0.1 Introduktion

I denne del af øvelsen vil vi nu arbejde med kommunikations interfacet. Der vil hertil benyttes 2 styk PSoC 5LP, som implementeres som henholdsvist slave og master i systemet. Vha. en UART-forbindelse til PC, vil vi så forsøge at implementere vores master så instrukser sendt fra PC kan sendes gennem SPI-protokollen fra master til slave.

0.2 Overvejelser

Kommunikation mellem master og slave vil som sagt foregå gennem SPI. Vores fokus vil derfor ligge i at sikre, at signalet sendt fra master og signalet modtaget af slave benytter den korrekte protokol.

Vi forventer at benytte AD til at teste hvorvidt signalerne sendes og modtages korrekt, og om det rigtige antal bits benyttes. Desuden er vi blevet informeret om at SPI-protokollen i PSoC-Creator benytter en cirkulær buffer, hvilket vi vil uddybbet senere i opgaven.

0.2.1 SPI

SPI fungerer som en datastrøm imellem 2 devices. Det er kun muligt at sende på en SPI, hvis der også modtages fra den samtidig. Modtageren og senderen fungerer som en cirkulær buffer, som læses og skrives til hele tiden. Man bør derfor være opmærksomme på ikke at komme bagud i bufferen, så vores data bliver overskreveet. Da protokollen til denne opgave er forhodlsvist nem, og kan anses for binær, er det valgt simpelthen at ignorere bufferen, og nulstille den, hver gang den bruges.

Fobindelsen mellem Master og Slave er forholdsvist simpel i denne implementering. Der skal oprettes en MOSI, MISO og en SCLK. Det er vigtigt vores SPI Master og SPI Slave kører på samme modes. Vi har defor brugt stadardinstillingen på modes, som er $CPHA = 0$ og $CPOL = 0$. CPOL er vores clock polaritet. I mode 0 starter klokken LOW, når der ikke tælles. CPHA fortæller hvornår vores bits bliver læst. Ved 0 er det ved starten af en clock puls der samples, ved 1 er det i slutningen der samples.

Til at opstille Slave modulet, skal vi desuden bruge en Slave Select. Da der i dette system kun eksisterer en enekelt Slave, vælges der at forbinde Slave Select direkte til GND.

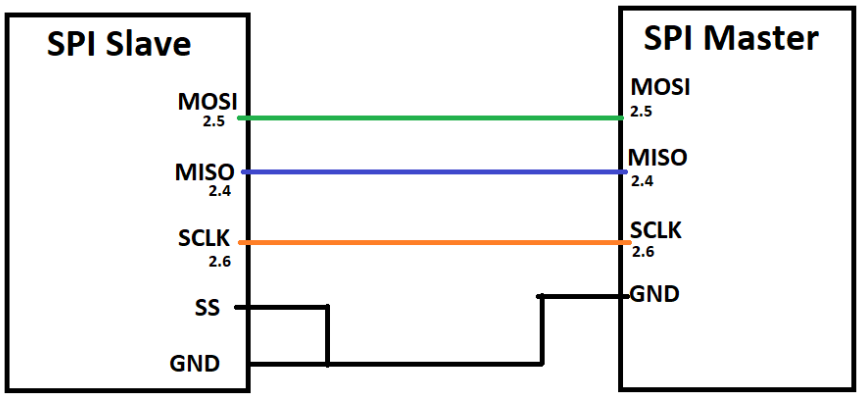

Figure 1: SPI setup diagram

Signal	Beskrivelse
MOSI	Står for Master-Out-Slave-In. Signal i bits der sendes fra master til slave.
MISO	Står for Master-In-Slave-Out. Signal i bits der sendes fra slave til master.
SCLK	Master intern clock, styrer synkronisering af sending af bits.
SS	Master Slave Select. Sættes LOW for at kommunikere til slaven er der sendes til den.
GND	Fælles stel forbindelse for slave og master

0.3 Implementering

Vi vil her introducere hvordan vi har opbygget vores master-slave system, og hvordan det er blevet implementeret i softwaren.

0.3.1 SPI Slave

SPI Slave top design ser ud som på figur 2. Den består af vores SPI modul, en UART til debugging samt en LED. LEDen er forbundet viar hardware til pin 2.1, UART er forbundet Tx=12.7 og Rx=12.6, da disse er fobundet til USBen. SPI har ikke er forudbestemt pin opsætning, så den er forbundet MISO=2.4, MOSI=2.5 og SCLK= 2.6. Slave Select er bundet direkte til GND, da vi ønsker at forsimple implementeringen.

