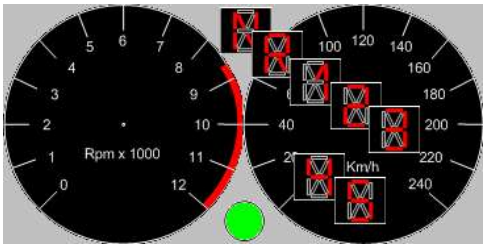
Projekt: Cruisecontrol

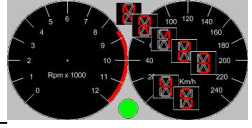
Dato: 18-12-2003

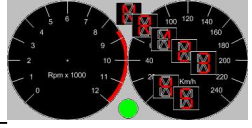
Titel:

Modultest
for
Cruisecontroller



Cruise International

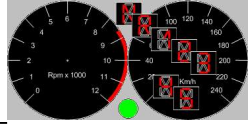




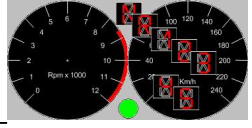
Dette dokument indeholder modultest af de funktioner, som indgår i de enkelte klasser.

1 ProcesJusterHastighed

Funktionsnavn	***1.1 ProcesJusterHastighed***
Parametre	Aktuator *AktuatorPtr, ProcesBeregnHastighed *hastighedPtr, RTKSemaphore *semBeregn, RTKSemaphore *semAfbryd, RTKMailbox *mbStart, RTKMailbox *mbstep, RTKSemaphore *semAlarm
Returværdi	Ingen.
Testforløb	Da constructoren altid bliver kaldt ved oprettelse af et objekt kan det testes, at de i klassen lokale parametre bliver sat ved at bruge debuggeren.
Forventet Resultat	Der forventes at de i klassen indeholdte pointerene får samme adresser som de pointere, der gives med i constructoren. Det forventes desuden, at GammelHastighed, SidsteHastighed og alarmTaeller får værdien 0.
Resultat	Det ses, at de i klassen indeholdte pointerene får de adresser der gives med, og at GammelHastighed, SidsteHastighed og alarmTaeller får værdien 0.
Konklusion	Funktionen er godkendt.



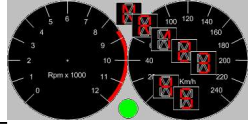
Funktionsnavn	***1.2 run***
Parametre	Ingen.
Returværdi	Ingen.
Testforløb	Der testes om: 1. Funktionen venter på at der kommer besked i mbStart ved at udskrive den værdi `AktuatorPtr->get()` returnerer. Dette gøres ved at tilføje følgende linie i koden efter `AktuatorPtr->get(): cout << AktuatorVaerdi;` 2. Når mbStart modtager '1' at satHastighed får værdien `hastighedPtr->getHastighed()` og at gammelHastighed får værdien satHastighed. Dette ses ved anvendelse af debugger hvor der steppen gennem koden. 3. Når mbStart modtager '2' at satHastighed får værdien gammelHastighed. Dette ses ved anvendelse af debugger hvor der steppen gennem koden. 4. Ved at betragte skærmen under kørsel af hele cruiseprogrammet og ved at tilføje: `std::cout << AktuatorVaerdi << " " << hastighedPtr->getHastighed() << "km > " << Sat Hastighed << " km" << std::endl;` til koden kan man hele tiden se, sammenhængen mellem Den hastighed, som man ønsker at holde (satHastighed), den aktuelle hastighed og aktuatorværdien. For at kunne godkende fastholdelsen skal: • Aktuatorværdien stige når den aktuelle hastighed er mindre end satHastighed. • Aktuatorværdien falde når den aktuelle hastighed er større end satHastighed • Aktuatorværdien forblive den samme, når den aktuelle hastighed er den samme som satHastighed.
Forventet Resultat	Der forventes at: 1. Funktionen venter på modtagelse af besked, og derefter udskriver det modtagne. 2. Når mbStart modtager '1' får satHastighed værdien `hastighedPtr->getHastighed()` og at gammelHastighed får værdien satHastighed. 3. Når mbStart modtager '2' får satHastighed værdien gammelHastighed. 4. Når der køres: • Aktuatorværdien stiger når den aktuelle hastighed er mindre end satHastighed. • Aktuatorværdien falder når den aktuelle hastighed er større end satHastighed • Aktuatorværdien forbliver den samme, når den aktuelle hastighed er den samme som satHastighed.



