Fra sekvensielt til parallelt

«Sanntidprogrammering etter 34 år»

Øyvind Teig
senior utviklingsingeniør
Autronica Fire and Security, «a UTC company»

Gjesteforelesning på Høgskolen i Sør-Trøndelag (HiST) 10.nov. 2010

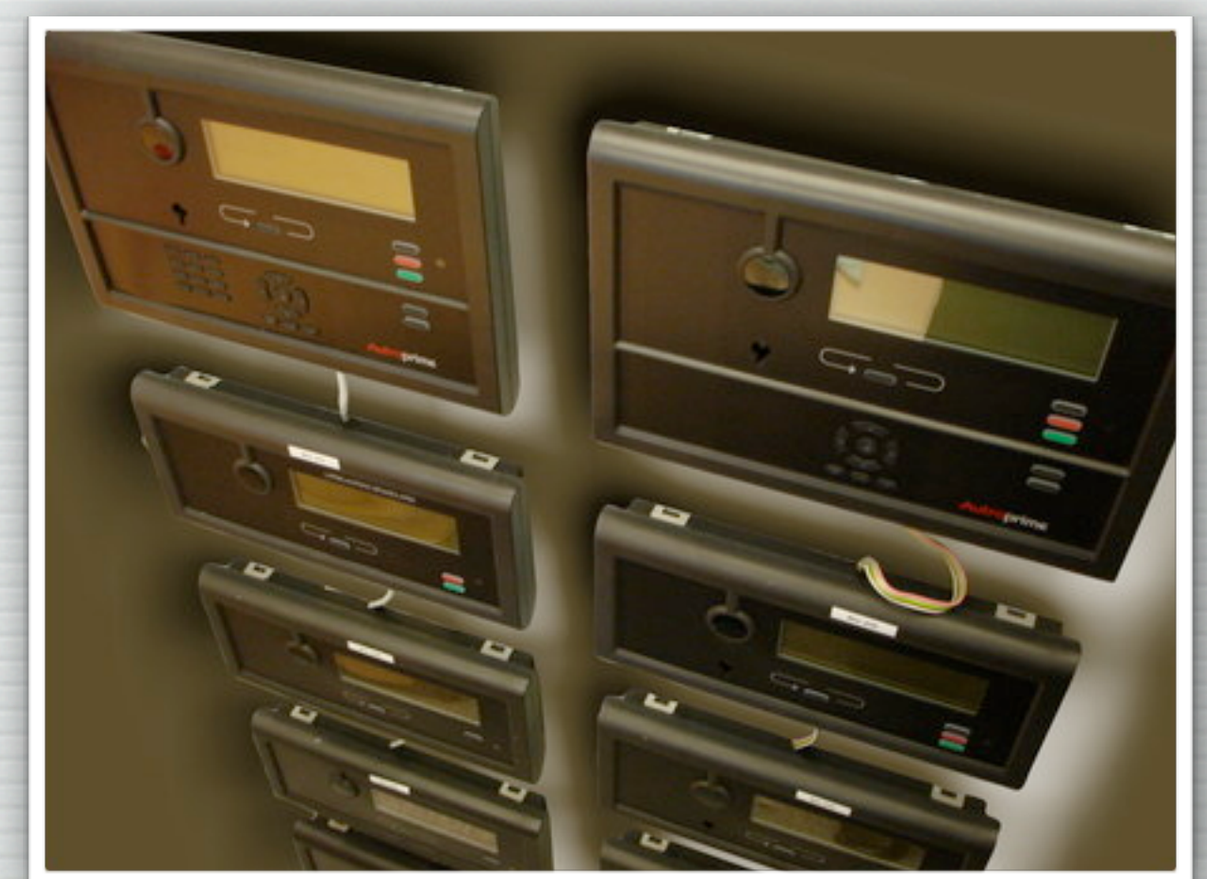
Fra sekvensielt til parallelt

«Sanntidprogrammering etter 34 år»

fremdeles gøy!

Øyvind Teig
senior utviklingsingeniør
Autronica Fire and Security, «a UTC company»

Gjesteforelesning på Høgskolen i Sør-Trøndelag (HiST) 10.nov. 2010



AFS: brannvarsling

Fra brannmeldere

Fra brannmeldere

til kontrollere

Fra brannmeldere

til kontrollere

til sentraler

Fra brannmeldere

til kontrollere

til sentraler
(forelesning med Ommund Øgård på nyåret)

Sekvensielt

C

Sekvensielt

C

C++

Sekvensielt

- C
- C++
- Java

Sekvensielt

C

C++

Java

main()

Sekvensielt

C

C++

Java

main()

new ThisObject(); new ThatObject();

