

TSEA29 Konstruktion med mikrodatorer, Fö6

Anders Nilsson 2021-09-27

Institutionen för systemteknik

Agenda : Design av inbyggda system

- Hårdvarunära design – Erfarenhet/Utmaningar värda att tänka på
- Avbrottsrutiner och huvudloopar – hantering av gemensamma data
- Verkligt parallella processer – hantering av gemensamma data
- Kopplingsschema – hur och varför
- 10 sätt att lyckas med ett projekt
- Designspecar – bra / dåligt / hur / varför / när
- Handledning
- I²C-labb

Hårdvarunära design

Hårdvarunära design

Inbyggt system = ”man ser inget”



Hårdvarunära design

- Systemet byggs samtidigt som man programmerar det



Osäkerhet : Ligger problemen i hårdvaran eller mjukvaran?

Hårdvarunära design

- Man behöver ta *SMÅ* utvecklingssteg, justera kompasskursen ofta.

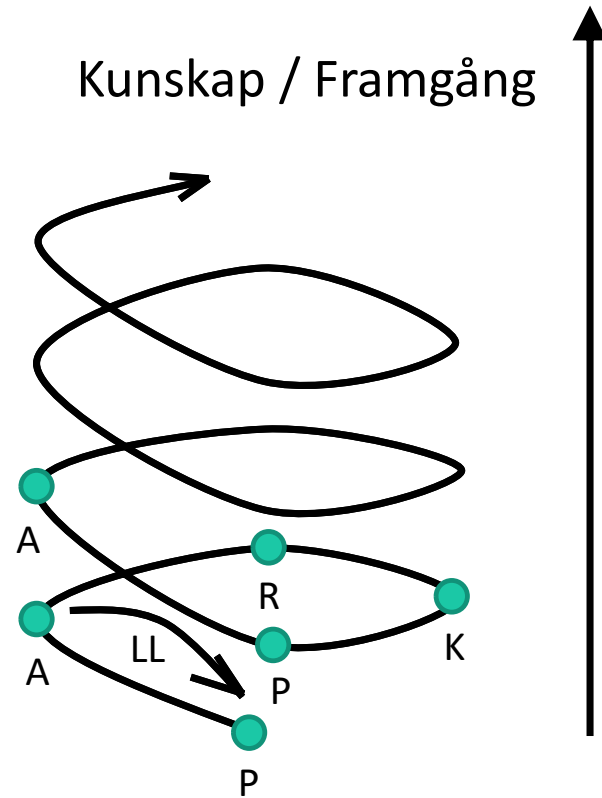


Varför?

- Helheten är svår att överblicka
- Små delar kan testas lättare
- Förutsättningar kanske förändras

Jämför med återkopplingstiden
i reglerloopen

Hårdvarunära design : P.A.R.K.

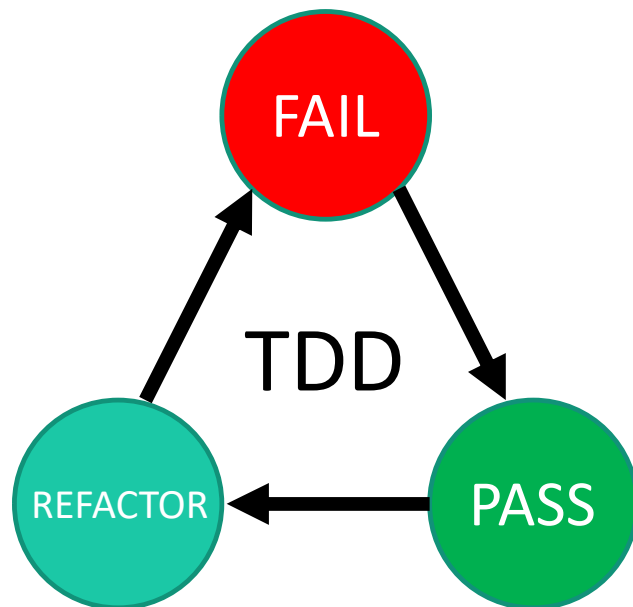


LL = Loosers Loop

- **Planera**
Samla information, strukturera, upprätta dokumentation
- **Agera**
Implementera, konstruera
- **Reflektera**
Varför fungerar det inte?
Varför fungerar det?
- **Korrigera**
Rätta till, uppdatera, optimera

Hårdvarunära design

TDD : Test Driven Development



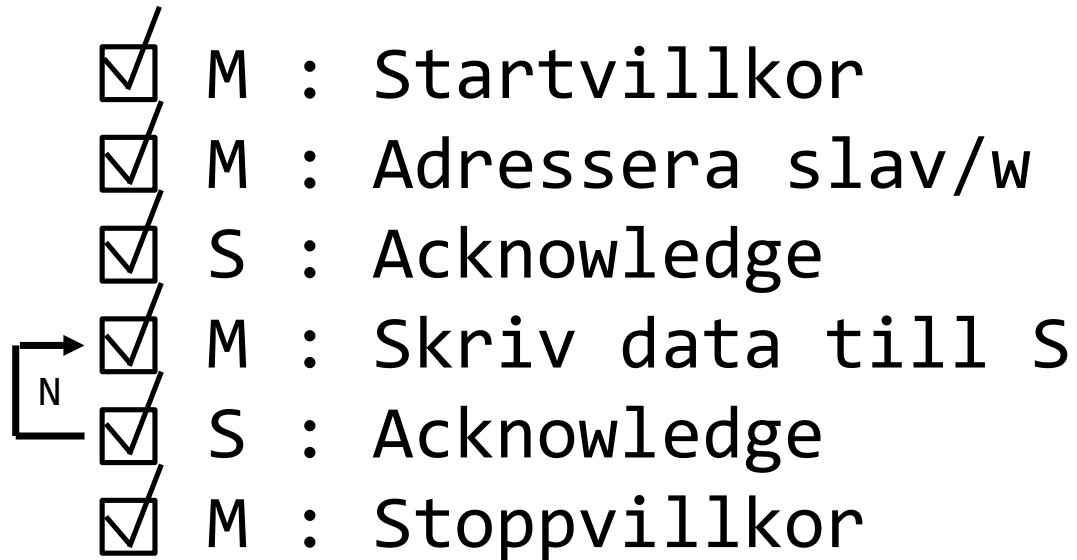
- FAIL : Skriv ingen kod förrän det tagits fram ett testfall som ännu inte uppfylls
- PASS : Skriv kod som uppfyller testfallet/en, men inte mer än det
- REFACTOR : Hyfsa till, strukturera, optimera koden och kontrollera att den fortfarande uppfyller testfallet/en

Överdrivet? Kanske...

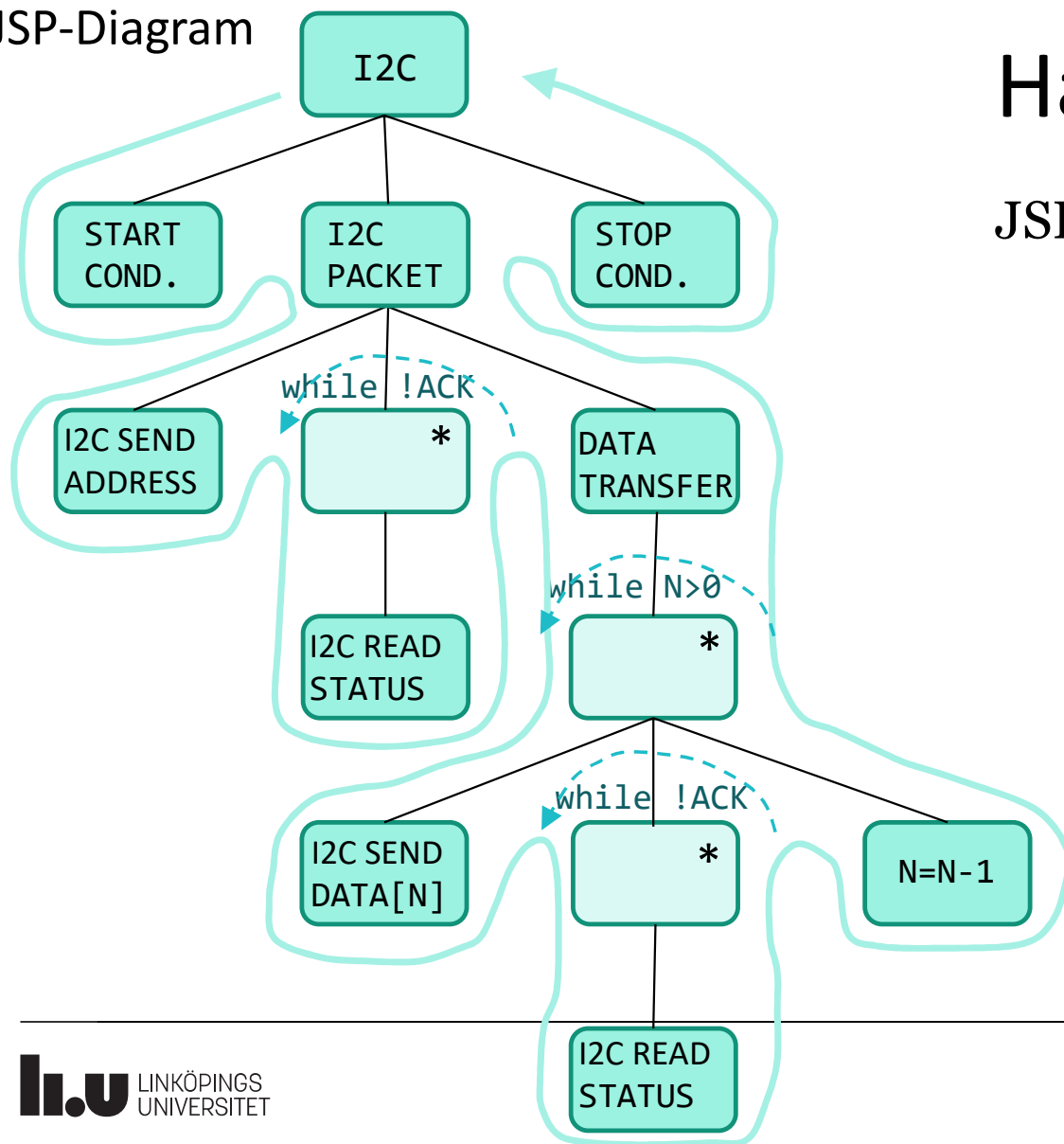
Bra: Leder till iterativ testning, ofta, och undviker big-bang-testning

