

1 Einleitung

Sie werden in diesem Praktikum das FAT File System (FATFS) kennenlernen. Informationen zum FATFS und seinem Application Interface (Funktionen, welche Sie in Ihrem C-Programm verwenden können) finden Sie auf http://elm-chan.org/fsw/ff/00index_e.html.

- In diesem Praktikum verwenden wir die HAL-Funktionen von STMicroelectronics.

2 Lernziele

- Sie verstehen, wie das FATFS funktioniert.
- Sie können mit Hilfe von FATFS Dateien auf einer SD Karte anlegen und eigene Messdaten in diese Dateien schreiben. Anschliessend können Sie diese Dateien auf Ihrem PC lesen.

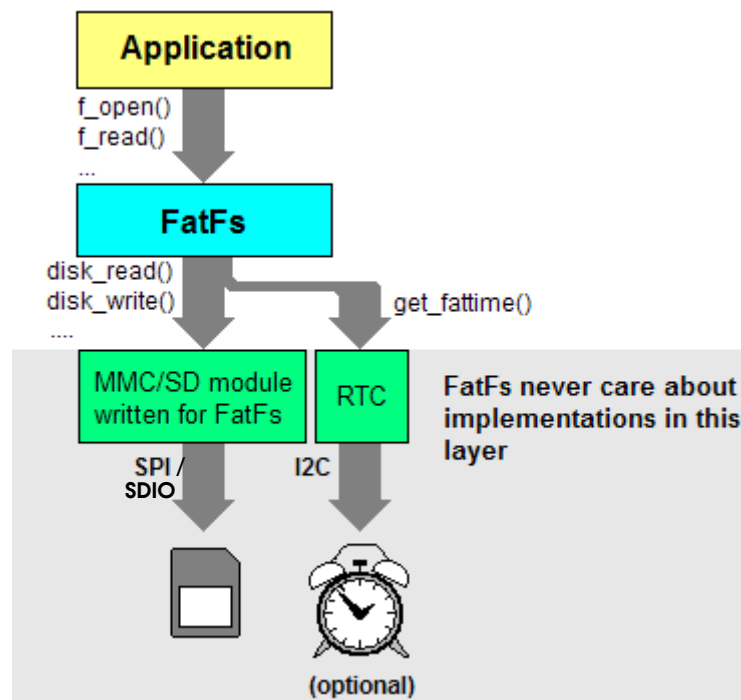


Abbildung 1: FatFs Single Drive System

3 Aufbau

3.1 Material

- 1 x Discovery Board auf Filesystem Board
- 1 x Accelerometer Board
- 1 x microSD Karte
- 1 x microSD Card Reader (für das Auslesen auf dem PC)

3.2 Hardware Plattform

Verwenden Sie für dieses Praktikum das Discovery Board mit dem Filesystem Board. Schliessen Sie das Accelerometer-Board an Port P7 an. Der Sensor des Accelerometer-Boards kommuniziert mit dem Discovery Board über die SPI Schnittstelle. Für die Verwendung der Buttons (T0-T3), der Schiebeschalter (S0-S3) und der LEDs (LED0-LED3) befinden sich im *mc1_fs_board.h* vordefinierte Makros (#define). Zuletzt befindet sich auf dem Board noch ein microSD Card Slot für die microSD Karte. Die SD Karte wird über die SDIO (Secure Digital Input Output) Schnittstelle angesprochen.

