PCM or IMA ADPCM), WMA, or MIDI decoding starts or ends.

VOL (RW)
VOL is a volume control for the player hardware. For each channel, a value in the range of 0..254 may be defined to set its attenuation from the maximum volume level (in 0.5 dB steps). The left channel value is to be multiplied by 256 and added to the right channel value for writing to this register. Thus, maximum volume is 0 and total silence is 0xFEFE. Setting VOL to 0xFFFF will activate analog power-down mode.

Example: For a volume of -2.0 dB for the left channel and -3.5 dB for the right channel: (4*256) + 7 = 0x407. Volume is set to full volume at startup. Resetting the software does not reset the volume setting.

Access to these registers is accomplished using the VRD and VWR commands described in the next section of these notes.

Selected VMusic2 Commands

Below are selected commands I am using, or will be using.

ASCII Command for Extended Command Set Mode	Hexadecimal Command for Shortened Command Set Mode	Command function	VMusic2 Firmware Response
Switching between Shortened and Extended Command sets			
'SCS'***<CR>***	$10,$0D	Switches to the shortened command set	This will return the prompt '>',$0D to indicate that the device is in shortened command set mode.
'ECS'***<CR>***	$11,$0D	Switches to the extended command set	This will return the prompt 'D:\>',$0D to indicate that the device is in extended command set mode.
'E'***<CR>***	'E'***<CR>***	Echo	This will return 'E',$0D for synchronization purposes
'e'***<CR>***	'e'***<CR>***	Echo	This will return 'e',$0D for synchronization purposes
'IPA'***<CR>***	$90,$0D	Input numbers in ASCII	<prompt>$0D (Only with new firmware 2.0 or greater)
'IPH'***<CR>***	$91,$0D	Input numbers in HEX	<prompt>$0D (Only with new firmware 2.0 or greater)
Command for UART Baud Rate mode change			
'SBD'<sp><divisor (3 bytes) LSB first >***<CR>***	$14, $20,divisor (3 bytes) LSB first >, $0D	Set Baud Rate (See Baud Rate Table)	<prompt>$0D
Responses to indicate if disk is online			
<CR>	$0D	Check if online	This will return the appropriate prompt or 'no disk' message for the current command set.
Response to Check if online for Extended Command Set Mode		If no valid disk is found	'No Disk',$0D
		If a valid disk is found	'D:\>',$0D
Response to Check if online for Shortened Command Set Mode		If no valid disk is found	'ND',$0D
		If a valid disk is found	'>',$0D

ASCII Command for Extended Command Set Mode	Hexadecimal Command for Shortened Command Set Mode	Command function	VMusic2 Firmware Response
Directory operations			
'DIR'***<CR>***	$01,$0D	Lists the current directory	A list of file names and directory names are returned. Each entry is terminated by $0D. A directory entry has <sp>'DIR' after the name and before the $0D.
'DIR'<sp> <name>***<CR>***	$01,$20, <name>, $0D	Lists the file name followed by the size. Use this before doing a file read to know how many bytes to expect.	$0D,<name><sp><size in hex (4 bytes) LSB first> $0D
File operations			
'RD'<sp> <name>***<CR>***	$04,$20,<name> $0D	Read file <name>	This will send back the entire file in binary to the monitor. The size should first be found by using the 'DIR' <sp> <name> ***<CR>*** command so that the expected number of bytes is known. <prompt>$0D
'RDF'<sp> <size in hex (4 bytes MSB first) >***<CR>***	$0B,$20,size in hex(4 bytes) , $0D	Reads the data of <size in hex(4 bytes) > from the current open file.	This will send back the requested amount of data to the monitor. <prompt>$0D
'DLF'<sp> <name>***<CR>***	$07,$20,<name> $0D	Delete file <name>	This will delete the file from the current directory and free up the FAT sectors. <prompt>$0D
'WRF'<sp> <size in hex (4 bytes MSB first) >***<CR>*** <data bytes of size>***<CR>***	$08,$20,size in hex(4 bytes) , $0D $data,$0D	Writes the data of <size in hex(4 bytes) > to the end of	<prompt>$0D

		the current open file.	
'OPW'<sp> <name>***<CR>***	$09,$20, <name>, $0D	Opens a file for writing to with 'WRF'	<prompt>$0D
'OPR'<sp> <name>***<CR>***	$0E,$20, <name>, $0D	Opens a file for reading to with 'RDF'	<prompt>$0D
'CLF'<sp> <name>***<CR>***	$0A,$20, <name>, $0D	Closes a file for writing.	<prompt>$0D

ASCII Command for Extended Command Set Mode	Hexadecimal Command for Shortened Command Set Mode	Command function	VMusic2 Firmware Response
VMUSIC commands (Only with VMSC1 firmware)			
'VPF'<sp><name> ***<CR>***	$1D, $20,<name>, $0D	Play an MP3 file	Sends file to SPI interface (I/O pins between VNC1 and VLSI1003) then returns <prompt>$0D
'VWR'<sp> <Address>(1 byte) <value (2 bytes) LSB first >***<CR>***	$1E, $20, <Address>(1 byte) <value (2 bytes) LSB first >,$0D	Write to command register of VS1003	<prompt>$0D
'VRD'<sp> <Address>(1 byte) ***<CR>***	$1F, $20, <Address>(1 byte), $0D	Read from command register of VS1003.	Returns 2 bytes followed by <prompt>$0D
'VST'***<CR>***	$20,$0D	Stop playing current track	<prompt>$0D
'V3A'***<CR>***	$21,$0D	Play all tracks with MP3 as the extension.	Sends all MP3 files in all sub directories to SPI interface then returns <prompt>$0D
'VSF'***<CR>***	$25,$0D	Skip to next track.	<prompt>$0D
'VSB'***<CR>***	$26,$0D	Skip to beginning of current track. If pressed twice within 1 second it will go to the beginning of the previous track.	<prompt>$0D

Baud Rate Table for SBD Command

Baud Rate	***1st Byte***	***2nd Byte***	***3rd Byte***
300	$10	$27	$00
600	$88	$13	$00
1200	$C4	$09	$00
2400	$E2	$04	$00
4800	$71	$02	$00
9600*	$38	$41	$00
19200	$9C	$80	$00
38400	$4E	$C0	$00
57600	$34	$C0	$00
115200	$1A	$00	$00
230400	$0D	$00	$00
460800	$06	$40	$00
921600	$03	$80	$00
1000000	$03	$00	$00
1500000	$02	$00	$00
2000000	$01	$00	$00
3000000	$00	$00	$00

* == Default

VMusic2 Module Firmware Documentation Errors:

I have found a couple errors in their firmware reference manual:

1.) Extra <CR> on VS1003 register reads.

The documentation says that it sends back two bytes (the VS1003 register's value) followed by the prompt (think DOS prompt) and a carriage return (<CR> character). HOWEVER, the VMusic2 firmware ACTUALLY sends the two bytes followed by a <CR>, THEN the prompt and ANOTHER <CR>.

2.) Sync bits for MP3 stream in VS1003 register $09 are NOT correct and do NOT update at end of song to $0000.

The VS1003 documentation I have says that this register will contain some header info for the MP3 file being played, and contain $0000 if no file is being played. I found that not to be the case. I was going to use this register to determine whether there was a current song (or sound for us droid builders!) playing, but the register keeps its last value after the song is done. The value isn't what I expected either. I do not know if this is a Vinculum issue, ESO, or a VLSI issue with the VS1003. In any case, I worked around it using the time status register ($04) and looking for a change in that register (updates every second).