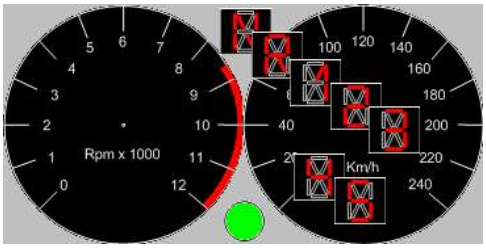
Projekt: Cruisecontrol

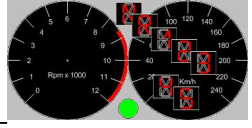
Dato: 18-12-2003

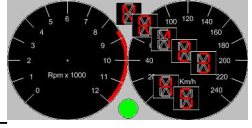
Titel:

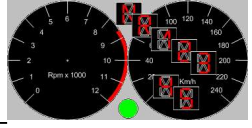
Design
for
Cruisecontroller



Cruise International



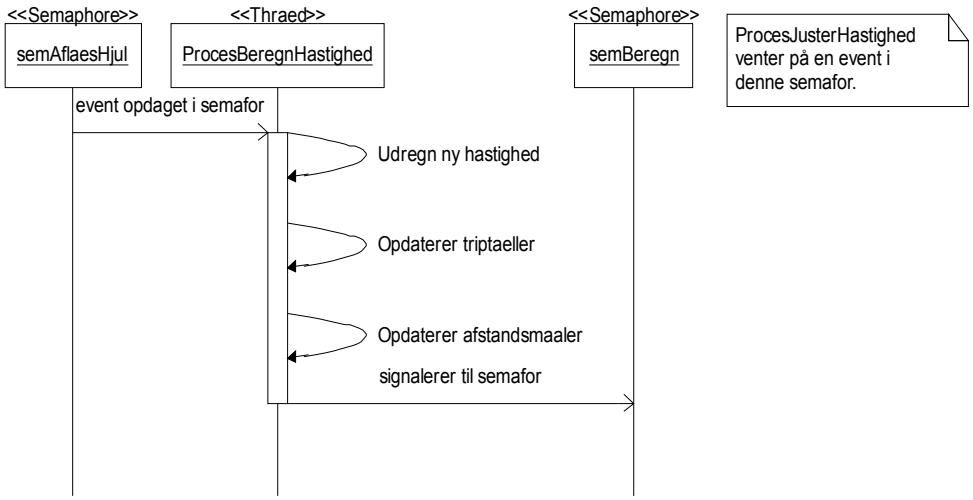




I det følgende beskrives de enkelte klassers interaktion med sekvensdiagrammer. Desuden beskrives klassernes medlemsfunktioners algoritmer.

1 Klasse ProcesBeregnHastighed

1.1 Sekvensdiagram

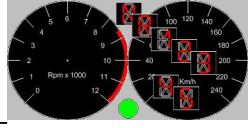


1.2 Beskrivelse af operatør: ProcesBeregnHastighed

*ProcesBeregnHastighed (RTKSemaphore *sAflaesHjul,*
*RTKSemaphore *sNyHastighed)*

1.2.1 Funktionalitet:

Der skal opsættes nogle medlemsdata.



1.2.2 Parametre:

RTKSemaphore *sAflaesHjul

RTKSemaphore *sNyHastighed

1.2.3 Returnering:

Ingen.

1.2.4 Pseudokode:

```
semAflaesHjul = sAflaesHjul
semNyHastighed = sNyHastighed
```

1.3 Beskrivelse af operatør: run

void run(void)

1.3.1 Funktionalitet:

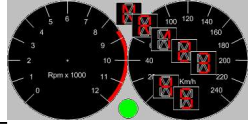
I denne funktion skal klassens funktionalitet implementeres, da funktionen vil fungerer som en procestråd.

1.3.2 Parametre:

Ingen.

1.3.3 Returnering:

Ingen.



1.3.4 Pseudokode:

```
while( true )
{
	wait( *sAflaesHjul )
	hastighed = ny hastighed
	triptaeller += ½ * hjulomkreds
	afstandsmaaler += ½ * hjulomkreds
	signal( *sNyHastighed )
}
```

1.4 Beskrivelse af operatør: getHastighed

unsigned int getHastighed(void)

1.4.1 Funktionalitet:

Funktionen skal returnerer den aktuelle hastighed.

1.4.2 Parametre:

Ingen.

1.4.3 Returnering:

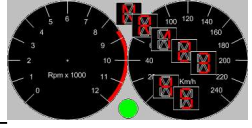
Der skal returneres km/t, som skal ligge i intervallet 0-240.

1.4.4 Pseudokode:

```
return hastighed
```

1.5 Beskrivelse af operatør: setTripTaeller

void setTripTaeller(unsigned long vaerdi)



1.5.1 Funktionalitet:

Triptællerens data skal indstilles til "vaerdi".

1.5.2 Parametre:

unsigned long vaerdi

1.5.3 Returnering:

Ingen.

1.5.4 Pseudokode:

```
triptaeller = vaerdi
```

1.6 Beskrivelse af operatør: getTriptaeller

unsigned long getTriptaeller(void)

1.6.1 Funktionalitet:

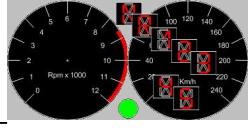
Funktionen skal returnerer den afmålte trip-værdi.

1.6.2 Parametre:

Ingen.

1.6.3 Returnering:

Trip-værdien skal returneres i meter.



