CT 2 – Praktikum

Ansteuerung eines TFT-Displays über SPI – Teil 1

1 Einleitung

In diesem Praktikum werden Sie auf dem CT System die Grundfunktionen für das Senden und Empfangen eines Bytes über SPI implementieren. Die SPI Grundfunktionen werden im nachfolgenden Praktikum für die Ansteuerung eines intelligenten Displays verwendet. Siehe Abbildung 1. Das Display wird erst im nächsten Praktikum angeschlossen.

Abbildung 1: TFT-Display mit SPI Schnittstelle

2 Lernziele

- Sie verstehen die verschiedenen Timing Optionen einer SPI Schnittstelle und können diese auf dem STM32F4 konfigurieren.
- Sie können aus Ihrer Software heraus einzelne Bytes über SPI versenden und empfangen.
- Sie können die übertragenen SPI Daten mithilfe des Oszilloskops analysieren
- Sie vertiefen Ihre Programmierkenntnisse in C

3 Schnittstelle

Die SPI-Schnittstelle wird verwendet, um mit einem intelligenten Display zu kommunizieren. Die SPI Signale werden auf dem CT Board über den Port 5 angesteuert. Abbildung 2 zeigt die PORT / Signal Belegung.

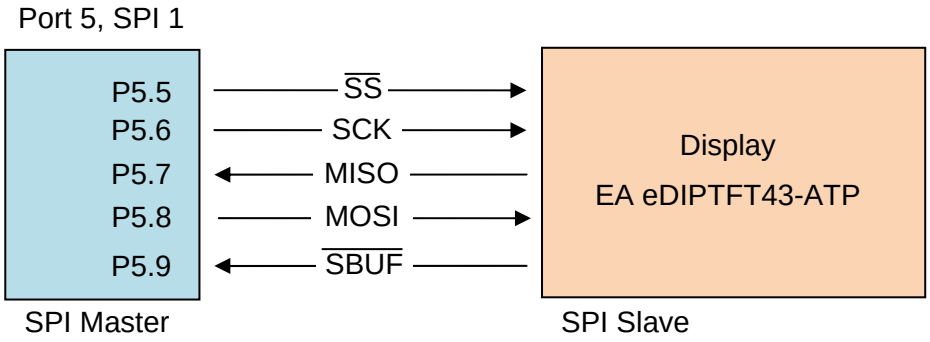

Abbildung 2: PORT / Signal Belegung

Eine Beschreibung des Port 5 finden Sie im InES CT Board Wiki (*www.ennis.zhaw.ch*).

Mit einem tiefen Pegel am Signal $\overline{\text{SBUF}}$ zeigt das Display an, dass im internen Buffer des Displays Daten zur Abholung durch den Microcontroller bereit stehen. Dieses Signal wird erst im nächsten Praktikum verwendet.

Für die Ansteuerung des Displays muss das SPI-Timing gemäss Abbildung 3 verwendet werden.