Resultat	1. Det ses, at der kun udskrives på skærmen., når der er modtaget besked fra mbStart. => OK 2. Det ses, at satHastighed får værdien `hastighedPtr->getHastighed()` og at gammelHastighed får værdien satHastighed. => OK 3. Det ses, at når mbStart modtager '2' får satHastighed værdien gammelHastighed. => OK 4. Det ses ved aflæsning af skærmen, at: • Aktuatorværdien stiger når den aktuelle hastighed er mindre end satHastighed. • Aktuatorværdien falder når den aktuelle hastighed er større end satHastighed • Aktuatorværdien forbliver den samme, når den aktuelle hastighed er den samme som satHastighed. DVS: => OK 5.
Konklusion	Funktionen er godkendt.

2 ProcesAflaesPorte

Funktionsnavn	***2.1 ProcesAflaesPorte***
Parametre	Aktuator *aktuatorPtr RTKMailbox *mbStart RTKSemaphore *semAfbryd RTKMailbox *mbStep ProcesAlarm *pAlarmPtr ProcesBeregnHastighed *pBeregnPtr Timer *timerPtr
Returværdi	Ingen

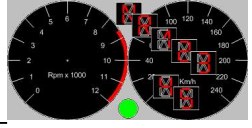


Testforløb	Funktionen undersøges ved at anvende stepfunktionerne i Visual Studio. På denne måde kan man visuelt aflæse værdien af alle ønskede data. 1. Funktionen bliver kaldt med adresser på objekter til de respektive værdier. Fx så aktuatorPtr får værdien til et Aktuator objekt. 2. De private medlemsdata bliver initieret til de værdier givet som parametre til funktionen. 3. De private medlemsdata som ikke bliver initieret vha. parametre, bliver initieret til 0. 4. Funktions kaldet er som følger: ProcesAflaesPorte pAP(aktuator, &mbStart, &semAfbryd, &mbStep, &pAL, &pBH, timerPtr); Derudover foretages der nogle RTK-systemkald, som ikke testes da de formodes at virke. Det til funktionen givne parametre antages at virke korrekt.
Forventet Resultat	Der forventes at: 1. Medlemsvariablene bliver initieret.
Resultat	Det observerede resultat: 1. Medlemsvariablene blev initieret.
Konklusion	Funktionen er godkendt.

De øvrige funktioner i klassen, anvender kun RTK-systemkald, så de testes ikke.

Funktionsnavn	***2.2 getCruising***
Parametre	Ingen
Returværdi	Bool
Testforløb	Funktionen undersøges ved at anvende stepfunktionerne i Visual Studio. På denne måde kan man visuelt aflæse værdien af alle ønskede data. 1. activeState sættes til true i konstructoren 2. activeState sættes til false.
Forventet Resultat	Der forventes at: 1. returværdien er true. 2. returværdien er false
Resultat	Det observerede resultat: 1. returværdien er true 2. returværdien er false.
Konklusion	Funktionen er godkendt.

Funktionsnavn	***2.3 getGear***
Parametre	Ingen
Returværdi	Bool
Testforløb	Funktionen undersøges ved at anvende stepfunktionerne i Visual Studio. På denne måde kan man visuelt aflæse værdien af alle ønskede data. 1. gear sættes til -1 i konstructoren 2. gear sættes til 3.

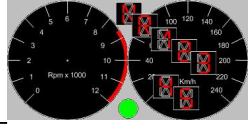


Forventet Resultat	Der forventes at: 1. returværdien er -1. 2. returværdien er 3
Resultat	Det observerede resultat: 1. returværdien er -1 2. returværdien er 3.
Konklusion	Funktionen er godkendt.