Fremdeles like

C

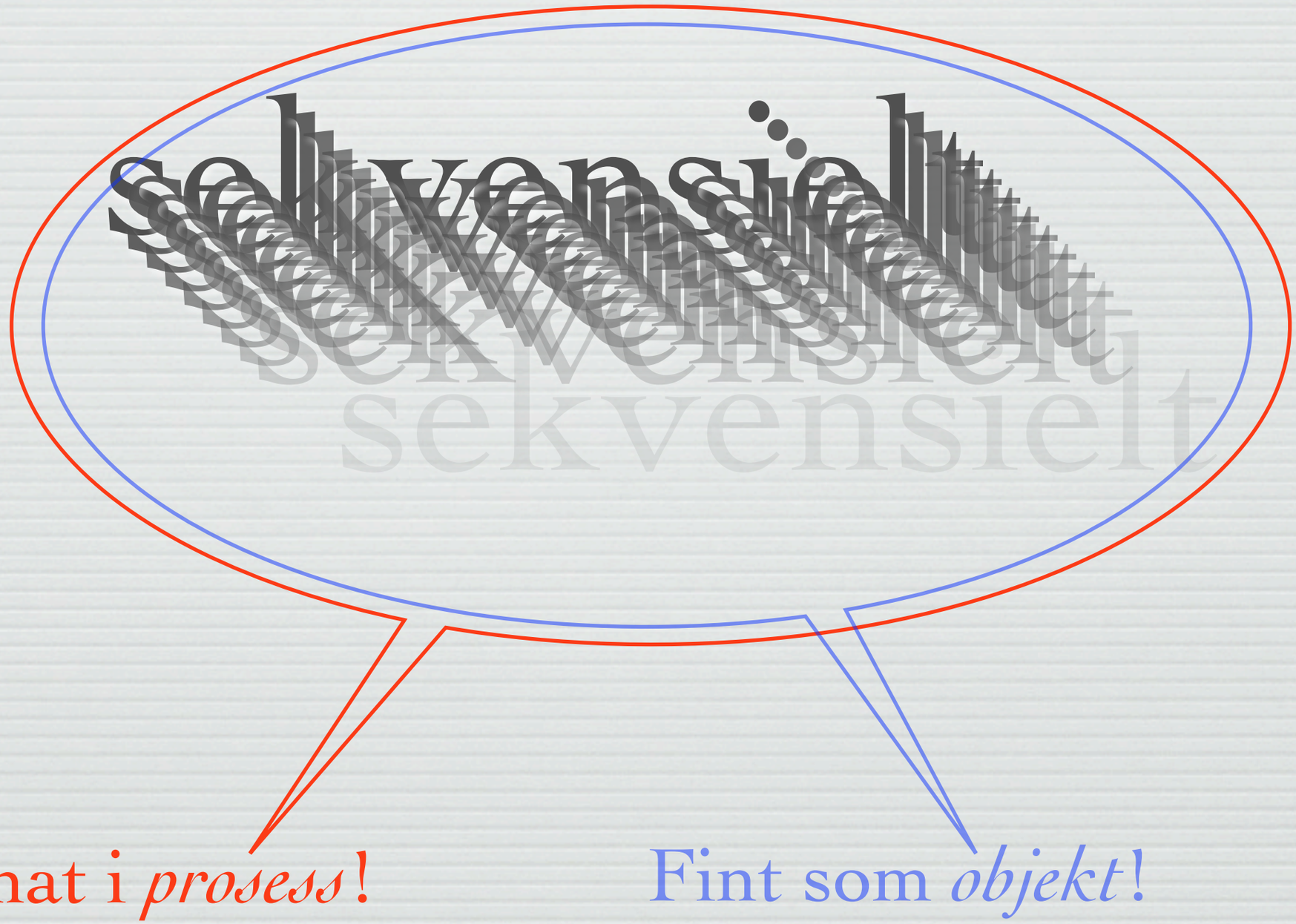
C++

Java

```
                                main()
new ThisObject(); new  ThatObject();
```

sekvensielt

Fint som *objekt*!



Fin innmat i *prosess*!

Fint som *objekt*!

Objekt

-orientert programmering

Prosess -orientert programmering

Objekt

Prosess

-orientert

programmering

ja, takk!

Objekt

Prosess

-orientert

når, hvor?

Hvilke,

programmering

ja, takk!

