Namn:
Laborationen godkänd:

Digitala system 15 p

LUNDS TEKNISKA HÖGSKOLA
Lunds universitet
LTH Ingenjörshögskolan vid Campus Helsingborg

Datorprojekt, del 1

Projektet består i att skapa en klocka där tiden visas på en LCD och som ställs med hjälp av knappar eller med en insignal som kommer från en annan klocka.

Projektet är i 4 laborationer där deluppgifter ska sättas ihop till en fungerande helhet. En extra (frivillig) uppgift är att göra en väckarklocka. Laborationerna får göras i valfri takt, men för att få hjälp får du komma till de schemalagda tillfällena.

Projektet ska redovisas i en skriftlig rapport som ska vara klar senast en vecka efter sista laborationstillfället. Rapporten ska innehålla en beskrivning av hela projektet och de ingående delarna. Dessutom ska flödesplaner och en väl kommenterad källkod ingå. En fördel med väl kommenterad källkod är att det blir lättare att sätta dig in i koden som ska användas i en annan kurs.

Rapporten skickas som PDF till lars-goran.larsson@eit.lth.se och en pappersversion utan källkod men med försättsblad lämnas i fack utanför C426.

I projektet tas följande moment upp:

- Tangentavkodning.
- Skrivning till LCD - teckenfönster.
- Hantering av klocka.
- Asynkron seriekommunikation.

I projektet kommer du att behöva skriva en mängd drivrutiner för att kunna handskas lätt med de olika komponenterna på laborationskortet. Det är därför viktigt att organisera programmen bra.

Tänk på följande:

- Alla filer ska ligga i samma mapp, t.ex. ”AVR-projekt”.
- Funktioner som hör samman ska ligga i samma fil.
- Huvudprogrammet ska vara i filen AVR-MT-128.c

Erik Larsson

- Skriv funktionerna som de beskrivs i labbuppgifterna. Om du gör dem på något annat sätt, måste du motivera detta.
- Se alltid till att du och din labbkamrat kan nå hela projektet och alla filer vid de återkommande laborationstillfällena. Detta är speciellt viktigt när någon av er av någon anledning inte kan närvara vid det ordinarie tillfället.

Projektlaboration 1, tangentavkodning.

Förberedelser:

Läs igenom denna handledning. **Gör i förväg förslag till de kompletteringarna som behövs i funktionen Initialize (i AVT-MT-128.c) och i filen AVR-MT-128-buttons.c**. Tänkt igenom dessa uppgifter innan du börjar laborera. Skal till laborationen finns på kurshemsidan. Skalen består av AVR-MT-128.c (huvudprogramfilen), AVR-MT-128-buttons.c och AVR-MT-128-buttons.h.

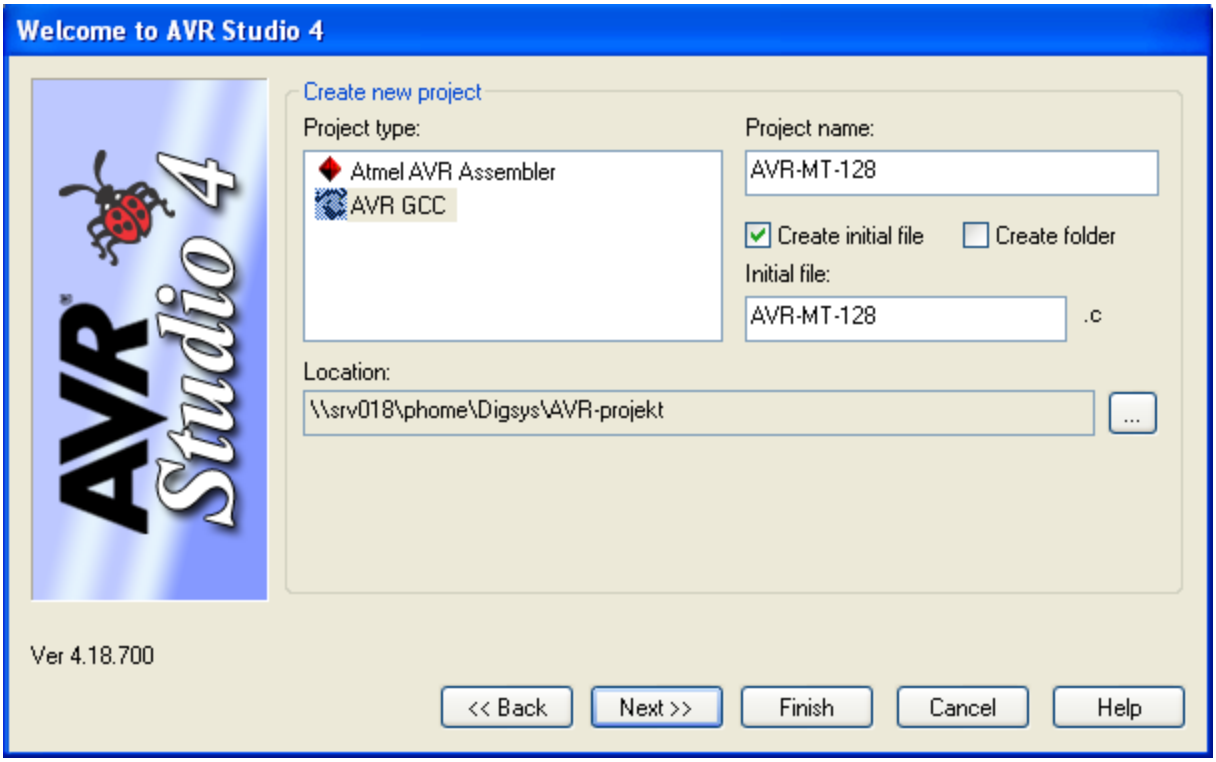
Laborationsuppgifter.

