SPI (SW/HW)

Hvordan virker SPI bussen?

- Serial Peripheral Interface
- SPI bus er et **serielt data link**, der opererer i **full duplex mode** (sende/modtage samtidig).
- Kører efter en **Master-slave princip:**
    - Master ønsker data fra slave
- **2 linjer at læse på:**
    - **MOSI** (MasterOutSlaveIn)**:** Master sender et bit på linjen, som slave læser fra
    - **MISO** (MasterInSlaveOut): Slave sender et bit på linjen, som master læser fra
- Overførsler mellem master og slave involvere to **shift registers**:
    - Master's og slave's registre er **forbundet i en ring**:
        - Master overfører **mest betydende bit til slavens mindst betydende**
        - Bitsene bliver **skubbet fremad** derefter
- En master kan have forskellige slaves.
- Der er **2 måder** hvorpå masteren kan vælge en slave til overførsel:
    - **Independant slave SPI configuration:**
        - Masteren har en **linje for hver slave**
        - Slave select **SS bit** bliver brugt til at afgør hvilken slave
    - **Daisy Chain SPI configuration:**
        - Alle **slaves er forbundet til hinanden** og derefter masteren
        - Den **tætteste slave** har **højest prioritet**

Hvilke parametre skal man være opmærksom på når man skal konfigurer interfacet til et SPI device?

- **Clockfrekvens** hos master skal være **=<** slavens max clockfrekvens. -> findes i datablad
- **Clock modes** angiver, hvordan data er klar til at blive aflæst:
    - **CPOL** = 0 aktiv høj:
        - **CPHA** = 0: data er **modtaget på rising edge**, data bliver **sendt på falling edge**
        - **CPHA** = 1: data er **modtaget på falling edge**, data bliver **sendt på rising edge**
    - **CPOL** = 1 aktiv lav:
        - **CPHA** = 0: data er **modtaget på falling edge**, data bliver **sendt på rising edge**
        - **CPHA** = 1: data er **modtaget på rising edge**, data bliver **sendt på falling edge**
- Master og slave skal have **samme word-størrelse**

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

- **Probe:** Kaldt af **SPI master** når en **SPI protokol driver er registreret**.
- **Remove:** Når SPI driver **afregistreres.**
- **Shutdown:** Shutdown callback, brugt ved system state overførsler
- **Resume:** Resume callback, brugt ved system state overførsler
- **Suspend:** Suspend callback, brugt ved system state overførsler

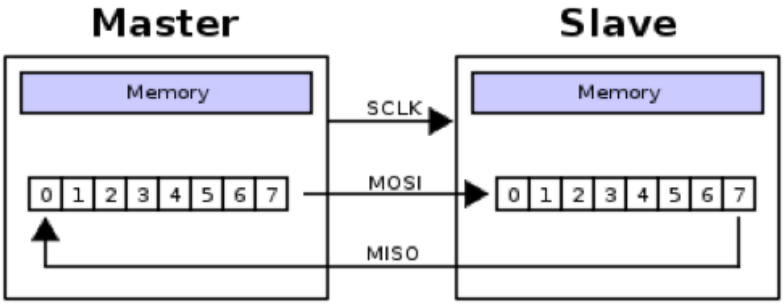
Hvordan kan man implem. et device driver modul med SPI? (init/exit)

