Datorteknik DALIA Lab 3
Digitalur — v0.5

Inledning

I den här laborationen skall du konstruera ett digitalur som visar minuter och sekunder.

Du kommer att använda en modul med fyra sammankopplade 7-segmentsdisplayer för att visa siffrorna. Denna gång behöver du själv styra varje segment på displayerna, det finns ingen inbyggd avkodning i modulen.

Tiden skall hållas exakt med hjälp av en tidbasmodul som ger en puls varje sekund. Modulen ger också pulser med frekvensen 10, 100 och 1000 Hz. De högre frekvenserna skall du använda för att *multiplexa* displayen.

Laborationsuppgifter

Laborationshandledningen beskriver inte någon detaljerad rekommenderad arbetsgång. Du får själv tänka ut tillvägagångssätt och lämpliga tester för att avgöra om programrutinerna fungerar.

Gå inte vidare utan att först vara *helt övertygad* om att dina programrutiner fungerar som avsett. Det är mycket lättare att rätta till fel tidigt i laborationen.

Även om arbetsgången inte är bestämd i detalj finns det nödvändiga förberedelseuppgifter.

Förberedelseuppgifterna skall vara klara innan du påbörjar laborationen och du får möjlighet att redovisa dem för laborationsassisten under laborationens gång.

Laborationsuppgiften är att sätta ihop programmets delar och hårdvaran till en fungerande klocka.

Under laborationens gång kan du behöva göra andra kopplingar och/eller skriva kod för att kontrollera att allt fungerar som det ska.

Samtliga förberedelseuppgifter är nödvändiga för en fungerande klocka. Du förväntas ha förberedd och väl simulerad kod med dig till laborationen.

Förberedelseuppgift 1 Skriv kod för att skapa de flanktriggade avbrotten `INT1` och `INT0`. Prova innan att avbrottskoden gör det den ska. Du får själv avgöra om du vill ha avbrottet på stigande eller fallande flank.

Obs. Icke inkopplade avbrottsingångar på kortet är ”passiva” och kan inte generera okynnesflanker.

Förberedelseuppgift 2 Skriv en avbrottsrutin, `BCD` som räknar upp tiden enligt *Tidskodning och uppräkning*. Tiden skall återfinnas BCD-kodad i fyra minnesceller med början i adressen `TIME` i SRAM. Med avbrottsfrekvensen 1 Hz kommer denna rutin räkna upp sekunder och minuter.

Förberedelseuppgift 3 Skriv en avbrottsubrutin, `MUX`, som visar rätt siffra på rätt position. För att visa en siffra måste sjusegmentmodulens A–G-ingångar förses med lämplig information. Det bitmönster som skall skickas ut till respektive siffra skall finnas i en tabell i programminnet (FLASH).

Förberedelseuppgift 4 (Extra) Läs om *timers* i databladet. Använd sedan timeravbrott för att skapa MUX- eller BCD-avbrottet (eller båda!)

Översättningen mellan BCD och sjusegmentmodulens A–G-ingångar skall vara sådan att BCD-värdet 0 resulterar i det binära värdet `$3F` som behövs för att tända de segment som visar siffran `0`.

För att säkerställa att rätt siffra visas på rätt position på displayen måste dessa vara synkroniserade.

Tips! **Ett sätt** kan vara att använda en separat modulo-4-räknare (som ger sekvensen 0→1→2→3→0...) och addera BCD-siffrornas basadress (t ex `TIME`=`$100`) till denna. Då kan modulo-4-sekvensen styra vilken siffra på displayen (0..3) som skall tändas och innehållet i `$100`–`$103` peka ut det data som skall användas.

Ett annat sätt är att använda en adressräknare direkt. Om denna adressräknare går igenom sekvensen `$100`→`$101`→`$102`→`$103`→`$100`..., kan adressens två minst signifikanta bitar användas för att styra displayens multiplexingångar (`00`, `01`, `10`, `11`, `00`,...).

Uppgift Sätt ihop programmets delar och anslut avbrottssignalerna till en fungerande klocka.

Uppgift (Extra) Ersätt tidbasmodulen med interna timers.

Hårdvara och kod

Till laborationen används enbart två hårdvarumoduler:

- Tidbasmodul för att förse konstruktionen med de sekund- och multiplexpulser som behövs.
- Displaymodul med fyra 7-segments displayer som kan väljas individuellt.

Det finns även två avstudsade tryckknappar och lysdioder som kan användas vid felsökning.

Programkoden skall skrivas i enlighet med god programmeringssed, dvs:

- Rutinerna skall ha minimal påverkan på omgivningen,
- Använd så få register som möjligt,
- Använd så lite minne om möjligt,
- Tänk strukturerat, använd subrutiner, och
- Ha vettiga kommentarer.

Multiplexning

Displaymodulens konstruktion tillåter oss att visa endast en siffra i taget. Med *multiplexning* kan vi visa siffror i samtliga positioner.

