Computer- og El-teknik A.

Holstebro Tekniske Gymnasium - HTX

Projekt Træningsmaskine

Afleveret: Fredag d. 10/10-2008.

Udarbejdet af: Bent Arnoldsen, Holstebro HTX.
Gruppemedlem: Hjalmar Krarup Andersen, Holstebro HTX

Forord

Denne rapport er skrevet som en dokumentation til et elevprojekt gennemført på Holstebro HTX i perioden 8/9 til 10/10 2008. Den er skrevet af læreren som et eksempel på hvordan dokumentationen for et projekt kunne være lavet.

Indholdsfortegenelse

Indledning

Det er blevet meget populært at træne i forskellige fitness-centre.
Træningen kan have forskellige formål:

- optræning efter skader
- alment fysisk velvære
- pumpning af muskler
- træning på specifikke muskelgrupper til eliteidræt

Uanset formålet kan det være praktisk at kunne registrere det antal gentagelser man laver, og evt. den tid det foregår over.

Problemformulering

Der skal fremstilles et kredsløb, der opfylder følgende krav:

- Det skal være muligt at registrere træningen.
- Det må ikke være mulige at snyde apparatet.
- Man skal kunne nulstille tællingen.
- Der skal være mulighed for at registrere tiden der er trænet.
- Man skal kunne starte og stoppe tiden.
- Mulighed for at man kan kommunikere resultaterne op til en PC.

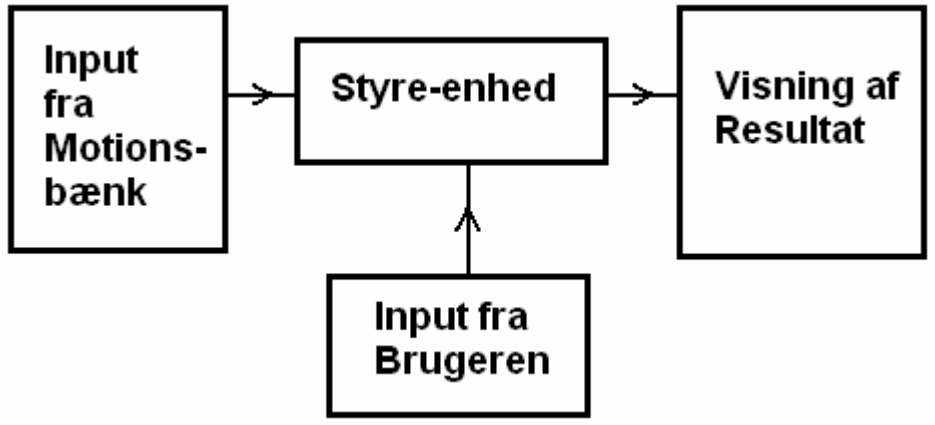
Da projektet her er et udviklingsprojekt, hvor der også skal indlæres en del elementer omkring både elektronik og programmering, så vil jeg dokumentere alle de del-løsninger jeg kommer igennem under vejs.
De fleste af del-løsningerne vil kunne komme til at indgå som elementer i den endelige løsning, men der vil også være mange dele, der bliver sorteret væk igen undervejs.
De forskellige dele kunne fungere som en selvstændig løsning, og det giver derfor god mening at dokumenter de principper der aktuelt indgår i de enkelte del-løsninger.

Overblik over elektronikken

Når vi skal arbejde med elektronik, så er det godt at danne sig et overblik over hvilke dele skal projektet indeholde.
Til denne brug er et blokdiagram en god ting at anvende.

Blokdiagrammet skal ikke illustrere præcist hvordan systemet skal fungere, men skal mere give en ide om hvilke dele systemet overordnet skal bestå af.
Blokdiagrammet tegnes ud fra de krav der stilles til produktet.

Blokdiagrammet for dette projekt kunne det se ud som følger:



Blokdiagram over det kredsløb der skal konstrueres.

Den centrale styre-enhed er det PIC-board vi bruger i undervisningen[1], hvor der sidder en PIC 16F84. Den kan vi programmere til forskellige funktioner.
Denne centrale styre-enhed vil ikke blive yderligere dokumenteret i denne rapport, da jeg betragter den som en færdig blok, der kan anvendes ud fra den dokumentation der ligger tilgængelig på nettet.

Inputtet fra træningsbænken er det vi skal registrere på. For at vi kan registrere om brugeren laver træningen korrekt forestiller jeg mig, at der kan registreres en hvileposition for motionsbænken, og en yderposition, hvor en vægt er trukket helt op, eller noget andet, alt efter hvordan motionsbænken er opbygget.

Visningen af resultatet er en visning af hvor mange repetationer der er foretaget, og evt. hvor lang tid det er foretaget over.

Inputtet fra Brugeren er de betjeningsmuligheder brugeren skal have. Det skal være sådan at brugeren kan få vist de ting der registreres og hvad der ellers er af muligheder.

Arbejdsmetode i forløbet, som det afspejles i rapporten

Udviklingen foretages løbende, og der dokumenteres de enkelte dele for sig, sammen med hver deres software.
Dette er ikke en dokumentation på et færdigt produkt, men en dokumentation over hvordan udviklingen er skredet frem, hvad der er testet undervejs, og hvilke resultater jeg kom frem til.

Input fra Motionsbænken

Der er to kontakter på motionsbænken, en i hvile-positionen og en i yderpositionen.

Jeg vil i de følgende afsnit gennemgå hvordan jeg har fået kontakternes funktion ind i styre-enheden, og hvordan jeg har testet at de kommer der ind, og at brugeren ikke kan snyde med optællingen af træningen.

[1] PIC 16F84 Board fra http://www.holstebrohtx.dk/bar/index.php?pkt=1120 set 5/10-2008

De elektriske signaler fra mikroswitchene

Det signal vi ønsker at få ind i PIC'en skal være højt (+5V) når kontakten er aktiveret og lavt når kontakten ikke er aktiveret.

Ved at koble kontakten S1 op mod +5V får vi et højt signal ind til PIC'en når kontakten er aktiveret. For at signalet ikke bare skal svæve når kontakten er afbrudt, så skal vi bruge en pull-down-modstand R1 der er forbundet mellem kontakten og stel, som vist på diagrammet.

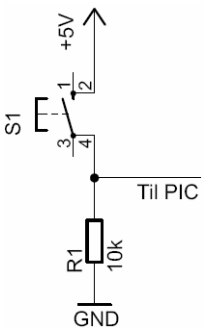


Diagram der illustrerer en enkelt kontakt.

Det er ikke umiddelbart let at regne på en pull-down modstand. Den skal dels sørge for at lækstrømmen fra PIC'en ikke giver et højt signal, men da denne strøm er maksimalt 1µA, så skal man over 1MΩ for at det betyder noget.
Den anden funktion pull-down modstanden har, er at sikre at støj ikke giver falske signaler. Det er også svært at regne på, men erfaringen siger at 10kΩ kan tage støj fra selv ret lange ledninger, og da disse kontakter sidder på motionsbænken kan der godt være ret lange ledninger – jeg vælger altså R1 og R2 til 10kΩ.

Diagrammet skal ændres, da kontakten ikke skal sidde på printet, men i stedet skal sidde på motionsbænken, fordi det er en mikroswitch som vist her:

Eksempel på en mikroswitch.[2]

I stedet for kontakter vælges skrueterminaler, så man kan skrue ledningerne i printet, og forbinde ud til mikroswitchen.

Da mikroswitchene er generelle for alle de forskellige løsningsmodeller der skal laves, så vælger jeg at lægge forbindelserne til dem på deres eget lille print, så det kan forbindes til PIC'en ved at sætte er ekstra stik på fladkablet.

Snitflade mellem mikroswitch-print og styre-enhed (PIC)

Til snitfladen anvendes stikket på PIC-boardet, der har forbindelse til Port A. Ud fra dokumentationen til PIC-boardet vælges de ben der ønskes anvendt til kontaktfunktionerne.

PIC-stik Ben	PIC-funktion	Beskrivelse
Ben 1	Port A0	Kontakt der er aktiv i hvileposition
Ben 2	Port A1	Kontakt der er aktiv i yderposition
Ben 9	+5V	Forsyning fra PIC-print
Ben 10	Stel	Stelforbindelse fra PIC-print

Tabel over snitfladen mikroswitch-print og styre-enhed.

[2] Billedet stammer fra www.cypax.com set 6/10-2008

Print til mikroswitche

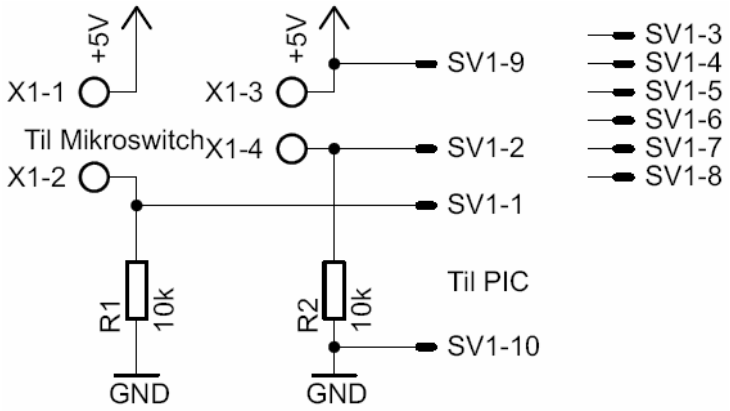
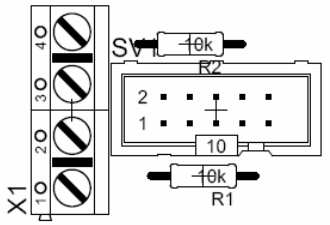


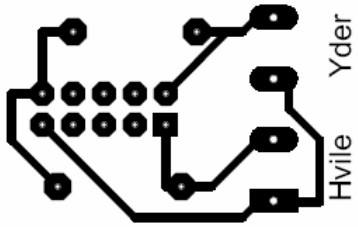
Diagram over printet til mikroswitchene

X1 er forbindelsen ud til de to mikroswitche
R1 og R2 er de to pull-down modstande
SV1 er Molex fladkabel stikket over til PIC-boardet.

