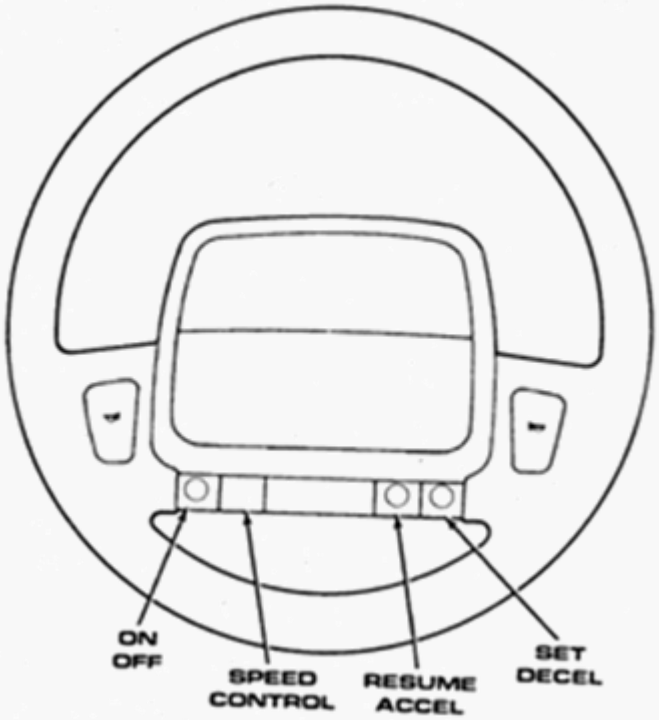
Cruise Control 1993 Jeep Cherokee

Owner's Manual System Description:

Cruise Control System Interface

When engaged, the electronic cruise control device takes over the accelerator operations of the vehicle at speeds above 35M/H, (60KM/H). The cruise controls are located on the steering wheel and consist of the following buttons: ON/OFF, RESUME/ACCEL and SET/DECEL.



To Activate – Push the ON/OFF switch. When depressed the cruise control system is activated. Push the ON/OFF switch again to raise the button and deactivate the cruise control system.

To Set a Desired Speed – When the vehicle has reached the desired cruising speed, press and release the SET button to engage the cruise control. Remove foot pressure from the accelerator pedal.

Cruise Control 93 Cherokee - manual

To Deactivate – Softly tap the brake or clutch pedal or stop normally. The cruise control will disengage, but will remember the last cruising speed. The previous cruising speed will be retained while the cruise control system is ON or until the ignition is turned off.

To Resume Speed – To resume the last retained cruising speed, push and release the RESUME button. Resume can be used at any speed above 30M/H, (50KM/H).

To Vary the Speed Setting – When the cruise control is engaged, speed can be increased by pressing and holding the ACCEL button. When the button is released, a new set speed will be established. Tapping the ACCEL button once will result in a 2 mph speed increase. Each time the ACCEL button is tapped speed increases, so tapping it three times will increase speed by 6 mph, etc. To decrease speed while the cruise control is engaged, press and hold the DECEL button. Release the button when the desired speed is reached, and the new speed will be set.

Cruise Control 93 Cherokee - manual

To Accelerate for Passing – Depress the accelerator foot pedal as you would normally. When the pedal is released the vehicle will return to the set speed.

Note – When driving uphill, or when heavily loaded the vehicle may slow below the set speed. If the vehicle speed drops below 30M/H, (50KM/H) the cruise control will automatically disengage.

Warning – leaving the cruise control ON when not in use is dangerous. You could accidentally set the system or cause it go faster than desired. Always leave the system OFF when you aren't using it.

Cruise Control 93 Cherokee - Specification

The automobile computer controller (ACC) system monitors all sensors. It communicates readings and signals the appropriate automobile sub-system.

Cruise Control Sub-System Functioning

The ON and OFF actions to the cruise control manager (CCM) sub-system will be initiated by the ACC system. The ACC will signal the CCM when any of the cruise control buttons have been pressed. Note: when buttons are held down the ACC will send a series of signals to the CCM, one after another until the button is released. The CCM must interact with the ACC and the fuel sub-system administrator, (FSA). The current automobile speed is monitored by the ACC. The CCM will send the driver's desired speed (in MPH) to the FSA, which regulates fuel flow in accordance with automobile speed. The FSA determines the amount of fuel to supply by getting the current speed from the ACC. If the current speed drops below 30MPH the FSA halts its fuel flow regulation. When the automobile operator depresses the accelerator pedal the ACC will inform the CCM to deactivate and when the operator releases the accelerator the ACC will inform the CCM to resume.

