Serial Peripherial Interface

Full Duplex → gleichzeitiges Senden von Daten auf in beide Richtungen (Master / Slave)
Serial Clock – Taktsignal auf eigener Leitung (SCK):

- Übertragung entweder bei Taktzustand oder Taktflanke
- Bitfolge wird entsprechend aus Datenstrom entnommen

Leitungszuordnung:

- SCK: Serial Clock → Taktleitung (einmal für alle angeschlossenen Geräte)
- MOSI: Master Out, Slave In → Datenleitung von Master zu Slave (einmal für alle Geräte)
- MISO: Master In, Slave Out → Datenleitung von Slave zu Master (einmal für alle Geräte)
- SSLEX: Slave Select (X = Slave-Nr.) → Wahl des anzusprechenden Slave-Gerätes → High-Pegel (Ruhezustand) wird kurz vor dem Start der Übertragung auf Low-Pegel gesetzt

Taktleitung:

- Ruhezustand (LOW/HIGH wählbar)
- Schalten bei fallender / steigender Flanke
  ➔ 2 Bit (CPOL, CPHA) → Clock Polarität, Clock Phase (Zustände s. Tabelle)

Registerübersicht:

- SXSPCCR → Prescaler für SPI-Schnitstelle
- SXSPCR → Control Register der SPI-Schnittstelle → u.a. CPOL/CPHA → Einrichten der Schnittstelle
- SXSPDR → Daten Register (im Master-Modus → sofortiges Senden von geschriebenen Daten
- SXSPINT → Interrupt Info → wenn Schnittstelle Interrupt auslöst wird Bit gesetzt → eigene ISR muss anschließend quittiert werden
- SXSPSR → Status / Fehlermeldungen der Schnittstelle

<u>Das SPI des LPC 2148</u>

- benötigt: SCK, MOSI, MISO, SSEL
- einstellen durch PINSEL0 → P0.4 – 0.7 (auf jeweils 01)
- Bsp.: SPI-Schnittstelle verwenden

```
PINSEL0 = PINSEL0 | 0x5500;     //P0.4 – P0.7 auf 01 setzen
```

- Register SXSPCCR
  - Prescaler → SPI Clock Control Register
  - Wert > 8 und ganze Zahl
- Register SXSPCR
  - Bit 0, 1 reserviert, ebenso 12 – 31
  - Bit 2 → Wortbreite einstellen (0 = 8 Bit, 1 = Bitwert in Bits 8 - 11)
  - Bit 3 → CPHA, Bit 4 → CPOL
  - Bit 5 → Betriebsart (0 = Slave, 1 = Master)
  - Bit 6 → LSBF → Übertragung beginnt mit MSB (0) oder LSB (1)
  - Bsp.:

S0SPCR = 0x90; //Auswertung: 8 Bit, Clock: LOW, steigende Flanke, Slave, MSB first, Interrupts eingeschaltet

- Register SXSPSR
  - Status Register
  - Bit 3 → ABORT → Abbruch stattgefunden
  - Bit 4 / 5 / 6 → Fehler
  - Bit 7 → SPIF → Erfolgreicher Abschluss der Übertragung
- Register SXSPDR
  - SPI Data Register
  - enthält Sende-/Empfangsdaten → können von dort gelesen werden
  - im Master-Modus → direktes senden nach schreiben

Interrupts an SPI Schnittstelle

- erfolgreiches Versenden / Empfangen von Daten (→ SPIF)
- durch Fehlerzustände → MODF
- Bits stehen im Status Register SXSPSR
- Register SXSPINT:
  - Interrupt Register
  - Info, ob Interrupt ausgelöst werden soll
  - dient zum Quittieren von Interrupts in der ISR

SPI im Master-Modus

- SPI generiert das Taktsignal → SCK (SCK ist Ausgang)
- Vor der Übertragung muss der Slave über SSEL aktiviert werden
- SSEL muss vor der Übertragung von HIGH auf LOW gesetzt werden, nach Ende der Übertragung wird der Pagel von LOW auf HIGH zurückgesetzt
- im Master-Modus ist der SSEL als GPIO (PINSELX → 00) definiert, nicht als SSEL
- es können mehrere Slaves vorhanden sein, die jeweils über eine eigene SSEL-Leitung aktiviert werden müssen
- der Programmierer muss die Aktivierung des Slaves per SSEL selbst vornehmen
- ein Schreibzugriff auf SXSPDR bewirkt sofortige Datenübertragung
- es können gleichzeitig Daten empfangen werden (Full-Duplex), diese stehen anschließend im SPIDR-Register
- erfolgreiche Übertragung von Daten wird mit SPIF-Bit im SXSPSR angezeigt

SPI im Slave-Modus

- SCK ist für den Slave ein Eingang
- Slave kann nicht selbst aktiviert werden, er reagiert auf Anforderung des Masters
- Master teilt dem Slave per SSEL von HIGH auf LOW mit, dass Slave Daten senden kann
- gesendete Daten stehen anschließend im SXSPDR des Masters
- der Abschluss einer erfolgreichen Datenübertragung wird mit dem SPIF-Flag des SXSPSR angezeigt

Beispiel einer SPI-Konfiguration bzw. Kommunikation

- SPI-Schnittstelle im Master-Modus für eine 12 Bit-Übertragung
- Serial-Clock soll im Ruhezustand HIGH-Pegel haben
- Daten sollen bei steigender Flanke des SCK übernommen werden
- LSBF bei Übertragung soll vorliegen
- SSEL soll auf P0.8 als GPIO genutzt werden (→ digitaler Ausgang)
- Slave sendet nichts → keine MISO Leitung nötig (gilt auch für Experimentierboard)
- Programm arbeitet ohne Interrupts → Polling
- Programmablauf:

```
#include LPC214X.h;                    //Definitionen u.a. für LPC2148
int main()      {
        int x;
        PINSEL0 = 0x1100;              //Ports für SPI setzen (0.4 → SCK, 0.6 → MOSI)
        IODIR0 = 0x100;                //P0.8 als dig. Ausgang setzen
        IOSET = 0x100;                 //SSEL → HIGH-Pegel als Ruhezustand
        S0SPCCR = 30;                  //SCK = 500 kHz
        S0SPCR = 0xC7C;                // → 1100 0111 1100 (High-Pegel, steigende Flanke,
Bitfolge in Bit 8-11, Mastermodus, LSBF, keine Interrupts, 8 – 11 = 1100 → 12 Bit

        while(1)        {
                IOCLR = 0x100;         //LOW-Pegel auf SSEL → Übertragung startet gleich
                S0SPDR = 0x8FA;        //zu übertragende Werte in Data-Register schreiben
                while(!(S0SPDR & 0x80)) {      //erfolgreiche Übertragung abwarten
                        IOSET = 0x100;         //HIGH-Pegel auf SSEL → Ruhezustand
                }
        for(x = 0; x< 7500; x++) {}    // 1 ms warten
        }
}
```