Uppgiften i denna laboration består i att skapa funktioner så att tangenterna kan avläsas på ett säkert sätt. Dessutom ska några funktioner skrivas så att reläet med dess lysdiod kan manövreras. Jämför med tidigare laborationer för avläsning av knapptryck och tändning av lysdiod.

Tangenterna är inkopplade till port A. De är betecknade B1, B2, B3, B4 och B5 på kortet. Men vi kommer att kalla dem **up**, **left**, **middle**, **right** och **down**. Det gör det lättare att hålla reda på dem i programmeringen. Ett schema som visar hur de är inkopplade finns i slutet av handledningen.

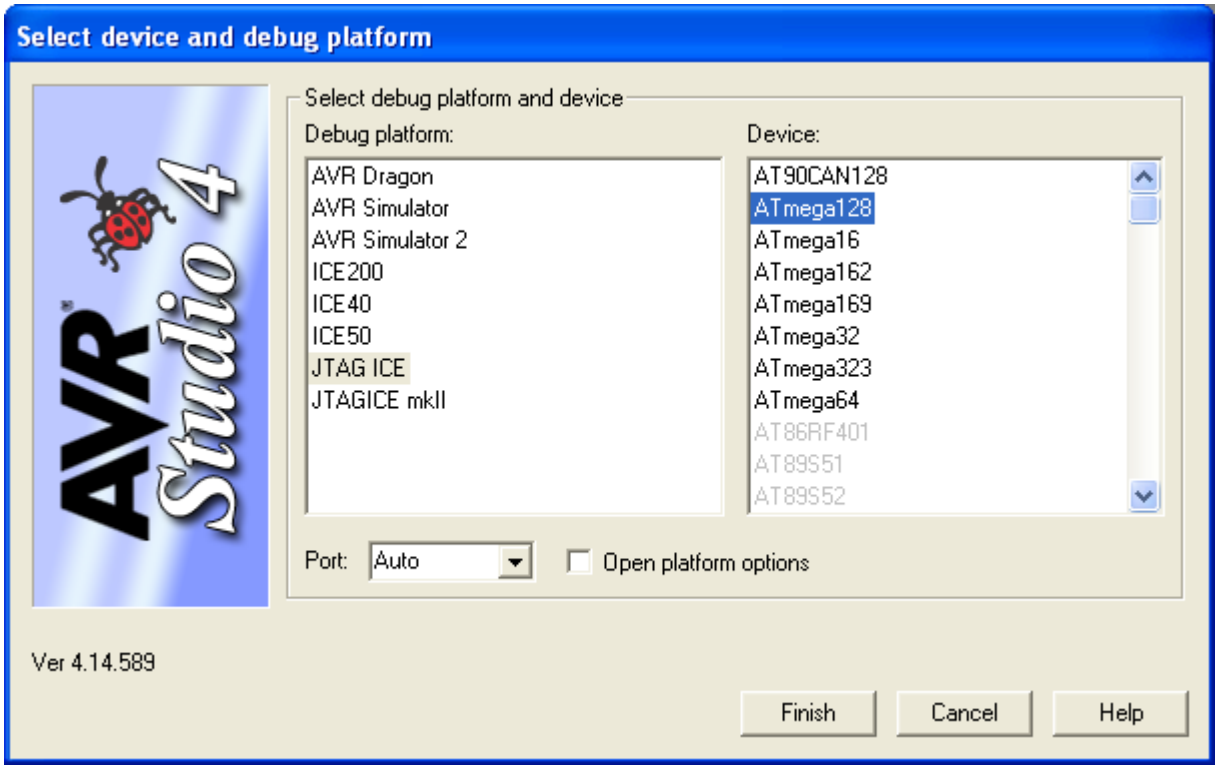
Uppgift 1. Testning av laborationskortet AVR-MT-128

