



AUTOMOTIVE SOFTWARE DEV

HVAC SYSTEM BY KACPER WOLOSZYN

Contents

COMPOSITION DIAGRAMS.....	2
App Component Diagram.....	2
Inputs to ECU	3
Outputs from ECU	4
Inside ECU	5
Inside HeatingVariables	5
Inside OutsideVariables	6
Introduction.....	6
ApAirBlow SWC DESCRIPTION	7
ApFaceFootVentCtrl SWC DESCRIPTION.....	8
ApDesiredTemp SWC DESCRIPTION	9
SeatController SWC DESCRIPTION	10
ApHeatingSeats SWC DESCRIPTION.....	11
CabinTempController SWC DESCRIPTION	12
FlapController SWC DESCRIPTION	13
ApFlapsDir SWC DESCRIPTION	14
AirDirection SWC DESCRIPTION.....	15
HeatingVariables SWC DESCRIPTION	16
APStart SWC DESCRIPTION	17
TempOutside SWC DESCRIPTION	18
Ignition SWC DESCRIPTION	19
OutsideVariables SWC DESCRIPTION	20
ECU SWC DESCRIPTION	21
AirBlow SWC DESCRIPTION.....	22
AirBlowOut SWC DESCRIPTION	23
AirMixIn SWC DESCRIPTION.....	24
AirMixOut SWC DESCRIPTION	25
CabinTempClosedLoopIn SWC DESCRIPTION.....	26
CabinTempClosedLoopOut SWC DESCRIPTION.....	27
HeatedSeats SWC DESCRIPTION	28
HeatedSeatsOut SWC DESCRIPTION.....	29
Switch SWC DESCRIPTION.....	30
SwitchOut SWC DESCRIPTION.....	31
Temp SWC DESCRIPTION	32
TempOut SWC DESCRIPTION	33

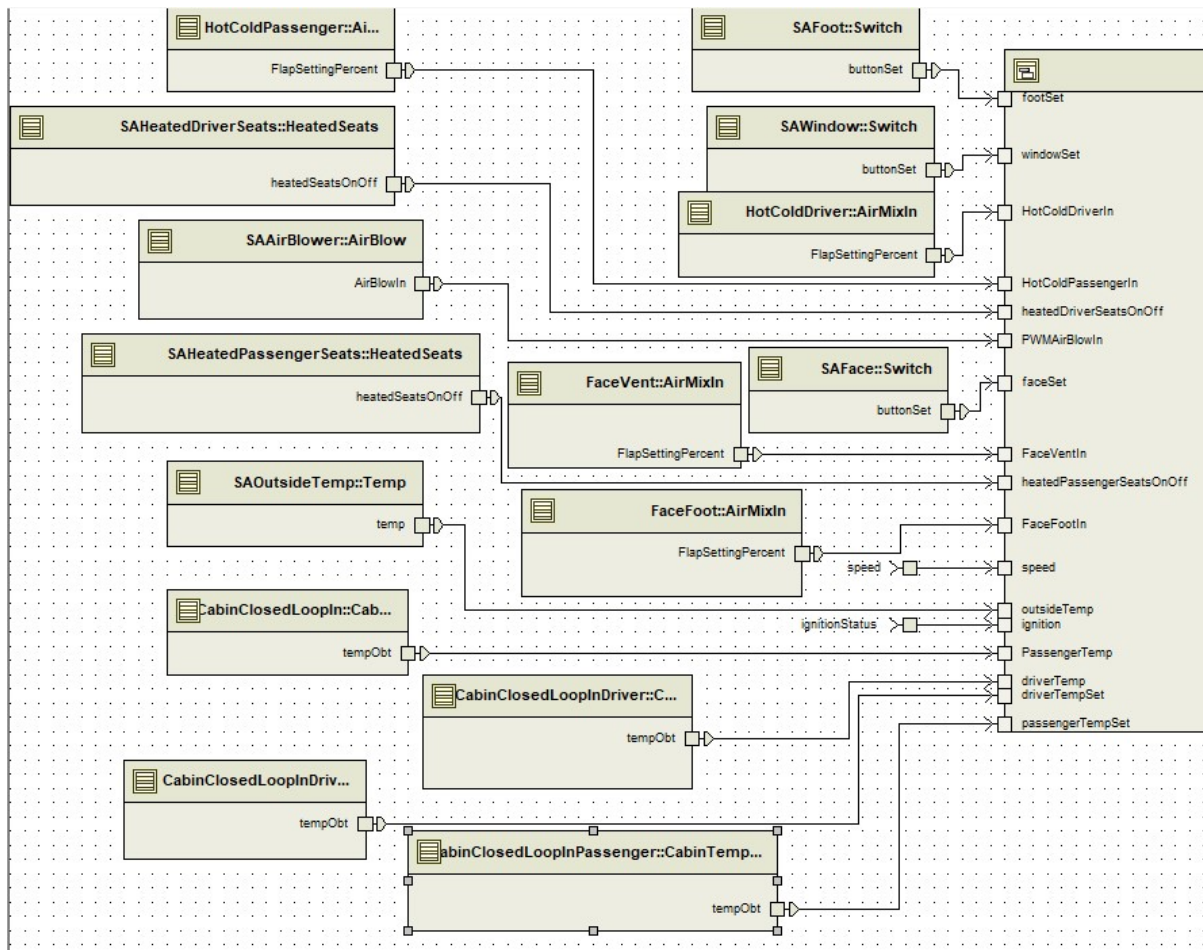
HVAC SWC DESCRIPTION	34
Runnable C Code	35
Data types Explained.....	37
Application Port Interfaces Explained	38
Mapping Sets Explained.....	38

COMPOSITION DIAGRAMS

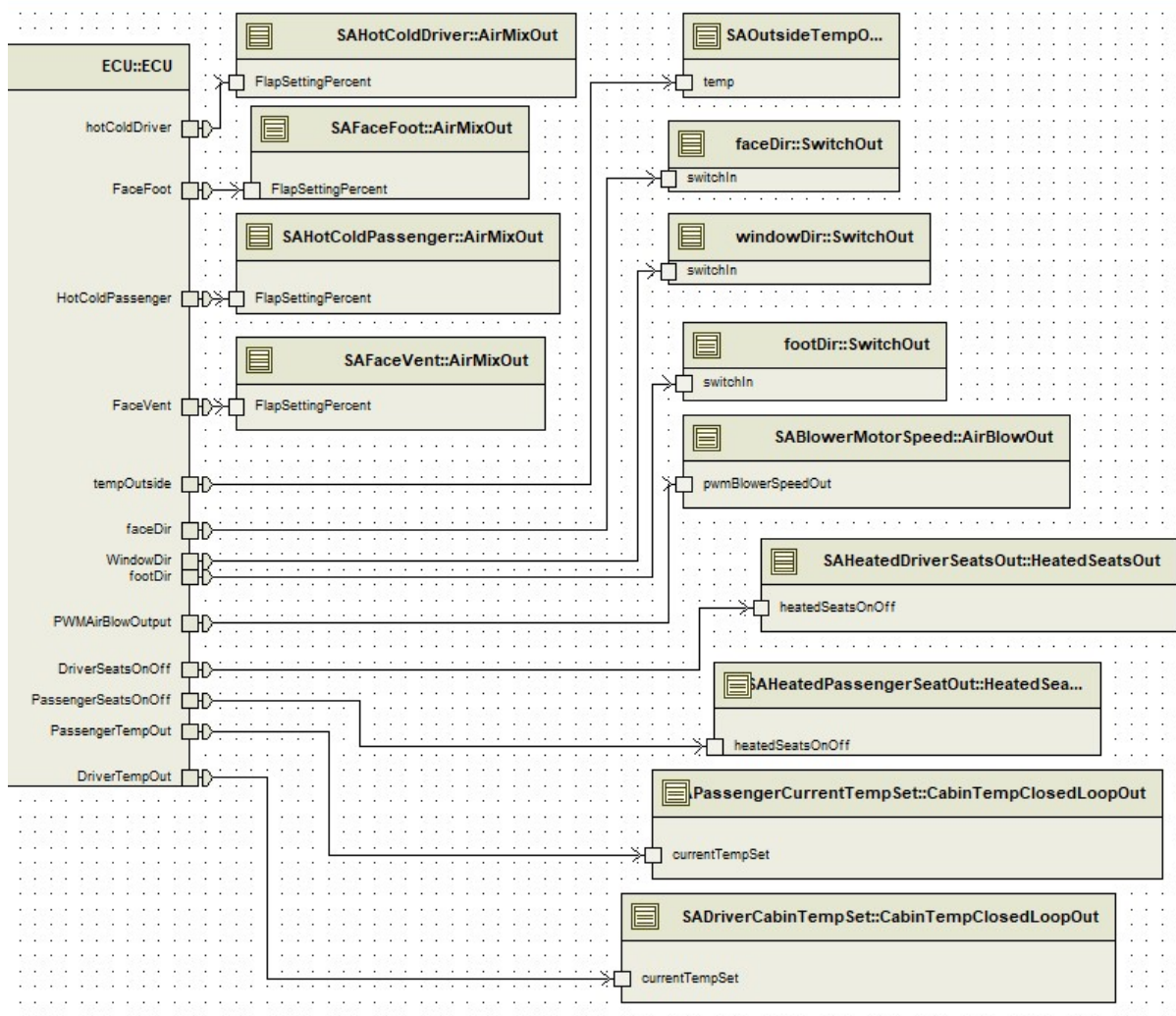
App Component Diagram

Application Component Types	
AirBlow	/ComponentTypes
AirBlowController	/ComponentTypes
AirBlowOut	/ComponentTypes
AirDirection	/ComponentTypes
AirMixIn	/ComponentTypes
AirMixOut	/ComponentTypes
ApAirBlow	/ComponentTypes
APDesiredTemp	/ComponentTypes
ApFlapsDir	/ComponentTypes
ApHeatingSeats	/ComponentTypes
APStart	/ComponentTypes
CabinTempClosedLoopIn	/ComponentTypes
CabinTempClosedLoopOut	/ComponentTypes
CabinTempController	/ComponentTypes
ECU	/ComponentTypes
FaceFootVentCtrl	/ComponentTypes
FlapController	/ComponentTypes
HeatedSeats	/ComponentTypes
HeatedSeatsOut	/ComponentTypes
HeatingVariables	/ComponentTypes
HVAC	/ComponentTypes
Ignition	/ComponentTypes
OutsideVariables	/ComponentTypes
SeatController	/ComponentTypes
Switch	/ComponentTypes
SwitchOut	/ComponentTypes
Temp	/ComponentTypes
TempOut	/ComponentTypes
tempOutside	/ComponentTypes

