10.21. SPI-Bus

Allgemeines

Beim SPI-Bus (Serial Peripheral Interface) handelt es sich um einen synchronen 3-Draht Bus mit einer zusätzlichen Steuerleitung (/SS) (siehe Abbildung 221). Er wurde von Motorola Anfang 1989 entwickelt und bietet einen volldupplex Datentransfer. Spezifiziert wurde er damals für 3 MBits/s. Heute gibt es für diesen Bus Bausteine mit bis zu 33 Mbits/s. Für jedes /SS-Signal wird ein eigener Portpin am Master benötigt. Ist das /SS-Signal am Slave auf High, sind die anderen Eingänge des Slave im Tri-State. Der Master selbst benötigt keinen /SS-Pin.

Die Übertragung mit einen SPI-Bus erfolgt folgendermaßen:

1. Der Master setzt den /SS-Pin auf Low. Dies wird mit Hilfe von Software realisiert.
2. Danach wird ein Wert in das Übertragungsregister (SPDAT) geschrieben. Durch das Einschreiben wird die Übertragung automatisch gestartet und das zu sendende Byte wird über den Portpin MOSI seriell herausgeschoben synchron zum SPICLK. Gleichzeitig werden acht Bits über den Portpin MISO empfangen. Somit ist jeder Sendevorgang grundsätzlich mit einem Empfangsvorgang automatisch verbunden. Will der Master nur einen Wert senden, so ist die Empfangsinformation irrelevant. Der Slave-Baustein sendet keine Quittung zurück. Es können nun ein weitere Bytes ohne Zeitverzögerung in gleicher Weise gesendet und empfangen werden.
3. Soll nur ein Wert aus dem Slave gelesen werden, muss dennoch ein Dummywert gesendet werden.
4. Die Übertragung an den Slave wird gestoppt, indem kein weiterer Wert mehr in das SPDAT geschrieben und das /SS-Pin auf High gelegt wird. Erst danach kann der Master einen anderen Slave ansprechen.
5. Die Verwendung des SPI-Buses ist softwaremäßig einfacher, als die des I²C-Busses, da kein Adressfeld mit gesendet und ausgewertet werden muss.

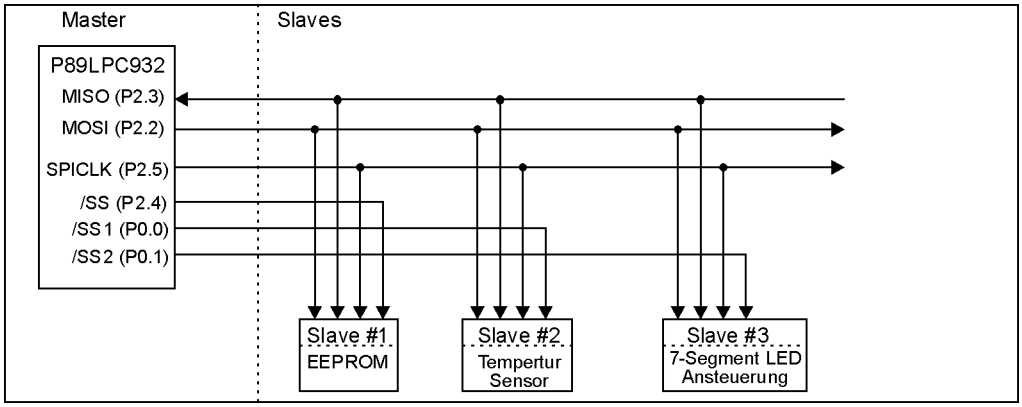


Abbildung 221 Anschlussbelegung beim SPI-Bus

Mittels dem Signal /SS wird der Slave aktiviert. Sind mehrere Slaves am SPI-Bus angeschlossen , wird für jeden Slave ein weiteres /SSx-Signal benötigt. Mit der MOSI-Leitung

werden die Daten seriell vom Master zum Slave gesendet. Mit der MISO-Leitung werden die Daten vom Slave empfangen. Die SPICLK-Leitung sorgt für die Taktvorgabe vom Master.