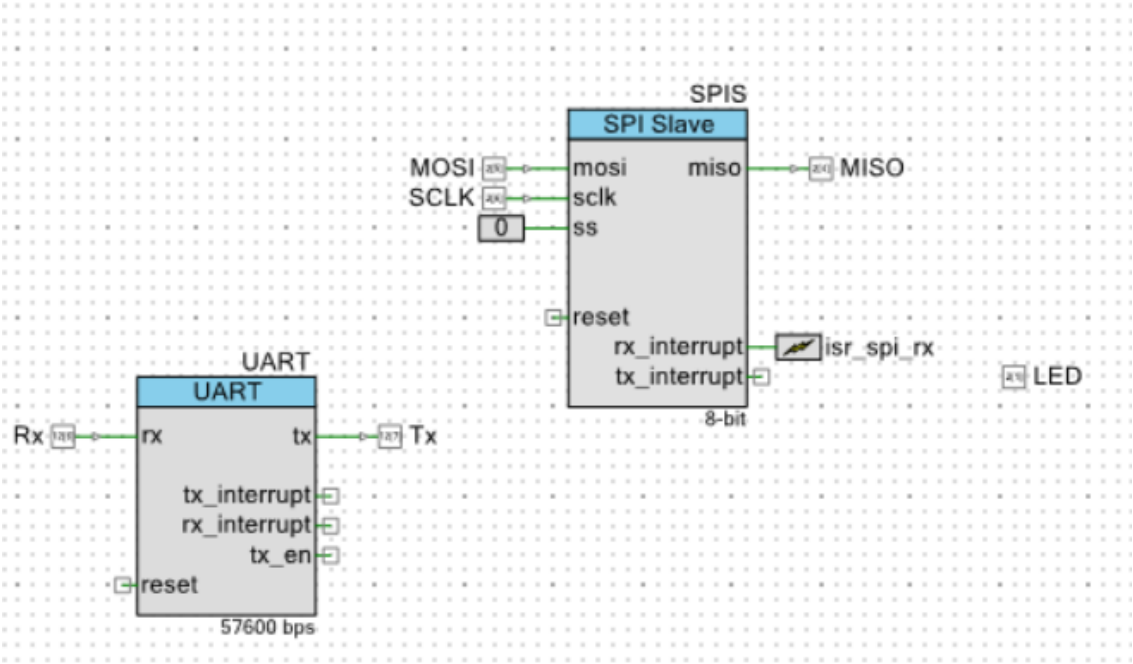

Figure 2: SPI Slave top design

Selve Slaves funktionalitet ligger i den SPI interrupt handler der er beskrevet i listings 1. Funktionen tager læser data fra SPI bufferen, printere dette til UART-debuggeren, nultiller bufferen og tager så stilling til LEDens tilstand ud fra hvilken data er læst. Da funktionaliteten af SPI slave er den samme i de to implementeringer (med og uden knap) er her kun vist en.

Listing 1: SPI Slave

```
1  CY_ISR(isr_spi_rx_handler)
2  {
3     uint8_t data = SPIS_ReadByte();
4
5     // Write to terminal for DEBUGGING
6     char buf[20];
7     sprintf(buf, "Data received: %x\r\n", data);
8     UART_PutString(buf);
9
10    // Clear buffer, makes it easier
11    SPIS_ClearRxBuffer();
12
13    switch(data)
14    {
15        case 0xcc:
16        {
17            LED_Write(1);
18        break;
19        }
20        case 0x55:
21        {
```

```
22                  LED_Write(0);
23                  break;
24              }
25              default:
26              {}
27      }
28  }
```

0.3.2 SPI Master

Da der både skal implementeres en SPI Master der kan styres via UART, og en der kan styres via en knap på PSoC, er der implementeret to forskellige versioner ad SPI Master. Deres top design kan ses her under på figur 3 og 4. De er begge implementeret med en UART, hvor dens pins er sat Tx=12.7 og Rx=12.6. SPI er sat op på samme måde som i SPI Slave, med MISO=2.4, MOSI=2.5 og SCLK=2.6. Der er ikke brug for at definere Slave Select, da dette ikke er relevant for opgaven. Desuden er der tilføjet en software forbindelse til kanppen på PSoC boardet, via pin 2.2.