Hårdvarunära design

I2C-kommunikation



JSP-Diagram

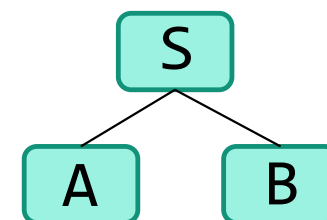


Hårdvarunära design

JSP : Jackson Structured Programming

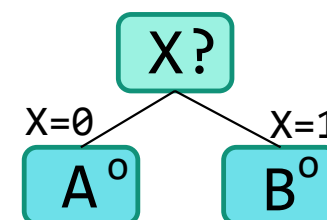
Sekvens:

Först A sen B



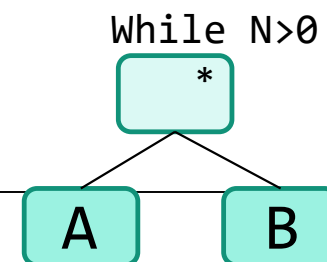
Selektion:

A eller B
beroende på X

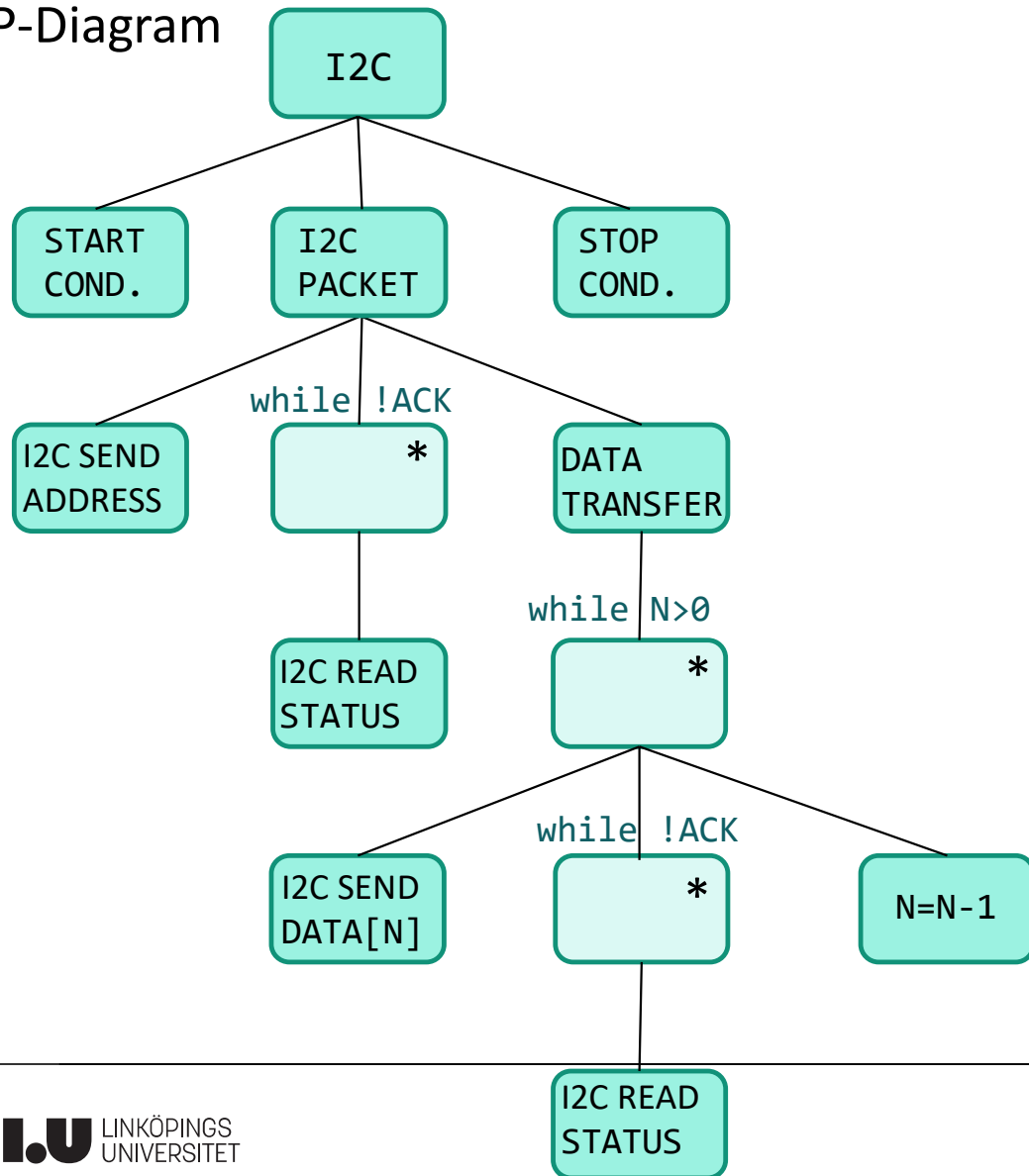


Iteration:

Först A sen B
Så länge $N > 0$



JSP-Diagram



Programkod

```

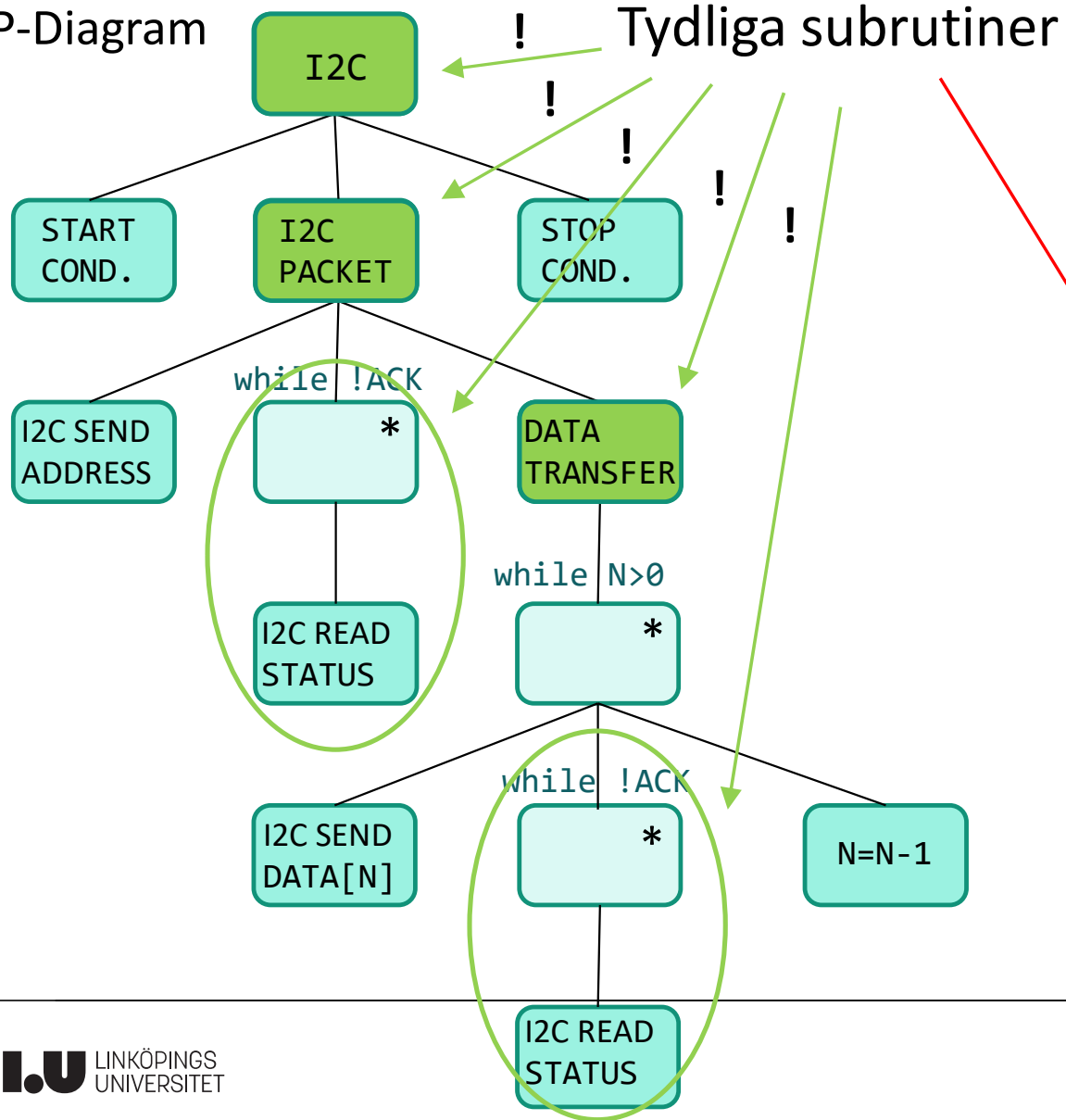
I2C() {
    I2C_Start_Cond();
    I2C_Packet();
    I2C_Stop_Cond();
}

I2C_Packet() {
    I2C_Send(ADDRESS);
    Wait_For_ACK();
    Data_Transfer();
}

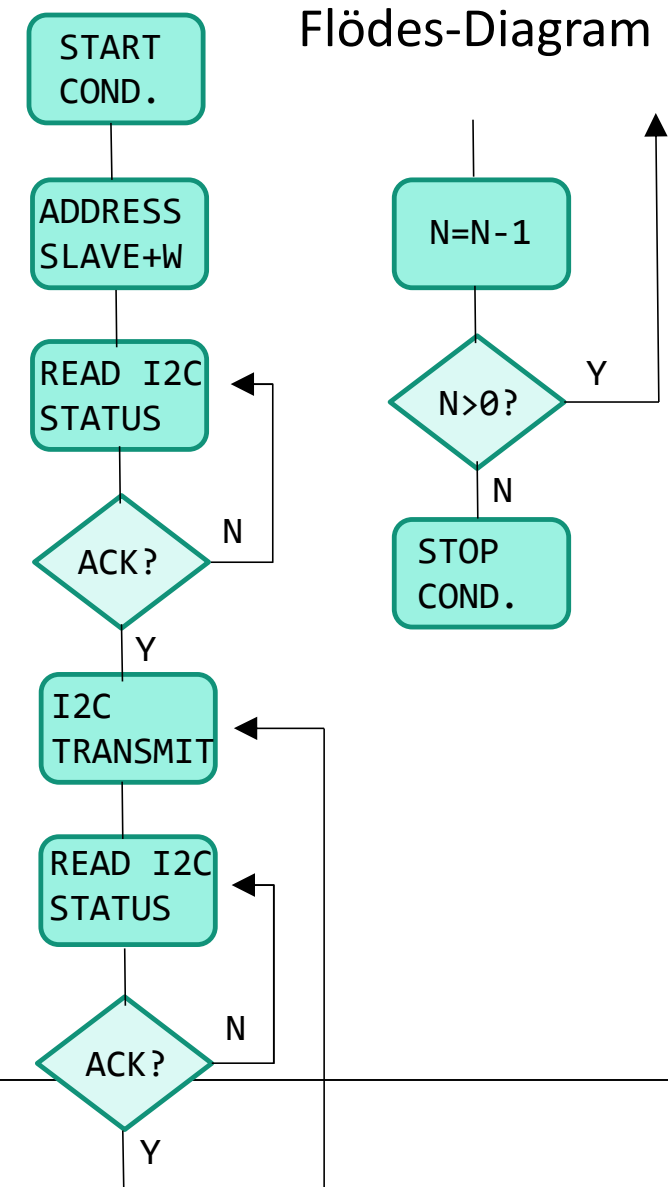
Data_Transfer() {
    while (N>0) {
        I2C_Send(DATA[N]);
        Wait_For_ACK();
        N--;
    }
}

Wait_For_ACK() {
    while (Get_I2C_Status() != ACK);
}
    
```

JSP-Diagram



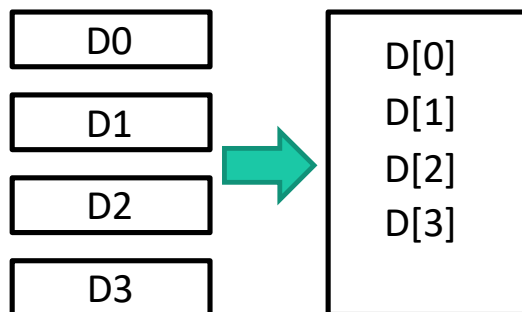
Flödes-Diagram



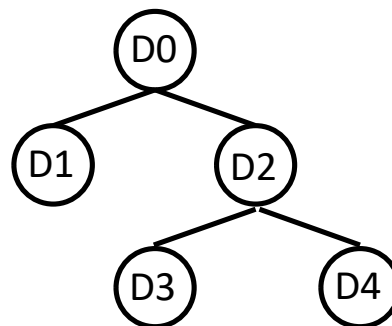
Datastrukturer

14

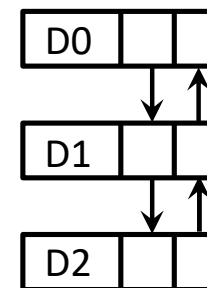
Buffer



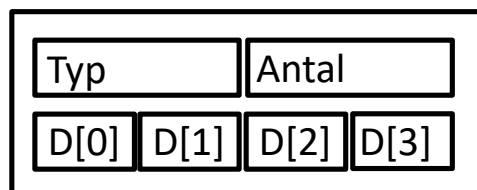
Träd



Länkad lista



Paketering med huvud



Rådata / SI-enheter / Flyttal / Heltal

35

Rådata

11cm

SI-enheter

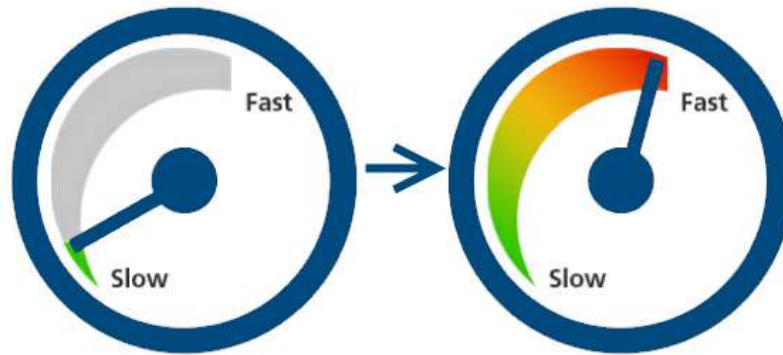
3.14

Flyttal eller Heltal

Hårdvarunära design

- Börja med låga hastigheter – öka vid behov

- Robustare
- Energisnålare



- Störningskänsligt
- Energiträvande

Det kan fungera bra att köra med full fart, **i början**, när det digitala bruset är lågt. Men med mer och mer funktionalitet blir systemet störningskänsligt. Plötsligt kan t ex ytterligare en sensor orsaka att I2C-kommunikationen slutar fungera, trots att det inte hör ihop. Problemet är egentligen relaterat till hastigheten.

Hårdvarunära design

- Spana framåt – undvik hinder
- Är vi på väg åt rätt håll?
- Leder det här till målet?
- Är det här en framkomlig väg?
- Finns det en lättare lösning?



Tänk efter före!

Hårdvarunära design

- Se även bakåt – ingen del av systemet är färdigt förrän hela produkten är klar
- Man kan få riva upp och göra om
- Då är man glad om man tagit små steg



Hårdvarunära design

- *Vår paranoid!!*

Koll på allt!



Överallt!

Hela tiden!