Prosess

Prosess

Startes
lik objekter eller *function* med *static* data

Prosess

Startes
lik objekter eller *function* med *static* data

Kjører
av seg sjøl

Prosess

Startes
lik objekter eller *function* med *static* data

Kjører
av seg sjøl

Er tette
ingen lekkasje

Prosess

Startes
lik objekter eller *function* med *static* data

Kjører
av seg sjøl

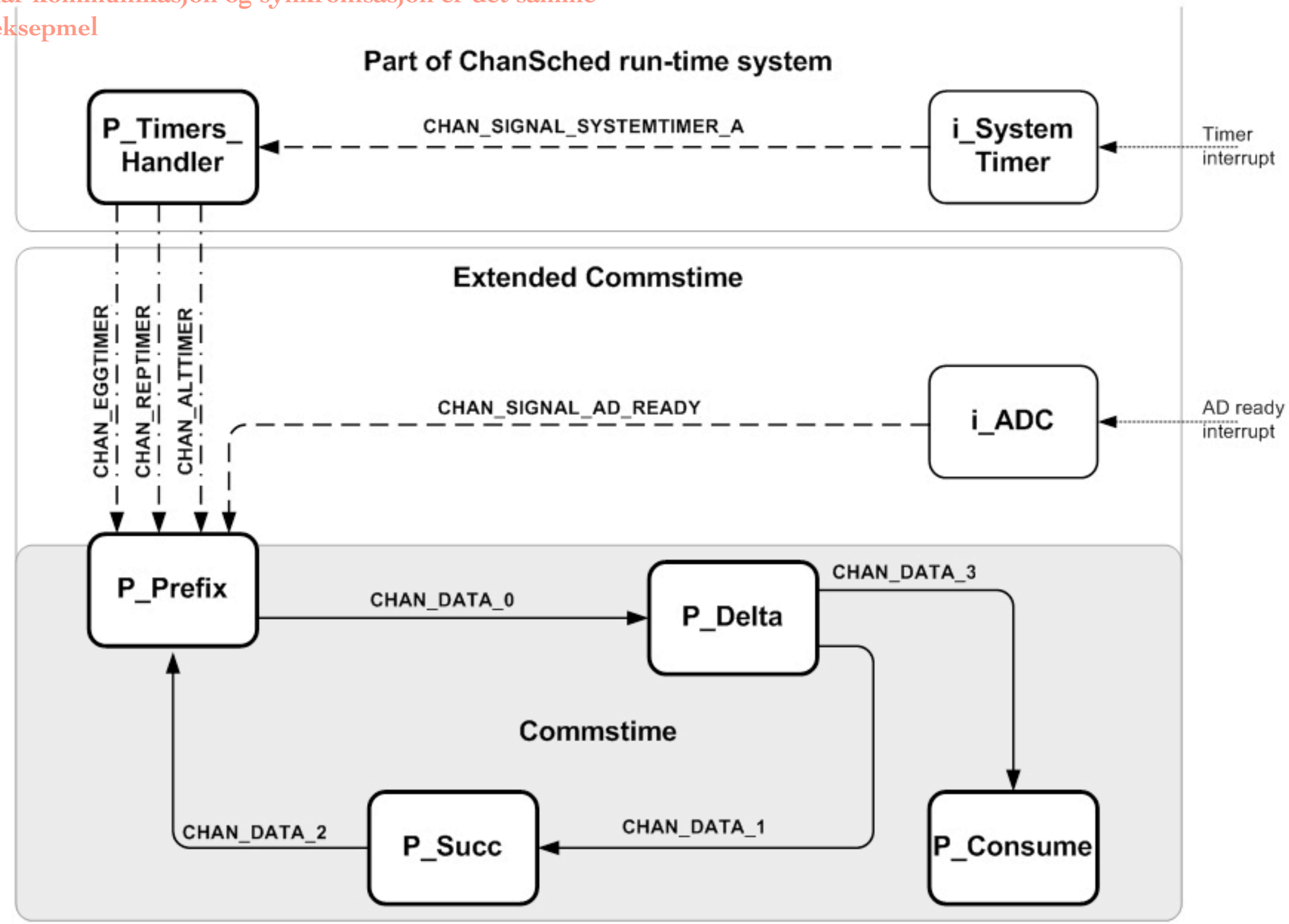
Er tette
ingen lekkasje

Kan ha egne tidskrav

Forklar hovedfunksjon

Forklar når kommunikasjon og synkronisasjon er det samme

Vis kodeeksepmel



Meldinger, *ikke* vha

Meldinger, *ikke* vha

funksjonskall

Meldinger, *ikke* vha

funksjonskall

pekere til fellesdata

Meldinger, *ikke* vha

funksjonskall

pekere til fellesdata

semaforebeskyttede data

Meldinger, *ikke* vha

funksjonskall

pekere til fellesdata

semaforebeskyttede data

pipes

Meldinger, *ikke* vha

funksjonskall

pekere til fellesdata

semaforebeskyttede data

pipes

meldinger

Meldinger, *ikke* vha

funksjonskall

pekere til fellesdata

semaforebeskyttede data

pipes

asynkrone meldinger

Meldinger, *ikke* vha

funksjonskall

pekere til fellesdata

semaforebeskyttede data

pipes

asynkrone meldinger

«skjeve» meldingsdiagram

Meldinger, *ikke* vha

funksjonskall

pekere til ellesdata

semaforebeskyttede data

pipes

asynkrone

meldinger

«skjeve» meldingsdiagram

Som (alle) *kan* være ok til sitt bruk

Meldinger, vha

Meldinger, vha

synkrone meldinger

Meldinger, vha

synkrone meldinger

asynkrone «signaler»