Vorteile des SPI gegenüber dem I²C-Bus:

- Schneller Datentransfer
- Bidirektionale Datentransfer
- Einfaches Protokollhandling
- Jeder Slave kann mit der maximalen Taktfrequenz betrieben werden, da jeweils nur ein Slave aktiv am SPI-Bus ist. Beim I²C-Bus bestimmt der langsamste Teilnehmer die Taktfrequenz.

Nachteile des SPI gegenüber dem I²C-Bus:

- Für jeden Slave wird eine zusätzliche Steuerleitung (/SS) benötigt.
- Es kann keine Überprüfung stattfinden, ob der Slave angeschlossen ist, bzw. die Daten empfangen hat.

Der SPI-Bus der LPC900-Familie arbeitet mit bis zu 3 Mbit/s und unterstützt den Master- / Slave-Mode. Tabelle 118 enthält eine Übersicht, welche Derivate einen SPI-Bus haben. Wird der RC-Oscillator verwendet, erreicht man eine Übertragungsrate von 1,84 Mbit/s.

LPC	901	902	903	906	907	908	912	913	914	920	921	922	930	931	932	933	934	935
SPI	×	×	×	×	×	×	✓	✓	✓	×	×	×	✓	✓	✓	✓	✓	✓

Tabelle 125 LPC900-Derivate mit SPI-Bus

Aufbau des SPI-Busses

Der SPI-Bus in der LPC900-Familie wird über die SFR SPCTL, SPSTAT und SPDAT angesprochen (siehe Abbildung 222). Die Taktversorgung erfolgt über den CCLK. Der SPI-Block kann über das Bit SPEN aktiviert werden. Nach dem Reset ist der SPI-Block deaktiviert. Die Portpins können beim deaktivierten SPI-Block anderweitig verwendet werden.

Die Portpins müssen bei aktivem SPI-Block in der Betriebsart bidirektional konfiguriert sein (siehe Kapitel 10.6, Seite 154), damit an den Portpins der Takt und die Daten ausgegeben werden.

☞ !! Sie **müssen** zudem auf die Portpins den Wert „1“ schreiben. !!

Der SPICLK arbeitet als Sender, wenn der SPI-Block im Masterbetrieb arbeitet und als Empfänger, wenn sich der SPI-Block im Slavebetrieb befindet.

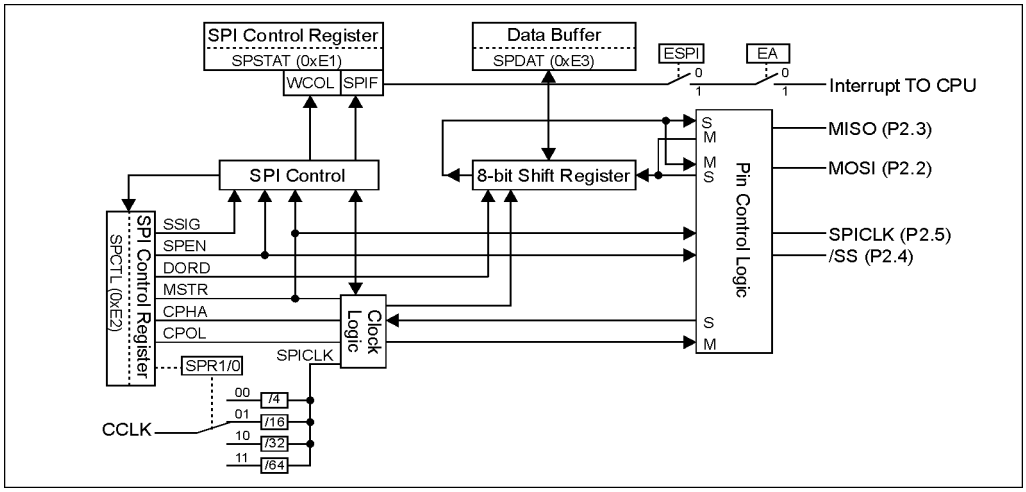


Abbildung 222 Aufbau des SPI-Busses

Tabelle 126 enthält eine Übersicht über alle zulässigen Konfigurationen für den SPI-Bus. Wenn in der Spalte /SS „P2^4“ angegeben ist, kann dieser als Portpin verwendet werden.

