A Simple Air Conditioning System

The HVAC systems is presumed to consist of three computational elements in addition to the mechanical and electrical elements. These three computational elements are: the HVAC controller proper, which controls the actuators and responds to the sensors; the human interaction system, which includes the buttons and display; and finally the HVAC control settings element, which manages the goal variables used by the other two elements. It is the control settings element which the focus of this project.

Abbreviations

HVAC-CON	the HVAC controller element
HID	the human input and display element
HCS	the HVAC control settings element

Assumptions:

Only one external event is processed at a time.

Asynchronous processing is assumed. All internal transitions which enable must have fired before any new external events will be processed.

Functional Specification:

There is ambiguity in what is to be done with the fan-up (fan-down) events regarding the manual-mode and the fan-speed-current variable. When in a non-manual mode does the event only cause a change to manual-mode or does it also cause a fan-speed-current change? I have decided that the initial event should not increment (decrement) the fan-speed-current but restore the previous manually entered value. This would produce a reminder to the operator before updating the fan-speed-current. A pair of additional variable is introduced to store the manually entered temperature and fan speed, manual-temp and manual-fan.

A further ambiguity is the factory initial value for manually entered goal-temp and fan-speed-current. I choose to use AUTO-F for the default manual-fan and MAXT for the default manual-temp.

In the following the provided specifications are revised and restated along with an implementation note indicating where implemented and a justification:

Power

- The HID produces power-press events which toggles the HCS between power-on and power-off states [in the root state the two transitions cannot be in conflict as the source states are orthogonal]
- While in the power-off states all events are ignored except the power-press event [the power-on and power-off states are orthogonal and the only transition out of that state is enabled by the power-press event]
- During power-off all variables and their values are maintained; in particular the goal-temp and fan-speed-current variables are maintained for use by the HID and the HVAC-CON [these variables are maintained as state-variables]

Input Mode

- There are three mutually exclusive input modes: manual, auto and turbo [implemented by orthogonally of the sub-substates of Mode state [it is an AND state]]
- The auto-mode may only be entered by an auto-mode event from the HID [the auto-on event only occurs on one transition]
- The turbo-mode may only be entered by a turbo-mode event from the HID [the turbo-on event only occurs on one transition]
- The manual-mode may be entered by either a fan-up or fan-down event from the HID [the two events are at the same state and cannot be simultaneously enabled]
- The factory default is auto-mode [when there is no history the initialization transitions lead to the mode.computed.auto state]
- The entering power-on results in returning to the most recently active mode. [implemented by a pair of history connectors, only the mode state needs to be preserved as the state-variables are preserved generally]

Temperature (goal-temp)

- The HID produces TempUp and TempDown events [all inputs are events from SimuLink]
- The HCS maintains a goal-temperature (goal-temp) which is used by the HVAC-CON and the HID. [the goal-temp is maintained as an output state-variable]
- The goal-temp is an integer (as are all associated variables and constants)
- The goal-temp must be between MINT and MAXT (MINT < goal-temp < MAXT)
- MINT and MAXT are constants. [these values are constants in the model]
- In either Mode.manual or Mode.turbo, the TempUp/TempDown event results in the manual-temp incrementing/decrementing by 1 degree C and the goal-temp being set to it. [there is no

possible conflict as the events appear on mutually exclusive transitions, this change occurs in the Temp_setting state]

- The turbo-on event results in the goal-temp being set to MAXT. [the turbo-on event occurs on only one transition]
- Upon entering the Mode.manual the goal-temp is restored to manual-temp [although the fan-up/fan-down events appear in two places they are guarded by conditions preventing their conflict]

Fan Speed (fan-speed-current)

