

Project Final Report

zPlayer

04/27/2010

Michael Nazareno, XXXX@gwmail.gwu.edu*Abstract*

The zPlayer is an MP3 player based on the Zilog ZNEO Z16F flash microcontroller. The device takes MP3 files transferred over UART from a PC or stored on a Secure Digital (SD) card and streams them to a VLSI Solutions Oy VS1002d audio decoder chip for decoding and digital-to-analog conversion for audio output through a TRS connector. zPlayer supports the MP3 format, but other formats are supported as well as specified by the audio decoder chip. The purpose of this project was to learn embedded programming techniques and how to interact with other devices using various standard interfaces and protocols. Originally, this project was going to implement accelerometer input control support, Zigbee-based control communication, and a liquid crystal display (LCD) for information; however those proposed features were removed from this design to complete a design for this semester. Such features can be implemented after the course.

Status

The VS1002d MP3 decoder chip from VLSI Solutions Oy worked as planned, with that plan based on an example simple implementation interfacing the MP3 decoder chip with an ATMEL ATmega88 microcontroller. The ZNEO microcontroller used its Enhanced Serial Peripheral Interface (ESPI) to communicate with the MP3 decoder chip, which, with the preliminary version of the software, initially worked. VS10023 analog-to-digital conversion (ADC) was performed on-chip and the connections went directly to the TRS (tip, ring, sleeve) connector (audio jack). The SD/MMC (SecureDigital/MultiMediaCard) card reader also required use of a Serial Peripheral Interface (SPI). An additional SPI was to be software emulated via a bit-banging technique. Unfortunately, this solution did not work out within the timeframe of the project.

In conjunction with failing to interface with the card reader, implementation of a hardware abstraction layer (HAL) along with FAT16-based routines for the file allocation table file system also did

not work. Since data could not be successfully read from the SD card, audio data was provided as input over the ZNEO microcontroller's universal asynchronous receiver/transmitter (UART) from terminal software running on the development PC. This was enough to successfully play out MP3 files to earphones inserted into the audio jack.

However, revisions to the project after achieving this working setup, namely to implement bit-banging, SD card reading, and the FAT16 file system, broke things. Without a version control system in place to revert to, it became too late to present a functional in-class demonstration of the zPlayer.

Specification

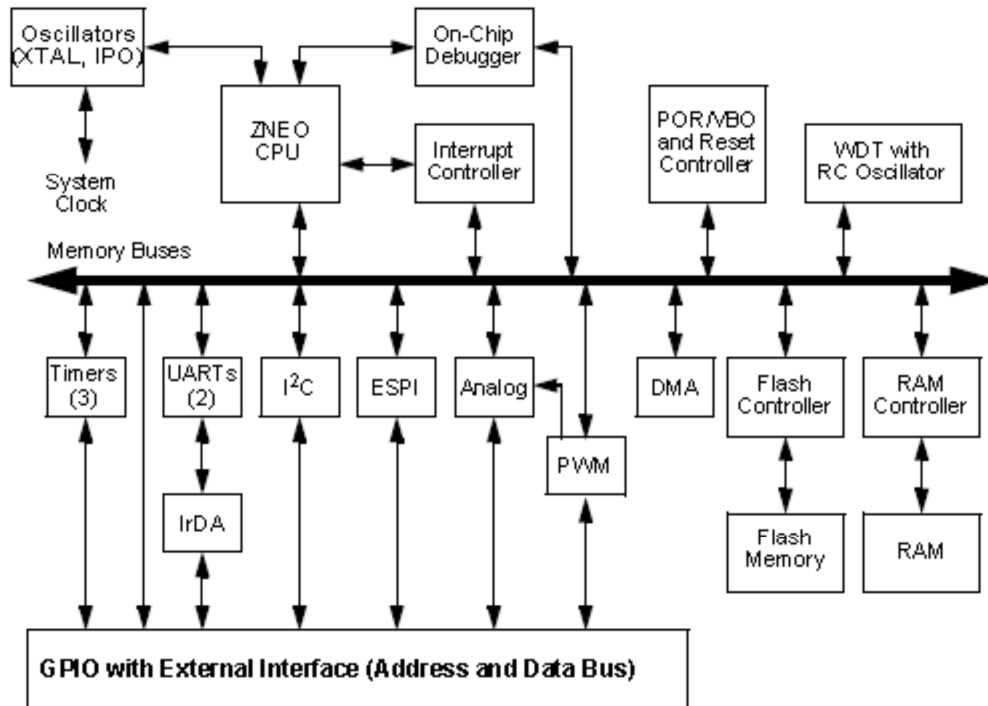
The zPlayer requires the following hardware and components:

- Zilog ZNEO Z16F series microcontroller contest kit
- VLSI Solutions Oy VS1002d MP3 decoder chip with SparkFun breakout board
- SD/MMC card reader with SparkFun breakout board
- Header pins
- Jumper wires
- Breadboard
- Serial cable

Suggested tools for this project include the USB Smart Cable for programming and debugging and a soldering iron.

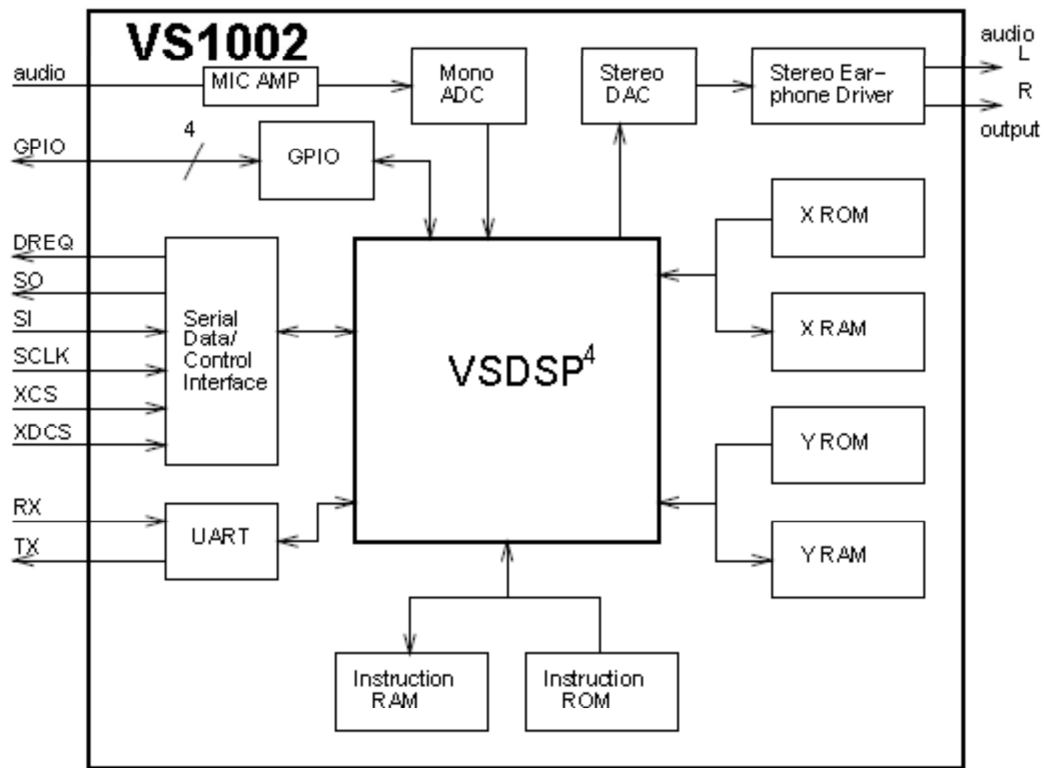
The Zilog ZNEO Z16G series microcontroller contest kit is based on Zilog's 16-bit CPU core. Capabilities of the contest kit board include an light emitting diode (LED) array, Inter-Integrated Circuit (I²C) and ESPI serial communication devices, an Infrared Data Association (IrDA) transceiver, UART, General Purpose Input/Output (GPIO), push-buttons, timers, and interrupts. The zPlayer design uses the ESPI, UART, GPIO, the LED array, timers, and interrupts.

The following is an overview of the architecture of the ZNEO Z16F microcontroller.



