Opgave 1 output, input og port-register visning på OLED display

Ole Schultz DTU Engineering Technology ETCS 02/10/2024

1 Formål

Formålet med opgaven er:

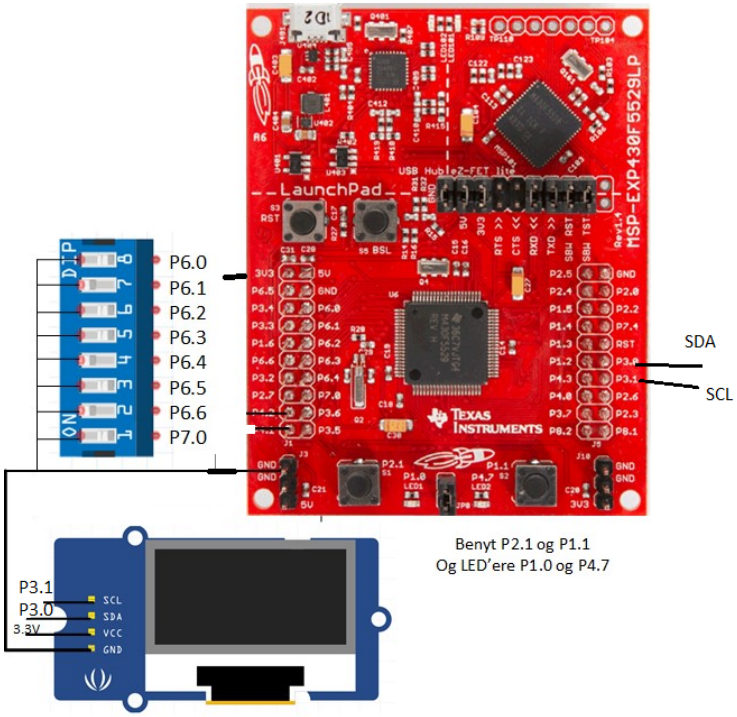
- at forstå at benytte en timer i MSP430 til at styre et ur programmeret i MSP430 og få vist tiden fortløbende på OLED displayet
- at kunne programmere et timer compare interrupt til styring et ur
- at kunne forklare interrupt fra en timer og en tryk-kontakt
- at kunne opdatere tiden for henholdsvis sekunder, minutter og timer indlæst fra en dip-swtich vha. en tryk-kontakt, når der trykkes på kontakten

2 Læringsmål

- At lære at initialisere en timer i MSP430 til at tælle i count up mode i et program
- At få skrevet et program til skrive urets tid i hh:mm:ss format på OLED display
- At læse input fra 8 bit dip-switch ind gennem en port og benytte værdien til opdatering af henholdsvis sekunder (ss), minutter(mm) og timer (hh)
- at programmere relevante selection statements til at styre indlæsning af tiderne i uret når der trykkes på tryk-kontakt og til håndtering "meddelser"fra interrupt service rutinerne.

3 Schematic til forbindelse af OLED display og lysdiode

Her ses hvordan forbindelserne skal være for at benytte det i nedenstående programmerings koncept - terminaler på OLED display kan være i anden rækkefølge end det der er vist i figuren for OLED!



- Benyt switch på P2.1, dip-switch som vist - Forbind sck til sck på P3.1, sda på P3.0 til sda, Vcc til 5V og gnd til gnd på OLED-display.

4 Fremgangsmåde:

(a) Tegn et flowchart for hvordan et digitalt ur kører med sekunder ss, minutter mm og timer. Husk midnat er et tilfælde hvor uret starter forfra

(b) Opret et nyt projekt i Visual studio code insider vha. platformio

(c) Includer OLED displayfilerne i projektet som du gjorde i opgave 1 ellers se her `https://www.youtube.com/watch?v=yZ50jOVLVRw`

(d) Slet setup() og loop() funktionerne i main.c og skriv en int main() { while(1) { her skrives dit program }}

(e) Før while(1) initialiser display som i opgave 1. Skriv en lille tekst ud som check af at display virker. Med funktionen: ssd1306_printText(0,0,"din tekst").

(f) Build projektet og upload programmet i MSP430 se på display.

4.1 TimerA0 opsætning

TimerA0 skal konfigureres til at køre i count up mode og det betyder når maksimum nås resetter timerens tæller til 0. clock-frekvensen ind på timeren styrer hvor hurtigt timeren (tælleren) tæller. Så eksempel - hvis clock frekvensen er 1 Mhz (en million Hz) så tager et tælle skridt 1/1000000 = 1 mikro-sekund. Hvis der ønskes et sekund interval skal tælleren tælle til en million, så ved vi tiden er 1 sekund. I princippet er det tællerens opgave at styre sekunderne. Ud fra et sekund kan uret så programmeres til at gå, så vi ser

sekunder, minutter, og timer.

