Datorteknik

— — —

Exempeluppgifter i Laborativ Examination

Michael Josefsson

Version 0.1

Inledning

Nedan följer förslag på några representativa demonstrationslaxar. Uppgifterna är konstruerade så att de ska innehålla *sekvens*, *iteration* och *selektion* enligt JSP. Dessutom bör de komma ihåg någon tidigare händelse. De ”skarpa” LAX-arna är av samma komplexitetsgrad, har samma eller liknande hårdvara men är *inte* dessa uppgifter.

Inlärningen sker i den kreativa processen i hjärnan när du själv konstruerar en lösning. Titta inte på lösningsförslagen om du inte har ett eget förslag att jämföra med!

Namn	In-enhet	Ut-enhet
`LAX-DEMO 1`	Hextangentbord	2 x 7-segmentsdisplay
`LAX-DEMO 2`	2 x Tryckknapp	2 x 7-segmentsdisplay
`LAX-DEMO 4`	Hextangentbord	2 x 7-segmentsdisplay
`LAX-DEMO 5`	Hextangentbord	Lysdioddisplay

Träna så att du kan ta fram en fungerande lösning på 30–45 minuter.

Datorteknik
LAX-DEMO 1

Tidsomfattning: 90 minuter
inkl redovisning

Uppgift I labsatsen finns ett hexadecimalt tangentbord. Det ger ut fyra bitar data (D,C,B,A) vid nedtryckt tangent men även en *strobe*-signal som är hög så länge *någon* tangent är nedtryckt.

Din uppgift är att läsa av det hexadecimala tangentbordet, en siffra åt gången, och presentera dess decimala motsvarighet på två sjusegmentsdisplayer, tiotalssiffran till vänster och entalssiffran till höger. Displayerna kan visa de hexadecimala siffrorna `0-F`, men ska här bara visa `0-9`. Senaste decimaltal ska kontinuerligt visas tills en ny siffra trycks ned på tangentbordet.

Tangentbordet kan inte ge flera utsignaler även om flera tangenter trycks ned samtidigt, följaktligen behöver inte programmet ta någon hänsyn till detta fall.

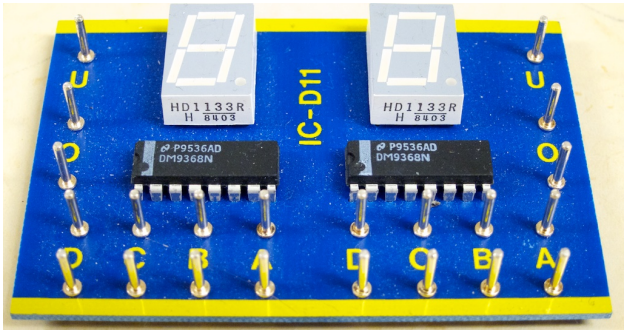
Obs! Hårdvaruinitieringen **måste** utföras som en subrutin.

Hårdvara

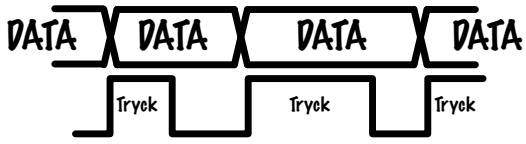
- labsats
- sjusegmentsdisplay
- hexadecimalt tangentbord

Funktionskontroll och examination Funktionen prioriteras! Någon kodgranskning, utöver kontroll av att hårdvaruinitieringen är utförd som subrutin, kommer inte ske. Funktionen kontrolleras genom upprepade tryckningar på det hexadecimala tangentbordet och kontroll på sjusegmentsdisplayen. Nöjaktig funktion resulterar i godkänd LAX.

Sjusegmentsdisplayen Matningsspänning är U (5 V) och 0 (0 V). Indata till respektive segment är de fyra bitarna D, C, B, A.



Tangentbordet med kopplingsplatta Matningsspänning +5 V påförs pinne 1, 0 V pinne 8. Utdata, fyra bitar, återfinns på pinnarna 3, 5, 7 och 9. *Strobe*-signalen är pinne 2. Stroben är hög så länge någon knapp är nedtryckt. Datat kommer samtidigt och ligger kvar tills nästa knappnedtryckning:



Datorteknik
LAX-DEMO 2

Tidsomfattning: 90 minuter
inkl redovisning

Uppgift I labsatsen finns två tryckknappar. Dessa ger en positiv och en negativ flank som utsignal från var sina stift, för respektive tryckknapp.

