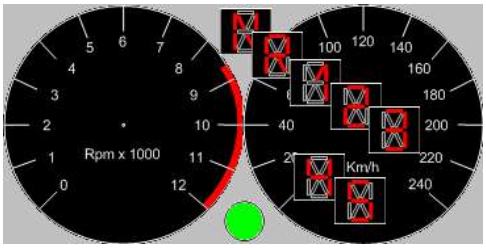
Projekt: Cruisecontrol

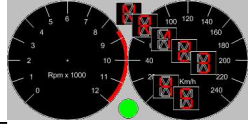
Dato: 18-12-2003

Titel:

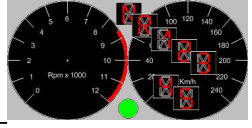
Analyse
for
Cruisecontroller



Cruise International



Indholdsfortegnelse



1. Analyse

Formålet med vores analyse er at finde objekter, opdele objekterne i klasser og specificere deres relationer og opførsel. Vi bruger vores Use Case diagrammer som udgangspunkt og ender med en strukturmodel af vores system samt en opførselsmodel for objekterne i systemet.

1. Objekter

Til at finde objekter har vi brugt tre strategier. Understreg navneord i opgaveformuleringen, identificer services og identificer kausal objekter. For at kontrollere de fundne objekter har vi udført relevante scenarier.

2. Relationer

Da de fleste af vores objekter realiserer processer/tråde er relationerne fastlagt på forhånd, altså med 'uses a' eller 'knows a' forhold. De få resterende objekter er interface klasser, de har også 'uses a' relationer.

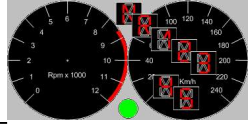
3. Opførsel

Der er til de relevante objekter lavet tilstandsdiagrammer og sekvensdiagrammer.

4. Identifikation af objekter

Ved gennemgang af opgaveformuleringen findes følgende interessante objektkandidater: (Er opstillet efter forekomst og med antal):

Cruise Control System	20
Hastighed / Fart	11
Motor	4
Gearskifte	3
Speedometer	3
Display	2
Terræn	2
Acceleration	1
Brændstofforbrug	1
Dækstørrelse	1
Km-tæller	1
Kørselsøkonomi	1
Omdrejningstallet	1



Sensor	1
Triptæller	1

Som aktører findes:

Fører	5
Speeder	3
Aktuator	2
Bremse	1
Resume Knap	1
Accelerationsknap	1
Gear	1
Hjulomdrejningstæller	1
Kobling	1

Derudover identificeres følgende objektkandidater:
IO686, som er driver til I/O kortets PPI og PIT, på SBC686, og Aktuator som er interface til PV2019 A/D og D/A konverter.

De objekter, hvis navn starter med Proces angiver, at objektet realiseres som en proces/tråd.

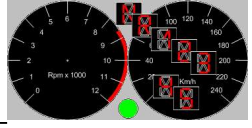
Det ses ud fra objektkandidaterne, at alle knapper, bremse, gear og kobling skal kunne aftastes. Disse aflæses på I/O portene på I/O kortet. dette giver ophav til objektet: *ProcesAflaesPorte*.

Objektets ansvarsområde er Use Case Afbryd Hastighed og Foretag Overhaling og er rygraden i systemet.

Ud fra Use Cases Fasthold Hastighed og Genoptag Hastighed, ses det, at en hastighed skal kunne justeres (fastholdes, genoptages, afbrydes), dette giver ophav til objektet *ProcesJusterHastighed*.

Objektets ansvarsområde bliver som beskrevet Use Cases:

- At kunne fastholde den nuværende hastighed.
- At kunne genoptage fastholdelse af en tidligere fastholdt hastighed.
- At kunne afbryde en fastholdelse af en hastighed.
- At kunne øge eller sænke cruisehastigheden under cruising.



Ud fra aktøren Aktuator oprettes et objekt til at kommunikere med aktuatoren. Dette giver ophav til objektet: *Aktuator.*

Objektets ansvarsområde er at kunne læse den aktuelle speederværdi og sende en værdi til aktuatoren.

Ud fra Use Case Alarmer Chauffør oprettes objektet *ProcesAlarmer.*

Objektets ansvar er at alarmere chaufføren med lyd.
For at kunne fastholde en aftastet hastighed skal denne kunne beregnes ud fra Hjulstørrelse, dertil oprettes objektet *ProcesBeregnHastighed.*

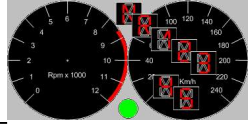
Objektets ansvar er at beregne køretøjets aktuelle hastighed.

Ud fra Speedometer, Display, Km-tæller, Triptæller og Use Casen Vis Data, oprettes objektet *ProcesVisData*, som har samme ansvarsområde som Use Casen Vis data.

Da der I dokumentationen er beskrevet at hjulomdrejningerne aflæses via interrupt oprettes et objekt til håndtering af interruptet. *interruptAflaesHjul*. Dette objekt bruges udelukkende af ProcesBeregnHastighed.

De fundne objekter kan stilles op:

- ProcesAflæsporte
- ProcesBegregnHastighed
- ProcesJusterHastighed
- ProcesAlarm
- ProcesVisData
- Aktuator
- IO686



- PV2019
- InterruptAflaesHjul

5. Sammenknytning af Use Case modellen med objektmodellen

For at undersøge om alle Use Cases er implementeret opstilles hvilke objekter, der opfylder hvilke Use Cases, med det hovedansvarlige objekt først, og hjælpe objekter efterfølgende.

NB: IO686, PV2019 og interruptAflaesHjul er ikke nævnt herunder da de virker som drivere, og PV2019 er ikke implementeret af os selv.

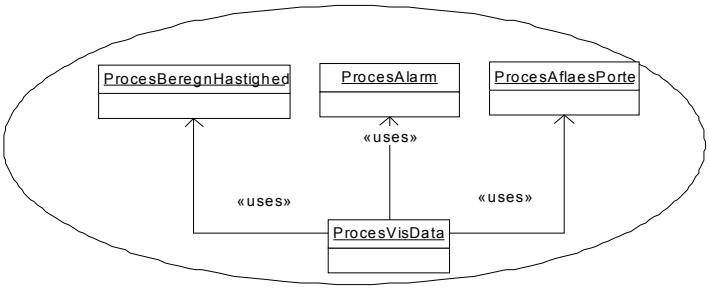
Use Case Kalibrer Hjulstørrelse:

Realiseres af følgende objekter:	**ProcesBegregnHastighed**

Ingen diagram (trivielt)

Use Case Vis data:

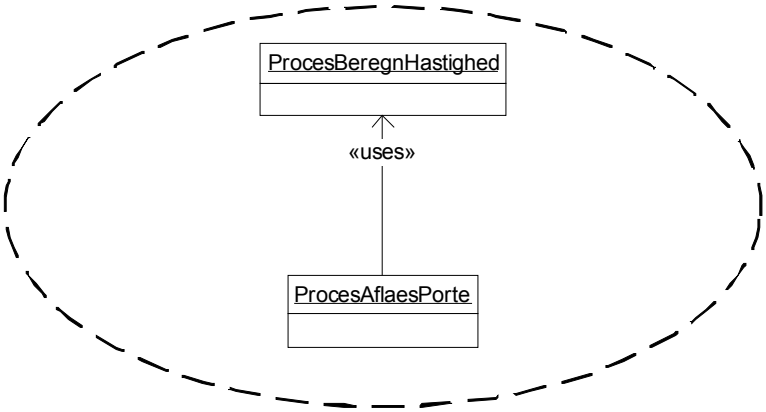
Realiseres af følgende objekter:	**ProcesVisData** ProcesAflaesPorte ProcesBegregnHastighed ProcesAlarm



Figur illustrerer de objekter der indgår i en kollaboration for at realisere Use Case Vis Data.

