

Namn:
Laborationen godkänd:

Digitala system 15 hp



LUNDS TEKNISKA HÖGSKOLA

Lunds universitet

LTH Ingenjörshögskolan vid Campus Helsingborg

Datorteknik 1 (AVR 1)

Introduktion till datorteknikutrustningen.

Laborationens syfte.

Syftet med laborationen är att förstå hur en enchipsdator (mikrostyrcrets, ennkapseldator) kan vara uppbyggd, hur den programmeras och hur den ansluts till omgivningen. Stor vikt ligger på maskinnära programmering.

I laborationen kommer en Atmega32, som är en enchipsdator från ATMEL, att användas. Atmega32 tillhör AVR-familjen och är en arkitektur utvecklad vid Tekniska högskolan i Trondheim. I laborationen kommer programmering ske i C.

I laborationen ska du bekanta dig med utrustningen och programmet **AVR Studio**.

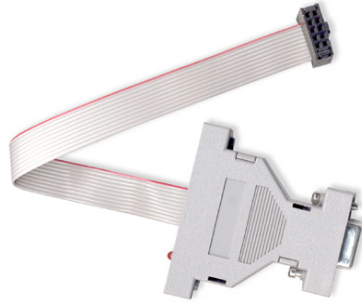
AVR Studio kan fritt laddas hem om du vill prova hemma.

Utrustning.

Utrustning för laborationen är enligt följande:

- **Målsystem.** Med målsystem menas den hårdvara (kretskort) konstruktören bygger och som i vårt fall ska innehålla en enchipsdator. Målsystemet som används under kursen varierar. I denna laboration används ett kretskort som förutom Atmega32 har en tryckknapp och en så kallad sju-segmentsdisplay. Målsystemet spänningsmatas via ett kopplingsdäck och ansluter till en JTAG-programmerare.
- **AVR-JTAG-programmerare** från Olimex. **JTAG** står för "Joint Test Action Group" och var en grupp som tog fram IEEE 1149, vilket är en standard för testning. Standarden används för andra ändamål, t. ex. programmering av kretsar. JTAG består fysiskt sett av en liten sekvensmaskin som är inbyggd i enchipsdatorn. Med JTAG kan man styra enchipsdatorn: starta och stoppa den, läsa och skriva inuti den. JTAG kommunicerar via några ben på kretsen med JTAG-programmeraren. Detta sätt att testa program och hårdvara brukar kallas **OCD** – On Chip Debugging. Ibland kallas denna typ av testning för "In Circuit

Emulation” **ICE**. Programmeraren sitter alltså mellan AVR-processorn och PC:n som du använder. I PC:n kopplas den in på en vanlig COM-port.



Programmeraren, OCD

- *AVR Studio*. En utvecklingsmiljö för AVR-processorer som innehåller assembler och debugger. Programmet används för att skriva och testa program direkt i en processor. Till detta program kan man koppla olika kompilatorer, dvs. översättare av högnivåspråk. I kursen använder vi:
- *Win-AVR*. C-kompilator, freeware, som används från AVR Studio.
- *Spänningsaggregat 5 volt*.

Laborationsuppgifter.

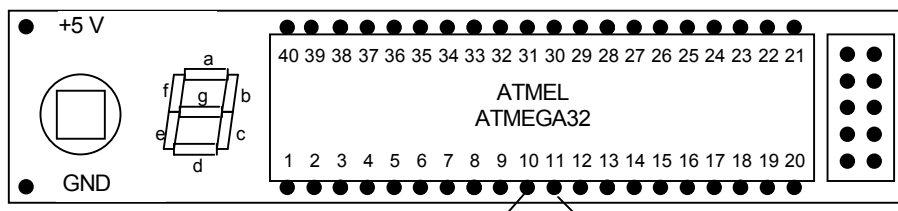
Med hjälp av ett litet projekt går vi igenom de viktigaste momenten för att utveckla och testa program med **AVR Studio** och med JTAG-programmeraren.

Med AVR Studio kan du förbereda laborationerna hemma och prova dina lösningsförslag då AVR Studio är ett gratisprogram som kan laddas ner från <http://www.atmel.com/products/avr/>. Välj ”Tools & Software”. AVR Studio kan också laddas ner från kursens hemsida. Där finns också länken till WinAVR som innehåller själva C-kompilatorn.

AVR Studio är installerat på datorerna i laborationssalarna.

Uppgift 1. Inkoppling av hårdvaran.

För att underlätta den elektriska inkopplingen kommer vi att använda en enchipsdator med en färdig anslutning till emulatorn.



