Namn:
Laborationen godkänd:

Digitala system 15 hp

LUNDS TEKNISKA HÖGSKOLA
Lunds universitet
LTH Ingenjörshögskolan vid Campus Helsingborg

AVR 3 – datorteknik

Avbrott.

Syften med den här laborationen är att introducera avbrott.

Avbrott som uppkommer periodiskt kallas periodiska avbrott. Med ett periodiskt avbrott kan vissa uppgifter utföras med jämna mellanrum. Periodiskt avbrott används i alla sorters datorsammanhang. Ibland kallas avbrott för ”undantag” eller ”exception”. I denna laboration ska vi se hur avbrott används för att få kontroll över programmet och dess kommunikation med omvärlden.

Förberedelser

Lösningen i förra laborationen har nackdelar. Datorn kommer att ödsla tid till att göra ingenting på grund av looparna för fördröjning. Ett bättre alternativ är att låta programmet läsa av tryckknappen med jämna mellanrum.

Så här kan programmet från förra laborationen beskrivas:

1. Vänta tills knappen är nedtryckt.
2. Vänta med hjälp av vänteloop en viss tid så studsarna slutat.
3. ***Utför uppgiften*** (*öka n med ett; om n blir 10 sätt n till noll; anropa utskriftsfunktionen med n som inparameter.*)
4. Vänta tills knappen är släppt.
5. Vänta med hjälp av vänteloop en viss tid så studsarna slutat.
6. Gå till punkt 1.

Med hjälp av periodiskt avbrott ska vi skapa en lösning som med jämna intervall gör följande:

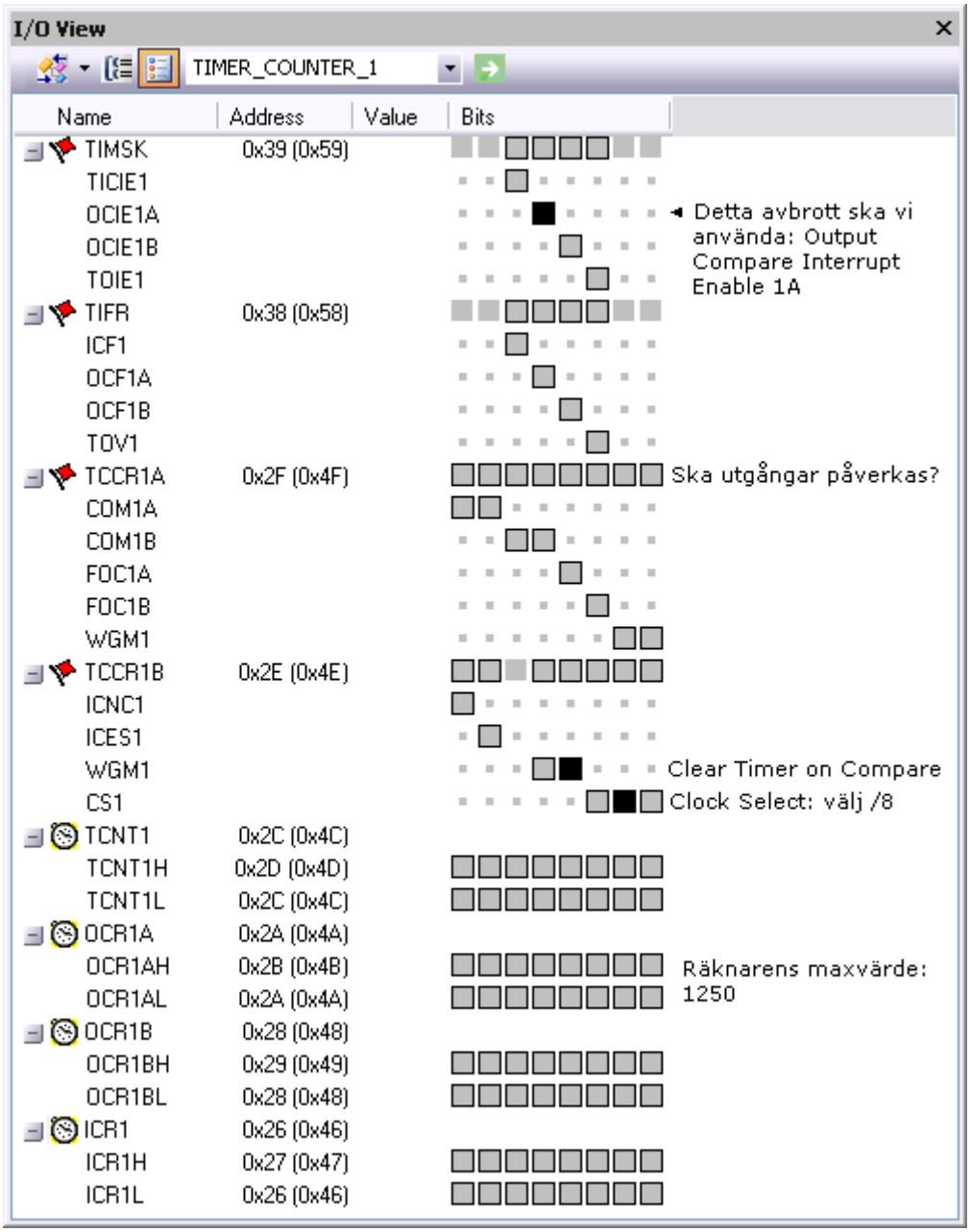
1. Läs av knappen.
2. Om den är nere nu, men var uppe förra gången: *Utför uppgiften* (som är: *”öka n med ett; om n blir 10 sätt n till noll; anropa utskriftsfunktionen med n som inparameter.”*)