Din uppgift är att räkna antalet nedtryckningar av den vänstra tryckknappen. När den högra tryckknappen trycks ned ska detta antal visas på en sjusegmentsdisplay, och fortsätta att visas även efter att den högra tryckknappen släppts upp. Därefter ska man kunna börja om med att räkna nedtryckningar av den vänstra tryckknappen. Displayerna kan visa de hexadecimala siffrorna `0-F`. Trycker man mer än 15 gånger på den vänstra tryckknappen så ska displayen visa `F`, dvs 15_{10} hexadecimalt.

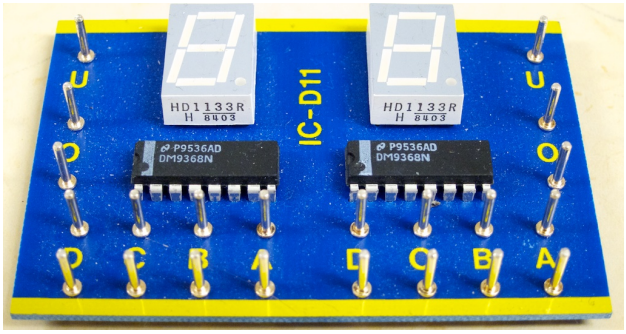
Obs! Hårdvaruinitieringen **måste** utföras som en subrutin.

Hårdvara

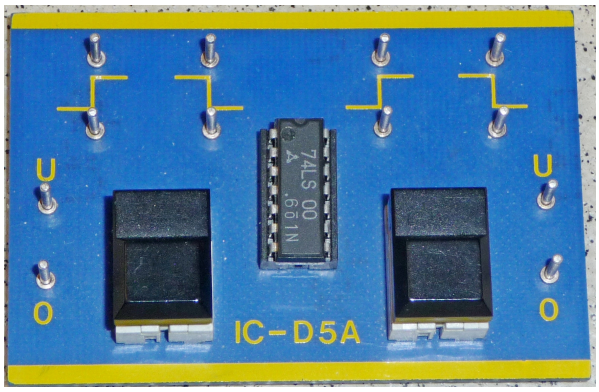
- sjusegmentsdisplay
- tryckknappar

Funktionskontroll och examination Funktionen prioriteras! Någon kodgranskning, utöver kontroll av att hårdvaruinitieringen är utförd som subrutin, kommer inte ske. Funktionen kontrolleras genom upprepade tryckningar på tryckknapparna och kontroll på sjusegmentsdisplayen. Nöjaktig funktion resulterar i godkänd LAX.

Sjusegmentsdisplayen Matningsspänning är U (5 V) och 0 (0 V). Indata till respektive segment är de fyra bitarna D, C, B, A.



Tryckknappar Matningsspänning är U (5 V) och 0 (0 V). Varje knapp ger en positiv och en negativ flank som utsignal från var sina stift då knappen trycks ned. Utsignalen återgår sedan när knappen släpps upp.



Datorteknik
LAX-DEMO 4

Tidsomfattning: 90 minuter
inkl redovisning

Uppgift: I labsatsen finns ett hexadecimalt tangentbord. Det ger ut fyra bitar data (D,C,B,A) vid nedtryckt tangent men även en *strobe*-signal som är hög så länge *någon* tangent är nedtryckt.

Din uppgift är att visa nedtryckta decimala tangentvärden från tangentbordet på vänster alternativt höger indikator på sjusegmentsdisplayen. Med tangenten F ska man kunna växla (toggla) indikator så att efterföljande tangentvärden hamnar till vänster ifall höger indikator tidigare användes, och vice versa. Gamla värden ska dock alltid ligga kvar tills dom ersätts av nya från tangentbordet. Tangenterna A, B, C, D och E ska inte ha någon funktion eller påverkan.

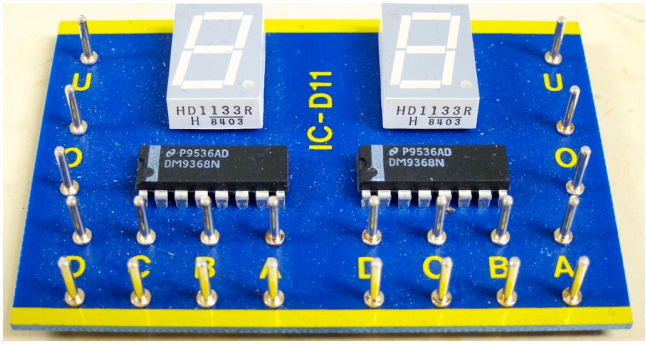
Obs! Hårdvaruinitieringen **måste** utföras som en subrutin.