1.6.4 Pseudokode:

```
return triptaeller
```

1.7 Beskrivelse af operatør: getAfstandsmaaler

unsigned long getAfstandsmaaler(void)

1.7.1 Funktionalitet:

Funktionen skal returnerer den totale aflagte afstand.

1.7.2 Parametre:

Ingen.

1.7.3 Returnering:

Den aflagte afstand skal returneres i meter.

1.7.4 Pseudokode:

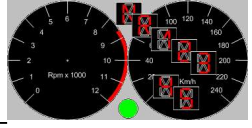
```
return afstandmaaler
```

1.8 Beskrivelse af operatør: readLog

readLog(void)

1.8.1 Funktionalitet:

Funktionen skal skrive hjulomkreds, triptaeller og afstandsmaaler til en fil kaldet cruise.cfg.



1.8.2 Parametre:

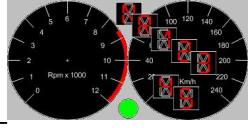
Ingen.

1.8.3 Returnering:

Ingen.

1.8.4 Pseudokode:

```
open( cruise.cfg )
hjulomkreds << cruise.cfg
triptaeller << cruise.cfg
afstandsmaaler << cruise.cfg
close( cruise.cfg )
```



1.9 Beskrivelser af attributter

RTKSemaphore *semAflaesHjul:
Når der opdages en event i denne semafor betyder det at køretøjets hjul har drejet en halv omgang.

RTKSemaphore *semNyHastighed:
Når en ny hastighed er udregnet skal der signaleres til denne semafor.

RTKSemaphore semTriptaeller:
Når der skrives til attributten triptaeller, så skal denne semafor bruges til at skrivebestykke den.

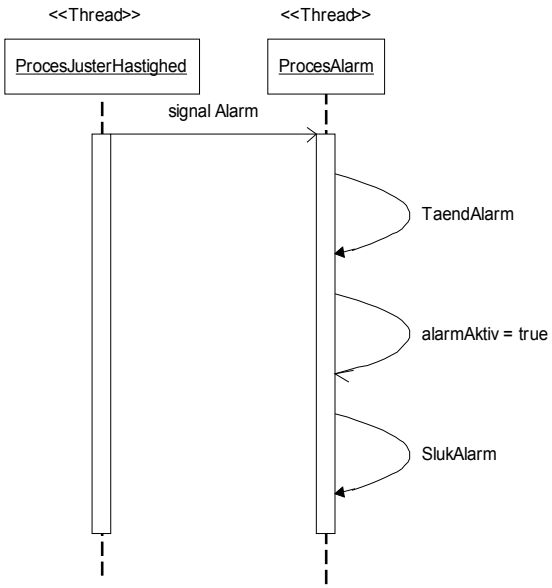
unsigned int hjulomkreds:
Denne integer skal indeholde værdien for køretøjets hjuomkreds i millimeter.

unsigned long triptaeller:
Denne integer skal indeholde den aktuelle triptæller værdi.

unsigned long afstandsmaaler:
Denne integer skal indeholde den aktuelle værdi for den aflagte afstand.

2 Klasse ProcesAlarm

2.1 Sekvensdiagram



2.2 Beskrivelse af operatør: ProcesAlarm

*ProcesAlarm(RTKSemaphore *sAlarm)*

2.2.1 Funktionalitet:

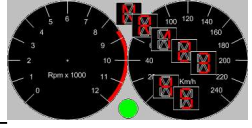
Der skal opsættes nogle medlemsdata.

2.2.2 Parametre:

RTKSemaphore *sAlarm

2.2.3 Returnering:

Ingen.



2.2.4 Pseudokode:

```
semAlarm = sAlarm
```

2.3 Beskrivelse af operatør: run

void run(void)

2.3.1 Funktionalitet:

I denne funktion skal klassens funktionalitet implementeres, da funktionen vil fungerer som en procestråd.

2.3.2 Parametre:

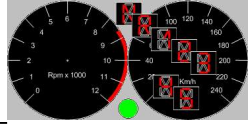
Ingen.

2.3.3 Returnering:

Ingen.

2.3.4 Pseudokode:

```
while( true )
{
      wait( *semAlarm )
      alarmAktiv = true
      start beep på pc-speaker
      delay i 5 sekunder
      stop beep på pc-speaker
      alarmAktiv = false
}
```



2.4 Beskrivelse af operatør: getAlarmAktiv

bool getAlarmAktiv(void)

2.4.1 Funktionalitet:

Funktionen skal returnerer om der alarmeres eller ej.

2.4.2 Parametre:

Ingen.

2.4.3 Returnering:

Hvis der alarmeres, så skal true returneres. Ellers skal false returneres.