Use Case Nulstil Triptæller:

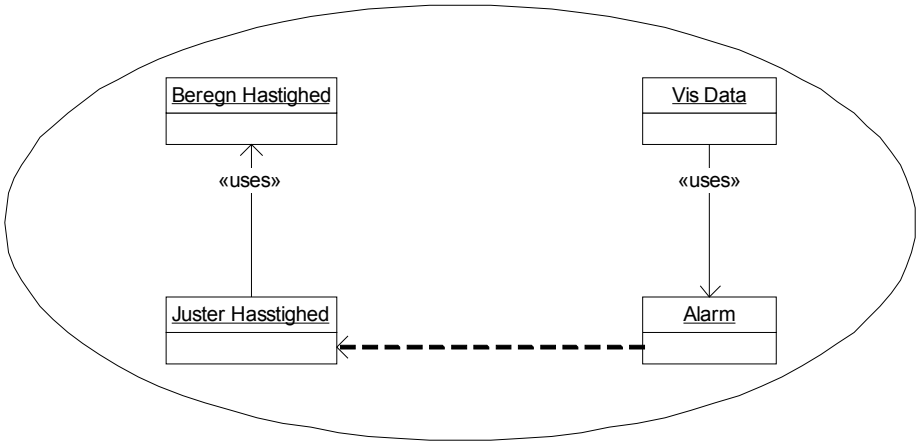
Realiseres af følgende objekter:	**ProcesBegregnHastighed** ProcesAflaesPorte



Figur illustrerer de objekter der indgår i en kollaboration for at realisere Use Case Nulstil Triptæller.

Use Case Alamer chauffør:

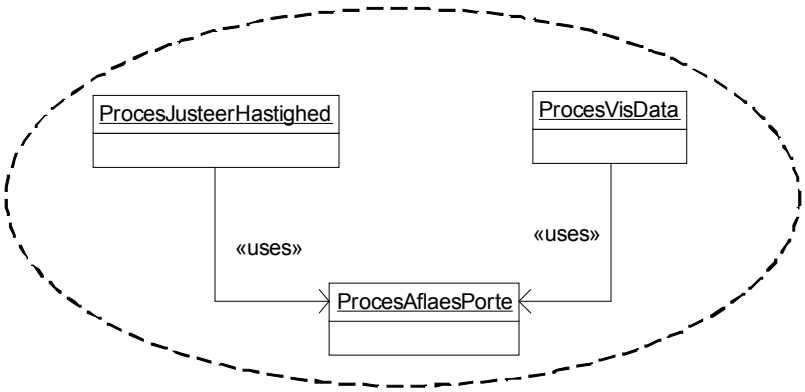
Realiseres af følgende objekter:	**ProcesAlarm** ProcesVisData ProcesJusterHastighed ProcesBeregnHastighed



Figur illustrerer de objekter der indgår i en kollaboration for at realisere Use Case Alamer chauffør.

Use Case Afbryd Hastighed

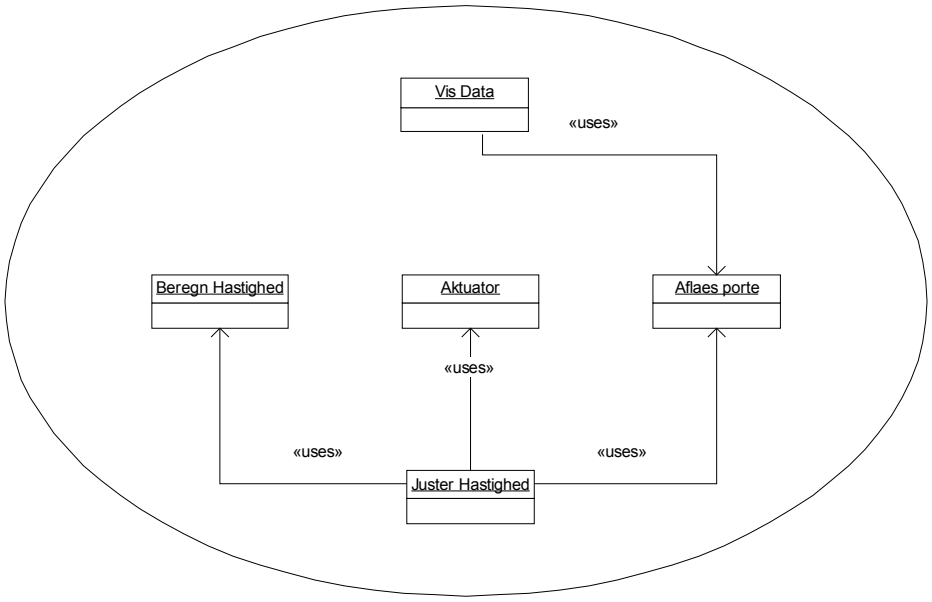
Realiseres af følgende objekter:	**ProcesAflaesPorte** ProcesJusterHastighed ProcesVisData



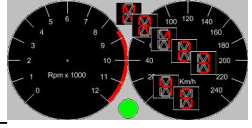
Figur illustrerer de objekter der indgår i en kollaboration for at realisere Use Case Afbryd Hastighed.

Use Case Fasthold Hastighed

Realiseres af følgende objekter:	**ProcesJusterHastighed** ProcesBeregnHastighed ProcesAflaesPorte ProcesVisData Aktuator



Figur illustrerer de objekter der indgår i en kollaboration for at realisere Use Case Fasthold Hastighed.



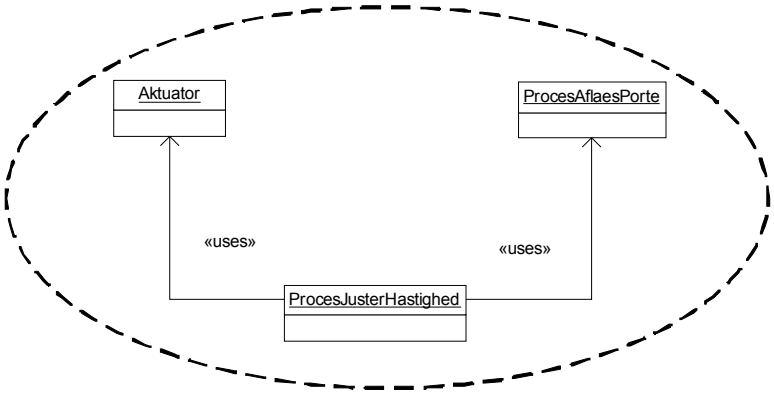
Use Case Genoptag Hastighed

Realiseres af følgende objekter:	**ProcesJusterHastighed** ProcesBeregnHastighed ProcesAflaesPorte ProcesVisData Aktuator

Genoptag Hastighed anvender de samme objekter som Fasthold Hastighed. Se figur.

