Eksamen 2012  
3. semester  
I3MPS - Microprocessor systemer  
Kasper Nissen

**Spørgsmål 6**

**SPI**

**Spørgsmål**

* Hvordan virker SPI bussen?
* Hvilke parametre skal man være opmærksom på når man skal konfigurer interfacet til et spi device?
* Hvilke metoder skal implementeres i en device driver som benytter SPI?
* Hvordan kan man implementere en SPI device driver?

**Pensum:**Wikipedea's afsnit om SPI + Dokumentation i kernel tree: ~/embest/work/linux-2.6.28-omap/Documentation/spi/ spi-summary + Skim spi.h igennem (~/embest/work/linux-2.6.28-omap/include/linux/spi/spi.h)

**Exercise 7**

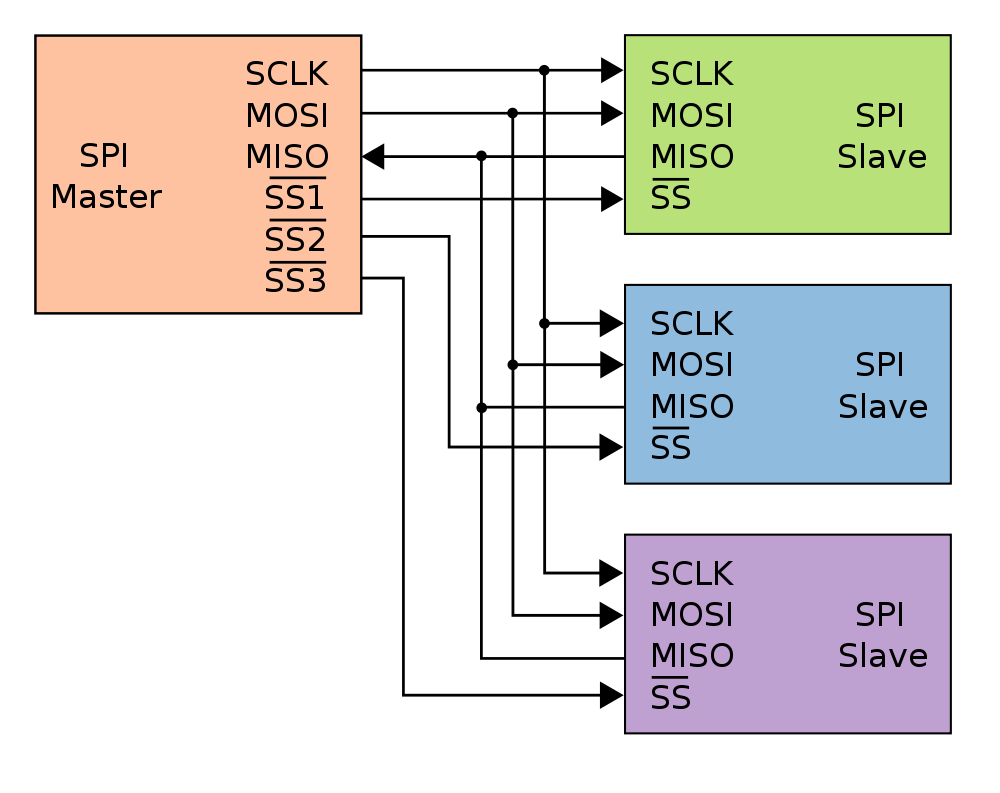
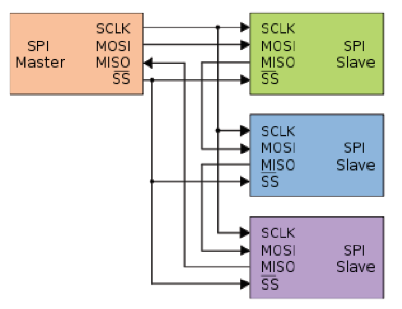
**Hvad er SPI?**

SPI er en bus og navnet står for **Serial Peripheral Interface.** SPI bussen er en synkron **seriel data link standart**, navngivet af Motorola. SPI operere i **full duplex**, hvilket jo vil sige at vi kan kommunikere begge veje. Enheder kommunikere **i master/slave forho**ld, hvor det er master enheden der opsætter rammerne. Der kan sagtens være **flere slave enheder**, som kan vælges **ved slave select** også kaldet chipselect. Et andet navn for SPI er **4-wire serial bus**.