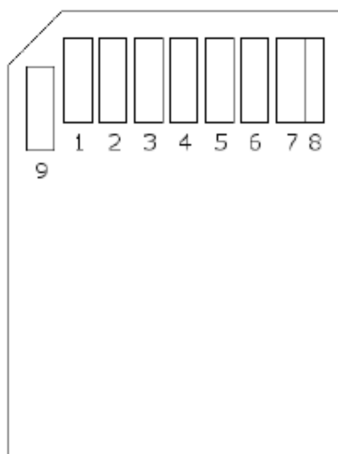
The VLSI Solutions Oy VS1002d MP3 decoder chip is itself a small microcontroller based on their VS_DSP core and thus programmable on its own, but for this zPlayer design, it will be functioning solely as a digital audio decoder and converter SPI slave device with the ZNEO as the master. In addition to SPI, capabilities of the VS1002d chip (on SparkFun's breakout board) include UART, GPIO, ADC, analog output, timers, and interrupts. The VS1002 is the oldest in VLSI Solutions Oy's class of audio chip decoders, with their current flagship product being the VS1053; however, for the purpose of this project, the features of the VS1002d are more than enough.

The following is an overview of the architecture of the VS1002 MP3 decoder.



The SD/MMC socket on SparkFun’s breakout board gives access to the headers of SD and MMC cards. The SD specification allows access to the cards by a couple of transfer modes, including 1-bit SD, 4-bit SD, and SPI, some of which are proprietary transfer modes. For this project, SPI transfer would be the interface used for communication with the ZNEO microcontroller.

The following illustrates the physical connections to the SD/MMC socket in SPI mode.



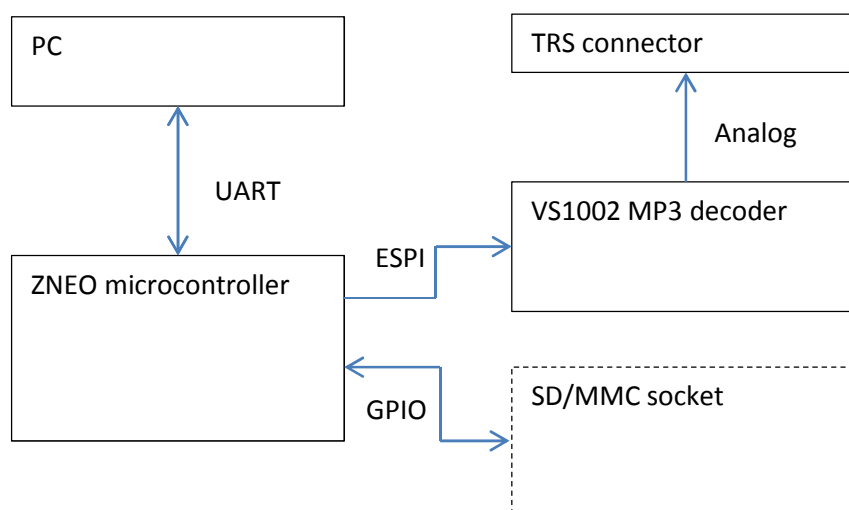
Pin	SPI Function
1	Chip select
2	MOSI
3	GND
4	VCC
5	SCLK
6	GND
7	MISO
8	NC (SD only)
9	NC (SD only)

Serial communication from a PC to the ZNEO microcontroller will be done using UART over the provided RS-232 connector along with a serial cable. Header pins, jumper wires, and a breadboard will be used to connect things. Header pins will have to be soldered onto the SparkFun breakout boards.

Software modules needed for the zPlayer include an SPI protocol for interfacing with the MP3 decoder, dataflow control for reading terminal input over the UART, and a software-emulated SPI protocol over GPIO pins via bit-banging to provide an additional SPI for a second SPI device, like the SD/MMC socket. A working FAT16 software module and accompanying HAL would be needed to provide SD card reading operation.