Cruise Control 93 Cherokee - problem

Task

Design the Cruise Control Manager (CCM) software. The design should consist of class relationships and class/method interface(s). The design contain description forms for the class and methods

Sample class description form:

Class Name	Complex
Purpose	represent a complex number and provide associated operations
States	none
Constructors	Complex(Real = 0.0, Imaginary = 0.0)
Operations	
Mutators	setReal, setImaginary, Magnitude, Conjugate, +, -, /, *, ==
Accessors	getReal, getImaginary
Fields	Real, Imaginary

Cruise Control 93 Cherokee - problem

Sample operation description form:

Prototype	Complex operator+(Complex rightOperand)
Purpose	to compute the sum of two Complex objects
Data received	right operand of binary expression, left operand is the execution scope of the operation
Data returned	Complex object representing the sum of the objects
Remarks	Overloaded addition operator is provided to support natural coding of expressions involving Complex objects.

CCM Design: data elements

Initial data element identification:

1. Existence state (ON/OFF)
2. Activation/engage state
3. TempDisengage state (see the PASSING/COASTING service discussion)
4. Desired speed of the driver
5. Association link to the fuel sub-system administrator, (FSA)
6. Association link to the ACC - ?

CCM Data Element Deliberations

Existence state could be omitted if the CCM object is created & destroyed by the automobile computer controller (ACC) system whenever the ON/OFF buttons are operated.

If the CCM only needs to exist while the cruise control is ON and no other part of the system except the ACC needs to request its services then the CCM should be aggregated within the ACC. (From the abbreviated specification this is most likely the case and an aggregation would be the better design.)

Thus the best design would be not to represent the ON/OFF state, instead electing to **dynamically aggregate** the CCM inside the ACC.

CCM Data Element Deliberations

Each of these data elements will correspond to one or more data members.

The desired speed of the driver could be a simple integer or a current/desired speed pair, either as separate CCM data members or encapsulated within a sub-object.

For this design, I opt to not have a distinct class or to represent the current speed in the CCM, (which is problematic since the speed will always be changing and would thus be inaccurate at most times).

CCM Responsibilities

It is clear that the CCM is responsible for storing/updating the driver's desired speed and communicating with the ACC and FSA.

Each cruise control interface button action and communication mechanism will have a corresponding public member function.

The specification indicates the ACC will signal the CCM when any cruise controls buttons are depressed. The three buttons are labeled as dual purpose thus an assignment of responsibility as to which CCM service is to be invoked when a button is depressed must be made.

One poor design possibility would be to assign this responsibility to the ACC. This requires the ACC to know (store) or determine (invoke a CCM state reported function) the state of the CCM. This responsibility should be assigned to the CCM since it already knows its own state.

CCM Design: Behaviors

CCM Responsibilities Design

ON: This state, (existence), is handled by the constructor.

OFF: This state, (non-existence), is handled by the destructor.

CCM Design: Behaviors

CCM Responsibilities Design: Set

SET:
This operation (invoked by the ACC) can be designed in one of two ways, (with either no or one parameter).

With no parameter the operation will query the ACC for the current operating speed and store it as the driver's desired speed.

Alternatively the ACC could go ahead and pass the current speed to the SET operation.

Either would be an appropriate design. (One could argue that since the ACC has responsibility for the current speed that it would be a better design to have it go ahead and send it to the CCM for this operation. However, it could also be argued that the CCM should request the current speed from the ACC in order to obtain the most accurate up-to-date speed.)

CCM Responsibilities Design: Set (continued)

The Set operation also has to communicate with the FSA, to invoke its automatic fuel regulating flow service, sending it the stored desired speed.

However the current speed is obtained, a valid alternative design would be to consider a possible error check for a desired speed that is too low (< 35MPH).

This might cause some designs to show a link to some alert sub-system to warn the driver of an improper speed setting. However, no such hint at an alert sub-system is given in the specification thus it would be more appropriate to consider this to be handled by using some alert service of the ACC.

Alternatively it can be easily accomplished by making the appropriate member functions return a Boolean value.

CCM Responsibilities Design:

DEACTIVATE:
This operation can employ the link to the FSA to invoke its service to turn off its automatic fuel regulating flow (which will be responsible for returning speed control to the driver). The service will also toggle the Active/Inactive state flag.

RESUME:
This service can use the link to the FSA to invoke its automatic fuel regulating flow service sending it the previously stored desired driver speed. This service (like the SET service) could be designed to error check the current speed and notify another part of the system if it below the 30 MPH threshold. If included in the design the low speed check/notification should be implemented as a private operation to prevent duplication and invoked from both the RESUME and SET services.