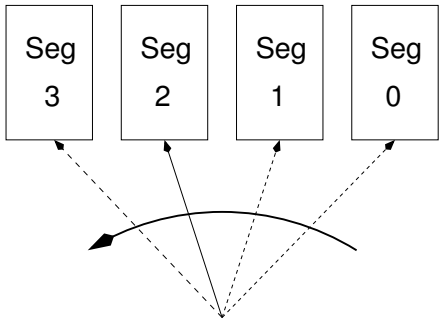
Multiplexning går till så att en siffra visas en stund, sedan växlas snabbt över till nästa siffra, så visas denna en stund, och så vidare, tills alla siffror visats, varefter det hela börjar om från början igen.

Görs detta tillräckligt snabbt kommer ögat att uppfatta det som att alla positioner lyser hela tiden.[1]

Praktiskt skall multiplexningen utföras med en avbrottsrutin med följande innehåll:

1. Bestäm nästa position med signalerna `A` och `B` på displaymodulen.
2. Lägg på önskat bitmönster på segmentingångarna `A`–`G`.
3. Gå ur avbrottet

I programmet är det under tiden *mellan* två avbrott som en siffra ska lysa. Avbrottet ombesörjer enbart växlingen mellan siffrorna.



Med multiplexning visas varje siffra bara så länge som behövs för att de inte skall flimra.

Beroende på hur du löst uppgiften kan man ibland skymta att även andra segment lyser upp. Detta försämrar kontrasten hos siffrorna och är inte önskvärt. Vidtag isåfall en lämplig åtgärd mot detta! (Ofta behöver bara en assemblerrad läggas till eller möjligen flyttas).

[1] Eftersom siffran nu i verkligheten blinkar, blir det något svagare ljus än om man tillät den att lysa kontinuerligt, men det är i modulen redan kompenserat genom att man ökat diodströmmen något.

Tidskodning och uppräkning

Tiden skall återfinnas BCD-kodad i fyra minnesceller med början i adressen `TIME` i SRAM.[2,3]

Endast de fyra lägsta bitarna i varje byte skall innehålla information om vilken *decimal* siffra byten representerar. Fyra bitar kan rymma siffrorna `0`–`F`, med BCD-kodning tillåter vi dock enbart siffrorna `0`–`9`.

Tiden ”04:53” (4 minuter och 53 sekunder) anges i fyra, efter varann följande, bytes i minnet enligt

0**0**	0**4**	0**5**	0**3**

Minnesadressen `TIME` pekar på sekundsiffran, här ”3”. `TIME`+1 pekar på tiosekundsiffran ”5” osv.

BCD-rutinen skall räkna upp tid som en klocka enligt tabellen till höger.

Tid	`TIME+3`	`TIME+2`	`TIME+1`	`TIME+0`
00:00	0	0	0	0
⋮	⋮	⋮	⋮	⋮
00:09	0	0	0	9
00:10	0	0	1	0
00:11	0	0	1	1
⋮	⋮	⋮	⋮	⋮
00:19	0	0	1	9
00:20	0	0	2	0
00:21	0	0	2	1
⋮	⋮	⋮	⋮	⋮
00:59	0	0	5	9
01:00	0	1	0	0
01:01	0	1	0	1
⋮	⋮	⋮	⋮	⋮
59:59	5	9	5	9

Tiduppräkningen skall utföras med 60 sekunder per minut och 60 minuter per timme varefter den börjar om igen.

Avslutning

Vi har nu två program som körs helt oberoende av varandra: `BCD`-avbrottsrutinen körs på anmodan av den yttre sekundpulsen medan `MUX`-avbrottsrutinen körs oftare och helt ovetande om `BCD` överhuvudtaget!

Huvudprogrammet exekveras så fort processorn kan men utför inga ”nyttiga” instruktioner, digitaluret utgörs helt och hållet av de två avbrottsrutinerna. Även huvudprogrammet körs helt oberoende och ovetande om avbrottsutinerna.

Den noggranna externa sekundpulsen ser till att klockan går rätt oberoende av vad processorn håller på med i övrigt.

En fördel med avbrott är att vi kan låta yttre hårdvara skapa dessa avbrott med en mycket högre noggranhet än vad processorn själv skulle kunnat. Processorns möjlighet att mäta tid, med till exempel tidsfördröjande loopar, är begränsad av både processorklockans noggrannhet och instruktionernas exekveringstid.

Bedömning för examination

En godkänd lösning måste uppfylla kriterierna nedan.

1. Generellt gäller att uppgiften skall lösas. Den presenterade koden skall dessutom vara läslig och utformad enligt vår kodstil, dvs

- att koden är indenterad.
- att subrutiner
  - har bra namn,
  - har nödvändiga lokala variabler,
  - avslutas på **ett** ställe, längst ner i rutinen.

2. I denna laboration speciellt:

- I ett korrekt program lyser siffrorna med jämn intensitet.
- Hårdvaruinitieringen skall utföras i separat rutin.
- BCD-uppräkningen skall loopas. Tiden skall vara i SRAM.
- MUX-rutinen skall styras av position i SRAM.
- En slutlig kodstorlek på cirka 100–150 rader är rimlig.

-o-O-o-

[2]Det kan finnas fördelar att låta adressens lägsta nibble vara `0`.
[3]Det kan också finnas fördelar att högsta byten i adressen är `00`.