# LABORATION

## Datorteknik Y

*Avbrottsprogrammering på Darma*

Version 4.4
Februari 2022 (OA, KP)

| **Namn och personnummer** | **Godkänd** |
|---|---|
| | |
| | |

# 1 Inledning

Syftet med laborationen är först att ge övning i avbrottsprogrammering (del A) samt därefter att tillämpa dessa kunskaper på ett större problem (del B). Syftet är också att skapa större förståelse för stackhantering i en processor.

Efter genomförd laboration ska du, med hjälp av manual över Darma, kunna skriva ett assemblerprogram som kommunicerar med omvärlden via avbrottsstyrd GPIO-port.

Till laborationen ska du ha med dig egna lösningsförslag på alla laborationsuppgifter. Till del B bör det även finnas ett flödesschema. Tänk på att utnyttja eventuella labförberedande lektioner som kan komma att ges i anslutning till laborationstillfället.

Extrauppgifterna görs i mån av tid och intresse.

# 2 Färdiga subrutiner

För att kunna fokusera på avbrottshanteringen finns det ett antal färdiga subrutiner. Dessa subrutiner är definierade i labskelettet som kan laddas ned från laborationshemsidan. Följande subrutiner ska användas.

- `inituart`: Initiera uart för utskrift till datorn
- `initGPIOB`: Initiera port B som utgång
- `initGPIOD`: Initiera port D som ingång
- `initGPIOE`: Initiera port E som ingång
- `initint`: Sätter avbrottnivå 5 på port E pin 4 och avbrottsnivå 2 på port D pin 7
- `SKBAK`: Skriver ut texten “Bakgrundsprogram”
- `SKAVV`: Skriver ut texten “AVBROTT vänster”samt rader med ’-’-tecken och en kolumn av stjärnor
- `SKAVH`; Skriver ut texten “AVBROTT höger”samt rader med ’=’-tecken och en kolumn av stjärnor
- `DELAY`: Fördröjning, register r1 anger antal millisekunder

Dessa subrutiner finns i labskelettet lab2.asm på kurshemsidan. Dessutom behöver filen ccstudio_startup.c bytas ut till den nya ccstudio_startup.c som finns att ladda ned på kurshemsidan.

# 3 Laborationsuppgift Del A

Del A av laborationen består i att göra grundläggande undersökningar av ett mycket enkelt och schematiskt avbrottssystem för Darma. Ett program som skriver texten “Bakgrundsprogram”på terminalen skall kunna avbrytas av två knappar, en som ger ett avbrott på nivå 5 och en som ger ett avbrott på nivå 2.

## 3.1 Uppgift 1

Skriv ett bakgrundsprogram som initierar systemet och sedan går i en oändlig slinga där texten "BAKGRUNDPROGRAM"skrivs på skärmen med en sekunds paus mellan varje utskrift. Använd de befintliga subrutinerna för utskrift och fördröjning.

För att senare i labben enklare kunna se vilket register som skrivit vilket värde på stacken, är det också rekommenderat att du sätter r0-r12 (om dom inte används till annnat) till värdena r0=0x00010203, r1=0x10111213, r2=0x20212223, r3=0x30313233, r4=0x40414243, ... r12=0xc0c1c2c3 innan initieringssubrutinerna anropas i main. Initiering behövs framförallt för uppgift 2 och framåt, och är enligt följande:

- Initiera uart, GPIOD, GPIOE och avbrott med hjälp av de färdiga subrutinerna
- Möjliggör avbrott i processorn.

För att förbereda laborationen utan tillgång till lablokalen (och vid distansläge) kan istället en speciell hårdvara användas. Den består av ett TiVA C LaunchPad-kort (det röda kortet) tillsammans med en Arduino Uno som styrs via programmet tsea28lab2. Programmet tsea28lab2 styr då Arduino Unbo som skickar knapptryck och visar utsignaler från port B.

Jorda avbrottsingångarna, dvs anslut Port D pin 7 och Port E pin 4 till GND på Darma. Om du kör labben på distans motsvarar detta att ingen knapp trycks ned. Provkör ditt program. Tag sedan bort jordanslutningarna till Port D och Port E och anslut istället de två tryckknapparna. Den vänstra knappen skall anslutas till Port E pin 4 och den högra till Port D pin 7, så att vänster knapp ger avbrott på nivå 5 och höger på nivå 2.

Denna uppgift behöver inte redovisas.

