Namn:
Laborationen godkänd:

LUNDS TEKNISKA HÖGSKOLA
Lunds universitet
LTH Ingenjörshögskolan vid Campus Helsingborg

Digitala system 15 p

Datorprojekt, del 3

Projektlaboration 3, hantering av klocka.

Laborationsuppgifter:

Laborationens mål är att få displayen att visa timmar, minuter och sekunder samt att kunna ställa klockan.

Uppgift 1.

Det periodiska avbrottet som användes i laboration 5 är lämpligt att använda till klockan. Kopiera funktionerna

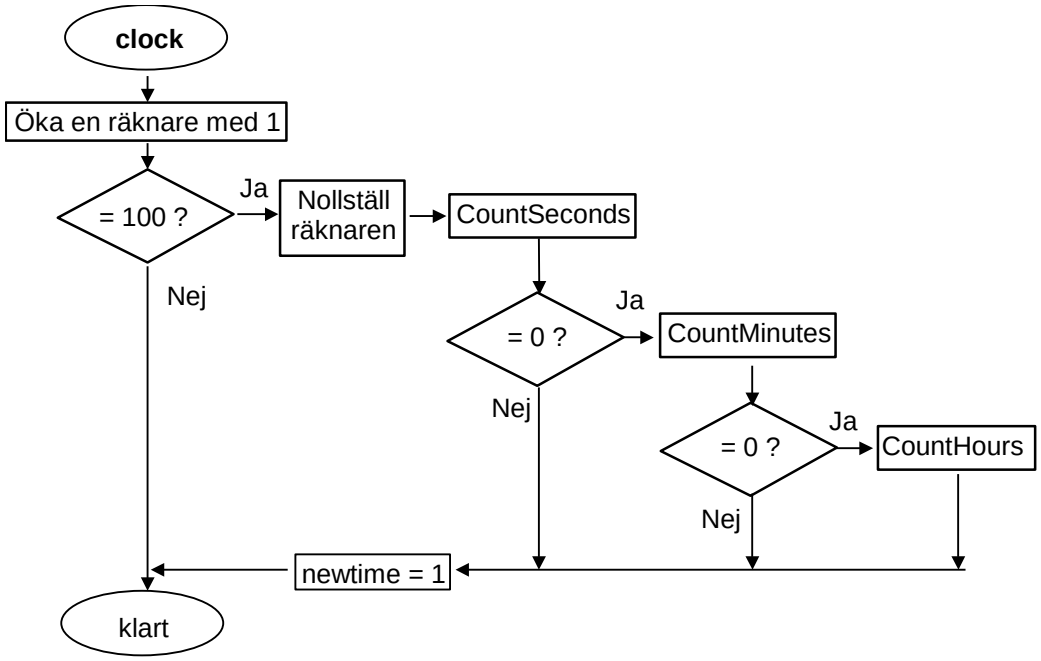
- **`initOC1`** och
- **`ISR(SIG_OUTPUT_COMPARE1A)`**

och placera dem i en ny fil, **AVR-MT-128-clock.c.** I laboration 5 användes en annan mikroprocessor och en annan klockfrekvens. Nu använder vi 16 MHz istället för 1 MHz. Du måste justera en av inställningarna i **`initOC1`** så att avbrottet fortfarande kommer var 10:e ms. Observera också **`#include`**-raden som måste vara med när vi har en fil med avbrottsfunktion.

Erik Larsson

Uppgift 2.

I filen **AVR-MT-128-clock.c** ska du nu skriva en ny funktion, **clock()**, som gör följande:



Funktionen ska anropas i avbrottsfunktionen och räkna upp variablerna *counter*, *seconds*, *minutes* och *hours* enligt flödesplanen ovan. Dessutom ska en variabel *newtime* användas som flagga att klockan ändrat värde. Den får ungefär samma funktion som bitarna i variabeln new_buttons i den första projektlabben.
Alla dessa variabler utom counter ska vara tillgängliga för omvärlden. Därför ska du ha en header-fil, **AVR-MT-128-clock.h**, där de deklareras.

Variablerna ska fungera såhär:

- **counter**: En räknare som räknar 100 avbrott. När counter = 100 ska den nollställas och sekunderna ska räknas upp.
- **seconds**: Räknas upp med ett varje gång. Om sekunderna fått värdet 60 ska de nollställas och minuterna ska räknas upp.
- **minutes**: Räknas upp och nollställs på samma sätt som sekunderna.
- **hours**: Räknas upp och nollställs vid värdet 24.
- **newtime**: Ettställs varje gång klockan (sekunderna) ändrat värde.

Tänk på att variablerna definieras som **volatile**!

Testa programmet!

□

Uppgift 3.

Använd dina funktioner från förra laborationen för att skriva ut klockan:

```
14:35:06
```

Skriv ut den i mitten på den översta raden. Lägg utskriften i en funktion, **writeClock**, som du anropar ***i huvudprogrammet*** varje gång det är dags att skriva ut tiden.
Funktionen placeras i filen **AVR-MT-128-clock.c**.
Om du inte vill att markören ska stå och blinka efter sekundsiffran, så kan du avsluta med att placera markören på en plats där den inte syns. I uppgift 4 gör du en sådan funktion.

