

### 1. Einleitung

Im ersten Teil dieses Praktikums konfigurieren Sie einen Beschleunigungssensor über SPI und lesen die Messwerte aus. Im zweiten Teil verlagern Sie die Steuerung der SPI Zugriffe von der CPU auf einen Direct Memory Access (DMA) Controller. Zusätzlich nehmen Sie das FIFO im Beschleunigungssensor in Betrieb. Abbildung 1 zeigt den verwendeten Test Aufbau.

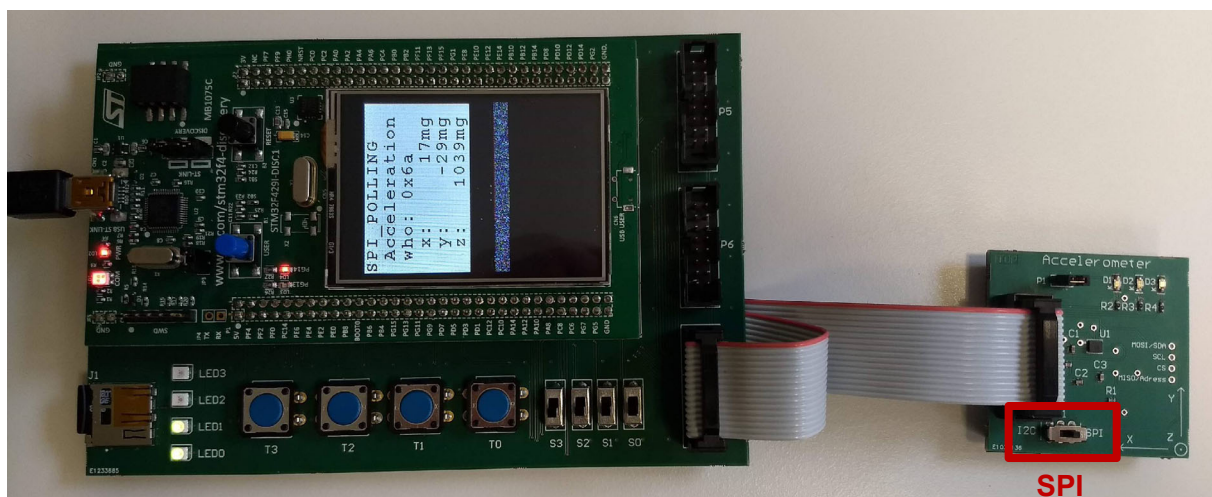


Abbildung 1: Test Aufbau

### 2. Lernziele

- Sie können einen über SPI verbundenen Sensor konfigurieren und auslesen.
- Sie können einen DMA Controller für Datentransfers konfigurieren.
- Sie können ein FIFO im Sensor konfigurieren und einsetzen.

### 3. Aufbau

#### 3.1. Material

- 1 x Filesystem Board
- 1 x Accelerometer-Board
- 1 x Flachbandkabel

#### 3.2. Programm Funktion

Das Filesystem-Board kommuniziert per SPI mit dem Beschleunigungssensor. Der Sensor misst die Beschleunigung in den 3 Achsen x, y und z in mg ( $1g = 9.8m/s^2$ ). Das Programm ist so vorbereitet, dass nach erfolgreicher Einstellung der Sensorkonfiguration, Messungen ausgeführt und auf dem LC-Display dargestellt werden.



Abbildung 2: SPI Beschaltung

Implementieren Sie die Funktionalität der Anwendung in der Reihenfolge, welche in dieser Praktikumsbeschreibung vorgegeben ist. Lesen Sie jeweils die gegebenen Kommentare im Code **bevor** Sie mit der Implementation beginnen.

Benutzen Sie die Unterlagen aus der Vorlesung.

## 4. Sensor konfigurieren und Messwerte auslesen

### 4.1. Lesezugriff auf Sensor-Register

Im ersten Schritt implementieren wir eine SPI Kommunikation zwischen Microcontroller und Sensor. Diese können wir testen indem wir das vordefinierte Sensor-Register WHO\_AM\_I (read-only) lesen.