Det enda som måste kopplas för hand är spänningsmatningen:

- + 5 V till kopplingsplattans polskruv.
- GND till kopplingsplattans polskruv.

När du kopplar in strömmen ska den röda lysdioden lysa på programmeraren. Om det inte gör det, stäng av strömmen och tillkalla handledaren.

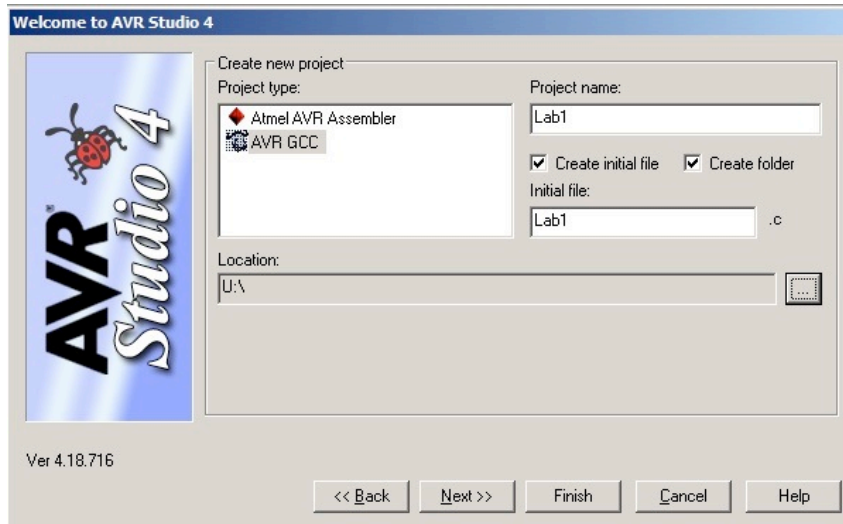
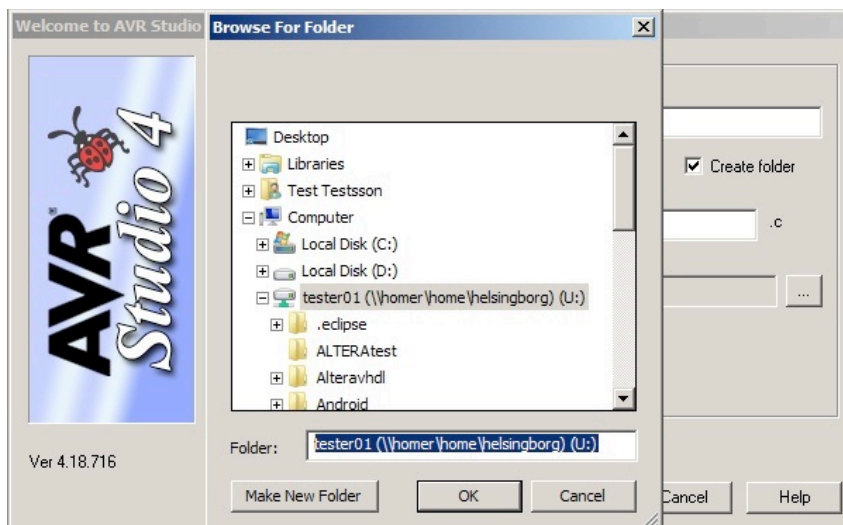
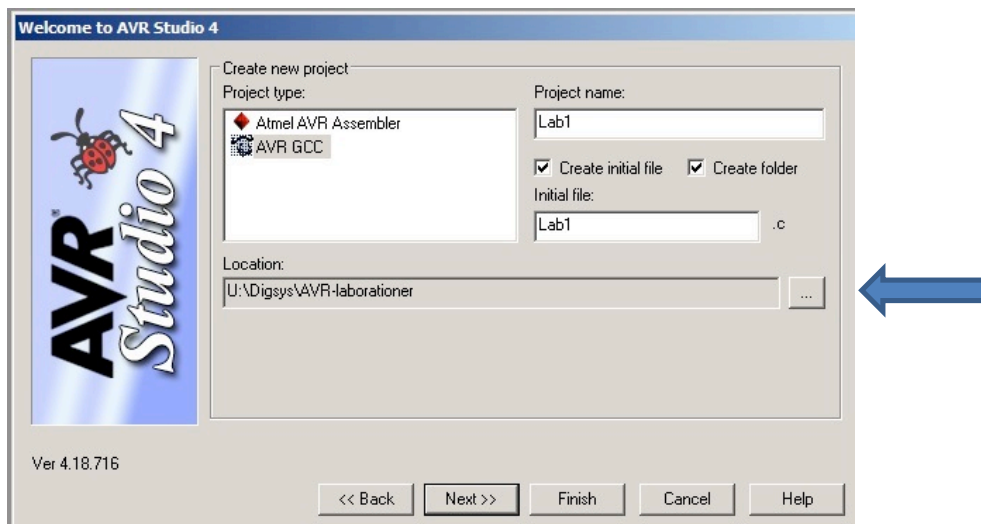
Uppgift 2. AVR Studio 4.