Hårdvara

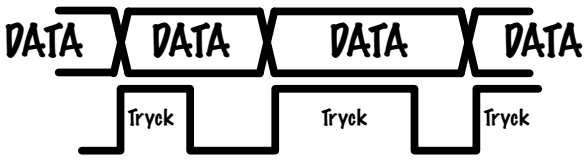
- labsats
- sjusegmentsdisplay
- hexadecimalt tangentbord

Funktionskontroll och examination Funktionen prioriteras! Någon kodgranskning, utöver kontroll av att hårdvaruinitieringen är utförd som subrutin, kommer inte ske. Funktionen kontrolleras genom upprepade tryckningar på det hexadecimala tangentbordet och kontroll på sjusegmentsdisplayen. Nöjaktig funktion resulterar i godkänd LAX.

Sjusegmentsdisplay Displayen har två sjusegments indikatorer. Matningsspänning är U (5 V) och 0 (0 V). Indata till respektive segment är de fyra bitarna D, C, B, A.



Tangentbordet med kopplingsplatta Matningsspänning +5 V påförs pinne 1, 0 V pinne 8. Utdata, fyra bitar, återfinns på pinnarna 3, 5, 7 och 9. *Strobe*-signalen är pinne 2. Stroben är hög så länge någon knapp är nedtryckt. Datat kommer samtidigt och ligger kvar tills nästa knappnedtryckning:



Datorteknik
LAX-DEMO 5

Tidsomfattning: 90 minuter
inkl redovisning

Uppgift: I labsatsen finns ett hexadecimalt tangentbord. Det ger ut fyra bitar data (D,C,B,A) vid nedtryckt tangent men även en *strobe*-signal som är hög så länge *någon* tangent är nedtryckt.

Din uppgift är att visa ett nedtryckt tangentvärde binärt på lysdioddisplayens fyra mest vänstra lysdioder. Samma tangentvärde ska även visas på dom fyra mest högra lysdioderna, inverterat (obs **ej** speglat) eller normalt. Inverterad eller normal visning för dom fyra mest högra lysdioderna togglas med tangent 0, som också ska visas på samma sätt som övriga tangentvärden. Dvs, om normal visning råder ska lysdioderna 7–4 och lysdioderna 3–0 visa samma sak, men om inverterad visning råder ska lysdioderna 3–0 visa det inverterade värdet av lysdioderna 7–4, vilka alltså alltid ska visa tangentvärdet.

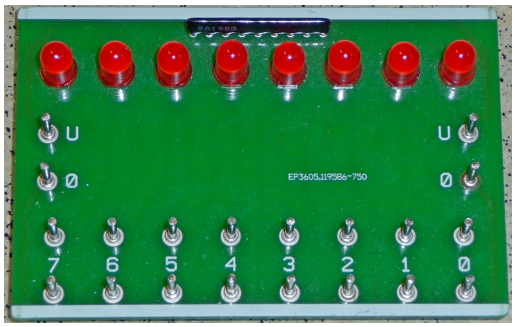
Obs! Hårdvaruinitieringen **måste** utföras som en subrutin.

Hårdvara

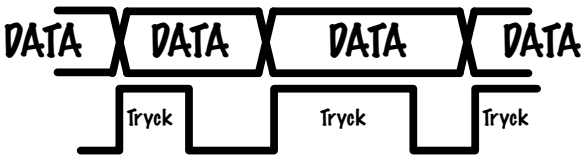
- labsats
- lysdioddisplay
- hexadecimalt tangentbord

Funktionskontroll och examination Funktionen prioriteras! Någon kodgranskning, utöver kontroll av att hårdvaruinitieringen är utförd som subrutin, kommer inte ske. Funktionen kontrolleras genom upprepade tryckningar på det hexadecimala tangentbordet och kontroll på lysdioddisplayen. Nöjaktig funktion resulterar i godkänd LAX.

Lysdioddisplay Matningsspänning är U (5 V) och 0 (0 V). Lysdioderna styrs individuellt av ingångarna 7 till 0 nederst på plattan.