- a) Suchen Sie das Register im Datenblatt und bestimmen Sie die Adresse für das Lesen sowie den erwarteten Rückgabewert.

Registeradresse für Lesen 0x0F Erwarteter Rückgabewert 0x6A

- b) Implementieren Sie die Funktion `hal_acc_spi_read_write()` im Modul `spi`. Die Initialisierung des SPI4 ist in diesem Modul bereits gemacht und wird aus `main()` heraus aufgerufen.
- c) Implementieren Sie die Funktion `accelerometer_who_am_i()` im Modul `accelerometer`. Diese liest das Register WHO\_AM\_I aus.
- d) Testen Sie Ihre Implementation. Bei erfolgreicher Kommunikation wird auf dem LCD unter *who*: der gelesene Wert angezeigt. Die Implementation der Ausgabe ist vorgegeben.

### 4.2. Sensor konfigurieren

In diesem Schritt konfigurieren wir den Sensor über SPI so, dass er fortlaufend die Beschleunigung in drei Achsen misst. Der Sensor zeigt über eine Interruptleitung (PE15) an, dass die nächsten Messwerte der Achsen verfügbar sind.

Wir arbeiten hier im Modus **SPI\_SINGLE**. Dafür müssen sich beide **Schalter S1 und S0** in der Stellung **'unten'** befinden. Die anderen Modi werden erst später implementiert.

Implementieren Sie im Modul `accelerometer` die folgenden Punkte:

- a) Funktion `write_reg()` um Sensorregister über SPI zu schreiben.
- b) Schalten Sie den Interrupt EXTI15\_10 frei und setzen Sie den zugehörigen Priority Level gemäss Angaben in der Funktion `accelerometer_init()`.
- c) Konfigurieren Sie in der Funktion `accelerometer_init()` und in der Funktion `init_continuous()` die Register des Sensors basierend auf dem Datenblatt und den Kommentaren im Code. Macros für die Registeradressen finden Sie im File `lsm6dsl_reg.h`.

Hinweis: In der Funktion `init_continuous()` muss auf dem Sensor nur das Register für den Interrupt gesetzt werden.

- d) Führen Sie das Programm aus und kontrollieren Sie, ob der Interrupt ausgelöst wird. Setzen Sie dazu einen Breakpoint im `EXTI15_10_IRQHandler()` im Modul `stm32f4xx_it`. Wird der Breakpoint erreicht? Allenfalls messen Sie das Interrupt Signal des Sensors mit dem Oszilloskop.

Der IRQHandler setzt das Flag `data_ready` und zeigt damit dem Hauptprogramm an, dass Daten auf dem Sensor zum Auslesen bereit sind. Das Hauptprogramm pollt `data_ready` in einer Endlosschleife und führt zu gegebener Zeit das Auslesen durch.

### 4.3. Sensorwerte auslesen

In diesem Schritt lesen wir die Messwerte über SPI aus und stellen sie auf dem Bildschirm dar. Dabei besteht ein SAMPLE aus drei VALUES (x,y,z). Ein VALUE wiederum besteht aus zwei BYTES. Siehe Abbildung 3.

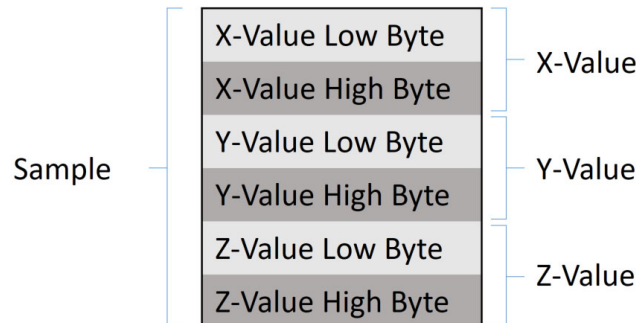


Abbildung 3: Sample vs. Values

Nach der Konfiguration im vorherigen Kapitel, implementieren Sie nun

- in der Funktion `accelerometer_read_acceleration()` den Case `'SPI_SINGLE'`.
- und rufen Sie im Hauptprogramm diese Funktion beim Case `'SPI_SINGLE'` auf.