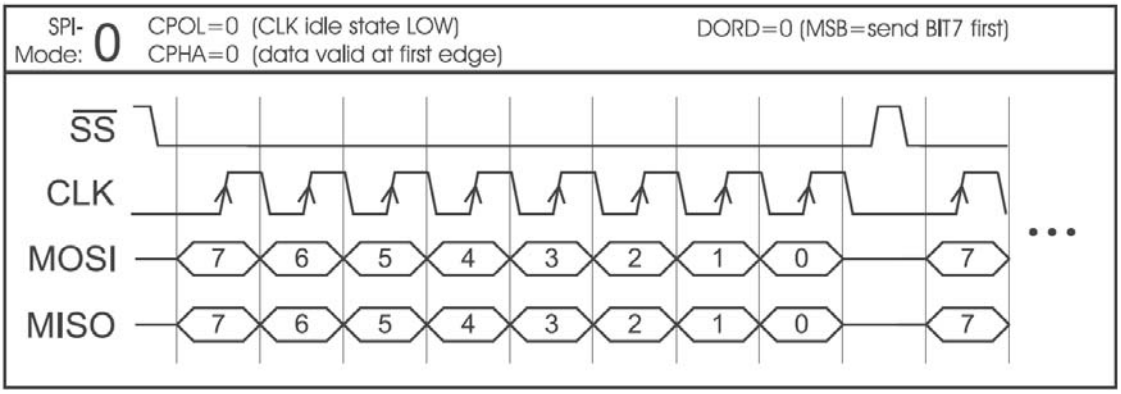

Abbildung 3: SPI-Timing

Das Display erlaubt bei einer pausenlosen Übertragung von Bytes eine Clockfrequenz von bis zu 200 kHz. Die SPI-Funktionseinheit ist im Rahmenprogramm auf einen Clock von 42 MHz eingestellt. D.h. f_{PCLK} gemäss Reference Manual ist gleich 42 MHz. Sie müssen daher in der folgenden Aufgabe den Prescaler der SPI-Schnittstelle entsprechend konfigurieren.

4 Aufgaben

Wir stellen Ihnen ein Testprogramm **`test.c`** sowie die Files **`hal_spi.c`** (Programmrahmen) und **`hal_spi.h`** (Header File) für die Implementation des SPI zur Verfügung.

4.1 Initialisierung

Die Funktion **`void hal_spi_init(void)`** zur Initialisierung der SPI Schnittstelle ist bereits teilweise implementiert. Die Konfiguration der GPIOs wurde bereits vorgenommen.

Überlegen Sie sich, welche Bits im Kontrollregister **`SPI1->CR1`** gesetzt werden müssen, damit das geforderte Timing des SPI-Displays eingehalten wird und vervollständigen Sie dann den Code.

Hinweise zur Initialisierung der SPI-Schnittstelle

- Im beigelegten Reference Manual finden Sie auf Seite 896 eine Beschreibung des Konfigurationsregisters CR1.
- Das Register CR1 kann über das bereits vordefinierte Macro

  `SPI1->CR1` angesprochen werden.
- Die Verdrahtung in Abbildung 2 erfordert den von ST als '2-line unidirectional data mode' bezeichneten Modus. Dabei muss full-duplex eingestellt werden, d.h. RXONLY = 0. Das Kontrollbit BIDIOE ist bedeutungslos und kann auf Null gesetzt werden (BIDIOE = 0).
- Das Display unterstützt keinen CRC (Cyclic Redundancy Check). Setzen Sie daher CRCEN = CRCNEXT = 0
- Im vorliegenden Fall wird das Slave Select Signal durch die Software über den GPIO PinA.5 auf dem Stecker P5.5 kontrolliert. Setzen Sie deshalb SSM = SSI = 1.
- Erst wenn alle Settings im Register CR1 gemacht sind, darf (und muss) das SPE Bit gesetzt werden. Ab dann sind alle Settings „scharf“ gestellt und der Daten Transfer kann beginnen.
- Das Register CR2 wird im vorgegebenen Rahmen auf 0x0000 initialisiert und muss nicht verändert werden. Dadurch sind alle Interrupts und die DMA (Direct Memory Access) disabled.

4.2 Senden/Empfangen

Implementieren Sie das Senden von Daten in der unten stehenden Sende-/Empfangsfunktion, die in **`hal_spi.h`** als Funktionsprototyp und in **`hal_spi.c`** als Rahmen vordefiniert ist:

Die Funktion **`uint8_t hal_spi_read_write (uint8_t send_byte)`** soll das übergebene Byte versenden und das empfangene Byte zurückgeben.

Hinweise zur Verwendung der SPI-Schnittstelle

- Die Funktionen zur Steuerung des $\overline{SS}$ Signals **`static void set_ss_pin_low()`** und **`static void set_ss_pin_high()`** sind gegeben.
- Zusätzlich ist auch die Funktion **`static void wait_10_us()`** gegeben. Verwenden Sie diese um sicherzustellen, dass das $\overline{SS}$ Signal erst nach der letzten Clock-Flanke auf High gesetzt wird. Dies können Sie mit dem Oszilloskop überprüfen.
- In den Vorlesungsunterlagen oder im beigelegten Reference Manual auf Seite 861 finden Sie, wie zum Senden und Empfangen vorgegangen werden muss. Das Datenregister **`SPI1->DR`** und das Kontrollregister **`SPI1->SR`** sind bereits vordefiniert. Eine Beschreibung des Kontrollregisters und der Flags finden Sie im Reference Manual auf den Seiten 872 und 900.