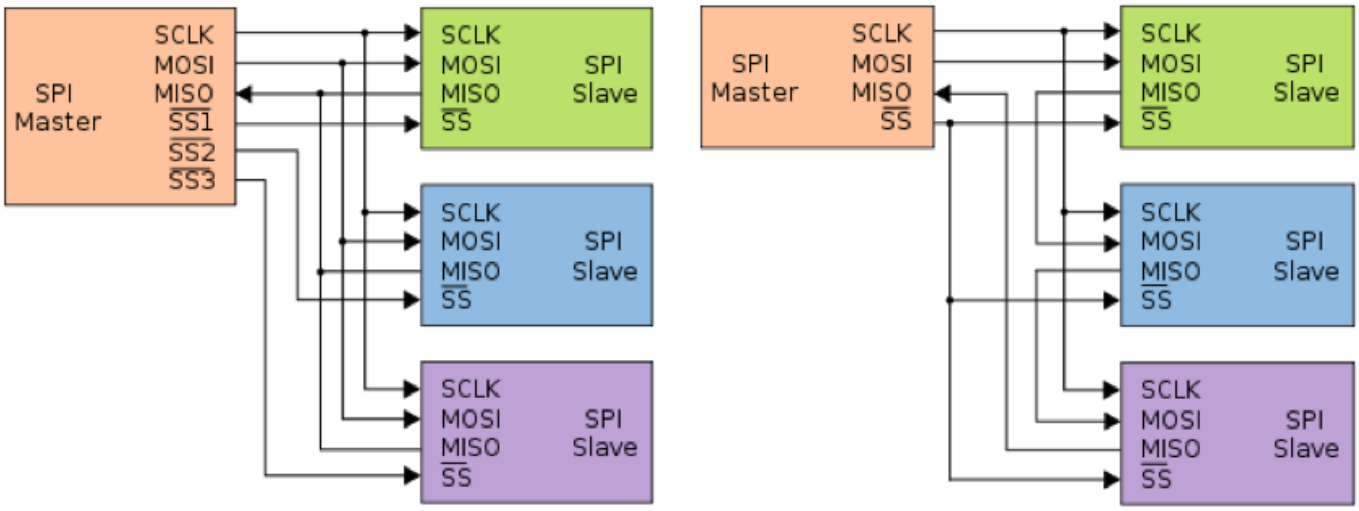
- Når man skal implementere et device driver modul med SPI -> bruge funktionerne **init/exit:**
    - **Init**:
        - En reference til SPI host på angivet busnummer:
            - **struct spi_master* spi_busnum_to_master(u16)**
        - Allokere SPI device med SPI host:
            - **struct spi_device*spi_new_device(struct spi_master*, struct spi_board_info*)**
        - Registrere den nye SPI driver:
            - **int spi_register_driver(struct spi_driver*)**
    - **Exit:**
        - Afregistrer
        - Deallokere
        - Frigiv busnummer

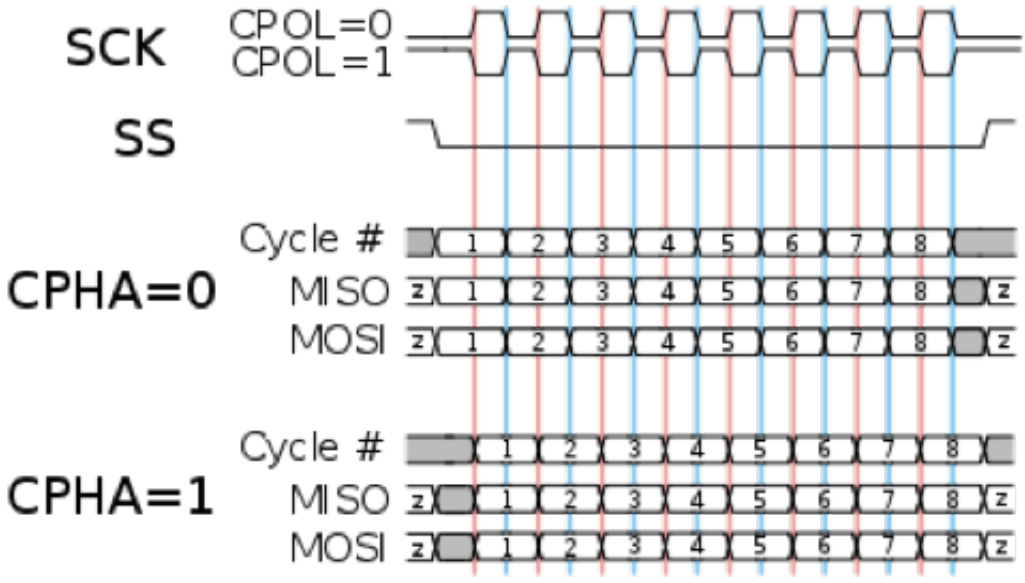
Hvad er design processen for at implementere en SPI device driver?

- 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

SPI (SW/HW)


Master
Slave
Memory
Memory
SCLK
MOSI
MISO
0 1 2 3 4 5 6 7
0 1 2 3 4 5 6 7


SPI Master
SCLK
MOSI
MISO
SS1
SS2
SS3
SPI Slave
SCLK
MOSI
MISO
SS


SCK
CPOL=0
CPOL=1
SS
Cycle #
CPHA=0
MISO
MOSI
CPHA=1
1 2 3 4 5 6 7 8
z