Funktionsnavn	***2.4 setOmdr***
Parametre	Ingen
Returværdi	Ingen
Testforløb	Funktionen undersøges ved at anvende stepfunktionerne i Visual Studio. På denne måde kan man visuelt aflæse værdien af alle ønskede data, dog springes selve udregningen over da den er tidsafhængig. 1. gear sættes til -1 i konstructoren. 2. og senere sættes den til 3.
Forventet Resultat	Der forventes at: 1. Returværdien er -1. 2. returværdien er 3
Resultat	Det observerede resultat: 1. returværdien er -1 2. returværdien er 3.
Konklusion	Funktionen er godkendt.

Funktionsnavn	getOmdr
Parametre	Ingen
Returværdi	Int
Testforløb	Funktionen undersøges ved at anvende stepfunktionerne i Visual Studio. På denne måde kan man visuelt aflæse værdien af alle ønskede data. 1. omdr sættes til 0 i konstructoren. 2. og senere sættes den til 3000.
Forventet Resultat	Der forventes at: 1. Returværdien er 0. 2. returværdien er 3000
Resultat	Det observerede resultat: 1. returværdien er 0 2. returværdien er 3000.
Konklusion	Funktionen er godkendt.

Funktionsnavn	Run
Parametre	Ingen
Returværdi	Ingen

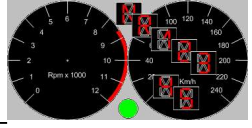


Testforløb	Funktionen undersøges ved at anvende stepfunktionerne i Visual Studio. På denne måde kan man visuelt aflæse værdien af alle ønskede data, dog springes selve udregningerne over da de er tidsafhængige. 1. o
Forventet Resultat	Der forventes at: 1. o
Resultat	Det observerede resultat: 1. o
Konklusion	Funktionen er godkendt.

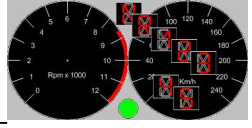
3 ProcesAlarm

Funktionsnavn	***3.1 ProcesAlarm***
Parametre	RTKSemaphore *sAlarm
Returværdi	Ingen
Testforløb	Funktionen undersøges ved at anvende stepfunktionerne i Visual Studio. På denne måde kan man visuelt aflæse værdien af alle ønskede data. 1. Det undersøges at alarmAktiv sættes til false 2. Det undersøges at semAlarm får samme adresse som sAlarm
Forventet Resultat	Der forventes at: 1. alarmAktiv = false 2. adressen på semAlarm = adressen på sAlarm
Resultat	Det observerede resultat: 1. alarmAktiv = false 2. adressen på semAlarm = adressen på sAlarm
Konklusion	Funktionen er godkendt.

Funktionsnavn	***3.2 getAlarmAktiv***
Parametre	Ingen
Returværdi	Bool
Testforløb	Funktionen undersøges ved at anvende stepfunktionerne i Visual Studio. På denne måde kan man visuelt aflæse værdien af alle ønskede data. Det undersøges om returværdien er den samme som værdien af getAlarmAktiv
Forventet Resultat	Der forventes at returværdien bliver false
Resultat	Det observerede resultat: Returnvalue = false
Konklusion	Funktionen er godkendt.



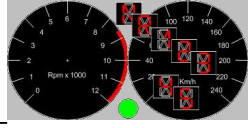
Funktionsnavn	***3.3 run***
Parametre	Ingen
Returværdi	Ingen
Testforløb	Processen undersøges på en SBC686, hvor der manuelt signaleres til processens RTKWait kald. Dette gøres for at aktivere processen, der ellers kun kører, hvis det ikke kan lade sig gøre at fastholde cruisehastigheden.
Forventet Resultat	Der forventes at der kommer en hørbar alarm i 5 sekunder
Resultat	Det observerede resultat: Der kom en alarm i 5 sekunder
Konklusion	Funktionen er godkendt.