Ud fra diagrammet fremstilles følgende layout i Eagle[3]



Komponentplacering



Print set fra kobber-siden

Funktionstest af mirko-switchene

For at se om mikro-switchene fra motionsbænken kommer ind i kommer korrekt ind i styre-enheden, så anvendes de to lysdioder der sidder på PIC-boardet.

Til dette skrives en lille test-kode.

Først defineres de to ben vi bruger til input fra de to kontakter og de to lysdioder jeg tester på. Der defineres også hvilke ben der skal være input og output.

```
-- Definer navne på port ben
var bit Hvile   is pin_a0
var bit Yder    is pin_a1
var bit LED1    is pin_b6
var bit LED2    is pin_b7

-- Definer Input / Output
pin_a0_direction = input
pin_a1_direction = input
pin_b6_direction = output
pin_b7_direction = output
```

[3] Diagramtegning og Layout fra www.cadsoft.de

Så laver jeg et meget simpelt program, der bare lægger niveauet af de to kontakter ud på lysdioderne. Det sker i et forever loop, så PIC'en fortsætter med det indtil der slukkes.

```
forever loop        -- fortsæt i det uendelige
  LED1 = Hvile
  LED2 = Yder
end loop
```

Koden til dette ligger som elektronisk bilag: **kontakt.jal**

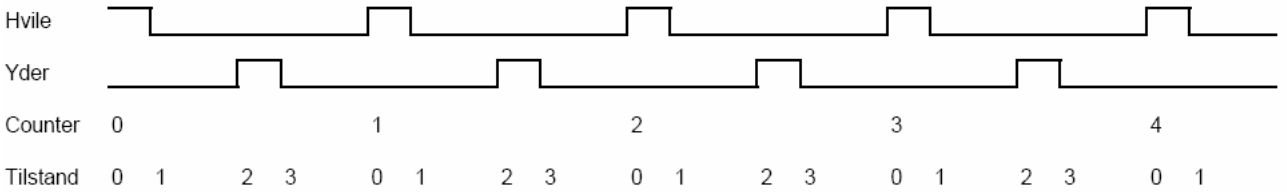
Gennemgang af tælle-sekvensen på motionsbænken

For at elektronikken kan registrere korrekt, så skal man sikre at begge kontakter aktiveres i den korrekte sekvens inden der tælles op.
De forskellige måder man kunne tænkes at ville snyde registreringen kunne være:
Gentagne aktiveringer af en af kontakterne, mens den anden kontakt er enten aktiv eller passiv.
Det vil give 4 forskellige måder at snyde på.
Måden jeg har valgt at forhindre snyd på er ved at registrere de tilstande træningsmaskinen går igennem, ved et normalt forløb af træningen, og så kun tillade at man går videre til den næste tilstand, hvis det er den tilstand man forventer:

Tilstand	Træningsmaskinen	Hvilekontakt	Yderkontakt
0	I hvileposition	1	0
1	På vej fra hvile- til yder-position	0	0
2	I yderposition	0	1
3	På vej fra yder- til hvile-position	0	0

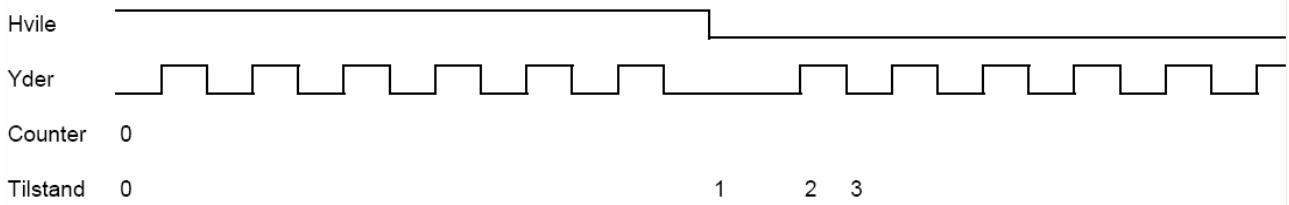
Tabel over de tilstande jeg forventer der kan komme fra motionsbænkens kontakter.

Dette kan illustreres ved hjælp af et tidsdiagram som vist her under. Counter er den tæller, der efterfølgende skal vises til brugeren. Tilstand er den interne variabel der bruges til at holde styr på at man ikke snyder registreringen.



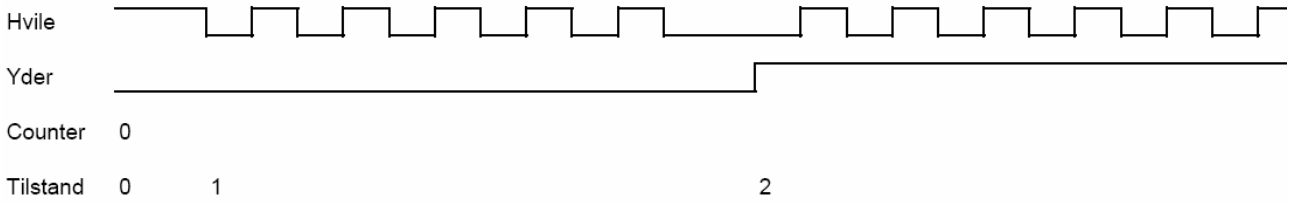
Tidsdiagram over et normalt forløb der tæller op.

For at kontrollere at der ikke kan tælles ved hjælp af snyd, så registreres de forskellige snydesituationer. Først hvor der vippes med yderkontakte, i begge stillinger af hvilekontakten.



Tidsdiagram hvor der forsøges snydt med yderkontakten.

På samme måde kontrolleres at der ikke kan tælles ved hjælp af snyd med hvilekontakten, og yderkontakten i de to forskellige stilinger.



Tidsdiagram hvor der forsøges snydt med hvilekontakten.

Jeg har testet de 4 forskellige måder at snyde på igennem, og ingen af dem tæller op på en utilsigtet måde.
Den test har jeg godt nok først lavet efter jeg fik visning med lysdioder på.
Der vil selvfølgelig forekomme tælling, hvis man vipper med begge, f.x. hvis man tager tidsdiagrammet hvor der snydes med yderkontakten, så vil der komme en tælling, hvis man sætter hvile høj. Det vil jeg ikke betragte som fejltælling.

Koden til tælling på motionsbænken.

Først defineres de to ben vi bruger til input fra de to kontakter (taget fra den første testsoftware)

```
-- Definer navne på port ben
var bit Hvile   is pin_a0
var bit Yder    is pin_a1

-- Definer Input / Output
pin_a0_direction = input
pin_a1_direction = input
```

Der defineres også de variabler vi bruger til at huske tilstanden og tællingen med:

```
-- Definer de variabler der bruges
var byte tilstand = 0
var byte count = 0
```

Begge variabler sættes til nul, når koden startes, så vi har styr på hvor vi er henne, når der kommer strøm på apparatet.

Efter opstart lander koden i et forever-loop.

```
forever loop        -- fortsæt i det uendelige
  .
  .
  -- Koden der skal afvikles hele tiden
end loop   -- end forever
```

I starten af projektet løber dette forever-loop bare så hurtigt rundt det kan, så tingene tjekkes mange gange i sekundet. Det betyder at det reagerer hurtigere en brugeren kan opfatte det.

Inde i forever-loopet, er det følgende konstruktion, der løser det der er skitseret i tidsdiagrammerne for tællingen.

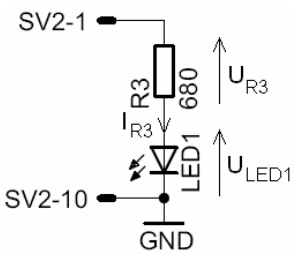
```
    -- Registrer hvilken tilstand maskinen er i
    if tilstand == 0 then    -- Den er i Hvile, vi venter på den trækkes ud
       if (! Yder) & (! Hvile) then
          tilstand = 1
       end if
    elsif tilstand == 1 then -- Den trækkes udad, vi venter på Yder
       if (Yder) & (! Hvile) then
          tilstand = 2
       end if
```

```
elsif tilstand == 2 then -- Den er i Yder, vi venter på den går tilbage
   if (! Yder) & (! Hvile) then
      tilstand = 3
   end if
elsif tilstand == 3 then -- Den er på vej tilbage, vi venter på Hvile
   if (! Yder) & (Hvile) then
      tilstand = 0
      count = count + 1  -- Det er i denne situation vi tæller
      if count == 100 then
         count = 99      -- Håndter at vi ikke går videre
      end if
   end if
else  -- Blot for en sikkerheds skyld, det skulle aldrig ske
   tilstand = 0
end if
```

Ideen i koden er, at man i hver tilstand ved hvilken betingelse der skal til for at bringe os til næste tilstand, så man f.x. fra tilstand 1 kun vil acceptere at Yder skal være aktiv og Hvile skal være passiv før vi accepterer at komme i tilstand 2.
Det er ved skiftet fra tilstand 3 til tilstand 0 at tælleren tælles en frem.
I denne version begrænses tællingen til 99, så den stopper der.
Der ligger ingen testkode til dette, da jeg ikke kunne teste før jeg fik visning på.