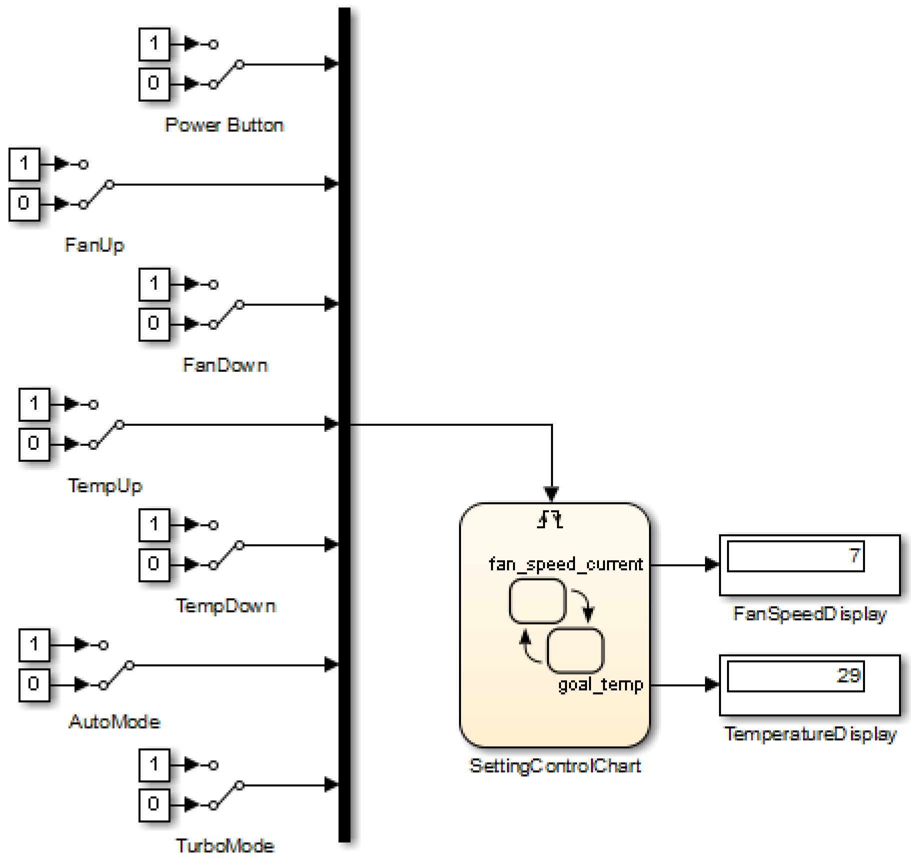
- The HID produces fan-up and fan-down events. [all inputs are events from SimuLink]
- The HCS maintains an air flow speed via a fan in a current fan speed (fan-speed-current) variable which is used by the HVAC-CON and the HID. [fan-speed-current is an output state-variable]
- The fan-speed-current is an integer (as are all associated variables and constants)
- The fan-speed-current must be between MINF and MAXF (MINF < fan-speed-current < MAXF) [implemented manually when the constants are set]
- MINF and MAXF are constants.
- In manual-mode the fan-up/fan-down event results in the fan-speed-current incrementing/decrementing by 1 unit [a guard condition only allowed once manual mode is enabled]
- The auto-on event results in the fan-speed-current being set to AUTO-F [the auto-on event only appears on one transition]
- AUTO-F is a constant between MINF and MAXF (MINF <= AUTO-F <= MAXF)
- The turbo-on event results in the fan-speed-current being set to MAXF [the auto-on event only appears on one transition and only one external events is admitted at a time]
- Upon entering the manual-mode the fan-speed-current is restored to manual-fan

