Laboration 2: LCD

1 Inledning

Laborationen ska bidra till en grundläggande förståelse för:

- Beräkning och mätning av exekveringstider.
- Hur man kan skapa enkla fördröjningsfunktioner.
- Hur man kan använda Excel för att göra flertalet beräkningar med varierande parametrar, med syftet att analysera hur en subrutin förväntas prestera.
- Att arbeta med en LCD-display.
- Hur man kan använda makrofunktioner.
- Mjukvarubaserad hantering av knappstuds.

1.1 Utrustning

- kopplingsdäck (840 anslutningar)
- LCD (2 rader, 16 kolumner) - Ny i denna lab
- 1st vridpotentiometer 10 $k\Omega$

2 Beskrivning av laboration

Fördröjningsrutiner behövs för att ”pausa” exekveringen av ett program. Det finns olika användningsområden för att använda fördröjningar, men i denna laboration är det framför allt viktigt för att kommunicera med styrkretsen på LCD-displayen. Dessutom kommer ni att behöva fördröjningar för att hantera knappstuds.

Laborationen består av följande moment:

1. Programmering av fördröjningsrutiner
2. Inkoppling av LCD
3. Programmering av LCD
4. Presentation av tangentkod på LCD

2.1 Syfte och mål

Ni kommer att få komplettera färdigskrivna fördröjningsrutiner så att de fungerar enligt specifikation. Rutinerna för den mest grundläggande hanteringen av displayen är också färdigskrivna. Däremot är dessa rutiner beroende av fördröjningsrutinerna, vilket ställer krav på att dessa fungerar korrekt. Laborationen avslutas med att ni ska skriva ett program som använder allt ni hittills har jobbat med: tangentbord, LCD och fördröjningar.

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 enligt Figur 4-1. 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.

3 Moment 1: Programmering av fördröjningsrutiner

3.1 Beskrivning

Detta moment går ut på att ni ska färdigställa fördröjningsrutiner. Detta gör ni genom att beräkna exekveringstider. Dessutom ska ni verifiera att koden är korrekt genom att mäta exekveringstiderna med ett oscilloskop. Ni ska även reflektera över hur fördröjningsrutinerna fungerar.

I laborationens mapp (på Canvas) finns en fil med namnet **delay_routines.zip**, som innehåller ett Atmel Studio projekt. Projektet består av två filer: **delay_routines.asm** och **delay.inc**. Den första filen, **delay_routines.asm**, är ett testprogram för fördröjningsrutinerna. Den andra filen, **delay.inc**, innehåller fördröjningsrutinerna. Dessa subrutiner har placerats i egen fil för att ge en bättre programstruktur. Dessa fördröjningsrutiner ska ni sedan återanvända i kommande laborationer.

I filen **delay.inc** hittar ni följande subrutiner:

- **delay_1_micros:**
  Dess syfte är att generera en fördröjning på $1\ \mu s$, inklusive **RCALL** (anropet av subrutinen ifråga). Er uppgift är alltså att komplettera subrutinen med ett antal **NOP**-instruktioner så att detta blir möjligt.
- **delay_micros:**
  Den här subrutinen ska generera en fördröjning (i mikrosekunder) baserat på vad som anges i **R24** innan subrutinen anropas. Detta register kan betraktas som en parameter till subrutinen. Om man exempelvis anger att **R24** ska ha värdet 10, så är det meningen att det ska genereras en fördröjning på $10\ \mu s$. Er uppgift är att komplettera subrutinen med ett antal **NOP**-instruktioner, så subrutinen fungerar enligt denna specifikation.
- **delay_ms:**
  Denna subrutin påminner om **delay_micros**, dock anges parametervärdet i millisekunder. Subrutinen är redan färdigskriven, men den är beroende av hur **delay_micros** fungerar.

Till er hjälp finns även en Excel-fil i laborationens mapp på It's Learning, med filnamnet **Fördröjningsrutiner – exekveringstider.xlsm**. För att göra vissa reflektioner är det viktigt att ni även gör vissa grundläggande studier i databladet för LCD-displayens styrkrets.

3.2 Färdigställning av fördröjningsfunktioner

Uppgift 3.2.1

Ladda ner och packa upp **delay_routines.zip**. Öppna projektet och se till att ni förstår programmets struktur och syfte. I programmets ”main-slinga” finns assembly-instruktioner för att anropa samtliga fördröjningsrutiner. Endast aktuell subrutin, dvs. den som ni utvecklar/testar, ska vara avkommenterad. Resten av fördröjningsrutinerna ska vara kommenterade.