2.4.4 Pseudokode:

```
return alarmAktiv
```

2.5 Beskrivelse af attributter

RTKSemaphore *semAlarm:

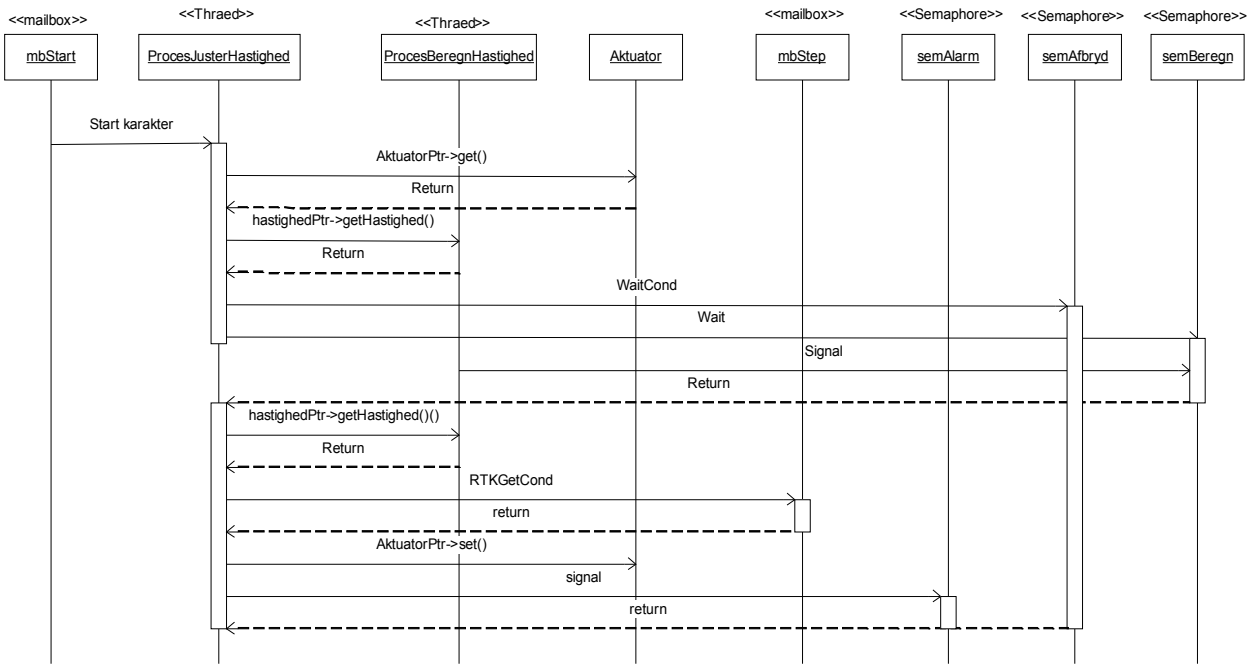
Når der opdages en event i denne semafor betyder det at der skal alarmeres.

bool alarmAktiv

Denne attribut vil være true imens der alarmeres.

3 Klasse ProcesJusterHastighed

3.1 Sekvensdiagram



3.2 Beskrivelse af operatør: ProcesJusterHastighed

*ProcesJusterHastighed(Aktuator *aPtr*

*ProcesBeregnHastighed *hPtr*

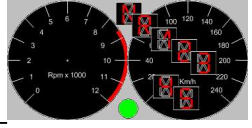
*RTKSemaphore *sBeregn*

*RTKSemaphore *sAfbryd*

*RTKMailbox *mbSrt*

*RTKMailbox *mbStp*

*RTKSemaphore *sAlarm)*



3.2.1 Funktionalitet:

Der skal opsættes nogle medlemsdata.

3.2.2 Parametre:

Aktuator *AktuatorPtr

ProcesBeregnHastighed *hastighedPtr

RTKSemaphore *semBeregn

RTKSemaphore *semAfbryd

RTKMailbox *mbStart

RTKMailbox *mbstep

RTKSemaphore *semAlarm

3.2.3 Returnering:

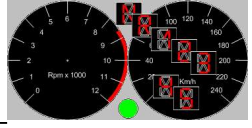
Ingen.

3.2.4 Pseudokode:

```
AktuatorPtr = aPtr
hastighedPtr = hPtr
semBeregn = sBeregn
semAfbryd = sAfbryd
mbStart = mbSrt
mbStep = mbStp
semAlarm = sAlarm
```

3.3 Beskrivelse af operatør: run

void run(void)



3.3.1 Funktionalitet:

I denne funktion skal klassens funktionalitet implementeres, da funktionen vil være en procestråd.

3.3.2 Parametre:

Ingen.