4 ProcesBeregnHastighed

Funktionsnavn	***4.1 ProcesBeregnHastighed***
Parametre	RTKSemaphore *sAflaesHjul RTKSemaphore *sNyHastighed
Returværdi	Ingen
Testforløb	Funktionen undersøges ved at anvende stepfunktionerne i Visual Studio. På denne måde kan man visuelt aflæse værdien af alle ønskede data. 5. Funktionen bliver kaldt med adresser på objekter til de respektive værdier. Fx så semAflaesHjul får værdien til en semafor. 6. De private medlemsdata bliver initieret til de værdier givet som parametre til funktionen. 7. De private medlemsdata som ikke bliver initieret vha. parametre, bliver initieret til 0 eller den ønskede type. 8. Funktions kaldet er som følger: ProcesBeregnHastighed(RTKSemaphore *sAflaesHjul, RTKSemaphore *sNyHastighed) Derudover foretages der nogle RTK-systemkald, som ikke testes da de formodes at virke.
Forventet Resultat	Der forventes at: 2. Medlemsvariablene bliver initieret.
Resultat	Det observerede resultat: 2. Medlemsvariablene blev initieret.
Konklusion	Funktionen er godkendt.

Funktionsnavn	***4.2 run***
Parametre	Ingen
Returværdi	Void
Testforløb	Funktionen undersøges ved at anvende stepfunktionerne i Visual Studio. På denne måde kan man visuelt aflæse værdien af alle ønskede data. 3. tid1, tid2, tid3 sættes til den aktuelle tid. 4. Funktionen venter på at der er forløbet 2000 systemtidsenheder eller en event forekommer i semAflaesHjul. 5. Hvis 2000 systemtidsenheder er forløbet, bliver hastighed sat lig 0. 6. Hvis en event forekommer i semAflaesHjul, så skal en ny hastighed udregnes og medlemsdata afstandsmaaler og triptaeller bliver opdateres med en halv hjulomkreds. writeLog(…) bliver kaldt.
Forventet Resultat	Der forventes at: 3. hastighed bliver sat lig 0, hvis 2000 systemtidsenheder er forløbet 4. en ny hastighed udregnes og medlemsdata bliver opdateret. Og writeLog(…) bliver kaldt.

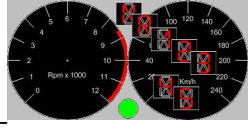


Resultat	Det observerede resultat: 3. hastighed blev sat lig 0. 4. Diverse data blev opdateret og writeLog(…) blev kaldt.
Konklusion	Funktionen er godkendt.

Funktionsnavn	***4.3 getHastighed***
Parametre	Ingen
Returværdi	unsigned int
Testforløb	Funktionen undersøges ved at anvende stepfunktionerne i Visual Studio. På denne måde kan man visuelt aflæse værdien af alle ønskede data. 3. Medlemsdata hastighed's værdi skal returneres.
Forventet Resultat	Der forventes at: 3. Medlemsdata hastighed's værdi bliver returneret.
Resultat	Det observerede resultat: 3. Medlemsdata hastighed's værdi blev returneret.
Konklusion	Funktionen er godkendt.

Funktionsnavn	***4.4 setTriptaeller***
Parametre	unsigned long ny
Returværdi	Ingen
Testforløb	Funktionen undersøges ved at anvende stepfunktionerne i Visual Studio. På denne måde kan man visuelt aflæse værdien af alle ønskede data, dog springes selve udregningen over da den er tidsafhængig. 3. Medlemsdata triptaeller skal sættes lig med parameter.
Forventet Resultat	Der forventes at: 3. Medlemsdata triptaeller bliver sat lig med parameter.
Resultat	Det observerede resultat: 3. Medlemsdata triptaeller blev sat lig med parameter.
Konklusion	Funktionen er godkendt.