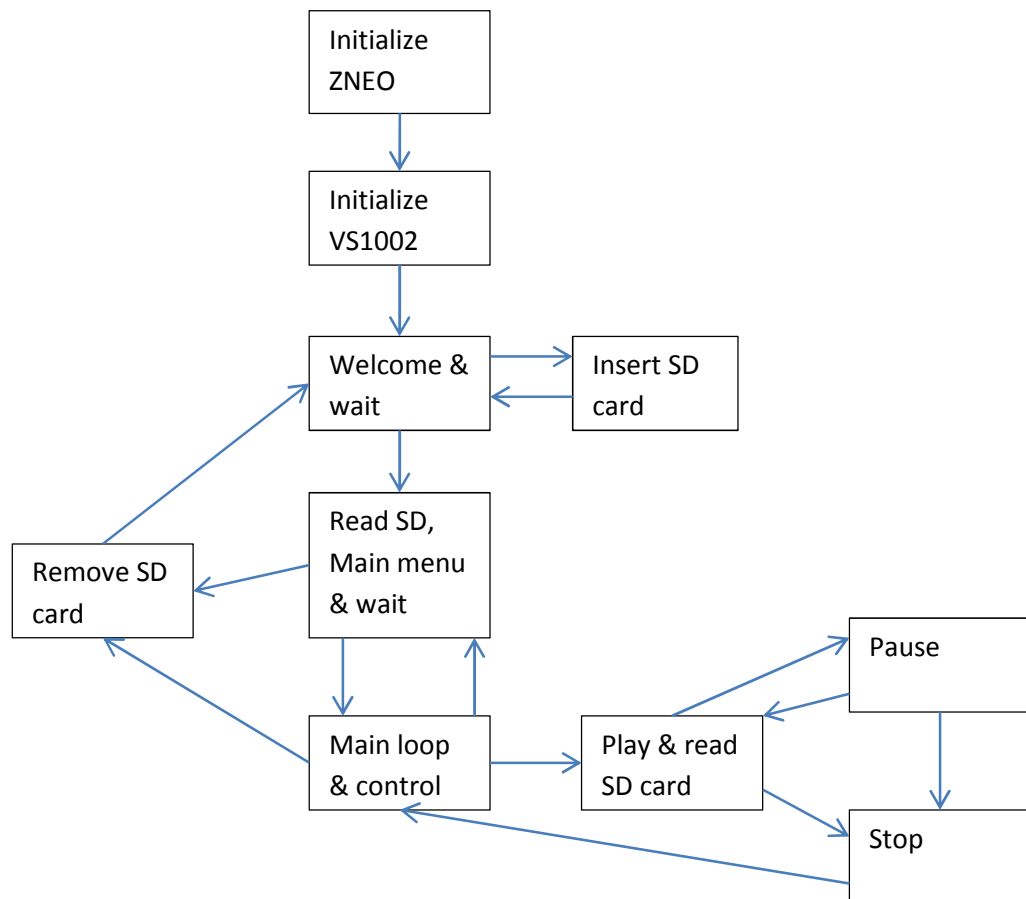
Implementation & Construction

The zPlayer implementation was been modified to remove the Zigbee, accelerometer, and LED screen modules. Therefore, the main work involved interfacing the VS1002MP3 decoder chip, the SD/MMC socket, and UART communication to a PC. The PC is connected via a serial cable to the UART of the ZNEO microcontroller. The VS1002 connects via jumper wires to the primary ESPI interface provided by the ZNEO's GPIO alternate functions. The SD/MMC socket connects to available unused GPIO pins used to provide another SPI to use for communication. The hardware block diagram is illustrated below.

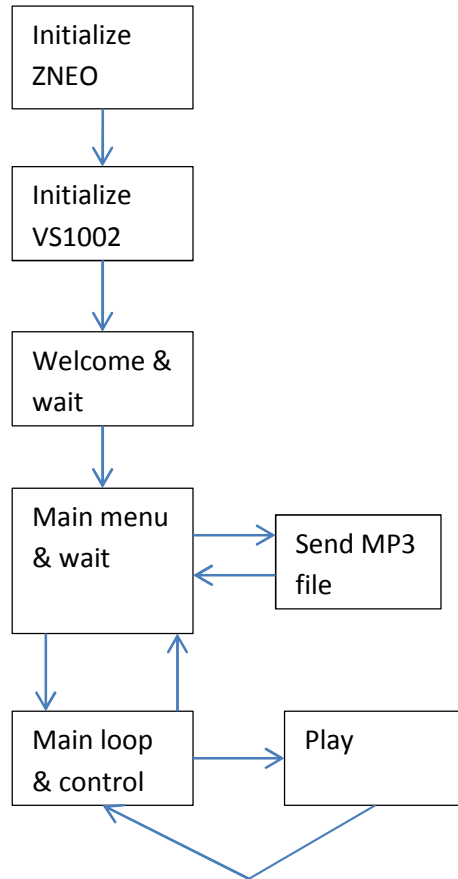


The dashed box around “SD/MMC socket” above is used to indicate the project’s optimal design from the final configuration. Ideally, the project would have successfully implemented the SD/MMC socket, the failure of which will be discussed in this document.