3.3.3 Returnering:

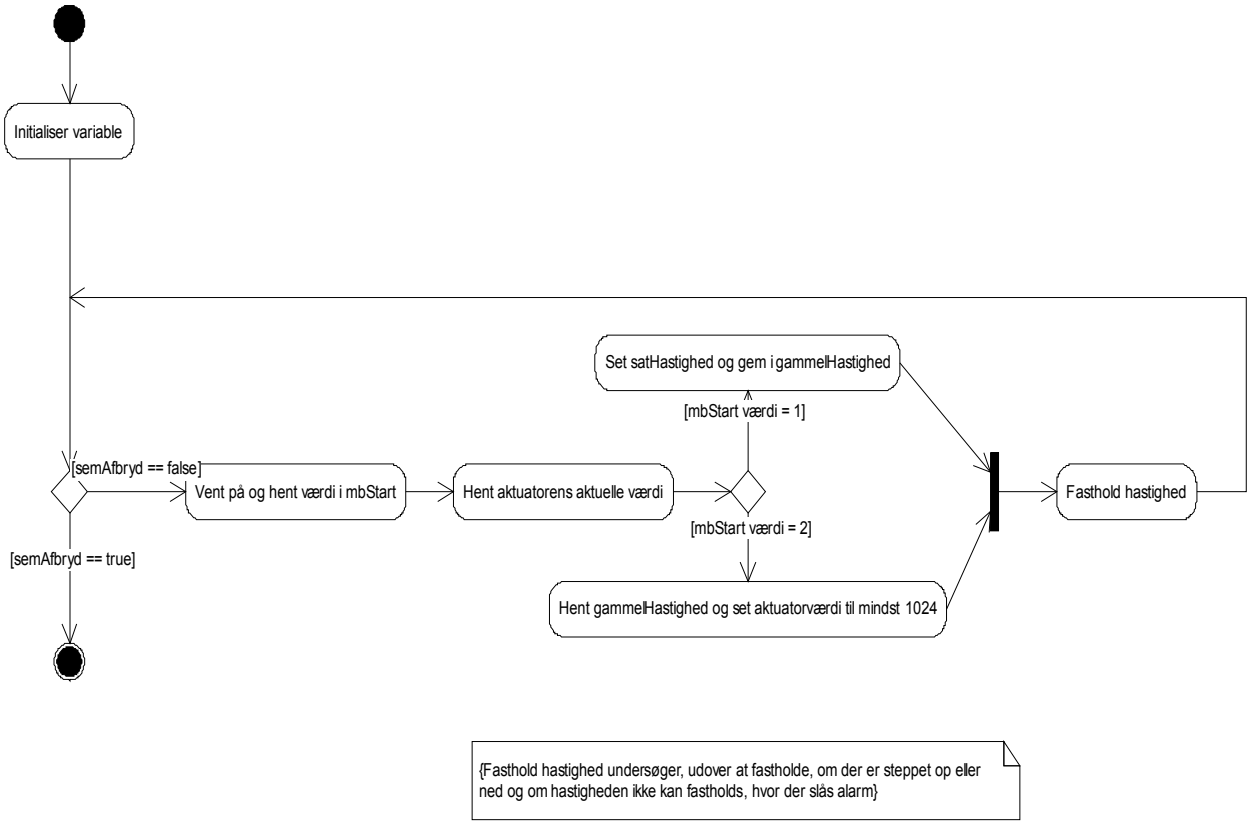
Ingen.

3.3.4 Pseudokode:

```
while( true )
{
	vent på start eller resume besked fra "mbStart"
	if( besked == start )
		cruisehastighed = ProcesBeregnHastighed->getHastighed()
	if( besked == resume )
		cruisehastighed = sidste cruising hastighed
	while( der ingen event er i *semAfbryd )
	{
		wait( *semBeregn )
		aktuelhastighed = ProcesBeregnHastighed->getHastighed()
		juster "aktuelhastighed" til "cruisehastighed"
		if( aktuelhastighed == cruisehastighed )
			break
	}
	while( der ingen event er i *semAfbryd )
	{
		wait( *semBeregn )
		aktuelhastighed = ProcesBeregnHastighed->getHastighed()
		juster "aktuelhastighed" til "cruisehastighed"
		if( aktuelhastighed er +/-5Km/h fra cruisehastighed )
```

```
                signal( *semAlarm )
            besked = undersøg om der er en besked i mbStep
            if( besked == hæv cruisehastighed )
                cruisehastighed += 1
            if( besked ==  sink cruisehastighed )
                cruisehastighed -= 1
        }
}
```

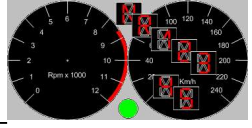
3.4 Aktivitetsdiagram



3.5 Beskrivelse af attributter

Aktuator *AktuatorPtr:

Denne pointer til et Aktuator objekt, skal bruges til at hente den aktuelle aktuatorværdi og sætte den aktuelle aktuatorværdi.



ProcesBeregnHastighed *hastighedPtr:
Denne pointer til et ProcesBeregnHastighed objekt, skal bruges til at hente den aktuelle hastighed.

RTKSemaphore *semBeregn:
Hver gang en ny aktuel hastighed er beregnet skal der opdages en event i til denne semafor.

RTKSemaphore *semAfbryd:
Når cruising ønskes afbrudt skal der opdages en event i denne semafor.

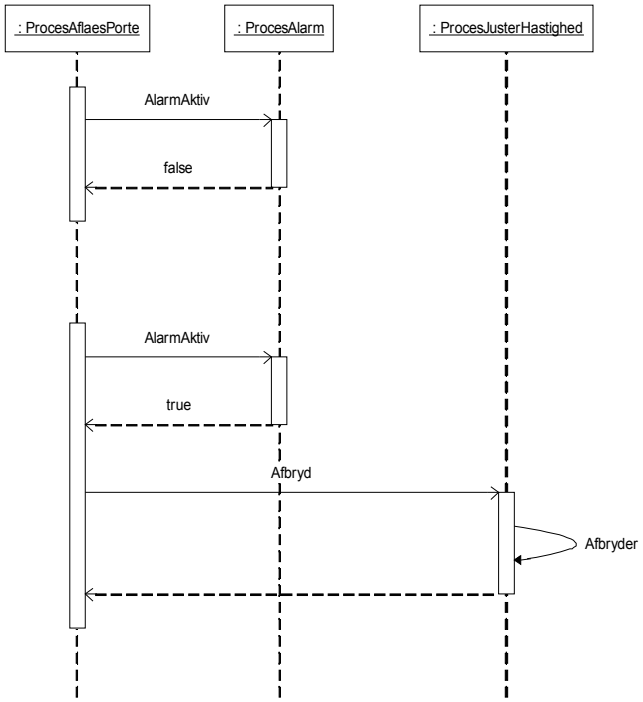
RTKMailbox *mbStart:
Når chaufføren ønsker at cruise vil denne mailbox indeholde en besked.

