# Laboration 4: Undantagshantering

Under denna laboration utförs sex uppgifter varav fyra behandlas utförligt utförligt i läroboken där de beskrivs i form av deluppgifter som succesivt måste klaras av innan själva laborationsuppgifterna kan lösas.

För att kunna genomföra hela laborationen måste du ha arbetat igenom hela kapitel 6 i läroboken. Du kan dock dela upp det så här:

- Studera avsnitten t.o.m 6.2 och uppgifter t.o.m 6.3 utför därefter laborationsuppgifter 4.1 och 4.2.
- Fortsätt därefter med avsnittet 6.3. Studera speciellt exemplen och utför uppgifterna 6.4 och 6.5. Utför därefter laborationsuppgift 4.3.
- Avsnitten 6.4 och 6.5 kan du sedan läsa kursivt. Dom är inte centrala för förståelsen av nästa uppgift.
- Avsnitt 6.6 och 6.7 är grundläggande för laborationsuppgifter 4.4 - 4.6. Försäkra dig om att du förstår avsnitten. Utför därefter dessa laborationsuppgifter.
- I den avslutande laborationsuppgiften 4.6 kombinerar du slutligen lösningar av tidigare uppgifter i denna laboration för att skapa ett *Stoppur*.

## Laborationsuppgift 4.1: Meddelandeskickning (Uppgift 6.3 i läroboken)

- Uppgiften är att konstruera en avbrottsdriven fördröjningsrutin med *SysTick*-räknaren.
- Kontrollera programmets funktion med *CodeLite* och *SimServer*.

## Laborationsuppgift 4.2: Meddelandeskickning utan blockering

För att belysa *väntetiden*, ska du nu utgå från laborationsuppgift 4.1, och lägga till kod som illustrerar hur mycket ”att göra under tiden” kan vara. Förändringen påverkar huvudsakligen huvudprogrammets utformning:

```
    if( systick_flag )
      break;
/* Här placeras kod som kan utföras under väntetiden
Lägg till en diodramp för utmatning från port D(bit 8..16). Inför en variabel som ökas med 1
varje gång i programslingan och skrivs till den nya diodrampen.
*/
```

Utöver förändringen i huvudprogrammet krävs alltså också att hela porten initierats för utmatning.

- Skapa ett nytt projekt `systick_nonblocking`.
- Exekvera programmet och observera speciellt hur mycket som hinner utföras under ”väntetiden”.

## Laborationsuppgift 4.3: Realtidsklocka (Uppgift 6.5 i läroboken)

- Uppgiften är att konstruera en simulerad realtidsklocka och justera denna mot ”verklig tid” i det värddatorsystem du använder.
- Kontrollera programmets funktion med *CodeLite* och *SimServer*.

Anm: Modifiera testprogrammets slinga enligt:

```
nbcd = ((seconds/10) << 4);
nbcd |= ((seconds % 10 ) & 0xF);
*GPIO_D_ODRLOW = nbcd;
```

## Laborationsuppgift 4.4: Externavbrott, en avbrottsvektor (Uppgift 6.9 i läroboken)

- Uppgiften är att konstruera en applikation med flera yttre enheter som genererar avbrott och där samma avbrottsrutin används för att hantera samtliga avbrott.
- Kontrollera programmets funktion med *CodeLite* och *SimServer*.

---

## Laborationsuppgift 4.5: Externavbrott, flera avbrottsvektorer (Uppgift 6.10 i läroboken)

- I denna uppgift modifieras föregående applikation så att varje avbrott dirigeras till en egen avbrottsrutin.
- Kontrollera programmets funktion med *CodeLite* och *SimServer*.

---

## Laborationsuppgift 4.6: Stoppur

Ett *stoppur* ska konstrueras. För detta använder vi en MD407 och två typer av laborationskort: IRQ Flip Flop med en inbyggd räknarkrets och 2 st visningsenheter 7 segment display, se figur 1.


*Figur 1*

Stoppuret kontrolleras med två brytare. Överst till vänster finns *start/stopp*-funktionen. Första nedtryckningen startar klockan, nästa stoppar den, ytterligare nedtryckning startar den igen osv.

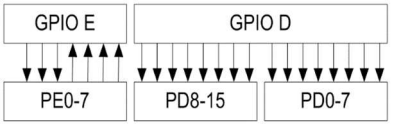
Nedanför denna brytare finns *återställningsfunktionen*. En tryckning på denna knapp återställer tiden till 00 00.

För tidmätningen används laborationskortets inbyggda räknarkrets med frekvensen 10 Hz. Stoppurets mätperiod blir därför max 9 minuter 59,9 sekunder och upplösningen 1/10-dels sekund. Tiden visas på de fyra sifferindikatorerna från vänster enligt:

- minuter
- 10-tal sekunder
- 1-tal sekunder
- tiondels sekunder

Om mätningen uppnår maximal tid (95 99) ska den stannas och stå kvar i detta läge ända tills återställningsfunktionen aktiveras.
Vid en återställning nollställs tiden oavsett om stoppuret har startats eller ej.

Figur 2 visar hur laborationskorten ansluts till laborationsdatorn.




*Figur 2*

Alla *händelser* dvs. knapptryckningar och indikation av periodintervall från räknarkretsen, ska hanteras med processorns *avbrottsmekanismer*.

- Skapa ett nytt projekt `stopwatch`.
- Implementera stoppuret enligt specifikationen.
- Kontrollera programmets funktion med *CodeLite* och *SimServer*.

Anm: Tänk på att IO-simulatorns klockfrekvens är betydligt lägre än hårdvaran, 1/10 sekund motsvarar snarare 1 sekund i simulatorn. Se även klippet med demonstration av uppgiften.