Visning af Resultat i binært format

For at få en simpel visning vælges der i første omgang at få tallet ud som binært på nogen lysdioder.
En enkelt lysdiode kan man få til at lyse fornuftigt ved ca. 5mA.
Jeg har valgt at de skal lyse når udgangen er høj, så jeg kobler lysdioderne til et Molex-stik SV2, der har forbindelse til port B.
Som vist her kommer der på SV2-1 et signal ud på ca. 4,85V når port B0 sættes høj. Der falder ca. 1,7 V over en rød lysdiode LED1, så resten af spændingen må ligge over R3, der kan udregnes som følger:



$$U_{R3} = U_{SV2-1} - U_{LED1} = 4,85V - 1,7V = 3,15V$$

$$R3 = \frac{U_{R3}}{I_{R3}} = \frac{3,15V}{5mA} = 630\Omega$$

R3 vælges til en standard-værdi på 680Ω hvilket giver en lidt mindre strøm, men det betyder ikke noget videre for lyset i lysdioden.

Opkobling af visning med lysdioder

De resterende lysdioder op til LED7 kobles op som vist:

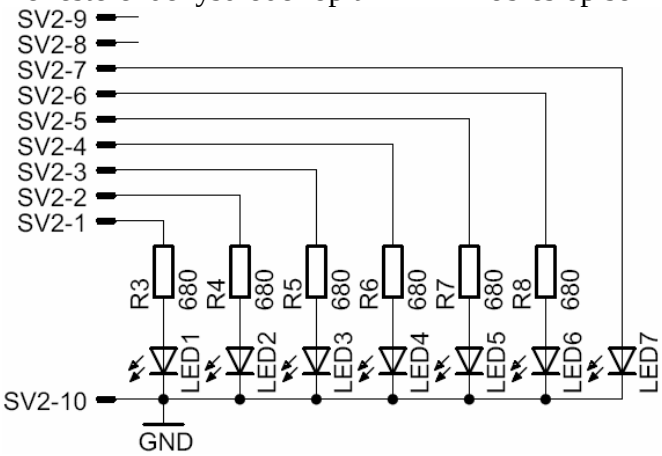


Diagram over lysdioderne til binær visning.

Grunden til at LED7 er koblet direkte til stikket er, at der inde på PIC-boardet sidder en 4,7 kΩ modstand i serie med signalet ind til Port B6.

Målinger på lysdioder

Jeg havde først koblet kredsløbet op med en 680 Ω på Port B6, men så at der var mindre lys i den lysdiode, så jeg målte hvad der skete.
Alle de andre lysdioder får omkring 4,56 mA, hvilket er acceptabelt, når jeg har dimensioneret dem til 5 mA.
Den sidste LED7 får kun 0,59 mA når der er både 4,7 kΩ og 680 Ω, hvilket giver alt for stor en forskel i lyset.
Hvis jeg fjerner de 680 Ω, så kommer strømmen op på 0,67 mA
Det giver noget mindre lys i LED7, da den får under men 0,67 mA, men til testformål er det acceptabelt. Det ville ikke have været i orden til noget der skulle bruges i en færdig opstilling.
Grunden til at jeg ikke kobler 8 lysdioder op er, at jeg begrænser tællingen til 99, og med 7 lysdioder kan jeg få vist op til $2^7 - 1 = 127$.

Koden til binær visning

Alle ben på Port B defineres til output:

```
port_b_direction = all_output
```

Det antal gange der er talt op fra motionsbænken ligger i variablen count, det tælles op som beskrevet:

```
  count = count + 1  -- Det er i denne situation vi tæller
```

Umiddelbart betragter jeg tallet inde i PIC'en som et decimaltal, og det tælles fremad ved at lægge 1 til hver gang. I praksis så er tallet faktisk lagret i en Byte, der består af 8 bit, og det kan jeg udnytte ved at lægge den Byte ud på port B, hvor den vil være at se som bit:

```
  -- Tallet er præsenteret binært i count, så tallet kan lægges direkte ud
  Port_B = count
```

Testkoden til dette ligger som elektronisk bilag: **binaer.jal**

Test af den binære visning

Jeg har testet at tællingen foregår korrekt i de binære koder, som vist her:

B6	B5	B4	B3	B2	B1	B0	Talværdi
0	0	0	0	0	0	0	0
0	0	0	0	0	0	1	1
0	0	0	0	0	1	0	2
0	0	0	0	0	1	1	3
0	0	0	0	1	0	0	4
0	0	0	0	1	0	1	5
0	0	0	0	1	1	0	6
0	0	0	0	1	1	1	7
0	0	0	1	0	0	0	8
0	0	0	1	0	0	1	9
0	0	0	1	0	1	0	10
0	0	0	1	0	1	1	11
0	0	0	1	1	0	0	12
0	0	0	1	1	0	1	13
.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.
1	1	0	0	0	1	0	98
1	1	0	0	0	1	1	99

Tabel der illustrerer den binære visning.

Jeg har testet tællingen op til de 99, og det forløber som forventet, og den stopper ved 99 som programmet specificerer.

Input fra Brugeren

De muligheder brugeren skal have for at betjene enheden er:
Nulstilling af tælleren.
Start og stop tællingen.

I første omgang giver det ikke så meget mening at man kan starte og stoppe tællingen, og den første del jeg lavede, var da også reset, så brugeren kan nulstille tællingen og begynde forfra.

Grunden til at der skal være start og stop er, at der på et tidspunkt skal laves sådan at den også kan registrere tiden der er forløbet i tællingen, så jeg har valgt at tage det med fra starten.

Princippet i kontakterne til brugeren er præcist det samme som til kontakterne på motionsbænken.

Diagram over input fra brugeren (bruger-betjening)

Ledningerne til disse kontakter er ikke nært så lange som dem der går ud til motionsbænken, og kommer ikke til at samle samme mængde støj op, så modstandene kan godt dimensioneres større.
Jeg vælger her 100kΩ

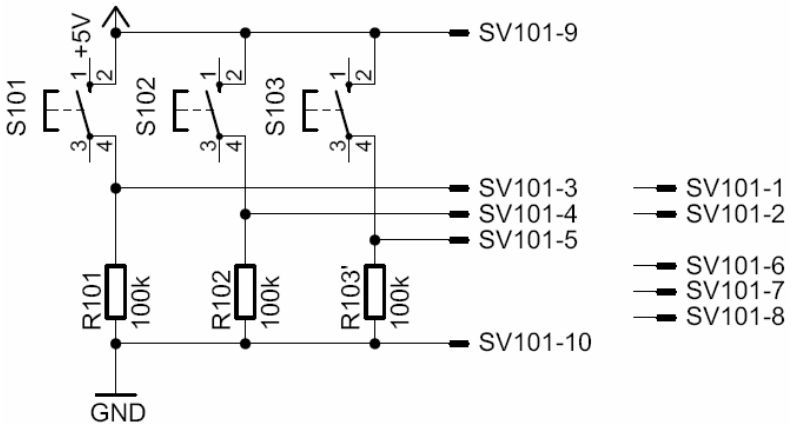
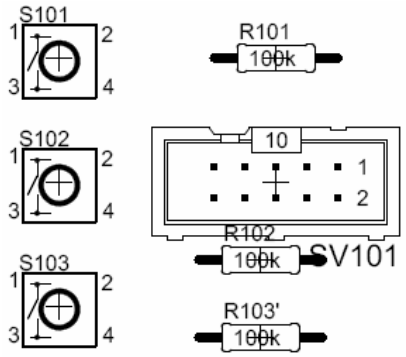
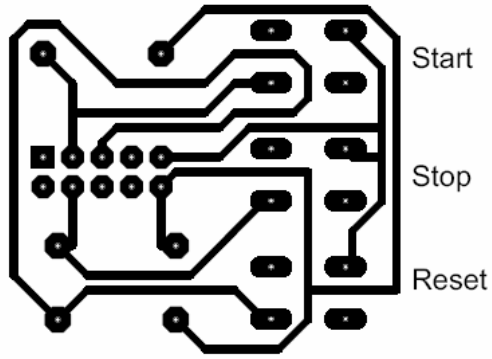


Diagram over kontaktprint til bruger-betjening.

Som det kan ses starter nummereringen af komponenterne her fra 101. Der er for at jeg kan skelne mellem de forskellige print.
SV101 skal også kobles til Port A, så jeg bruger nogen andre ben til disse kontakter.
Det har jeg gjort ved at klemme et ekstra fladkabel-stik på fladkablet over til mikroswitch-printet.



Komponentplacering kontakter.



Kobbersiden af printet med kontakter til brugerbetj.

Snitflade mellem styre-enhed og printet til input fra brugeren

PIC-stik Ben	PIC-funktion	Beskrivelse
Ben 3	Port A2	Start af træningsperiode
Ben 4	Port A3	Stop af træningsperiode
Ben 5	Port A4	Nulstil tæller (og tid)
Ben 9	+5V	+5V på PIC-kredsløbet
Ben 10	Stel	Fælles stel

Tabel over snitfladen mellem printet til input fra brugeren og styre-enheden.