- Starta AVR Studio (Klicka på den lilla röda skalbaggen).

Under menyn **Project**, välj **New Project**.

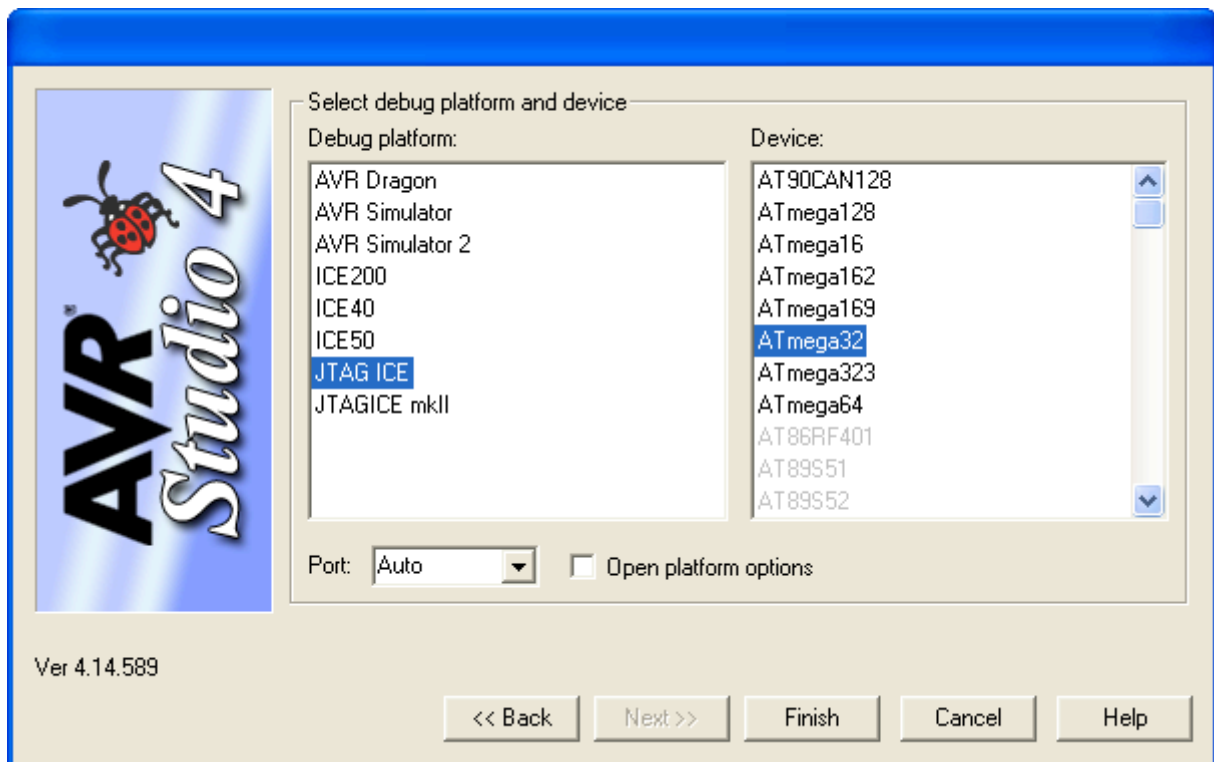
Dialogrutan ”Create new project” kommer upp.

- Välj under **Project type: AVR GCC**



Välj projektnamn och det skapas en mapp och en fil med samma namn.

- Välj eller skapa en lämplig mapp för denna laborationskurs.
- Tryck på **Next>>**



Nu kan du välja om du vill använda simulatoren i AVR Studio (AVR Simulator) eller om du har en riktig processor inkopplad (JTAG ICE). Vi kommer under laborationerna att använda **JTAGICE**, men du kan testa dina program genom att använda simulatoren om du inte har någon hårdvara inkopplad. Under den första delen av laborationskursen använder vi **ATMega32**.

- Ställ in enligt bilden ovan och tryck på **Finish!**

Om du senare vill ändra på detta, gör du det under menyn:
Debug → Select Platform and Device.