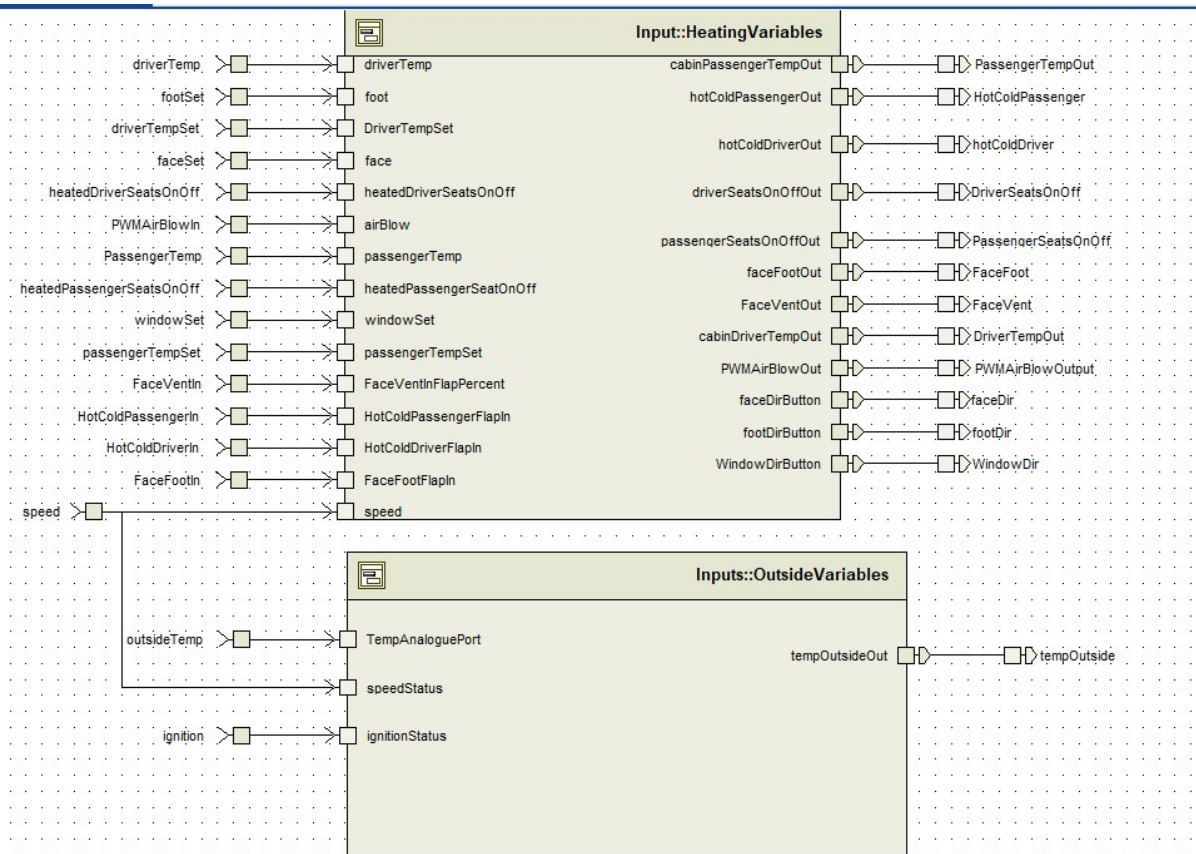
Inputs to ECU



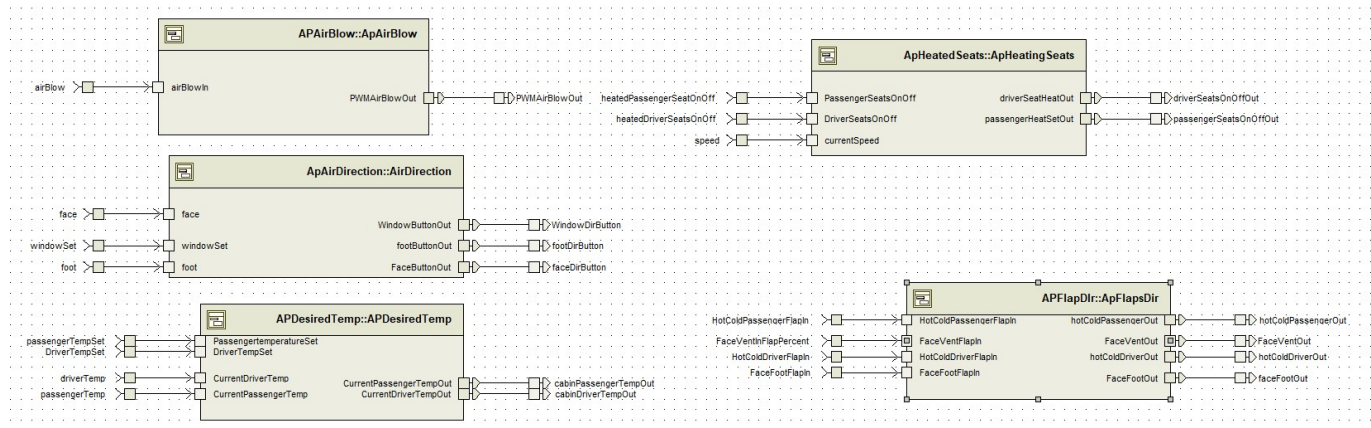
Outputs from ECU



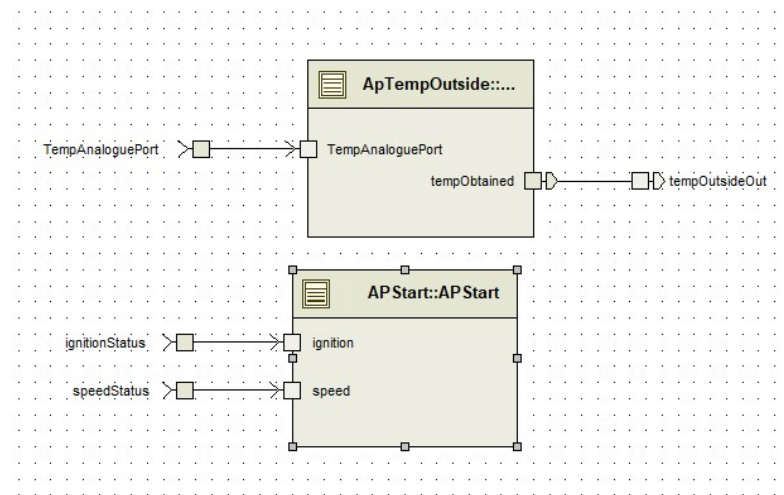
Inside ECU



Inside HeatingVariables



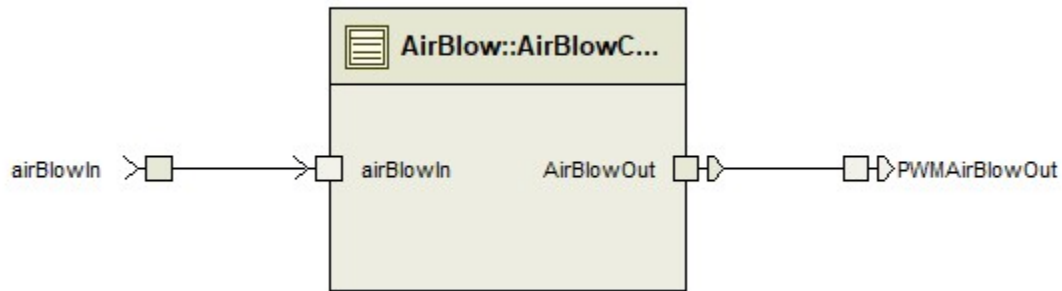
Inside OutsideVariables



Introduction

I will start with the lowest blocks, and then move up through the hierarchy, the last block being the HVAC itself. I have included port information, data mapping, C code and data types explanation at the end of this report.

ApAirBlow SWC DESCRIPTION



Functional

This is the block, that exchanges the value of AirBlowIn, it reads in AirBlowIn and it writes it out to AirBlowOut and that is then later written out to PWMAirBlowOut.

Interface

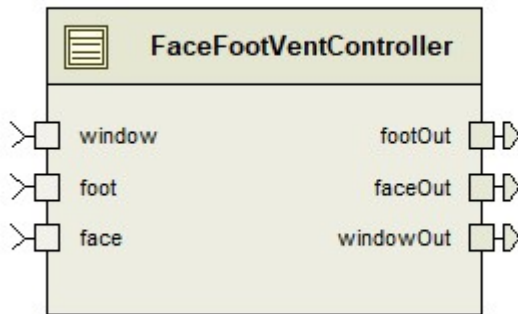
I used two application ports of Analoguelo, which reads in the value of AirBlowIn and AirBlowOut. These are mapped to speed.