Software til input fra brugeren

De 3 kontakter skal have betjeningsmulighederne som beskrevet, og de defineres som følger

```
var bit Start   is pin_a2
var bit Stop    is pin_a3
var bit Reset   is pin_a4

pin_a2_direction = input
pin_a3_direction = input
pin_a4_direction = input
```

Der defineres en variable, der angiver om træningen er i gang.

```
var bit igang = false
```

Start og stop af tællingen kodes på følgende måde:

```
  -- Håndter start og stop-knapperne
  if Start then
     igang = true
  end if
  if Stop then
     igang = false
  end if
```

Det er variablen igang der bestemmer om maskinen må tælle.
Uden om testen på tilstandene, som jeg har vist koden til, der laver jeg en if på om tællingen er i gang, så jeg kun tæller når jeg må det:

```
  -- Vi tæller kun hvis maskinen er i gang
  if igang then
     -- Registrer hvilken tilstand maskinen er i
        .
        .
        .
  end if
```

Og det er kun hvis maskinen ikke er i gang, at man kan resette:

```
  if ! igang then
     if Reset then -- Nulstiller tællingen
        count = 0
        tilstand = 0
     end if
  end if
```

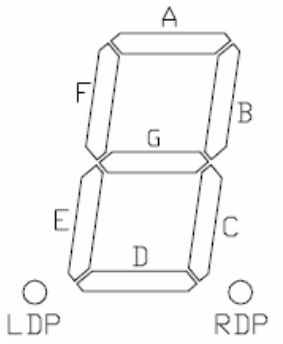
Når jeg bruger ! igang betyder det, at jeg tester på om den er det omvendte (inverterede), og da igang er en logisk variabel, der kun kan være sand eller falsk, så bliver det det modsatte af hvad igang er.
Reset nulstiller også tilstanden, så den starter forfra I sekvensen.

Test af koden til input fra brugeren

Koden er testet igennem, og det virker at man kan starte og stoppe tællingen, og man kan nulstille, når den ikke er i gang med at tælle, så det fungerer som det skal.
Denne kode ligger som elektronisk bilag: **input.jal**

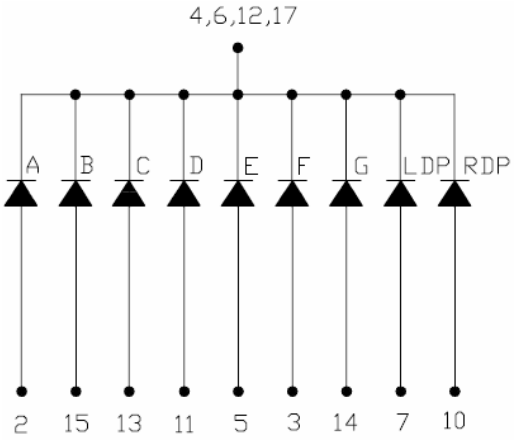
Visning af resultat på ét 7-segment

Det kan være svært for brugeren at aflæse det binære tal, så derfor skal der laves noget der er mere brugervenligt.
Et naturligt valg et er 7-segment display, der egentlig bare et 7 lysdioder placeret i samme hus, så man kan tænde dem individuelt. Det gør at man kan få et display til at forme cifrene fra 0 til 9 (og nogle bogstaver, men det er ligegyldigt her).



Segmenternes placering.

Jeg har valgt at bruge et 7-segment af typen LTS-3403, da det passer fint til det formål jeg har. Databladet er fundet på El-Teknik siden[4]
I databladet kan jeg se, at de 7 segmenter navngives A til G. Der er også to muligheder for komma, men det har jeg ikke brug for i denne sammenhæng.



7-segmentets interne diagram.

Elektrisk er de 7 lysdioder koblet sammen som det er vist i databladet (de to sidste lysdioder er naturligvis de to decimal-kommaer).

Som det kan ses, så har lysdioderne fælles forbindelse på katoderne (common cathode), og det passer med at jeg kan koble fælles-benene til stel, så jeg rent elektrisk kan lave samme opkobling som ved de binære lysdioder.

Der kommer blot et problem her, når jeg bruger Port B6, nemlig at det segment ikke lyser nær så meget, og da jeg gør dette for brugere, så er det ikke acceptabelt.

Snitflade mellem PIC Port B og et 7-segment display

Snitfladen er forberedt, så den kan bruges til to 7-segment display, ved at Port B0 er holdt fri

PIC-stik Ben	PIC-funktion	Beskrivelse
Ben 1	Port B0	Reserveret til fremtidig brug
Ben 2	Port B1	Segment A
Ben 3	Port B2	Segment B
Ben 4	Port B3	Segment C
Ben 5	Port B4	Segment D
Ben 6	Port B5	Segment E
Ben 7	Port B6	Segment F Aktiv Lav
Ben 8	Port B7	Segment G Aktiv lav
Ben 9	+5V	+5V på PIC-kredsløbet
Ben 10	Stel	Fælles stel

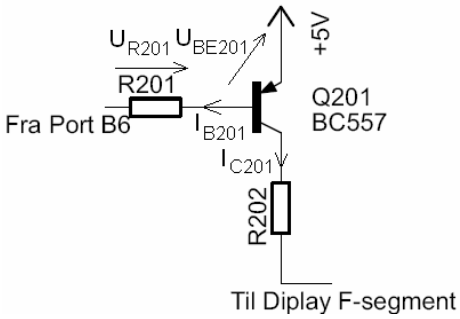
Tabel over benforbindelserne ud til et 7-segment

[4] Datablade på El-Teknik hjemmesiden http://www.holstebrohtx.dk/bar/index.php?pkt=150 set 6/10 2008.

Transistor til forstærkning af strømmen til 2 segmenter

Jeg vil gerne reservere Port B0 til noget specielt, så jeg skal bruge både Port B6 og B7 til at tænde segmenter med, og da de begge har 4,7 kΩ i serie med signalet, så må jeg gøre noget specielt.

Jeg vælger at sætte en transistor på til at trække signalet højt, og til det formål vælger jeg en BC557, der er en PNP transistor, og den skal kobles som vist her.



Transistor der forstærker strømmen til et segment.

I første omgang vil jeg kunne regne med at R202 skal have samme værdi på 680 Ω som den R3 jeg beregnede tidligere til en lysdiode.
Transistoren Q201 anvendes i dette tilfælde som switch (en kontakt), og derfor skal jeg sikre mig, at der går strøm nok i Basis-benet, til at trække den ønskede strøm i Collector-benet.
Ved at læse i databladet kan jeg se at, med en collector-strøm på 10 mA regner de med en basis-strøm på 0,5 mA.

Det svarer til en strømforstærkning på $H_{FE} = \frac{I_C}{I_B} = \frac{10mA}{0,5mA} = 20gange$

Jeg kan også se at de regner med samme strømforstærkning op til en collector-strøm på 100 mA, så det må være en acceptabel størrelse at regne med.

Ud fra dette kan jeg regne min basis-strøm som $I_{B201} = \frac{I_{C201}}{20} = \frac{5mA}{20} = 0,25mA$

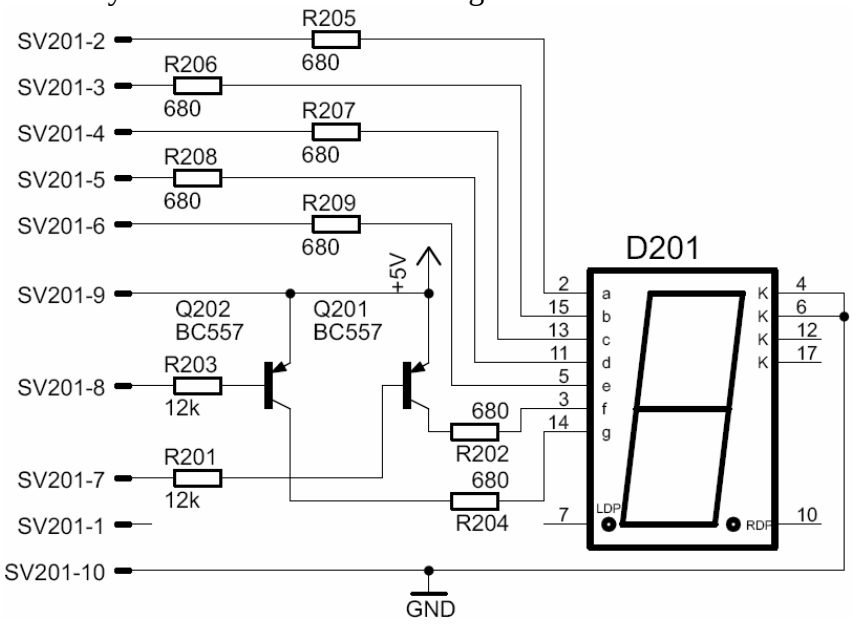
Når Q201 skal tændes, så sker det ved at lægge 0 ud på Port B6, hvilket giver en lav spænding omkring 0V. En transistor der er tændt har ca. 0,7 V liggende over basis-emitter, så den spænding der er tilbage til R201 må være $U_{R201} = 5V - U_{BE201} = 5V - 0,7V = 4,3V$

Ud fra dette kan R201 beregnes til $R201 = \frac{U_{R201}}{I_{B201}} = \frac{4,3V}{0,25mA} = 17,2k\Omega$

Da der sidder 4,7 kΩ inde på printet skal denne størrelse trækkes fra så vi ender på 12,5 kΩ.
Den nærmeste værdi vi har, der er mindre end det er en 12 kΩ. Grunden til at der rundes ned er, at så er jeg sikker på at transistoren får strøm nok til at tænde.

Diagram til Visning af resultat på et 7-segment

Kredsløbet er kobles op ud fra det kredsløb der forstærker til et segment, og ellers på samme måde som til lysdioderne ved binær visning.



Diagrammet over kredsløbet med et 7-segment display.

Som nævnt i beregningerne, så er vi nu nødt til at sætte Port B6 og B7 lave, for at tænde segmenterne, men det klares ganske enkelt ved at rette det inde i koden der styrer displayet.

