Fachhochschule Münster University of Applied Sciences

Labor für Halbleiterbaulelemente und

Übung Bussysteme SS2015
Tutor: Diplom Informatiker Gregor Rebel

Versuch 3

Serial Peripheral Interface - SPI

Für die Durchführung dieses Versuches wird teilweise auf Beschreibungen und Inhalte des vorangegangenen Versuches 2 (Serielle Schnittstelle USART) zurückgegriffen. Lesen Sie die entsprechenden Abschnitte bei Bedarf dort nach.

Der Serial Peripheral Interface Bus oder SPI Bus ist ein synchroner, serieller Datenübertragungsstandard im Full Duplex Modus. Teilnehmer kommunizieren im Master- oder Slavemodus, wobei der Master den Beginn einer Übertragung anstößt. Der Zugriff auf mehrere Slaves ist mit jeweils einzelnen Slave-Select-Leitungen möglich. Gelegentlich wird SPI auch als „four-wire serial bus“ oder auch SSI (Synchronous Serial Interface) bezeichnet.

Interface/Schnittstelle
Der SPI Bus spezifiziert vier logische Signale:

- SCLK: serial clock (Generiert durch den Master);
- MOSI: master output, slave input (Ausgang des Masters);
- MISO: master input, slave output (Ausgang des Slaves);
- SS: slave select (active low, Gesetzt durch den Master).

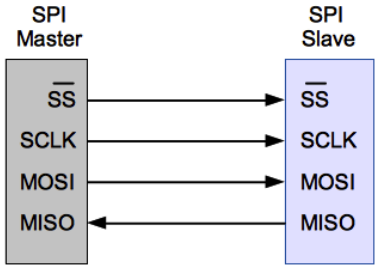

Bild 1: Darstellung der SPI Signale

Betrieb
Der SPI Bus kann mit einem einzigen Master und einem oder mehreren Slaves betrieben werden. Wird nur ein einziger Slave verwendet, so kann der Slave-Select Pin auf logisch Null gehalten werden, falls der Slave dieses erlaubt. Mit mehreren Slaves an einem Master benötigt für jeder Slave eine separate SS Leitung. Die meisten Slave Bauelemente haben einen Tri-State-Ausgang, so dass ihr MISO hochohmig wird, wenn sie gerade nicht ausgewählt sind.

Datenübertragung
Bei einem typischen Schaltungsentwurf werden zwei Schieberegister benutzt um einen chipinternen Ringpuffer zu realisieren. Zum Anfang einer Übertragung erzeugt der Master das Clocksignal welches gleich oder kleiner der Maximalfrequenz ist, welche der Slave unterstützt. Diese Frequenzen liegen in der Regel zwischen 1-100 MHz. Als nächstes zieht der Master die SS-Leitung des gewünschten Slaves auf logisch Null. Dies geschieht, da die SS-Leitung active low ist, also eine logische Eins hat, wenn der Slave nicht ausgewählt ist. Während jedes SPI Taktzyklus findet eine Full-Duplex-Übertragung statt:

- Der Master sendet ein Bit auf der MOSI Leitung; der Slave liest es von derselben Leitung.
- Der Slave sendet ein Bit auf der MISO Leitung; der Master liest es von derselben Leitung.

Genau genommen benötigen nicht alle Übertragungen diese vier Vorgänge aber sie finden trotzdem so statt.
Normalerweise beinhaltet die Übertragung zwei Schieberegister mit einer festen Größe, beispielsweise acht Bit. Jeweils eines befindet sich im Master und im Slave, sodass eine ringförmige Anordnung besteht. Normalerweise werden Daten mit dem MSB zuerst gesendet und gleichzeitig wird ein MSB im selben Register empfangen. Nachdem alle Registerwerte gesendet wurden haben Master und Slave also die Registerwerte getauscht. Jedes Gerät kann dann mit dem Registerinhalt wie gewünscht verfahren, es z.B. in den Speicher schreiben. Falls es mehr Daten zu übertragen gibt, so werden die Register mit neuen Werten geladen und der Vorgang beginnt von vorne. Die Übertragung kann eine beliebige Anzahl an Taktzyklen betragen. Sind keine Daten für den Transport mehr vorgesehen, so stoppt der Master die Taktgenerierung. Dabei wird dann üblicherweise auch der Slave deselektiert. Übertragungen bestehen oft aus 8-Bit Datenwörtern und ein Master kann je nach Bedarf mehrere Übertragungen initiieren. Jedoch sind auch andere Wortlängen gebräuchlich, wie beispielsweise 16- oder 12-Bit Datenwörter.