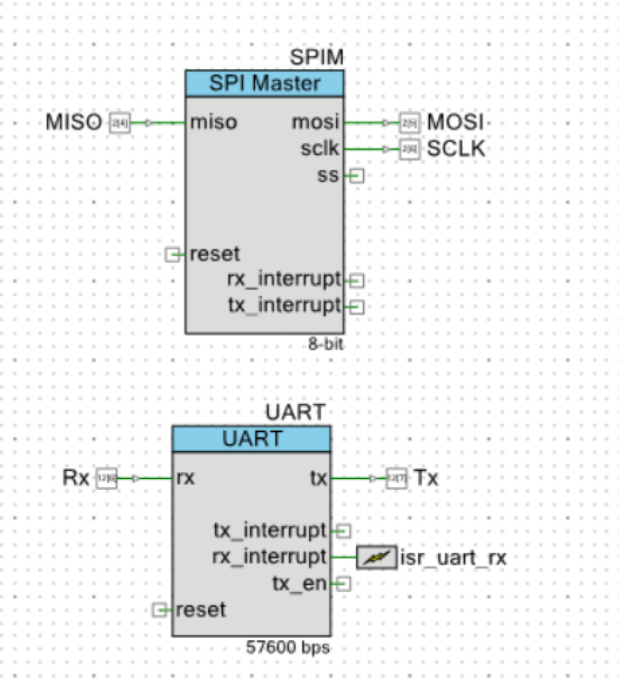

Figure 3: SPI Master top design

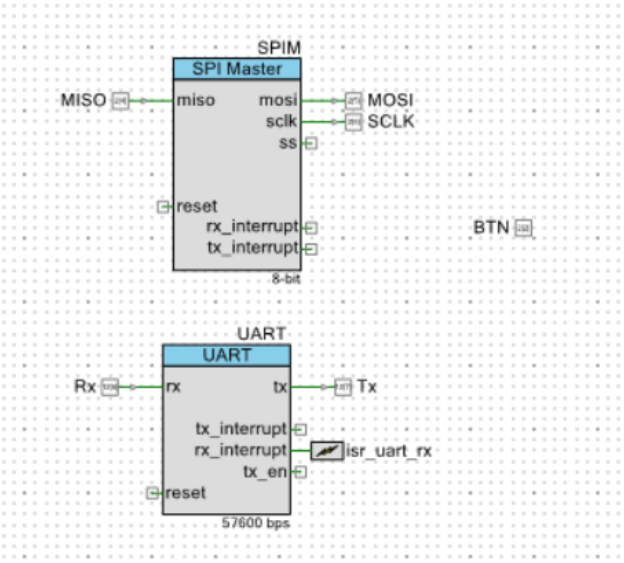

Figure 4: SPI Master top design - med kanp

På listing 2 kan ses den funktion der håndterer hvad der modtages på UART. Når der er modtaget besked via UART,

Listing 2: Håndtergin af Bytes Received fra UART til SPI Master

```
1  void handleByteReceived(uint8_t byteReceived)
2  {
3    switch(byteReceived)
4    {
5        //ON case
6        case 'i':
7        {
8            SPIM_ClearTxBuffer();
9            SPIM_WriteTxData(0xcc);
10           break;
11       }
12       //OFF case
13       case 'o':
14       {
15           SPIM_ClearTxBuffer();
16           SPIM_WriteTxData(0x55);
17           break;
18       }
19       default:
20       {}
21   }
22 }
```

Listing 3: Håndtering af knappens input på SPI Master - med knap

```
1  for(;;)
2  {
```

```
3         CyDelay(20);
4         if(!BTN_Read())
5         {
6             SPIM_ClearTxBuffer();
7             SPIM_WriteTxData(0xcc);
8         }
9         else
10        {
11            SPIM_ClearTxBuffer();
12            SPIM_WriteTxData(0x55);
13        }
14    }
```

0.4 Dokumentation

På Figure 5 nedenfor ses vores endelige opstilling. Forbindelserne mellem slave og master, beskrevet i implementeringsafsnittet, er etableret som specificeret , og vi ser som forventet at vi kan tænde og slukke for LED'en på slave PSoC'en ved at sende i/o til vores master PSoC.

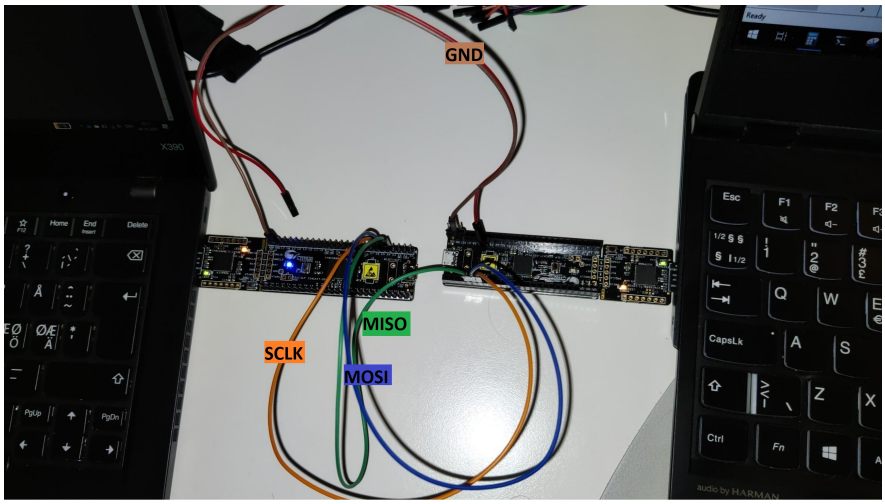

Figure 5: DER SKAL TEKST IGEN

For at få en bedre forståelse af hvad der sker signalmæssigt, og for at sikre systemet ikke bare virker ved et uheld, benytter vi Analog Discovery Oscilloskop til at måle i/o signalet sendt mellem PSoC parret. På Figure 6 ses vores måling af signalet for at tænde LED'en, mens Figure 7 viser signalet sendt for at slukke LED'en. Den blå graf er her vores clk-signal, mens den orange graf er vores SPI-signal.