Use Case Foretag Overhaling

Realiseres af følgende objekter:	**ProcesAflaesPorte** Aktuator ProcesJusterHastighed



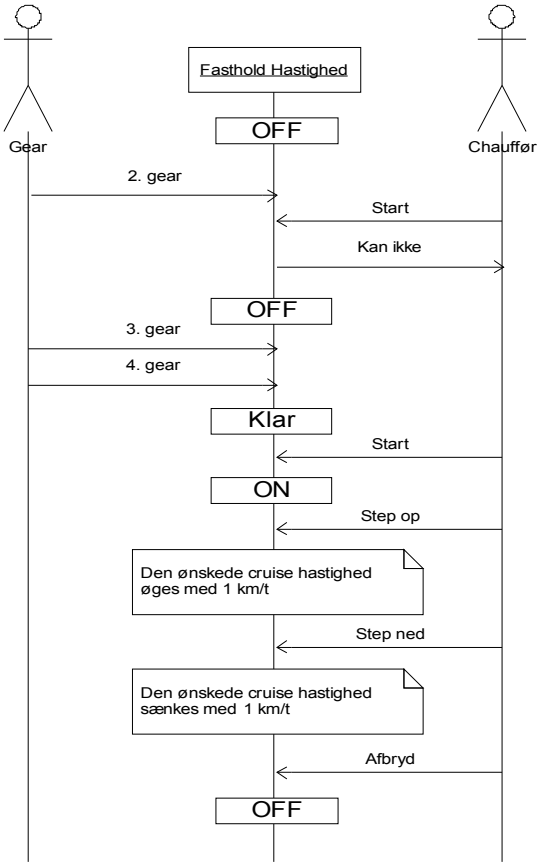
Figur der illustrerer de objekter der indgår i en kollaboration for at realisere Use Case Foretag Overhaling.

6. Use Case kommunikation

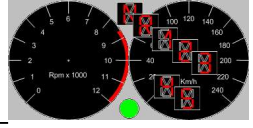
I det følgende illustreres hvorledes de enkelte Use Cases kommunikerer med hinanden og med aktører. Dette illustreres med sekvensdiagrammer, hvor Use Cases er symboliseret med en firkantet box, og aktører med en tændstikmand.

Der er ikke fremstillet sekvensdiagrammer for Use Case Kalibrer, da den kun indeholder at teknikeren redigerer en værdi, og nulstil triptæller, da den kun indebærer, at chaufføren trykker på reset.

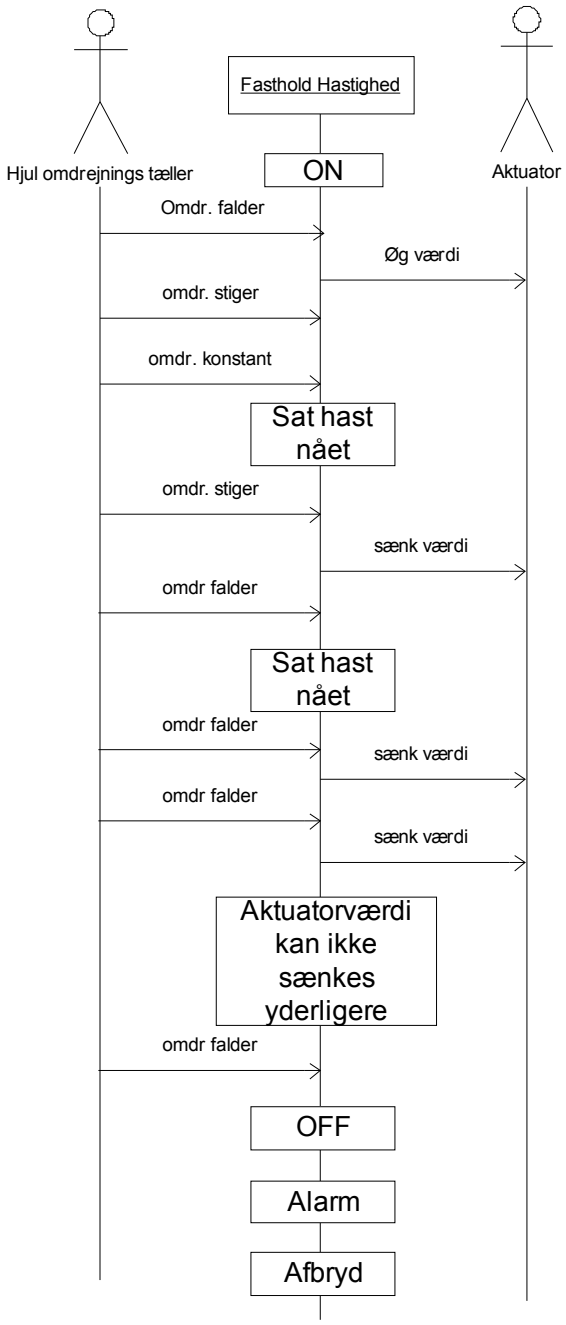
Use Case Fasthold Hastighed:



Figur Sekvensdiagram, som illustrerer kommunikationen mellem Use Case Fasthold Hastighed og aktørerne Gear og Chauffør.

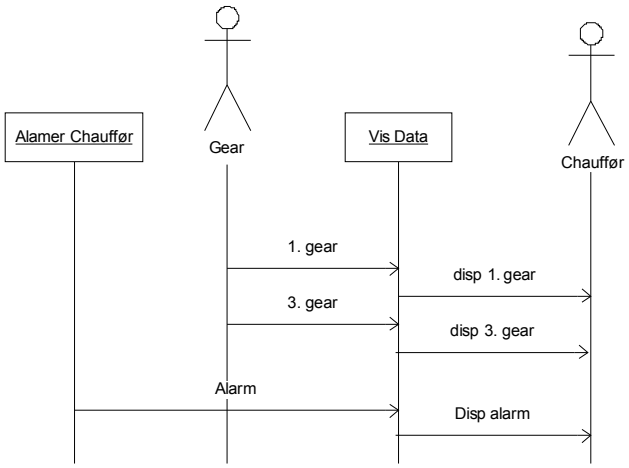


Figur Sekvens diagram, som illustrerer kommunikationen mellem Use Case Fasthold Hastighed og aktørerne Hjulomdrejningstæller og Aktuator.