Meldinger, vha

synkrone meldinger

asynkrone «signaler»

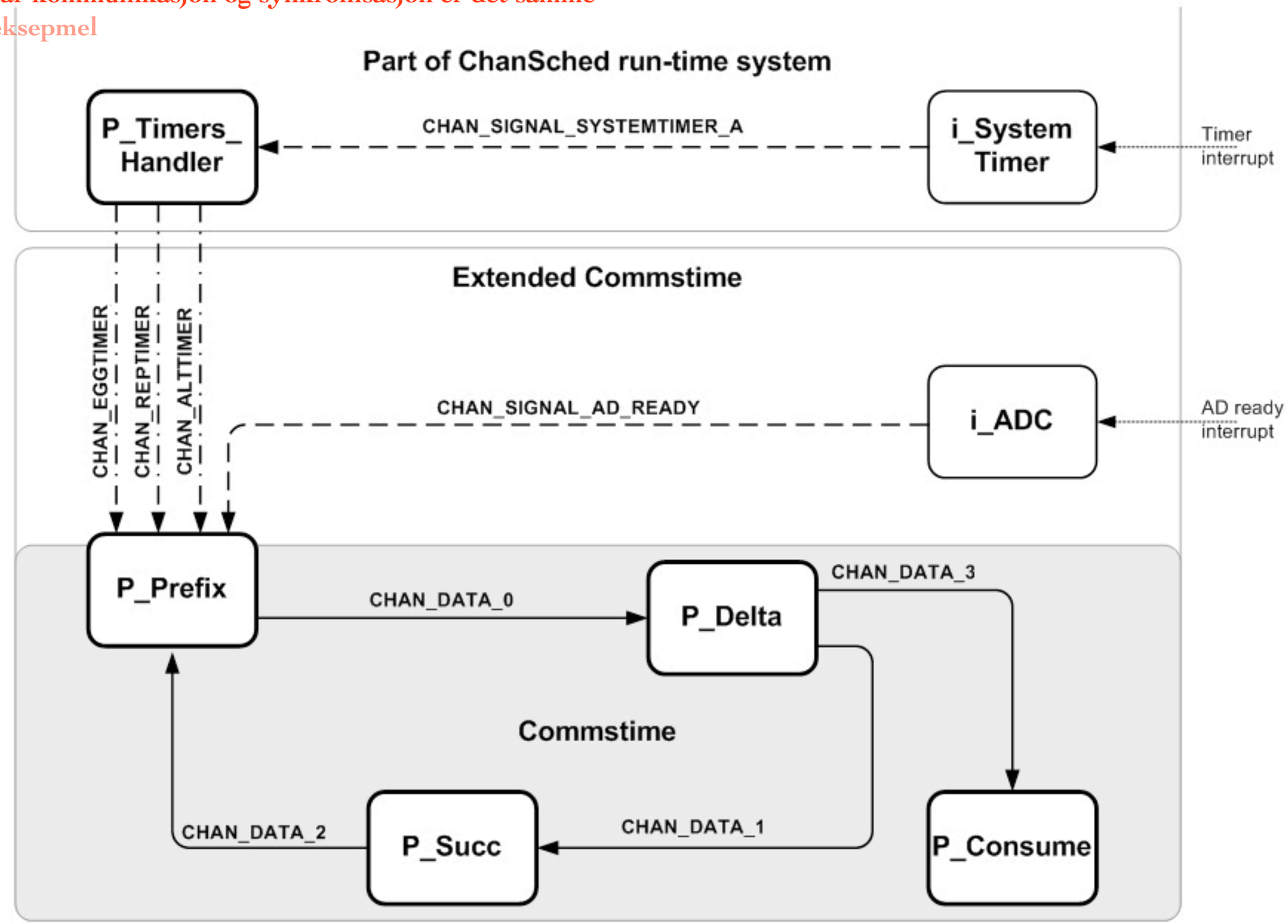
«rette» meldingsdiagram

kommunikasjon=

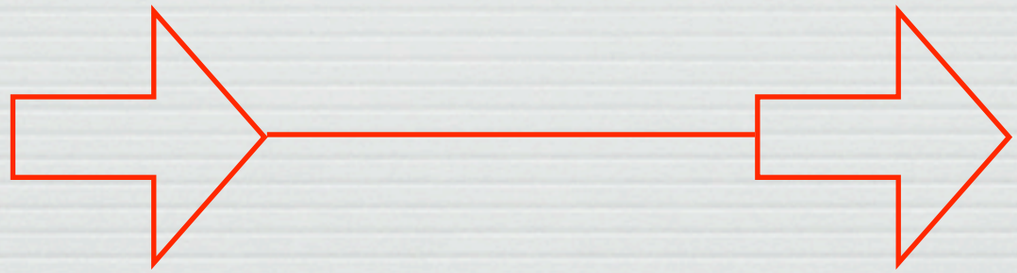
Forklar hovedfunksjon

Forklar når kommunikasjon og synkronisasjon er det samme

Vis kodeeksepmel



kommunikasjon=



synkronisering

halvferdig

kommunikasjon=

synkronisering

halvferdig

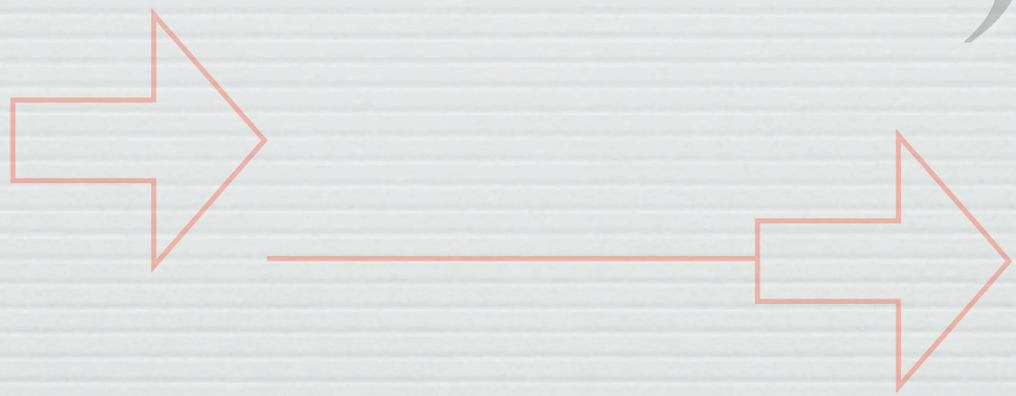
kommunikasjon=

synkronisering

blokkering

halvferdig

kommunikasjon=



synkronisering

blokkering

betyr at prosessoren kan gjøre andre ting

CSP

Communicating Sequential Processes (Hoare)

Egen formalisme og brukes i språk og bibliotek

Ada

CSP

Communicating Sequential Processes (Hoare)
Egen formalisme og brukes i språk og bibliotek

Ada

CSP

KRoC

Communicating Sequential Processes (Hoare)

Egen formalisme og brukes i språk og bibliotek

Ada

CSP

KRoC

JCSP

Ada

CSP

KRoC

Transterpreter

JCSP

Ada

CSP

KRoC

JCSP

Transter-
preter

C+kjøresystem

parallellitet

parallellitet

alt blir enklere

parallellitet

alt blir enklere

WYSIWYG-semantikk

parallellitet

alt blir enklere

WYSIWYG-semantikk
= se på MSC, ikke inni motpart

parallellitet

alt blir enklere

WYSIWYG-semantikk
= se på MSC, ikke inni motpart

Byggeklosser / bibliotek

parallellitet

alt blir enklere

WYSIWYG-semantikk
= se på MSC, ikke inni motpart

Byggeklosser / bibliotek

Klient / tjener