CCM Design: Behaviors

CCM Responsibilities Design:

ACCEL:
Each time this service is invoked it will increase the stored desired speed by 2 MPH and use the link to the FSA to invoke its automatic fuel regulating flow service sending it the updated desired speed.

Consider a check of the Active/Inactive flag to determine if the button has been pressed while a speed has not been set, which would require the invoking of an alert service either of the ACC or some other sub-system.

CCM Design: Behaviors

CCM Responsibilities Design:

DECEL: (possible inverse of the ACCEL service)
The specification of the exact effect of DECEL is incomplete. It is named in the interface as an inverse operation.)

Each time this service is invoked it will decrease the stored desired speed by two, (one would also be appropriate since the interface and spec does not give complete DECEL details), and use the link to the FSA to invoke its automatic fuel regulating flow service sending it the updated desired speed.

Consider a check of the Active/Inactive flag to determine if the button has been pressed while a speed has not been set, which would require the invoking of an alert service either of the ACC or some other sub-system.

CCM Design: Behaviors

CCM Responsibilities Design:

PASSING:
According to the interface, this service is necessary to temporarily deactivate/disengage the CCM while passing.

The possibility exists that the ACC may invoke the service unnecessarily or incorrectly. The service would need to check its active/engage state to determine if it should ignore the service request.

In fully considering this service it should be apparent that an inverse to this service is needed. In a rolling hilly terrain the ACC may request multiple PASSING/COASTING services as the driver repeatedly accelerates/coasts up/down hills.

This service should change the TempDisengage state to indicate that passing is occurring.

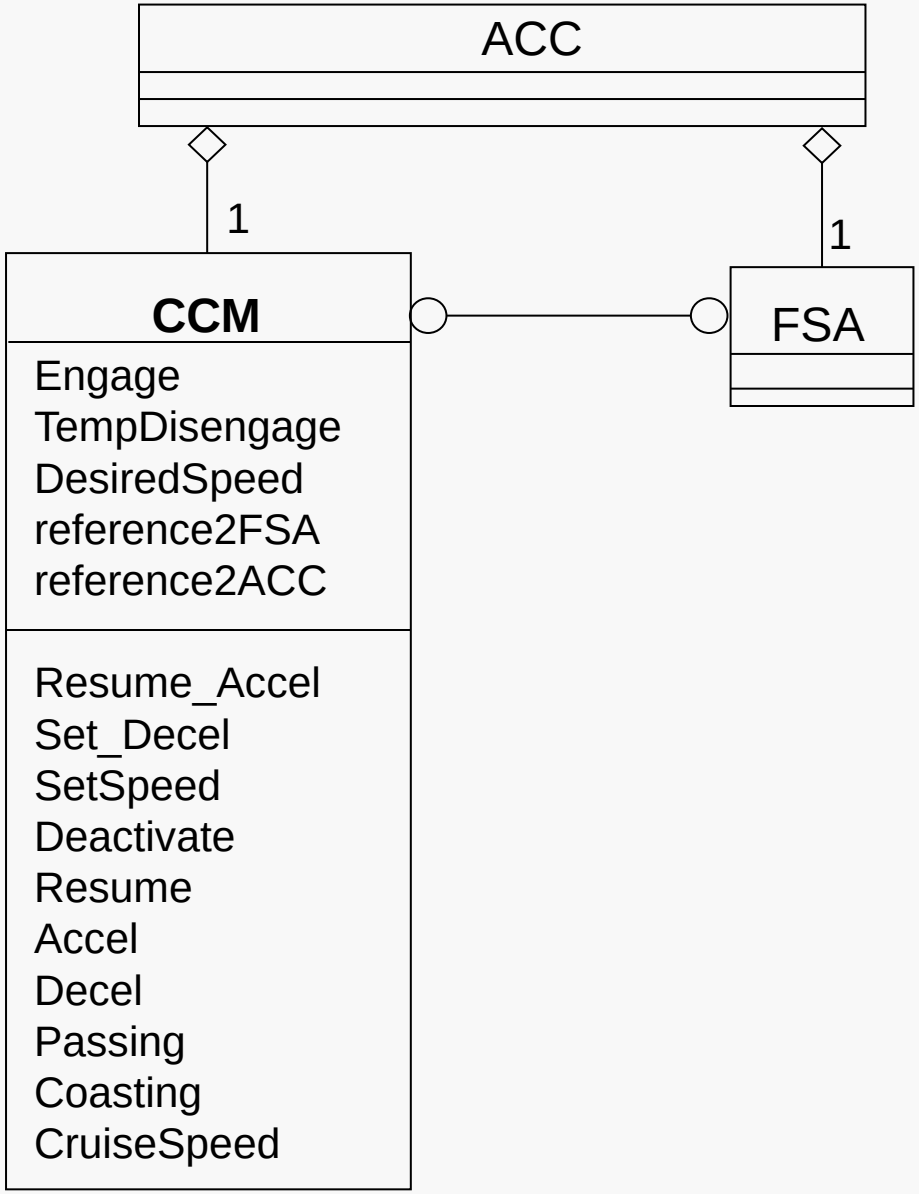
CCM Design: Behaviors

CCM Responsibilities Design:

COASTING:
While not present in the interface, this service is necessary to re-activate/re-engage the CCM after passing has occurred.