☞ !! Ist der SPI-Bus als Master konfiguriert, kann dieser auf Slave umkonfiguriert werden, wenn am /SS von außen eine „0“ angelegt wird und SSIG den Wert „0“ enthält. Der Portpin 2^4 ms als Input oder Quasi-bidirektional konfiguriert sein. Tritt eine solche Umkonfiguration ein, wird das Flag SPIF im SFR SPSTAT gesetzt. !!

SPEN	SSIG	/SS	MSTR	Mode	MISO	MOSI	SPICLK	Bemerkungen
0	X	P2^4	X	SPI deaktiviert	P2^3	P2^2	P2^5	SPI ist deaktiviert. Die Portpins sind frei verfügbar.
1	0	0	0	Slave	output	input	input	SPI-Bus arbeitet als Slave.
1	0	1	0	Slave	Hi-Z	input	input	Der SPI-Bus ist nicht selektiert. Das Signal MISO ist Hi-Z, um Buskonflikte zu vermeiden.
1	0	0	1 ->0	Slave	output	input	input	P2^4 ist konfiguriert als Quasi-bidirektional bzw. als Input-Only. Wird /SS auf „0“ gezogen, wechselt der Zustand von MSTR auf „0“.
1	0	1	1	Master	input	Hi-Z	Hi-Z	MOSI und SPICLK sind Hi-Z, um Buskonflikte zu vermeiden. In Abhängigkeit von CPOL muss ein Pull-Up bzw. ein Pull-Down für den SPCLK eingebaut werden.
1	0	X	1	Master	input	output	output	MOSI und SPICLK arbeiten in der Betriebsart Push-Pull.
1	1	P2^4	0	Slave	output	input	input	
1	1	P2^4	1	Master	input	output	output	

Tabelle 126 Aktivierung des SPI-Busses (Master/ Slave)

Der Portpin /SS ist ein optionales Slave-Select Pin, wenn sich der SPI-Block im Slavebetrieb befindet. Andernfalls kann dieser Pin als zusätzliches Select-Pin verwendet werden, wenn sich mehrere Slaves im SPI-Bus befinden (Masterbetrieb, siehe Abbildung 221).

Der Portpin /SS wird ignoriert, wenn eine der folgenden Bedingungen erfüllt ist:

- Der SPI-Block ist deaktiviert (SPEN = 0)
- Der SPI-Bus ist als Master konfiguriert (MSTR = 1) und der Portpin 2.4 als Input.
- Das Bit SSIG enthält den Wert 1.

Typische SPI-Bus Konfigurationen

In den nachfolgenden Beschreibungen finden sie typische SPI-Bus Konfigurationen.

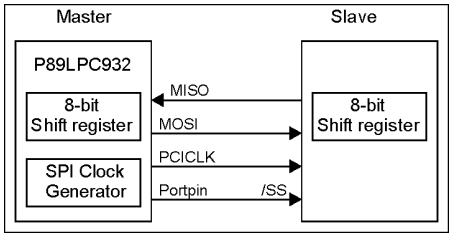


Abbildung 223 enthält einen Aufbau mit einem Master und einem Slave-Baustein. Die Bits SSIG und MSTR enthalten den Wert „1". Der Portpin /SS für die Ansteuerung des Slaves muss vor dem Schreibvorgang auf das SFR SPDAT per Software auf „0" gesetzt und nach Beendigung des Sendevorgangs auf „1" zurückgesetzt werden.