Uppgift 3.2.2 (redovisas i programkod)

Komplettera **delay_1_micros** med ett antal **NOP**-instruktioner.

Uppgift 3.2.3 (redovisas i rapport)

Testa **delay_1_micros** genom att mäta exekveringstiden med oscilloskopet. Fotografera bilden.

OBS! Tänk på att instruktionerna för att sätta D13 hög respektive låg, samt att repetera detta, kräver viss tid! Det kan påverka era mätresultat vid mindre fördröjningar!

Uppgift 3.2.4 (redovisas i programkod)
Komplettera **delay_micros** med ett antal **NOP**-instruktioner. Man kan inte generera exakta fördröjningar, däremot kan noggrannheten bli mer eller mindre god. Eftersom fördröjningsrutinen är av det enklare slaget, så är det intressant att reflektera över hur noggrann subrutinen är.
OBS! Använd Excel-filen som stöd!

Uppgift 3.2.5 (redovisas i rapport)
Besvara följande frågor:

- Hur många **NOP**-instruktioner väljer ni att komplettera subrutinen med? Motivera ert val!
- Finns det något intervall för parametervärdet hos **delay_micros**, där avvikelserna är större i förhållande till parametervärdet?
- Kommer avvikelserna att påverka LCD-kommunikationen nämnvärt? Kan avvikelserna försummas?
- Hur påverkas **delay_ms** av beroendet av **delay_micros** – är avvikelserna ett stort problem?

Tips! Försök att svara på frågorna så gott ni kan, men det kan vara en god idé att återkomma till den här uppgiften när ni har avslutat Moment 3!

Uppgift 3.2.6 (redovisas i rapport)
Kontrollera att **delay_micros** fungerar. Mät och fotografera oscilloskopsbilden.

Uppgift 3.2.7 (redovisas i programkod)
Finns det någon begränsning i hur stort respektive litet värde som kan anges som parametervärde till **delay_micros** och **delay_ms**? Finns det något värde som inte är bra att använda? Ange dessa begränsningar i kommentarerna, förslagsvis i kommentarsblocket ovanför respektive subrutin.

Uppgift 3.2.8 (redovisas i programkod)
Kommentera koden i **delay.inc**: komplettera befintliga kommentarer, ange datum och författare, etc.
*OBS! Notera hur **delay.inc** har inkluderats i huvudprogrammet! Ni ska själva inkludera den här filen i kommande projekt!*
Därefter kan ni spara och stänga projektet.

