Laboration 6: Motorstyrning

1 Inledning

Laborationen ska bidra till en grundläggande förståelse för:

- Hur man hanterar olika ”moder” i ett program, genom att konstruera en tillståndsmaskin.
- Hur man använder PWM (Pulse Width Modulation) för motorstyrning.
- Hur man använder ADC-enheten med avbrott för att avläsa en potentiometer.
- Hur en applikation är uppbyggd av flera abstraktionsnivåer av programkod.
- Hur man utvecklar och testar ett program i små steg.

1.1 Utrustning

- Fläkt (samma som använts i tidiga elektroniklabbar)
- trimpotentiometer 10 kΩ (samma som använts i IS-Lab 5)

2 Beskrivning av laboration

Den här laborationen utgör en fortsättning på programmering av inbyggda system i C. Laborationen återanvänder kod från föregående laboration, men det tillkommer även en del nya saker: PWM och en lite mer avancerad tillståndsmaskin. Laborationen består av följande moment:

1. Skapa nytt C-projekt.
2. Huvudprogram
3. Inkoppling av potentiometer.
4. Drivrutin för motorstyrning med PWM.
5. Test och färdigställning av system.
6. Fördjupande moment: Prova olika inställningar för PWM/Prova tachometer-signalen.

2.1 Syfte och mål

I denna laboration skall ni fortsätta att arbeta med mindre kodmoduler med tydligt definierade och avgränsade ”ansvarsuppgifter”. Ni kommer att utveckla huvudprogrammet själva. Ni skall återanvända kopplingen med ADC från förra labben, och använda denna för att läsa av spänningen från en potentiometer (som skall styra varvtalet på motorn, via PWM)

A/D-omvandlingen ska nu ske utan differentiell förstärkning (1x). Referensspänningen är 5V. På samma sätt som i föregående labb, ska ni använda en avbrottsrutin.

Huvudprogrammet ska konstrueras som en tillståndsmaskin (se Figur 1).

Denna laboration är mindre styrd är tidigare laborationer. Ni har därför ett större ansvar att skapa en god struktur och ett väldokumenterat program, samt att hitta namn på register, etc. i datablad/referensmanual.

Tips! För att säkerställa att vissa saker verkligen fungerar (exempelvis avbrottsrutinen) kan ni alltid använda pinne 13 (PC7), som ni kan sätta hög/låg alternativt toggla. Pinnen mäter ni med oscilloskop!

2.2 Examination

2.2.1 Inlämning av rapport och programkod

Instruktioner för inlämning finns beskrivet på inlämningssidan för denna laboration (på Canvas), samt i dokumentet **Allmänna instruktioner för laborationer.**

2.2.2 Praktisk och muntlig redovisning

Den praktiska delen av laborationen examineras efter att laborationens sista moment har avslutats. Följande ska redovisas:

- Uppkopplad krets. Kretsen ska vara snyggt kopplad. Dessutom ska kablarna ha korrekt och enhetlig färgkodning.
- Programkod (med god struktur och kommentarer).
- Demonstration av systemet, fullt fungerande enligt specifikation.

Ni ska även kunna redogöra för funktionalitet avseende krets och programkod.

OBS! Precis som i tidigare laborationer ska ni alltid redovisa ert bidrag i de filer som ni redigerar eller skapar. Från och med nu förutsätts att ni gör det självmant utan uppmaningar i laborationsmanualen.

3 Moment 1: Skapa nytt C-projekt

3.1 Beskrivning

Innan ni kan börja programmera måste ni göra vissa förberedelser. Till att börja med måste ni skapa ett nytt C-projekt i Atmel Studio (instruktioner finns på Canvas). Därefter ska ni importera kod som ni använde i laboration 5.

3.2 Uppgifter

Uppgift 3.2.1

Skapa ett nytt C-projekt med namnet ”lab6”. I detta projekt ska ni återanvända kod från den förra laborationen. Ni ska importera lämpliga filer och se till att projektet går att bygga.