Vi ska lösa detta med en ***periodisk avbrottsfunktion.*** Det är en klocka som kommer generera ett avbrott med jämna perioder, exempelvis 10 ms intervall. Avsikten är att få saker gjorda regelbundet; inte för ofta och inte för sällan.

Erik Larsson

Vi ska använda en av Atmega32:s inbyggda enheter som kallas **Output Compare Unit.** Den består i princip av en 16-bitarsräknare som man kan få att räkna i några olika farter och ett 16-bitars jämförelseregister vars värde hela tiden jämförs med räknaren. När de överensstämmer kan man få olika saker att hända, bl. a. att en utgång påverkas (därav namnet ”Output Compare”) eller att dess avbrottsfunktion anropas.

Vi ska använda Timer/Counter1, se översikten här nedanför. Här i bilden får du tips med hur ett periodiskt avbrott med ett intervall på 10 ms skapas:



I korthet fungerar det så här:

1. Vi ställer in så att 16-bitarsräknaren TCNT1 räknar i en lämplig fart.
2. När den nått sitt ett maxvärde ser vi till att den nollställs.
3. Vi ställer in maxvärdet OCR1A så det passar oss.
4. Vi bestämmer att avbrottsfunktionen ska köras varje gång maxvärdet nås.

Ett antal register ska ställas in:

TIMSK, Timer Interrupt Mask:
Biten OCIE1A (Output Compare A Match Interrupt Enable) är ettställd (s 112 i databoken). När denna bit är ettställd, kommer en speciell funktion (avbrottsfunktionen) automatiskt att köras varje gång 16-bitarsräknaren har uppnått ett förinställt värde, maxvärdet.

TCCR1A, Timer/Counter Control Register 1A:
Här bestäms om och hur denna enhet direkt ska kunna påverka utgångar. (s 107-109 i databoken). Lägg märke till hur bitarna är inställda från början.

TCCR1B, Timer/Counter Control Register 1B:
Detta register innehåller bl.a. biten WGM12 (bit 3) som styr om räknaren ska nollställas automatiskt när den nått maxvärdet. Med de tre lägsta bitarna [CS12, CS11, CS10] kan processorns klockfrekvens delas så att den här timern inte går så fort. Titta i databoken sidan 110! Om vi t.ex. ställer in de tre bitarna på [0, 1, 1], så kommer en klockfrekvens på 1 MHz att delas med 64 så att vi får en frekvens på 15625 Hz.