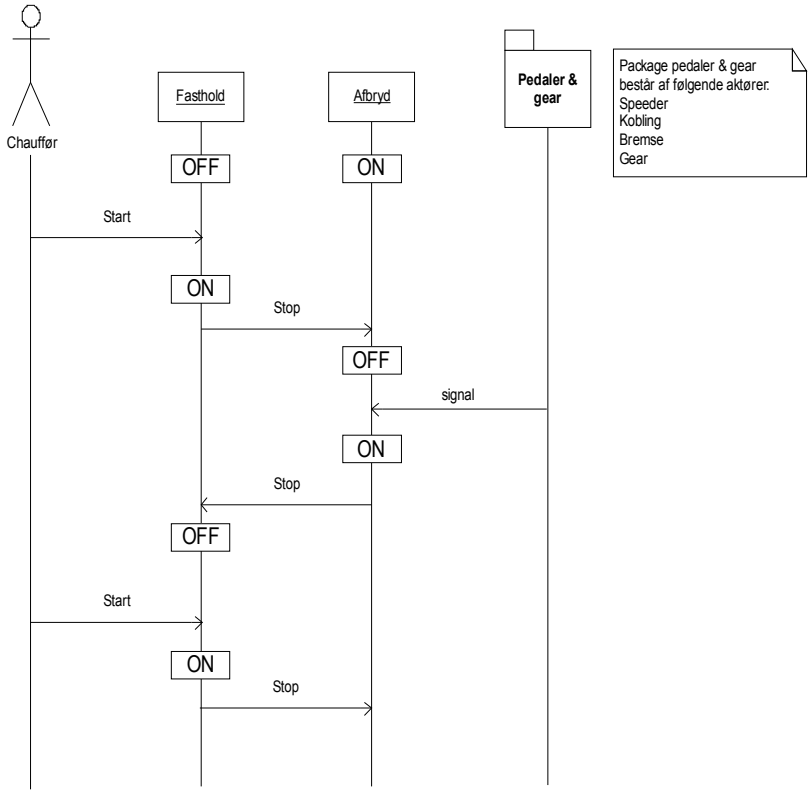
Figur Sekvens diagram, som illustrerer kommunikationen mellem Use Case Fasthold Hastighed og aktørerne Chauffør og Aktuator.

Use Case Vis Data:

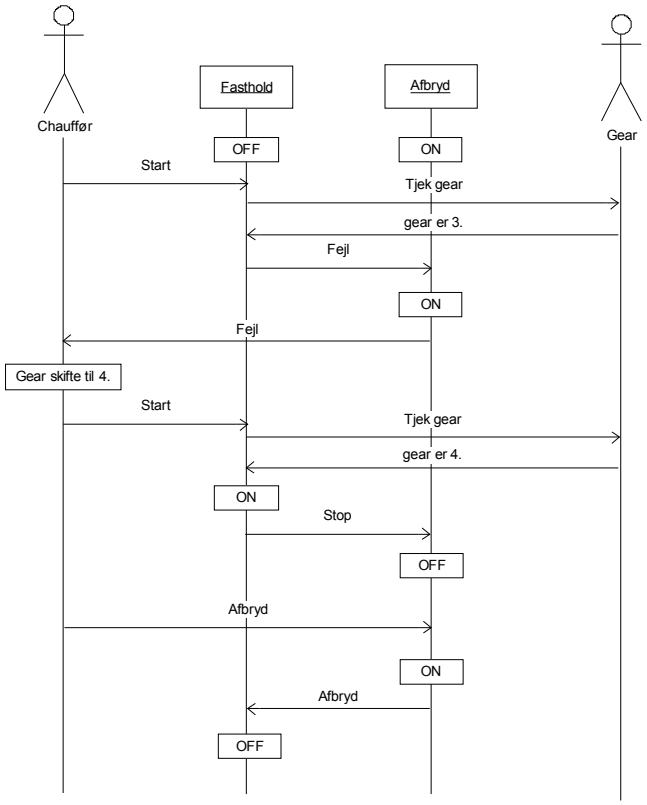


Figur Sekvens diagram, som illustrerer kommunikationen mellem Use Case Vis Data og Alamer Chauffør og aktørerne Gear og Chauffør.

Use Case Afbryd Hastighed:

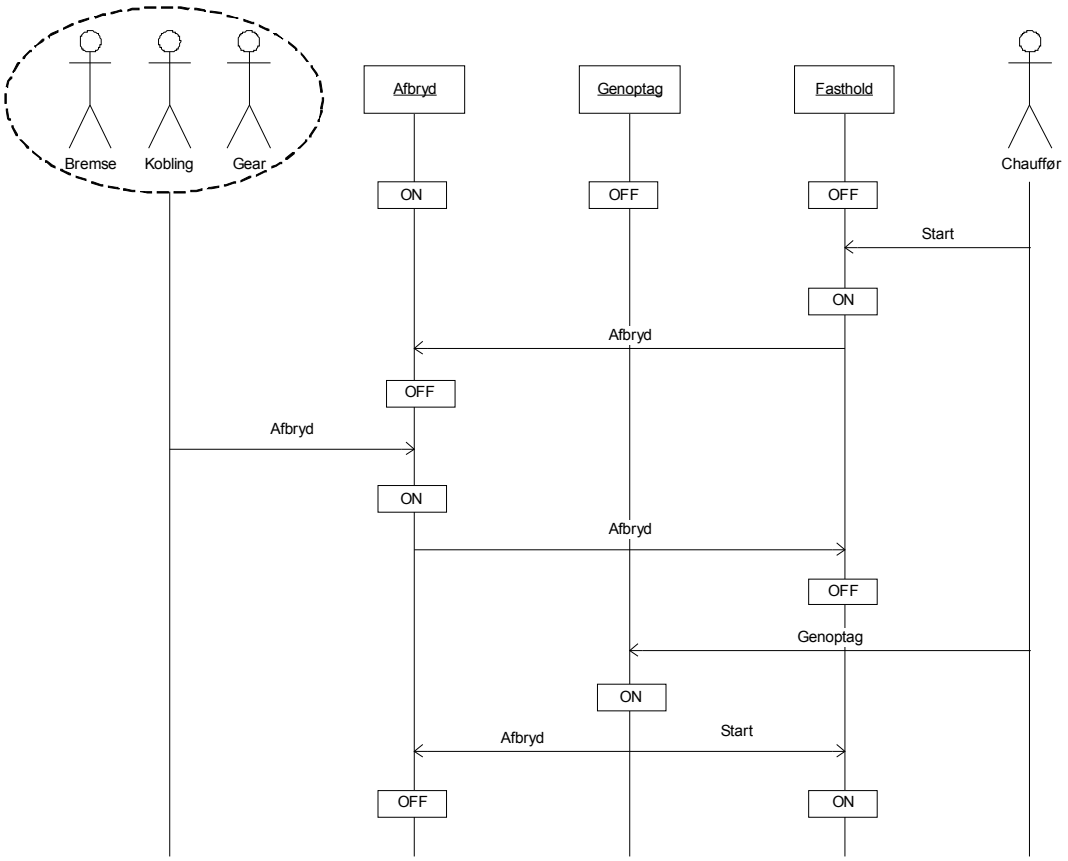


Figur Sekvensdiagram der illustrerer kommunikationen mellem Use Case Afbryd Hastighed og Fasthold Hastighed. Og aktører Chauffør, Speeder, Kobling, Bremse og Gear.