Avbrottsrutiner och huvudloopar

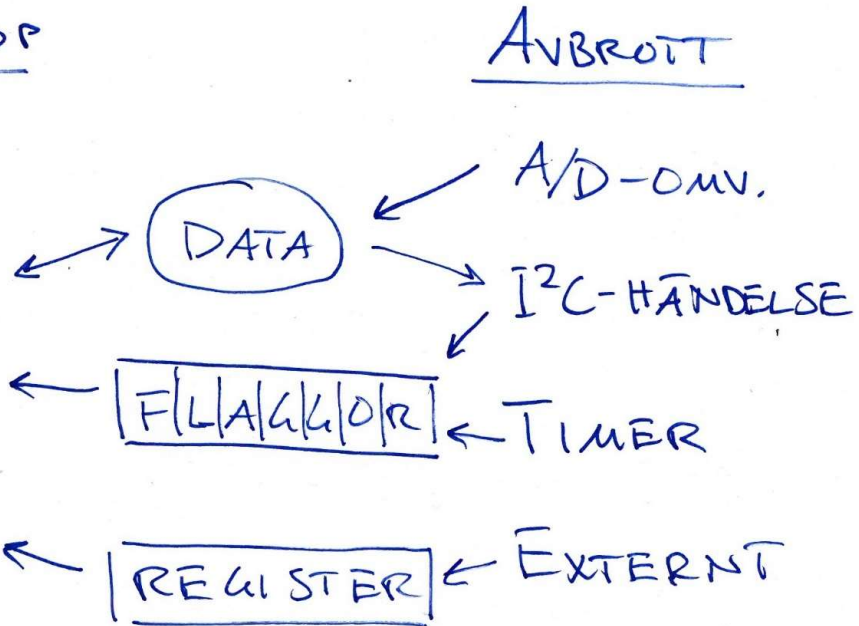
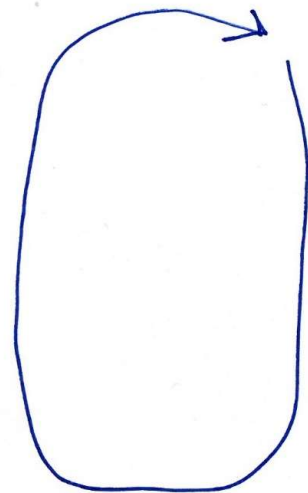
Ett sätt att uppnå realtidsfunktionalitet och virtuell parallellism

Avbrottsrutiner och huvudloopar

- Vilka avbrott ska vi hantera?
- Vad ska göras i avbrottsrutiner?
- Vad ska göras i kod som inte är avbrottsrutiner?
- Vad är lagom mycket för en avbrottsrutin?
- Hur klarar man krav på timing?
- Vilka data delas mellan avbrottsrutiner och övrig kod?
- Hur delas data? Ena läser och andra skriver, båda läser, båda skriver, etc.

Avbrottsrutiner och huvudloopar

HUVUDLOOP



AVBROTT

VARFÖR AVBROTT

TAR TID

INTE VILL POLLA

KRAV PÅ TIMING

YTRE HÄNDELSE/
SIGNAL

Avbrottsrutiner och huvudloopar

22

Avbrott i AVR, förutsättningar:

- Nya avbrott blockeras när avbrottsrutin körs (dvs pågående avbrott avbryts inte)
- Nya avbrott som inkommer när avbrottsrutin körs latchas/memoreras (en flagga per avbrottsvektor)
- Vissa variabler kan läsas och skrivas atomärt (dvs accessen av variabeln avbryts inte), andra är icke atomära

Avbrottsrutiner och huvudloopar

Avbrott i AVR, atomär läsning av en variabel vid upprepade tillfällen

Gemensam variabel mellan ISR (Interrupt Service Routine) och huvudloop

```
int Distance;
```

Variabeln sätt av ISR

Avbrottsrutiner och huvudloopar

Avbrott i AVR, atomär läsning av en variabel vid upprepade tillfällen

Kod i huvudloop

```
if (Distance < 10)
    decrease robot speed
else if (Distance > 20)
    increase robot speed
else
    do not change robot speed
```

← Vad händer om ISR förändrar
Distance mellan avläsningarna?

Avbrottsrutiner och huvudloopar

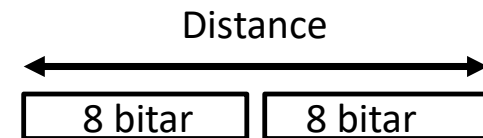
Avbrott i AVR, atomär läsning av en variabel vid ett tillfälle

```
local_distance = Distance;  
if (local_distance < 10)  
    decrease robot speed  
else if (local_distance > 20)  
    increase robot speed  
else  
    do not change robot speed
```


Avbrottsrutiner och huvudloopar

Avbrott i AVR, icke-atomär läsning av en variabel

```
DISABLE_INTERRUPTS;  
local_distance = Distance;  
ENABLE_INTERRUPTS;  
if (local_distance < 10)  
    decrease robot speed  
else if (local_distance > 20)  
    increase robot speed  
else  
    do not change robot speed
```



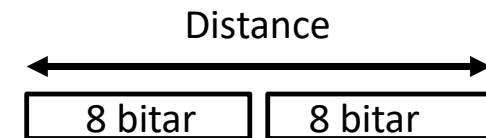
Avbrottsrutiner och huvudloopar

Avbrott i AVR, läsning av en *flyktig* variabel

```
uint_16t Distance = 0;
```

```
DISABLE_INTERRUPTS;  
local_distance = Distance;  
ENABLE_INTERRUPTS;
```

```
if (local_distance < 10)  
    decrease robot speed  
else if (local_distance > 20)  
    increase robot speed  
else  
    do not change robot speed
```



```
ISR {  
    ...  
    Distance = ...;  
    ...  
}
```

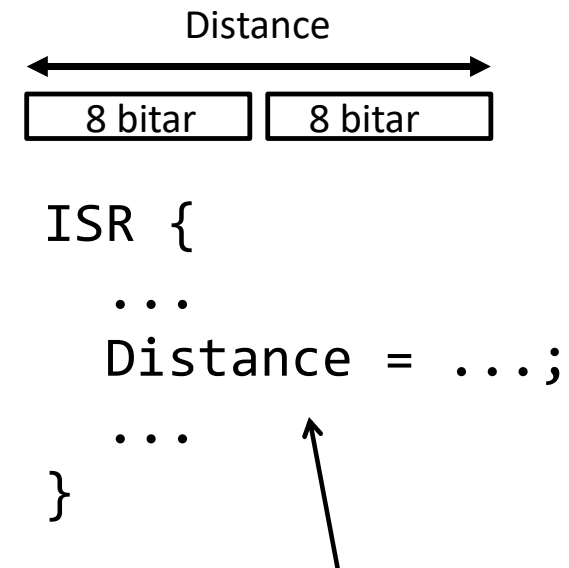
↖ Kompilatorn kanske
optimerar bort Distance.

Avbrottsrutiner och huvudloopar

Avbrott i AVR, läsning av en *flyktig* variabel : volatile

```
volatile uint_16t Distance = 0;
```

```
DISABLE_INTERRUPTS;
local_distance = Distance;
ENABLE_INTERRUPTS;
if (local_distance < 10)
    decrease robot speed
else if (local_distance > 20)
    increase robot speed
else
    do not change robot speed
```



Kompilatorn förstår att Distance är flyktig och kan ändras, t ex av ISR.

Avbrottsrutiner och huvudloopar

Avbrott, andra typer av scenarier – gemensamma data

- Buffertar – avbrottsrutin skriver/läser, huvudloop skriver/läser
- Gemensamma datablock – huvudloop skriver/läser hela block med avbrott stängda (för att säkerställa konsistens av data inom ett block)
- Dubbelbuffring – huvudloop arbetar med kopia, avbrottsrutin arbetar med en annan - avbrottsrutin gör "swap" genom att flytta pekare – huvudloop stänger av avbrott vid läsning av pekare om dessa inte kan läsas atomärt

Avbrottsrutiner och huvudloopar

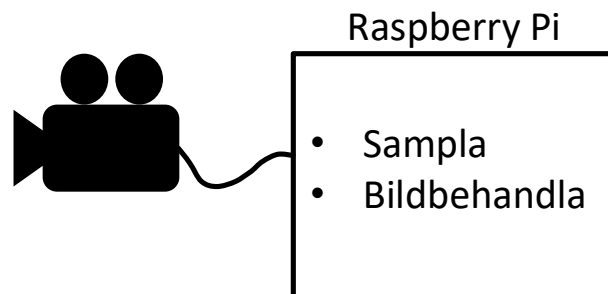
Avbrott - embedded, referensmaterial

- AVR : Tutorial, GCC m m – <https://www.mikrocontroller.net/articles/AVR>
Tysk site, men funkar bra med Google translate
- Artikel på vanheden – <http://vanheden.isy.liu.se>
- En artikelserie från embedded.com – <http://bit.ly/SVHrIh>

Parallella processer

Verkligt parallella processer i flera kärnor

Parallella processer



Sampla först -> buffer -> bildbehandla sen?