Tangentbordet med kopplingsplatta Matningsspänning +5 V påförs pinne 1, 0 V pinne 8. Utdata, fyra bitar, återfinns på pinnarna 3, 5, 7 och 9. *Strobe*-signalen är pinne 2. Stroben är hög så länge någon knapp är nedtryckt. Datat kommer samtidigt och ligger kvar tills nästa knappnedtryckning:



Lösningsförslag

Instruktioner

Inlärningen sker i den kreativa processen i hjärnan när du själv konstruerar en lösning. Titta inte på förslagen nedan om du inte har ett eget förslag att jämföra med!

Man lär sig koda bättre genom att läsa mycket kod. Jämför förslagen med din egen lösning och förbättra dem båda. Vad kan göras för att få mer lättläst kod? Är strukturen den bästa? Skulle koden tjäna på globala konstanter? Variabelnamn? Finns det alternativa lösningsmetoder?

Vid LAX-tillfället sker visserligen ingen kodgranskning men man tjänar ändå på att ha ett strukturerat angreppssätt med subrutiner och bra namngivning av *labels*.

Namn	In-enhet	Ut-enhet
`LaxDemo1.asm`	Hextangentbord	2 x 7-segmentsdisplay
`LaxDemo2.asm`	2 x Tryckknapp	2 x 7-segmentsdisplay
`LaxDemo2a.asm`	Dalia	Dalia
`LaxDemo4.asm`	Hextangentbord	2 x 7-segmentsdisplay
`LaxDemo5.asm`	Hextangentbord	Lysdioddisplay
`LaxDemo5mini.asm`	Hextangentbord	Lysdioddisplay

