# Laboration 1: Avläsning av tangentbord

## 1 Inledning

Laborationen ska bidra till en grundläggande förståelse för:

- Uppbyggnaden av ett program skrivet med assembly-kod.
- Mappning mellan anslutningarna på Arduino-kortets och mikrokontrollern.
- Hantering av in- och utenheter.
- Hur man kan skapa en drivrutin för hantering av ett tangentbord.
- Begreppet *pollning*, avseende tangentavläsning.
- Använda externa komponenter för att minimera användandet av antalet in- och utgångar.
- Vad en multiplexer är, samt hur den kan användas.

### 1.1 Utrustning

Från och med den här laborationen antas att studenterna självmant utför eventuell felsökning och kontrollmätning, utöver det som är specificerat i laborationsmanualen. Dessutom antas även att studenterna förser sig med utrustning och materiel som behövs för uppkoppling och mätning. Exempelvis behöver man alltid använda kopplingsdäck och Arduino-kort. Dessutom är det en god idé att alltid ha ett oscilloskop nära till hands.

Komponenter som behövs för denna laboration:

- telefontangentbord
- 4st $10\ k\Omega$
- 1st lysdiodkapsel (10st lysdioder i DIL-kapsel)
- 1st SIL-kapsel $8\ x\ 220\ \Omega$
- avkodare 4555
- multiplexer 4051

# 2 Beskrivning av laboration

I den här laborationen ska ni för första gången programmera mikrokontrollern som sitter på Arduino-kortet. Denna laboration, samt de två kommande laborationerna, går ut på att ni ska skriva program bestående av assembly-instruktioner. Utöver programmeringen ska ni även ansluta extern elektronik, dels för att göra systemet mer intressant, dels för att fördjupa kunskaperna från föregående delkurs.
För att klara av laborationen måste ni, precis som i tidigare laborationer, förbereda laborationen genom att läsa instruktionerna noggrant, studera teori och datablad, etc.

Laborationen består av följande moment:

1. Grundläggande programkod
2. Inkoppling av tangentbord och lysdiodkapsel
3. Avläsning av ett fåtal knappar
4. Inkoppling av avkodare och multiplexer
5. Komplett tangentbordsdrivrutin

## 2.1 Syfte och mål

Laborationen går ut på att ni ska programmera mikrokontrollern för att hantera avläsning av ett numeriskt tangentbord (”telefontangentbord”). Till en början ska ni endast hantera ett fåtal tangenter. Därefter ska ni skriva en algoritm som hanterar hela tangentbordet. Algoritmen utgör vad man även kallar för en *drivrutin*. Denna drivrutin kommer sedan utgöra en grund för kommande laborationer.

## 2.2 Examination

### 2.2.1 Inlämning av rapport och programkod

Utöver en rapport ska ni även lämna in programkod. Instruktioner för detta är beskrivet på inlämningssidan för denna laboration (på It’s Learning), 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 6-1. 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.

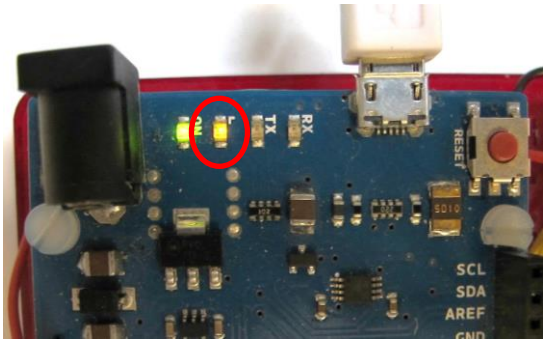
# 3 Moment 1: Grundläggande programkod

## 3.1 Överföring av grundläggande programkod

**Uppgift 3.1.1**

1. Ladda ner **lab1.zip** från It's Learning och packa upp filerna på ett lämpligt ställe. I denna ZIP-fil finns en mapp med projektfiler och programkod.
2. Öppna sedan projektet i Atmel Studio.
3. Assemblera programmet och för över det till mikrokontrollern enligt instruktionerna i dokumentet *Introduktion till Atmel Studio* (finns på It's Learning).