- Skriv in följande i källkods-fönstret som öppnas:


```

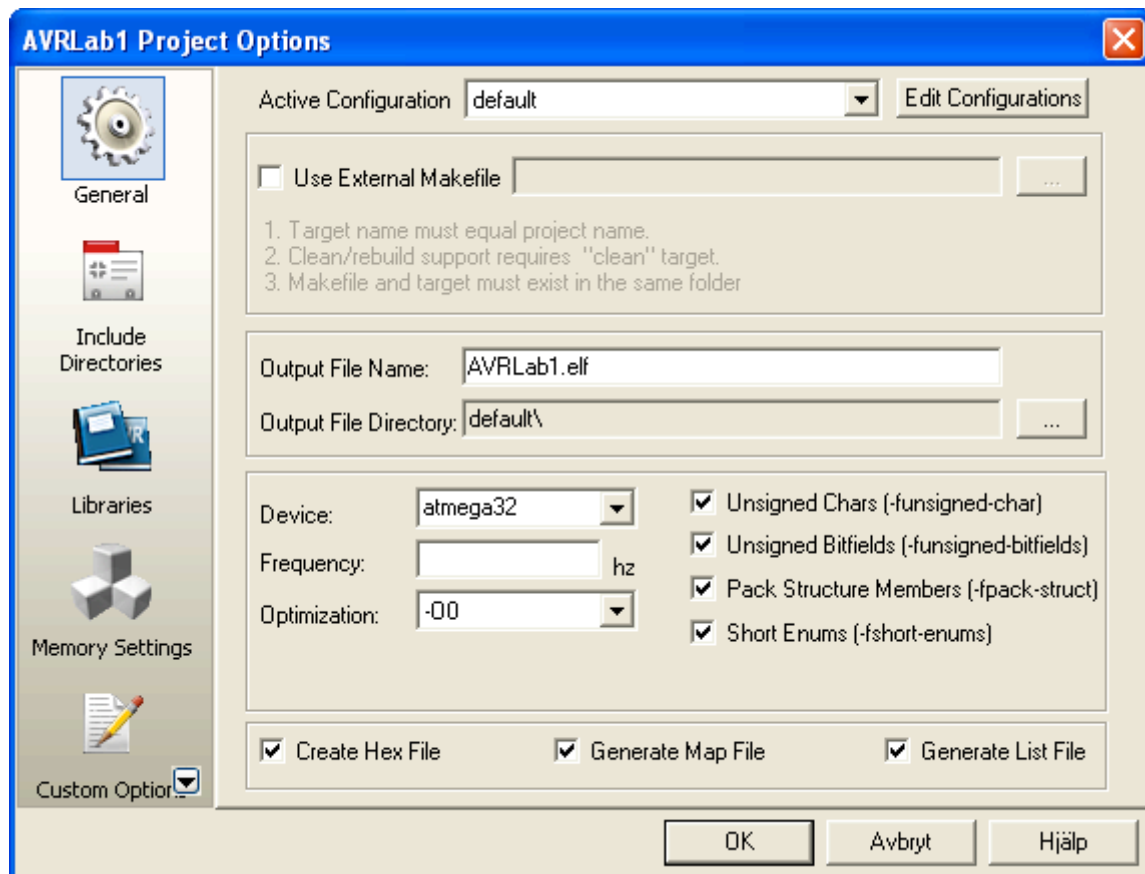
unsigned char b;
int main(){
    b = 5;
    while(1==1) {

        while(b<10) {
            b = b + 1;
        }
    }
}

```

Innan vi kompilerar denna kod, bör vi kontrollera ett par inställningar.

- Tryck på  (Edit Current Configuration Options)



Kontrollera dessa rutor och se till att det står:

Device: atmega32



Optimization: -O0

Att det står **atmega32** innebär att kompilatorn som översätter koden till maskininstruktioner får veta att vi använder en atmega32.

-O0 (streck, stora O, noll) innebär att ingen optimering ska vara igång; dvs. kompilatorn ska inte försöka sig på att vara smart och förenkla översättningen.

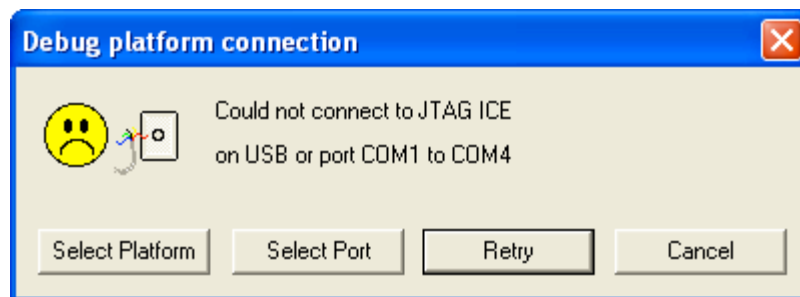
- Tryck på  (Build Active Configuration) !