Hinner man bildbehandla mellan bilderna?

Knappast!

Skippa varannan bild?

Dålig framrate!

Kan man sampla och bildbehandla samtidigt?

Ja, med parallella trådar/processer!

På samma buffer, samtidigt?



Vad händer om sampling skriver samtidigt som bildbehandling läser?

Risk för korrupt/inkonsistent data!

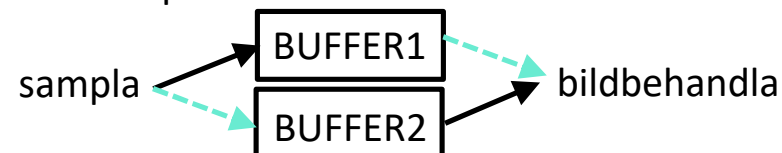
Hjälper det att stänga av avbrott?

Nej! Trådarna/processerna är inga avbrott

Kopiera mellan dubbla buffrar?

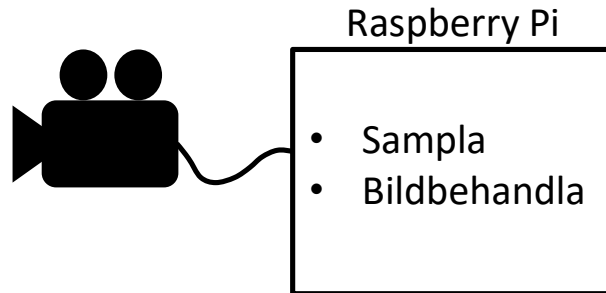


Växla buffrar med pekare?



Ja! MEN NÄR???

Parallella processer



Trådarna/processerna måste synkroniseras!

33

Trådar : -Har sitt ursprung från samma program
-Kan synkroniseras med semaforer
-Kan kommunicera via gemensamt/delat minne

Process: -Är ett program, skrivet i t ex Python, Java, C
-Kan synkroniseras med meddelanden
-Kan kommunicera via filer, pipes, sockets

I ett operativsystem förekommer många processer, som alla har en viss prioritet.

Vilken prioritet behöver sampling och bildbehandling ha?

I förhållande till varandra?

Sampling högre prio än bildbehandling

I förhållande till andra processer i operativsystemet?

Beror på situationen, dvs finns det fler viktiga tidskritiska saker som operativsystemet måste hantera

Parallella processer

34

```
//thread1 : sampling
```

```
while(1) {  
    for (i=0; i<count; i++)  
        buf[bp][i] = sample();  
    bp = 1 - bp;  
    sem_post(&Semaphore);  
}
```

```
//thread2 : img_processing
```

```
while(1) {  
    sem_wait(&Semaphore);  
    img_process(buf[1-bp]);  
}
```

Trådar inom samma program, kan kommunicera via gemensamt minne.

Synkroniseras med semafor.

```
//process1 : sampling
```

```
while(1) {  
    for (i=0; i<count; i++)  
        fputc(sample(), FileP[bp]);  
    M.bp = bp;  
    bp = 1 - bp;  
    message_send(&M, PORT);  
}
```

```
//process2 : img_processing
```

```
while(1) {  
    message_recieve(&M, PORT)  
    img_process(M.bp);  
}
```

```
int img_process(int bp) {  
    data = fgetc(FileP[bp]);  
    ...  
}
```

Processer i olika program, kan kommunicera via externa filer, lämpligen virtuella i RAM.

Synkroniseras med meddelandehantering.

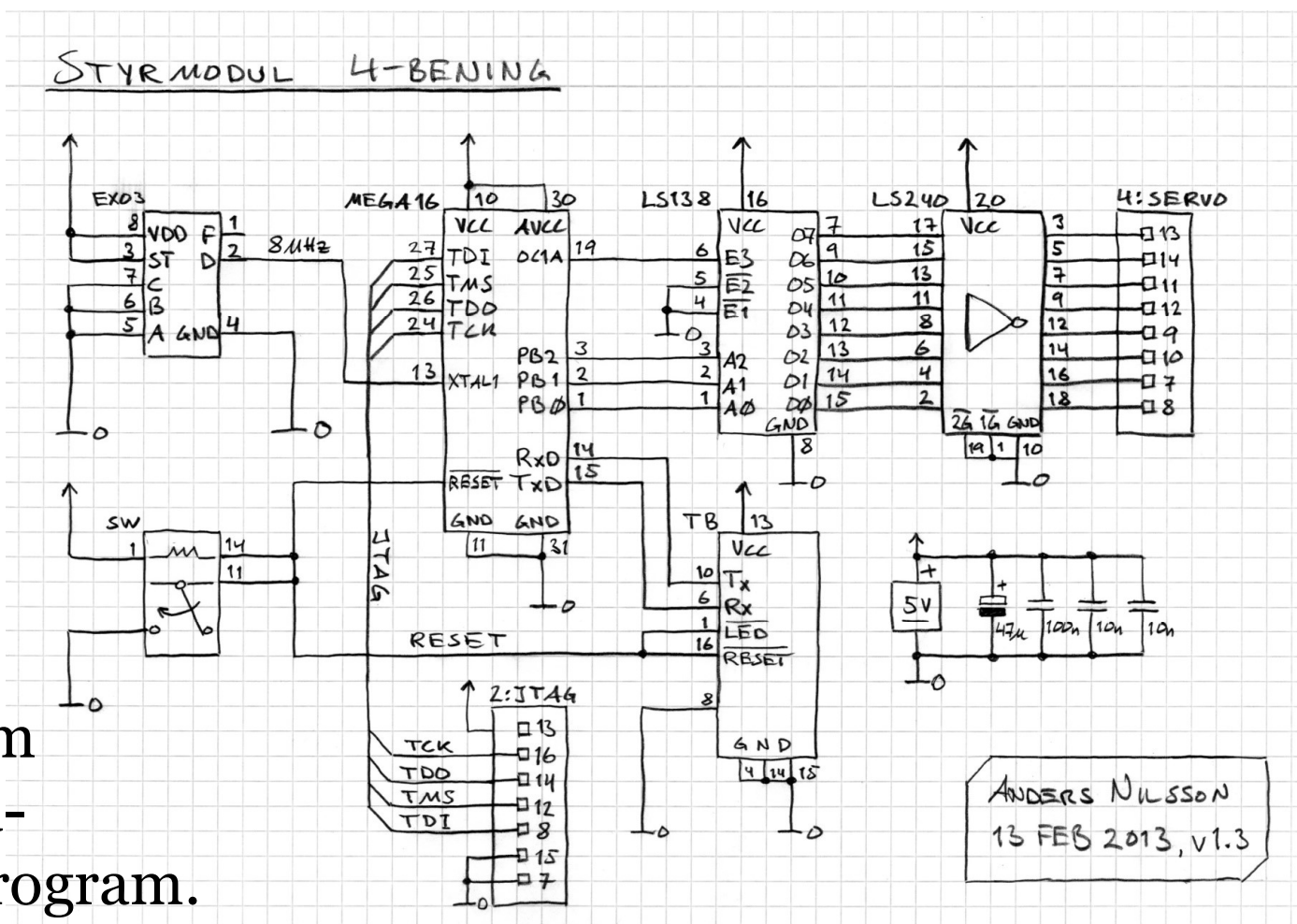
Kopplingsschema

Kopplingsschema

36

- Hur?
- Varför?

- Renrita!!
- Använd vanligt ritprogram el schema-ritningsprogram.