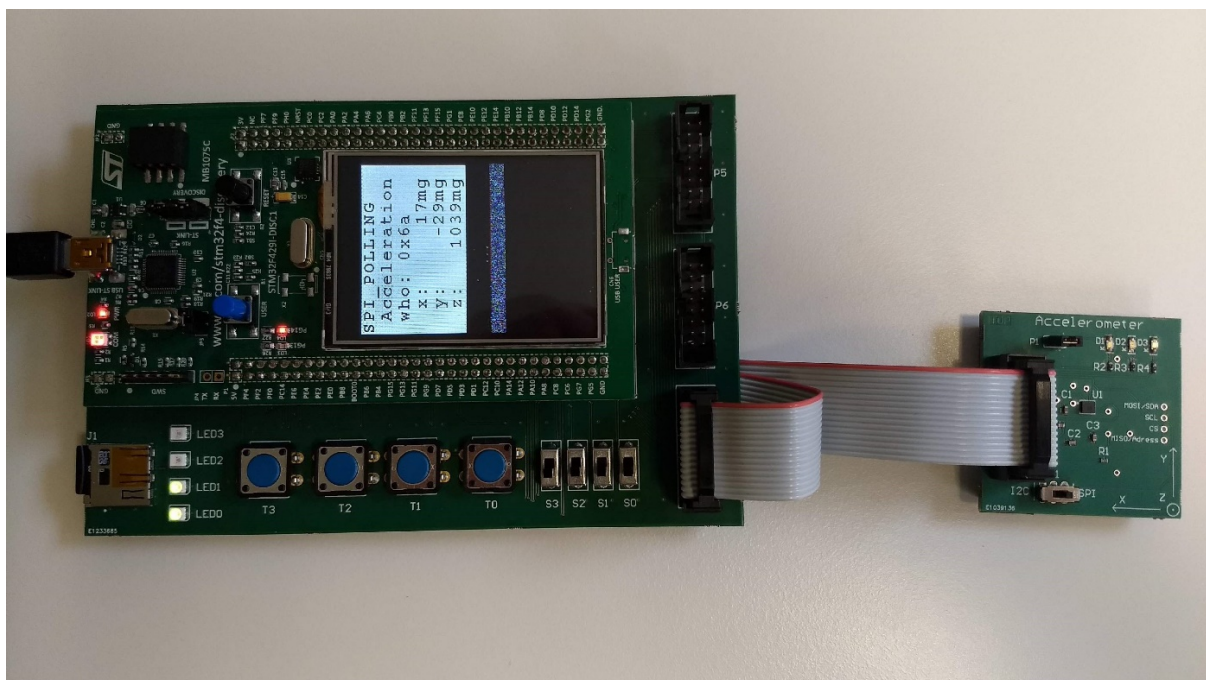


Abbildung 2: Hardware Aufbau

3.3 Programm Funktion

Verwenden Sie das Programm aus dem vorherigen Praktikum (P05/P06: Motion Sensor), in welchem das Accelerometer bereits initialisiert wird. Die Messwerte werden durch das Hauptprogramm über die SPI Schnittstelle ausgelesen. Ein Interrupt des Accelerometers signalisiert dem Hauptprogramm, dass der FIFO im Accelerometer gefüllt ist und somit die Messdaten ausgelesen werden können. Jedes Mal, wenn ein Interrupt detektiert wird, wird die rote LED (PG14) auf dem Discovery Board getoggelt.



Zum einen werden die Messwerte aus dem FIFO des Sensors in das File **raw_values.csv** und zum anderen die ermittelten Durchschnittswerte in das File **average_values.csv** geschrieben. Damit nicht ein Grossteil der Laufzeit für Schreibvorgänge verwendet wird, werden die Werte nur in den beiden FIFO Modi (SPI_LSM_FIFO & SPI_DMA_FIFO) auf die SD Karte geschrieben.

- Hinweis: Bei diesem Praktikum wird beim Kompilieren die Code-Limitierung der kostenlosen Keil-Version von 32KB erreicht. Folgen Sie den Anweisungen auf ennis.zhaw.ch um die **Keil uVision FlexLM** Lizenz zu installieren.

4 Aufgaben

4.1 Registrieren und Deregistrieren des Filesystems (mount / unmount)

Bevor auf Dateien der SD Karte zugegriffen werden kann, muss zuerst das FAT-Filesystem der SD Karte beim FatFs Modul registriert werden. Dieser Vorgang erfolgt durch die **f_mount()** Funktion. Lesen Sie in der Dokumentation (unter: http://elm-chan.org/fsw/ff/00index_e.html) die Beschreibung dieser Funktion.