Om allt fungerar som det ska, så bör den inbyggda lysdioden på Arduino-kortet lysa nu (se Figur 3-1). Lysdioden styrs genom att konfigurera PC7 (PORTC, bit 7) som en utgång, samt genom att sätta hög nivå på denna utgång. Anslutningen PC7 är även sammankopplad med Arduino-kortets pinne 13. Denna information är dock inget man lär sig utantill. Därför behövs en tabell som anger *mappningen*, d.v.s. relationen mellan de interna och externa anslutningarna (finns på It's Learning eller på produktens hemsida).




Figur 3-1: Inbyggd lysdiod (markerad med röd ring)

**Uppgift 3.1.2**

Mät på pinne 13, med ett oscilloskop. Använd DC-läget för den kanal ni har valt. Verifiera att ni kan mäta en likspänning på $5\ V$. *OBS! Behåll uppkopplingen med oscilloskopet!*

## 3.2 Analys av grundläggande programkod

När man lär sig att programmera behöver man inte bara skriva kod, utan också läsa och förstå kod som redan är skriven. Till att börja med ska ni analysera den grundläggande programkoden som redan är given. Koden finns i filen **lab1.asm**.

**Uppgift 3.2.1 (redovisas i rapport)**

De första raderna, i blocket som kallas för "Start of program" i kommentarerna, ser ut så här:

```
.CSEG
.ORG RESET
RJMP init
.ORG PM_START
```

Följande frågor ska besvaras:

- Vad har **`.CSEG`** för innebörd?
- Vad händer om man skriver **`.ORG`**? Vad menas det som skrivs till höger?
- Vad sker när den tredje raden (**`RJMP init`**) exekveras?

**Uppgift 3.2.2 (redovisas i rapport)**
Nästa kodblock inleds med att konfigurera *stackpekaren*, men detta behandlas i en kommande laboration. Efter detta anropas en subrutin (**init_pins**) som hanterar konfigurering (även kallad initialisering) av in- och utgångar. Koden för subrutinen ser ut så här:

```
init_pins:

          LDI R16, 0b10000000
          OUT DDRC, R16
          RET
```

Följande frågor ska besvaras:

- Vad har instruktionen **LDI** för syfte?
- Vad har instruktionen **OUT** för syfte?
- Vad har instruktionen **RET** för syfte?
- Vad är **R16**?
- Vad är **DDRC**?

**Uppgift 3.2.3 (redovisas i rapport)**
Beskriv (steg för steg) vad som händer när koden körs:

```
LDI R16, 0b10000000
OUT DDRC, R16
```

**Uppgift 3.2.4 (redovisas i programkod)**
Det binära talet (0b10000000) i koden ovan kan istället skrivas i hexadecimal form. Då blir det lättare att läsa talet. Dessutom är det inte lika lätt att råka missa en nolla, vilket skulle ställa till med problem. Konvertera det binära talet till ett hexadecimalt tal. Ändra detta i programkoden.

**Uppgift 3.2.5 (redovisas i rapport)**
Huvuddelen av programmet exekveras som en oändlig slinga (*infinite loop*). Det enda som utförs, förutom hoppet tillbaka till ”main”, är exekveringen av:

```
SBI PORTC, 7
```

Vad sker när ovanstående exekveras?

**Uppgift 3.2.6 (redovisas i rapport)**
I den utkommenterade koden finns en liknande instruktion:

```
CBI PORTC, 7
```

Vad sker om detta exekveras?

**Uppgift 3.2.7 (redovisas i rapport)**
I den utkommenterade koden finns även ett antal förekomster av denna instruktion:

```
NOP
```

Vad sker om denna instruktion exekveras?

## 3.3 Mätning och beräkning av exekveringstid

### 3.3.1 Mätning av exekveringstid

**Uppgift 3.3.1.1**

Ta bort kommentarerna (/* och */) kring koden i huvuddelen av programmet. Assemblera och för över programmet. Nu bör du kunna mäta upp en fyrkantsvåg på oscilloskopet.