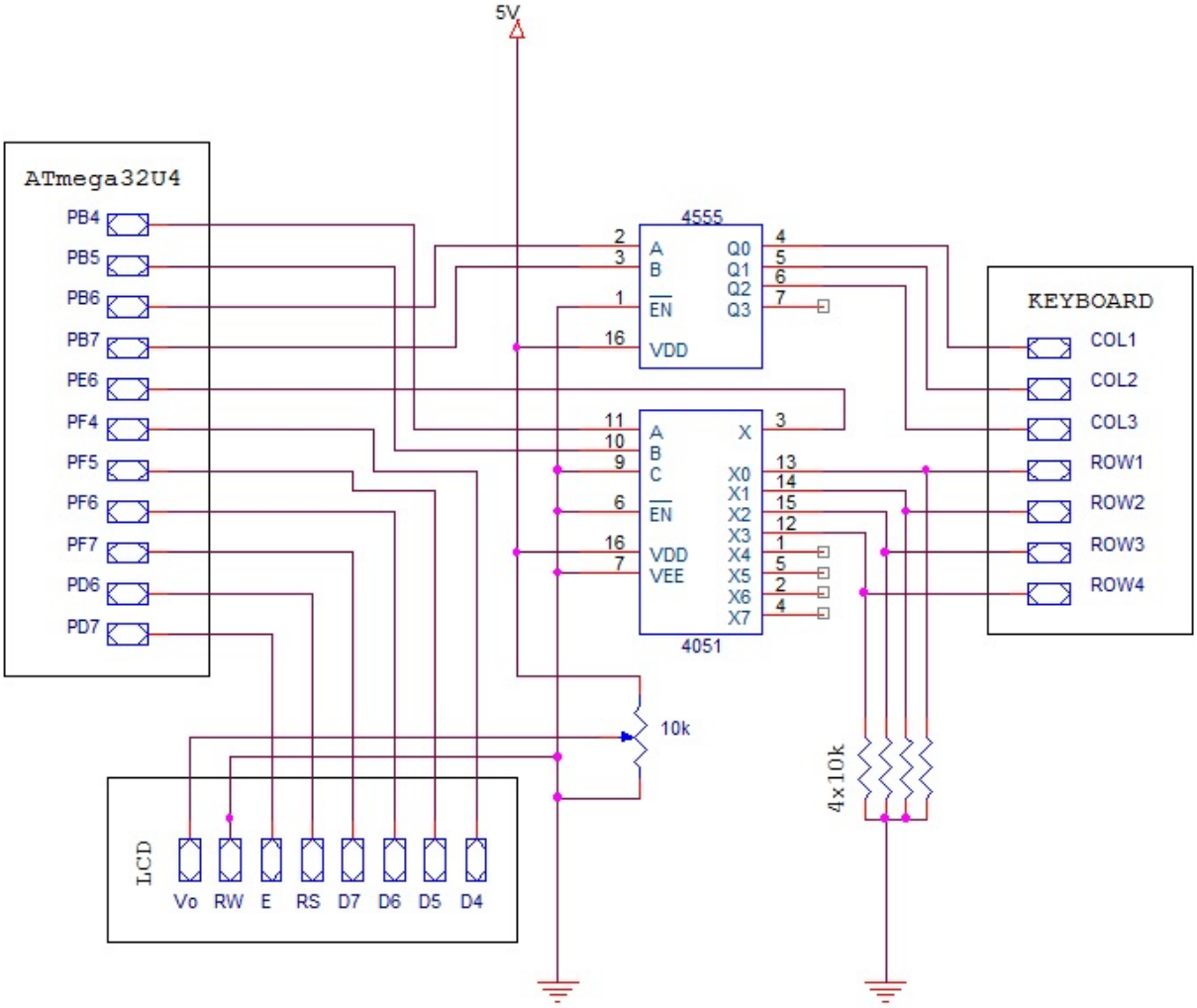
4 Moment 2: Inkoppling av LCD

4.1 Uppkoppling av krets

Komplettera den befintliga kretsen genom att koppla in potentiometern och LCD-displayen, enligt Figur 4-1. Potentiometern används till att ställa in hur tydligt tecken ska visas på displayen (kontrast). För att koppla in LCD-displayen behöver ni läsa databladet för denna. Det finns dock två datablad som är relaterade till displayen. Det ena databladet är kortfattat och beskriver bland annat displayens anslutningar. Det andra databladet beskriver hur styrkretsen fungerar, men det ska ni inte använda nu.

Uppgift 4.1.1

Koppla upp enligt schemat nedan. Tänk även på spänningsmatningen till LCD'n (kolla datablad).



Figur 4-1: Schema för utökad krets, med LCD och potentiometer

4.2 Kommunikation med display

I denna tillämpning kommunicerar mikrokontrollern med displayen genom att skicka instruktioner och tecken (som skrivs ut på displayen). Instruktioner och tecken har en bredd på 8 bitar. Eftersom mikrokontrollern har ett fåtal anslutningar så nöjer vi oss med 4 anslutningar (**D4-D7**) för överföring av data (instruktioner eller tecken) till displayen. För att skicka 8 bitar måste man således skicka 4 bitar (en nibble) i taget. Man börjar med de mest signifikanta bitarna.

Vi ska endast skriva till displayen. Därför kopplas **RW**-pinnen (Read/Write) till jord. **RS** (Register Selector) används för att specificera vilket register på styrkretsen man ska kommunicera med. En logisk nolla innebär att man kommunicerar med instruktionsregistret, en logisk etta innebär att man kommunicerar med dataregistret. En vanlig instruktion är exempelvis att rensa displayen.

E (Enable) är mycket viktig i samband med kommunikationen. När man håller dess nivå hög under en viss stund och sedan sänker den, signalerar man att en instruktion eller ett tecken skickas.

Eftersom det tar en viss tid för styrkretsen att utföra vissa operationer så måste man vänta en kort stund innan man skickar något nytt igen. För att säkerställa att kommunikationen ska fungera måste

man ha fördröjningar.
Vo används för att ställa in kontrasten på displayen, vilket gör att ni själva bestämmer hur tydligt tecken ska synas.

5 Moment 3: Programmering av LCD

5.1 Beskrivning

I detta moment ska ni använda färdigskrivna rutiner för displayen. För er egen skull, samt för att kunna lösa vissa uppgifter, behöver ni sätta er in i hur mikrokontrollern kommunicerar med displayen.

5.2 Konfigurering av projekt

Uppgift 5.2.1
Skapa ett nytt assembler-projekt i Atmel Studio (processor: ATmega 32u4) . Kalla det för **lab2**.

Uppgift 5.2.2 (redovisas i programkod)
Ersätt det som finns i **lab2.asm** med koden ni skrev i Laboration 1.