Speichern Sie die Messwerte (Samples) fortlaufend im Array `acc_buffer`. Die vorgegebene Laufvariable `nr_of_samples` gibt Ihnen die Position des gegenwärtigen Samples vor. Wenn der eingestellte Schwellwert erreicht ist, erfolgt die Ausgabe auf dem LCD und die Speicherung der Samples beginnt wieder am Anfang des Arrays.

- Testen Sie Ihr Programm. Werden die Messwerte auf dem Display ausgegeben? Wenn Sie das Board flach auf den Tisch legen sollten die Werte im Bereich  $x \approx 0$  mg,  $y \approx 0$  mg,  $z \approx 1000$  mg liegen. Bewegen Sie das Board und beobachten Sie die Werte.

### 4.4. Mittelwert berechnen

Implementieren Sie die Funktion `calculate_acc_average()` um Mittelwerte über die Messwerte im Array `acc_buffer` zu berechnen und damit schnelle Wechsel zu unterdrücken.

Speichern Sie das Ergebnis im Speicherbereich, auf den der Pointer `result` zeigt.

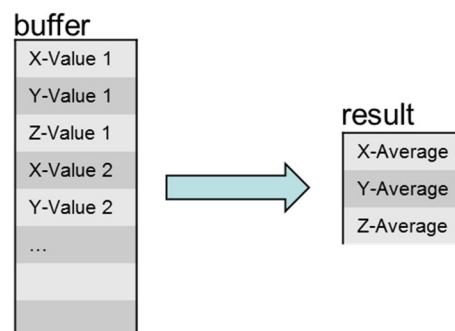


Abbildung 4: Berechnung der Mittelwerte

Testen Sie auch hier die Funktion des Programms. Vergleichen Sie, ob sich die Beschleunigungswerte stabiler verhalten, als ohne die auscodierte Funktion `calculate_acc_average()`.

Sehen Sie einen Unterschied? Ja, schwankt weniger

Zum Testen: Die Orientierung des Acc nicht verändern und dabei mit dem Finger darauf tippen. Die beiden Achsen, welche horizontal liegen, zeigen grössere Schwankungen, wenn kein Mittelwert gebildet wird.

Der Silkscreen (Beschriftung) auf dem PCB gibt eine andere Orientierung vor, als effektiv ausgegeben wird. Die X-Achse wird als negativ angezeigt, obwohl man den "Pfeil" nach oben hält. Dies hat unter anderem damit zu tun, dass der Silkscreen die Rechte-Hand-Regel missachtet.

## 5. SPI Transfers mit DMA

Nun verwenden wir für die Transfers der Daten vom Memory zum SPI4 und umgekehrt den DMA2 Controller. Dieser greift direkt auf das SPI read/write Register zu (SPI4->DR).

Stellen Sie dafür die Schalter S1 und S0 auf SPI\_DMA um.

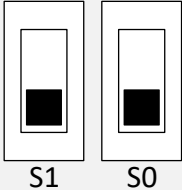
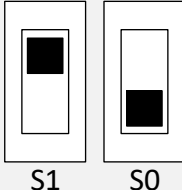
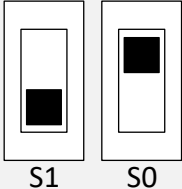
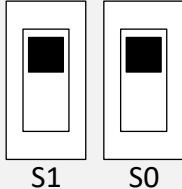
POLLING	SPI_SINGLE	SPI_LSM_FIFO
	 S1 S0	 S1 S0
DMA	SPI_DMA	SPI_DMA_LSM_FIFO
	 S1 S0	 S1 S0

Tabelle 1: Schalterstellungen und zugehörige Modes. Die Schalter werden nur beim Reset des Boards gelesen

### 5.1. DMA und Stream herauslesen

Suchen Sie in der Dokumentation das Request-Mapping des DMA2 Controllers und notieren Sie jeweils Channel und Stream für die Transfers.

Transmit:	Stream:	<u>1</u>
	Channel:	<u>4</u>
Receive:	Stream:	<u>0</u>
	Channel:	<u>4</u>

### 5.2. DMA Implementieren

a) Implementieren Sie die Funktion `dma_read_write()` im Modul `accelerometer`.