**Uppgift 3.3.1.2 (redovisas i rapport)**

Fotografera oscilloskopsbilden och bifoga den i rapporten.

### 3.3.2 Beräkning av exekveringstid

Fyrkantsvågens periodtid bör bli 1 $\mu s$. Detta kan även beräknas genom att ta reda på hur många klockcykler varje enskild instruktion behöver för att exekveras. Antalet klockcykler är angivna i kommentarerna. Det tar alltså totalt 16 klockcykler för att exekvera koden innan den börjar om igen. Total exekveringstid får man sedan genom att multiplicera antalet klockcykler med periodtiden för processorns klockfrekvens:

$$f_{osc} = 16\,MHz \;\Rightarrow\; T_{osc} = \frac{1}{16}\,\mu s$$

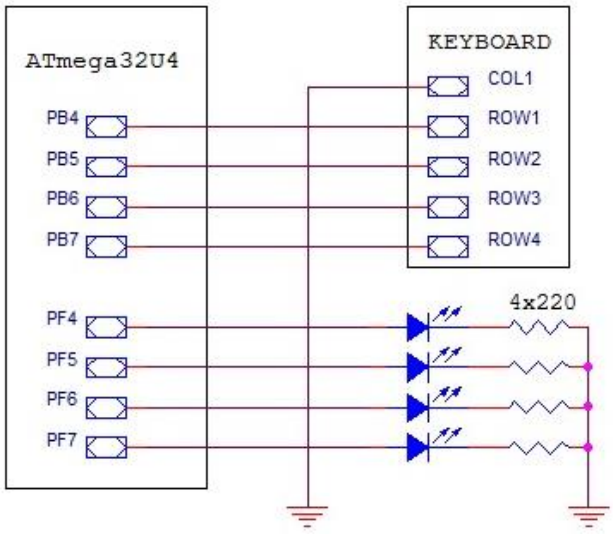
$$16 \cdot T_{osc} = 1\,\mu s$$

Beräkning och mätning av exekveringstider ska vi återkomma till i kommande laborationer. Att sätta en utgång till hög nivå eller låg nivå kan också vara ett sätt att felsöka ett program, eftersom det ibland kan vara svårt att få någon uppfattning om koden exekveras på rätt sätt.

# 4 Moment 2: Inkoppling av tangentbord och lysdiodkapsel

## 4.1 Uppkoppling av krets

Koppla in tangentbordet, lysdioder och resistorkapseln till Arduino-kortet enligt Figur 4-1. Tänk på att ha koll på mappningen mellan interna och externa anslutningar.




Figur 4-1: Kretsschema för moment 2 och 3

# 5 Moment 3: Avläsning av ett fåtal knappar

## 5.1 Beskrivning

Detta moment går ut på att ni ska programmera en hantering av ett fåtal tangenter (1, 4, 7, *). Anslutningarna som tangentbordets ROW1 – ROW4 är kopplade till, ska konfigureras som ingångar med aktiverade pullup-motstånd. När någon av tangenterna trycks ner ska detta noteras genom att en låg nivå detekteras på motsvarande ingång.
Övriga anslutningar ska konfigureras som utgångar.

När man trycker på någon av tangenterna så ska detta indikeras genom att tända lysdioder enligt följande mönster:

| Ingångar | | | | Utgångar | | | |
|---|---|---|---|---|---|---|---|
| **PB4** | **PB5** | **PB6** | **PB7** | **PF4** | **PF5** | **PF6** | **PF7** |
| 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 |
| 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 |
| 1 | 0 | 1 | 1 | 0 | 1 | 0 | 0 |
| 1 | 1 | 0 | 1 | 0 | 0 | 1 | 0 |
| 1 | 1 | 1 | 0 | 0 | 0 | 0 | 1 |