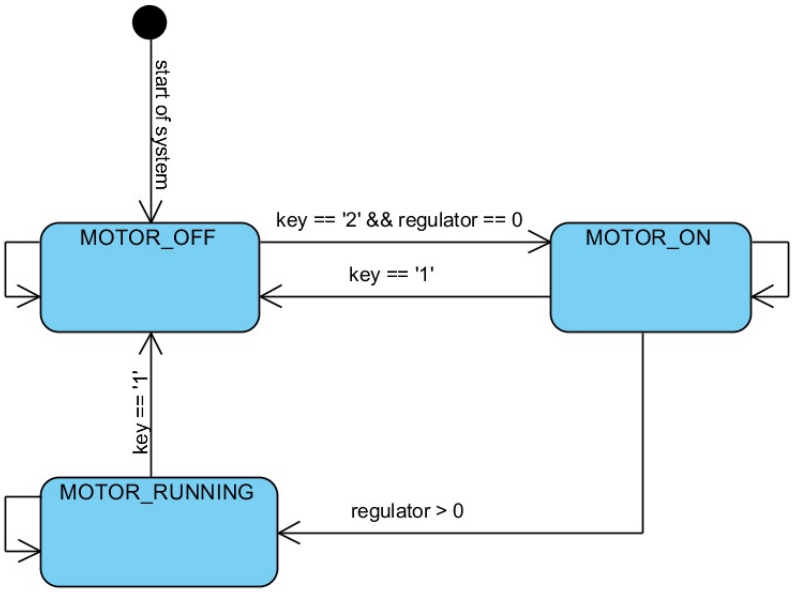
Uppgift 3.2.2

Ni bör redan i föregående uppgift ha importerat temp.h och temp.c. Byt namn på dessa till ”regulator.h” och ”regulator.c”

4 Moment2: Huvudprogram

4.1 Beskrivning

Huvudprogrammet ska konstrueras som en tillståndsmaskin, se nedan:



Figur 1: Tillståndsdiagram för huvudprogrammet

Samtliga uppgifter i detta moment ska utföras i **lab6.c**.

4.2 Uppgifter

Uppgift 4.2.1

Uppdatera tillståndsmaskinen (från lab 5) till att följa Figur 1 och se till att lämpliga textsträngar skrivs ut (använd tex namnen på tillstånden...) Förutom textsträngarna, skall även motorstyrningens värde skrivas ut (dvs 0-100%) - i lämpligt tillstånd.

Notera begränsningarna i tillståndsövergångarna i denna tillståndsmaskin jämfört med tillståndsmaskinen i lab 5! Koden som implementerar tillståndsmaskinen måste vara lite mer avancerad för att klara detta på ett korrekt sätt.

5 Moment 3: Inkoppling av potentiometer

5.1 Beskrivning

Nu skall ni koppla in en potentiometer på liknande sätt som ni gjorde i förra labben (då för att testa funktionen). Ni skall dock använda lite andra parametrar för ADC, så inkopplingen blir inte riktigt identisk.

5.2 Inkoppling av potentiometern

Uppgift 5.2.1

Koppla upp potentiometern till Analog Input 4 (ADC1). Kontrollera att potentiometern lämnar maximalt 0-5V!

5.3 Konfigurering av ADC

Uppgift 5.3.1

Nu ska ni konfigurera ADC-enheten och avbrottsrutinen i funktionen regulator_init(). Till skillnad från förra labben så skall ni nu endast använda ADC1 (”Single Ended Input for ADC1”). Ni skall också bara använda 8 bitar, vilket innebär att värdet från ADC skall hanteras annorlunda.

Uppgift 5.3.2

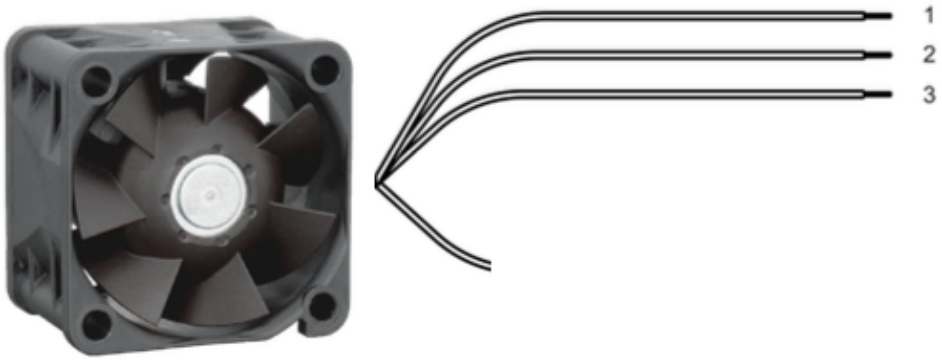
Se till att den rutin som läser ut resultatet från avläsningen av potentiometerns värde returnerar ett tal mellan 0-100 (dvs procent).

Testa! **(Blir det fel? Kolla implementationen av macrot: SET_BIT_LEVEL, som ni gjorde i lab4!)**

6 Moment 4: Drivrutin för motorstyrning med PWM

6.1 Beskrivning

Nu skall själva motorn kopplas in och motorstyrning konfigureras. Motorn skall kopplas in enligt följande (**Vänta med att koppla in fläkten till ni kommer till 6.3.1. Fläkten har ett internt pull-up-motstånd, så det går för fullt då den får spänning på röd/svart)**



Figur 2: Inkoppling av motor

1. Tråd 1 – Röd – kopplas till +5V
2. Tråd 2 – Svart – kopplas till Jord
3. Tråd 3 – Lila – kopplas till PC6
4. Tråd 4 – Vit – (Tachometer) används för extrauppgift

Fläkten bör ni ha i komponentlådan redan.