RTKMailbox *mbstep:
Når chaufføren ønsker at hæve eller sinke hastigheden under cruising, vil denne mailbox indeholde en besked.

RTKSemaphore *semAlarm:
Når den aktuelle hastighed afviger med mere end +/-5Km/h fra cruisehastigheden, skal der signaleres til denne semafor.

4 Klasse ProcesAflaesPorte

4.1 Sekvensdiagram



4.2 Beskrivelse af operatør: ProcesAflaesPorte

*ProcesAflaesPorte(Aktuator *aPtr,*

*RTKMailbox *mbStrt,*

*RTKSemaphore *sAfbryd,*

*RTKMailbox *mbStp,*

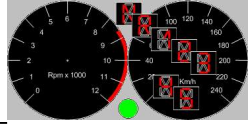
*ProcesAlarm *pAPtr*

*ProcesBeregnHastighed *pBPtr*

*Timer *tPtr)*

4.2.1 Funktionalitet:

Der skal opsættes nogle medlemsdata.



4.2.2 Parametre:

Aktuator *aPtr

RTKMailbox *mbStrt

RTKSemaphore *sAfbryd

RTKMailbox *mbStp

ProcesAlarm *pAPtr

ProcesBeregnHastighed *pBPtr

Timer *tPtr

4.2.3 Returnering:

Ingen.

4.2.4 Pseudokode:

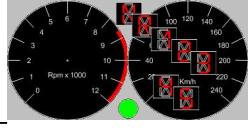
```
aktuatorPtr = aPtr
mbStart = mbSrt
semAfbryd = sAfbryd
mbStep = mbStp
pAlarmPtr = pAPtr
pBeregnPtr = pBPtr
timerPtr = tPtr
```

4.3 Beskrivelse af operatør: run

void run(void)

4.3.1 Funktionalitet:

I denne funktion skal klassens funktionalitet implementeres, da funktionen vil fungerer som en procestråd.



4.3.2 Parametre:

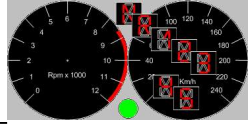
Ingen.

4.3.3 Returnering:

Ingen.

4.3.4 Pseudokode:

```
while( true )
{
      portWord = aflæs port B
      if( pAlarmPtr->getAktivAlarm() )
            if( der cruises )
                  signal( *semAfbryd )
      else if( chaufføren har trykket på start/afbryd knappen )
            if( der cruises )
                  signal( *semAfbryd )
            if( der cruises ikke )
                  skriv start-besked til "mbStart"
      else if( chaufføren har trykket på resume )
            if( der cruises ikke )
                  skriv resume-besked til "mbStart"
      if( portWord == nulstil triptæller )
            pBeregnPtr->setTriptaeller( 0 )
      if( der cruises )
      {
            if( portWord == hæv cruisehastigheden )
                  skriv hæv-besked til "mbStep"
            if( portWord == sink cruisehastigheden )
                  skriv sink-besked til "mbStep"
      }
      portWord = aflæs port A
      gear = find aktulle geartrin i "portWord"
      if( der cruises )
```



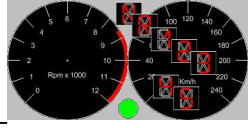
```
            if( gear != 4 || gear != 5 )
                  signal( *semAfbryd )
      omdr = timerPtr->read()
}
```

4.4 Beskrivelse af operatør: getGear

char getGear(void)

4.4.1 Funktionalitet:

Funktionen skal returnerer det geartrin som køretøjet befinder sig i.



4.4.2 Parametre:

Ingen.

4.4.3 Returnering:

Der skal returneres en af følgende værdier: -1 for frigear, 0 for bakgear, 1 for første, 2 for andet, 3 for tredje, 4 for fjerde og 5 for femte gear.

4.4.4 Pseudokode:

```
return gear
```

4.5 Beskrivelse af operatør: getCruising

bool getCruising(void)

4.5.1 Funktionalitet:

Funktionen returnerer om der cruises eller ej.

4.5.2 Parametre:

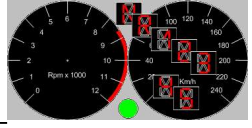
Ingen.

4.5.3 Returnering:

Hvis der cruises skal true returneres ellers skal false returneres.

4.5.4 Pseudokode:

```
return activeState
```



4.6 Beskrivelse af operatør: getOmdr

int getOmdr(void)

4.6.1 Funktionalitet:

Funktionen skal returnerer de aktuelle omdrejninger i motoren.

4.6.2 Parametre:

Ingen.