4.3 Testen der Funktion

Verifizieren Sie Ihre Sende/Empfangsfunktion mit Hilfe des Testprogrammes **`test.c`**. Das Testprogram liest ein Byte von S7 bis S0 ein, gibt den Wert auf Led7 bis Led0 aus und sendet ihn via SPI. Der von der SPI-Schnittstelle empfangene Wert wird auf Led23 bis Led16 ausgegeben.

Verbinden Sie für den Test die MOSI und MISO mit einer Drahtbrücke auf dem Port 5 gemäss Abbildung 4.

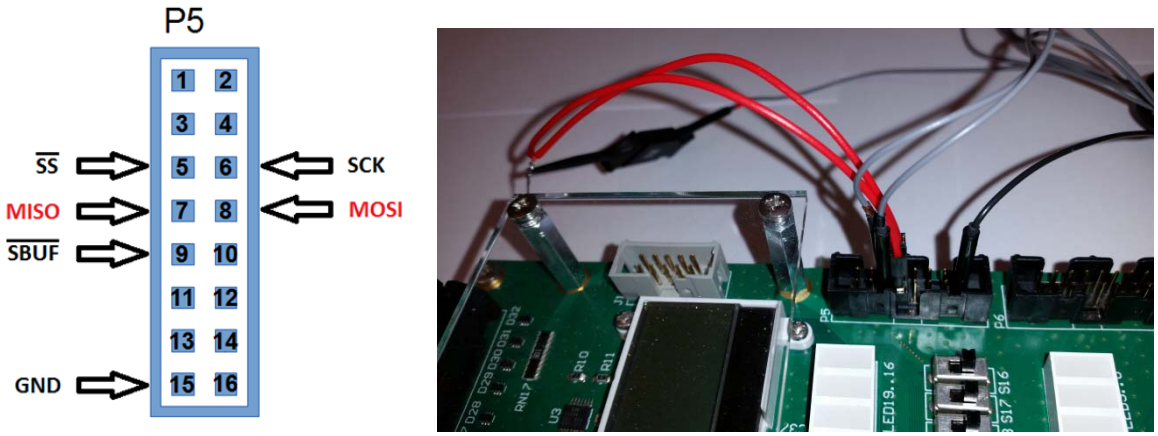

Abbildung 4: Verbindung von MOSI und MISO Leitung

Messen Sie zusätzlich die Signale $MOSI$, CLK und $\overline{SS}$ mit der SPI-Analysefunktion des Oszilloskops am Port 5.

Achtung: Für diesen Test darf das Display nicht angeschlossen werden → Kurzschluss

Hinweise zur SPI-Analysefunktion des Oszilloskops:

- Die SPI Pins können über die analogen oder über die digitalen Kanäle angeschlossen werden.
- Definieren Sie den Bus über die blaue Taste „Bus B1“. Wählen Sie „SPI“, definieren Sie die Eingänge, kontrollieren Sie die Schwellenwerte und konfigurieren Sie Polarität und Phase des SPI unter „SPI Einstellungen“.
- Haben Sie Spikes auf den Signalen, dann überprüfen sie den Thresholdwert und korrigieren ihn gegebenenfalls auf 2.5 V.

4.4 Bewertung

Die lauffähigen Programme müssen präsentiert werden. Die einzelnen Studierenden müssen die Lösungen und den Quellcode verstanden haben und erklären können.

Bewertungskriterien	**Gewichtung**
Die SPI-Schnittstelle wird wie in Aufgabe 4.1 gefordert initialisiert	1/3
Das Programm erfüllt die in Aufgabe 4.2 geforderte Funktionalität.	1/3
Das Programm wurde entsprechend Aufgabe 4.3 getestet und funktioniert korrekt. Ein Screenshot des Oszilloskops wird gezeigt und erklärt.	1/3