Runnable

I have two runnable values. The first one is initAirBlow. This is firstly when the system is started it reads in the AirBlowIn percentage value and it writes out the percentage of AirBlow to PWMAirBlowOut.

The second runnable is readWriteMotor. It is triggered when it receives a signal from AirBlowIn of type speed. It has access to read from that port as it needs to know what is going out, and it has access to AirBlowOut. Speed to write it out to the HVAC system and to the user.

ApFaceFootVentCtrl SWC DESCRIPTION



Functional

This is the controller that is responsible for turning on the window heating, foot heating or face heating. It writes out a digitallo that the Boolean value was changed. It takes in the values for window, foot and face from the sensors actuators.

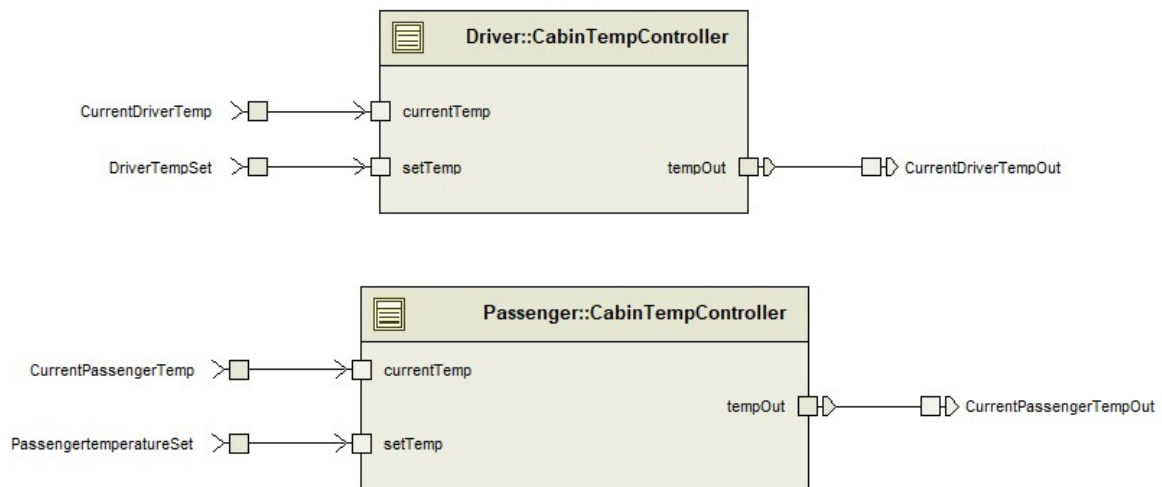
Interface

Three ports, for buttons, digitallo called window, foot and face. They can be either 0 or 1, Boolean value. They write out to specific ports if they have been turned on or off.

Runnable

For this block I have a runnable called **windBlowDir**, which is triggered when some button is pressed, and the buttons are polled every 20 ms. The runnable has access to read in the Boolean values for window, foot and face and to write out new Boolean values for those ports.

ApDesiredTemp SWC DESCRIPTION



Functional

This is an AP block, which does the work. Here I have two blocks, both coming from a block called CabinTempController, one for the passenger and the other one for the driver. As specified in the spec of the project, both the driver cabin and passenger cabin read in the set temperature the users have set and it also reads in the current value of temperature. It writes out one value, the current temperature of both driver and passenger cabin.

Interface

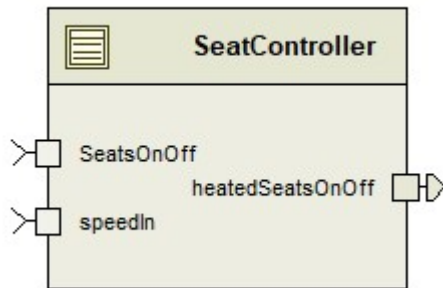
For this block I use two read ports and one write ports. The application port interface uses TemperatureAnaloguePort which has two values, tempIn and tempOut. TempSet is the temperature set by the user, and currentTemp is the temperature of the cabin.

Runnable

For the passenger block I have a runnable called TempControl. It is polled every 20ms, or it can be triggered by a temp change setTemp.tempIn. It has access to read all the ports, to see if the temperature has changed, and it writes out the tempOut port.

For the driver, due to it being the same AP Block, both runnable are an instance of APDesiredTemp

SeatController SWC DESCRIPTION



Functional

This block is responsible at the lowest end to heat the seats. I have two implementations of this. It takes a digital value of the **SeatsOnOffStatus**, and it takes in the speed and has an output of **heatedSeatsOnOff**. This block is where the input is written to an output due to the runnable and the speed is considered, as I wrote a runnable C code that if speed is below 10 km/h do not turn on the heated seats.

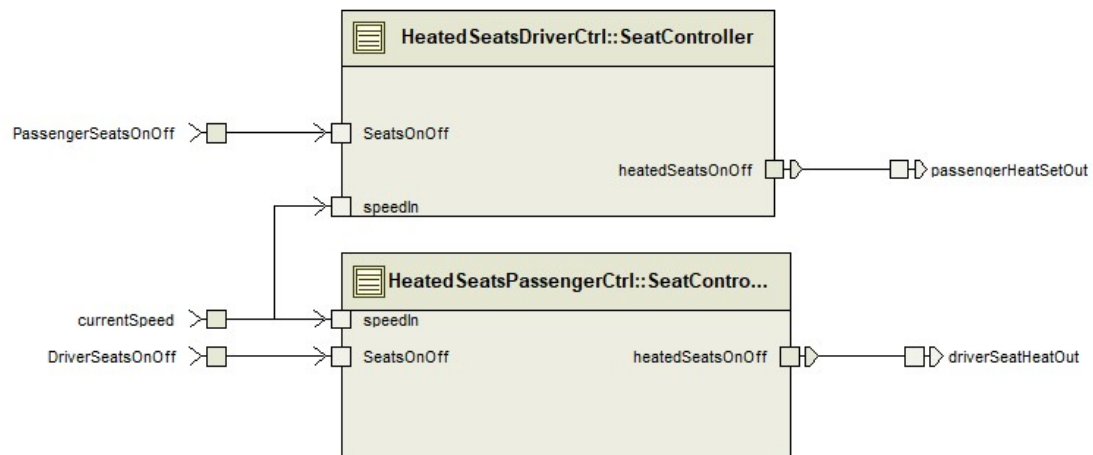
Interface

Two inputs that have been passed a few layers of the composition blocks, of **speedIn** which is an **AnalogueIO** value of speed, and **SeatsOnOff** which is a **Boolean** value. Those two are read in, and **heatedSeatsOnOff** is written out.

Runnable

The runnable I have for this block is called **heatedSeatsRun**. It is triggered on the reception of the **seatsOnOff** data port value. It has access to read from port **seatsOnOff**, where it gets the **Boolean** value and to **speedIn** where the speed is read in. It writes a **Boolean** value to **heatedSeatsOnOff**.

ApHeatingSeats SWC DESCRIPTION



Functional

There are two inputs to each of the blocks of SeatController. The input being the speed and then the digital IOs if the buttons were pressed. The speed is factored in due to the spec specifying that if the speed is below 10, the seats don't turn on. This is the runnable I do the code for, in which I specify an if statement, if speed is less than 10, don't set the output to be on.

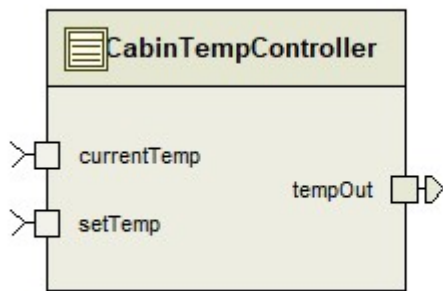
Interface

Three port inputs, PassengerSeatsOnOff, DriverSeatsOnOff and currentSpeed. The following are 2 digital IO's, Boolean (whether on or off) and speed is analogue. The outputs are HeatedSeatsOnOff for the driver or passenger, both of which are Boolean.

Runnable

The runnable I have set up is called heatedSeatsRun. It's triggered when a new value is set for the Boolean (so a button is pressed). It has access to read in the speed value, and the value of the seatsOnOff and it writes out the value for the heatedSeatsOnOff Boolean.

CabinTempController SWC DESCRIPTION



Functional

This block is responsible for taking in the value set by the user and the current temperature, and it writes out the desired temperature. This is the block that would have code to turn on the heating unit to the desired temperature.

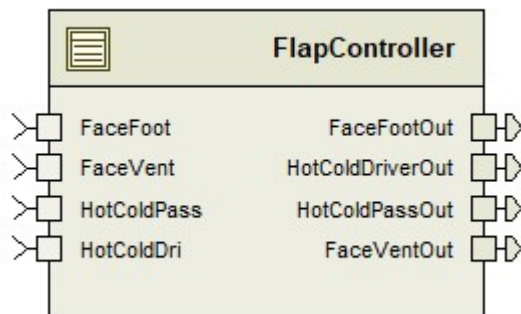
Interface

The block has two interfaces, `currentTemp` and `setTemp` of the `TempAnalogueIO`, which are both temperature values, one is the desired and the other one is the current. The `tempOut` would realistically give out the difference and an ECU would heat up the cabin to the desired temperature.