Uppgift 5.2.3 (redovisas i programkod)
Inkludera filen **delay.inc** i ert projekt. Kom även ihåg att anpassa huvudprogrammet för detta! (högerklicka på projektet i Solution Explorer, välj Add/Add existing item)

Uppgift 5.2.4 (redovisas i programkod)
Vissa av anslutningarna är redan konfigurerade. Däremot saknas konfigurering av PD6 och PD7. Konfigurera dem som utgångar. Ändra även **Port B till utgång** och sätt PE6 som ingång.

Uppgift 5.2.5 (redovisas i programkod)
Ladda ner **lcd.inc** från laborationens mapp på It's Learning. Inkludera sedan den i projektet. Glöm inte att anpassa huvudprogrammet för detta!

Uppgift 5.2.6 (redovisas i programkod)
Ni måste se till att displayen initialiseras. Ändra subrutinen **init** (i filen **lab2.asm**) så att **lcd_init** anropas efter **init_pins**.

Uppgift 5.2.7
Ladda ner **main_lcd.txt** från It's Learning. Ersätt koden som finns i "main-slingan" med koden som finns i den här filen. Assemblera och för över programmet. Om ni har följt instruktionerna så borde det stå "HELLO!" på displayen. *Tips! Om ni inte ser någon text, prova att justera kontrasten!*

Uppgift 5.2.8 (redovisas i rapport)
Fotografera displayen, som visar texten "HELLO!".

5.3 Anpassning och utökning av LCD-rutinerna

Uppgift 5.3.1 (redovisas i programkod)
Som ni ser blir det ganska jobbigt att skriva kod för att få några tecken på displayen. Detta skulle kunna underlättas med en makrofunktion. Skriv en makrofunktion, med namnet **LCD_WRITE_CHAR**, i

lcd.inc. För att skriva ut ”HELLO!” med hjälp av makrofunktionen ska det nu skrivas i ”main-slingan” enligt:

```
LCD_WRITE_CHAR 'H'
LCD_WRITE_CHAR 'E'
LCD_WRITE_CHAR 'L'
LCD_WRITE_CHAR 'L'
LCD_WRITE_CHAR 'O'
LCD_WRITE_CHAR '!'
```

Uppgift 5.3.2 (redovisas i rapport)
I displayens initialiseringsrutin (**lcd_init**) förekommer en instruktion som gör så att man får en blinkande markör. Hur ska instruktionen se ut om man vill ha en markör som inte blinkar?

Uppgift 5.3.3 (redovisas i rapport)
Hur ska instruktionen se ut om man inte vill ha en markör överhuvudtaget?

Uppgift 5.3.4 (redovisas i programkod)
Ändra i **lcd_init** så att man inte ser någon markör.

Uppgift 5.3.5
Nu ska ni se till att ”HELLO WORLD” skrivs på displayen. ”HELLO” ska dock skrivas vid den fjärde kolumnen (kolumn 3) på den första raden (rad 0). ”WORLD” ska skrivas på den andra raden (rad 1), vid den sjunde kolumnen (kolumn 6).
Tips! Man specificerar var text ska hamna genom att ange DDRAM-adress!
Läsanvisningar: sidorna 23 och 26 i databladet för LCD-displayens styrkrets.

Uppgift 5.3.6 (redovisas i rapport)
Fotografera displayen, som visar texten ”HELLO WORLD” enligt specifikationen ovan.