6.2 Konfiguration och kontrollmätning

Uppgift 6.2.1

Skapa en ny fil ”motor.h” och ”motor.c”. Lägg till en funktion (tex) ”motor_init”. Konfigurera PWM för Timer 3, på det sätt som beskrivits i föreläsning:

```
// set PC6 (digital pin 5) as output

....

// Set OC3A (PC6) to be cleared on Compare Match (Channel A) TCCR3A |= ....

// Waveform Generation Mode 5, Fast PWM (8-bit)

TCCR3A |= ....

TCCR3B |= ....

// Timer Clock, 16/64 MHz = 1/4 MHz

TCCR3B |= ....
```

Lägg också till en funktion för att sätta motor-hastigheten, tex ”motor_set_speed(speed)”

Uppgift 6.2.2

Uppdatera huvudprogrammet, så att värdet som läses från potentiometern används för att styra motorns hastighet (använd funktionen som skapades i 6.2.1). Se till att värdena som skickas till motorstyrningen också visas på displayen.

Uppgift 6.2.3

Koppla in ett oscilloskop på PC6 och studera hur signalen ser ut. Testa ett antal värden på styrsignalen (mellan 0 och 100%). Mät och kontroller att (hyfsat) korrekt ”duty cycle” erhålls.

Uppgift 6.2.4 (redovisas i Rapport)

Fotografera oscilloskopsbilden för ett par olika värden på styrsignalen. Bilderna ska visa oscilloskopsbilden med en vettig upplösning och stabil bild (tänk på triggningen).

Uppgift 6.2.5 (redovisas i Rapport)

Vad är frekvensen på PWM-signalen? Stämmer detta med inställningarna?

6.3 Inkoppling av motor

Uppgift 6.3.1

Anslut fläkten till plus och minus 5V, samt styrsignalen till PC6 (Digital pin 5 på kortet) och testa om motorn nu fungerar som tänkt.

7 Moment 5_ Test och färdigställning av system

7.1 Beskrivning

Vid det här laget bör allting fungera som den ska. Men för säkerhets skull ska ni göra slutgiltiga tester. Ni bör även snygga till koden, fixa kommentarer, etc.

7.2 Funktionstest

Uppgift 7.2.1 (redovisas i Rapport)

Nu ska ni testa så att allt fungerar enligt specifikation. Skriv ett enkelt testprotokoll. Ni bestämmer vad som ska testas. Utför och dokumentera sedan testerna enligt ert testprotokoll.

7.3 Färdigställning av program

Uppgift 7.3.1

Snygga till koden – Se till att kommentarerna är vettiga och ange ert bidrag i filerna. Uppdatera datum.

7.4 Slutlig funktionstest

Uppgift 7.4.1

För att verkligen säkerställa att programmet fungerar efter er att ni modifierat koden en sista gång, testa systemet ytterligare en gång (enligt ert testprotokoll)!

7.5 Reflektion

Uppgift 7.5.1 (redovisas i Rapport)

Redogör för era erfarenheter och kunskaper från denna laboration (minst en halv A4-sida – t**ips: låt bägge skriva individuella bidrag):**

- Vad har ni lärt er?
- Om ni får välja en sak, upplevde ni något som var intressant/givande?
- Fanns det något som upplevdes som svårt?
- Gick allting bra eller stötte ni på problem? Om allting gick bra, vad var i så fall anledningen detta? Om ni stötte på problem, hur löste ni i så fall dem?

Fördjupande moment

8 Moment 6: Prova olika inställningar för PWM

I föregående moment har ni använt givna inställningar för PWM. Laborera lite med andra inställningar för att lära er mer om hur PWM fungerar i systemet.

8.1 Implementering

Prova olika inställningar på prescalern, tex /8 och /1. Prova också med att använda 10 bitars PWM och 16 bitar's PWM (tänk på att både OCR3AH och OCR3AL måste skrivas då). Vad händer tex med frekvensen på PWM-signalen då man byter?

9 Moment 7: Använd tachometer-signalen

Tanken är att använda tachometer-signalen på något sätt. Först för att skriva ut varvtalet och sedan för att styra fläkten. Fläktens datablad ligger i laborationsmappen på Canvas.

9.1 Implementering

Fläkten har 4 ingångar: Plus, Minus, PWM-ingång och tachometerutgång (se datablad).

- Tachometer-utgången ger 2 pulser per varv, så en extrauppgift kan vara att läsa varvtalet (antingen direkt i huvudprogrammet eller via avbrott) och visa varvtalet efter procenttalet på motorstyrningen (på displayen)

9.2 Läs av och visa varvtalet

Uppgift 9.2.1

Koppla in tachometer-signalen till oscilloskopet och se hur signalen ser ut. Konfigurera sedan en timer (läs på i databladet) så att ni kan mäta tiden som en signal räkna fram frekvensen/fläktens hastighet. Skriv ut detta på skärmen.

9.3 Styr varvtalet

Uppgift 9.3.1

Använd tachometer-signalen för att styra PWM-signalen, så att ni kan få ett precist värde (i varv/minut) istället för procent.