Tabell 5-1: Sanningstabell för sambandet mellan nivåerna på in- och utgångarna

## 5.2 Konfigurering av in- och utgångar

**Uppgift 5.2.1**

Ta bort koden som styr utgången PC7 från huvuddelen av programmet. Koden ska se ut så här:

```
main:
          RJMP main
```

**Uppgift 5.2.2 (redovisas i programkod)**

Komplettera subrutinen **init_pins** med konfigureringen av PB4-PB7 och PF4-PF7. Använd **R16** för att lagra värdena ni ska ange för I/O-registren. Kommentera koden ni skriver (gäller även befintlig kod).
*Tips! Ett enkelt sätt att testa utgångarna PF4-PF7 skulle kunna vara att sätta samtliga utgångar till hög nivå (i huvuddelen av programmet). Då ser ni om både krets och programkod är korrekt!*

**Uppgift 5.2.3 (redovisas i programkod)**

Som ni ser används **R16** på flera olika ställen. Vissa instruktioner accepterar inte konstanta värden och då måste man mellanlagra temporär data i ett allmänt register. Eftersom detta sker ofta så är det god idé att reservera **R16** för detta ändamål. Dessutom är det bra göra koden lite tydligare. Detta gör ni genom att:

- Definiera benämningen **TEMP** att vara detsamma som **R16**. Skriv detta i början av programmet, i blocket som benämns som ”Definitions of registers”.
- Ersätt alla förekomster av **R16** med **TEMP**.

## 5.3 Avläsning av tangenter

**Uppgift 5.3.1**
Testa om konfigureringen av in- och utgångarna fungerar, genom att programmera följande i ”main-slingan”:

1. Läs in innehållet av **PINB** till **TEMP**.
2. Kopiera innehållet av **TEMP** till **R24**.
3. Sätt **PORTF** till värdet av **R24**.
4. Skapa en ”paus” bestående av 2 klockcykler.

Assemblera och för över programmet. Nu bör samtliga lysdioder tändas om ingen tangent är nertryckt. De tangenter som trycks ner kommer att representeras med en släckt lysdiod.

**Uppgift 5.3.2 (redovisas i rapport)**
Eftersom Tabell 5-1 specificerar att nertryckta knappar ska indikeras med en tänd lysdiod, medan andra är släckta, så måste ni invertera bitarna. Redovisa koden för:

1. Läs in innehållet av **PINB** till **TEMP**.
2. Invertera innehållet i **TEMP**.
3. Kopiera innehållet av **TEMP** till **R24**.
4. Sätt **PORTF** till värdet av **R24**.
5. Skapa en ”paus” bestående av 2 klockcykler.

**Uppgift 5.3.3 (redovisas i programkod)**
Definiera **R24** till att ha ”namnet” **RVAL**. Ersätt sedan alla förekomster av **R24** med denna benämning. **RVAL** står för ”return value”. I detta fall syftar detta på ett returvärde från en subrutin, som ni ska skriva i nästa uppgift.

**Uppgift 5.3.4**
Skriv en subrutin som läser av tangenterna. Kalla subrutinen för **read_keyboard**. Subrutinen ska utföra steg 1-3 från uppgift 5.3.2 (flytta koden till subrutinen).