Andre ting:

* Overførsler mellem master og slave **involvere shift registers:**
  + Master’s og slave’s registre er forbundet i en ring.
    - Master overfører **mest betydende bit(MSB) til slavens mindst betydende(LSB).**
    - Bitsene bliver skubbet fremad derefter.
* 2 måder hvorpå master kan vælge en slave til overførsel:
  + Parallel med master
    - Master har en SS(Slave Select) bit til hver slave
    - Slave select SS bit bliver brugt til at afgøre hvilken slave.
  + Serielt med master (daisy-chained)
    - Alle slave forbindes serielt med hinanden og derefter til master
    - Den tætteste slave har højest prioritet

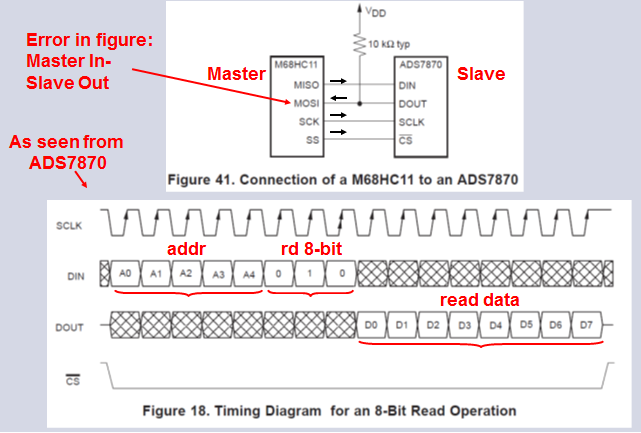
Nedenfor ses et eksempel, hvor vi har 3 slave enheder og en master enhed. De 3 slave enheder kunne f.eks. være en A/D converter, en D/A Converter og noget flash hukommelse. Som det ses på figuren så er det altså slave select(SS) der bruges til at udvælge den enhed som master vil kommunikere med. Som det ses på figuren har vi **4 forskellige signaler:**

* **SCLK**, som er en serial clock, der er et output fra master.
* **MOSI,** som er en forkortelse for master output slave input, hvilket som navnet antyder at det er hvis master ønsker at sende noget til slave.
* **MISO**, som er en forkortelse for maser input, slave output, og som jo selvfølgelig er modsat retning af MOSI.
* **SS**, Slave Select, aktiv lav, og bruges som sagt af master til at udvælge den enhed som der ønskes kommunikation med.

**Daisy-Chained**

**Parallelt**

Et eksempel på en SPI bus, kunne være eksemplet nedenfor hvor vi har en MCU(En kontrol enhed) og en A/D konverter. BEMÆRK: Der er byttet rundt på MISO OG MOSI for M68HC11.



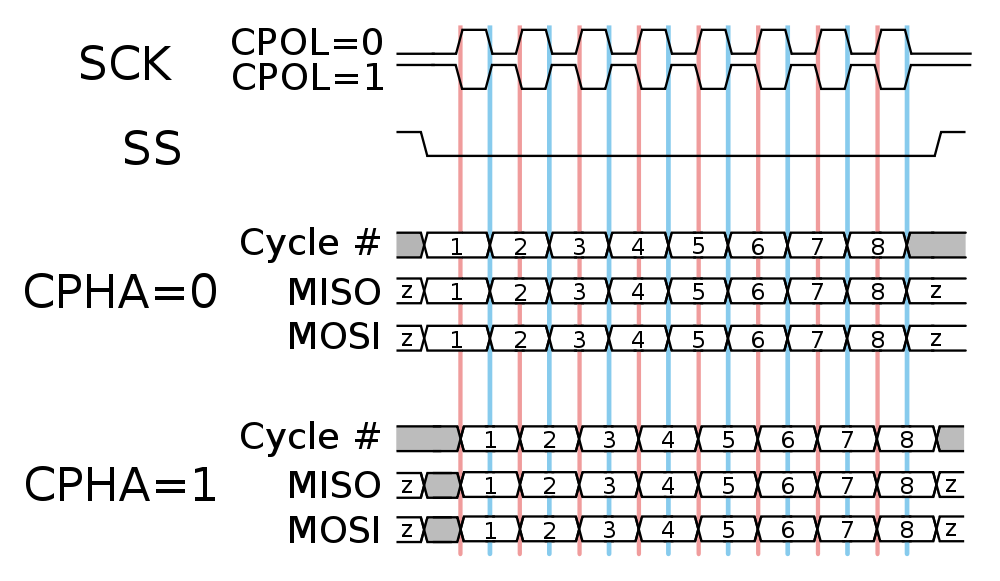
Som timing diagrammet nedenfor viser, så sker der det når slave enheden skal lave en læse operation at, de først 5 bit er adressebit,som angiver hvilket register som enheden vil skrive fra eller til. Dernæst kommer er 3 bits, som angiver hvilken størrelse af data som enheden vil læse, samtidig fungere den også som valg mellem read/write.

**Hvilke parametre skal man være opmærksom på når man skal konfigurer interfacet til et SPI Device?** Clock modes: (Clock polaritet og fase)

Udover at sætte clock frekvensen skal master også opsætte clock polariteten og fasen, ifølge Freescale’s SPI Block Guide – som de fleste producenter følger. Denne nævner op mulige opsætninge: CPOL og CPHA.