The same possibility exists that the ACC may invoke the service unnecessarily. The service would simply need to check its TempDisengage state to determine if it should ignore the service request.

CCM Class Relationships



CCM Design:

CCM Class details:

Class Name	CCM
Purpose	To store/update the driver's desired speed and communicating with the ACC and FSA.
States	Engage, TempDisengage
Constructors	CCM()
Operations	
Mutators	Resume_Accel, Set_Decel, SetSpeed, Deactivate, Resume,
	Accel, Decel, Passing, Coasting
Accessors	CruiseSpeed(), None
Fields	Engage, TempDisengage, DesiredSpeed, reference2FSA, reference2ACC

CCM Design:

CCM Class method details:

Prototype	CCM()
Purpose	construct a default CCM object
Data received	References to the FSA & ACC
Data returned	none
Remarks	Constructs a CCM object, setting the Engage field to false, TempDisengage to false, DesiredSpeed to 0, and setting the FSA & ACC links to the passed pointers.

CCM Class method details:

Prototype	Resume_Accel()
Purpose	Invoke Resume or Accel private member function
Data received	none
Data returned	none
Remarks	Invoked when user hits RESUME/ACCEL button. This service checks the Engage state. If engaged (true) it invokes Accel(), otherwise it invokes Resume().

CCM Class method details:

Prototype	SetDecel(int speed)
Purpose	Invoke the Set or Decel private member function
Data received	a non-negative integer vehicle speed
Data returned	none
Remarks	Invoked when user hits SET/DECEL button. This service checks the Engage state. If engaged (true) it invokes Decel(speed), otherwise it invokes Set().

CCM Design:

CCM Class method details:

Prototype	SetSpeed(int speed)
Purpose	Store the desired cruising speed, invoking the FSA FuelControl service.
Data received	a non-negative integer vehicle speed
Data returned	Boolean value indicating success or failure
Remarks	If speed is < 35 Set performs no action returns false Else it sets DesiredSpeed = speed, Engage=true and invokes the FSA -> FuelControl(speed) service, returns true.

CCM Class method details:

Prototype	Deactivate()
Purpose	Disengages the Cruise Control
Data received	none
Data returned	none
Remarks	Sets Engage=false and invokes the FSA -> HaltControl() service.

CCM Class method details:

Prototype	Resume()
Purpose	Re-engage the Cruise Control with stored speed.
Data received	none
Data returned	Boolean value indicating success or failure
Remarks	If ACC->currentspeed() service returns < 30 Resume performs no action, returns false Else it sets Engage=true and invokes the FSA -> FuelControl(DesiredSpeed) sevice, returns true.

CCM Design:

CCM Class method details:

Prototype	Accel()
Purpose	To increase vehicle by 2 MPH each time invoked.
Data received	none
Data returned	none
Remarks	It adds 2 to DesiredSpeed, and invokes the FSA -> FuelControl(DesiredSpeed) service

CCM Design:

CCM Class method details:

Prototype	Decel()
Purpose	To decrease vehicle by 2 MPH each time invoked.
Data received	none
Data returned	none
Remarks	It subtracts 2 from DesiredSpeed, and invokes the FSA -> FuelControl(DesiredSpeed) service

CCM Design:

CCM Class method details:

Prototype	Passing()
Purpose	To temporarily disengage the cruise control while passing.
Data received	none
Data returned	Boolean value indicating success or failure
Remarks	This service checks the Engage state. If engaged (true) and not temporarily Disengaged(false) it sets TempDisengage to true, invokes the FSA -> HaltControl() service and returns true else it returns false.

CCM Class method details:

Prototype	Coasting()
Purpose	To re-engage the cruise control after passing.
Data received	none
Data returned	none
Remarks	This service checks the Engage state. If engaged (true) and temporarily Disengaged(true) it sets TempDisengage to false, invokes the FSA -> FuelControl(DesiredSpeed) service and returns true else it returns false.

CCM Design:

CCM Class method details:

Prototype	CruiseSpeed()
Purpose	To return the stored desired cruising speed.
Data received	none
Data returned	a non-negative integer speed
Remarks	Returns a copy of the DesiredSpeed field.