## 3.2 Uppgift 2

Skriv de två avbrottsrutinerna av vilka den ena ska ge utskriften "AVBROTT vänster"och den andra utskriften "AVBROTT höger". Resepektive avbrottsflagga (men bara den bit som respektive tryckknapp är ansluten till) i GPIO-portarna ska nollställas innan utskriften påbörjas, så att ytterligare en avbrottsbegäran kan tas emot under tiden avbrottsrutinen körs.

I en verklig tillämpning sparar man normalt undan en del interna register på stacken. Vi skall emellertid i laborationen gå in och titta på stackens innehåll, och för att göra det lätt att hitta och inte få för mycket ovidkommande data på stacken skall du inte spara några andra register än LR i dina avbrottsrutiner. Ett antal register sparas automatiskt på stacken vid avbrott, och det kan du naturligtvis utnyttja i dina rutiner. De färdiga subrutinerna, SKAVV och SKAVH, är skrivna så att de inte förstör varandras registerinnehåll.

Kör huvudprogrammet och kontrollera funktionen hos ditt system enligt följande:

- Tryck på vänster knapp

- Vänta tills avbrottsrutinen är klar
- Tryck på höger knapp

Om programmet uppgförde sig som väntat gör du följande test:

- Tryck på vänster knapp
- Vänta någon sekund
- Tryck i snabb följd (ca 0.5s mellan varje tryck) höger, vänster, höger.

Om dina avbrottsrutiner är korrekt skrivna skall samtliga avbrott (totalt fyra stycken) tas om hand. Rita ett enkelt diagram som visar i vilken ordning begäran om avbrott kommer in, och i vilken ordning avbrottsrutinerna anropas. Förklara orsaken till att avbrotten inte åtgärdas i den ordning de begärs.

.....

## 3.3 Uppgift 3

Du kan nu genom tryckningar på knapparna på olämpliga ställen, få texten Bakgrundsprogram att bli osammanhängande. Vi anser det vara väldigt viktigt att utskriften inte avbryts mitt i ett ord, utan kan bara acceptera avbrott i pauserna mellan utskrifterna. Vad kan man göra för att få programmet att bete sig så? Gör det! (GPIO-portarna skall inte programmeras om.)

.....

## 3.4 Uppgift 4

Tryck på paus-knappen (suspend) i Code Composer Studio mellan två utskrifter av Bakgrundsprogram. Programmet befinner sig nu med största sannolikhet i väntesubrutinen DELAY och stackdjupet blir då två ord (8 bytes). (Med detta menas att det ligger 8 bytes på stacken, och att stackpekarens värde alltså är 0x200001f8 om stackpekaren var 0x20000200 från början.) Om du inte skulle få stackdjupet 8 bytes när du pausar, försök igen.

Undersök i memory browser innehållet på stacken. Identifiera vad det är som ligger där och varifrån det kommer. Ledning: titta på vad som händer först i subrutinen DELAY. Du kan även byta presentationsläge från 32-bit Hex till 8-bit Hex för att se minnesinnehållet byte för byte.

| Stack Adress (SP) | Data som 32-bitars värde | Data som byte | | | |
|---|---|---|---|---|---|
| | | SP | SP+1 | SP+2 | SP+3 |
| | | | | | |
| | | | | | |

## 3.5 Uppgift 5

Sätt nu en brytpunkt på den andra instruktionen i subrutinen SKAVH (adr r3, Righttext). Kör programmet och tryck i följd mellan två utskrifter av Bakgrundsprogram först på vänster knapp, och sedan i pausen direkt efter stjärnorna men innan nästa rad börjar skrivas ut på höger knapp. Om du inte har en väldig otur kommer såväl bakgrundsprogrammet som vänster avbrottsrutin att bli avbrutna i subrutinen DELAY. Stackdjupen blir då 24 ord (96 bytes). Om du skulle få ett annat stackdjup, starta om och försök igen. Vad bör stackpekaren ha för värde?

.....

Läs återigen av vad som finns på stacken, hela vägen ner till botten, och identifiera exakt vad som ligger där och varifrån det kommer. Rita även en enkel figur som med pilar visar de hopp som sker mellan de olika rutinerna i programmet innan brytpunkten nås. Förklara anledningen till samtliga hopp och vad som lagras på stacken i varje steg.