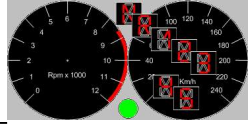


Figur Sekvensdiagram, som viser kommunikationen mellem Use Casene Afbryd Hastighed og Fasthold Hastighed. og aktørerne Chauffør og Gear,

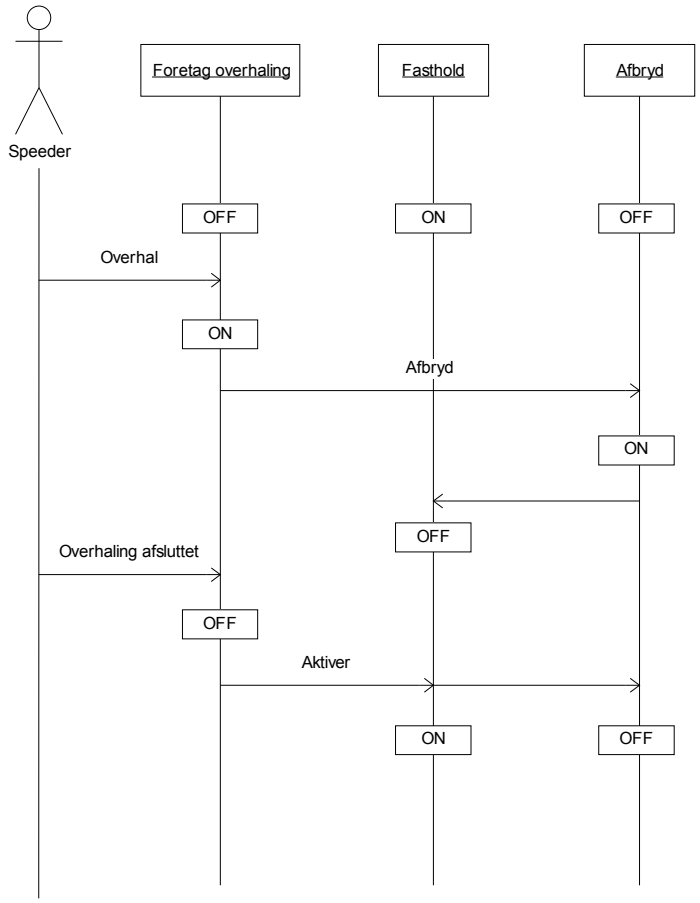
Use Case GenoptagHastighed:



Figur Sekvensdiagram der illustrerer kommunikationen mellem Use Cases Genoptag Hastighed, Fasthold Hastighed,, Afbryd Hastighed og aktørerne Chauffør og Bremse, Kobling og Gear

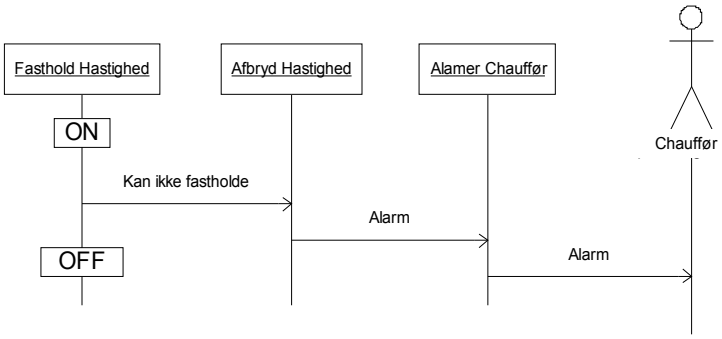


Use Case Foretag Overhaling:

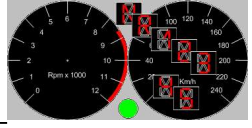


Sekvensdiagram der illustrerer kommunikationen mellem Use Cases Foretag Overhaling, Fasthold Hastighed og Afbryd Hastighed og aktør Speeder.

Use Case Alamer Chauffør:



Sekvensdiagram der illustrerer kommunikationen mellem Use Case Alamer Chauffør, Fasthold Hastighed og Afbryd Hastighed og aktør Chauffør.



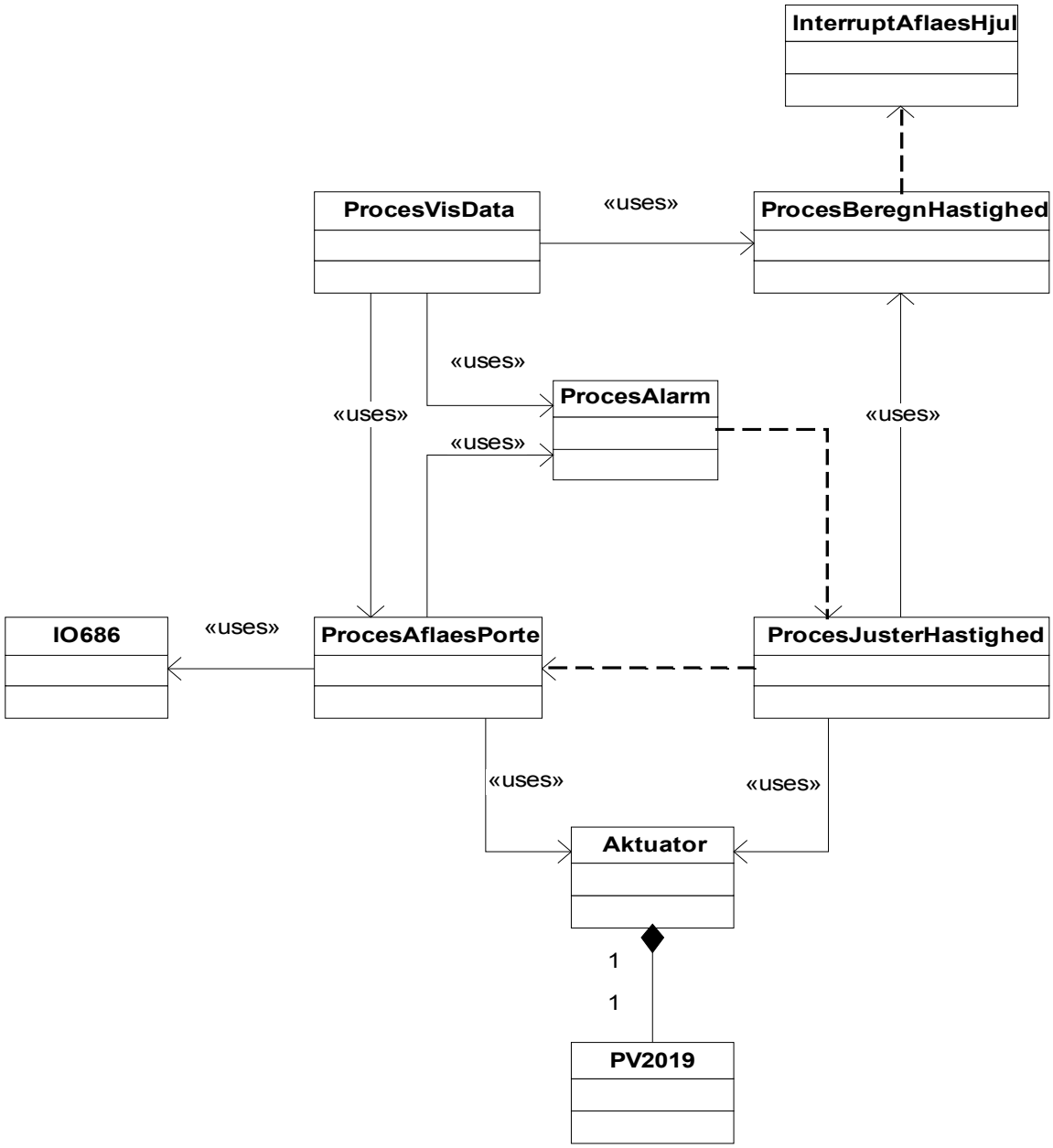
7. Gruppering i klasser

Alle objekter bliver realiseret i hver sin klasse. Dette giver ophav til følgende klassediagram. Alle proces klasser arver public fra klassen Thread, som er udeladt i diagrammet for overskuelighedens skyld.