parallellitet

nesten alt blir enklere

WYSIWYG-semantikk
= se på MSC, ikke inni motpart

Byggeklosser / bibliotek

Klient / tjener

parallellitet

nesten alt blir enklere

WYSIWYG-semantikk
= se på MSC, ikke inni motpart

Byggeklosser / bibliotek

Klient / tjener

Vranglås!

parallellitet

nesten alt blir enklere

WYSIWYG-semantikk
= se på MSC, ikke inni motpart

Byggeklosser / bibliotek

Klient / tjener

Vranglås!

Bruk sikre design mønstre

parallellitet

nesten alt blir enklere

WYSIWYG-semantikk
= se på MSC, ikke inni motpart

Byggeklosser / bibliotek

Klient / tjener

Vranglås!

Bruk sikre design mønstre

Formell modellering = bevis

rett modell for

parallellitet

(nesten) alt blir enklere

WYSIWYG-semantikk
= se på MSC, ikke inni motpart

Byggeklosser / bibliotek

Klient / tjener

Vranglås!

Bruk sikre design mønstre

Formell modellering = bevis

```
mtype = {M_UP, M_DW};
chan Chan_data_down = [0] of {mtype};
chan Chan_data_up   = [0] of {mtype};
proctype P1 (chan Chan_data_in, Chan_data_out)
{
    do
    ::  Chan_data_in  ? M_UP -> skip;
    ::  Chan_data_out ! M_DW -> skip;
    od;
}
proctype P2 (chan Chan_data_in, Chan_data_out)
{
    do
    ::  Chan_data_in  ? M_DW -> skip;
    ::  Chan_data_out ! M_UP -> skip;
    od;
};
init
{
    atomic
    {
        run P1 (Chan_data_down, Chan_data_up);
        run P2 (Chan_data_up,   Chan_data_down);
    }
}
```