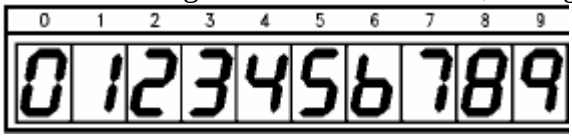
Kredsløbet er opbygget på fumlebræt, og i første omgang prøvede jeg at teste med den kode der bare talte binært på Port B, men der kunne jeg ikke få slukket G-segmentet, da jeg ikke kunne få ben 8 til at gå høj, men det testede jeg bare ved at sætte den på ben 1 (Port B0), så kunne alle segmenter tænde og slukke.

Koden til et styre et 7-segment display

Ideen i at styre displayet er nogenlunde den samme som det at lave den binære visning, nemlig at hele værdien til displayet lægges ud på port B på en gang.

Opbygningen af cifrene 0 – 9

Lysdioderne skal tændes, så de kan danne de cifre vi ønsker. Jeg har fundet et godt oplæg i databladet for en 7 segment dekoder 74LS47, der også findes på El-Teknik hjemmesiden.



Segmenternes måde at danne cifrene 0 – 9.

For at gøre det lettere at skrive programmet, har jeg lavet en tabel over hvilke segmenter der er tændt i hvilke cifre.
I tabellen betyder 1 at segmentet er tændt og 0 betyder slukket.

Ciffer	G	F	E	D	C	B	A
0	0	1	1	1	1	1	1
1	0	0	0	0	1	1	0
2	1	0	1	1	0	1	1
3	1	0	0	1	1	1	1
4	1	1	0	0	1	1	0
5	1	1	0	1	1	0	1
6	1	1	1	1	1	0	0
7	0	0	0	0	1	1	1
8	1	1	1	1	1	1	1
9	1	1	0	0	1	1	1

Tabel over hvordan segmenterne danner cifrene.

For at gøre det koden enklere, så oprettes der en række konstanter i softwaren, der svarer til tabellen der danner cifrene. Det ser ud som følger:

```
-- Definer konstanter til 7-segment
-- Segmenterne er forbundet G F E D C B A
-- til Port B pin           7 6 5 4 3 2 1
-- Ben 6 og 7 er inverterede fordi der skal en PNP transistor på
-- Bit 0 er altid 0
const byte vis0 = 0b1011_1110
const byte vis1 = 0b1100_1100
const byte vis2 = 0b0111_0110
const byte vis3 = 0b0101_1110
const byte vis4 = 0b0000_1100
const byte vis5 = 0b0001_1010
const byte vis6 = 0b0011_1000
const byte vis7 = 0b1100_1110
const byte vis8 = 0b0011_1110
const byte vis9 = 0b0000_1110
```

Selve visningen sker nu nede i forever-loopet, efter vi har registreret om der skal tælles. Det sker med følgende kode:

```
  -- Sæt den værdi ud på displayet, der svarer til cifferet
  if count == 0 then
     Port_B = vis0
  elsif count == 1 then
     Port_B = vis1
  elsif count == 2 then
     Port_B = vis2
  elsif count == 3 then
     Port_B = vis3
  elsif count == 4 then
     Port_B = vis4
  elsif count == 5 then
     Port_B = vis5
  elsif count == 6 then
     Port_B = vis6
  elsif count == 7 then
     Port_B = vis7
  elsif count == 8 then
     Port_B = vis8
  elsif count == 9 then
```

```
    Port_B = vis9
end if
```

Test af koden til visning af resultat på et 7-segment

Koden er testet, og displayet kan vise alle 10 cifre som ønsket.
Hvis man tæller ud over 9, så sker der det, at displayet bliver ved med at vise 9, lige indtil man resetter, hvor der så vises 0.
Grunden til det er, at den ikke rammer noget i else-if strukturen, når count bliver større end 9, og dermed kommer det ikke noget nyt ud på porten, når count kommer over 9.

Koden har den ønskede funktion, og jeg accepterer, at den kun virker begrænset, da næste trin er at få to 7-segmenter koblet op, så jeg kan vise op til 99.
Denne kode ligger som elektronisk bilag: **1-syv-segmentjal**

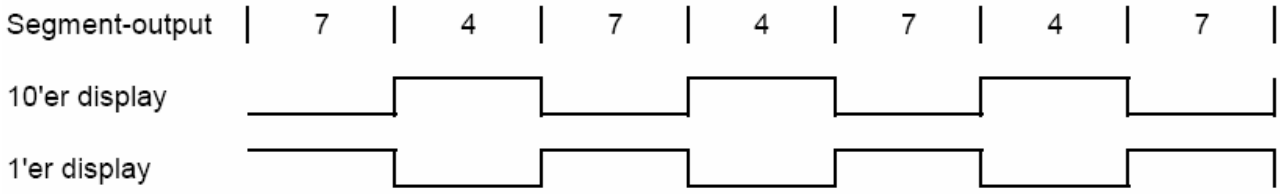
Visning af resultat på to 7-segmenter

Den umiddelbart simple måde at koble to 7-segment displays på PIC-boardet ville være at bruge en port mere til det næste display, men på det PIC-board vi har er der ikke flere porte, så vi må til at genbruge noget.

Princippet i Multiplexing

Teknikken jeg vil bruge for at arbejde med 2 displays er det der hedder multiplexing, hvilket går ud på, at man tænder først det ene display kortvarigt, med det tal der skal stå i det, og derefter tændes det andet display kortvarigt med hvad det skal vise.
På den måde kan man genbruge de 7 portben, der går ud til det ene display.
Den eneste betingelse er, at det sker så hurtigt at øjet ikke opfatter det. Jeg valgte at lade hvert display være tændt i 5 ms.

På et tidsdiagram vil en visning af tallet 47 kunne skitseres som følger:



Visningen af tallet 47 multiplexet ud, så segmenterne genbruges.

De to displays kan i værste fald komme til at trække 7 gange så stor strøm som de enkelte segmenter, så de kan ikke trækkes direkte af PIC'ens port. Derfor skal der en transistor til at trække mod stel.

I det første forsøg jeg lavede brugte jeg to portben fra port A til at vælge hvilket display der skulle være tændt. Jeg fandt ud af flere ting jeg skulle tage højde for i konstruktionen ved multiplexing.
Den første var at lysstyrken faldt til det halve i displayet, hvilket også er logisk nok, når det enkelte display kun er tændt i halvdelen af tiden. Løsningen må være at sende dobbelt så stor strøm ud til displayet når det er tændt.
Det næste jeg fandt ud af var, at jeg skulle passe på, at det ene displays visning ikke kom over i det andet display. Problemet er at man ikke kunne skifte segmenterne og hvilket display der skulle lyse på præcist samme tidspunkt, da det skulle ud på hver sin port.

Løsningen i første forsøg var at jeg slukkede begge displays mens jeg skiftede segmenterne.
Det første forsøg med multiplexing er ikke dokumenteret yderligere her i rapporten, da jeg ikke kom videre med det, end til at få det testet på fumlebræt, og konstateret at det virkede.
Denne kode ligger som elektronisk bilag: **temp-2-syv.jal**

Snitflade mellem PIC Port B og to 7-segment displays

Snitfladen er stort set den samme, som den der bruges til et 7-segment display. Den eneste ændring er, at Port B0 nu styrer af hvilket display der er aktivt.

PIC-stik Ben	PIC-funktion	Beskrivelse
Ben 1	Port B0	Valg af display (høj er 1'erne)
Ben 2	Port B1	Segment A
Ben 3	Port B2	Segment B
Ben 4	Port B3	Segment C
Ben 5	Port B4	Segment D
Ben 6	Port B5	Segment E
Ben 7	Port B6	Segment F Aktiv Lav
Ben 8	Port B7	Segment G Aktiv lav
Ben 9	+5V	+5V på PIC-kredsløbet
Ben 10	Stel	Fælles stel

Tabel over benforbindelserne ud til to 7-segment displays.

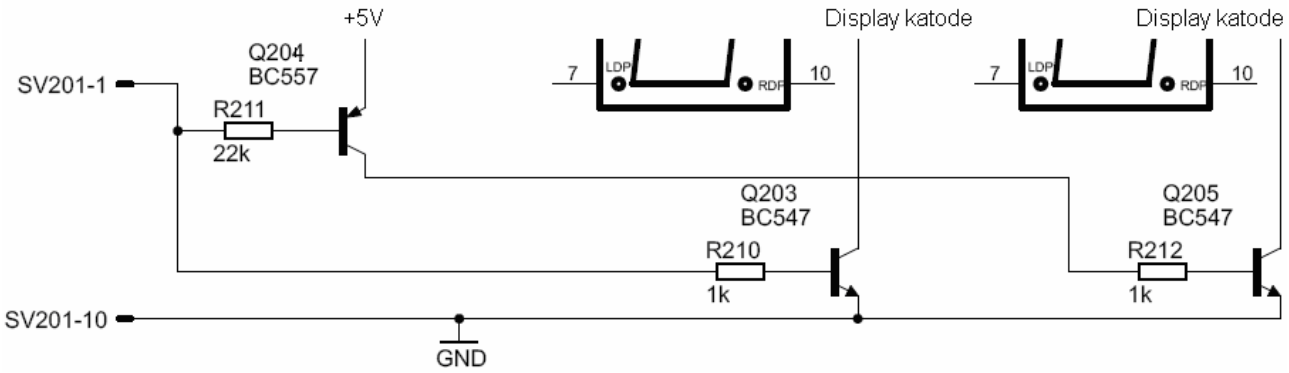
I den videre konstruktion af displayet udnytter jeg, at store dele af diagrammet er det samme som ved et display.