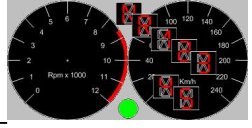
Det skal bemærkes, at klassediagrammet ikke indeholdes egenskaber og attributter, da disse er illustreret ved beskrivelsen af de enkelte klasser.

Under identifikation af klasser overvejede vi, at de data, som stammer fra procesklasserne, skulle indkapsles i nogle små klasser med en get og set funktion. På denne måde kunne vi beskytte vores run funktion i procesklasserne, som kun må kaldes en gang! Resultatet blev dog, at data indkapsles der hvor de er beregnet, dvs. i de proces klasser hvor de ”hører hjemme”.

Klassediagram:



Figur. Komplet klassediagram hvor relationerne mellem klasserne framgår.



I det følgende beskrives de enkelte klasser med ansvarsområde, funktionalitet, tilstands diagrammer og klassediagrammer.

Klasse ProcesBeregnHastighed

Klassediagram

ProcesBeregnHastighed
-semAflaesHjul : RTKSemaphore* -semJusterHastighed : RTKSemaphore* -semTriptaeller : RTKSemaphore -hjulomkreds : unsigned int -hastighed : unsigned int -triptaeller : unsigned long -afstandsmaaler : unsigned long
+ProcesBeregnHastighed(in sAflaesHjul : RTKSemaphore*, in sNyHastighed : RTKSemaphore*) +run() : void +getHastighed() : unsigned int +setTripTaeller(in nyAfstand : unsigned long) : void +getTriptaeller() : unsigned long -readLog() : void -writeLog(in nyHjulomkreds : unsigned int, in nyTriptaeller : unsigned long, in nyAfstandsmaaler : unsigned long) : void

Ansvar

Klassen skal udregne den aktuelle hastighed, opdatere en afstandsmåler og holde styr på en triptæller.

Afhængighed

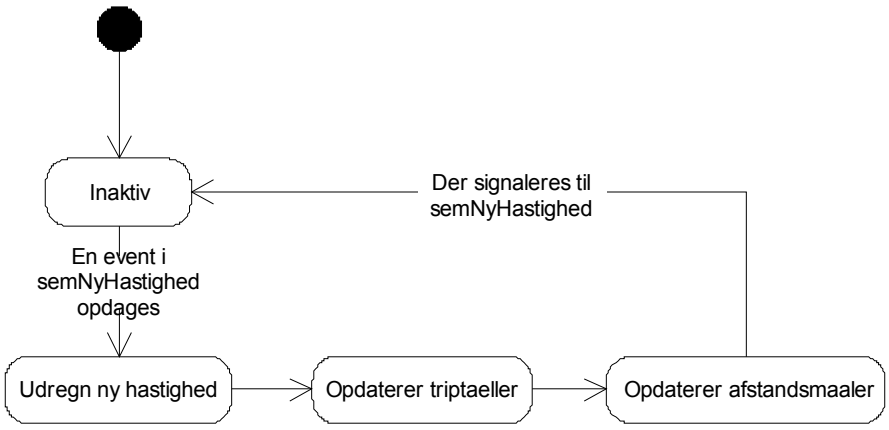
Ingen.

Funktionalitet

Hver eneste gang at køretøjets hjul har bevæget sig en halv omdrejning vil en interrupthandler signalerer til en semafor. Med kendskab til denne semafor og hjulomkredsen kan hastigheden udregnes.

Når signalet fra interrupthandleren opdages skal afstandsmåleren og triptælleren også opdateres. Dette kan gøres ved at addere halvdelen af hjulomkredsen til begge data.

Tilstandsdiagram



Klasse ProcesAlarm

Klassediagram

ProcesAlarm
-semAlarm : RTKSemaphore* -alarmAktiv : bool
+ProcesAlarm(in sAlarm : RTKSemaphore*) +~ProcesAlarm() +getAlarmAktiv() : bool +run() : void

Ansvar

Klassen skal alarmerer chaufføren med en hørbar indikation og give andre klasser mulighed for at vide om der er alarmeres.

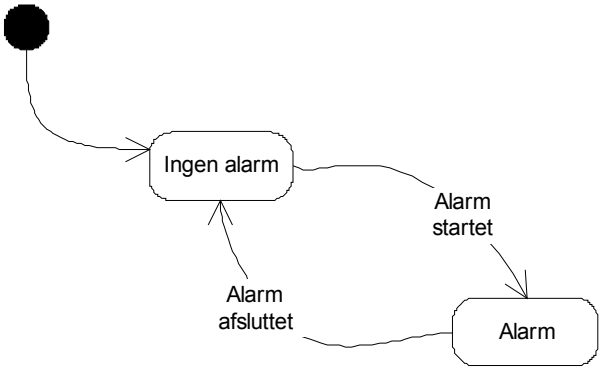
Afhængighed

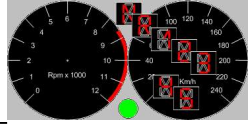
Ingen.

Funktionalitet

Hver gang der skal alarmeres, bliver der signaleret til en semafor, som ProcesAlarm kender til. Når sådan en event i denne semafor opdages, skal en hørbar indikation aktiveres og en privat variablen sættes til true. Efter en given forsinkelse skal den hørbar indikation deaktiveres og variablen skal sættes til false.

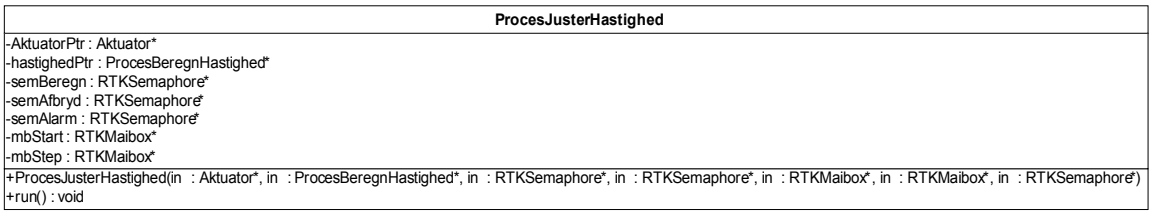
Tilstandsdiagram





Klasse ProcesJusterHastighed

Klassediagram



Ansvar

Klassen skal fastholde en given hastighed for køretøjet, uanset hvilket terræn det bevæger sig i.

Afhængighed

Aktuator.

ProcesBeregnHastighed.

Funktionalitet

Hver gang chaufføren ønsker at cruise, vil der blive sendt en besked til en mailbox, som ProcesJusterHastighed kender til.