Om inga varningar eller felmeddelanden kommer upp, så är du klar att provköra mikroprocessorn. Programmet finns nu färdigkompilerat och klart att ladda ner till utrustningen på bordet.

- Tryck på  (Start debugging). Det finns också en kombinationsknapp  (Build and Run) som kompilerar och startar debuggern direkt. Observera att

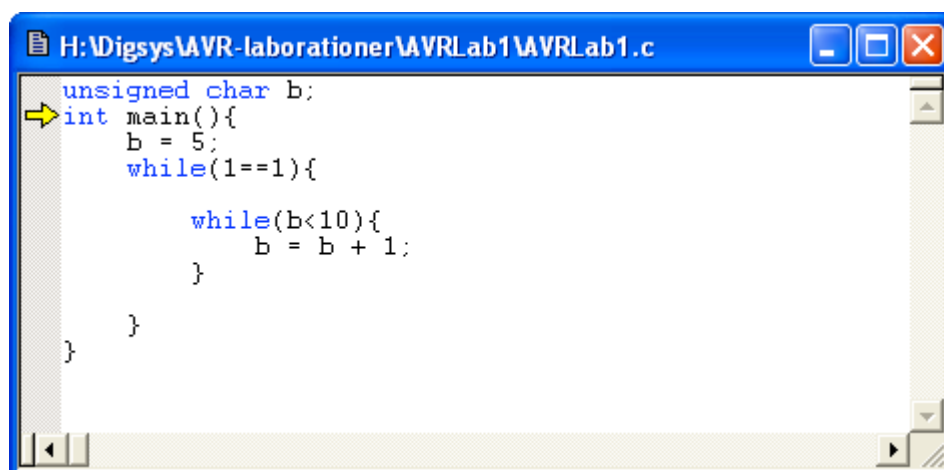
programmet ännu inte kör (även om det står ”Run”).

Ibland kommer följande meddelande kommer upp:



Anledningen kan vara att emulatoern inte är ansluten eller att spänningsmatningen inte finns. Ibland kommer meddelandet ändå. Tryck då på knappen ”Retry” så brukar det ordna sig.

När nerladdningen sker, syns en blå stapel röra sig nere till vänster. Om allt är rätt inkopplat öppnas bl.a. detta fönster:



Observera den gula pilen. Den visar vilken sats som ska utföras. När pilen syns vet du att programmet är i debug-läge. I vårt fall så innebär det att C-programmet är i enchipsdatorns programminne. Programmet är omkodat (kompilerat) till datorns eget språk och består nu av ett antal instruktioner kodade med 16 bitar var. Du kan se koden genom att välja **View** → **Disassembler**.

Under menyn View kan man öppna ett antal fönster. Välj fönstret ”Watch”.

- Dubbelklicka i kolumnen ”Name”. Skriv in variabelns namn.

Name	Value	Type	Location
b	0 ''	unsigned char	0x0060 [SRAM]

Så här blir det:

Variabeln heter b. Den har fått startvärdet 0. Typen är ”unsigned char”, dvs. åttabits positivt talområde. Kolumnen ”Location” anger adressen till den plats i minnet där variabeln finns.

- Stega nu programmet med knappen  (Step Into) och se hur värdet ändras.

Uppgift 3.

- Ändra nu i programmet så att b räknas *ner* istället. Behåll resten av koden oförändrad!

Vad händer med variabeln b?

Varför?



Uppgift 4.

- Ändra nu deklARATIONEN av b till ”**signed char**”.

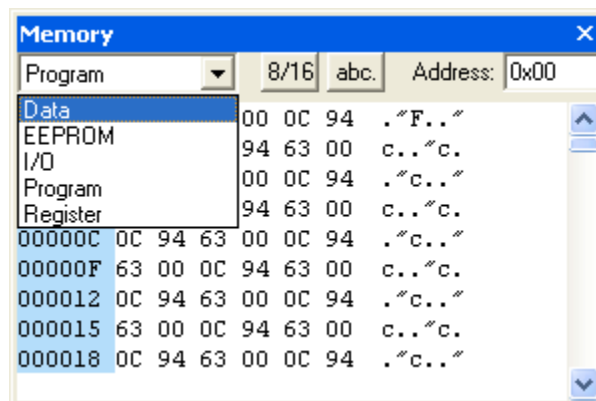
Vad händer nu när b räknas ner?

Varför?



Uppgift 5. Variabelminnet.

- Använd menyn ”View” igen och välj ”Memory”. Du kan här välja mellan fem olika alternativ.



Nästan alla platser där värden kan läggas kan man nå med adresser.