OCR1A, Output Compare Register 1A:
Räknarens maxvärde (s 111 i databoken). För att ställa in en speciell avbrottsfrekvens (hur många avbrott som ska komma i sekunden), ska alltså klockfrekvensen, 1 MHz, delas ner med hjälp av de tre delningsbitarna i TCCR1B och värdet i OCR1A. Alltså: Om vi mot förmodan skulle vilja ha avbrott en gång per sekund, så kan vi exempelvis ställa in de tre delningsbitarna på [0, 1, 1] och sätta värdet i OCR1A till 15625.

Fundera på följande:

- Om vi har en klockfrekvens på 1 MHz, vad blir då den lägsta avbrottsfrekvens som man kan använda? Hur ställer man in för att få den?

……

- Och vad blir den högsta frekvensen?

……

Laborationsuppgifter

Uppgift 1. Inställningar för avbrottet.

Komplettera funktionen initOC1 så att den initierar avbrott så de kommer var 10:e ms. 1 MHz ska alltså delas ner så vi får 100 Hz. Se tipsen på sidan 3!

```
void initOC1(){              // Output Compare 1 (s 94-113)
    TIMSK =                ; // Aktivera avbrottet
    TCCR1A = 0;              // Normal användning
    TCCR1B =               ; // Nollställ vid OC, systemklocka/?
    OCR1A =                ; // 1250x8=10000 (10 ms vid 1 MHz)

}
```

❑

Uppgift 2. Avbrottsfunktionen.

Avbrottsfunktionen som utförs automatiskt med jämna mellanrum skrivs så här:

```
ISR (SIG_OUTPUT_COMPARE1A){
  if((knappen nere nu)&&(knappen uppe förra gången)){
    Utför uppgiften (öka n med ett; om n blir 10 sätt n till noll; anropa
                     utskriftsfunktionen med n som inparameter.)
}
```

`ISR` betyder ”Interrupt Service Routine”.
`SIG_OUTPUT_COMPARE1A` är namnet på avbrottsfunktionen, och den **måste** heta så!.
Det gäller nu att skriva **`if`**-satsen i avbrottsfunktionen.

```
((knappen nere nu)&&(knappen uppe förra gången))
```

Sanningshalten i **`(knappen nere nu)`** kan avgöras genom att direkt läsa av knappen.
Hur skriver du detta?

…….

Sanningshalten i (**`knappen uppe förra gången`**) kan du bara avgöra om du spar hur det såg ut förra gången dvs. för 10 ms sedan. Du behöver en variabel för att komma ihåg detta.

Regler för variabler i avbrottsfunktioner:

1. *Använd alltid typen volatile*
2. *Använd globala variabler eller static*

Ditt förslag hur detta ska lösas:

……...

❑

Uppgift 3. Testa programmet

Innan du kan kompilera programmet, måste ett par tillägg göras i koden.

1. För att kompilatorn ska kunna översätta avbrottsrutinen rätt, måste man lägga till **`#include <avr/interrupt.h>`**. Titta gärna i den filen, men den är lite svår att förstå sig på. Men en sak man bör veta är att den ser till att ett hopp till en funktion med namnet `SIG_OUTPUT_COMPARE1A` läggs in i programminnet på adress 0x000E. Dit tvingas nämligen programmet var 10:e ms. Och det är därför vår avbrottsfunktion måste ha detta namn.
2. Hela avbrottssystemet måste sättas igång genom att ettställa bit 7 i registret **`SREG`**.
    - Gör dessa kompletteringar!
    - Kompilera programmet och testa!

Det ska nu gå att räkna upp displayen genom att trycka på knappen.

❑

Uppgift 4. Testa gränserna!

Ändra nu inställningarna i initOC1 så du får längsta tänkbara intervall mellan avbrotten.

Hur ofta läses knappen av nu?.....

Ändra nu så att du får kortast tänkbara intervall mellan avbrotten.

Hur ofta läses knappen av nu?.....

Uppgift 5. Experiment.

Ändra tillbaka ditt program så att du återigen har 10 ms mellan avbrotten.

Förändra det så att uppräkningar istället sker varje gång man släpper upp knappen.

Förslag:…….

…….

…….

Försök få programmet att räkna upp både vid nedtryckning och vid uppsläpp.

Förslag:…….

…….

…….

…….

…….