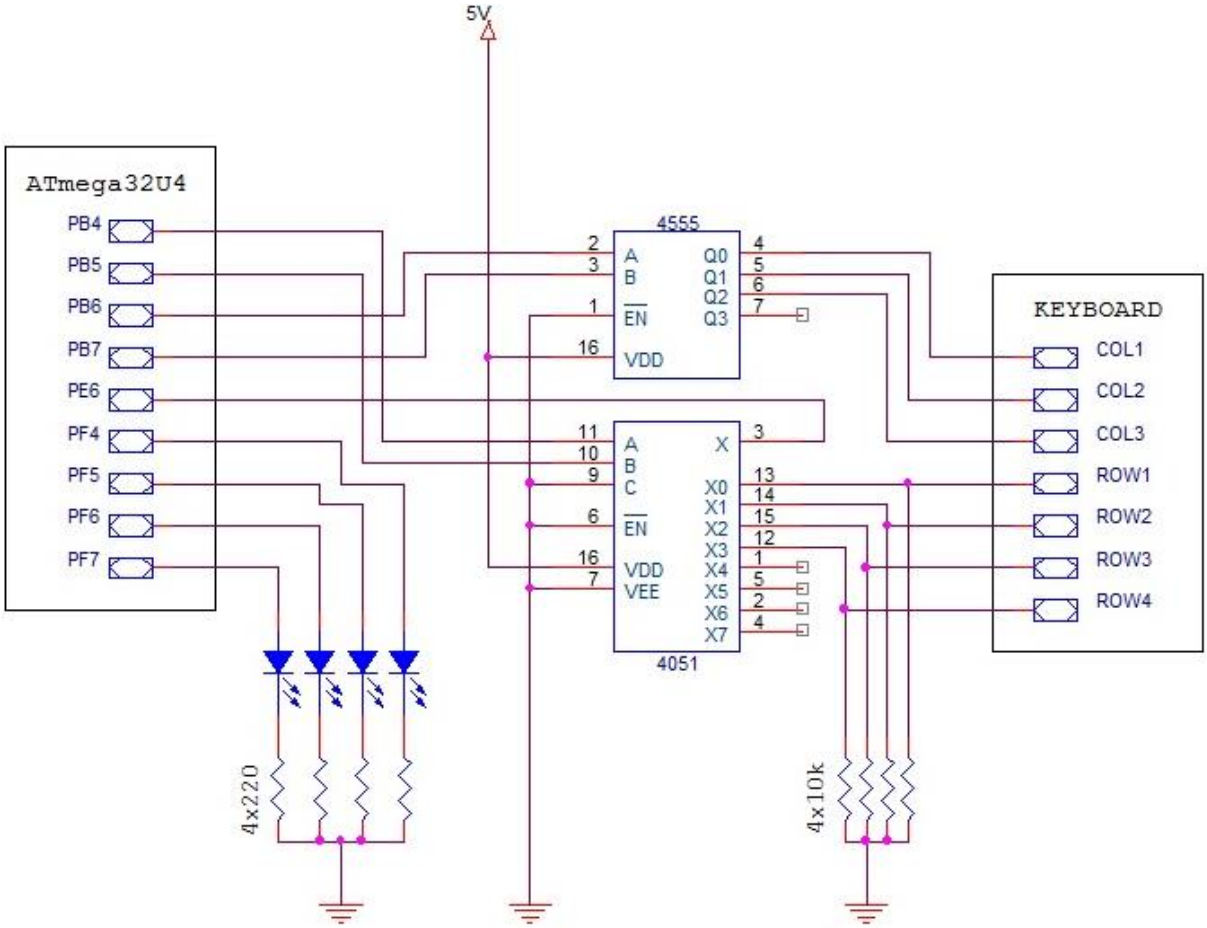
**Uppgift 5.3.5**
Ändra koden i ”main-slingan” så att **read_keyboard** anropas innan **PORTF** tilldelas värdet av **RVAL**. Resultatet borde vara detsamma som innan, d.v.s. att lysdioderna indikerar tangenttryckningarna.