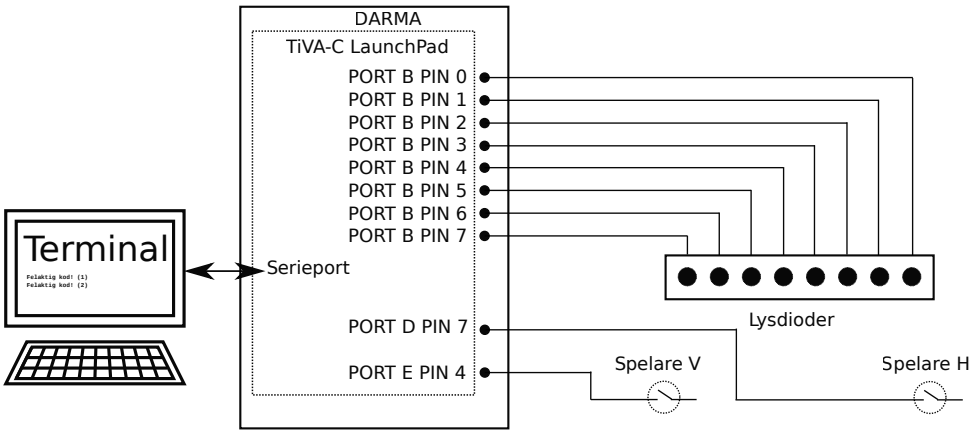
Notera först vilka adresser SKBAK, SKAVV, SKAVH, DELAY, och slowprint har. Dessa kan du hitta med hjälp av disassembly view.

| Funktion | Adress till funktion |
|---|---|
| intgpiod | |
| intgpioe | |
| SKBAK | |
| SKAVV | |
| SKAVH | |
| DELAY | |
| slowprintstring | |

| Adress | Data (32-bit) | Register som värdet kommer från | Vilken instruktion/händelse (t ex avbrott, push, etc.) orsakar kopiering till stack (inkl. adress för instruktioner) |
|---|---|---|---|
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| 0x200001fc | | | |

På vilken adress blev bakgrundsprogrammet avbrutet? På vilken adress blev den vänstra avbrottsrutinen avbruten?

.....




Figur 1: Inkoppling av lysdioder och tryckknapp för ping-pong spel

### 3.6 Uppgift 6 (Extrauppgift)

Antag samma situation som i uppgift 5, men utan brytpunkt. Vad blir det maximala stackdjupet?

.....

## 4 Laborationsuppgifter Del B

### 4.1 Uppgift 7

Konstruera ett PING-PONG spel. Använd två studsfria tryckomkopplare som ansluts port D pin 7 respektive port E pin 4, samt åtta lysdioder som ansluts till port B, se figur 1. "Bollen"markeras med en tänd lysdiod, och ska förflyttas fram och tillbaka över lysdiodarrayen så länge de bägge spelarna trycker ned sin knapp exakt då bollen befinner sig i respektive ändläge. Poäng ska lagras för respektive spelare i var sin adress i minnet.

#### 4.1.1 Spelregler

Den som vinner en boll får serva. Serve markeras med stillastående boll på den servandes sida. Bollen börjar att röra sig då knappen trycks ned. Spelare V börjar att serva vid spelets start. Om en spelare trycker antingen för tidigt eller för sent

när bollen är i spel vinner den andre bollen och får poäng. Likaså om bollen går ut och spelaren inte trycker, då vinner den andre bollen. Om den spelare som inte ska trycka ändå trycker förlorar denne poängen. Nytt serveläge ska komma automatiskt när en spelare har fått poäng.

### 4.1.2 Utförande

Knappnedtryckningar ska detekteras mha avbrott. Avbrottsrutinen ska byta flyttningsriktning om bollen är i ändläget hos respektive spelare, annars vidta lämplig åtgärd.

Avbrottsrutinerna får inte påverka huvudprogrammets registervärden och inte heller förutsätta specifika värden på registren vid avbrottet.

Följande information placeras lämpligen i dataminnet på adress 0x20001000 och högre:

- Lagring av hur lysdioderna är tända.
- Poäng spelare A
- Poäng spelare B
- Flyttningsriktning: 00=vänster, FF=höger
- Servstatus: 00=ej serve, FF=serve

## 4.2 Uppgift 8 (Extrauppgift)

Komplettera spelet med poängvisning genom utskrift på serieporten.

# Revisioner

| | |
|---|---|
| 4.00 | Byte till Cortex-M |
| 4.01 | Figur 1 inlagd, bytt initavbrott initint |
| 4.02 | Korrigerat avbrottsnivåer och portar som används |
| 4.03 | Förtydligat uppgifter |
| 4.04 | Korrigerat avbrottsnivåer |
| 4.1 | Bytt portar, extrauppgift, distansläge |
| 4.2 | Uppdateringar inför 2021 på distans |
| 4.3 | Korrigering stackdjup uppgift 4 |
| 4.4 | Utökat tabell avsnitt 3.5 |