Abbildung 223 SPI-Bus mit einem Master und einem Slave-Baustein

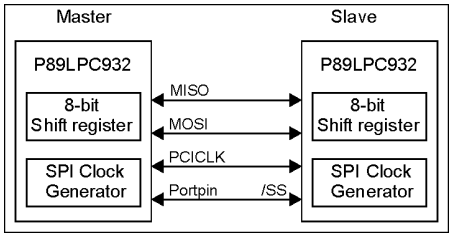


Abbildung 224 enthält einen Aufbau mit zwei Bausteinen mit SPI-Controllern. Die Bits SSIG und MSTR enthalten den Wert „0". Beide SPI-Controller können als Master konfiguriert sein. Beginnt einer von beiden mit einem Sendevorgang, indem er den Portpin /SS auf „0" zieht, wird automatisch beim anderen der Slave-Mode aktiviert.

Abbildung 224 SPI-Bus mit zwei SPI-Controllern (wechselnder Master-Betrieb)

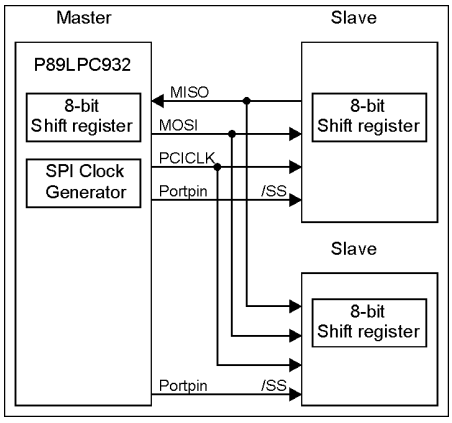


Abbildung 225 enthält einen Aufbau mit einem Master und zwei Slave-Bausteinen. Die Ansteuerung des /SS-Signal erfolgt per Software. Die Bits SSIG und MSTR enthalten den Wert „1".

Abbildung 225 SPI-Bus mit einem Master und zwei Slave-Bausteinen

SPI Control register (SPCTL)

Mit dem SFR SPCTL wird die Funktionsweise des SPI-Busses programmiert. Sie umfasst die Takt- und Datenlage, die Clockfrequenz des SPI-Busses und ob sich der SPI-Block als Master bzw. Slave verhalten soll.

Bezeichnung	MSB							LSB
SPCTL 0xE2	SSIG	SPEN	DORD	MSTR	CPOL	CPHA	SPR1	SPR0
Reset Value 0x04	0	0	0	0	0	1	0	0

SSIG (**SS IG**nore): Ist dieses Bit gesetzt, so entscheidet das Bit MSTR ob der SPI-Controller Master oder Slave ist. Enthält das Bit den Wert “0”, entscheidet das Pin /SS ob der SPI-Controller Master oder Slave ist (siehe Tabelle 126).

SPEN (**SP**I **EN**able): Mit diesem Bit wird der SPI-Block aktiviert. Enthält das Bit den Wert „0“, sind alle SPI-Pins als Portpins verfügbar.

DORD (SPI **D**ata **ORD**er): Mit diesem Bit wird bestimmt, ob das MSB oder das LSB zuerst ausgegeben bzw. empfangen wird.

0: Das MSB wird zuerst gesendet bzw. empfangen.

1: Das LSB wird zuerst gesendet bzw. empfangen.

MSTR (**M**aster/**S**lave mode Select): Mit diesem Bit wird der Master/Slave Mode festgelegt.

0: Slave

1: Master

CPOL (SPI **C**lock **POL**arity): Mit diesem Bit wird die Polarität des SPICLK bestimmt.

0: Enthält SPICLK den Wert „1“, befindet sich der SPICLK im Idlezustand. Die fallende Flanke des SPICLK leitet die Übertragung ein.

1: Enthält SPICLK den Wert „0“, befindet sich der SPICLK im Idlezustand. Die steigende Flanke des SPICLK leitet die Übertragung ein.

CPHA (SPI **C**lock **PHA**se): Mit diesem Bit wird eingestellt, ob bei der steigenden bzw. der fallenden Flanke von SPICLK der Datenwechsel erfolgt.

0: Die Daten werden ausgegeben, wenn an /SS eine „0“ anliegt und das Bit SSIG den Wert „0“ enthält. Der Datenwechsel erfolgt bei der nachlaufenden Flanke von SPICLK. Die Datenabtastung (empfangen) erfolgt bei der führenden Flanke von SPICLK.