# 6 Moment 4: Inkoppling av avkodare och multiplexer

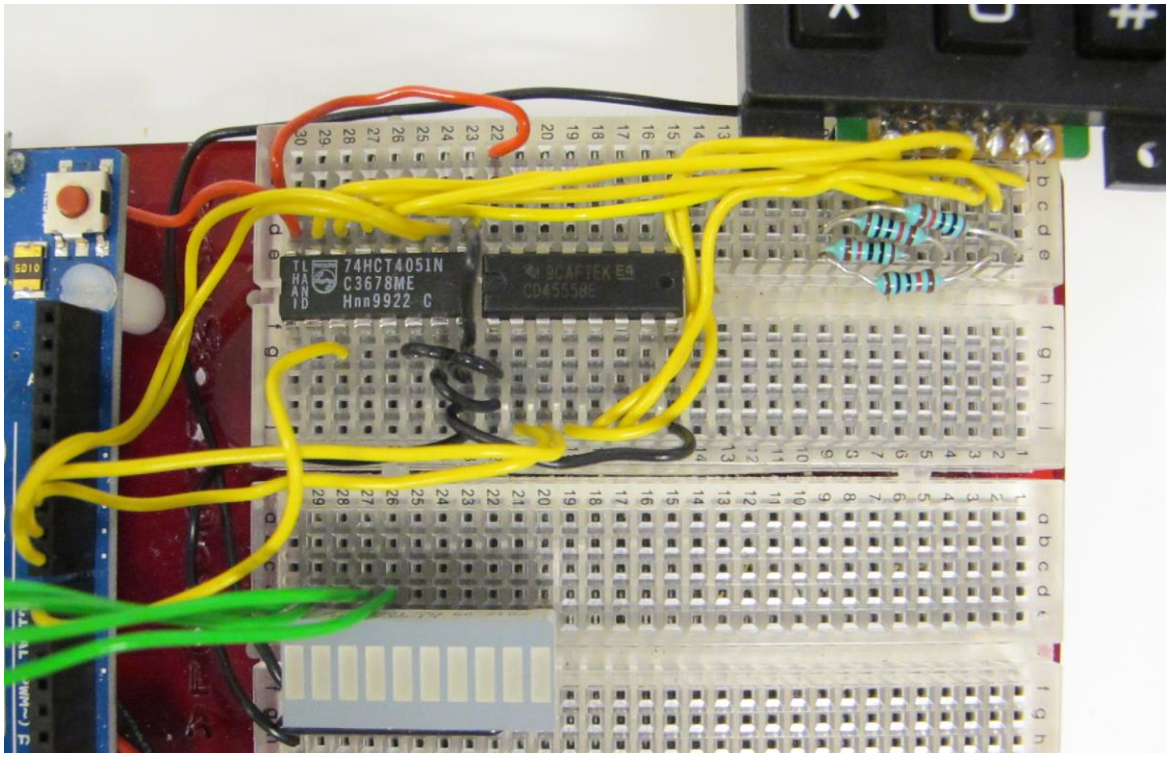
## 6.1 Uppkoppling av krets

Ersätt den befintliga kretsen genom att koppla enligt Figur 6-1. Eftersom utrymmet på kopplingsdäcket är väldigt begränsat så bör ni hushålla med utrymmet. Placera komponenterna enligt Figur 6-2.
*OBS! Det är väldigt viktigt att ni följer dessa instruktioner! Övrig yta på kopplingsdäcket kommer att behövas i kommande laborationer!*

Med denna krets blir det möjligt att hantera hela tangentbordet med endast 5 anslutningar, istället för 7. Kretsen kommer dessutom att utökas i en av de kommande laborationerna.




Figur 6-1: Krets för avläsning av tangentbordet




Figur 6-2: Placering av komponenter (multiplexern till vänster om avkodaren)

## 6.2 Detektera enskild tangenttryckning

I den här tillämpningen fokuserar vi på att urskilja enstaka tangenttryckningar. Kombination av flera tangenter samtidigt utesluts. För att notera om någon av tangenterna har tryckts ner, tillämpas följande algoritm:

1. Aktivera kolumn 1, se till att övriga kolumner är inaktiverade.
2. Läs av rad 1.
3. Om tangent är nertryckt, notera detta och avbryt algoritm.
4. Läs av rad 2.
5. Om tangent är nertryckt, notera detta och avbryt algoritm.
6. Läs av rad 3.
7. Om tangent är nertryckt, notera detta och avbryt algoritm.
8. Läs av rad 4.
9. Om tangent är nertryckt, notera detta och avbryt algoritm.
10. Aktivera kolumn 2, se till att övriga kolumner är inaktiverade.