Jeder Slave der nicht über die SS Leitung aktiviert wurde ignoriert seine Eingänge und treibt nicht seinen MISO Ausgang. Der Master darf immer nur einen Slave gleichzeitig aktivieren.

Clockpolarität und Phase
Zusätzlich zur Taktfrequenz muss der Master auch die Clockpolarität und die Phase in Hinblick auf die Daten berücksichtigen. Das Timing wird im Nachfolgenden genauer beschrieben und bezieht sich sowohl auf den Master als auch auf den Slave.

Mit CPOL=0 ist Logisch Null der Grundzustand der Clock
Mit **CPHA=0** werden Daten mit steigender Taktflanke gelesen (low→high Übergang) und mit fallender Flanke geschrieben (high→low Übergang).
Mit **CPHA=1** werden Daten mit fallender Taktflanke gelesen und mit steigender Taktflanke geschrieben.

Mit CPOL=1 ist Logisch Eins der Grundzustand der Clock(Invertierung von CPOL=0)
Mit **CPHA=0** werden Daten mit steigender Taktflanke gelesen und mit fallender Flanke geschrieben.
Mit **CPHA=1** werden Daten mit fallender Taktflanke gelesen und mit steigender Taktflanke geschrieben.

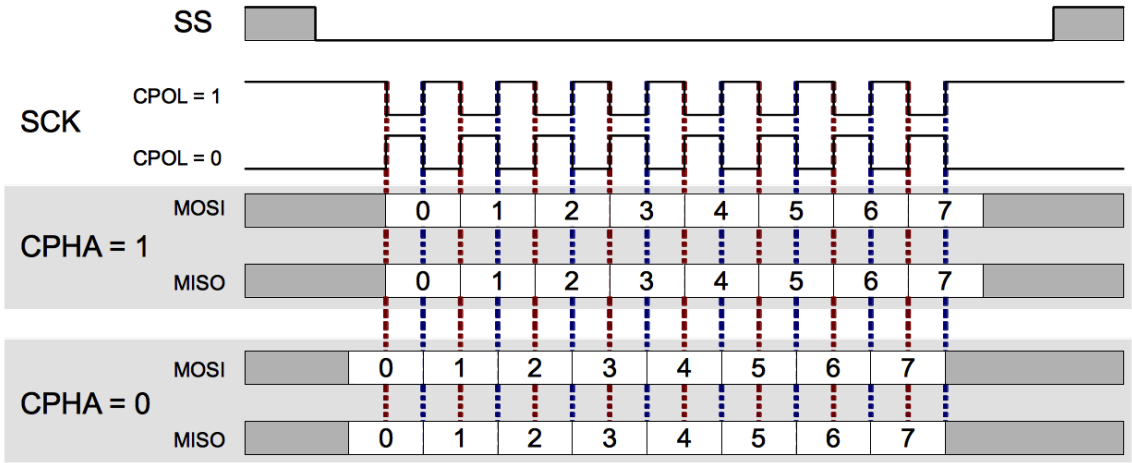

Bild 2: SPI Timing Diagramm

Kurz gesagt, CPHA = 0 bedeutet Auswerten an der ersten Flanke, wahrend CPHA = 1 Auswerten an der zweiten Flanke bedeutet, egal ob die Taktflanke fällt oder steigt.
Man beachte, dass bei CPHA = 0 die Daten einen halben Takt stabil anliegen müssen bevor der erste Taktzyklus beginnt. Für beide CPOL und CPHA Zustände muss der Grundzustand des Taktes stabil sein bevor ein Slave ausgewählt wird.

Modusnummerierung
Die Kombinationen aus Polarität und Phase werden oft als Modus beschrieben, welche gewöhnlicherweise nach folgender Konvention nummeriert werden. Dabei ist CPOL das höherwertige Bit und CPHA das niederwertige Bit.