Runnable

For this block I have a runnable called `TempControl`. It is polled every 20msec to see the current temperature and to see if the switch of the other one has been changed, or it is triggered when a new temperature is passed in to `setTemp`. It has access to read all the `currentTemp` ports, to read both `setTemp` values and to write out the temperature for both driver and passenger cabin.

FlapController SWC DESCRIPTION



Functional

This is the main block that is responsible for reading the values of percentages that have been set on the flaps and writing the new values out. It is the controller, meaning all the values for the ports have been passed through the hierarchy. The analoguePercentage is the four flap position stepper motors that range from 0 to 100%.

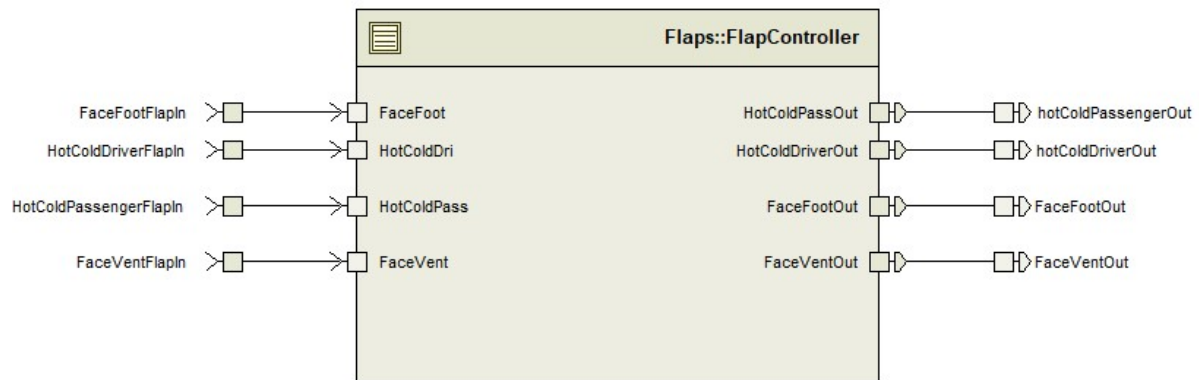
Interface

Four inputs of AnaloguePercent for FaceFoot, FaceVent, HotColdPassenger and HotColdDriver. These values are read in. The values written out are FaceFootOut, HotColdDriverOut, HotColdPassengerOut and FaceVentOut.

Runnable

The runnable I have set up for this block is called flapPos. It polls the switches every 20 ms and it has a trigger that makes the runnable run on data reception from the four input ports. The runnable has access to read the four ports for the analoguePercentage and to write out the new values for analoguePercentage.

ApFlapsDir SWC DESCRIPTION



Functional

This is the APFlapController. This is the block that reads in the percentage values for specific flaps, such as face foot, hot cold passenger, face vent or hot cold driver. It then runs the runnable and writes out new values for the four ports as stated. These ports write out to the HVAC system.

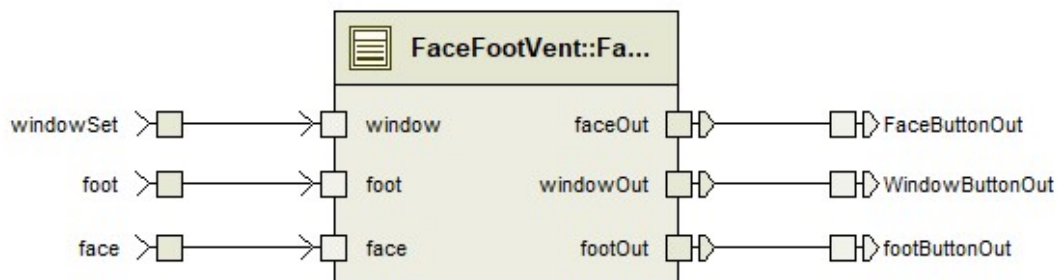
Interface

There are 8 inputs of Analogue Percentage in the block. They take 4 inputs and have 4 outputs. The percentage analogue inputs correspond to the flap in.

Runnable

I have a flapsPos runnable. This is triggered every 20msec to see if a changed occurred, it also reads the value of the ports, so whenever the flapsSettingPercentage is changed it triggers the runnable. The runnable has access to all the 8 ports, to read if a change occurred and to write the new analoguePercentage values of the block.

AirDirection SWC DESCRIPTION



Functional

This is the lowest level for the AirDirection, it passes in the window, foot or face button status which has been pressed in the HVAC. The block has 3 receiver ports, and three write ports. It writes out to the ECU ports telling the ports if the button was pressed or not.

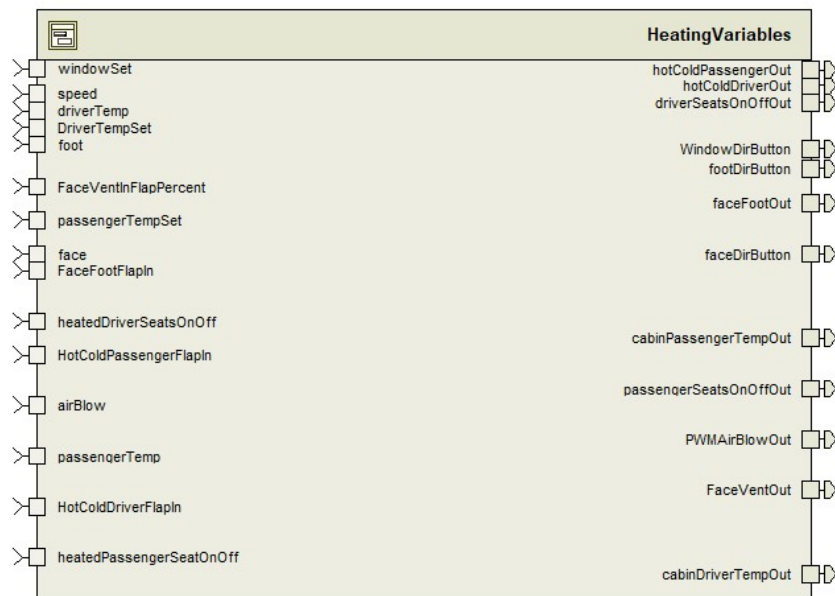
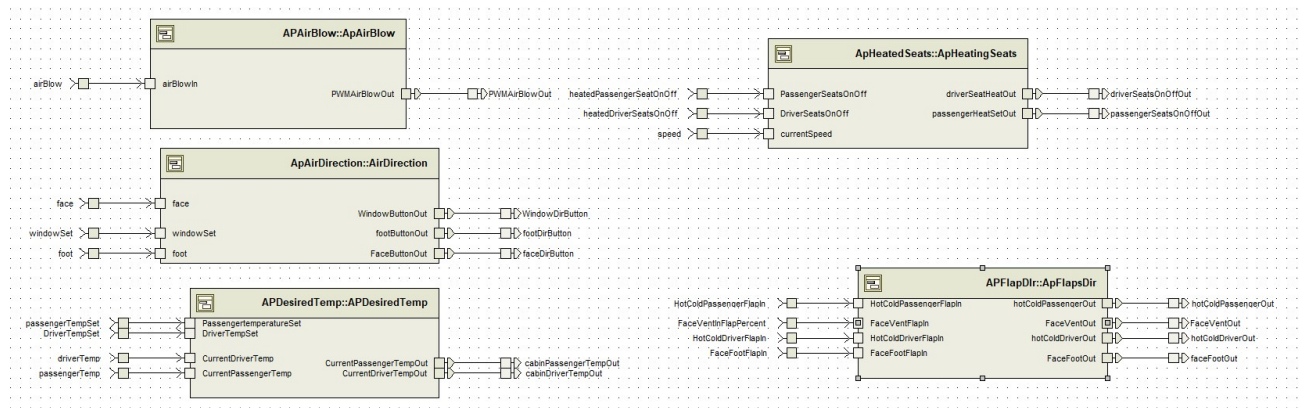
Interface

All the interfaces here are digitallo as we are dealing with buttons. The windowSet, foot and face buttons are from the ECU, and they were initialized by the Sensor actuator at the start. The output ports are writing to the output part of the ECU. The runnable here swaps the value for a pressed button to be the exact one and it can also show which buttons have not been pressed.

Runnable

The runnable I have for this is called `windBlowDir`, it is triggered on a polling of 20msec or when the buttons face, foot or window receive input thanks to polling. The runnable has access to read the ports and to write new values out onto the ports, the values being all Boolean digitallo values.

HeatingVariables SWC DESCRIPTION



Functional

The aim of this block was to keep all the heating variables and outside variables separate. This composition. This links all the ports. There are five compositions inside of this composition, each one responsible for a different heating part of the system. There is the direction, PWMAirBlowOut, DesiredTemperature, HeatedSeats and FlapDirection composition blocks inside of this block.

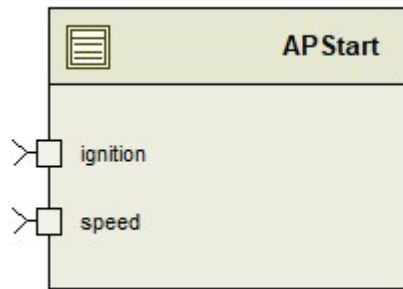
Interface

It takes most of the ports from the ECU, except the ones that are not necessary for the HeatingVariables, such as outsideTemperature. The ports vary from analoguePercentage, digitallo, TempAnalogueIO and AnalogueIO. This block reads and writes values from the ECU

Runnable

This is a composition, so it has no runnable.

APStart SWC DESCRIPTION



Functional

This block reads in the value for ignition and speed. I have this block as I was getting errors when trying to pull the block of HVAC inside into the OutsideVariables, hence why this is created. It reads in values of ignition and speed and how they change, they are not written out as we do not have to account for that in this model.