Funktionsnavn	***4.5 getTriptaeller***
Parametre	Ingen
Returværdi	unsigned long
Testforløb	Funktionen undersøges ved at anvende stepfunktionerne i Visual Studio. På denne måde kan man visuelt aflæse værdien af alle ønskede data. 3. Medlemsdata triptaeller's værdi skal returneres.
Forventet Resultat	Der forventes at: 3. Medlemsdata triptaeller's værdi bliver returneret.
Resultat	Det observerede resultat: 3. Medlemsdata triptaeller's værdi blev returneret.
Konklusion	Funktionen er godkendt.

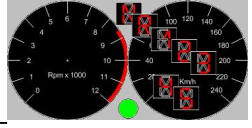


Funktionsnavn	***4.6 readLog***
Parametre	Ingen
Returværdi	Ingen
Testforløb	1. En fil skal læses og hjulomkreds opdateres.
Forventet Resultat	Der forventes at: 1. En fil bliver læst og hjulomkreds bliver opdateret.
Resultat	Det observerede resultat: 1. En fil blev læst og hjulomkreds blev opdateret.
Konklusion	Funktionen er godkendt.

5 ProcesVisData

Funktionsnavn	***5.1 ProcesVisData***
Parametre	ProcesAlarm *pAlarm ProcesAflaesPorte *pPorte ProcesBeregnHastighed *pBeregn
Returværdi	Ingen
Testforløb	Funktionen undersøges på en SBC686. Følgende oprettes og tegnes, og det kan ses på skærmen om det hele bliver tegnet: Displayets baggrund, omdrejningstæller, speedometer, kilometertæller, triptæller, alarmlampe, gearindikator og cruiselampe.
Forventet Resultat	Der forventes at displayets baggrund, omdrejningstæller, speedometer, kilometertæller, triptæller, alarmlampe, gearindikator og cruiselampe bliver tegnet.
Resultat	Det observerede resultat: Det hele blev vist på skærmen.
Konklusion	Funktionen er godkendt.

Funktionsnavn	***5.2 Run***
Parametre	Ingen
Returværdi	Ingen
Testforløb	Funktionen undersøges ved hjælp af simulator-cyklen. Følgende update funktioner kaldes med et fast interval: updateSpeed(), updateRPM(), updateGear(), updateAlarm(), updateCruise(). På skærmen holdes øje med om de forskellige elementer i displayet bliver opdateret løbende under kørslen.



Forventet Resultat	Der forventes at der efter simulationen har været ændringer på følgende, for at kunne konstatere at de bliver opdateret: Omdrejningstæller, speedometer, kilometertæller, triptæller, alarmlampe, gearindikator og cruiselampe
Resultat	Det observerede resultat: Det lykkedes at foretage ændringer på alle elementer i displayet, så det kunne ses at de blev opdateret.
Konklusion	Funktionen er godkendt.

Alle update funktioner har vist sig at virke under testen af run. Da det lykkedes at foretage ændringer i alle elementer i displayet, foretages der ikke yderligere test på disse.

6 InterruptAflaesHjul

Funktionsnavn	***6.1 InterruptAflaesHjul***
Parametre	unsigned char nyIRQ RTKSemaphore *sInterrupt
Returværdi	Ingen
Testforløb	Funktionen undersøges ved at anvende stepfunktionerne i Visual Studio. På denne måde kan man visuelt aflæse værdien af alle ønskede data. 9. Funktionen debugges hvor parametren nyIRQ får værdien 5. Den private variabel IRQ skal nu sættes lig med denne. 10. Funktionen debugges, og det undersøges om variablen semInterrupt får samme adresse som parametren sInterrupt. Derudover foretages der nogle RTK-systemkald, som ikke testes da de formodes at virke.
Forventet Resultat	Der forventes at: 3. IRQ = 5 4. semInterrupt får samme adresse som sInterrupt
Resultat	Det observerede resultat: 3. IRQ = 5 4. semInterrupt fik samme adresse som sInterrupt
Konklusion	Funktionen er godkendt.

De øvrige funktioner i klassen, anvender kun RTK-systemkald, så de testes ikke.