- Modus 0: CPOL = 0 und CPHA = 0
- Modus 1: CPOL = 0 und CPHA = 1
- Modus 2: CPOL = 1 und CPHA = 0
- Modus 3: CPOL = 1 und CPHA = 1

Unabhängige Slavekonfiguration
Bei der unabhängigen Slavekonfiguration existiert für jeden Slave eine eigene Select-Leitung. Dies ist die gebräuchliche Art SPI zu nutzen. Da alle MISO Pins der Slaves somit verbunden sind müssen diese Tri-State Pins sein.

Daisychain SPI Konfiguration (Master und Kooperative Slaves)
Einige Anwendungen mit dem SPI Bus sind entworfen um in einer Daisychain verbunden werden zu können. Dazu wird der Ausgang des ersten Slaves mit dem Eingang des zweiten Slaves verbunden, usw. Die Slaves sind dabei so konfiguriert, dass er eine Kopie dessen was er während der ersten Taktphase bekommen hat, in der zweiten Taktphase wieder ausgibt. Die ganze Kette agiert also als eine Art großes SPI Schieberegister. Diese Art der Konfiguration bietet den Vorteil, dass nur noch eine SS Leitung benötigt wird und nicht mehr eine für jeden einzelnen Slave.

Aufgabe 1

Für diese Aufgabe kann zunächst ein beliebiges Prototypboard verwendet werden. Es soll die erste SPI-Schnittstelle auf dem aktuellen Board konfiguriert, initialisiert und zum Senden von Nachrichten verwendet werden. Die versendeten Nachrichten werden dann mit einem Oszilloskop dargestellt.

High-Level Treiber ttc_spi

Die generelle Verwendung von ttc_-Treibern wurde bereits im vorigen Praktikum vorgestellt. Lesen Sie diesen Abschnitt bei Bedarf nach.

Schritte

1. Erstellen Sie ein neues Projekt mit dem Namen BS_Grupper1_Versuch3 bzw. BS_Gruppe2_Versuch3.
2. Ändern Sie die Konfiguration in activate_project.sh
   1. Prototypboard + Programmer
   2. Aktivieren Sie **500_ttc_spi, 500_ttc_gpio**
   3. Deaktivieren Sie 600_example_leds
3. Fügen Sie eine neue Quelltextdatei namens test_spi mittels _/source.pl zum Projekt hinzu
4. Legen Sie die Funktion **void test_spi_start()** in test_spi.c/.h an und rufen Sie diese von taskMain() aus 1x auf.
   Hierzu können Sie wieder _/source.pl nutzen um das Hinzufügen von Funktionen zu vereinfachen:
   _/source.pl function public 'void start()' test_spi
   Beachten Sie, dass der Funktionsname start automatisch um den Dateinamen ergänzt wurde um den eindeutigen Namen test_spi_start zu erzeugen.
5. Implementieren Sie test_spi_start()
   1. Laden Sie die Konfiguration der ersten SPI-Schnittstelle.
   2. Konfigurieren als **Master** für **8-Bit Datenworte** und eine Datenrate von **200 kBit/s** im **Modus 2**
   3. Initialisieren Sie die konfigurierte Schnittstelle
   4. Senden Sie den Text „SOS“ im Abstand von 500 ms
   5. Verwenden Sie die erste LED indem Sie deren Zustand bei jedem Sendevorgang invertieren

Aufgabe 2

Der SPI-Bus zeichnet sich durch eine unmodulierte Übertragung von Spannungssignalen mit steilen Signalflanken und Tiefpasscharakter aus. Die Leitungen sind nicht terminiert. Wie wirkt sich das bei verschiedenen Leitungslängen aus?

Schritte

Messen Sie den Signalverlauf der MOSI und SCK Leitung direkt an den Anschlusspins der Platine.
Zeichnen Sie die Signale in das Koordinatensystem ein.

1. Messen Sie den Signalverlauf der MOSI und SCK Leitung direkt an den Anschlusspins der Platine.
   Zeichnen Sie die Signale in das Koordinatensystem ein.
2. Schließen Sie eine Ader eines 100 cm langen Flachbandkabels an den Pins SCK an.
3. Schließen Sie die rechte Nachbarleitung im Kabel an GND an.
4. Messen Sie die Spannung im Flachbandkabel am Ende der linken Nachbarleitung von SCK.
5. Wie kommt das Signal auf die freie Nachbarleitung?