Interface

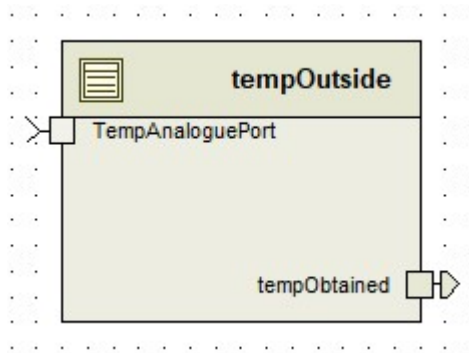
Two receiver ports, of speed and ignition. They receive the values for ignition and speed that were set in the HVAC and passed in through the ECU and OutsideVariables.

Runnable

I have two runnables, one called ignition, which is triggered at Init, it has a port access to read in the ignition status Boolean value.

The second runnable is called speedInit, it is also called on initialization, and it has port access to read the speed port.

TempOutside SWC DESCRIPTION



Functional

This block is used inside the composition of OutsideVariables. An analoguePort temp value comes to TempAnaloguePort and this block then writes out a new tempObtained value to tempOutsideOut port. I use this block to read and write the outside temperature

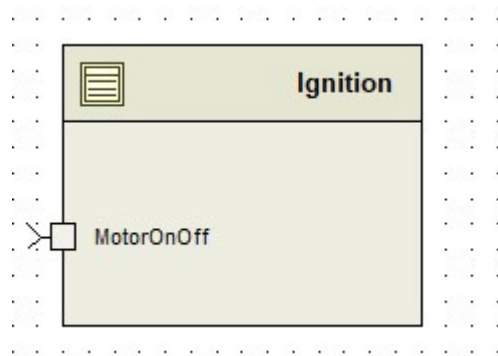
Interface

One receiver and one sender port. TempAnaloguePort and tempObtained. The temperature is received, and a new value is written out through the sender port.

Runnable

For TempOutside I have two runnables, one Read runnable that is triggered on data received and it reads the data, the other runnable is called Write and is triggered on TempIn again but it has port accesses to write to tempObtained and to read the TempAnaloguePort.

Ignition SWC DESCRIPTION



Functional

This block is used in the OutsideVariables. It receives a digitallo signal whether the motor is on or off. The signal will be a Boolean.

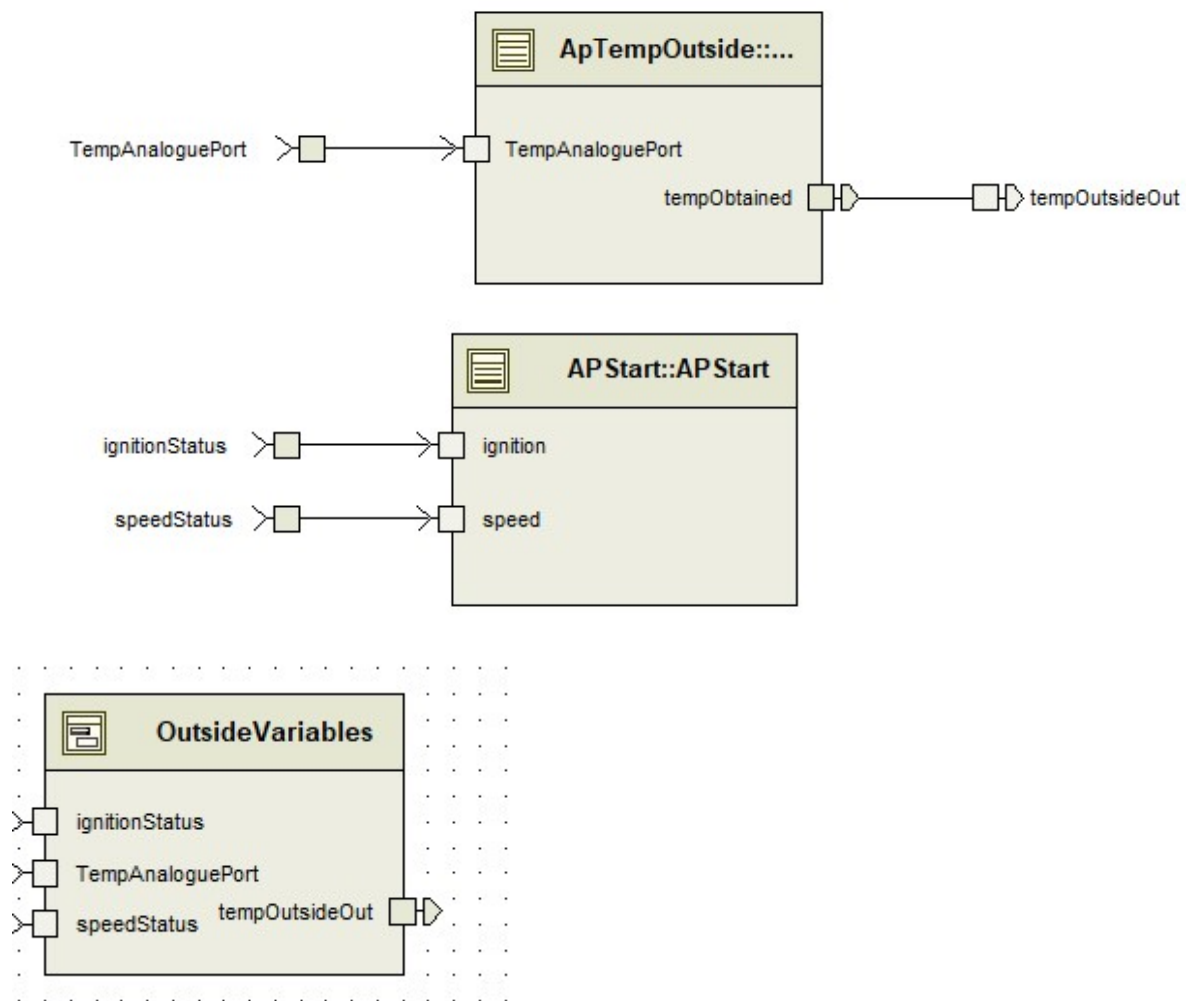
Interface

The MotorOnOff is a digitallo port that receives information about the ignition status.

Runnable

I have a runnable for ignitionPoll which is triggered at Init, so when the runnable is started it reads the value of the ignition status, the Boolean value if the motor is on or off.

OutsideVariables SWC DESCRIPTION



Functional

This SWC block is a composition. It has three inputs and one output. It takes in the outside variables into account. Here the speed and ignition status are read in a block called **APStart** and the **ApTempOutside** block is responsible for reading the value of the **TemperatureAnaloguePort** and to write the temperature to a port. This **OutsideVariables** block is responsible for gathering everything that does not have to do with the heating system itself together. There are two AP blocks in this composition

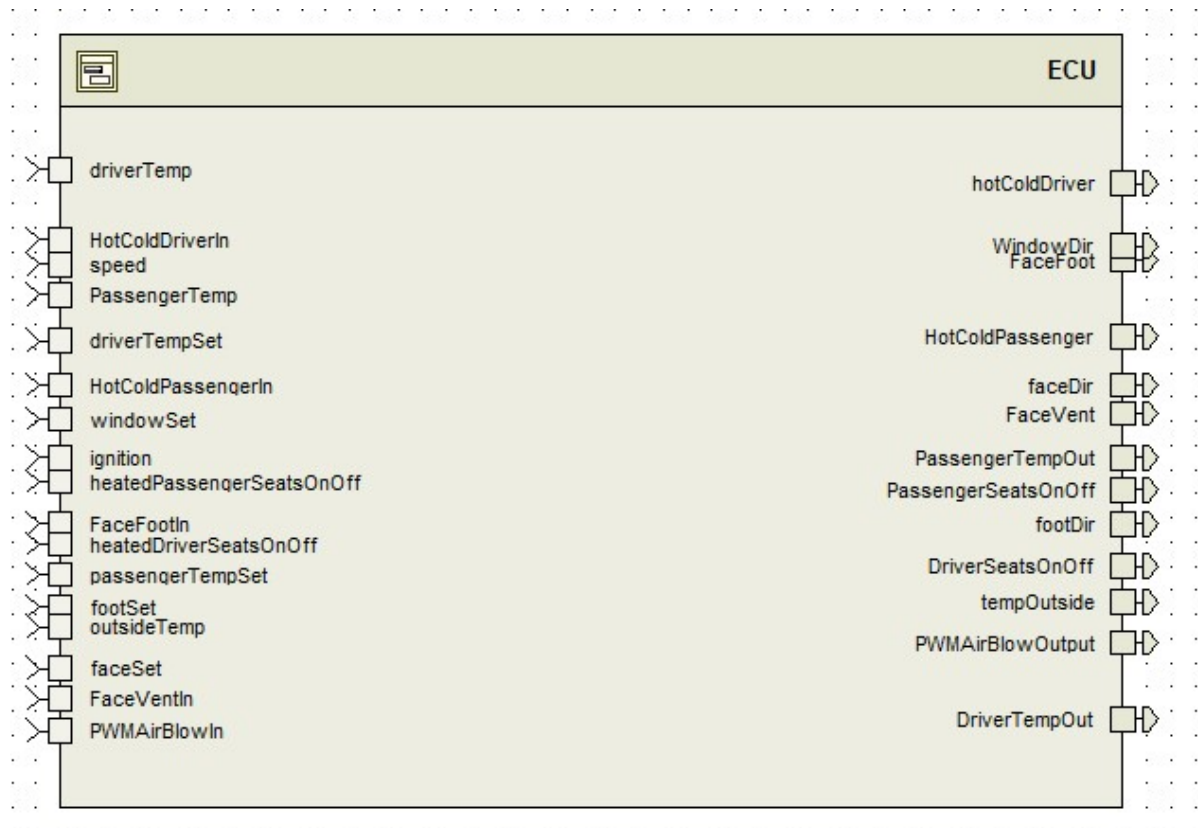
Interface

Three ports, one **AnalogueIO** for speed, one **TempAnaloguePort** to read in the outside temperature and one **Boolean digitalIO** to see whether the ignition is turned on or off. It writes out the temperature as the **temperatureOutside** is used in a block in the HVAC.