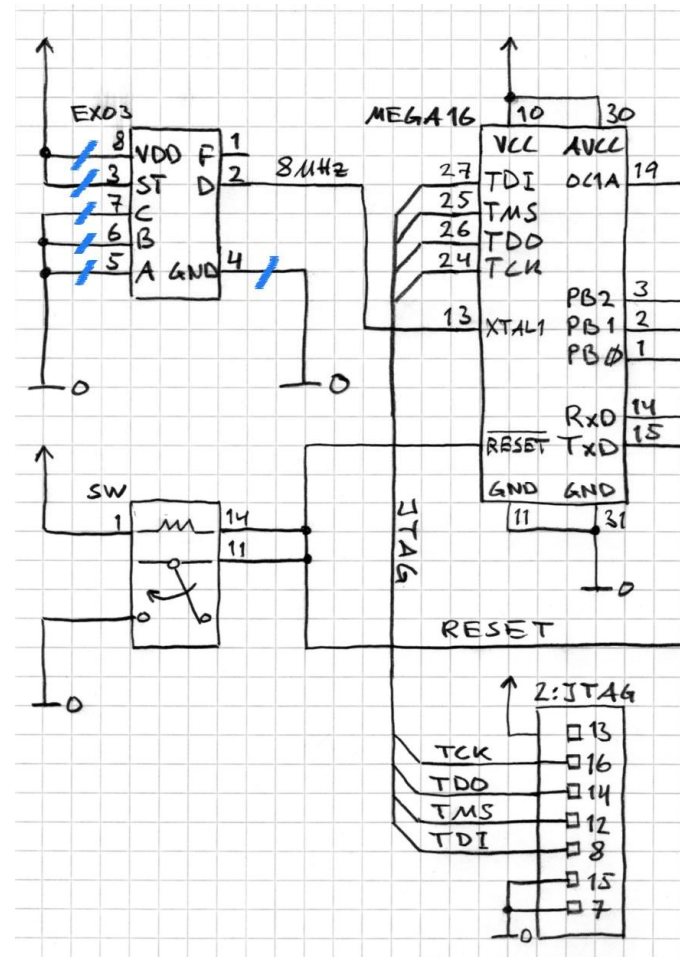


Kopplingsschema

37

Från schema till bygge

- Vira in en liten del åt gången
- Bocka av virade trådar i schemat



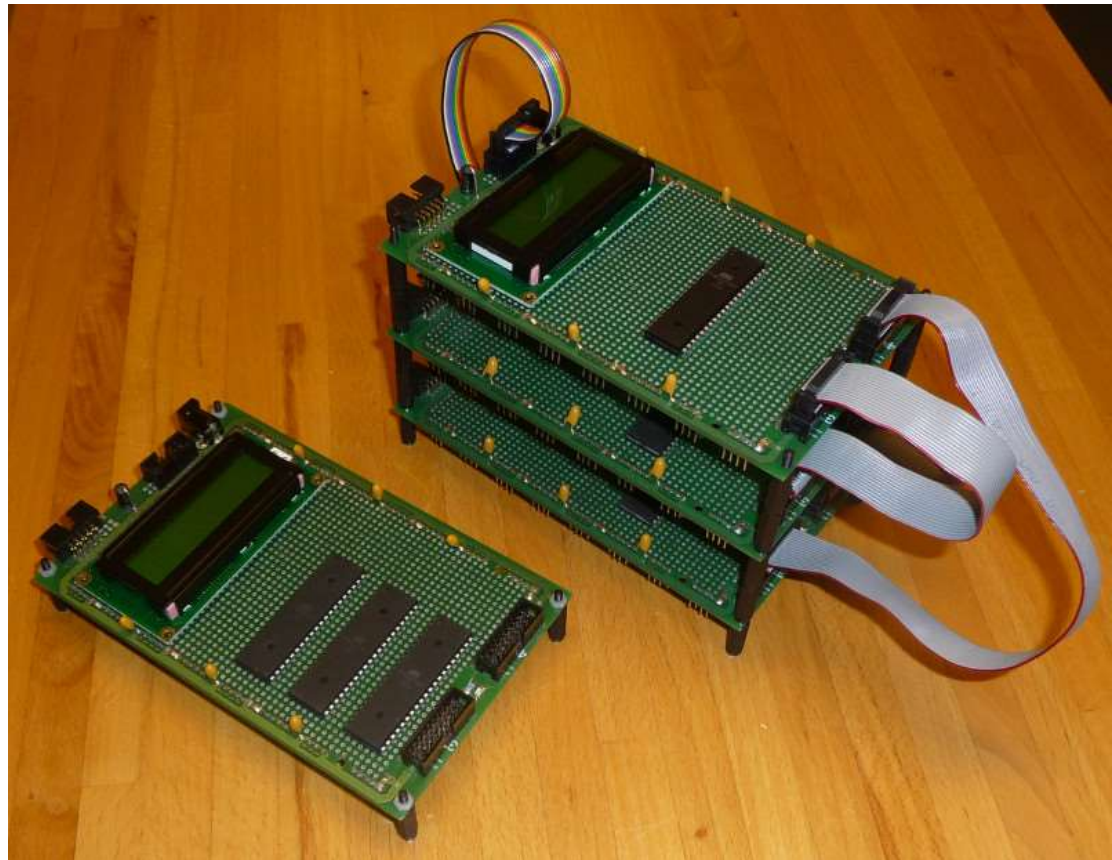
Kopplingsschema

38

Från schema till bygge

- Ett eller flera virkort

...

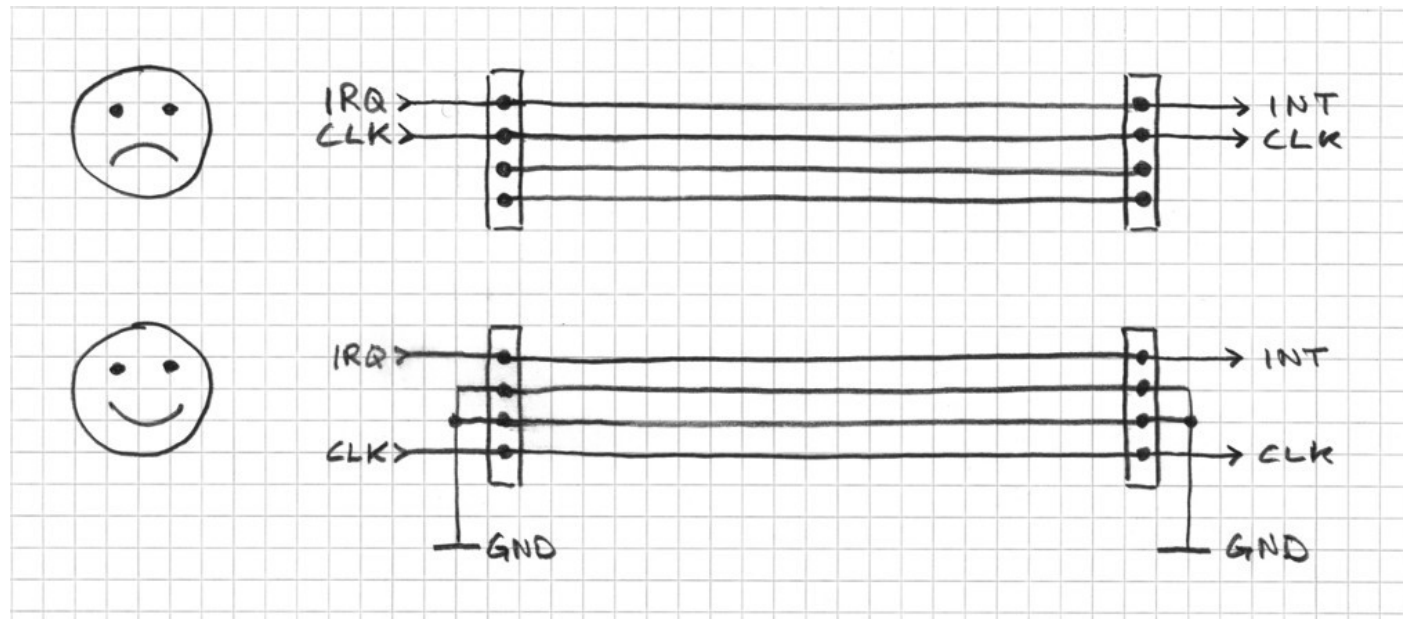


Kopplingsschema

39

Från schema till bygge

- Parallella ledare ...

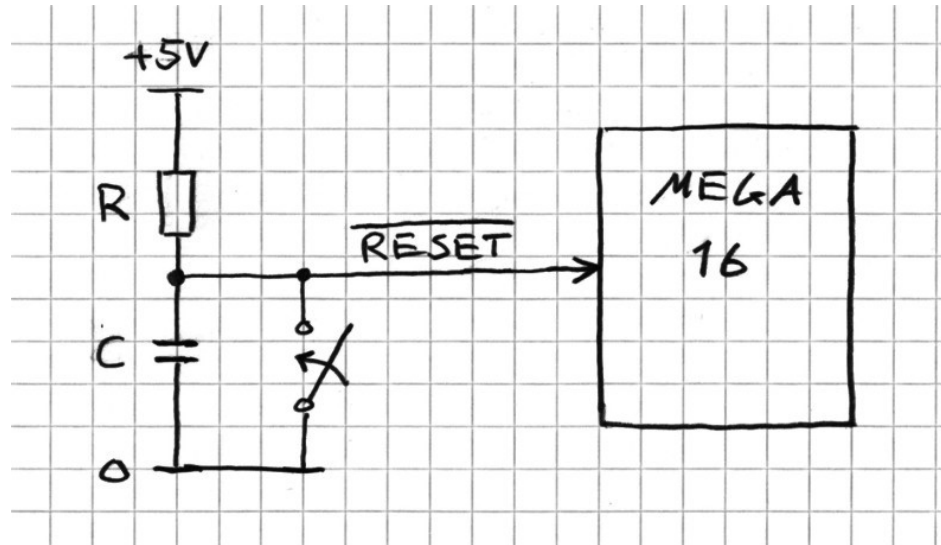


Kopplingsschema

40

Från schema till bygge

- Anslut reset!