6 Moment 4: Presentation av tangentkod på LCD

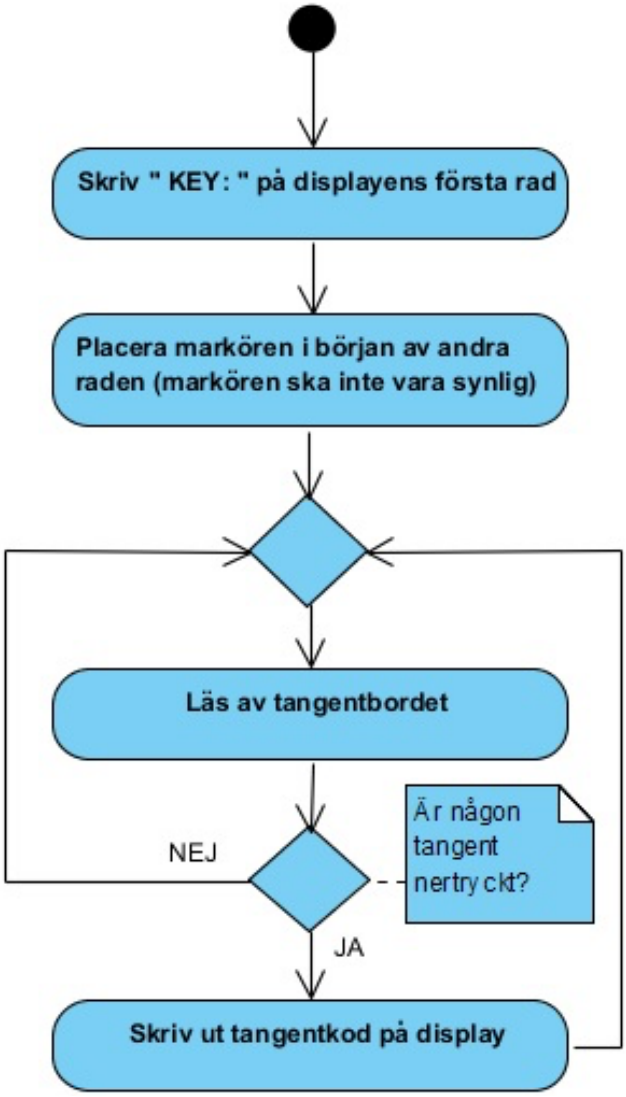
6.1 Beskrivning

Labben avslutas med att ni ska skapa ett program som använder subrutinerna för fördröjningar och hantering av tangentbord och display. Programmets funktion är beskrivet med aktivitetsdiagrammet i Figur 6-1.

Subrutinen **read_keyboard** returnerar en unik kod för varje tangent, enligt de uppgifter ni fick i Laboration 1. Denna kod ska skrivas ut på displayens andra rad (efter att ha **konverterats till ASCII!**). Koderna ska skrivas ut efter varandra, dvs. första gången visas koden på den första positionen, andra gången på den andra positionen, etc. Varje tangentkod ska representeras av endast ett tecken. Eftersom tangentkodernas värden varierar mellan 0-11 (decimalt), blir ni tvungna representera koderna i hexadecimal form på displayen (0-9 och A-B).
En tangenttryckning ska endast generera ett tecken. Det ska inte skrivas ut mer än ett tecken så länge en tangent är nertryckt. Ni skall INTE flytta tillbaka positionen till 0 för att lösa detta! Man skall

kunna skriva ut ett antal tecken och alla skall visas samtidigt.
Fördröjningsrutinerna kommer att behövas för att hantera knappstuds.

OBS! Aktivitetsdiagrammet är ingen detaljerad beskrivning av programmet!



Figur 6-1: Programmets funktion beskrivet med aktivitetsdiagram

6.2 Programmering

Uppgift 6.2.1 (redovisas i programkod)
Börja med att programmera enligt specifikationen ovan. Ni blir tvungna att bryta ner huvudproblemet i så små delproblem som möjligt.

Uppgift 6.2.2 (redovisas i rapport)
Förutsatt att ni har löst föregående uppgift, anser ni att aktivitetsdiagrammet behöver kompletteras, för att programmet skall uppfylla **kraven som testas i 6.2.3**? Bifoga i så fall det uppdaterade aktivitetsdiagrammet i rapporten. Om ni anser att diagrammet inte behöver en komplettering, motivera i så fall er ståndpunkt.

Uppgift 6.2.3

Testa! Inget skall alltså skrivas ut om man inte trycker någon tangent. **En** tangenttryckning ska endast generera **ett** tecken. Det ska inte skrivas ut mer än ett tecken så länge en tangent är nertryckt.

- Prova att trycka på samma knapp 2 gånger – skall resultera i 2 tecken på displayen.
- Prova även att trycka flera knappar i följd utan att släppa emellan, tex ’1’-’6’-släpp 1-släpp 6, skall resultera i ”16” på skärmen.

Tips: Det kanske inte är en bra ide att först testa på NO_KEY och avbryta om detta hittas….

Uppgift 6.2.3 (redovisas i programkod)

Innan ni avslutar laborationen ska ni:

- Se till att ni förstår vad programmet gör - gäller all kod!
- Snygga till koden och kommentera programmet där det saknas kommentarer. Ni ska kunna återvända till koden om några månader och fortfarande förstå vad som sker!
- Se till att samtliga filer som är redigerade av er ska innehålla era namn och aktuellt datum.
- Beskriva i kommentarerna, i början av varje fil (med undantag för **lcd.inc**), filens funktion och syfte.

När detta är klart kan ni redovisa för labhandledare!

Uppgift 6.2.4 (redovisas i rapport)

Redogör för era erfarenheter från denna laboration. Vad har ni lärt er? 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?