Skapa ett nytt projekt! Detta ska du arbeta med under resten av laborationerna. Var noggrann med hur du väljer mapp där projektet ska ligga. Följ gärna exemplet nedan:

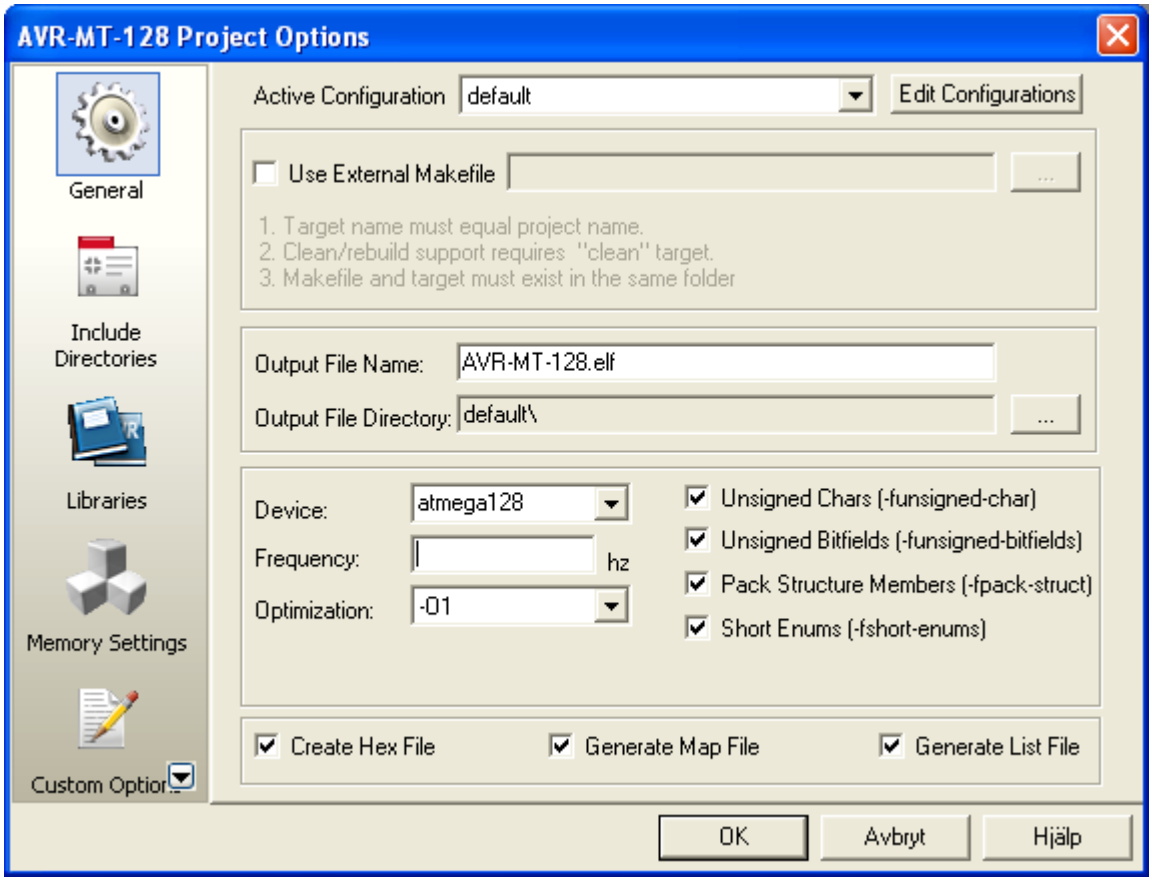


På det här sättet får du en mapp som heter ”AVR-MT-128” och en C-fil med samma namn. Programskalet som finns på kurshemsidan kopieras till filen.