Jeg kobler de to displays sammen, så alle segmenterne er fælles, og kan styres fra port B. Grunden til jeg kan gøre det er, at segmenterne består af dioder, så segmenterne i det display der ikke skal lyse kommer til at være i spærre-retningen, ved at jeg slukker for den fælles katode til displayet.

Den måde jeg tænder og slukker for den fælles katode til displayene er ved at koble en NPN transistor til stel, og tænder transistoren, når displayet skal være tændt, og slukker den, når den skal være slukket.

Beregninger på modstande og transistorer ved 2 displays.

Den første ændring kræver ikke så meget beregning, da jeg blot skal have fordoblet strømmen i segmenterne, og det kan jeg gøre ved simpelthen at halvere modstandene R202, R204 – R209, så de bliver 330 Ω. Jeg skal ligeledes have fordoblet strømmen i R201 og R203 – det kan ikke ske ved en simpel halvering af dem, da der sidder 4,7 kΩ i serie med dem, så det er de 17,2 kΩ fra beregningen der skal halveres til 8,6 kΩ, og når de 4,7 kΩ trækkes fra, så kommer vi ned på 3,9 kΩ til R201 og R203.



Deldiagram, der viser de 3 transistorer, der kan tænde de to 7-segment displays på skift.

Når hvert segment nu kan trække ca. 10 mA så bliver den samlede strøm i segmentet 7 gange så stor, altså 70 mA, så den strøm skal Q203 kunne trække lav.
Hvis vi stadig regner med en H_{FE} på 20 gange, så må vi komme frem til følgende basis-strøm:

$$I_{B203} = \frac{I_{C203}}{H_{FE}} = \frac{70mA}{20} = 3,5mA$$

Spændingen over R210 må blive: $U_{R210} = U_{PIC,high} - U_{BE203} = 4,8V - 0,7V = 4,1V$

R210 bliver derfor: $R210 = \frac{U_{R210}}{I_{B203}} = \frac{4,1V}{3,5mA} = 1,17k\Omega$

R210 rundes ned, så vi er sikre på at transistoren tænder, R210 og R212 bliver 1 kΩ.

Q204 skal blot sikre strømmen til Q205, altså ca. 3,5 mA, så basis må være 20 gange mindre:

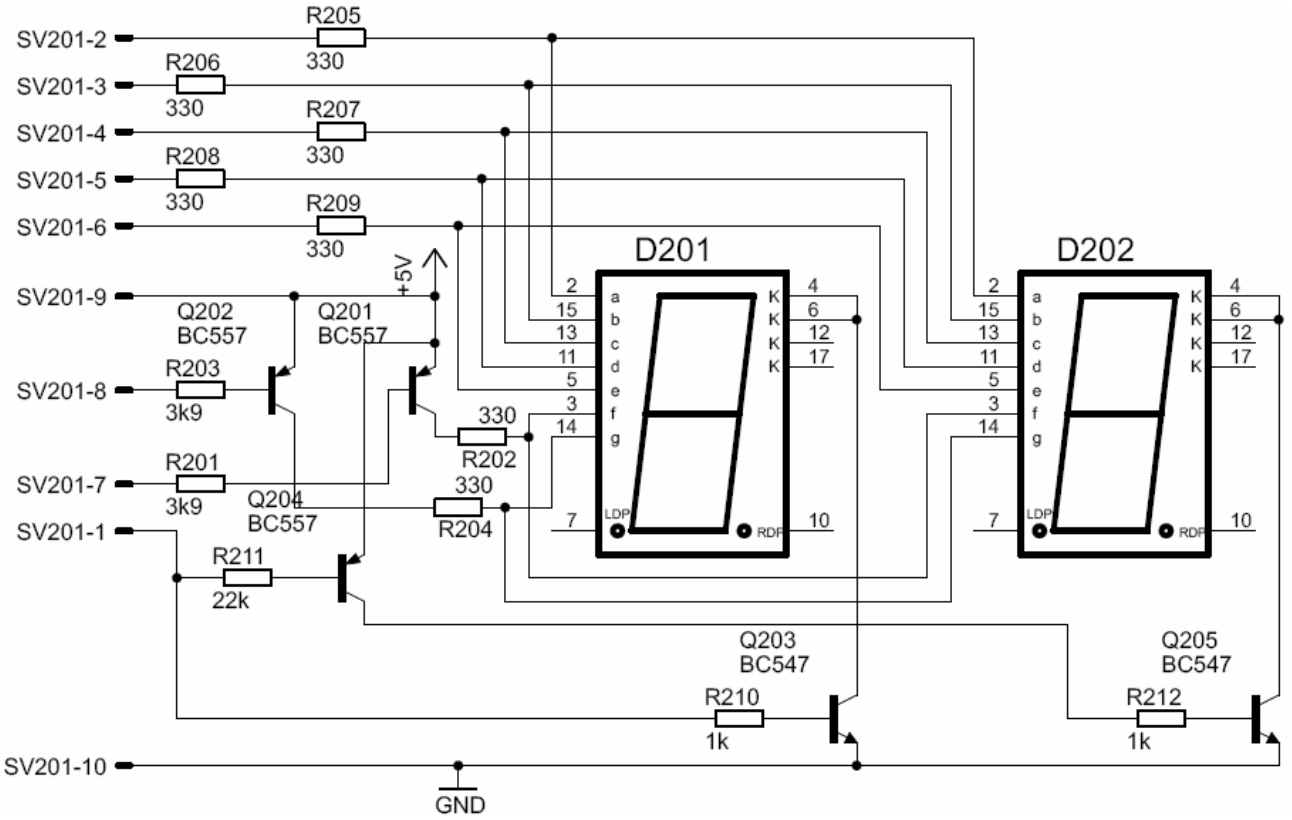
$$I_{B204} = \frac{I_{C204}}{H_{FE}} = \frac{3,5mA}{20} = 0,175mA$$

Der må igen være ca. 4,1 V over R211, så den bliver:

$$R211 = \frac{U_{R211}}{I_{B204}} = \frac{4,1V}{0,175mA} = 23,4k\Omega$$

Igen rundes værdien ned, så R211 bliver 22 kΩ.

Diagram med to 7-segment displays



Diagrammet med to 7-segment displays til multiplexing.

Diagrammet viser de to displays til visning af hhv. 10'ere og 1'ere.
Q203 tænder D201, når Port B0 er høj. Samtidigt slukkes Q204 og Q205, så D202 er slukket.
Når Port B0 er lav, så slukker Q203, og Q204 tænder. Når Q204 er tændt tænder Q205, der tænder D202.
Princippet i de resterende modstande og transistorer er det samme som ved et display.

Koden til multiplexing af to displays

Koden er lidt mere kompliceret, så jeg vil starte med at give et overblik i pseudo-kode:

```
forever loop
    registrering af tælling
    registrring af bruger input
    Beregn et ciffer, efter hvilket ciffer der skal vises
    Find segmenterne til cifferet
    læg segmenterne sammen med hvilket display der skal aktiveres
    læg resultatet ud på porten
    vent 5 ms
    skift hvilket display der skal vises næste gang
end forever loop
```

Pseudo-kode af multiplexingen.

Til multiplexingen defineres nogle ekstra variabler

```
var byte ciffer
var byte temp
var bit skift = false
```

Registreringen af tællingen og af bruger input er ikke ændret fra visningen med binære dioder, så den vil jeg ikke forklare igen.

```
-- Multiplexing af displays
if skift then
   ciffer = count / 10  -- heltals division, runder ikke af
else
   ciffer = count % 10  -- resultatet er resten ved en division med 10
end if
```

Variablen skift angiver hvilket display der er aktivt, hvis den er sand, så er det 10'erne, og hvis den er falsk, så er det 1'erne.
Resultatet gemmes i variablen ciffer, der et det der skal ud på segmenterne.
Måden 1'erne bliver beregnet på er lidt speciel, da % i programmeringssproget JAL (lige som i C) er en modulus operator, der beregner resten ved en heltals-division, så f.x. 47 % 10 giver resultatet 7, altså det der er resten ved en division med 10.

Når det ønskede ciffer er fundet, så skal segmenterne beregnes, og der går efter same princip som ved visningen på et 7-segment display, bortset fra at vi ikke kan lægge det direkte ud på porten.

```
-- Husk segmenterne alt efter hvilket ciffer det er
-- Vi kan ikke tænde direkte, da der altid er et display der lyser
if ciffer == 0 then
   temp = vis0
elsif ciffer == 1 then
   temp = vis1
elsif ciffer == 2 then
   temp = vis2
elsif ciffer == 3 then
   temp = vis3
elsif ciffer == 4 then
   temp = vis4
elsif ciffer == 5 then
   temp = vis5
elsif ciffer == 6 then
   temp = vis6
elsif ciffer == 7 then
   temp = vis7
elsif ciffer == 8 then
   temp = vis8
elsif ciffer == 9 then
   temp = vis9
end if
```

Ud fra variablen ciffer bestemmes hvilke segmenter der skal tændes, og det gemmes i variablen temp.

```
-- Tænd så det display der hører til det ciffer der vises
if skift then
   temp = temp | 0b0000_0001  -- sætter bitten i b0
end if
```

Hvis skift er sand, så skal mest betydende ciffer vises, og derfor lægges der et 1-tal ind på bit 0, så port B0 bliver høj.
Det vil gøre at display D201, altså 10'erne tændes som de skal.
Hvis skift er falsk, så er det D202 der tændes.

```
-- Nu kan vi skrive cifferet ud, samtidig med at det rigtige display
-- bliver tændt
port_B = temp

-- Vent de 5 ms, så brugeren ikke ser det flimrer
delay_5ms
```

Da vi har både informationen om hvilket display der skal tændes, og hvilke segmenter der skal lyse i det display, stående i temp, så kan vi lægge det hele ud på Port B på en gang, og dermed undgå at indholdet af de to displays "skygger" ind over hinanden.