……

19. Aktivera kolumn 3, se till att övriga kolumner är inaktiverade.

……

## 6.3 Avkodarens funktion

Avkodarens funktion i denna krets är att aktivera respektive inaktivera tangentbordets kolumner. Detta underlättas eftersom endast en utgång kan ha hög nivå (vara aktiv), medan de andra är har låga nivåer. Aktuell kolumn aktiveras genom att sätta logiska nivåer på ingångarna A och B. I programkoden behöver man endast specificera två bitar för att aktivera aktuell kolumn. Dessa bitar kan betraktas som en adress.

## 6.4 Multiplexerns funktion

Multiplexern ska användas till att läsa av aktuell rad, som anges genom att sätta logiska nivåer på pinne A och B. Dessa ingångar specificerar en adress, som styr vilken av pinnarna X0-X3 som ska vara ihopkopplad med X (gemensam utgång, i detta fall). X-utgången används för att detektera en eventuell knapptryckning, vilket indikeras med en logisk etta.

**Uppgift 6.4.1 (redovisas i rapport)**
Beskriv med egna ord hur multiplexern fungerar. Redogör inte bara för hur den fungerar i denna tillämpning, utan i ett generellt sammanhang.

**Uppgift 6.4.2 (redovisas i rapport)**
Varför behöver man ansluta multiplexerns pinne 9 (ingång C) till jord?

**Uppgift 6.4.3 (redovisas i rapport)**
Hur lång tid tar det för en signal att färdas från ingång till utgång (Signal Input to Output)?
*Ange maximalt värde! Kolla upp detta i databladet!*

**Uppgift 6.4.4 (redovisas i rapport)**
Hur lång tid krävs för omställning av adress till att motsvarande ingång kan avläsas på den gemensamma utgången (Address-to-Signal OUT)?
*Ange maximalt värde! Kolla upp detta i databladet!*

# 7 Moment 5: Komplett tangentbordsdrivrutin

## 7.1 Beskrivning

Syftet med drivrutinen är att detektera om en tangent har tryckts ner, samt vilken tangent det är. Man aktiverar en kolumn i taget och läser av varje rad, för att avgöra om vilken logisk nivå som förekommer på PE6. Således ska denna anslutning konfigureras som en ingång (ej pullup). Övriga anslutningar ska konfigureras som utgångar.

Metoden för att regelbundet utföra en kontroll om en tangent har blivit nertryckt kallas för *pollning*.

Ni ska stega igenom samtliga kolumner och rader (totalt 12 steg) genom att ange ett 4-bitars tal (adress) på utgångarna PB4-PB7. PB4 och PB5 utgör de minst signifikanta bitarna, som representerar raderna. PB6 och PB7 utgör de mest signifikanta bitarna, som representerar kolumnerna. Tabell 7-1 anger mappningen mellan tangenterna och de möjliga kombinationerna adressen.

| Tangent | COL 1 (PB7) | COL 0 (PB6) | ROW 1 (PB5) | ROW 0 (PB4) |
|---|---|---|---|---|
| 1 | 0 | 0 | 0 | 0 |
| 4 | 0 | 0 | 0 | 1 |
| 7 | 0 | 0 | 1 | 0 |
| * | 0 | 0 | 1 | 1 |
| 2 | 0 | 1 | 0 | 0 |
| 5 | 0 | 1 | 0 | 1 |
| 8 | 0 | 1 | 1 | 0 |
| 0 | 0 | 1 | 1 | 1 |
| 3 | 1 | 0 | 0 | 0 |
| 6 | 1 | 0 | 0 | 1 |
| 9 | 1 | 0 | 1 | 0 |
| # | 1 | 0 | 1 | 1 |

Tabell 7-1: Mappning mellan 4-bitarskombinationen och tangenterna

Precis som i föregående moment ska lysdioderna representera aktuell tangent.