Laborationskortet är försett med en ATMEGA128. En kortfattad presentation och en fullständig manual finns på kurshemsidan. Några gula klisterlappar med förklaringar till sådant vi stöter på under laborationen finns i manualen. Eftersom det är en ATMEGA128, väljer vi det i ”Device” inställningarna:



Ytterligare en inställning måste göras i ”Configuration Options”:



På kurshemsidan finns tre filer som ska användas som start i projektet:
AVR-MT-128.c En fil som än så länge bara innehåller ett tomt huvudprogram och en funktion för inställning av portarnas riktning så att det stämmer med konstruktionen. Koden här kopierar du till din egen **AVR-MT-128.c**
AVR-MT-128-buttons.c som tillsammans med **AVR-MT-128-buttons.h** ska innehålla vad som behövs för knappavläsningen.

Den första uppgiften är att komplettera funktionen ”Initialize” (se nedan) med inställningar av portar. Jämför med tidigare laborationer.

```
void Initialize(){
    DDRA = 0;  // Relä och tryckknappar
    DDRC = 0;  // Styrsignaler LCD
    DDRE = 0;  // Högtalare
}
```

Komplettera och starta debuggern med bara denna fil i projektet!

Testa nu med hjälp av debuggern att du kan manövrera reläet genom att ettställa och nollställa rätt bit i porten.

Tangentavsökning

Tangentavsökningen görs med vissa intervall. Intervallen får inte vara så långa att man kan missa en knappnedtryckning (storleksordningen sekunder). Det är också olämpligt att avsöka knapparna så ofta att studsar också registreras. Ett ”lagom” intervall kan vara ungefär 10 ms (jämför med tidigare laborationer). I **AVR-MT-128-buttons.c** finns funktionerna `SIG_OUTPUT_COMPARE2` och `initOC2`. Vi använder ett annat periodiskt avbrott för knappavsökningen än det 10-ms-avbrott som användes i föregående laboration. Istället för exakt var 10:e ms kommer avbrottet OC2 att köras ungefär 122 gånger i sekunden. Detta duger bra för knappavläsning. Vi spar OC1-avbrottet till klockan, som kommer att konstrueras i projektlaboration 3. Tangenterna är inkopplade till de fem lägsta bitarna i PORTA.

Som vi sett i en tidigare laboration behöver vi spara knapparnas status mellan avläsningarna:

```
unsigned char remembered_buttons;
```

Genom att med jämna mellanrum jämföra knapparnas aktuella läge (`actual_buttons`) med det förra (som vi sparat i `remembered_buttons`), kan vi lätt ta reda på om en knapp har tryckts ned på nytt.
För det ändamålet skapar vi även dessa variabler:

```
unsigned char actual_buttons;
unsigned char new_buttons;
```

Titta på följande programrader:

```
actual_buttons = ~PINA & 0x1F;

new_buttons = new_buttons|((actual_buttons)&(~remembered_buttons));

remembered_buttons = actual_buttons;
```

Vi börjar med att läsa av de fem tryckknapparna. Tecknet ~ gör att värdet inverteras, dvs. en etta blir nolla och tvärtom. Anledningen är helt känslomässig: en etta verkar bättre symbolisera en nedtryckt knapp! Knapparnas aktuella lägen lägger vi i variabeln `actual_buttons`:

`actual_buttons:`

0	0	0	**down**	**right**	**middle**	**left**	**up**

De tre översta bitarna nollställer vi eftersom vi inte vill ha ovidkommande signaler med i variabeln.

Att fundera över: borde vi inte klara oss utan variabeln ”**`actual_buttons`**”?