- Hinweis: Da das Projekt aus vielen Source-Dateien besteht, dauert das erste Kompilieren lange. Um nur die veränderten Dateien neu zu kompilieren, und so die Dauer zu verkürzen, sollen Sie den Befehl «Build» (F7)  anstatt «ReBuild»  verwenden.

Implementieren Sie die Funktion **file_io_register_filesystem()** im File **file_io.c** und greifen sie in der **main** Funktion darauf zu. Verwenden Sie die bereits angelegten Modulvariablen **SDFatFS** und **SDPath** und wählen Sie die *mounting option 1*. Falls bei der Registrierung ein Fehler erkannt wird (z.B. return value von **f_mount()**), dann soll der dazugehörige **FILE_IO_STATUS** Code zurückgegeben werden. Diese Codes befinden sich in der Datei **file_io.h**.

In der **main** Funktion soll dieser Rückgabewert mit der Funktion **halt_on_file_io_error()** getestet werden. Im Fehlerfall wird der dazugehörige Binärcode auf den LEDs in einer Endlosschleife ausgegeben (siehe **error.h** und Abbildung 5 im Anhang). Ein **FILE_IO_MOUNT_ERROR** kann ganz einfach erzwungen werden, in dem Sie die SD Karte nicht einsetzen. Diese Codes und die **halt_on_file_io_error()** Funktion sollen im ganzen Praktikum verwendet werden, um einen Fehler beim Dateihandling zu erkennen. Falls das Registrieren erfolgreich war, soll die LED0 durchgehend eingeschaltet werden.

Nach dem Mounten der SD Karte werden die Messwerte in einer Endlosschleife erfasst. Die Messungen können durch einen Druck auf den Taster T0 beendet werden. Anschliessend muss die SD Karte vom FatFS Modul sachgemäss deregistriert werden. Danach kann die SD Karte aus dem Halter entfernt werden.

Implementieren Sie das Deregistrieren des Filesystems am Ende der **main()** Funktion mit Hilfe von **file_io_unregister_filesystem()**. Bei erfolgreicher Deregistration sollen LED0 und LED1 beide ausgeschaltet werden (die Funktion von LED1 wird später implementiert). Im Fehlerfall sollen sie, wie oben beschrieben, den Fehler anzeigen lassen.

4.2 Anlegen eines Files

Legen Sie nach erfolgreicher Registrierung des Filesystems ein File auf der SD Karte an. Implementieren Sie dazu die vorgegebene Funktion `file_io_create()`. Verwenden Sie die Funktion `f_open()`. Der Name des Files ist mit `avg_path[]` im Programmrahmen bereits definiert. Falls ein File mit dem gleichen Namen existiert, soll es gelöscht werden und ein neues File erstellt werden.

In dieses File schreiben Sie dann eine Header-Line mithilfe der `f_write()` Funktion. Der Header soll wie in der Abbildung 3 dargestellt aussehen. In der ersten Spalte steht die Sample Nummer und in den restlichen Spalten stehen die Messwerte der X-, Y- und Z-Achse. Die Spalten sollen durch Strichpunkte getrennt sein. Vergessen Sie nicht, nach erfolgreichem Schreibzugriff das File mit `f_close()` zu schliessen. Falls das Erstellen der Datei erfolgreich durchgeführt werden konnte, soll die LED1 durchgehend leuchten.

	A	B	C	D
1	Average Sample Nr.	Acceleration X	Acceleration Y	Acceleration Z
2	1	-1	-27	1030
3	2	-1	-28	1028
4				
5				
6				

Abbildung 3: Header-Line bestehend aus den Spalten: Average Sample Nr., Acceleration X, Y und Z

Nach erfolgreichem Schreiben und Deregistration entfernen Sie die SD Karte und schliessen diese an Ihren PC an. Überprüfen Sie Ihr File sowohl in einem Texteditor als auch in Excel.

Falls das Öffnen, Beschreiben oder Schliessen des Files nicht erfolgreich ist, soll wieder die Funktion `halt_on_file_io_error()` benutzt werden, um den Fehler anzuzeigen.