En formell modell i Promela kan formelt verifiseres i Spin

http://en.wikipedia.org/wiki/Promela#Executability

Promela / Spin

```
mtype = {M_UP, M_DW};
chan Chan_data_down = [0] of {mtype};
chan Chan_data_up   = [0] of {mtype};
proctype P1 (chan Chan_data_in, Chan_data_out)
{
    do
    ::  Chan_data_in  ? M_UP -> skip;
    ::  Chan_data_out ! M_DW -> skip;
    od;
};
proctype P2 (chan Chan_data_in, Chan_data_out)
{
    do
    ::  Chan_data_in  ? M_DW -> skip;
    ::  Chan_data_out ! M_UP -> skip;
    od;
};
init
{
    atomic
    {
        run P1 (Chan_data_down, Chan_data_up);
        run P2 (Chan_data_up,   Chan_data_down);
    }
}
```

En formell modell i Promela kan formelt verifiseres i Spin

http://en.wikipedia.org/wiki/Promela#Executability

Promela / Spin

Ada

CSP

Transter-
preter

KRoC

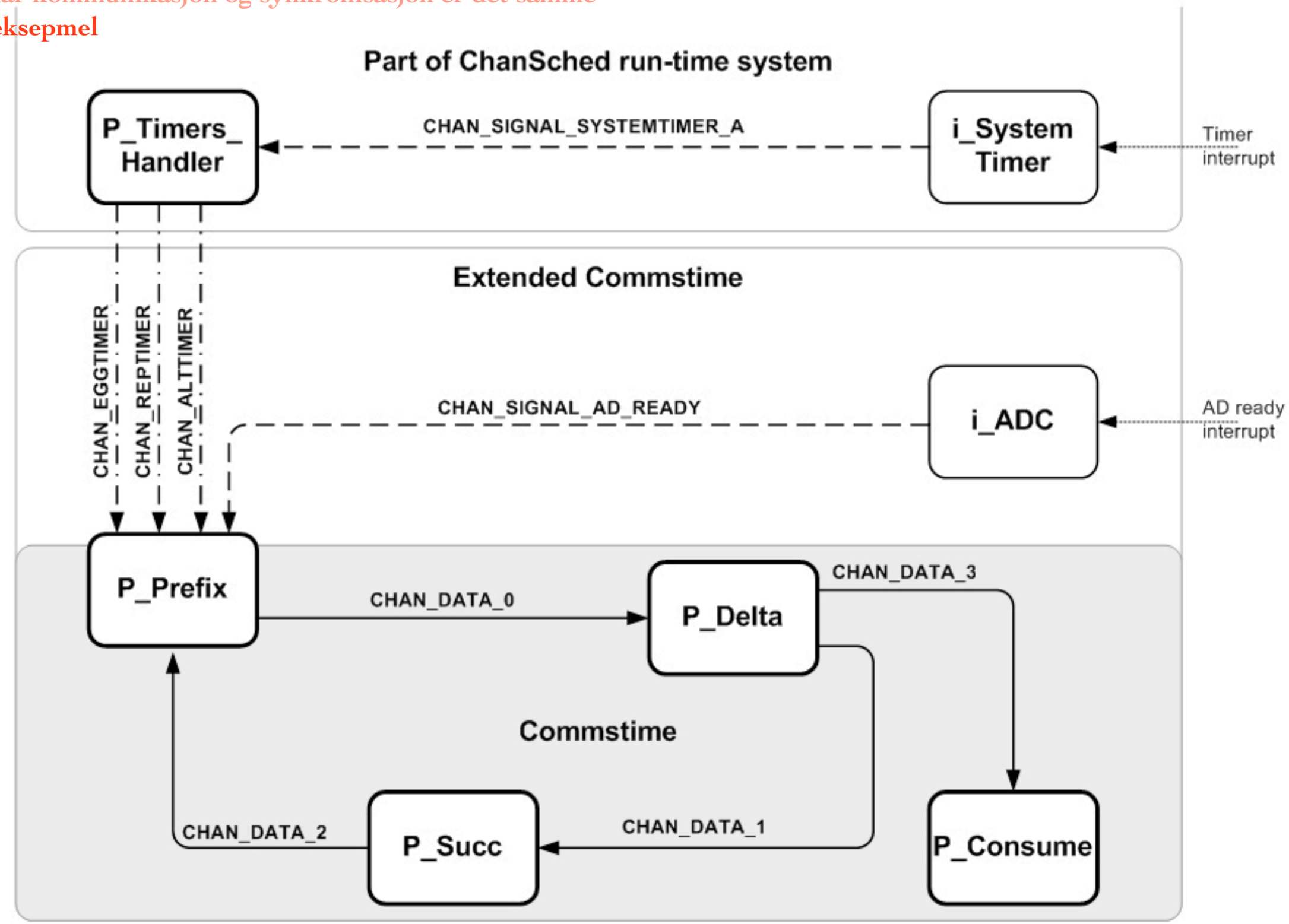
JCSP

C+kjøresystem
«ChanSched»