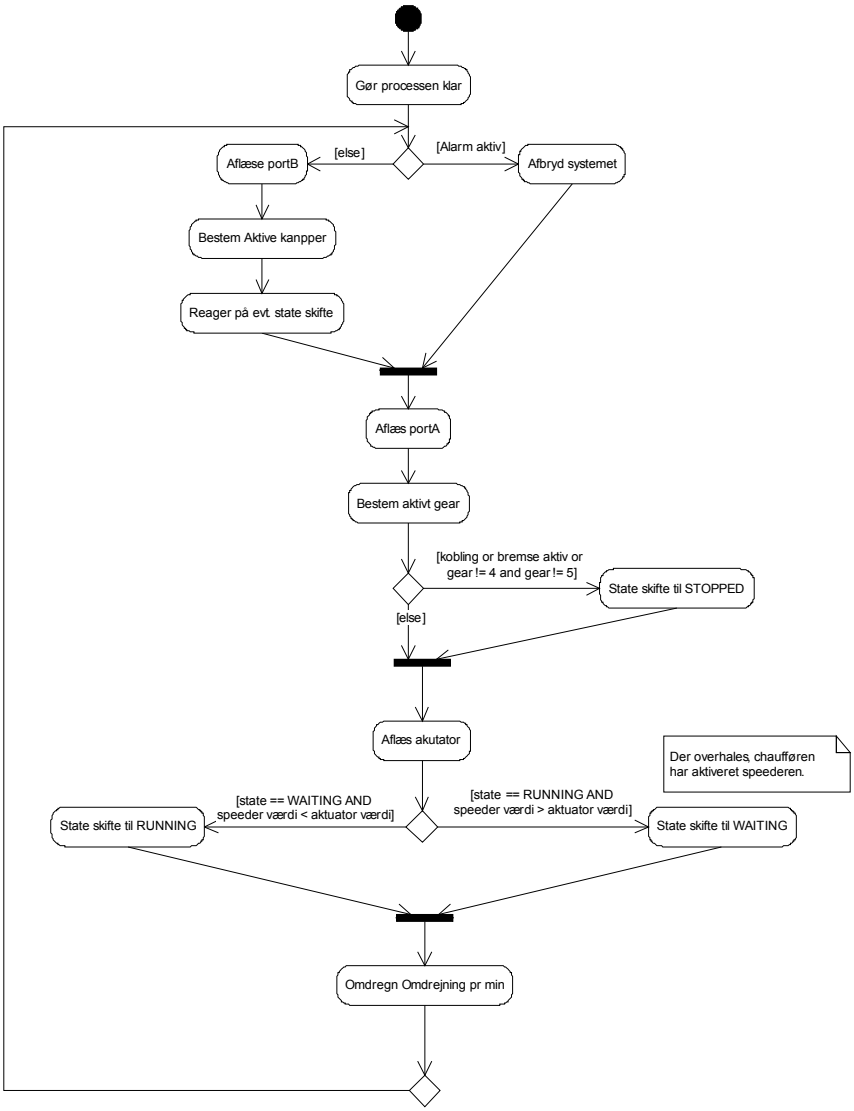
4.6.3 Returnering:

Der skal returneres en integer der svarer til omdrejningstallet i motoren.

4.6.4 Pseudokode:

```
return omdr
```

4.7 Aktivitetsdiagram



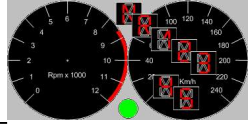
4.8 Beskrivelse af attributter

Aktuator *aktuatorPtr:

Denne pointer til et Aktuator objekt, skal bruges til at hente den aktuelle aktuatorværdi.

RTKMailbox *mbStart:

Hver gang chaufføren ønsker at starte eller genoptage cruising, så skal der sendes en besked til denne mailbox.



RTKSemaphore *semAfbryd:
Når chaufføren trykker på kobling, bremse, afbryd, skifter geartrin eller der alarmeres, imens der cruises, skal der sendes et signal til denne semafor.

RTKMailbox *mbStep:
Når chaufføren vil hæve eller sinke den ønskede crusinghastighed, imens der cruises, skal der sendes en besked til denne mailbox.

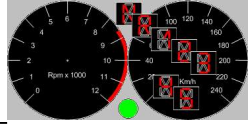
ProcesAlarm *pAlarmPtr:
Denne pointer til et ProcesAlarm objekt, skal bruges til at undersøge om der alarmeres.

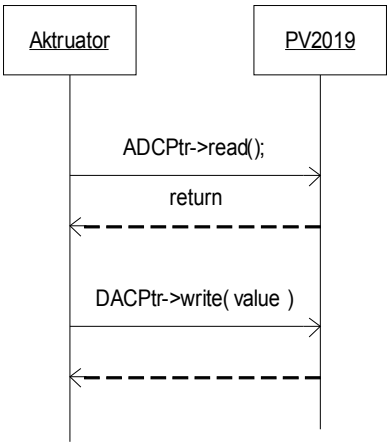
ProcesBeregnHastighed *pBeregnPtr:
Denne pointer til et ProcesBeregnHastighed objekt, skal bruges til at nulstille triptælleren.

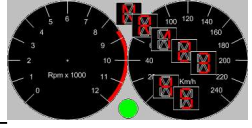
Timer *timerPtr:
Denne pointer til et Timer objekt, skal bruges til at hente de aktuelle omdrejninger i motoren.

5 Klasse Aktuator

5.1 Sekvensdiagram:




Aktruator
PV2019
ADCPtr->read();
return
DACPtr->write(value)



5.2 Beskrivelse af operatør: Aktuator

Aktuator(void)

5.2.1 Funktionalitet:

Der skal opsættes nogle medlemsdata.

5.2.2 Parametre:

Ingen.

5.2.3 Returnering:

Ingen.

5.2.4 Pseudokode:

```
DACPtr = opret nyt PV2019_DAC_V objekt
ADCPtr = opret nyt PV2019_ADC_V objekt
currentValue = 0
```

5.3 Beskrivelse af operatør: get

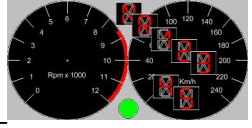
int get(void)

5.3.1 Funktionalitet:

Funktionen skal aflæse A/D konverteren ved at kalde read() i klassen PV2019_ADC_V.