Runnable

This is a composition, so there is no runnables.

ECU SWC DESCRIPTION



Functional

This block is the linking point for the Sensor Actuators, where they are read in, and the writes of them are then written out to the HVAC system. Inside of this are two compositions called HeatingVariables, and OutsideVariables. Heating is for HVAC and Outside are things like speed, ignition and outsideTemperature.

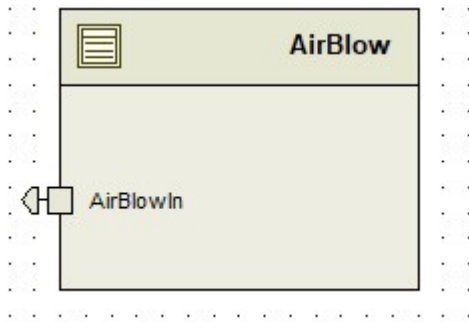
Interface

Interfaces can be seen on the diagram, this ECU composition reads in TempAnalogueIO, Analoguelo, digitallo, analoguePercent and speed and writes out TempAnalogueIO, Analoguelo, AnaloguePercent and digitallo.

Runnable

No Runnable as this is a composition.

AirBlow SWC DESCRIPTION



Functional

There is a sender port that is inside the component, that sends a signal to the HVAC. The component is a sensor actuator. Whenever a signal is received it sends a signal to the Heating system.

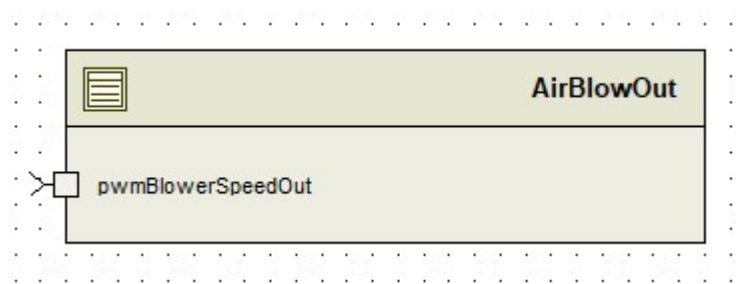
Interface

I use an Analogue input output as the port interface, as I would like to send a PWM signal of AirBlowIn to the HVAC system.

Runnable

PolIDial is the runnable for this component. It is triggered every 20msec and when a signal is received, the runnable has access to write out the speed of the AirBlowIn.

AirBlowOut SWC DESCRIPTION



Functional

This is a block that is the output from the ECU. This has a signal that it receives from the ECU called pwmBlowerSpeedOut.

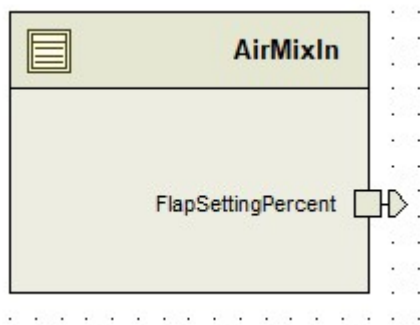
Interface

I use an AnalogueIO port for receiving the speed for the motor that has been set by AirBlow by the user.

Runnable

The runnable I use is SpeedOutOnReception. This is triggered when the ECU sends out a signal and bwmBlowerSpeedOut receives a signal. The port access I have is pwmBlowerSpeedOut to read in the value.

AirMixIn SWC DESCRIPTION



Functional

This is a Sensor Actuator Block, I have a sender port that is sending out the current **FlapSettingPercent**. In the HVAC system I have one block for each Hot Cold Passenger setting, Hot Cold Driver setting, Face Vent flaps settings and Face Foot flaps settings.

Interface

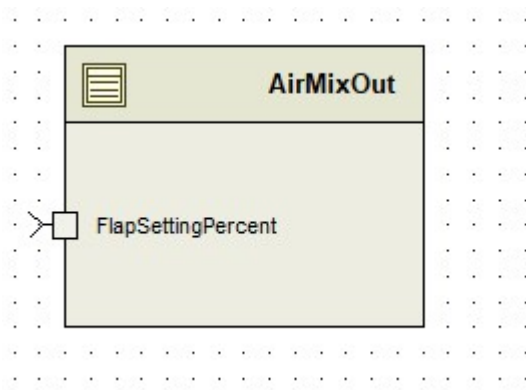
I use an **analoguePercent** port that sends out the value of the flaps, the percentage value for each of the flaps that have been set.

Runnable

I have two runnables, the first one being triggered on the initialization, this writes out the current value of the percentages that have been set for the flaps. It is trigger on initialization

The second runnable is **flapsPosPoll**. This runnable is triggered by a periodic time of 20ms, and every 20 ms it writes out the **flapSettingPercentage**.

AirMixOut SWC DESCRIPTION



Functional

This is a block that obtains the value of the analogue percentage after it passes through the whole system. This block contains the value for the flaps position that have been written onto the ECU.

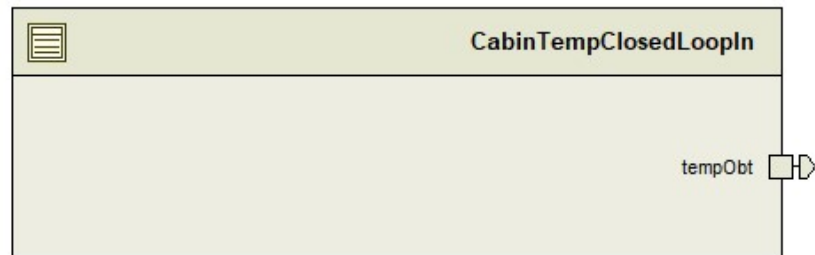
Interface

This block has an analogue percentage port that reads in the value that has been written onto the ECU. It contains a mapping set of `flapMapSet`, which I will talk about in the later part of this report.

Runnable

I have a runnable called `flapsPosRead`. This runnable is triggered when a new value of analogue percent is used. This runnable has access to read the `FlapSettingPercent`.

CabinTempClosedLoopIn SWC DESCRIPTION



Functional

I use this to obtain the status of the temperature of the cabin for the driver and for the passenger. This is also used to set the desired temperature. The block has a write port that writes the values to the ECU, in the HVAC I have four instances of this block.

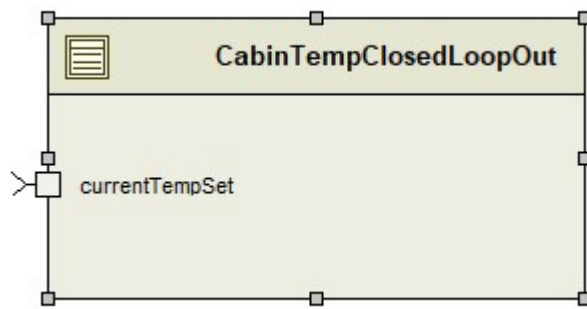
Interface

For this block I use a TempAnaloguePort that is the sender port, so this sends the information to the ECU of the HVAC System.

Runnable

The runnable I have for this block is called pollTemp. It polls every 20 msec, and it has access to write out the temperature set by the driver and the current temperature of the cabin. I have this as a polling trigger to make sure there is no latency, so when a driver wants a specific temperature, that temperature is inserted to system as soon as possible.

CabinTempClosedLoopOut SWC DESCRIPTION



Functional

This block is on the right-hand side, it is the output of the Ecu. It takes in the current temperature set. I have four instances in the HVAC, one for passenger heat, driver heat current temperatures and one for setting the passenger heat and one for setting driver heat. This block reads the set values.

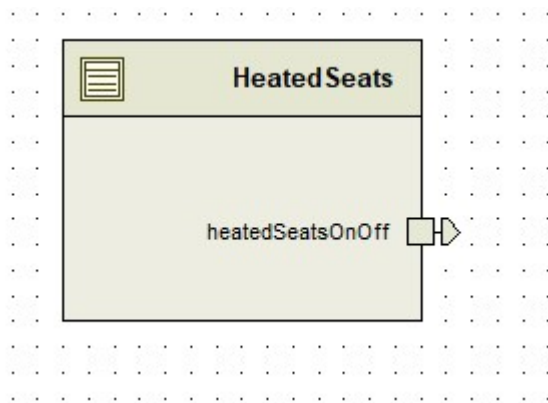
Interface

I have a TempAnaloguePort reading in the current temperatures for driver / passenger cabins and reading in the values of the set temperatures for driver and passenger.

Runnable

The runnable I have for this block is ReadCabinTemp. It is triggered when a new data element is received, it has access to read the temperature set and the current temperature of the driver and passenger cabin.

HeatedSeats SWC DESCRIPTION



Functional

This block has two instances in the HVAC. It writes out the new value of HeatedSeatsOnOff, so it is either on or off, the status of the port is written out.