☞ !! Enthält das Bit SSIG den Wert „1“, so ist das Verhalten undefiniert. !!

1: Der Datenwechsel (senden) erfolgt bei der führenden Flanke von SPICLK und die Datenabtastung (empfangen) erfolgt mit der nachlaufenden Flanke.

SPR1/SPR0 (**SP**I clock **R**ate): Mit Hilfe dieser zwei Bits wird die Taktrate für den SPI-Bus festgelegt.

SPR1	SPR0	SPI Clock Rate
0	0	CCLK/4
0	1	CCLK/16
1	0	CCLK/64
1	1	CCLK/128

SPI Data register (SPDAT)

Mit diesem SFR werden die Daten in das Transferregister für den SPI-Bus geschrieben bzw. ausgelesen. Das Flag SPIF im SFR SPSTAT zeigt an, wenn ein Wert komplett ausgesendet wurde. Wird während eines Sendevorgangs, also noch vor dem Setzen des Flags SPIF das SPDAT erneut beschrieben, so entsteht eine „Write collision". In diesem Fall wird das Flag WCOL im SFR SPSTAT gesetzt.

☞ !! In der Simulation kann über die virtuellen Register SPI_IN und SPI_OUT auf den SPI-Bus zugegriffen werden. !!

Bezeichnung	MSB							LSB
SPDAT 0xE3	7	6	5	4	3	2	1	0
Reset Value 0x00	0	0	0	0	0	0	0	0

SPI Status register (SPSTAT)

In diesem SFR sind die Flags für den SPI-Bus enthalten. Das Bit SPIF signalisiert, das der Sende-/Empfangsvorgang abgeschlossen ist. Dieses Bit kann auch die SPI-ISR auslösen.

Bezeichnung	MSB							LSB
SPSTAT 0xE1	SPIF	WCOL	-	-	-	-	-	-
Reset Value 0x04	0	0	-	-	-	-	-	-

SPIF (**SPI** Transfer Completion **F**lag): Dieses Flag wird gesetzt, wenn ein Sende-/Empfangsvorgang abgeschlossen ist. Zudem wird der Portpin /SS auf „1" zurückgesetzt, wenn der SPI-Controller im Master-Mode arbeitet und das Bit SSIG den Wert „0" enthält. Das Flag muss per Software durch Schreiben einer „1" zurückgesetzt werden.

WCOL (SPI **W**rite **COL**lision Flag): Dieses Flag wird gesetzt, wenn während eines Sende-/Empfangvorgangs auf das SFR SPDAT schreibend zugegriffen wird. Das Flag muss per Software durch Schreiben einer „1" zurückgesetzt werden.

-: Diese Bits sind für zukünftige Derivate reserviert. Beim Schreiben in das SFR sollten diese Bits den Wert 1 enthalten.

Die nachfolgenden Bilder enthalten den SPI-Bus mit unterschiedlichen Konfigurationen von CPOL, CPHA und DORD. Als Taktfrequenz wurde der RC-Oscillator/16 verwendet. Dies ergibt eine Frequenz von 459 kHz. Um die Unterschiede der einzelnen Betriebsarten besser erkennen zu können, wird immer der Wert 0x45 ausgesendet.