5.3.2 Parametre:

Ingen.



5.3.3 Returnering:

Der returneres en linier værdi i intervallet 0-4095, som tilsvarer det tryk der er på speederpedalen.

5.3.4 Pseudokode:

```
return ADCPtr->read()
```

5.4 Beskrivelse af operatør: set

void set(int value)

5.4.1 Funktionalitet:

Funktionen skal skrive en værdi, som skal ligge i intervallet 0-4095, til D/A konverteren og gemme den i en variabel, hvis den er forskellig fra 0.

5.4.2 Parametre:

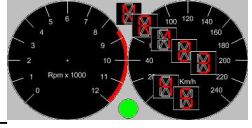
int value

5.4.3 Returnering:

Ingen.

5.4.4 Pseudokode:

```
if( 0 < value && value < 4096 )
{
      currentValue = value
      DACPtr->write( value )
}
```



5.5 Beskrivelse af operatør: getCurrentValue

int getCurrentValue(void)

5.5.1 Funktionalitet:

Funktionen skal returnerer den aktuelle værdi som er sendt til D/A konverteren.

5.5.2 Parametre:

Ingen.

5.5.3 Returnering:

Funktionen returnerer en værdi i intervallet 0-4095, som er den sidste brugte parameter ved funktionen set(int value).

5.5.4 Pseudokode:

```
return currentValue
```

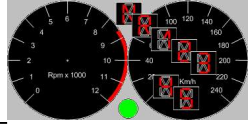
5.6 Beskrivelse af attributter

PV2019 pv2019:

Dette objekt skal gives som parameter, når objekter til ADCPtr og DACPtr oprettes.

PV2019_ADC_V *ADCPtr:

Dette objekt bruges til at læse værdien fra A/D konverteren.



PV2019_DAC_V *DACPtr:

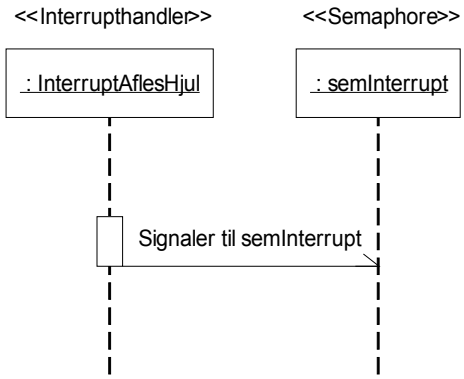
Dette object bruges til at skrive en ny værdi til D/A konverteren.

int currentValue:

Her skal den sidste sendte værdi til D/A konverteren gemmes.

6 Klasse InterruptAflaesHjul

6.1 Sekvensdiagram



6.2 Beskrivelse af operatør: InterruptAflaesHjul

InterruptAflaesHjul(unsigned char nyIRQ,
*RTKSemaphore *sInterrupt)*

6.2.1 Funktionalitet:

Der skal opsættes nogle medlemsdata og en interrupthandler.

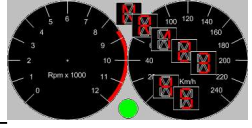
6.2.2 Parametre:

unsigned char nyIRQ

RTKSemaphore *sInterrupt

6.2.3 Returnering:

Ingen.



6.2.4 Pseudokode:

```
IRQ = nyIRQ
semInterrupt = sInterrupt
opret interrupthandler på "IRQ"
```

6.3 Beskrivelse af operatør: ~InterruptAflaesHjul

~InterruptAflaesHjul(void)

6.3.1 Funktionalitet:

Interrupthandleren skal nedlægges i destructor.

6.3.2 Parametre:

Ingen.

6.3.3 Returnering:

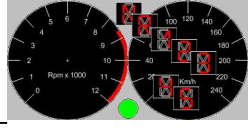
Ingen.

6.3.4 Pseudokode:

```
Nedlæg interrupthandler på IRQ
```

6.4 Beskrivelse af operatør: RTKAPI handleInt

void RTKAPI handleInt(void)



6.4.1 Funktionalitet:

Hver gang et interrupt forekommer, så skal denne funktion eksekveres. Funktionen skal sende et signal til semaforen, der er givet som parameter i klassens constructor.

6.4.2 Parametre:

Ingen.