På figuren nedenfor ses et timing diagram, og en forklaring på dette følger:

* Ved CPOL = 0 ses det at base værdien på clocken er lig med 0.
  + For CPHA=0, bliver data **opfanget på** **clockens rising edge** og data bliver **ført videre på falling edge.**
  + For CPHA=1, bliver **da opfanget på clockens falling edge** og data **bliver ført videre på rising edge.**
* Ved CPOL = 1 ses det at base værdien på clocken er lig med 1.
  + For CPHA = 0, **fanget på falling edge** og **ført videre på rising edge**
  + For CPHA = 1**, fanget på rising edge** og **ført videre på falling edge.**



Dette medfører 4 forskellige clock modes:

* + Mode 0 : CPOL/CPHA = 00 (Original Microwire)
  + Mode 1 : CPOL/CPHA = 01
  + Mode 2 : CPOL/CPHA = 10
  + Mode 3 : CPOL/CPHA = 11

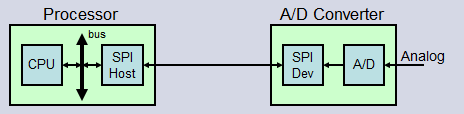
Disse skal bruges til at konfigurer SPI Host(Master) til den enhed der nu er tilsluttet. (disse informationer kan findes i data bladet for slaveenheden)

Det er vigtigt at både slave og master har samme mode, ellers kan der ikke kommunikeres.

**Opsamling på SPI Hardware**

* SPI bliver oftest brugt for middel hastigheds board adgang
  + < 70 Mb/s
  + A/D Convert
  + Serial Flash Hukommelse
* Normalt IKKE daisy chained(første slave enheds output til næste slaves input)
* En chip select pr. enhed
* Master/slave princippet
  + Master forespørger data på slave
  + Læsning fra en enhed kræver
    - Write(read\_request), read(data)
* Flere forskellige clock modes
  + CPOL/CPHA

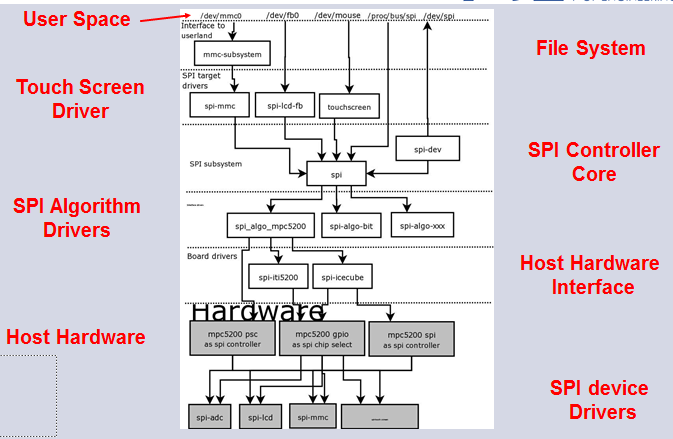
**SPI I Software**



I software skal følgende implementers:

* SPI Host
  + Hvordan skal der kommunikeres (SPI Host Driver)
* SPI Device
  + Hvordan et SPI device kommunikere på SPI bussen, f.eks. antallet af bits (SPI Device Driver)
* Device Driver
  + Hvordan selve devicet virker(A/D Converteren), hvilket registrer skal konfigures, sample rate mm.

**SPI i Linux**



**Hvilke metoder skal implementeres i en device driver som benytter SPI?**

SPI Protokol Driver

Modsat en charactor device driver som jo har init/exit, open/close/, read/write måske Ioctl, så har en SPI Protocol driver følgende metoder:

* **Probe** –Kaldes af SPI Host når en SPI Protokol driver er registeret. Bliver brugt til at konfigurer antallet af bits per ord (bits per word)
* **Remove** – Kaldes når SPI driveren skal fjernes igen, minder meget om char device drivers exit metode.
* **Shutdown** – En standard shutdown
* **Suspend** – En standard suspend
* **Resume** – En standard resume

Dette er en statisk implementering af SPI, kan også implementeres dynamisk i en Device driver, med nogle funktions kald, men kan være lidt kompliceret da det ikke er inkluderet i kerne træet.

**Hvordan kan man implem. et device driver modul med SPI? (init/exit)**

Når man skal implementere et device driver modul med SPI -> bruge funktionerne init/exit:

* **Init:**
  + En reference til SPI host på angivet busnummer:
  + struct spi\_master\* spi\_busnum\_to\_master(u16)
  + Allokere SPI device med SPI host:
  + struct spi\_device\*spi\_new\_device(struct spi\_master\*, struct spi\_board\_info\*)
  + Registrere den nye SPI driver:
  + int spi\_register\_driver(struct spi\_driver\*)
* **Exit:**
  + Afregistrer
  + Deallokere
  + Frigiv busnummer

**Hvad er design processen for at implementere en SPI device driver?**

Designprocessen:

* Find skema for CPU device SPI forbindelse
* Find hvilken bus & Find hvilken chip select
* Find device datasheet for SPI information
* Find SPI maxspeed & hvad CPOL og CPHA skal være & word størrelsen
* Find device datasheet for device initiering
* Find registreværdier & antallet af bytes der skal skrives/læses

**Snak om opgaven. Undersøg om det ikke er en dynamisk implementering.**