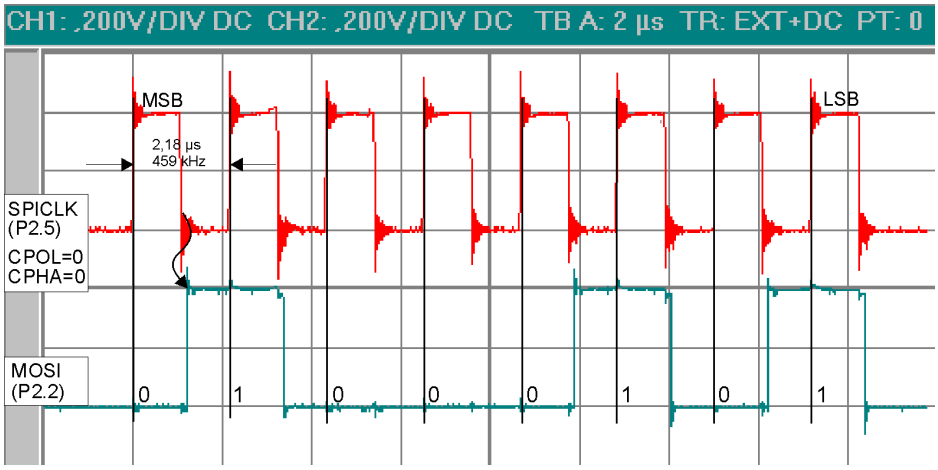


Abbildung 226 SPI im Masterbetrieb CPOL = 0, CPHA = 0, DORD = 0 (SPSTAT = 0xD1)

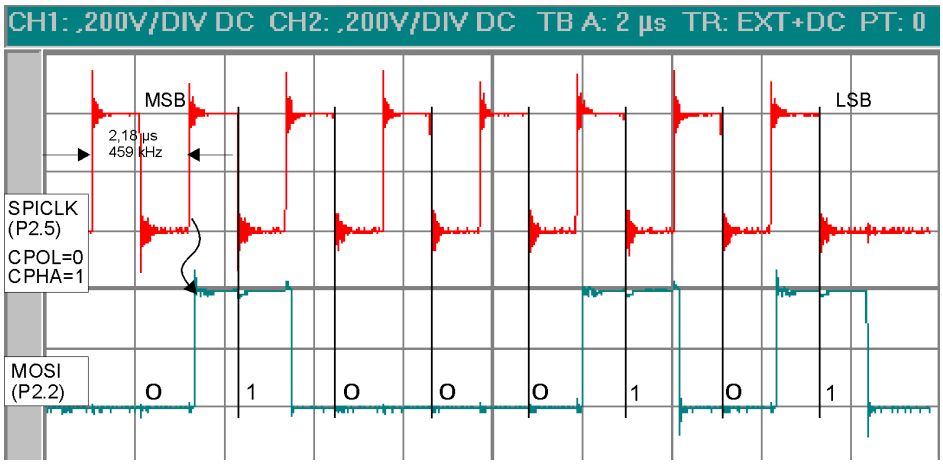


Abbildung 227 SPI im Masterbetrieb CPOL = 0, CPHA = 1, DORD = 0 (SPSTAT = 0xD5)

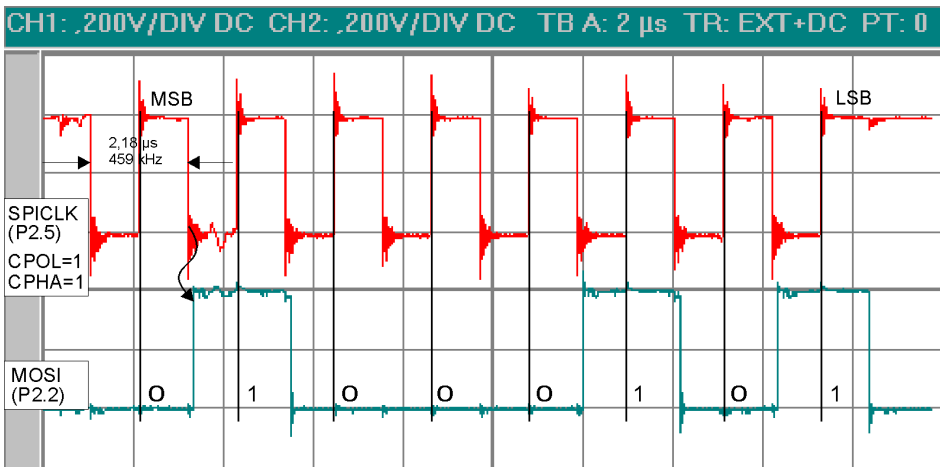


Abbildung 228 SPI im Masterbetrieb CPOL = 0, CPHA = 1, DORD = 0 (SPSTAT = 0xDD)