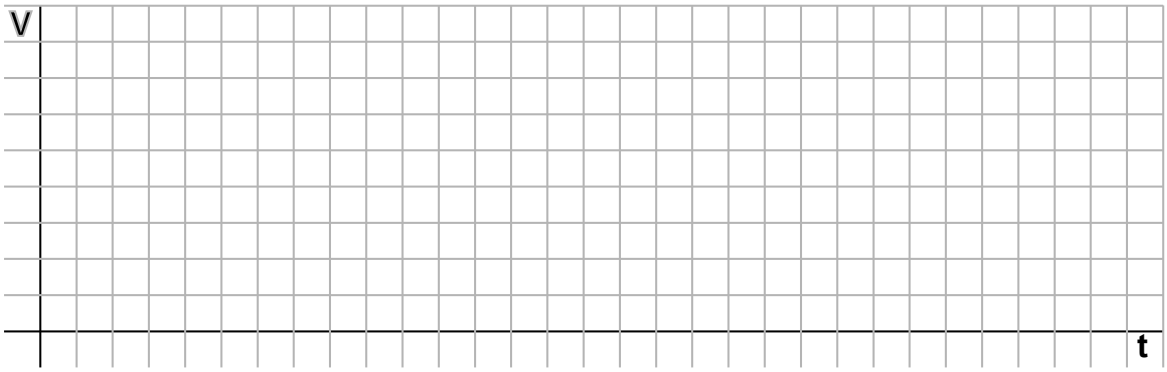

Abbildung 1: MOSI

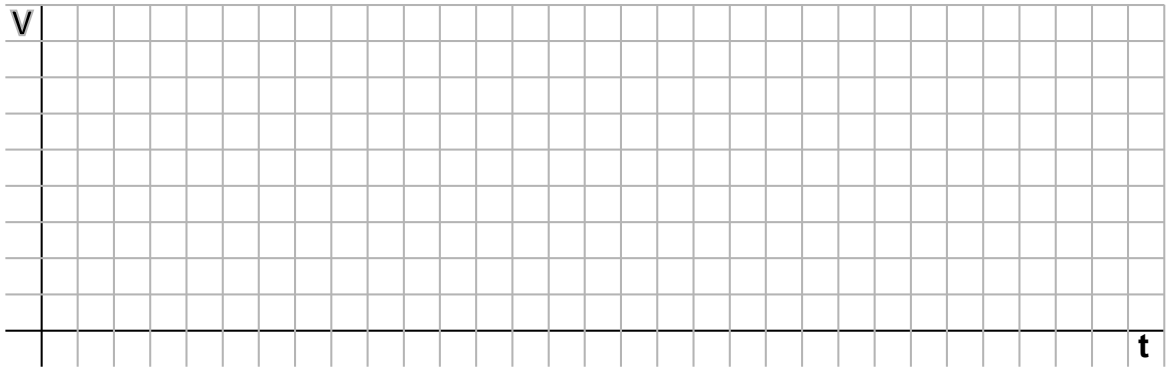

Abbildung 2: SCK

Aufgabe 3

Für diese Aufgabe müssen Sie sich mit einer Nachbargruppe zusammentun und zwei Protoboards per Kabel verbinden. Schließen Sie dazu beide Boards am selben Rechner an. Dadurch wird vermieden, dass die Ausgangstreiber durchbrennen. Diese sind nicht gegen Überspannungen und unterschiedliche Massepotentiale geschützt!
Verfolgen Sie die Datenübertragung auf der MOSI und MISO Leitung mit einem Oszilloskop.

Gruppe A

1. Initialisiert den SPI wie zuvor beschrieben.
2. Initialisiert zu Beginn eine Variable u8_t Data als 0.
3. In einer Endlosschleife
    1. Sendet den aktuellen Wert von Data
    2. Nach jedem gesendeten Byte wird ein Byte empfangen und dessen Wert in Data geschrieben

Gruppe B

1. Initialisiert den SPI als Slave.
2. Initialisiert zu Beginn eine Variable u8_t Data als 0.
3. In einer Endlosschleife
    1. Empfängt ein Byte und schreibt den Wert nach Data
    2. Zählt Data um eins hoch und sendet Data zurück