Når visningen er kommet ud på displayet, holder vi en aktiv pause på 5 ms.
Det gør at hele displayet er vist i løbet af 10 ms, altså en frekvens på 100 Hz, hvilket det menneskelige øje ikke kan registrere.

```
-- Gør klar til at vise det andet display
skift = ! skift
```

Det sidste der skal gøres er at invertere værdien i skift, så vi får vist det andet ciffer i næste gennemløb af forever-loopet.

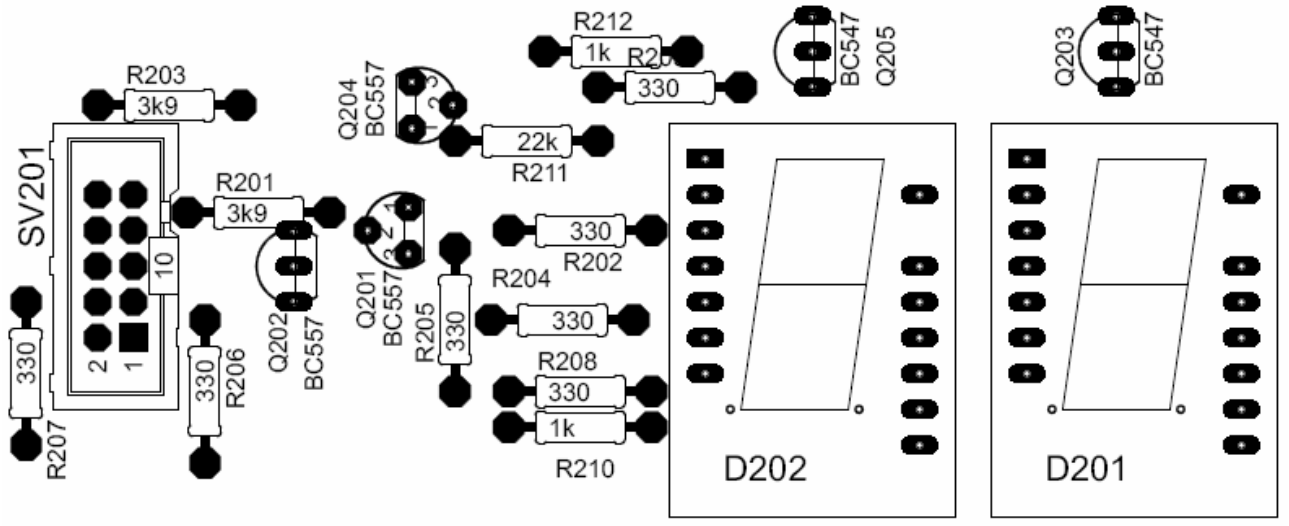
Test af koden til to 7-segment displays

Jeg har testet igennem, at jeg kan få visningen på de to displays til at fungere som jeg havde tænkt det.
Efter at have testet den basale funktion af multiplexingen, så testede jeg koden igennem igen ud i alle "hjørnerne", for at sikre mig, at jeg ikke havde fået indført nogen fejl, ved den videre udbygning af softwaren.
De ting jeg har testet er følgende:

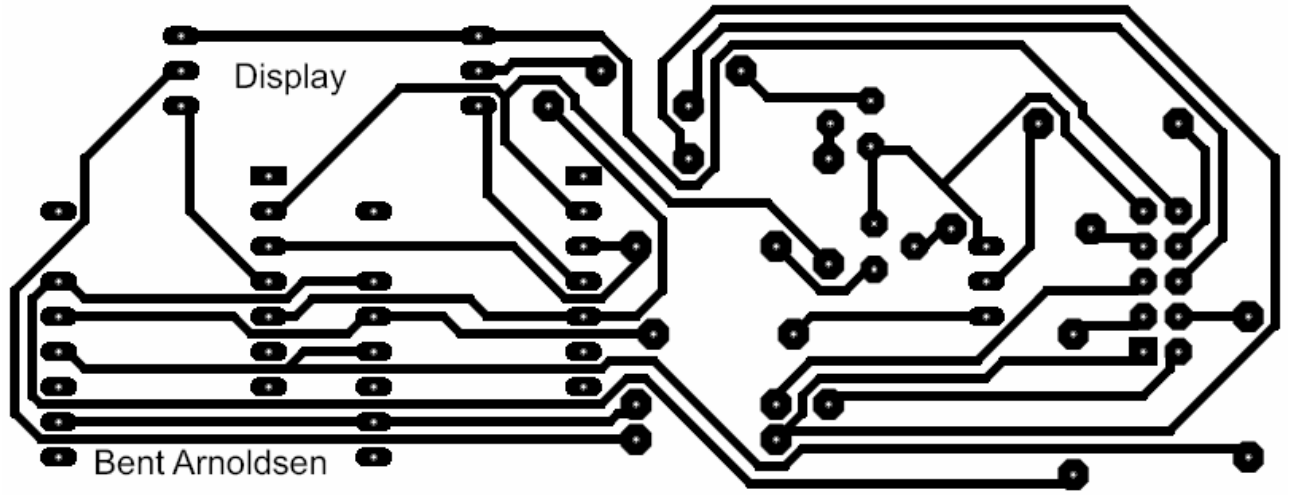
- start og stop fungerer. Der kan ikke tælles, når den er stoppet.
- reset fungerer. Der kan ikke resettes, når man er i gang med træningen.
- det er ikke muligt at snyde tællingen ud fra de 4 muligheder jeg har skitseret.
- tælleren kan gå fra 00 til 99, og stopper der. Alle 100 visninger er korrekte.

Print til to 7-segment Display

De to displays er lagt ud på print sammen med modstande, transistorer og stik til porten.



Komponentplaceringen til de to 7-segment display.



Printet til de to 7-segment display, set fra kobbersiden.

Test af printet med de to 7-segment display

Da jeg testede printet til de to 7-segment display fandt jeg ud af, at jeg havde lavet en fejl, nemlig at de to display sad omvendt, med 1'erne først og 10'erne til sidst.

I stedet for at lave printet om valgte jeg den enkle løsning, at rette i softwaren.

```
-- Multiplexing af displays
if skift then
   ciffer = count % 10  -- resultatet er resten ved en division med 10
else
   ciffer = count / 10  -- heltals division, runder ikke af
end if
```

Rettelsen består i, at jeg beregner det modsatte ciffer, når jeg tester på skift.
Det løste problemet.
Denne endelige kode ligger både som Bilag 1 og som elektronisk bilag: **traen-2-syv.jal**

Konklusion

Jeg har fået lavet en konstruktion der kan tælle de sekvenser op, som træningsmaskinen gennemgår, og jeg har fået brugerens input til at fungere som ønsket.

Jeg har testet igennem at man ikke kan snyde konstruktionen ud fra de muligheder jeg har opstillet.

Jeg har forklaret hvordan koden til PIC'en er opbygget, og hvordan den udfører de funktioner jeg ønsker.

Jeg har nogle enkelte ting der ikke fungerede optimalt i de enkelte mellemstadier, man da det ikke skulle bruges i det færdige resultat, så finder jeg ikke det væsentligt.

Jeg har ikke fået arbejdet med tids-registrering inde i PIC'en, og har heller ikke fået koblet hverken LCD-display eller PC på.

Perspektivering

Der er stadig flere muligheder for at gøre konstruktionen bedre.

Det næste jeg ville arbejde med var at få den til at registrere tid, så jeg også kunne få det vist i displayet. Det ville godt nok kræve en ændring i betjeningen af enheden, da man skal kunne angive hvad man vil have vist. Det kunne gøres ved at lave start-stop på en knap, og visning af tid på den der var stop.

Når tiden fungerer, så ville jeg så udskifte visningen af resultatet fra to 7-segment med et LCD display.
Det ville give mig muligheden for at kunne vise både tid og antal samtidigt, så der ville jeg nok skifte tilbage til den oprindelige bruger-betjening.

En anden overvejelse er at der vil kunne ske overløb af tællerne.
Det har indtil videre været begrænset ved at stoppe tællingen ved 99, og det havde nok også været løsningen ved tiden.
Med et LCD display kunne man enkelt få visningen op til 255, og det ville sikkert være nok til tællingen, men det kunne måske være smart at registrere tiden i sekunder, og lade den løbe fra 0 til 59, og derefter tælle minutterne i en separat variabel., så visningen af tiden ville blive i minutter og sekunder.

Endelig kunne man gøre det, at man kunne lave kommunikation at resultaterne op til en PC. For at gøre det enkelt, så ville det være smart at udskifte PIC-boardet med den type, hvor der sidder en 16F628, der har seriel kommunikation indbygget, og på det board sidder der også et RS-232 stik, så der er enkelt at forbinde det til computeren.
Det ville så kræve ret meget programmering i PC'en, hvis man skulle lave et registrerings-system til at lagre sine trænings-pas med.

Bilag 1 - Den endelige kode til trænings-maskinen

Her er gengivet den endelige kode til træningsmaskinen, så man kan danne sig et totalt overblik over koden.
De forskellige udviklings-stadier i koden ligger som en ZIP-fil, der er vedlagt som bilag.