- Välj alternativet "Data".

Vilket värde står det i adressen 0x0060?

Vilken är den lägsta adressen där variabler kan läggas?

Och den högsta?

Hur stort är då dataminnet?

- Kör programmet några gånger från start tills det stoppar och kontrollera värdena i minnescell 60 och variabeln b.



Uppgift 6.

- Ändra b till "**signed int**". Kontrollera variabelns värde och jämför med dataminnet.

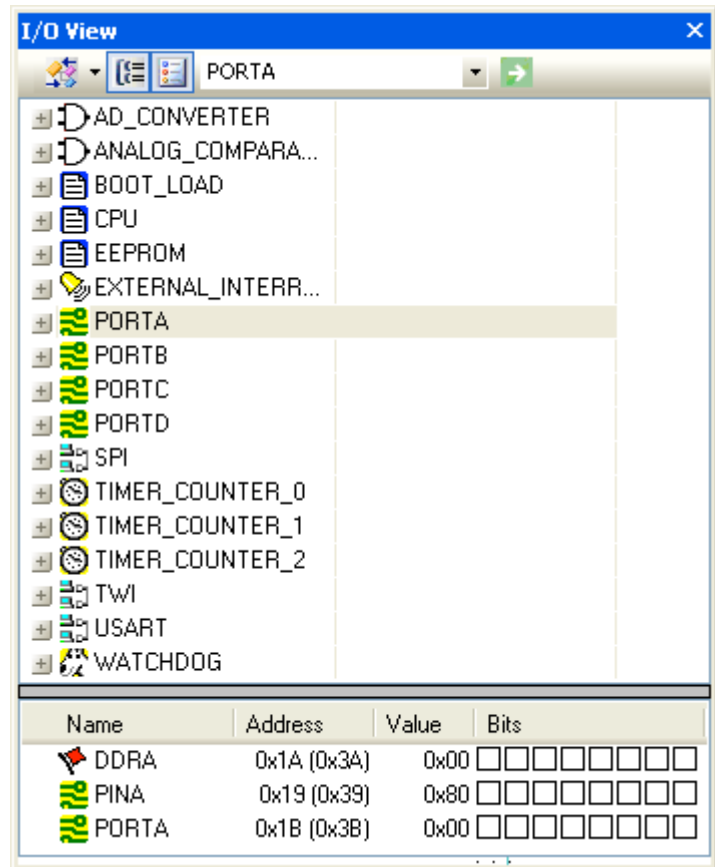
Förklara vad du ser

Hur lagras 16-bitarsvärdet i minnet?.....






Uppgift 7. Utsignaler

Fram tills nu har vi egentligen inte använt mikrodatorn på kopplingsplattan till något verkligt. Vi kunde lika gärna ha använt en simulator. Men det finns en tryckknapp och en sjusegmentsdisplay på det kretskortet med processorn. Med sjusegmentsdisplayen kan man visa vanliga siffror på ett ganska tydligt sätt. Vår display är uppbyggd av lysdioder och varje diod är inkopplad till varsin pinne på en av portarna, PORTA. Porten består av åtta bitar och alla utom den översta (den som är längst till vänster) biten är inkopplad till displayen. Den översta biten är inkopplad till tryckknappen. I den här uppgiften ska du undersöka hur displayen är inkopplad.



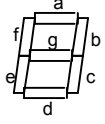
I fönstret ”I/O View” kan du se processorns hela innanmäte. De svarta fyrkanterna symboliserar de enstaka bitarna. En tom fyrkant är en nolla och en svart är en etta.

- Prova med att klicka i rutorna! Eftersom hela porten är riktad in från början, måste de signaler som går till lysdioderna göras till utsignaler. Det gör man i DDRA-rutorna.
- Ettställ de bitar i registret DDRA som hör till de sju lägsta bitarna i porten! Den översta biten (längst till vänster) är inkopplad till tryckknappen, så den ska fortfarande vara insignal. Så här ska det se ut:

 DDRA	0x1A (0x3A)	0x7F	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
 PINA	0x19 (0x39)	0x00	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
 PORTA	0x1B (0x3B)	0x00	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

Nu kan du testa varje bit i PORTA för sig och se vilket segment som tänds när du ettställer en bit i taget.

- Fyll i tabellen!