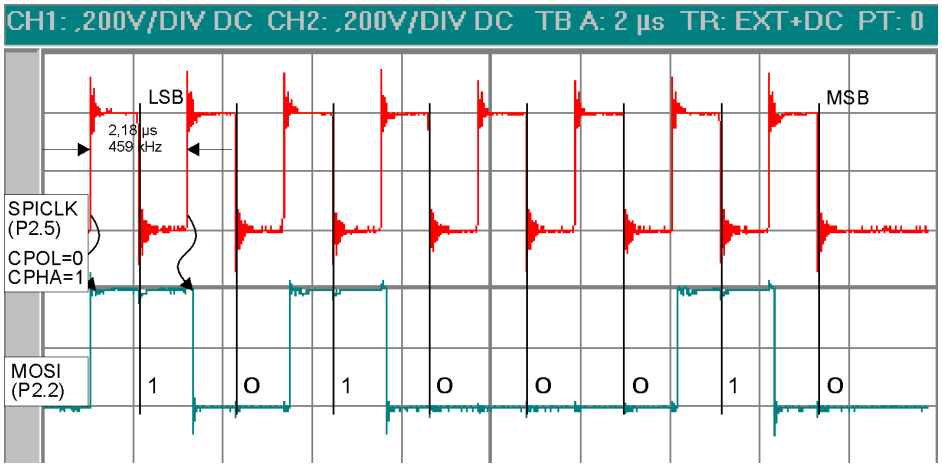


Abbildung 229 SPI im Masterbetrieb CPOL = 0, CPHA = 1, DORD = 1 (SPSTAT = 0xF1)

SPI-Bus im Master-Mode (Polling-Betrieb)

Das nachfolgende Projekt enthält den SPI-Block im Master-Betrieb. Dieser sendet an einen Slave den Wert 0x45 aus. Nach jedem Sendevorgang wird das Signal /SS wieder auf „1" gesetzt. Erstellen sie das Projekt aus Tabelle 127 und geben den Inhalt aus Listing 97 in das Sourcemodul TestSPI.c ein. Übersetzen sie danach das Projekt mit dem Button „Build All" und wechseln in den Simulator-Mode.

Projektname	Verzeichnis	Verwendete Sourcemodule
SPI_Test	Kap10_SPIMaster	TestSPI.c

Tabelle 127 Projekt SPI_Test

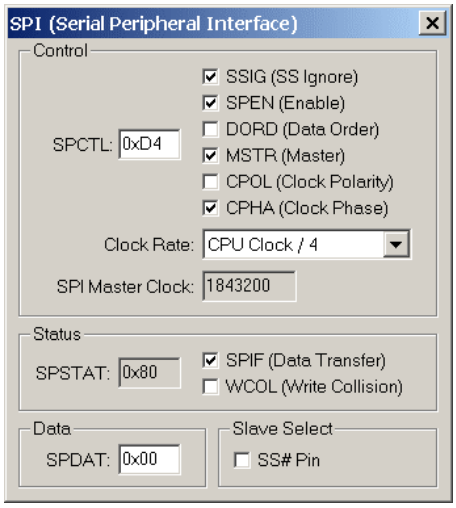
```
    #include <REG932.H>
    sbit _SS = P2^4;

    main()
    {
①      P2M1 = 0x3C; // Freigabe der SPI-Ports (MISO, MOSI, SPICLK, _SS)
①      P2M2 = 0x3C;
②      P2 = 0xFF;
③      SPCTL = 0xD4; // Freigabe des SPI-Blocks, MSTR = 1, SSIG = 1)

       while(1)
       {
④         _SS = 0; // SPI Slave-Signal aktivieren
⑤         SPDAT = 0x45; // Wert schreiben
⑥           while((SPSTAT & 0x80) == 0x00); // Warten bis Wert gesendet
⑦         SPSTAT = 0xFF; // SPIF zuruecksetzen
⑧         _SS = 1; // SPI Slave-Signal deaktivieren
       }
    }
```