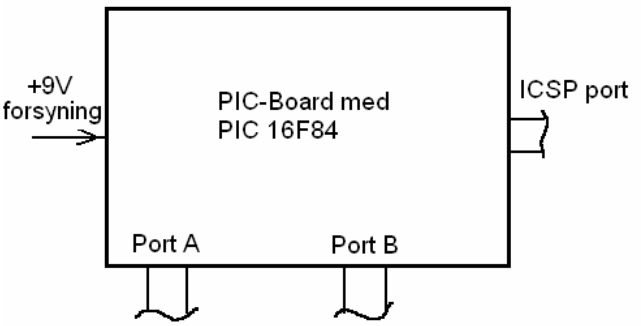
```
include 16f84_4
include jlib

-- Definer navne på port ben
var bit display is pin_b0   -- Blot for at reservere benet

var bit Passiv  is pin_a0
var bit Aktiv   is pin_a1
var bit Stop    is pin_a2
var bit Start   is pin_a3
var bit Reset   is pin_a4

-- Definer Input / Output
pin_a0_direction = input
pin_a1_direction = input
pin_a2_direction = input
pin_a3_direction = input
pin_a4_direction = input

port_b_direction = all_output

-- Definer konstanter til 7-segment
-- Segmenterne er forbundet g f e d c b a
-- til Port B pin           7 6 5 4 3 2 1
-- Ben 6 og 7 er inverterede fordi der skal en PNP transistor på
-- Bit 0 er altid 0
const byte vis0 = 0b1011_1110
const byte vis1 = 0b1100_1100
const byte vis2 = 0b0111_0110
const byte vis3 = 0b0101_1110
const byte vis4 = 0b0000_1100
const byte vis5 = 0b0001_1010
const byte vis6 = 0b0011_1010
const byte vis7 = 0b1100_1110
const byte vis8 = 0b0011_1110
const byte vis9 = 0b0001_1110

-- Definer de variabler der bruges
var byte tilstand = 0
var byte count = 0
var byte ciffer
var byte temp
var bit skift = false
var bit igang = false

forever loop        -- fortsæt i det uendelige

  -- Håndter start og stop-knapperne
  if Start then
     igang = true
  end if
  if Stop then
     igang = false
  end if
```

```
-- Vi tæller kun hvis maskinen er i gang
if igang then
   -- Registrer hvilken tilstand maskinen er i
   if tilstand == 0 then    -- Den er i Passiv, vi venter på den trækkes ud
      if (! Aktiv) & (! Passiv) then
         tilstand = 1
      end if
   elsif tilstand == 1 then -- Den trækkes udad, vi venter på Aktiv
      if (Aktiv) & (! Passiv) then
         tilstand = 2
      end if
   elsif tilstand == 2 then -- Den er i Aktiv, vi venter på den går tilbage
      if (! Aktiv) & (! Passiv) then
         tilstand = 3
      end if
   elsif tilstand == 3 then -- Den er på vej tilbage, vi venter på Passiv
      if (! Aktiv) & (Passiv) then
         tilstand = 0
         count = count + 1  -- Det er i denne situation vi tæller
         if count == 100 then
            count = 99      -- Håndter at vi ikke gå videre
         end if
      end if
   else  -- Blot for en sikkerheds skyld, det skulle aldrig ske
      tilstand = 0
   end if

else -- Vi kan kun resette hvis maskinen er stoppet
   if Reset then -- Nulstiller tællingen
      count = 0
      tilstand = 0
   end if
end if

-- Multiplexing af displays
if skift then
   ciffer = count % 10  -- resultatet er resten ved en division med 10
else
   ciffer = count / 10  -- heltals division, runder ikke af
end if

-- Husk segmenterne alt efter hvilket ciffer det er
-- Vi kan ikke tænde direkte, da der altid er et display der lyser
if ciffer == 0 then
   temp = vis0
elsif ciffer == 1 then
   temp = vis1
elsif ciffer == 2 then
   temp = vis2
elsif ciffer == 3 then
   temp = vis3
elsif ciffer == 4 then
   temp = vis4
elsif ciffer == 5 then
   temp = vis5
elsif ciffer == 6 then
   temp = vis6
elsif ciffer == 7 then
   temp = vis7
elsif ciffer == 8 then
   temp = vis8
elsif ciffer == 9 then
   temp = vis9
```

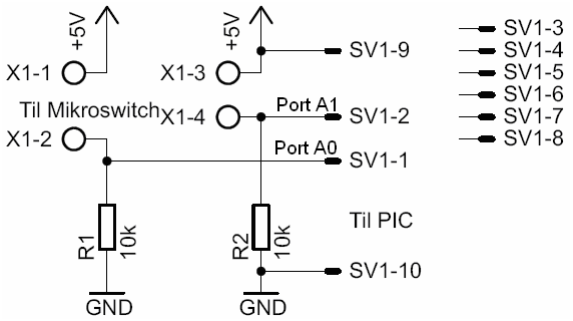
```
   end if

   -- Tænd så det display der hører til det ciffer der vises
   if skift then
      temp = temp | 0b0000_0001  -- sætter bitten i b0
   end if

   -- Nu kan vi skrive cifferet ud, samtidig med at det rigtige display
   -- bliver tændt
   port_B = temp

   -- Vent de 5 ms, så brugeren ikke ser det flimrer
   delay_5ms

   -- Gør klar til at vise det andet display
   skift = ! skift

end loop   -- end forever
```

Bilag 2 - Samlet diagram for den færdige trænings-registrerer

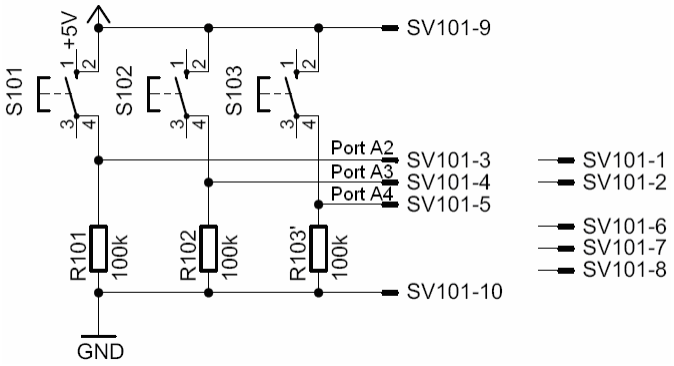
Dette bilag er lavet for at give et overblik over det elektriske system i træningsmaskinen.
PIC-boardet bliver i denne forbindelse betragtet som en færdig komponent vi kan anvende til at styre ting med, ved at den kan programmeres.



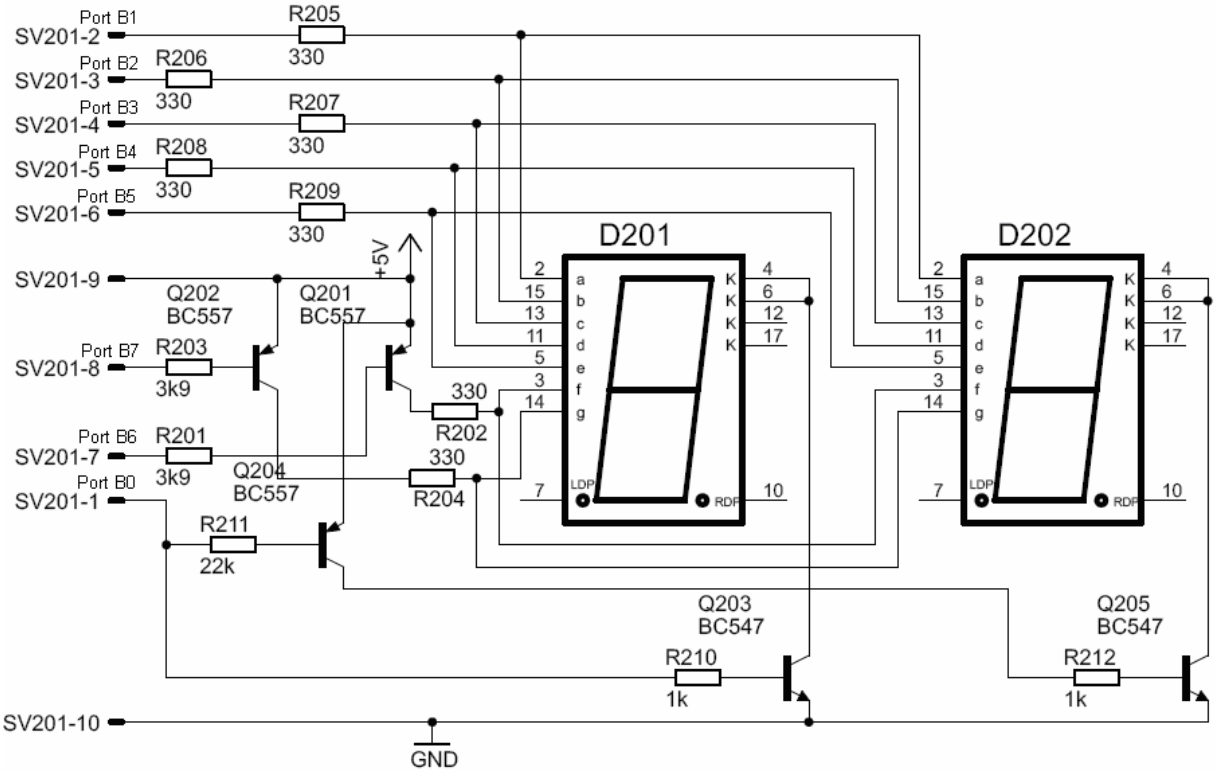
Skitse af PIC-boardet, med stikforbindelser.



Mikroswitch-print SV1 går til Port A



Bruger-kontakt-print SV101 går til Port A



Print med to 7-segment display SV201 går til Port B