 <p>OBS Bit 0 längst till höger.</p>	Bit nr:	Segment
	0	
	1	
	2	
	3	
	4	
	5	
6		



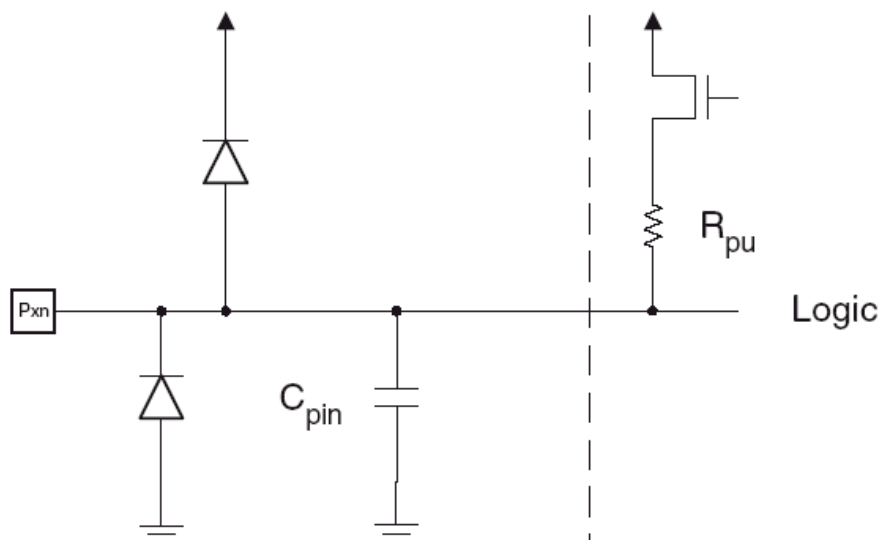
Uppgift 8. Insignal

- Kör programmet ovan med hjälp av ”Autostep” och tryck på tryckknappen. Titta på bitarna i PORTA (speciellt en av bitarna i PINA) medan programmet körs (Bry dej inte längre om variabeln b. Du kan stänga det fönstret).

Händer något och i så fall vad?

För att signalen från knappen ska fungera tillförlitligt, måste ett motstånd kopplas in till +5V så att pinnen inte ska ”hänga i luften”. Smartast gör man det genom att ettställa bit 7 i PORTA. Då kopplas det nämligen in ett internt pullup-motstånd till denna bit.

Här ser du en del av kopplingsschemat (ur databoken) för en enstaka signalpinne på processorn: R_{pu} är ungefär 40 kohm.



- Ettställ bit 7 i PORTA, dvs koppla in pullup-motstånd, och kör programmet med autostep. Tryck på knappen och kontrollera att bit 7 rätt avspeglar varje knapptryckning.



Uppgift 9. Det färdiga programmet

Nu ska vi se till att få med dessa inställningar i programmet. Här är ett lämpligt programskal:

```
#include <avr/io.h>    // Hämtar en fil som definierar portar mm

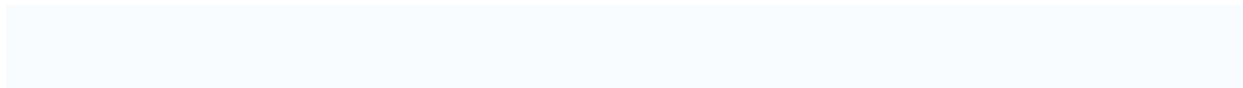
int main(void) {
    DDRA = 0b01111111; // gör 7 pinnar till utgångar
    PORTA = 0b10000000; // koppla in pullup-motståndet

    while (1==1) {
        if ((PINA & 0b10000000)==0b10000000) PORTA=0b10000000;
        else PORTA=0b11111111;
    }
}
```

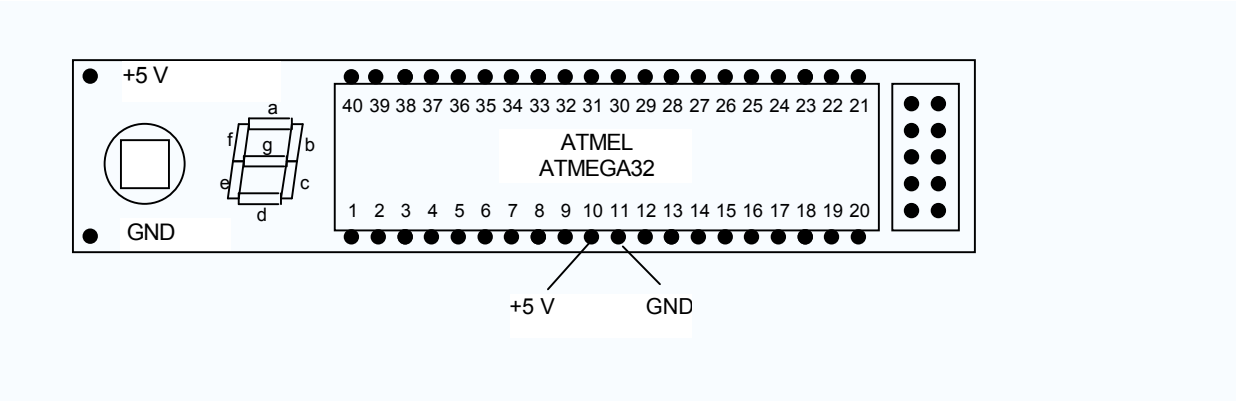
- Skriv in koden och provkör!



