6. SPI

Hvordan virker SPI bussen?

- Master/slave opsætning
- Masteren bestemmer når der må sendes eller modtages
- Som regel en master og en eller flere slaves
- Hver slave har enten sin egen CS, eller er daisy chained
- Flere mastere er mulige
- Består af 2 data forbindelser, clock, CS, gnd
  - MOSI, MISO
  - CS0-N
  - Clock fra master

Overførsel:

1. Masteren Sætter CS lav og clocken generes
2. Masteren sender data til devicen på MOSI
   Slave sender data data på MISO
3. CS sættes høj

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

- CPOL - 0 clock går fra høj til lav, 1 omvendt
- CPHA - Bestemmer hvornår data skiftes/læses
- CPOL/CPHA modes:
  - 00 - Mode 0 (Original Microwire)
  - 01 - Mode 1
  - 10 - Mode 2
  - 11 - Mode 3
- SPI host opsættes efter slave enheden

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

- Specielt for SPI delen er **probe/remove/shutdown/resume/suspend**
- SPI- init, exit, read, write kan alle implementeres som en del af char driverens metoder

Hvordan kan man implementere et device driver modul med SPI(Hvad skal i init / exit ...)?

- **SPI board info struct**
  - Modalias
  - Bus nr.
  - Chip select
  - Max frekvens
  - controller data (Device config struct)
  - mode
- **SPI driver struct**
  - name
  - bus
  - owner(THIS MODULE)
  - probe/remove methoder
- **Device probe metode**
  - Bits per word
  - spi setup
- **SPI init**
  - SPI device sættes op med SPI master og boardinfo struct
  - SPI driver registreres
- **SPI exit**
  - Unregister SPI device
  - Unregister SPI driver
  - (frigiv bus, dealloker, afregistrere)

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

- Find ud af slave enhedens egenskaber, CS etc.
- Opsæt alle parametrene i SPI delen af char driveren, i de korrekte SPI relaterede structs
- Implementer probe, remove, init, exit for SPI delen
- Implementer read/write metoder
- Implementer character device delen