Forklar hovedfunksjon

Forklar når kommunikasjon og synkronisasjon er det samme

Vis kodeeksepmel

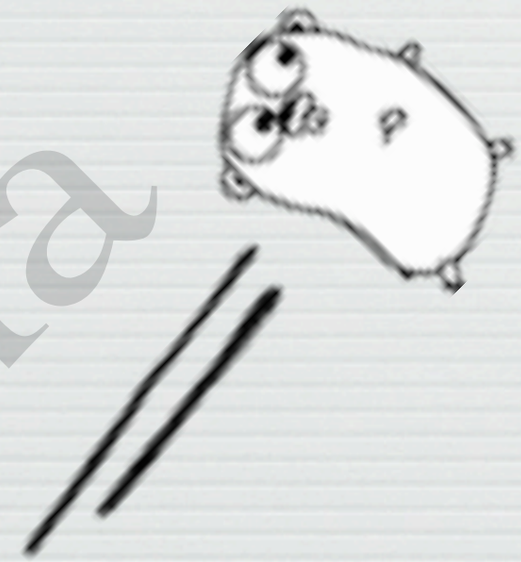


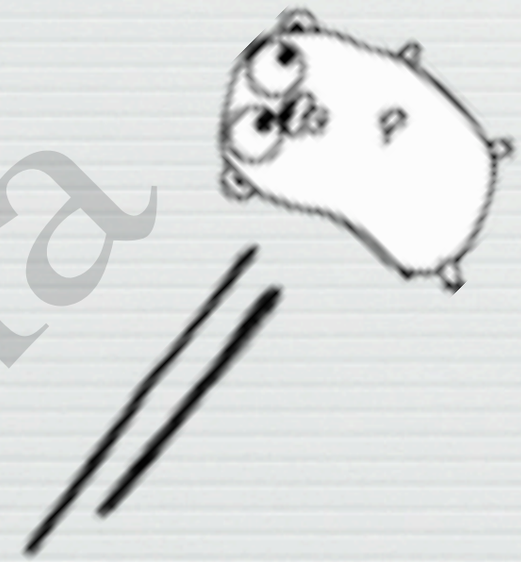
```
01 Void P_Prefix (void)                          // extended “Prefix”
02 {
03   Prefix_CP_a CP = (Prefix_CP_a)g_CP; // get proc Context from Scheduler
04   PROCTOR_PREFIX()                            // jump table (see Section 2)
05   ...  some initialisation
06   SET_EGGTIMER (CHAN_EGGTIMER, LED_Timeout_Tick);
07   SET_REPTIMER (CHAN_REPTIMER, ADC_TIME_TICKS);
08   CHAN_OUT (CHAN_DATA_0,Data_0);       //first output
09   while (TRUE)
10   {
11     ALT();                                    // this is the needed ”PRI_ALT”
12       ALT_EGGREPTIMER_IN (CHAN_EGGTIMER);
13       ALT_EGGREPTIMER_IN (CHAN_REPTIMER);
14       ALT_SIGNAL_CHAN_IN (CHAN_SIGNAL_AD_READY);
15       ALT_CHAN_IN        (CHAN_DATA_2,Data_2);
16       ALT_ALTTIMER_IN    (CHAN_ALTTIMER,TIME_TICKS_100_MSECS);
17     ALT_END();
18     switch (g_ThisChannelId)
19     {
20       ...  process the guard that has been taken, e.g. CHAN_DATA_2
21       CHAN_OUT (CHAN_DATA_0,Data_0);
22     };
23   }
24 }
```

Fjorårets side

```
01 Void P_Prefix (void)                          // extended “Prefix”
02 {
03   Prefix_CP_a CP = (Prefix_CP_a)g_CP; // get proc Context from Scheduler
04   PROCTOR_PREFIX()                            // jump table (see Section 2)
05   ...  some initialisation
06   SET_EGGTIMER (CHAN_EGGTIMER, CP->LED_Timeout_Tick);
07   SET_REPTIMER (CHAN_REPTIMER, ADC_TIME_TICKS);
08   CHAN_OUT (CHAN_DATA_0,&CP->Data_0,sizeof(CP->Data_0)); //first output
09   while (TRUE)
10   {
11     ALT();                                    // this is the needed ”PRI_ALT”
12       ALT_EGGREPTIMER_IN (CHAN_EGGTIMER);
13       ALT_EGGREPTIMER_IN (CHAN_REPTIMER);
14       ALT_SIGNAL_CHAN_IN (CHAN_SIGNAL_AD_READY);
15       ALT_CHAN_IN        (CHAN_DATA_2,&CP->Data_2,sizeof(CP->Data_2));
16       ALT_ALTTIMER_IN    (CHAN_ALTTIMER, TIME_TICKS_100_MSECS);
17     ALT_END();
18     switch (g_ThisChannelId)
19     {
20       ...  process the guard that has been taken, e.g. CHAN_DATA_2
21       CHAN_OUT (CHAN_DATA_0, &CP->Data_0, sizeof (CP->Data_0));
22     };
23   }
24 }
```