Verwenden Sie die Vorlesungsfolien als Beispiel.

b) Implementieren Sie den case `SPI_DMA` in der Funktion `accelerometer_read_acceleration()`. Rufen Sie dazu die oben erstellte Funktion `dma_read_write()` auf.

- c) Testen Sie Ihr Programm. Die Beschleunigungswerte werden nun wie vorher auf dem Display angezeigt. Setzen Sie innerhalb der Funktion `dma_read_write()` einen Breakpoint. Prüfen Sie damit, dass die Funktion auch tatsächlich ausgeführt wird.

### 5.3. Messungen von SPI Zugriffen ohne und mit DMA

#### SPI\_SINGLE (ohne DMA)

Wählen Sie mit den Schaltern SPI\_SINGLE aus und messen Sie mit einem Oszilloskop die Signale CS und SCL (GND befindet sich an Pin 14/15). Erstellen Sie ein Foto oder einen Screenshot Ihrer Messung.

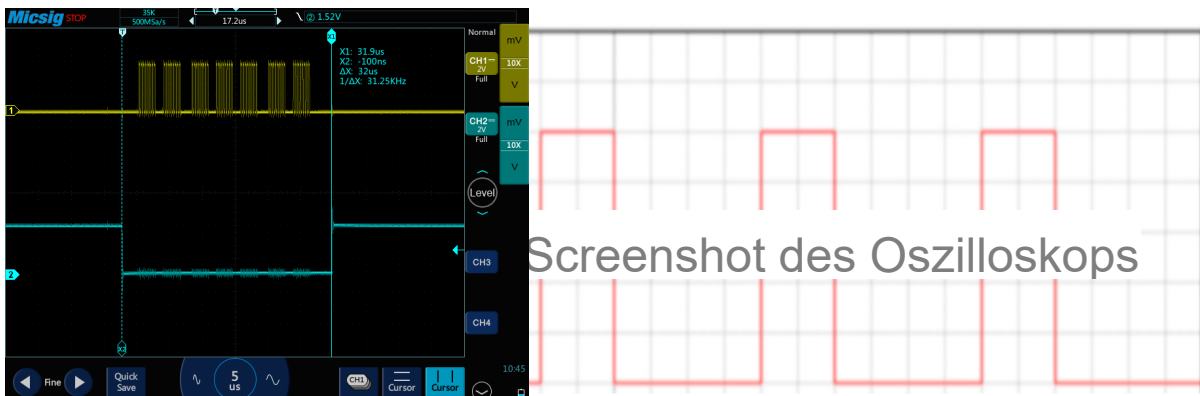


Abbildung 5 – SPI Zugriff Fall SPI\_SINGLE

Wie lange ist das Signal CS aktiv? Dauer 32us

#### SPI\_DMA (mit DMA)

Wählen Sie mit den Schaltern SPI\_DMA aus und messen Sie wiederum die Signale CS und SCL. Erstellen Sie ein Foto oder einen Screenshot Ihrer Messung.