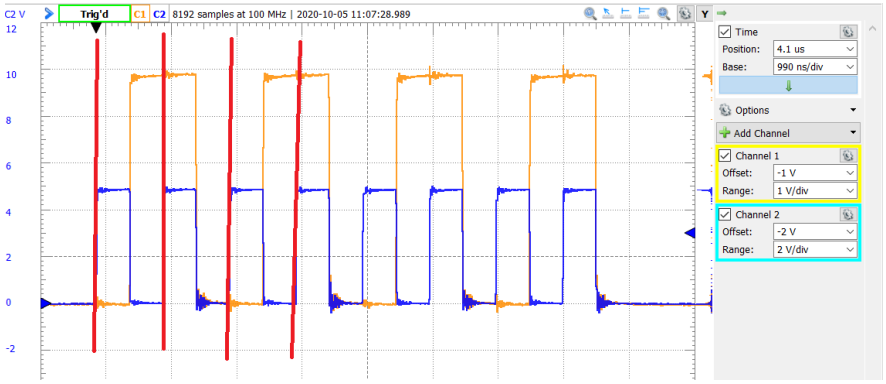

Figure 6: DER SKAL TEKST IGEN

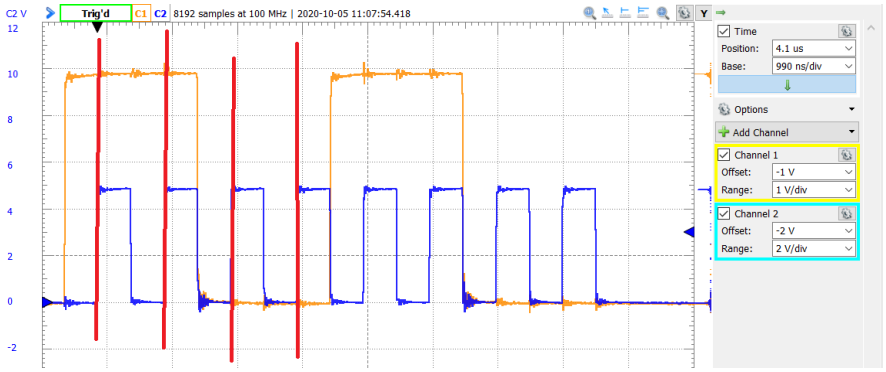

Figure 7: DER SKAL TEKST IGEN

0.5 Diskussion

Som vi først ser på Figure 5 virker vores implementering, og vi kan tænde og slukke for vores slave LED ved at sende et signal til vores master PSoC. Idet vi ville sikre at SPI protokollen blev kørt korrekt, valgte vi specifikke hex-tal til at repræsentere I/O signalerne.

Dette gav dog nogle problemer under implementeringen, da vi ved målinger med Analog Discoverys Spy funktion fandt at signalet modtaget af slave PSoC'en blev forskudt en halv byte ift. hvad der blev sendt af master. Vi fandt at fejlen opstod hvis et tidligere signal ikke var blevet modtaget korrekt. Hvis bufferen så ikke blev tømt efterfølgende, lagde protokollen blot dele af det nye signal ind i bufferen oveni fejlsignalet.

Problemet opstår, idet SPI-protokollen ikke har adgang til en acknowledge, og derfor blot sender en stream af bits fra master til slave. En evt. løsning kunne være at gøre brug af SPI slave-select. Når SS signalet sættes lav for at indikere til slaven at der nu sendes specifikt til den, kan bufferen så tømmes. Vi valgte imidlertidig ikke at bruge denne løsning, da vi ikke ønskede at overkomplicere opgaven.

Hvis vi kigger på Figure 6 ser vi det korrekte 'i' signal modtaget af slaven. Vi benyttede i dette tilfælde hex-tallet x55, i binært '1010101', hvilket er præcis den kombination vi ser på figuren. Ligeldes benyttede vi hextallet xcc, '11001100' i binært, til vores 'o' signal, og vi ser på Figure 5 at signalet stemmer overens.

Det er her værd at pointere at amplituden af vores SPI-signal er dobbelt størelsen af vores clock. Dette skyldes at vi har stillet på akserne, for at gøre målingerne mere visuelt overskuelige, da vi ikke er videre

interesserede i amplituderne af vores signaler.

0.6 Konklusion

[TEXT HERE]