Listing 97 Sourcemodul TestSPI.c

Damit die SPI-Ports freigeschaltet werden, müssen diese zuerst in die Betriebsart „Quasi-bidirektional“ gesetzt werden (siehe Listing 97, ①). Der interne Zustand der Portpins wird zudem auf „1“ gesetzt (siehe ②). Die Freigabe des SPI-Blocks und die Umschaltung in den Master-Betrieb erfolgt im SFR SPCTL (siehe ③). In der while()-Schleife wird jetzt kontinuierlich der Wert 0x45 ausgegeben (siehe ⑤). Die Aktivierung des SPI-Slaves erfolgt per Software über den Portpin _SS (siehe ④). Nachdem der Wert eingeschrieben wurde, wird jetzt so lange gewartet, bis das Flag SPIF von der Hardware auf 0 gesetzt und somit das vollständige Aussenden des Bytes signalisiert (siehe ⑥). Am Ende eines Sendevorgangs wird der Port _SS wieder auf „1“ gesetzt (siehe ⑧). Sie können den Ablauf am SPI-Bus



verfolgen, indem sie das SPI-Window aus dem Peripherals Menü öffnen (siehe Abbildung 230). Führen sie nun das Programm schrittweise mit F11 durch. Sie können somit die einzelnen Abläufe des SPI-Busses verfolgen. Der gesendete Wert wird im virtuellen Register SPI_OUT ausgegeben. Soll ein Wert über SPI empfangen werden, muss dieser in SPI_IN geschrieben werden. Eine ausführliche Beschreibung zu der Funktionalität der virtuellen Register (VTREGs) können sie in Teil 2 nachlesen. Die VTREGs ermöglichen ihnen auch SPI-Slavebausteine zu simulieren.

Abbildung 230 SPI-Window

SPI-Bus im Master-Mode (Interrupt-Betrieb)

Das vorangegangene Beispiel soll nun mit Hilfe des SPI-Interrupts realisiert werden. Nach dem Sendevorgang wird in der SPI-ISR das _SS-Bit auf „1“ gesetzt. Der Zustand des _SS-Bits wird auch als Erkennung verwendet, wann eine komplette Sende-/Empfangsübertragung abgeschlossen ist. Die Freigabe des SPI-ISR erfolgt über das Bit ESPI (siehe Listing 98, ①). Vergessen sie nicht, die allgemeine Interruptsperre (EA) aufzuheben (siehe ②).

Das Programm sendet nun den Wert 0x45 aus und wartet, bis der Portpin _SS den Wert „1“ angenommen hat (siehe ③). In der SPI-ISR werden die Flags zurück- (siehe ④) und der Portpin _SS auf „1“ gesetzt (siehe ⑤). Nachdem der Portpin _SS den Wert „1“ angenommen hat, beginnt der nächste Sendevorgang.

Projektname	Verzeichnis	Verwendete Sourcemodule
SPIInt_Test	Kap10_SPIMasterInt	TestSPIInt.c

Tabelle 128 Projekt SPI_Test

```
   #include <REG932.H>
   sbit _SS = P2^4;

   main()
   {
      P2M1 = 0x3C; // Freigabe der SPI-Ports (MISO, MOSI, SPICLK, _SS)
      P2M2 = 0x3C;
      P2 = 0xFF;
      SPCTL =  0xD4; // Freigabe des SPI-Blocks, MSTR = 1, SSIG = 1)
①     ESPI = 1;
②     EA = 1;

      while(1)
      {
        _SS = 0;          // SPI-Slave selektieren
        SPDAT = 0x45;     // Wert schreiben
③       while(_SS == 0); // Warten bis Schreibvorgang zu Ende
      }
   }

   void v_SPIInt(void) interrupt 9
   {
④     SPSTAT = 0xFF;      // SPI-Flags zuruecksetzen
⑤     _SS = 1;            // SPI-Slave Selektierung zuruecknehmen
   }
```

Listing 98 Sourcemodul TestSPIInt.c