FYS3240 Lab 2

XMEGA128B1 kontrolleren har en innebygget LCD-driver. Laboppgave 2 vil handle om å utnytte denne til å skrive ut tekst på LCD-skjermen som sitter på kortet.
I denne oppgaven skal dere lære å bruke makroer i tillegg. Makroer kan brukes på to måter: som en kortversjon av en funksjon eller som en "erstatning". En fordel med funksjons-makroer er at de ikke har samme overhead som en funksjon. En makro kan brukes til å erstatte en verditilordning, for eksempel hvis man vil sette en verdi til et register. En konstant defineres på mye av den samme måten.

Et eksempel på hver type:

```
#define ADD_MACRO(verdi1, verdi2) (verdi1 + verdi2)
#define PORTB_ALLHIGH (PORTB.OUT = 0xFF)
#define CONSTANT verdi
```

Tips: Det kan være lurt å konfigurere knapper og LEDs, da disse kan brukes til enkel debugging. Det kan også være lurt å sette opp en av knappene til å avslutte main-løkken slik at det ikke er nødvendig å skru av kortet for å kunne laste opp ny kode.

Oppgave 1:
Dere har fått utdelt et ferdig kodeskall med følgende rutiner og makroer:

`#define LCD_HOME:`
Setter segmentadressen til begynnelsen av linjen. (Hint: linjen begynner på posisjon 12)

`#define LCD_CLEAR:`
Hint: det er en bit i et av registrene spesifikt for dette.

`#define LCD_BACKLIGHT_TOGGLE`
Aktiverer/deaktiverer bakbelysningen. Dette gjøres ved å sette power-pin til LCD-skjermen høy. Hvor dette skal gjøres finner dere i "Atmel XMEGA-B1 Xplained Hardware User Guide". (Hint: XOR)
`init_lcd():`

I denne metoden skal de aktuelle registrene for å bruke LCD-skjermen initialiseres. (Se databladet fra side 310)

Før vi får brukt LCD-driveren til noe nyttig må vi sette opp et klokkesignal til den. Denne velges og aktiveres i systemklokka sitt kontrollregister for RTC (Real-Time Counter, databladet side 87).

LCD-driveren har også sin egen klokkeprescaler, som bestemmer oppdateringsfrekvensen. Denne oppdateringsfrekvensen bestemmer hvor ofte driveren genererer en interrupt. Denne frekvensen bør derfor ikke være høyere enn vi har behov for. Kommenter valget dere tar i forhold til dette.

`lcd_write_char(uint8_t c):`
Denne metoden brukes til å skrive en karakter til skjermen.

Bent Furevik/ Snorre Bjørnstad, revidert 02.02.2015

For å sette ut en character på LCD-skjermen kan vi skrive en ascii-character direkte til CTRLH-registeret til LCD-driveren, da driveren tolker det selv. (ref. datablad side 317)
Hvor karakteren blir plassert styres av telleren STSEG i CTRLG-registeret. Denne telleren inkrementeres automatisk med mindre CTRLH[7] settes høy. Den vil i såfall dekrementere.

`lcd_write_char_p(uint8_t c, int position)`:
I denne rutinen skal posisjonen for karakteren spesifiseres manuelt. Dette kan være mer effektivt hvis for eksempel bare et felt på LCD'en skal oppdateres.

`lcd_set_blink(uint8_t blinkrate)`:
Denne metoden skal aktivere blinking av segmentene. Hvordan dette gjøres finnes i databladet. Her skal det være mulig å velge frekvens for blinkingen, enten 0.5, 1, 2 eller 4 Hz.

`lcd_write_int(int value, int position)`:
Denne metoden brukes til å skrive et tall til skjermen. Det greieste her er å bruke `lcd_write_char()`, da det ikke kreves mye for å oversette fra int til character. (Hint: ASCII-tabellen)

Oppgaven er å få kodeskallet i fys3240_lcd.h til å fungere som beskrevet og lage en testapplikasjon i fys3240_lab2.c for å vise at dette fungerer.

Oppgave 2:
Del 2 har følgende rutiner i fys3240_lcd.h:
`init_lcd_interrupts(void)`:

Til denne oppgaven skal vi sette opp interrupts for LCD-driveren. Pass på å sette disse opp til å trigge slik at det går noen frames mellom hver gang. (Hint: databladet s. 314)

Husk å programmere opp Interrupt kontrolleren (databladet: s 119) slik at interrupts faktisk er slått på. Dere må også aktivere interrupts i applikasjonen med funksjonen `sei()`. Merk at interrupts kan slå til med en gang denne kommandoen er kjørt.
Denne rutinen skal kalles fra `lcd_init()` fra oppgave 1.

`lcd_callback_lab2(void)`:
Denne rutinen kalles fra ISR'en (Interrupt Service Routine) som hører til lcd-driverens interrupt-vektor. Den skal også sjekke hvilken eventuell touchbryter som er trykket og sette hvilken tekst som blir satt fram til en annen bryter blir trykket. Teksten som gis av CS1 skal være en tellerverdi som inkrementeres hver gang CS1 trykkes.

Alle variabler som brukes i en interrupt-rutine, og som skal brukes et annet sted i programmet, bør være volatile. Dette forteller kompilatoren at verdien kan endres når som helst (for eksempel ved en interrupt) slik at den må lastes inn på nytt hver gang den leses. Hvis dette unnlates vil kompilatoren gjerne optimalisere programmet slik at det ikke fungerer som tiltenkt.

Navnet på interrupt-vektoren kan være vanskelig å finne i databladet, men en komplett liste med alle vektorene ligger også i io-headeren for mikrokontrolleren (iox128b1.h).

Eksempel på ISR:

```
ISR(<vektornavn>)
{
/* Kode */
}
```

I fys3240_lab2.c:
ISR(<lcd-vektor>):
Kalles når interrupt-vektoren slår til. Denne skal ikke skrives i headerfila, da vi vil lage en ny i neste laboppgave.

Les mer om hvordan Interupts fungerer på XMEGA mikrokontrollerene her:
http://www.atmel.com/Images/doc8043.pdf (AVR1305: XMEGA Interrupts and the Programmable Multi-level Interrupt Controller)