Kanal i Amsterdam, nov. 09





Ada

GCC

Transter-

preter

KRoC

JCSP

C+kjøresystem

Google

Ada

Transter-

preter

KRoC

JCSP

C+kjøresystem

"Concurrent"

Go (programming language)

From Wikipedia, the free encyclopedia

Go is a compiled, concurrent programming language. It is being developed by Google[2], with initial design by Robert Griesemer, Rob Pike and Ken Thompson.

Go	
Paradigm	compiled, concurrent
Appeared in	2009
Designed by	Robert Griesemer, Rob Pike and Ken Thompson
Major implementations	Gc is in C using yacc/bison for the parser and Gccgo has C++ front-end with a recursive descent parser coupled to the standard GCC back end[1].
OS	Linux, Mac OS X
License	BSD
Website	http://golang.org

References

1. ^ http://golang.org/doc/go_faq.html#Implementation
2. ^ Google-go-language (http://www.techcrunch.com/2009/11/10/google-go-language/)

External links

- Go Programming Language Homepage (http://golang.org/)
- Go language FAQ (http://golang.org/doc/go_lang_faq.html)
- The Go Programming Language (http://www.youtube.com/watch?v=rKnDgT73v8s) at YouTube (requires Adobe Flash)
- IRC : #go-nuts (irc://irc.freenode.net/go-nuts) on irc.freenode.net
- Mailing list : http://groups.google.com/group/golang-nuts

Categories: Computer language stubs | C++ programming language family | Curly bracket programming languages | Concurrent programming languages | Google

http://en.wikipedia.org/wiki/Go_(programming_language)

Go (programming language) - Wikipedia, the free encyclopedia

11.11.09 09.17

Nytt i fjor

Go (programming language)

From Wikipedia, the free encyclopedia

Go is a compiled, concurrent programming language. It is being developed by Google[2], with initial design by Robert Griesemer, Rob Pike and Ken Thompson.

Go	
Paradigm	compiled, concurrent
Appeared in	2009
Designed by	Robert Griesemer, Rob Pike and Ken Thompson
Major implementations	Gc is in C using yacc/bison for the parser and Gccgo has C++ front-end with a recursive descent parser coupled to the standard GCC back end[1].
OS	Linux, Mac OS X
License	BSD
Website	http://golang.org

References

1. ^ http://golang.org/doc/go_faq.html#Implementation
2. ^ Google-go-language (http://www.techcrunch.com/2009/11/10/google-go-language/)

External links

- Go Programming Language Homepage (http://golang.org/)
- Go language FAQ (http://golang.org/doc/go_lang_faq.html)
- The Go Programming Language (http://www.youtube.com/watch?v=rKnDgT73v8s) at YouTube (requires Adobe Flash)
- IRC : #go-nuts (irc://irc.freenode.net/go-nuts) on irc.freenode.net
- Mailing list : http://groups.google.com/group/golang-nuts

Categories: Computer language stubs | C++ programming language family | Curly bracket programming languages | Concurrent programming languages | Google

http://en.wikipedia.org/wiki/Go_(programming_language)

Certain concurrency-related structural conventions of Go

channels and
alternative channel inputs

are borrowed from Tony Hoare's CSP.

http://en.wikipedia.org/wiki/Go_(programming_language)

Why build concurrency on the ideas of CSP?

Concurrency and multi-threaded programming have a reputation for difficulty. We believe the problem is due partly to complex designs such as pthreads and partly to overemphasis on low-level details such as mutexes, condition variables, and even memory barriers. Higher-level interfaces enable much simpler code, even if there are still mutexes and such under the covers.

One of the most successful models for providing high-level linguistic support for concurrency comes from Hoare's Communicating Sequential Processes, or CSP. Occam and Erlang are two well known languages that stem from CSP. Go's concurrency primitives derive from a different part of the family tree whose main contribution is the powerful notion of channels as first class objects.

http://golang.org/doc/go_faq.html#csp (2010 - ingen forandring)

http://golang.org/doc/go_lang_faq.html#csp (2009)

Skal vi gå og prøve Go?

Go
vises mer i faget
sanntidsprogrammering på NTNU

http://www.itk.ntnu.no/fag/TTK4145/information/

Fra sekvensielt til parallelt

Fra sekvensielt til parallelt og multikjerne

Fra sekvensielt til parallelt og multikjerne

Øyvind Teig

http://www.teigfam.net/oyvind/