SD card Tiva C library

Version 1.0

Javier Martínez Arrieta

martinezarrietajavier@gmail.com

Contents

Abstract

This is a short guide to be used as guidance in order to know how to use the library so as to read an SD card and (hopefully in the future) write in it. This library has been tested with the TM4C123GH6PM microcontroller, so take it into account in case you wish to use it with another microcontroller as there might be necessary to make some changes.

License

OR CONSEQUENTIAL DAMAGES, FOR ANY REASON WHATSOEVER.

For more information about my classes, my research, and my books, see

http://users.ece.utexas.edu/~valvano/

Acknowledgements

I would like to thank the people which work helped me to understand how the FAT32 filesystem works and also to convert it to the final code. Those people are:

-Paul Stoffregen for summarising how does FAT32 works at https://www.pjrc.com/tech/8051/ide/fat32.html

-Jonathan Valvano for the example code that I could make use of for some of the functions. The SD card example plus others are available at http://users.ece.utexas.edu/~valvano/arm/

-Those who wrote the content of http://elm-chan.org/docs/mmc/mmc_e.html

Files

There are three files used for this CCS project along with the Tivaware library, so in case you don't have it installed you will need it as some definitions come from that library. The main files are:

-main.c : This file contains references to the library files and example code.

-sdcard.h: This header file contains the definitions of functions and constants.

-sdcard.c: This file contains the implementation of the functions to read from the SD card.

Available functions

The functions available are the following:

-unsigned char sd_read(**enum** SSI SSI_number): Reads from the SD card by getting the data received through the SSI line.
-unsigned char is_ready(**enum** SSI SSI_number): Waits until the SD card is ready to be read.
-unsigned char send_command(**unsigned char** command, **unsigned long** argument, **enum** SSI SSI_number): Sends a command to the SD card.
-void initialize_sd(**enum** SSI SSI_number): Initialises the SD card to be read as SSI
-void startSSI0(): Configures SSI0 to read from the SD card.
-void startSSI1(): Configures SSI1 to read from the SD card.
-void startSSI2(): Configures SSI2 to read from the SD card.
-void startSSI3(): Configures SSI3 to read from the SD card.
-void tx_SSI(**enum** SSI SSI_number)
-void change_speed(**enum** SSI SSI_number): Changes the communication speed after the SD card has been initialized.
-void read_first_sector(**enum** SSI SSI_number): Reads the first sector of the SD card to get information required to be able to read from the SD card.

-**void** **read_disk_data**(**enum** SSI SSI_number): Gets information required to be able to read the SD card
-**long** **list_dirs_and_files**(**long** next_cluster,**enum** name_type name, **enum** get_subdirs subdirs, **enum** SSI SSI_number): Lists the directories and files that are found in the SD card, including the subdirectories if specified and showing the short or the long filename
-**long** **open_file**(**long** next_cluster,**enum** SSI SSI_number):Reads the content of the file specified. Please note that this fully works for txt files. In case of other type of files, it will be read but some action will be required with its content.
-**void** **clean_name**(): Removes the NUL characters from the directory or file name before showing to the user.
-**void** **Timer5_Init**(**void**): Initialises Timer number five.
-**void** **Timer5A_Handler**(**void**):Acknowledges the timer number five that the timeout has been reached.
-**void** **disk_timerproc**(**void**):Decrements the timers.
-**unsigned int** **rcvr_datablock** (**unsigned char** *buff, **unsigned int** btr, **enum** SSI SSI_number) : Receives a 512 byte block from the SD card through the SSI specified.
-**void** **rcvr_spi_m**(**unsigned char** *dst,**enum** SSI SSI_number):Reads a block from the SSI.

Bugs found

If using more than one SSI, I discovered that when configuring the clock of a second SSI, the clock of the first one was disabled (e.g. if startSSI0 is used, if startSSI1 is called, then SSI0 will not be usable). During the time I discover how to solve the problem, there is one thing that can be done, which is changing one line. If, for example, SSI0 and SSI1 are going to be used, go to startSSI1 in sdcard.c and change `SYSCTL_RCGCSSI_R|=SYSCTL_RCGCSSI_R1;` to `SYSCTL_RCGCSSI_R|=SYSCTL_RCGCSSI_R0+SYSCTL_RCGCSSI_R1;`

Connections

There are four possible SSIs to be used in order to connect the screen. Depending on which SSI is used, the connection should be as follows:

SSI0

Board	SD card adapter
PA2	SCK
PA3	CS
PA4	MISO
PA5	MOSI
3.3V	VCC
GND	GND

SSI1

Board	SD card adapter
PF0	MISO
PF1	MOSI
PF2	SCK
PF3	CS
3.3 V	VCC
GND	GND

SSI2

Board	SD card adapter
PB4	SCK
PB5	CS
PB6	MISO
PB7	MOSI
3.3 V	VCC
GND	GND

SSI3

Board	SD card adapter
PD0	SCK
PD1	CS
PD2	MISO
PD3	MOSI
3.3 V	VCC
GND	GND