Utan reset : svår felsökning!

Utan reset : felaktig programmering!

Kopplingsschema

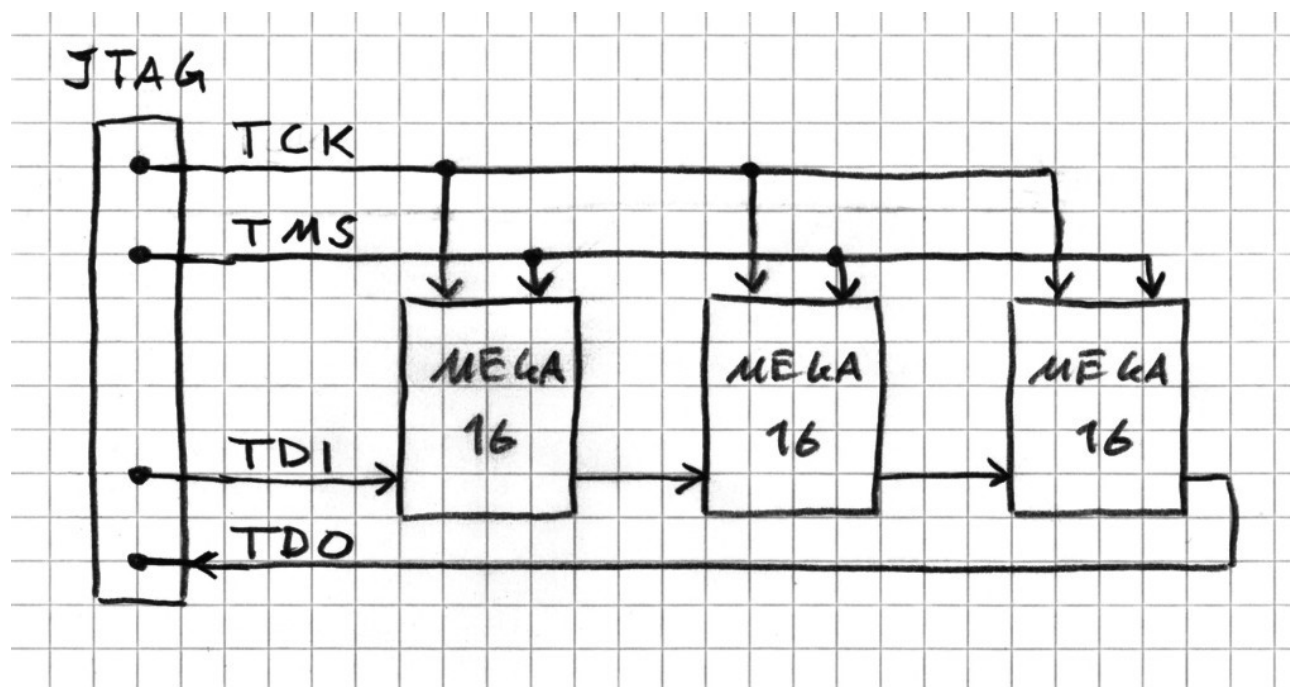
41

Från schema till bygge

- JTAG används för programmering och debugging

JTAG-kedja ...

... eller inte



Schemaritningsprogram

- EAGLE – <https://www.autodesk.com/products/eagle/>
 - Kommersiell, men gratis för mindre storlekar (80 cm^2) av konstruktioner
- KiCad – www.kicad-pcb.org
 - Open Source via GNU GPL v3

Ritprogram

- Inkscape – <https://inkscape.org>
 - Finns till Windows/Linux/MacOS

10 sätt att lyckas med ett projekt

Baserat på råd från tidigare års studenter

10 sätt att lyckas med ett projekt

45

1 : Börja i tid

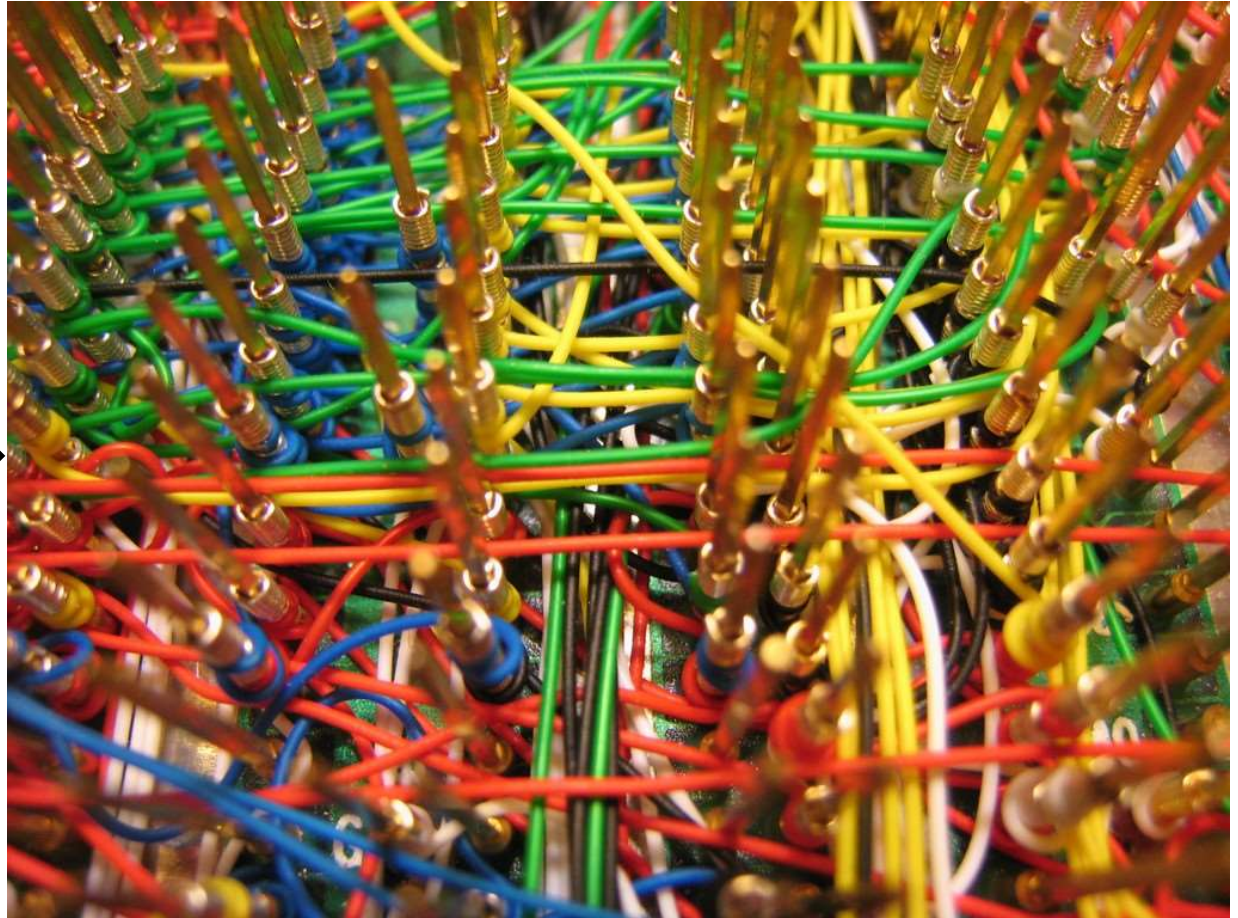


10 sätt att lyckas med ett projekt

46

2 : Gör ordentliga
virningar

www.bigmessofwires.com →



10 sätt att lyckas med ett projekt

47

3 : Skriv programkoden själv



```
seed = DateTime.Now.Millisecond;
number = new Random(seed);
int i = number.Next(0,9);

(maskedSet[i] == 0)

maskedSet[i]=1;
setCount++;
// Mask each set

seed = DateTime.Now.Millisecond;
number = new Random(seed);
int maskPos = number.Next(minPos,maxPos);
int j=0;
do
{
    seed = DateTime.Now.Millisecond;
    number = new Random(seed);
    int newPos = number.Next(1,9);
    int x = _setRowPosition[i]+pos;
    int y= _setColPosition[i]+pos;
    if(!_problemSet[x,y]==0)
    {
        _problemSet[x,y] = number;
        j++;
    }
}
```

48

```

graph TD
    A[I2C] --> B[START COND.]
    A --> C[I2C PACKET]
    A --> D[STOP COND.]
    C --> E[I2C SEND ADDRESS]
    C --> F[while !ACK  
*]
    F --> G[I2C READ STATUS]
    G --> C
    C --> H[DATA TRANSFER]
    H --> I[while N>0  
*]
    I --> J[I2C READ STATUS]
    J --> K[while !ACK  
*]
    K --> L[I2C SEND DATA[N]]
    L --> J
    J --> M[N=N-1]
    M --> I
    I --> N[Ctrl]
    N --> D
  
```

10 sätt att lyckas med ett projekt

49

5 : Använd versionshantering



<https://gitlab.liu.se>

Håll även kopplingsschemat uppdaterat!!!