The software block diagram follows the hardware block diagram in a straightforward manner of operation. The priority of the project was to output analog audio given digital audio data in the MP3 format. The software block diagram would have ideally been implemented as follows.



Taking into account the failure to implement correct SD card functionality, that role has been simplified to UART transfer from a connected PC running terminal software to the RS-232 connector of the ZNEO. The modified resulting software block diagram is as follows

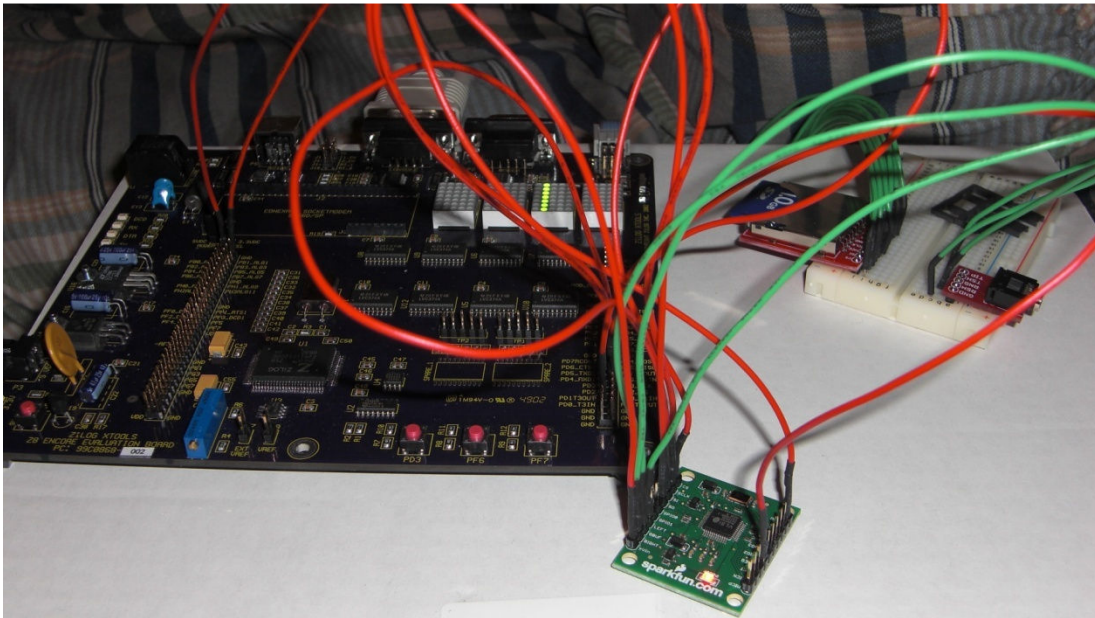


In this implementation, rather than reading MP3 data from an SD card and sending it to decoder unit, the information is instead simply streamed over UART. Playback control has also been simplified to accommodate a working design.

PC to ZNEO over UART

With a serial cable connected from a PC's RS-232 port to the RS-232 port on the ZNEO microcontroller, one can communicate with the ZNEO via a terminal application such as TeraTerm. Initialization of the UART within the ZNEO using the specified default configuration suffices for this project. Thus, one can use this for input/output. For this project, the terminal is used to display zPlayer instructions as well as receive MP3 data.

The ZNEO microcontroller is the master for this SPI configuration. The VS1002d is a slave device. In fact, it is two slave devices as it contains two SPI sub-modes: a serial command interface (SCI) and a serial data interface (SDI). The ZNEO provides its ESPI using the alternate function on four Port C GPIO pins. Pin 5 (PC5) is MISO and should connect to the SO of the VS1002d breakout board. Pin 4 (PC4) is MOSI and connects to SI. Pin 3 (PC3) is the SCK and connects to SCLK. Finally, pin 2 (PC2) is SS (slave select) and would connect to CS on the VS1002; however, since this signal is shared with the on-board DS1722 temperature sensor (active-high select), the chip select signal to the VS1002d will be done through an available GPIO pin, in this case Port D's pin 1 (PD1). The VS1002d requires a supply voltage of 3.3V, so a wire should connect a VDD pin on the ZNEO to the VS1002d's V_{in} pin. A ZNEO GND pin should connect to the VS1002 GND pin. Finally, another ZNEO GPIO pin (PD0) should connect to the DREQ pin on the VS1002.



