

Design Notes

Integrating An SD Card Into An Embedded Design

Ever wanted large amounts of non-volatile storage capacity in an embedded product at a relatively low cost? Do you also want the ability to easily remove and replace the memory? If so, the humble SD card is your answer.

A typical SD card can hold between 2GB and 32GB of data. It includes an on-board controller to manage data I/O, and offers features such as wear levelling, write delay management, and erasing, negating the need for the host controller to manage these issues.

An SD card can interface to a host controller using two methods. The high speed interface utilises the native SD Bus command and data lines to transfer data at relatively high speed. The alternative method is good old fashioned SPI bus utilising 4 lines, namely clock, data in, data out and chip select lines. The SPI interface makes it simple to interface to a host controller and access virtually all features and functions of the SD card.

There are low level drivers on the web for accessing the SD card, as well as higher level drivers for implementing a FAT file system. We recently developed a project using drivers from www.dharmanitech.com and found them very easy to understand and modify.

Hardware Interfacing With SPI

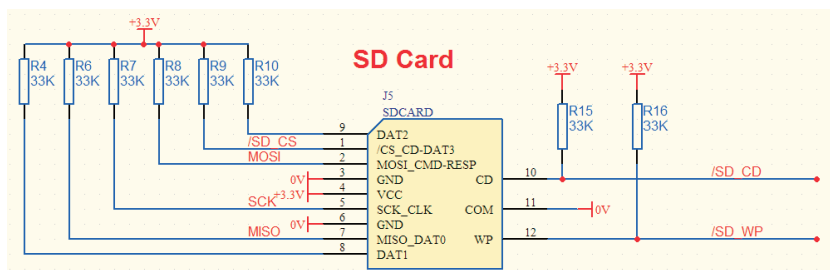
Interfacing to an SD card is simple: There are 9 contact pads (connections). Below is an SD card schematic symbol for the 12

pin Kyocera 5738 series spring loaded connector. Pin 12 is used for detecting if the lock switch on the SD card is active. Pin 10 is used to detect when the card is present (which is effectively shared with the CD pin of the card).

Four lines are utilised, all of which connect to the SPI interface of the host controller. CS pin 1, CLOCK pin 5, MOSI pin 2 and MISO pin 7. Unused lines are terminated with 10K-50K pull-up resistors. The SD card operates from 2.0V to 3.3V however some functionality requires a minimum of 2.7V. We suggest you power it off a 3.3V bus and it will work fine. Be aware the I/O lines are not 5V tolerant.

Software Drivers

The original software drivers from www.dharmanitech.com were written for an AVR processor. They were easy to follow, and were logically written in a layered format so the SPI drivers were separate from the



electronics by design

Electronics Design Services

- Microcontroller design
- Wireless Communications
- PDA software development
- TCP/IP, Ethernet development
- Embedded C & PC applications
- Protel & EMC/EMI compliance

Electronics By Design

W: electronicsbydesign.com.au
A: Suite 16, 469-475 Parramatta Road
Leichhardt NSW 2040
E: contactus@electronicsbydesign.com.au
P: (02) 8212-3951

remaining code. This allows the developer to easily modify the low level SPI drivers to suit the host controller.

We changed them to work with a Coldfire V1 processor and were able to read and write the SD card memory in the native format, hence using it as a storage medium without any further changes to software. At the time of writing we haven't tested the FAT32 drivers included with the package, but have no reason to doubt the author's claims they operate successfully.

The SPI initialisation phase requires the host micro controller to limit the SPI bus speed to 400kHz or less. We used 100kHz to be safe. Once initialisation is complete, the card is in SPI mode and remains in this mode until it's power cycled. At this point, the SPI bus speed can be dramatically increased. Initialisation is included in the above mentioned drivers and should not be modified.

The software communicates to a PC running a terminal emulator (HyperTerminal) via a serial link. This allows the user to choose a variety of options including writing the SD Card in native format, reading it, erasing it, reading and writing FAT32 and other useful functions. The software reads and writes the card in 512 byte sectors.

Step By Step Integration Guide

Here's a step by step guide on how to implement SD card functionality into your project.

- Design the SD card hardware by implementing the schematic symbol shown on the previous page. Your host controller is

the master, and the SD card is the slave. Series resistors in the data and clock lines are not required however pull-ups for all lines should be used (use a pull down for the clock line).

- We recommend using a good quality SD card connector such as a Kyocera 5738 series connector available from Farnell (P/N 155-8175). Generate a footprint for this connector and implement on your PCB. Remember to allow enough clearance in front of the SD card so there's room to remove and refit it.

- Import the software files from www.dharmanitech.com. SD_main.c is the main file. SD_routines.c and .h contain the SD card routines. FAT32.c and .h contain the FAT32 routines. SPI_routines.c and .h contain the SPI functions and will need to be changed to suit your host controller. UART_routines.c and .h contain code to talk to a host PC for testing and evaluation. After you complete integration and testing, the functions in these files can be discarded.

Once you have completed the above, you will be able to read and write the SD card.

If you wish to read additional information on SD card interfacing there are some great articles. You will need the Physical Layer Specification off www.sdcard.org. There's a similar document off the San Disk website. A good article explaining the FAT32 file system is www.pjrc.com/tech/8051/ide/fat32.html

Good luck and enjoy!

Author: Ef Misoyannis