## 7.2 Utökning av subrutinen för tangentbordsavläsning

Nu ska ni utöka subrutinen **read_keyboard**. Subrutinen ska returnera aktuell 4-bitars adress, om en tangent har blivit nertryckt (se Tabell 7-1). Om ingen tangent trycks ner ska det hexadecimala värdet 0x0F returneras.
Uppgift 7.2.5 går ut på att använda färdig kod för subrutinen. Den är dock inte komplett. Det går också bra (för den som vill ha lite mer utmaning) att försöka skriva koden på egen hand. Men innan ni gör detta bör ni lösa andra problem först. Därför rekommenderas ni att följa instruktionerna i rätt ordning. Framför allt är det mycket viktigt att ni förstår vad ni gör samt varför ni gör det!

*OBS! Glöm inte att konfigurera in- och utgångar!*

**Uppgift 7.2.1 (redovisas i programkod)**
För att göra koden mer läsbar ska ni definiera en benämning för värdet 0x0F. Kalla den för **NO_KEY**.

**Uppgift 7.2.2**
Modifiera **read_keyboard** så att subrutinen ser ut enligt:

```
read_keyboard:

          LDI R18, NO_KEY
return_key_val:

          MOV RVAL, R18
          RET
```

**Uppgift 7.2.3 (redovisas i rapport)**
Assemblera och för över programmet. Förmodligen kommer inte en enda av lysdioderna att lysa. En förklaring till problemet ges i följande illustration:

| Register | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R24 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 |
| PORTF | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 |

Tabell 7-2: Illustration av registerinnehåll

För stunden får **PORTF** ett värde enligt tabellen ovan. Men eftersom lysdioderna är anslutna till PF4-PF7 så krävs det att man skiftar bitarna i registret **R24** (**RVAL**) till vänster, innan värdet överförs till **PORTF.** Vilken instruktion ska användas för detta?

**Uppgift 7.2.4 (redovisas i programkod)**
Programmera bit-skiftningen och testkör programmet. Nu bör samtliga lysdioder vara tända. Vid det här laget bör följande utföras i ”main-slingan”:

1. Anropa subrutinen **read_keyboard**.
2. Bit-skifta innehållet i **RVAL** ett visst antal gånger till vänster.
3. Överför innehållet i **RVAL** till **PORTF**.
4. Skapa en ”paus” bestående av 2 klockcykler.
5. Gör ett hopp till början av ”main-slingan”.

**Uppgift 7.2.5 (redovisas i programkod)**
På It’s Learning (i mappen ”Programkod”, för aktuell labb) kan ni hitta filen **read_keyboard.txt**. I filen hittar ni färdig kod för subrutinen **read_keyboard**. Kopiera koden från filen och ersätt den befintliga subrutinen med detta. Testkör programmet genom att kontrollera om samtliga tangenter avkodas på ett korrekt sätt.

**Uppgift 7.2.6 (redovisas i rapport)**
Som ni säkert lägger märke till så fungerar inte hanteringen av vissa tangenter. Detta beror på att koden i **read_keyboard** inte tar hänsyn till uppgifterna som ni fick fram i uppgift 6.4.3 och 6.4.4. För att lösa detta måste ni lägga till några fler NOP-instruktioner.
Hur många NOP-instruktioner behöver man inkludera totalt (inkl. befintliga)? Redovisa även beräkningar och hänvisa till uppgifterna ni fick från databladet!

**Uppgift 7.2.7 (redovisas i programkod)**
Lägg till ytterligare NOP-instruktioner, så att det totala antalet överensstämmer med det ni kom fram till i föregående uppgift.

**Uppgift 7.2.8 (redovisas i programkod)**
Testkör programmet! Om det fungerar som det ska är det dags att avsluta genom att:

- Se till att ni förstår vad programmet gör, samt hur mikrokontrollern samverkar med den externa kretsen.
- 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!
- I kommentarsblocket i början av filen ska ni skriva in era namn och aktuellt datum. Dessutom ska ni med egna ord beskriva vad programmet gör.

När detta är klart kan ni redovisa för labhandledare!

**Uppgift 7.2.9 (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?