In software, the ESPI should be initialized with phase 0 and clock polarity 0 to match the VS1002d's required timing. The baud rate should be less than 2Mbps to avoid glitches, so an ESPIBR value of 0x0008 works, which is the equivalent of 1.152Mbps.

The VS1002d is in fact two SPI slaves, the SCI and SDI. The SCI is used to send operational commands to get VS1002d to read/modify various settings such as modes, DSP features, volume, etc. The SDI is used to actually transmit raw audio (MP3) data for the VS1002d to decode. To communicate with the SCI, CS should be set low. Then over MOSI, a 1-byte opcode should be send to either read (0x03) or write (0x02) to a register (address). The next byte will be the address of the register to

read/write. Then depending on the operation, the next 2 bytes sent will come over MISO as the contents of the register for a read operation or continue to be sent over MOSI as the new value for the register for a write operation. To communicate with the SDI, CS should be set high, and then the data is simply sent. The MP3 file format consists of MP3 frames consisting of MP3 headers and data. The MP3 headers should be considered as just data as well in terms of information being sent over the SDI.

The main register of interest for the VS1002d is the 0x00 register: MODE. On reset, its value is 0x0800, i.e. VS1002d native SPI mode, which partially configures the functionality above. In addition, to this bit (bit 11), bit 6 (stream mode) and bit 10 (share SPI chip select) should be enabled (set to 1) as well.

As MP3 data bytes come from the UART, they can be forwarded to the VS1002d for decoding and audio output. Finally, the aforementioned DREQ signal is an input into the ZNEO active low that should be configured as an interrupt. This signal comes from the VS1002d to indicate that its internal memory buffer is full and that the host should stop sending bytes. When it returns high, dataflow may continue.

VS1002d to TRS Connector

The VS1002d features a high-quality on-chip stereo DAC. This stereo analog output comes out through the LEFT and RIGHT pins on the VS1002d SparkFun breakout board. In addition, a ground buffer (GBUF) is provided. LEFT should connect to the TIP pin on the TRS connector. RIGHT should connect to RNG, and GBUF should connect to GND. As MP3 data is decoded on the VS1002d, the decoder chip will take care of outputting it to the TRS connector to hear on speakers or headphones.

Milestones

The milestones for this project have been updated as follows:

1. Interface with MP3 audio decoder – completed 4/12.
2. File reading user interface through terminal – completed 4/12.
3. Interface with SD card reader – overdue.
4. Complete implementation – overdue.

The final project consisted of the following notable sources files:

- *main.c* – Program entry, system startup library function calls (initializers), and main loop; handles forwarding of MP3 bytes.
- *spi.c* – SPI send/receive.
- *mp3.c* – Interface to VS1002d; initializes the decoder chip and handles signaling CS and DREQ interrupt, sending instructions to SCI, and sending data to SDI.

Retrospective

Design decisions included how to interface with various peripheral devices over SPI. My line of thinking was since the onboard temperature sensor was consuming one select signaling line, I would have to use an extra GPIO pin to perform the mode selection. In a similar manner, to interface with SD/MMC socket, I attempted to emulate yet another SPI interface using bit-banging. This did not work for a reason not exactly identified; it may have been due to the bit-banging implementation or the SD/MMC file system handling source code.

Learned from this project was a great deal about reading datasheets. Seemingly cryptic at first, what you are looking for is somewhere in there. Of course, there is quite a large embedded development community on the Internet as well, so that became an invaluable resource as well. From this project, I learned more intimately SPI communication. I also became a better debugger out of it in resolving implementation issues not previously encountered during the course labs.

If I started this project over, other than better time management, I would have run more experiments on the SPI slave devices, and test out various connection set-ups. In addition, I would have liked to use a digital logic analyzer to test my SPI signaling and an oscilloscope to verify the MP3 decoder chip output.

Attachments

The following datasheets were important in this project.

- ZNEO Z16F Series Product Specification (<http://www.zilog.com/docs/zneo/PS0220.pdf>)
- VS1002 MP3 Audio Codec Datasheet (<http://www.vlsi.fi/datasheets/vs1002.pdf>)

The following references were used.

- <http://www.sparkfun.com>
- <http://www.penjuinlabs.com/blog/?p=42>
- <http://www.zws.com/products/dosfs/>
- <http://www.maxim-ic.com/app-notes/index.mvp/id/3969>