`LabDemo5mini.asm` är ett försök att konstruera en resurssnål lösning. Här har man eliminerat kod genom att bland annat koppla om hårdvaran och ta bort — i det här fallet — onödiga instruktioner. Lösningen tillhör kategorin ”ful kod”.

```
        /* LaxDemo1.asm
         * Compiles to 50 bytes (42 if call etc)
         */

COLD:
        ldi     r16,HIGH(RAMEND)
        out     SPH,r16
        ldi     r16,LOW(RAMEND)
        out     SPL,r16
        call    HW_INIT
MAIN:
        sbis    PINA,4          ; wait for strobe/key press
        jmp     MAIN
        in      r16,PINA        ; read key
        andi    r16,$0F
        cpi     r16,10
        brmi    PRINT
        subi    r16,$FA
PRINT:
        out     PORTB,r16
WAIT:
        sbic    PINA,4          ; wait for key release
        jmp     WAIT
        jmp     MAIN            ; process next digit

        ; --- Config I/O
HW_INIT:
        ldi     r16,0
        out     DDRA,r16        ; PORTA<4> strobe, PORTA<3-0> data
        dec     r16
        out     DDRB,r16        ; PORTB<7-0> display
        ret
```

```
        /* LaxDemo2.asm
         * Compiles to 70 bytes (62 if call etc)
         */

COLD:
        ldi     r16,HIGH(RAMEND)
        out     SPH,r16
        ldi     r16,LOW(RAMEND)
        out     SPL,r16
        call    HW_INIT
WARM:
        call    GETKEYS
        sbrc    r16,1
        inc     r17             ; left pressed
        sbrc    r16,0
        call    SHOWIT          ; right pressed
        jmp     WARM

SHOWIT:
        cpi     r17,$0F
        brmi    SHOWIT2
        ldi     r17,$0F
SHOWIT2:
        out     PORTB,r17
        ; clr   r17             ; show progress
        ret

GETKEYS:
        in      r16,PIND
        andi    r16,$03
        cpi     r16,$00
        brne    GETKEYS         ; wait for release
GETKEYS2:
        in      r16,PIND
        andi    r16,$03
        cpi     r16,$00
        breq    GETKEYS2        ; wait for press
        ret

HW_INIT:
        ldi     r16,$00
        out     DDRD,r16        ; bit1 - left, bit0 - right
        ldi     r16,$FF
        out     DDRB,r16
        clr     r17             ; no sum yet
        ret
```

```
        /* LaxDemo2a.asm
         * Compiles to 72 bytes (64 if call etc)
         * Version for Dalia
         * Input: Buttons INT1 and INT0
         * Output: on-board LED
         */

COLD:
        ldi     r16,HIGH(RAMEND)
        out     SPH,r16
        ldi     r16,LOW(RAMEND)
        out     SPL,r16
        call    HW_INIT
WARM:
        call    GETKEYS
        sbrs    r16,3
        inc     r17             ; left pressed
        sbrs    r16,2
        call    SHOWIT          ; right pressed
        jmp     WARM

SHOWIT:
        cpi     r17,$0F
        brmi    SHOWIT2
        ldi     r17,$0F
SHOWIT2:
        out     PORTB,r17
        ; clr   r17             ; show progress
        ret

GETKEYS:
        in      r16,PIND
        andi    r16,$0C
        cpi     r16,$0C
        brne    GETKEYS         ; wait for release
GETKEYS2:
        in      r16,PIND
        andi    r16,$0C
        cpi     r16,$0C
        breq    GETKEYS2        ; wait for press
        ret

HW_INIT:
        ldi     r16,$00
        out     DDRD,r16
        ldi     r16,$FF
        out     DDRB,r16
        out     PORTD,r16       ; pull-up, PD3 left, PD2 right
        clr     r17             ; no sum yet
        ret
```

```
        /* LaxDemo4.asm
         * Compiles to 86 bytes (72 if rcall etc)
         */

COLD:
        ldi     r16,HIGH(RAMEND)
        out     SPH,r16
        ldi     r16,LOW(RAMEND)
        out     SPL,r16
        call    HW_INIT
WARM:
        call    GETKEY
        cpi     r16,$0F         ; "F"?
        brne    NO_TOGGLE
        com     r18             ; yep!
NO_TOGGLE:
        cpi     r16,10          ; 0-9?
        brpl    WARM            ; A-F no update
        in      r17,PORTB       ; get displayed
        sbrs    r18,0           ; 0 -> right, 1 -> left
        jmp     RIGHT
LEFT:
        andi    r17,$0F         ; clear left
        swap    r16             ; put digit in place
        jmp     SHOWIT
RIGHT:
        andi    r17,$F0         ; clear right
SHOWIT:
        or      r17,r16         ; merge
        out     PORTB,r17       ; and display
        jmp     WARM

        ; --- GETKEY returns key in r16
GETKEY:
        sbic    PINA,4          ; wait for release
        jmp     GETKEY
GETKEY2:
        sbis    PINA,4
        jmp     GETKEY2         ; wait for press
        in      r16,PINA        ; get key
        andi    r16,$0F
        ret

        ; --- I/O
        ; PA4   STROBE
        ; PA3-0 Data
        ; PB7-4 Left digit
        ; PB3-0 Right digit
HW_INIT:
        ldi     r16,$00
        out     DDRA,r16
        ldi     r16,$FF
        out     DDRB,r16
        ldi     r16,$00
        out     PORTB,r16       ; "00"
        clr     r18             ; toggle byte
        ret
```

```
        /* LaxDemo5.asm
         * Compiles to 70 bytes (60 if rcall etc)
         */

COLD:
        ldi     r16,HIGH(RAMEND)
        out     SPH,r16
        ldi     r16,LOW(RAMEND)
        out     SPL,r16
        rcall   HW_INIT
WARM:
        rcall   GETKEY
        mov     r17,r16         ; r16 rightmost
        brne    NO_ZERO
        com     r19             ; was a "0"
NO_ZERO:
        cpi     r19,0
        breq    WARM3
        ldi     r18,$0F         ; invert right
        eor     r16,r18
WARM3:
        swap    r17
        or      r16,r17
        out     PORTB,r16
        rjmp    WARM

        ; --- GETKEY Return pressed key in r16
GETKEY:
        sbic    PINA,4          ; wait for release
        rjmp    GETKEY
GETKEY2:
        sbis    PINA,4          ; wait for press
        rjmp    GETKEY2
        in      r16,PINA
        andi    r16,$0F         ; return key
        ret

        ; --- I/O init, initial state
HW_INIT:
        clr     r16
        out     DDRA,r16
        ldi     r16,$FF
        out     DDRB,r16
        clr     r19             ; 0 -> normal, 1 -> inverted
        ret
```

```
        /*
         * LaxDemo5mini.asm
         * Attempt at minimal, and hence ugly, code
         * Compiles to 32 bytes
         */

COLD:
        ldi     r16,HIGH(RAMEND)
        out     SPH,r16
        ldi     r16,$FF
        out     DDRB,r16
WARM:
        sbic    PIND,0          ; wait for release
        rjmp    WARM
KEY:
        sbis    PIND,0          ; wait for press
        rjmp    KEY
        in      r16,PINA
        cpi     r16,0
        brne    CONT
        com     r17             ; was a "0"
CONT:
        sbrc    r17,0
        eor     r16,r17
        out     PORTB,r16
        rjmp    WARM
```