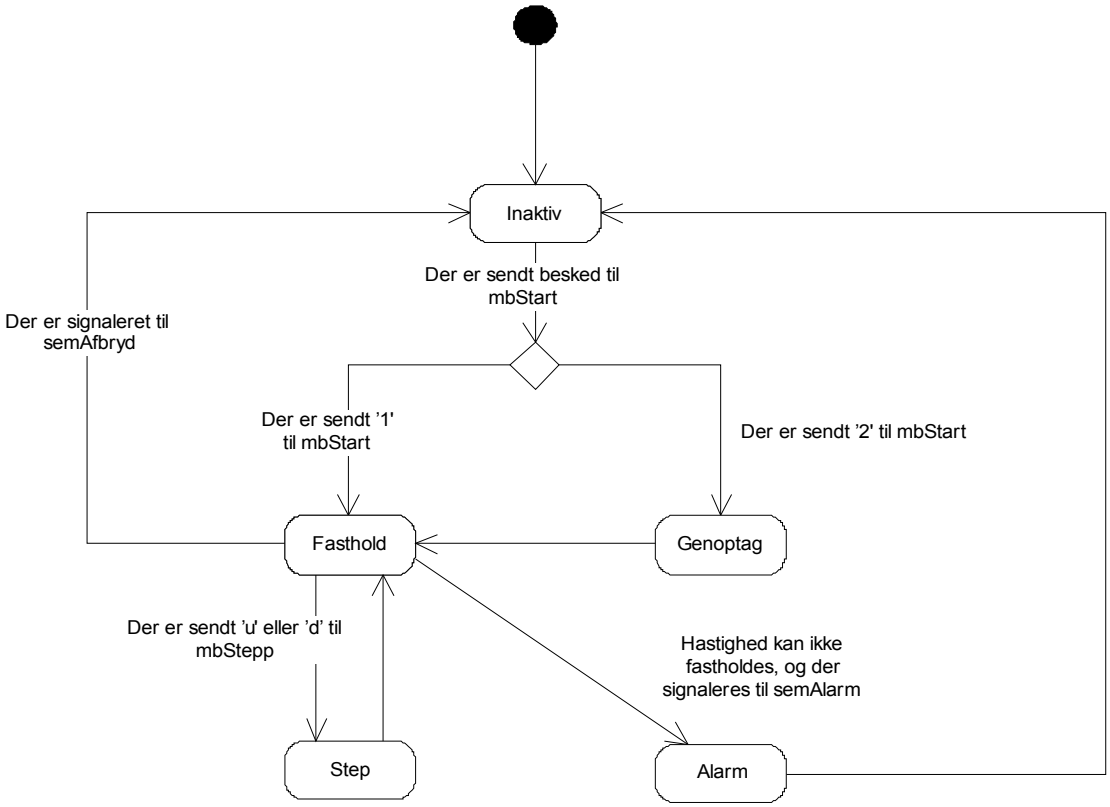
Når en besked i denne mailbox opdages skal der først undersøges om der skal cruises med den aktuelle hastighed eller om der cruises med forrige cruisehastighed.

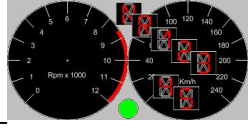
Efter cruisehastigheden er fundet skal der ventes på en event fra en semafor som ProcesBeregnHastighed signalerer til. Når denne event opdages skal den aktuelle hastighed hentes fra ProcesBeregnHastighed.

Herefter skal den aktuelle hastighed øges eller sinkes for at opnå den ønskede cruisehastighed.

Efter den ønskede cruisehastighed er opnået, skal der hele tiden undersøges om den aktuelle hastighed svarer til den ønskede cruisehastighed. Hvis dette ikke er tilfældet, skal den aktuelle hastighed justeres.
Når den ønskede cruisehastighed er opnået for første gang, skal der herefter undersøges om den aktuelle hastighed fraviger med mere en +/-5 Km/t. Hvis dette er tilfældet skal der signaleres til en semafor, og cruising skal afbrydes.
ProcesJusterHastighed kender også til en anden mailbox. Fra denne skal der modtages beskeder, når chaufføren interaktivt ønsker at hæve eller sinke cruisehastigheden.
Når der opdages en ny besked skal cruisehastigheden derfor hæves eller sinkes afhængigt af indholdet af beskeden.
ProcesJusterHastighed skal også undersøge en 3. semafor. Når der opdages en event i denne, skal cruising afbrydes.

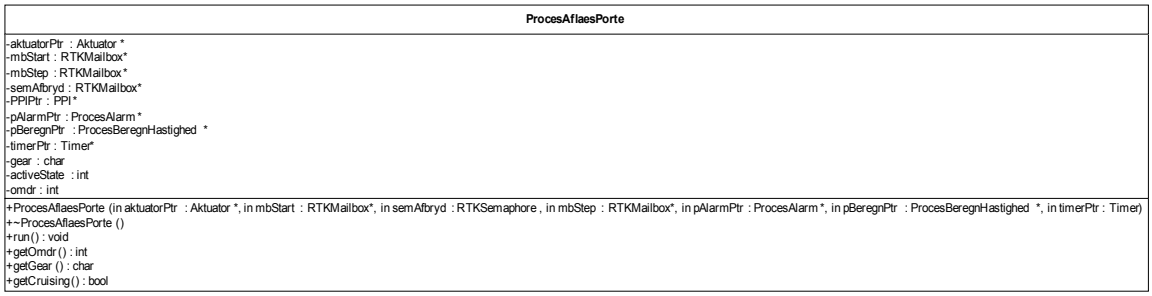
Tilstandsdiagram





Klasse ProcesAflaesPorte

Klassediagram



Ansvar

Klassen skal holde styr på chaufførens aktivitet.

Afhængighed

ProcesAlarm

Aktuator

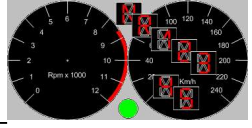
PPI

Funktionalitet

Klassen aflæser port A og B på SBC686's indsættelseskort.

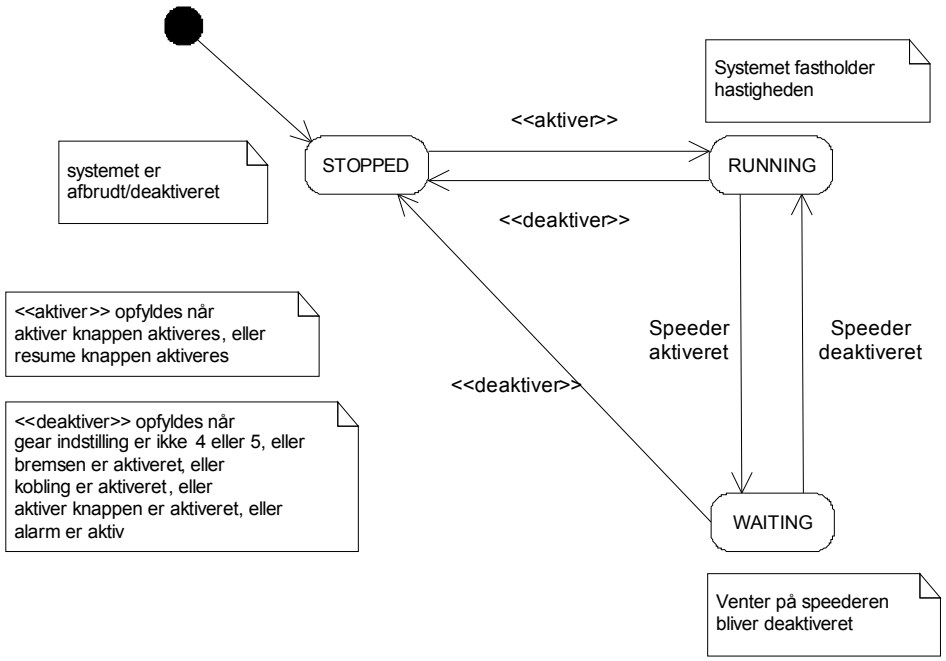
På port A skal det aflæses om chaufføren trykker på bremse, kobling og hvilket geartrin køretøjet befinder sig i.