10 sätt att lyckas med ett projekt

50

6 : Läs databladen

1. Översiktligt
2. Detaljerat
3. Verifiering



10 sätt att lyckas med ett projekt

51

7 : Verifiera komponentens funktion ur databladet



REKLAM

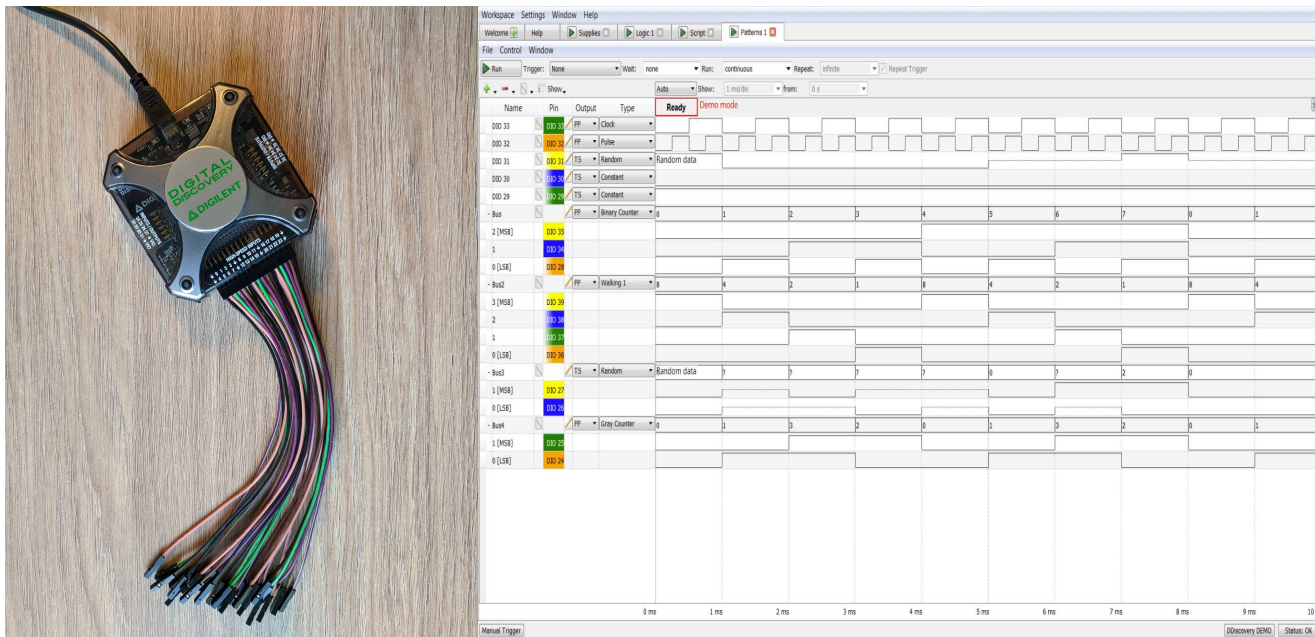


VERKLIGHET

10 sätt att lyckas med ett projekt

52

8 : MÄT!



Logikanalysatorlab
Flera tillfällen, gå på ett av dom
Anmälan i Lisam

"Zu messen ist zu wissen"



10 sätt att lyckas med ett projekt

53

9 : Samarbeta



Kommunicera!

Färdig med veckans arbete
i förväg? Tar man helg då?

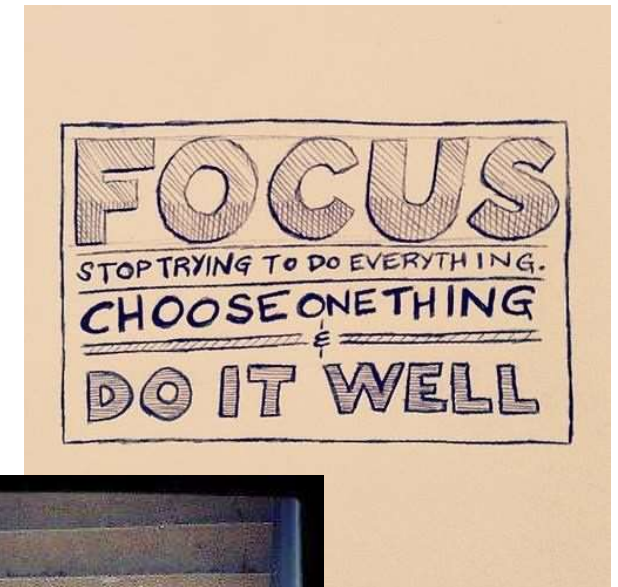
Sjuk? Berättar men det först
när man är tillbaka igen?

Alla har ett gemensamt ansvar!

10 sätt att lyckas med ett projekt

54

10 : Konstruera och lös problemen steg för steg



Designspecar

Designspecar

56

- Checklista för designspec =>
- Exempel på designspecar =>
 - + Fokus på hårdvaran
 - + Pseudokod som bilaga
 - + Upprepa inte datablad

Inlämningstid

- Senast 6/10 kl 16:00 ska en första version av designspecen vara inlämnad till handledaren
- Senast 13/10 ska designspecen kunna vara godkänd av handledaren

Designspecar

57

Designspecifikation : Förslag på arbetsordning

1 : Kopplingsschema

- Så detaljerat som möjligt
- Ta bort ovidkommande information
- Vilken / vilka bussar är lämpliga / krävs?
- Namnge signaler och bussar
- Planera in anslutning för JTAG samt reset
- Behövs nivåkonvertering? 5V <-> 3V

2 : Datastrukturer

- Hur ska data lagras?
- Vad ska skickas?
- På vilken form? Rådata eller SI-enheter?
- Behövs nåt protokoll, paketering av data?

3 : Programflöde / struktur

- Vad ska vara avbrott?
- Vad passar i huvudloopen

4 : Beskrivande information

- Textuell
- Bilder / figurer

Handledning

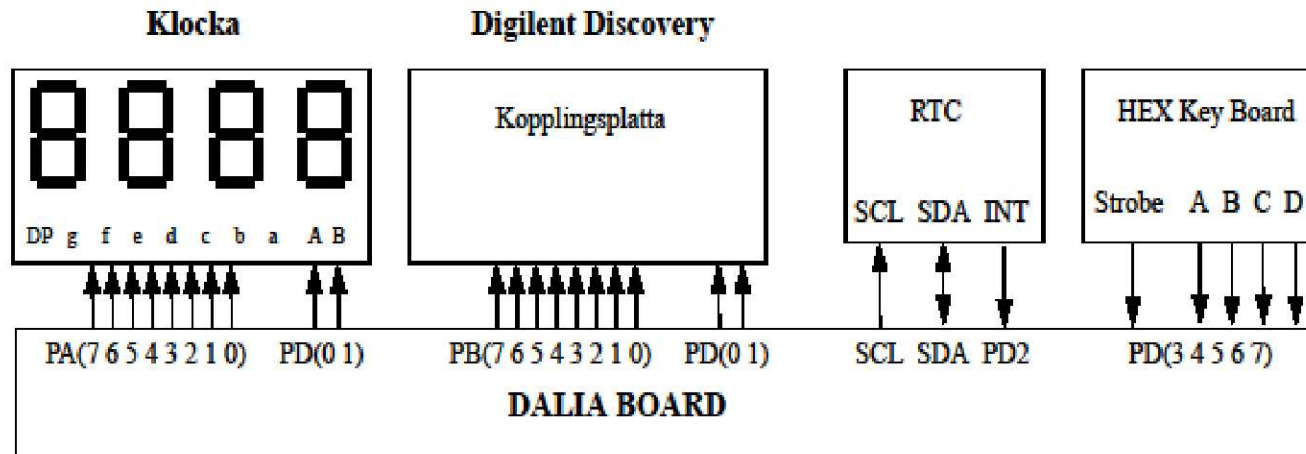
- Två timmar handledning per vecka
- Använd tilldelad handledare i första hand
Vid enstaka tillfällen då din handledare inte är tillgänglig kan andra handledare vara behjälpliga med mindre insatser, t ex lämna ut en komponent eller svara på en akut fråga.
- Handledarens huvudsakliga uppgift är att förmedla arbetssätt, t ex felsökningsmetodik, snarare än att utföra felsökning
- Handledningsform:
 - Uppsöka vid behov?
 - Boka möten?
 - E-post? Teams?

Diskutera upplägget med er handledare

I2C-labb

I²C-labb

60



Uppgift A : Multiplexad display

Uppgift B : I²C-buss (pollad)

Anders Nilsson

www.liu.se