4.3 Write Average Acceleration

Im nächsten Schritt sollen Zeilen mit berechneten Durchschnittswerten ins vorhin erstellte File geschrieben werden. Implementieren Sie dazu die vorgegebene Funktion `file_io_write_avg_to_sd_card()` und rufen Sie sie bei den `SPI_SINGLE_LSM_FIFO` und `SPI_DMA_LSM_FIFO` cases in der `main` Funktion auf. Ein einzelner Aufruf der Funktion öffnet das File, schreibt eine Zeile mit Messwerten hinein und schliesst das File wieder. Jede Zeile soll eine aufsteigende Sample Nr. haben.

- Der Schalter S1 muss oben sein, damit der Sensor in einem der beiden FIFO Modi betrieben wird.
- Achten Sie darauf, dass Ihr vorhin erstelltes File nicht gelöscht wird, sondern Ihre Messwerte beim File angehängt werden.
- Verwenden Sie für das Umwandeln der Daten in einen String die `snprintf()` Funktion.
- Mit der Funktion `strlen()` können Sie Länge in Bytes eines char Arrays bestimmen.
- Im vorgegebenen Code wird die Funktion solange aufgerufen, bis Sie die Taste T0 drücken.

Testen Sie Ihr Programm, indem Sie die SD Karte an Ihrem PC auslesen. Sie müssen dazu nach einigen Durchläufen die Taste T0 drücken. Das Toggeln der LED PG14 signalisiert einen Durchlauf.

4.4 Write Raw Acceleration Data

Erweitern Sie die Funktion `file_io_create()`, damit neben dem File für die Durchschnittswerte nun auch ein File für die «rohen» Messwerte erstellt wird. Der Header soll wie in der Abbildung 4 dargestellt aussehen.

	A	B	C	D	
1	Raw Sample Nr.	Acceleration X	Acceleration Y	Acceleration Z	
2	1	-2	-29	1033	
3	2	-4	-25	1028	
4					
5					

Abbildung 4: Header-Line bestehend aus den Spalten: Raw Sample Nr., Acceleration X, Y und Z

Implementieren Sie die vorgegebene Funktion `file_io_write_raw_to_sd_card()`. Dabei wird der ganze FIFO Inhalt (ohne den Mittelwert zu berechnen) ins File gespeichert. Eine Zeile pro Messwert, d.h. die Funktion iteriert über die Länge des Buffers.

Testen Sie wiederum durch Auslesen am PC. Im File `raw_values.csv` sollte es 100 Mal mehr Einträge haben, da für jeden Durchschnittswert 100 «rohe» Messwerte verrechnet wurden.

5 Bewertung

Das Praktikum wird mit maximal 3 Punkten bewertet:

- Registrieren und Deregistrieren des Filesystems (mount / unmount) sowie Anlegen eines Files 1 Punkt
- Write Average Acceleration 1 Punkt
- Write Raw Acceleration Data 1 Punkt

Punkte werden nur gutgeschrieben, wenn die folgenden Bedingungen erfüllt sind:

- Der Code muss sauber strukturiert und kommentiert sein.
- Das Programm ist softwaretechnisch sauber aufgebaut.
- Die Funktion des Programmes wird erfolgreich vorgeführt.
- Der/die Studierende muss den Code erklären und zugehörige Fragen beantworten können.
- Der Zusatzpunkt wird nur vergeben, wenn alle anderen Aufgaben gelöst sind.

6 Anhang

6.1 File IO Errorhandling

	LED3	LED2	LED1	LED0
FILE_IO_MOUNT_ERROR	TOGGLE	OFF	OFF	OFF
FILE_IO_OPEN_ERROR	TOGGLE	OFF	OFF	TOGGLE
FILE_IO_WRITE_ERROR	TOGGLE	OFF	TOGGLE	OFF
FILE_IO_CLOSE_ERROR	TOGGLE	OFF	TOGGLE	TOGGLE

Abbildung 5 Zusammenstellung Ausgabe FILE_IO_ERROR auf LEDs