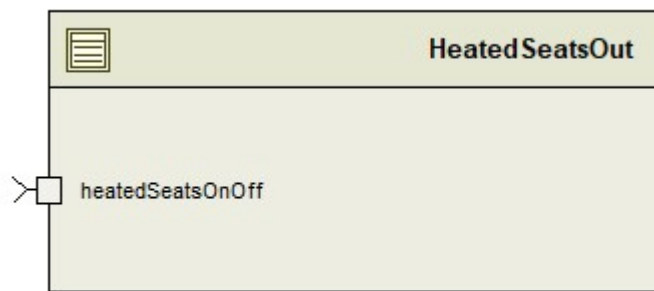
Interface

This block writes out the value/ status for the HeatedSeatsOnOff button for either the driver or the passenger.

Runnable

I have a heatedSeatsSwitchPoll runnable for this block. It checks every 20 msec if the button was pressed, so therefore it has access to write the value of the button to the ECU.

HeatedSeatsOut SWC DESCRIPTION



Functional

This block reads in the value of the button, or the HeatedSeatsIn status. It contains a receiver port that takes in a Boolean value. It is in the output part of the ECU. In the HVAC system I have two instances, one for the driver and the other for passenger.

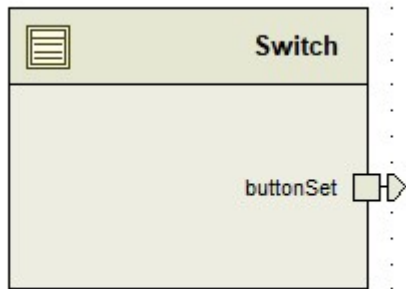
Interface

One receiver port that takes in a value of the button that has been polled. It is a receiver port of digitallo.

Runnable

The runnable I have set up for this block is `readHeatedSeatsStatus`. This is triggered when on data reception of the button `HeatedSeatsOnOff` for either the driver or the passenger. It has a port access to read of the `HeatedSeatsOnOff` Boolean value.

Switch SWC DESCRIPTION



Functional

This block is the switch and is responsible for setting the window, face or foot air blow. The button Set is a Boolean variable, so it is digital. The switch has three calls in the HVAC for foot, face and window. The buttonSet writes out a Boolean value of which button was pressed to the ECU.

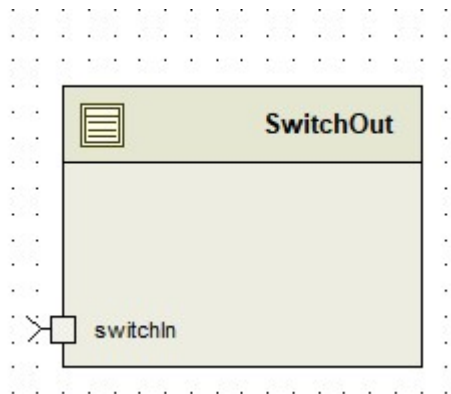
Interface

ButtonSet port that is of digital and contains a Boolean value of 1 or 0. Writes the value to the ECU port so that the ECU knows which button was pressed.

Runnable

The runnable I have for this block is called pollSwitch. It polls every 20msec whether the switch is pressed. The port access for this runnable is a write to buttonSet, so if the button has been pressed it writes the Boolean value out to the ECU.

SwitchOut SWC DESCRIPTION



Functional

This block is responsible for reading in the Boolean value that has been sent by the ECU whether switch is on or off. There are three instances of the SwitchOut, one for face, one for window and one for foot.

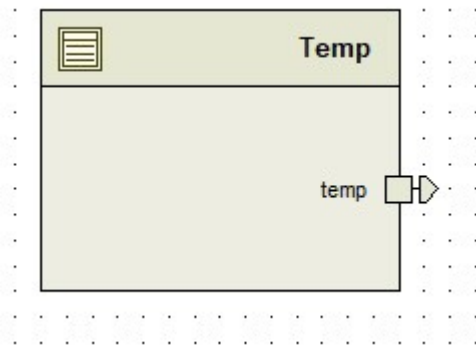
Interface

The switchIn port is a digitalIo port that reads in a value of the specific port if it's on or off. I use it three times to find out if the switch is on for foot, face or window direction.

Runnable

The runnable I have for this block is called readSwitchvalue, it is triggered on switchIn data reception and it has access to the port that is called switchIn, with a read access only. The runnable is across all three of the instances of the block.

Temp SWC DESCRIPTION



Functional

This block is used to obtain and write a temperature value for a port called temp, that is a TempAnaloguePort. I use this block in the HVAC as outsideTemperature. It has one sender port that sends the temp value to the ECU block.

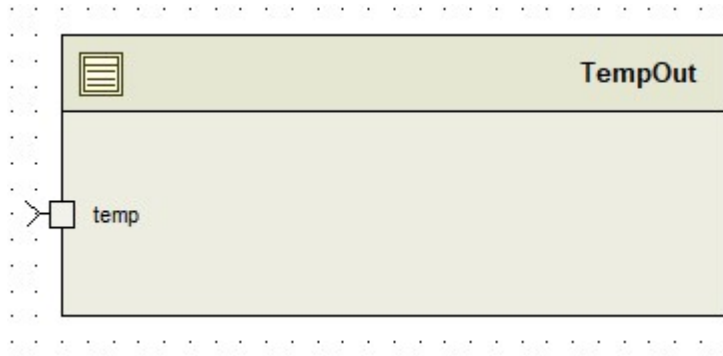
Interface

This block has one port interface of type TempAnaloguePort that writes out and sends the value of outsideTemperature in the HVAC system.

Runnable

For this block I have one runnable Entity called writeTemp, which is triggered every 20 msec, due to a polling periodic trigger. The port access for this block is the temp port where it is written out.

TempOut SWC DESCRIPTION



Functional

This block reads in the value set for temp outside in the ECU. It has a TempAnaloguePort as a receiver. This block receives the temperature.

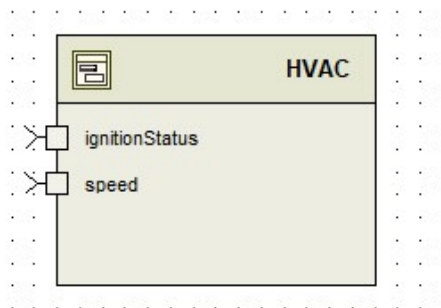
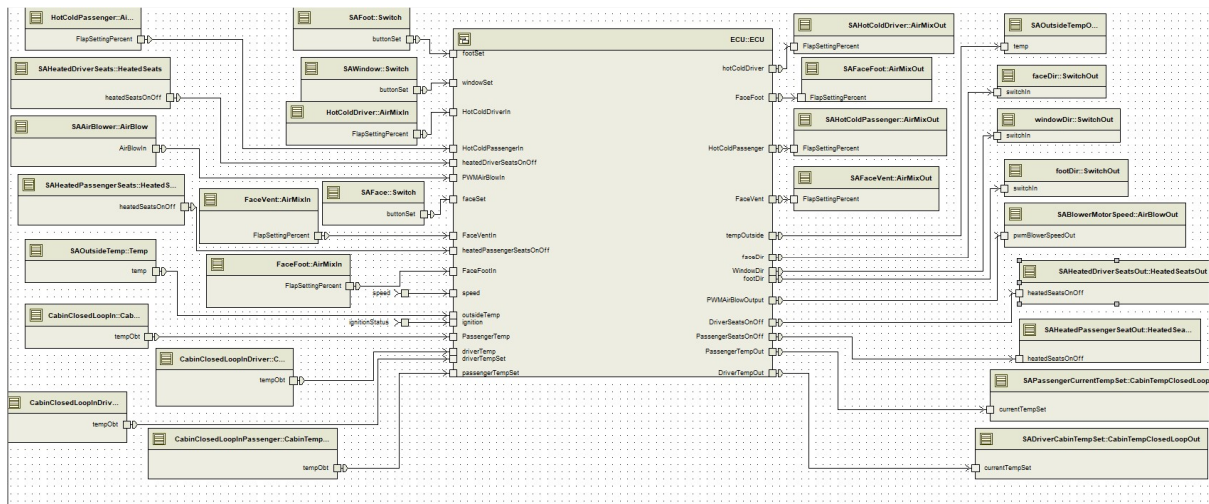
Interface

This block has a TempAnaloguePort port interface as a receiver. This block receives the temperature.

Runnable

The runnable in this SWC is called readTemp, as this block reads the temperature value set. It is triggered on temp data reception and it has access to read from the temp port.

HVAC SWC DESCRIPTION



Functional

This is the composition of all the whole system. On the left we can see the sensor actuator inputs to the ECU and on the right the outputs from the ECU. This makes up all the HVAC system. The function of this composition is to bring all the components and component outputs together. There are also two in depended HVAC ports that initialize the ignition Status and speed.

Interface

Two independent ports that are Boolean digitallo port and speed, AnalogueIO port. The components of sensor actuators (all the buttons) are initialized in the HVAC here. The application port interfaces used in the HVAC are: AnalogueIO, AnaloguePercent, digitallo and TempAnaloguePort.

Runnable

There are no runnables since this is a composition block.