Skriv funktionen, komplettera AVR-MT-128-clock.h och testa!

□

De återstående uppgifterna i denna laboration går ut på att kunna ställa klockan.
I nästa laborationsavsnitt ska vi lägga till den programkod som gör att en tidsignal kan tas emot från en fristående enhet.

Manuell ställning av klockan.

Man kan tänka sig många olika sätt att ställa klockan.
Här är ett par exempel:

1. Använd 3 knappar för att öka värdet på timmar, minuter och sekunder.
   Låt **`[left]`** räkna upp timmar, **`[middle]`** räkna upp minuter och **`[right]`** sekunder.

2. Låt markören vara placerad på en av siffrorna. Då ska du kunna ändra den siffran genom att trycka på **`[up]`** eller **`[down]`** och sen med knapparna **`[left]`** och **`[right]`** positionera markören på önskad siffra.

Vi väljer alternativ 2 eftersom det fungerar mycket tydligare.

Mittenknappen ska vi till att börja med använda för att få klockan att gå till och från tillståndet ”running”.

Uppgift 4. En funktion för att placera markören på en bestämd plats.

Uppenbarligen behöver vi en funktion som placerar markören på önskad plats. När vi ställer klockan ska ju markören märka ut det tal (timmar, minuter eller sekunder) som ska ökas eller minskas.

```
void LCD_PlaceCursor(unsigned char position){

     ????????????????????????????????????
     ????????????????????????????????????

}
```

Om du tänker efter så har du redan skrivit denhär koden!
Funktionen läggs lämpligen i filen **AVR-MT-128-LCD.c**

Skriv funktionen, komplettera AVR-MT-128-LCD.h och testa funktionen!

□

Uppgift 5. Huvudprogrammet.

Som det mesta av den elektronik vi har omkring oss, så ska den här klockan kunna vara i några olika tillstånd, ”moder” . Det gör det lätt att hantera den.

Tillstånd ”**`running`**”:

1. Om en sekund har gått: skriv ut klockan.
2. ”`new_left`”: nästa tillstånd ska bli ”`set_hours`” och markören placeras på timmarnas entalssiffra.

Tillstånd ”**`set_hours`**”:

1. ”`new_up`”: timmarna räknas upp med ett, klockan skrivs ut och markören sätts på timmarnas entalssiffra.
2. ”`new_down`”: timmarna räknas ned med ett, klockan skrivs ut och markören sätts på timmarnas entalssiffra.
3. ”`new_right`”: nästa tillstånd ska bli ”`set_minutes`” och markören placeras på minuternas entalssiffra.
4. ”`new_middle`”: nästa tillstånd ska bli ”`running`”

Tillstånd ”**set_minutes**”:

1. ”new_up”: minuterna räknas upp med ett, klockan skrivs ut och markören sätts t på minuternas entalssiffra.
2. ”new_down”: minuterna räknas ned med ett, klockan skrivs ut och markören sätts på minuternas entalssiffra.
3. ”new_left”: nästa tillstånd ska bli ”set_hours” och markören placeras på timmarnas entalssiffra.
4. ”new_middle”: nollställ sekunderna, nästa tillstånd ska bli ”running”

Här är en lämplig struktur för den oändliga slingan i huvudprogrammet:

```
switch(state){

    case running:
        ??????????????????????????????????????
        ??????????????????????????????????????
        ??????????????????????????????????????
        ??????????????????????????????????????
    break;

    case set_hours:
        ??????????????????????????????????????
        ??????????????????????????????????????
        ??????????????????????????????????????
        ??????????????????????????????????????
    break;

    case set_minutes:
        ??????????????????????????????????????
        ??????????????????????????????????????
        ??????????????????????????????????????
        ??????????????????????????????????????
}// end switch
```

Variabeldeklarationen för tillståndsvariabeln ”state” är enklast att lägga i en headerfil. Du kan göra en headerfil för huvudprogramfilen, AVR-MT-128.h och lägga den där. I den filen kan du också lägga annat som tillhör huvudprogramfilen och som andra filer kan behöva känna till.

```
enum {running, set_hours, set_minutes} state;
```

Bonusuppgift : Gör ett stoppur eller en väckarklocka.

Programmerarens bevingade ord

1. Konstigt.
2. Det har ju fungerat förut!
3. Det här kan inte påverka det där.
4. Men jag trodde att jag hade fixat det.
5. Hur kunde det bli så?
6. Det måste vara något fel på kretsen.
7. Dom kanske har bytt version?
8. Nu ska det fungera!
9. Men jag har inte ändrat nånting.
10. Jodå, det blir klart till dess.
11. Det är bara ett skönhetsfel.
12. Jag är nästan klar.
13. Jag tänkte inte på det.
14. Det tar ingen tid.
15. Det är bara några småsaker som ska fixas.
16. Det hade jag ingen aning om.
17. Man kan inte testa allting!
18. Det är med, det är bara inte testat.
19. Egentligen fungerar det, fast det inte verkar så.
20. Frånsett att det inte fungerar, hur tycker du att det verkar?
21. Programmet är helt rätt, men det funkar inte.
22. Hur skulle jag kunna veta det?
23. Det känns som det ska fungera.
24. Skit!!!
25. Hjälp!!!
26. Bara för att det fungerar behöver det inte vara rätt!