6.4.3 Pseudokode:

```
vent på interrupt
signal( *semInterrupt )
```

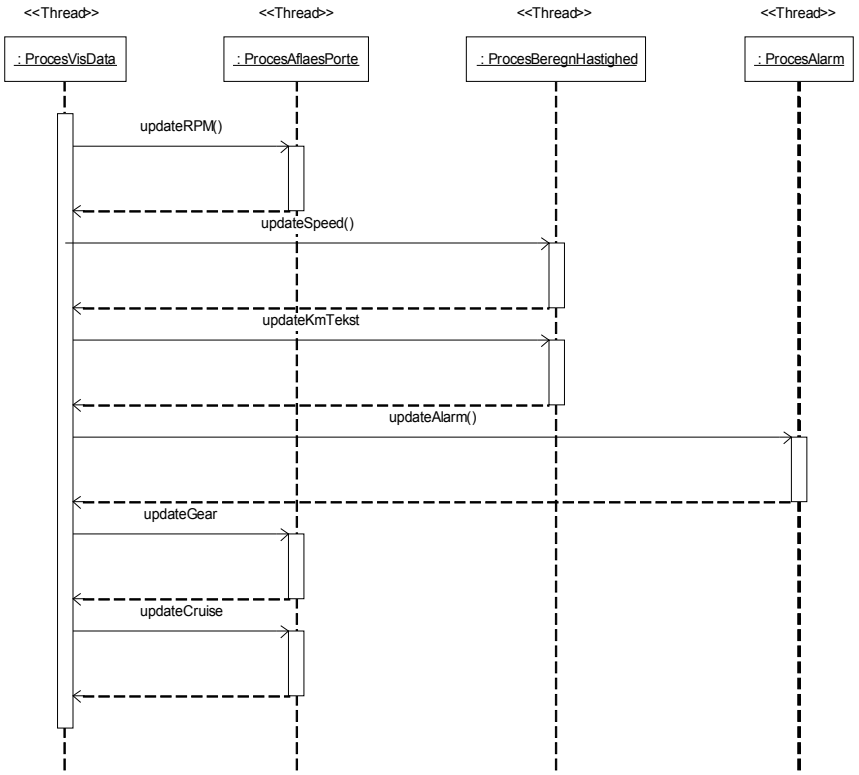
6.5 Beskrivelse af attributter

unsigned char IRQ:
Værdien af dette parameter vil det IRQ nummer, hvor interruptet forekommer.

RTKSemaphore *semInterrupt:
Hver gang et interrupt forekommer, skal et signal sendes til denne semafor.

7 Klasse ProcesVisData

7.1 Sekvensdiagram

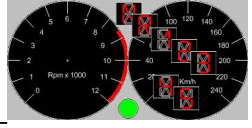


7.2 Beskrivelse af operatør: ProcesVisData

*ProcesVisData(ProcesAlarm *pAPtr,*

*ProcesAflaesPorte *pAPortePtr,*

*ProcesBeregnHastighed *pBHastighedPtr)*



7.2.1 Funktionalitet:

Der skal opsættes nogle medlemsdata og et display.

7.2.2 Parametre:

ProcesAlarm *pAPtr

ProcesAflaesPorte *pAPortePtr

ProcesBeregnHastighed *pBHastighedPtr

7.2.3 Returnering:

Ingen.

7.2.4 Pseudokode:

```
pAlarm = pAPtr
pPorte = pAPortePtr
pBeregn = pBHastighedPtr
opret display vha. RTPeg
```

7.3 Beskrivelse af operatør: run

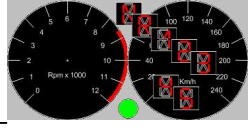
void run(void)

7.3.1 Funktionalitet:

Opdaterer samtlige data løbende, ved hjælp af update-funktionerne.

7.3.2 Parametre:

Ingen.



7.3.3 Returnering:

Ingen.

7.3.4 Pseudokode:

```
while( true )
{
      updateCruise()
      updateGear()
      updateAlarm()
      updateRPM()
      updateSpeed()
      delay( 25 millisec )
}
```

7.4 Beskrivelse af operatør: updateCruise

void updateCruise(void)

7.4.1 Funktionalitet:

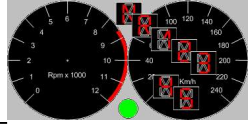
Henter den aktuelle hasstighed vha. ProcesBeregnHastighed.

7.4.2 Parametre:

Ingen.

7.4.3 Pseudokode:

```
hastighed = pBeregn->getHastighed()
```



7.5 Beskrivelse af operatør: updateGear

void updateGear(void)

7.5.1 Funktionalitet:

Henter det aktuelle gear i ProcesAflaesPorte.

7.5.2 Parametre:

Ingen.

7.5.3 Returnering:

Ingen.

7.5.4 Pseudokode:

```
gear = pPorte->getGear()
```

7.5.5 Beskrivelse af operatør: updateAlarm

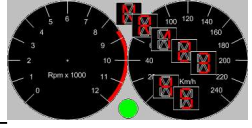
void updateAlarm(void)

7.5.6 Funktionalitet:

Undersøger om der alarmeres vha. ProcesAlarm.

7.5.7 Parametre:

Ingen.



7.5.8 Returnering:

Ingen.

7.5.9 Pseudokode:

```
alarm = pAlarm->getAlarmAktiv()
```

7.6 Beskrivelse af operatør: updateRPM

void updateRPM(void)

7.6.1 Funktionalitet:

Henter det aktuelle RPM i ProcesAflaesPorte.

7.6.2 Parametre:

Ingen.

7.6.3 Returnering:

Ingen.

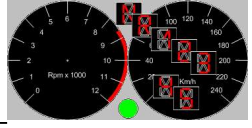
7.6.4 Pseudokode:

```
rpm = pPorte->getOmdr()
```

7.6.5 Beskrivelse af attributter

ProcesAlarm *pAlarm:

Vha. dette objekt skal det undersøges om der alarmeres.



ProcesAflaesPorte *pPorte:

Vha. dette objekt skal det undersøges hvilket geartrin køretøjet befinder sig i, aktuelle motoromdrejninger og om der cruises.

ProcesBeregnHastighed *pBeregn:

Vha. dette objekt skal data om den aktuelle hastighed, triptæller og afstandsmåler hentes.

Følgende attributter er beskrevet i RTPeg dokumentationen:

PegFiniteBitmapDial *omdr,*speed;

PegBitmapLight *back,*alarm,*gear,*cruise;

PegPrompt *kmTekstPtr,*tripTekstPtr,*speedometerPtr;

PegPresentationManager *presentPtr;