Runnable C Code

```
1  /******  
2  * FILE DESCRIPTION  
3  *  
4  * File: SeatController.c  
5  * Config: 0  
6  * SW-C Type: SeatController  
7  * Generated at: Wed Nov 21 14:19:15 2018  
8  *  
9  * Generator: MICROSAR RTE ContractPhase Generator Version 3.13.5  
10 * RTE Core Version 1.18.0  
11 *  
12 * License:  
13 * Description: C-Code implementation template for SW-C <SeatController>  
14 *  
15 *  
16 *  
17 /******  
18 * DO NOT CHANGE THIS COMMENT!      << Start of version logging area >>      DO NOT CHANGE THIS COMMENT!  
19 *  
20 *  
21 /* PRQA S 0777, 0779 EOF */ /* MD_MSR_5.1_777, MD_MSR_5.1_779 */  
22 *  
23 /******  
24 * DO NOT CHANGE THIS COMMENT!      << End of version logging area >>      DO NOT CHANGE THIS COMMENT!  
25 *  
26 *  
27 #include "Rte_SeatController.h" /* PRQA S 0857 */ /* MD_MSR_1.1_857 */  
28 *  
29 *  
30 /******  
31 * DO NOT CHANGE THIS COMMENT!      << Start of include and declaration area >>      DO NOT CHANGE THIS COMMENT!  
32 *  
33 *  
34 *  
35 /******  
36 * DO NOT CHANGE THIS COMMENT!      << End of include and declaration area >>      DO NOT CHANGE THIS COMMENT!  
37 *  
38 *  
39 *  
40 /******  
41 *  
42 * Used AUTOSAR Data Types  
43 *  
44 *  
45 *  
46 * Primitive Types:  
47 * =====  
48 * boolean: Boolean (standard type)  
49 * uint8: Integer in interval [0..255] (standard type)  
50 *  
51 *  
52 *  
53 *  
54 #define SeatController_START_SEC_CODE  
55 #include "SeatController_MemMap.h" /* PRQA S 5087 */ /* MD_MSR_19.1 */  
56 *  
57 /******  
58 *  
59 * Runnable Entity Name: run  
60 *  
61 *  
62 *  
63 * Executed if at least one of the following trigger conditions occurred:  
64 * - triggered on DataReceivedEvent for DataElementPrototype <boolean> of PortPrototype <SeatsOnOff>  
65 * - triggered on DataReceivedEvent for DataElementPrototype <speed> of PortPrototype <speedIn>  
66 *  
67 *  
68 *  
69 * Input Interfaces:  
70 * =====  
71 * Explicit S/R API:  
72 * -----  
73 * Std_ReturnType Rte_Read_SeatsOnOff_boolean(boolean *data)  
74 * Std_ReturnType Rte_Read_speedIn_speed(sint16 *data)  
75 *  
76 * Output Interfaces:  
77 * =====  
78 * Explicit S/R API:  
79 * -----  
80 * Std_ReturnType Rte_Write_heatedSeatsOnOff_boolean(boolean data)  
81 *  
82 *  
83 /******  
84 * DO NOT CHANGE THIS COMMENT!      << Start of documentation area >>      DO NOT CHANGE THIS COMMENT!  
85 * Symbol: run_doc  
86 *  
87 *  
88 */
```

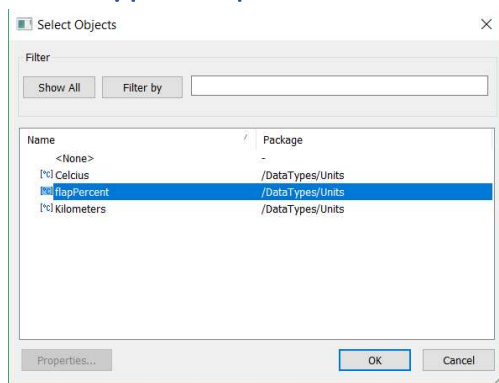


```

89  [
90  [ * DO NOT CHANGE THIS COMMENT!          << End of documentation area >>          DO NOT CHANGE THIS COMMENT!
91  [
92  [
93  FUNC(void, SeatController_CODE) run(void) /* PRQA S 0850 */ /* MD_MSR_19.8 */
94  {
95  [
96  [ * DO NOT CHANGE THIS COMMENT!          << Start of runnable implementation >>          DO NOT CHANGE THIS COMMENT!
97  [ * Symbol: run
98  [
99  sint16 speed;
100  boolean SeatController;
101  Std_ReturnType retVal;
102  (void) Rte_Read_FpSpeed_SeatController(self, &speed);
103  (void) Rte_Call_FpGetSpeedStatusIoHwAb_IoHwAb_Get_SeatController(self, &SeatController);
104
105  // if speed is less than 10, don't heat seats
106  if (speed < 10 )
107  {
108  [
109  [ Rte_Call_FpSpeed_SeatController_SetEventStatus(self, DEM_EVENT_STATUS_FAILED);
110  [ retVal = Rte_Call_FpSetSpeedStatusIoHwAb_IoHwAb_Set_Pwm_Signals(self, 0);
111  [
112  else
113  {
114  [ Rte_Call_FpSpeed_SeatController_SetEventStatus(self, DEM_EVENT_STATUS_PASSED);
115  [ retVal = Rte_Call_FpSetSpeedStatusIoHwAb_IoHwAb_Set_Pwm_Signals(self, (sin16)speed);
116  [
117  (void)self;
118  [
119  [ * DO NOT CHANGE THIS COMMENT!          << End of runnable implementation >>          DO NOT CHANGE THIS COMMENT!
120  [
121  [
122  [
123
124  #define SeatController_STOP_SEC_CODE
125  #include "SeatController_MemMap.h" /* PRQA S 5087 */ /* MD_MSR_19.1 */
126
127
128  [
129  [ * DO NOT CHANGE THIS COMMENT!          << Start of function definition area >>          DO NOT CHANGE THIS COMMENT!
130  [
131  [
132  [
133  [
134  [ * DO NOT CHANGE THIS COMMENT!          << End of function definition area >>          DO NOT CHANGE THIS COMMENT!
135  [
136  [
137
138  [
139  [ * DO NOT CHANGE THIS COMMENT!          << Start of removed code area >>          DO NOT CHANGE THIS COMMENT!
140  [
141  [
142  [
143  [
144  [ * DO NOT CHANGE THIS COMMENT!          << End of removed code area >>          DO NOT CHANGE THIS COMMENT!
145  [
146  [

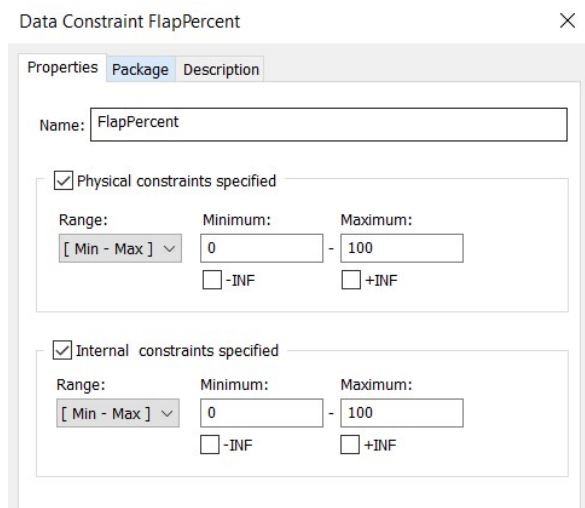
```

Data types Explained

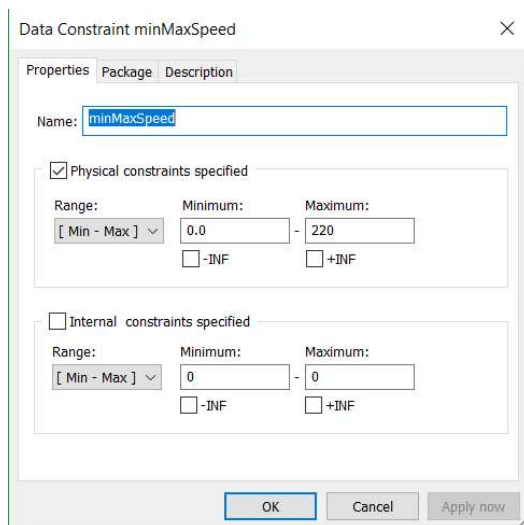


The following are the units for the application data types.

I use three Application Data types, FlapSettingPercent, speed and temperature. FlapPercent Constraint (can only go from 0 to 100).



Speed constraint, called minMaxSpeed (cars can go from 0 to 220 in my model).



Temperature Constraint (cabin temperature can go from 16 to 28 as in specification)

Application Port Interfaces Explained

For this assignment I have used AnalogueIO, AnaloguePercent, digitallo and TempAnaloguePort. All the ports except digitallo were Analogue, but I decided to keep them separate, regardless of being Analogue as each had measured a different unit value.

Mapping Sets Explained

I have three mapping sets, all were created for the application data types, and all are used throughout the project.

FlapMapSet








Application Data Type	Package	Implementation Data Type
flapSettingPercent	DataTypes	uint16

Assigned Component Types		
Component Type	Package	
AirMixIn	ComponentTypes	...
AirMixOut	ComponentTypes	✕
FlapController	ComponentTypes	📄

SpeedMapSet

Application Data Type	Package	Implementation Data Type
speed	DataTypes	sint16







Assigned Component Types

Component Type	Package	
 AirBlow	ComponentTypes	...
 AirBlowController	ComponentTypes	✕
 AirBlowOut	ComponentTypes	📄
 APStart	ComponentTypes	
 ECU	ComponentTypes	
 OutsideVariables	ComponentTypes	
 SeatController	ComponentTypes	

TempMapSet

Application Data Type	Package	Implementation Data Type
temperature	DataTypes	sint16

Assigned Component Types

Component Type	Package	
 CabinTempClosedLoopIn	ComponentTypes	...
 CabinTempClosedLoopOut	ComponentTypes	✕
 CabinTempController	ComponentTypes	📄
 Temp	ComponentTypes	
 TempOut	ComponentTypes	
 tempOutside	ComponentTypes	