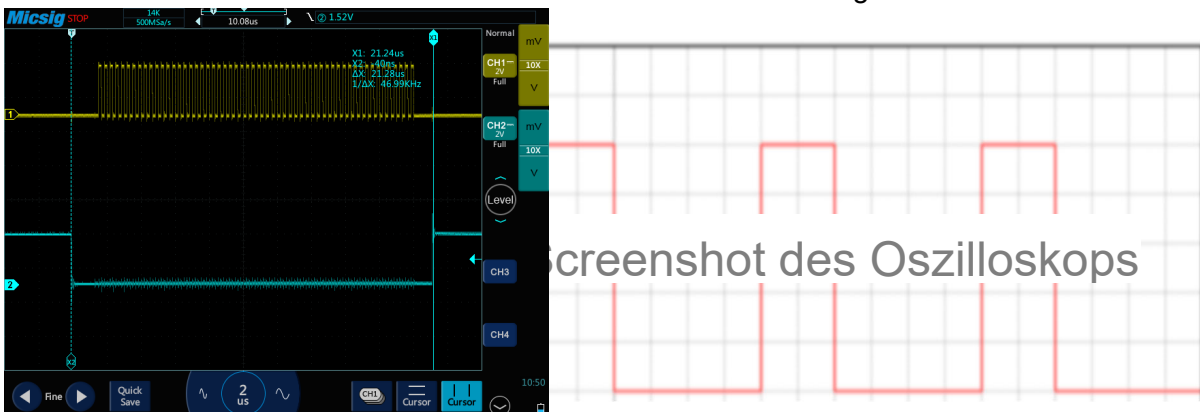


Abbildung 6 – SPI Zugriff Fall SPI\_DMA

Wie lange ist das Signal CS in diesem Fall aktiv? Dauer 21.28us

Wie unterscheidet sich der Zugriff vom Fall SPI\_SINGLE?

Durch DMA, durch nicht verwenden des Prozessors kann Zeit gespart werden, da DMA genau für diese Übertragung konfiguriert wurde

## 6. FIFO verwenden

Nun verwenden wir zusätzlich das FIFO auf dem Accelerometer. Das heisst, bei jedem Interrupt lesen wir nicht nur eine einzelne x,y,z Messung aus, sondern eine Vielzahl von Messungen. Damit kann Rechenzeit und Energie gespart werden.

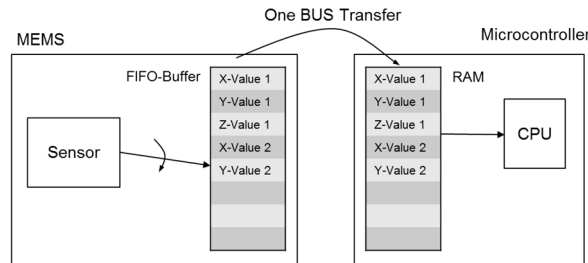


Abbildung 7: FIFO

### 6.1. FIFO ohne DMA

- a) Konfigurieren Sie das FIFO in der Funktion `init_fifo()` im Modul `accelerometer`.

Dazu müssen Sie 6 Register setzen: `FIFO_CTRL1...5` und `INT1_CTRL`. Lesen Sie die Kommentare im Code und das Datenblatt.

- b) Implementieren Sie in der Funktion `accelerometer_read_acceleration()` den Case `'SPI_SINGLE_LSM_FIFO'`.

Im Gegensatz zu vorher (ohne FIFO) liegt der erste Value bei der FIFO Variante bei `rx_buffer[0]` und nicht bei `rx_buffer[1]`.

- c) Implementieren Sie in `main.c` den Case für `SPI_SINGLE_LSM_FIFO`. Anders als bisher, werden im FIFO-Mode 300 Werte gleichzeitig verarbeitet.

Der Rückgabewert der Funktion `accelerometer_read_acceleration()` liefert die Anzahl empfangener Samples.

- d) Testen Sie den neu implementierten Mode mit den zugehörigen Schalterstellungen.

### 6.2. FIFO mit DMA

Nun wollen wir DMA und FIFO kombinieren.

- a) Implementieren Sie dazu in `accelerometer_read_acceleration()` den CASE `SPI_DMA_LSM_FIFO`.

Kombinieren Sie dabei die Cases `SPI_DMA` und `SPI_DMA_LSM_FIFO`.

- b) Testen Sie den neu implementierten Mode mit den zugehörigen Schalterstellungen.



## 7. Bewertung

Das Praktikum wird mit maximal 6 Punkten bewertet:

Lesezugriff auf Sensor-Register	1 Punkt
Sensor konfigurieren	1 Punkt
Sensorwerte auslesen und Mittelwert berechnen	1 Punkt
DMA Implementieren	1 Punkt
FIFO mit und ohne DMA	1 Punkt
Oszilloskop Screenshots und Interpretationen	1 Punkt

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

- a) Der Code muss sauber strukturiert und kommentiert sein.
- b) Das Programm ist softwaretechnisch sauber aufgebaut.
- c) Die Funktion des Programmes wird erfolgreich vorgeführt.
- d) Der/die Studierende muss den Code erklären und zugehörige Fragen beantworten können.