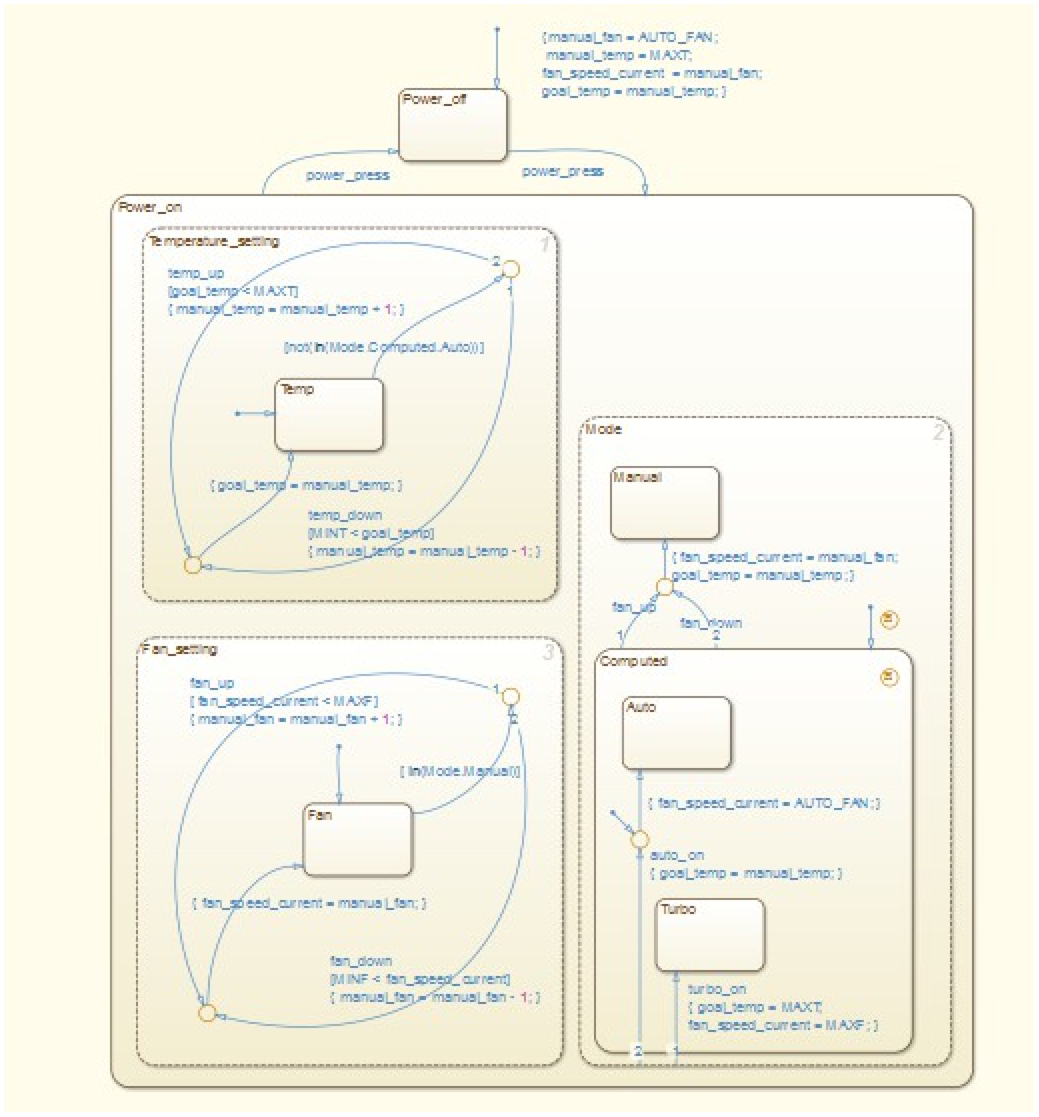
Design and Implementation Approach

There are many ways to approach the model for this system. It could be modeled as a flat FSM resulting in a model for which it may be easier to argue about properties not related to the functional specification. I have chosen to build a model that reflects the re-factored functional specifications as outlined above, this bias introduces an AND state over the fan, temperature and input.

The event generator is modeled with a manual switch and a pair of constants {0,1}. On either a rising or falling edge an event is generated. The edge detection events are mapped to integers which are then mapped to the set of indexed events. In the simulation it is difficult but not impossible to generate simultaneous events, this violates an assumption and is disallowed but only by convention. Similarly, it is possible to generate an external event before all internal events have been processed, this also violates an assumption and is disallowed, this asynchronous mode can be enforced by SimuLink.

The DisplayFan and DisplayTemp functions are imitated by a pair of Simulink Display objects. The goal-temp and fan-speed-current are maintained at all times even when power-off.




{manual_fan = AUTO_FAN;
manual_temp = MAXT;
fan_speed_current = manual_fan;
goal_temp = manual_temp; }
Power_off
power_press
power_press
Power_on
Temperature_setting
1
temp_up
[goal_temp < MAXT]
{ manual_temp = manual_temp + 1; }
2
1
[not(in(Mode.Computed.Auto))]
Temp
{ goal_temp = manual_temp; }
temp_down
[MINT < goal_temp]
{ manual_temp = manual_temp - 1; }
Mode
2
Manual
{ fan_speed_current = manual_fan;
goal_temp = manual_temp ;}
fan_up
fan_down
1
2
Computed
Auto
{ fan_speed_current = AUTO_FAN ;}
auto_on
{ goal_temp = manual_temp; }
Turbo
turbo_on
{ goal_temp = MAXT;
fan_speed_current = MAXF ; }
2
1
Fan_setting
3
fan_up
[fan_speed_current < MAXF]
{ manual_fan = manual_fan + 1; }
1
2
[in(Mode.Manual)]
Fan
{ fan_speed_current = manual_fan; }
fan_down
[MINF < fan_speed_current]
{ manual_fan = manual_fan - 1; }

It is sufficient to ensure robustness if it can be shown that there exist no conflicts or non-determinism in the states or transitions. This could be achieved by incorporating priority, or similar techniques but these make the model difficult to analyze. For this project model robustness is achieved by relying on simple direct arguments. All external events are handled as individual discrete event super-steps, exhausting all internal events before receiving a new external event and no time dependency (i.e. no timeout or schedule functions). Variable and transition conflict are prevented by reducing the number of transitions enabled by an event to one or providing guards resulting in this exclusivity.

In order to void event conflict, an event should only trigger one transition. If an event enables more than one transition there must be strong arguments that the two transitions can not be in conflict. In this model events do appear on more than one transition but they are exclusive as indicated in the following examples:

power-press event: enabled transitions have root-state scope in both cases but they originate from from mutually exclusive states so there is no conflict.

fan-up/fan-down events: transitions have conditions preventing them from being concurrently evaluated, there is not conflict.

In order to avoid a variable conflict, it is sufficient that it is not be possible to update a variable from more than one transition. This can be accomplished by only updating a variable on a single transition. It is also sufficient if the variable may be updated on a set of exclusive transitions, that is, the transitions emanate from the same state.