Nästa rad kräver en noggrann förklaring. Det vi är ute efter, är att ta reda på om en knapp är nere nu men inte var det förra gången. Uttrycket här nedanför avspeglar ju just detta.

```
(actual_buttons)&(~remembered_buttons)
```

I varje bitposition kommer vi att få en etta just när en knapp tryckts ned på nytt. Och vi behandlar alla knappar samtidigt!

```
new_buttons = new_buttons | ((actual_buttons)&(~remembered_buttons));
```

Tilldelningen är ett standardexempel på hur man ettställer bitar i en variabel.
Resultatet blir att vi har en variabel ”`new_buttons`” som innehåller fem ***flaggor*** som var och en med en etta meddelar att någon har tryckt på en viss knapp.
Meningen är nu att ett annat program (varsomhelst) ska kunna läsa av och kvittera någon av flaggorna för att utföra något speciellt.
Den sista raden: **`remembered_buttons = actual_buttons;`**
behövs för att vi ska kunna komma ihåg till nästa gång och inte blanda ihop de nyinlästa värdena med de gamla.

Sammanfattningsvis:

`actual_buttons:`

0	0	0	**down**	**right**	**middle**	**left**	**up**

och

`remembered_buttons:`

0	0	0	**down**	**right**	**middle**	**left**	**up**

Avspeglar knapparnas lägen.
”1” = nedtryckt.
”0” = uppe

`new_buttons:`

0	0	0	**down**	**right**	**middle**	**left**	**up**

Detta är händelseflaggor som var och en med en etta meddelar att motsvarande knapp tryckts ner. Flaggorna ***ska kvitteras*** i samband med att uppgiften utförts.

Funktionen finns tillsammans med avbrottsfunktionen (som anropar den) i filen
AVR-MT-128-buttons.c
Lägg märke till raden #include "AVR-MT-128-buttons.h" i main-filen. Vi har tidigare använt h-filer (header-filer) men nu har vi tagit med en egen som behöver kompletteras.

Uppgift 2. Kvittering av knappnedtryckningar.

Studera filen **AVR-MT-128-buttons.c** . Filen innehåller bl.a. fem funktioner som ska användas för att kvittera knappnedtryckning.

De fyra ofullständiga funktionerna ska du komplettera!

Uppgift 3. Definitioner för att förbättra läsligheten; knappdefinitioner.

Filen **AVR-MT-128-buttons.h** innehåller förutom externdeklarationer även några definitioner som vi inför för att underlätta programmeringen. Definitionerna ger oss namn på alla intressanta bitar i actual_buttons och new_buttons och ska fungera såhär:

Om vi t.ex. vill testa biten ”middle” i variabeln ”actual_buttons” (vi vill se om mittknappen är nere) så ser det ut såhär:

```
if (middle != 0) ........;
```

eller helt enkelt:

```
if (middle) ........;
```

Om vi vill testa biten ”new_middle” (i ”new_buttons”) så ser det ut såhär:

```
if (new_middle) ........;
```

Uppgift 4. Styrning av relä

Reläet är inkopplat tillsammans med en lysdiod som tänds när reläet är ”draget” (aktiverat). Du har tidigare tagit reda på var reläet är inkopplat. Uppgiften är nu att skriva dessa tre funktioner:

- **`relay_on`** Aktivera reläet.
- **`relay_off`** Släpp reläet.
- **`relay_toggle`** Växla reläets läge.

De tre funktionerna läggas i en ny fil ***AVR-MT-128-relay.c*** och du får också skapa en **header-fil** till denna.

Testa de tre funktionerna genom att anropa dem i huvudprogrammet. Stega försiktigt igenom dem så du ser att de fungerar.

Uppgift 5. Styrning med knapparna

Nu har vi ett fungerande programpaket så att vi kan

- läsa av alla knapparnas lägen.
- få meddelande för varje ny knappnedtryckning.
- tända lysdioden (aktivera reläet).
- släcka lysdioden (avaktivera reläet).
- växla lysdiodens status (om den är släckt så tänds den och tvärtom).

Nu ska du själv skriva nya programrader i main så att vi kan testa knapparnas funktion.

Om **`left`** inte är nere så ska det fungera såhär:

1) Ett tryck på **`up`** tänder lysdioden (drar reläet).
2) Ett tryck på **`down`** släcker lysdioden (släpper reläet).
3) Ett tryck på **`middle`** ger en växlande funktion, dvs. om lysdioden lyser, så ska den släckas och tvärtom.

Om **`left`** är nedtryckt så ska det fungera såhär:

4) Håller man **`right`** nere så ska lysdioden lysa; släpper man knappen så ska den slockna.

Bilaga: Schema för laborationskortet