På port B skal det aflæses om chaufføren trykker på Start, Genoptag eller Afbryd. Det skal også aflæses om chaufføren ønsker at hæve eller sinke den ønskede crusinghastighed.



Når chaufføren trykker på bremse imens der cruises, skal der signaleres til en semafor. Ellers skal et tryk på bremsen ignoreres.
Når chaufføren trykker på kobling imens der cruises, skal der signaleres til en semafor. Ellers skal et tryk på kobling ignoreres.
Når chaufføren trykker på Start, så skal det først undersøges om køretøjet befinder sig i 4. eller 5. geartrin og derefter om der cruises. Hvis køretøjet befinder sig i et af disse geartrin og der ikke cruises i forvejen, så skal der sendes en besked til en mailbox. Ellers skal et tryk på Start ignoreres.
Når chaufføren trykker på Genoptag, så skal det først undersøges om køretøjet befinder sig i 4. eller 5. geartrin og derefter om der cruises. Hvis køretøjet befinder sig i et af disse geartrin og der ikke cruises i forvejen, så skal der sendes en besked til en mailbox. Ellers skal et tryk på Genoptag ignoreres.
Når chaufføren trykker på Afbryd, så skal det undersøges om der cruises. Hvis der cruises, så skal der sendes et signal til en semafor. Ellers skal et tryk på Afbryd ignoreres.
Hvis chaufføren skifter geartrin imens der cruises, så skal der sendes et signal til en semafor.
Hvis chaufføren hæver eller sinker cruisinghastigheden imens der cruises, så skal der sendes en besked til en mailbox.

Tilstandsdiagram



Klasse Aktuator

Klassediagram

Aktuator
-sem : RTKSemaphore -pv2019 : PV2019 -pv2019dac : PV2019_DAC_V -pv2019adc : PV2019_ADC_V -sidsteVaerdi : unsigned int
+set(in aktuatorVaerdi : unsigned int) +get() : unsigned int

Ansvar

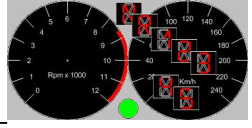
Driver til SBC686's indsættelseskort.

Afhængighed

PV2019.

PV2019_ADC_V.

PV2019_DAC_V.

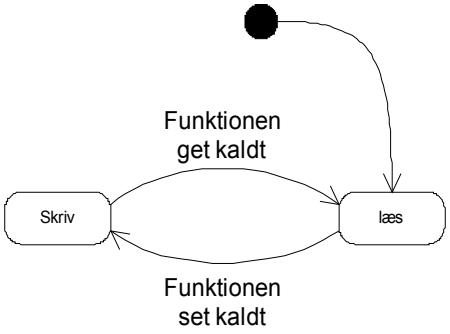


(Forfatteren til disse klasser er Micro Technic A-S og de er udleveret af skolen, og vil derfor ikke blive beskrevet nærmere.)

Funktionalitet

Klassen giver adgang til A/D og D/A konverteren på SBC686's indsættelseskort, så der henholdsvis kan læses fra og skrives til dem.

Tilstandsdiagram



Klasse InterruptAflaesHjul

Klassediagram

InterruptAflaesHjul
-semInterrupt : RTKSemaphore* -IRQ : unsigned char
+InterruptAflaesHjul(in nyIRQ : unsigned char, in sInterrupt : RTKSemaphore*) +~InterruptAflaesHjul() +handleInt() : static void RTKAPI

Ansvar

Hver gang køretøjets hjul har drejet en halv omdrejning vil et bestemt interrupt forekomme, og skal klassen skal derefter sende et signal til en semafor.

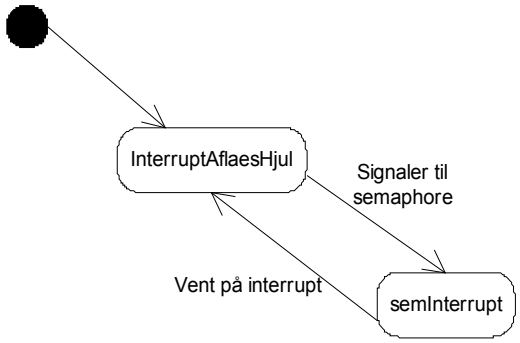
Afhængighed

Ingen.

Funktionalitet

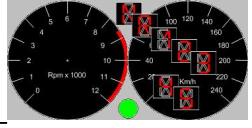
Når køretøjets hjul har drejet en halv omdrejning, vil et interrupt på IRQ5 forekomme. Klassen skal være en interrupthandler, der sender et signal til en semafor, hver gang dette interrupt opdages.

Tilstandsdiagram



Klasse ProcesVisData

Klassediagram



ProcesVisData
-pAlarm : ProcesAlarm* -pPorte : ProcesAflaesPorte* -pBeregn : ProcesBeregnHastighed* -omdr : PegFiniteBitmapDial* -speed : PegFiniteBitmapDial* -back : PegBitmapLight* -alarm : PegBitmapLight* -gear : PegBitmapLight* -cruise : PegBitmapLight* -presentPtr : PegPresentationManager* -kmTekstPtr : PegPrompt* -tripTekstPtr : PegPrompt* -speedometerPtr : PegPrompt*
+ProcesVisData(in pAlarm : ProcesAlarm*, in pPorte : ProcesAflaesPorte*, in pBeregn : ProcesBeregnHastighed*) +run() : void -updateSpeed() : void -updateRPM() : void -updateGear() : void -updateAlarm() : void -updateCruise() : void

Ansvar

Klassen skal oprette et display hvor chaufføren kan aflæse diverse data.

Afhængighed

RTPeg-biblioteket

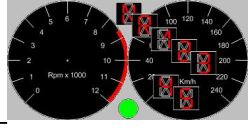
ProcesAflaesPorte

ProcesAlarm

ProcesBeregnHastighed

Funktionalitet

Ved hjælp af RTPeg-biblioteket skal der oprettes et grafisk display. På displayet skal chaufføren kunne aflæse hastigheden, triptæller, afstandsmåler, omdrejningstæller for motor, det aktuelle geartrin som køretøjet befinder sig i og om der alarmeres eller cruises.
Det ovennævnte data skal opdateres minimum 2 gange i sekundet på displayet.



Tilstandsdiagram

Der er ikke tegnet tilstands diagram for denne klasse. Man kan sige, at klassens tilstand er konstant ”vis data”.