I msp430 er der mindst 3 slags clock frekvenser, SMCLK (subsystem clock) fast clock -default 1.048 MHz, Aclk langsom clock på 32 kHz, Mclk (master clock) fast clock. I opgaven her skal timerA0 benytte SMCLK dvs. 1,048MHz. Timeren skal konfigureres til at benytte den og sættes op til at nedskalere den og styre at main programmet bliver afbrudt vært sekund så uret kan gå i main(). Det skal gøres ved at sætte nogle få bit i timerA0 control register. Timeren skal sættes til at tælle op til en sammenligningsværdi og derefter resettes når sammenligningsværdi er lig tællerens værdi- det kaldes et compare match. På det compare match skal interrupt aktiveres. Tiden i mellem to interrupts skal sættes op til at være 1 sekund. Det kan opnås ved at skalere clock frekvensen ned til timerens tæller og sætte en sammenligningsværdi. Når sammenligningsværdi er nået så starter timerens tæller forfra. Tiden der ønskes mellem to sammenligninger kan findes vha. følgende formel: $$TimeRequired = \frac{ID \times (1 + TAxCCRn)}{f_{source}}$$.

I den indgår tre variable clock skalering ID, frekvensen ind i timeren fsource og sammenligningsværdi TA0CCR. og du bliver nødt til at sætte ID til en værdi og den kan sættes til 2,4,8,64 og fsource = 1,048MHz udfra det kan man så regne sammenligningsværdi ud for 1 sekund i timerequired - så $$TAxCCRn = \frac{f_{source} \times TimeRequired}{ID} - 1$$ - så prøv nu at regne en værdi ud for TA0CCR - når time required er 1 sekund, og fsource=1048000 Hz, og ID er 64.

(a) Tælleren skal nu initialiseres til at give 1 sekunds delay med den udregnede sammenligningsværdi i TA0CCR og en neddeling af fsource med 64.
Kontrolregister TA0CTL register skal sættes op til at benytte SMCLK, og neddele fsource med 8 samt sættes til at tælle op - det gøres med tre forskellige bit TASSEL_2, ID_3, MC_1, se side 17 i timerA userguide, er bilag til opgaven. For at få nedskaleret fsource ned 64, så skal en faktor 8 sættes i et register for sig TA0EX0 Register Timer_A0 Expansion Register 0, i det skal sættes TAIDEX_3. Og interrupt på compare match skal aktiveres med et bit CCIE sat i et 3. register, TA0CCTL0 Register Timer_A0 Capture/Compare Control 0 Register. Samt globalt interrupt skal enables med et macrokald __bis_SR_register(GIE);

(b) Se eksempel i slides for lektion 4 Og til sidst skal servicerutinen skrives

(c) I C skriv et lille program der for hvert interrupt med 1 sekunds interval i main() inde i while(1) der kan tænde og slukke lysdioden på P1.0 styret af en variable sat til 1 i service rutinen - (se gennemgang i lektion 4). Kompiler og upload se at det virker.

(d) Benyt flow chartet i 4.a og implementer uret under betingelsen (if) af variablen sat i service rutinen er sand så kører uret.

(e) Før vi kan få urets tid vist på OLED display så skal urets tre variable timer, minutter og sekunder lægges ind i et array (char time[9]) vha. sprintf(time,"02d:02d:02d", hh, mm,ss); hvor time er array der holder ascii repræsentationen tiden der kommer ind vha. hh, mm og ss- henholdsvis timer, minutter og sekunder.

(f) Efter sprintf skrives nu et kald ssd1306_printText(0,0,time) for at få urets tid skrevet ud på OLED display

(g) Programmer nu MSP430 og se om det virker - ellers ret programmet til så det virker.

Overstående forventes at være færdigt inden d. 10 okt.

4.2 Eksternt interrupt på P2.1

Nu ønskes det at brugeren kan sætte tidspunktet på uret vha et tryk på P2.1 på boardet.Når kontakten trykkes så skal der komme et interrupt fra kontakten sat til P2.1. I service routinen for det externe interrupt sættes en variable til 1 og i main() testes der på om den er blevet sat med et if selection statement. For hvert interrupt : Hvis sand så skal der læses et sekund tal eller et minuttal eller et timetal ind fra dip-switch (som i opgave 1), dvs. **så der skal trykkes tre gange på P2.1**. For hver gang der trykkes skal variablene henholdsvis hh, mm og ss læses ind og de indlæste værdier skal overskrive dem der kører i uret. Dipswitch værdien for ss, min hh kan sættes op vha. lommeregneren i windows hvor du taster decimal tal ind og aflæser det binære tal. Når alle værdier er opdateret så skal uret køre med den nye tid skrevet ud på displayet.
For at styre denne indlæsning og give hjælp til brugeren så lav en tælle variable som tælles en op for hvert interrupt. Benyt denne tællevariable i en switch case og case 1: læs timeværdien ind -skriv ud på display til kontrol, case 2: læs min ind skriv den ud, case 3 læs sec ind og skriv ud. Derefter lad uret gå igen styret af timer interrupted.

(a) I lektion 5 og forberedelse introduceres eksternt interrupt -dvs. et interrupt, der aktiveres når man trykker på en af kontakterne P1.1 eller P2.1

5 Aflevering

(a) omdøb main.c til main.txt og læg den op og skriv en journal, med forside med navne billeder og gruppenummer, der generelt beskriver hvordan den timer i har anvendt er sat op, et flow chart for uret og et flowchart for hvordan opdatering af uret foregår med beskrivelse hvor vi kan se betingelsen (vist med en diamond) for at opdatere og et test afnsit med resultater inkl. forklaring . Det lægges op under opgave 2 senest tirsdag d 22 oktober. i

God arbejdslyst

2024-10-02 Ole Schultz