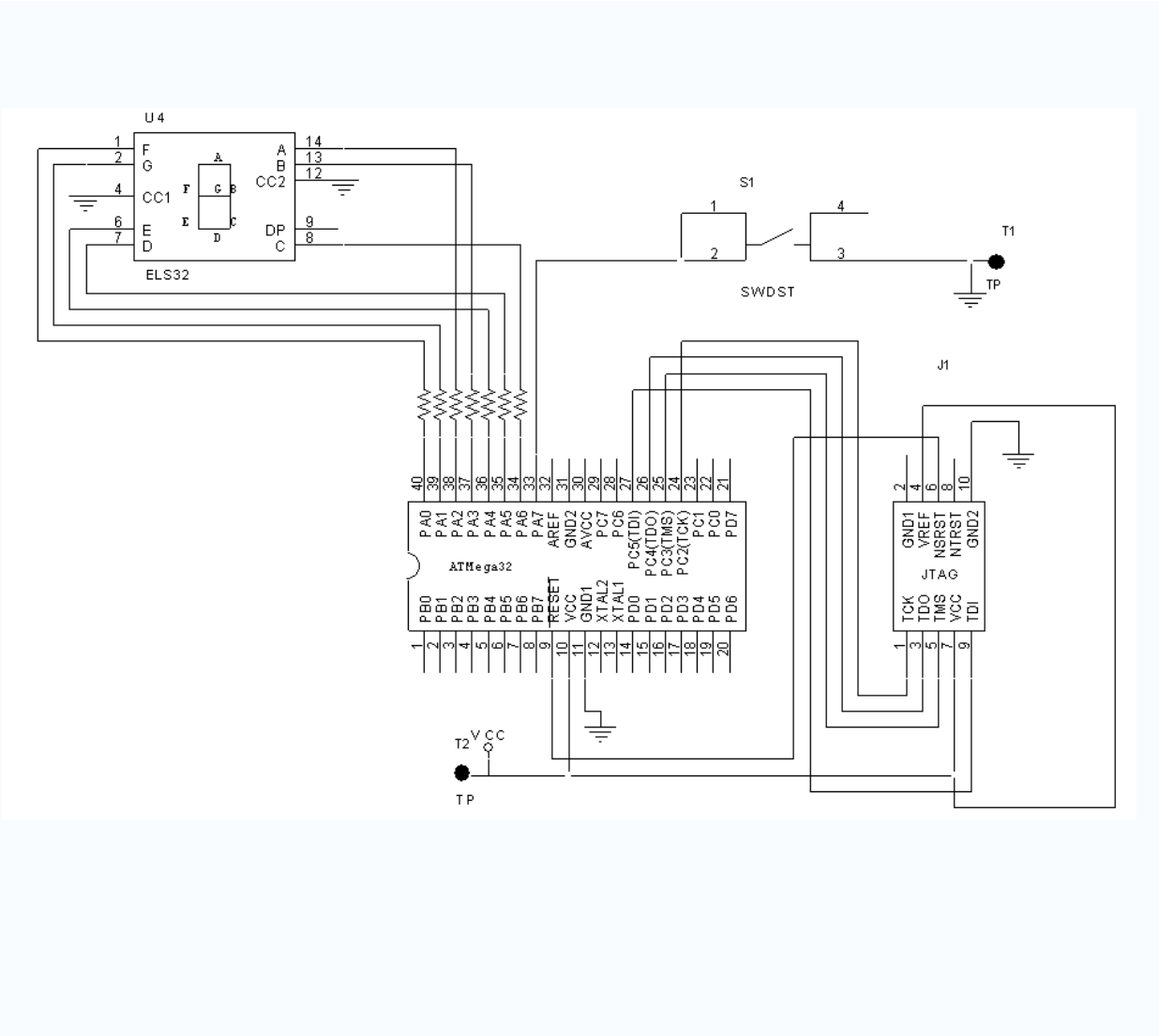
- Ändra nu denna kod så att det skrivs en etta när knappen är nedtryckt och en nolla annars.



Målsystem, layout.



Schema.



Erik Larsson

LTH Ingenjörshögskolan
vid Campus Helsingborg