Elevator-Environment

Parts

The elevator environment consists of the following parts that are more precisely described in the corresponding data sheets.

3 Button-Arrays
3 Button-Light-Arrays
1 Door
1 Cabin

There are three button-arrays. One representing all up buttons and one representing all down buttons that are located on the floors for calling the cabin. The third button-array represents the buttons within the cabin to select the floor the cabin should move to.

The button-arrays are renamed as follows.

Variables

button-array	**PhyButtons**	**snsrButtons**	**a**	**b**
up buttons on floor	PhyUpButtons	snsrUpButtons	0	LAST_FLOOR - 1
down buttons on floor	PhyDownButtons	snsrDownButtons	1	LAST_FLOOR
floor buttons within cabin	PhyFloorButtons	snsrFloorButtons	0	LAST_FLOOR

Events

button-array	**USER_PRESSES_BUTTON**	**USER_RELEASES_BUTTON**
up buttons on floor	USER_PRESSES_UP_BUTTON	USER_RELEASES_UP_BUTTON
down buttons on floor	USER_PRESSES_DOWN_BUTTON	USER_RELEASES_DOWN_BUTTON
floor buttons within cabin	USER_PRESSES_FLOOR_BUTTON	USER_RELEASES_FLOOR_BUTTON

To each button a light is attached. Therefore three button-light-arrays are needed. One for the up buttons, one for the down buttons and one for the floor buttons.

The button-light-arrays are renamed as follows.

Variables

button-light-array	**PhyButtonLights**	**ctrlButtonLights**	**a**	**b**
lights for up buttons	PhyUpButtonLights	ctrlUpButtonLights	0	LAST_FLOOR - 1
lights for down buttons	PhyDownButtonLights	ctrlDownButtonLights	1	LAST_FLOOR
lights for floor buttons	PhyFloorButtonLights	ctrlFloorButtonLights	0	LAST_FLOOR

Events

button-light-array	**TURNS_ON_BUTTON_LIGHT**	**TURNS_OFF_BUTTON_LIGHT**
lights for up buttons	TURNS_ON_UP_BUTTON_LIGHT	TURNS_OFF_UP_BUTTON_LIGHT
lights for down buttons	TURNS_ON_DOWN_BUTTON_LIGHT	TURNS_OFF_DOWN_BUTTON_LIGHT
lights for floor buttons	TURNS_ON_FLOOR_BUTTON_LIGHT	TURNS_OFF_FLOOR_BUTTON_LIGHT

Summary of Interface Variables

The following variables of the environment are accessible for the controller.

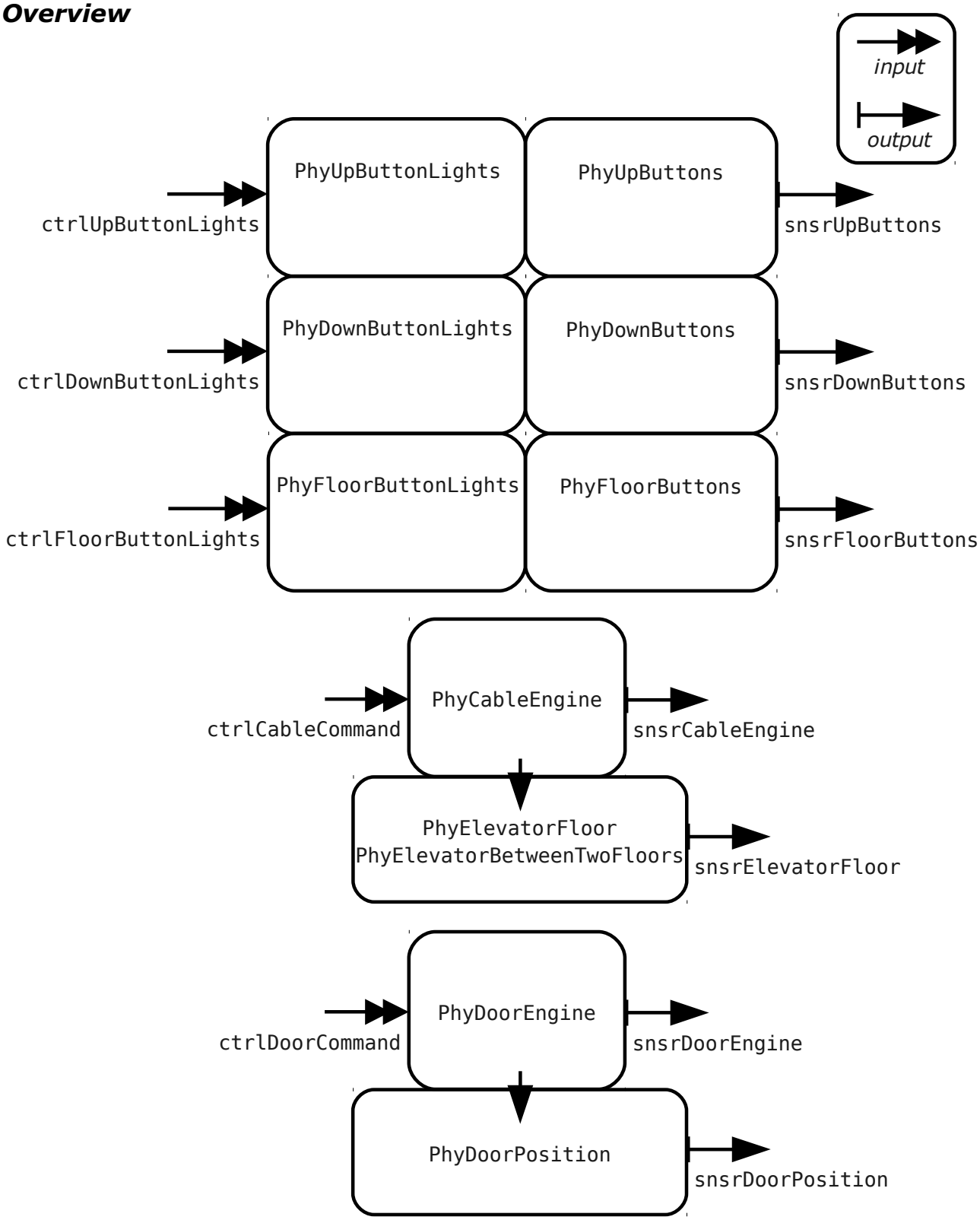
Inputs

```
ctrlUpButtonLights
ctrlDownButtonLights
ctrlFloorButtonLights
ctrlCableCommand
ctrlDoorCommand
```

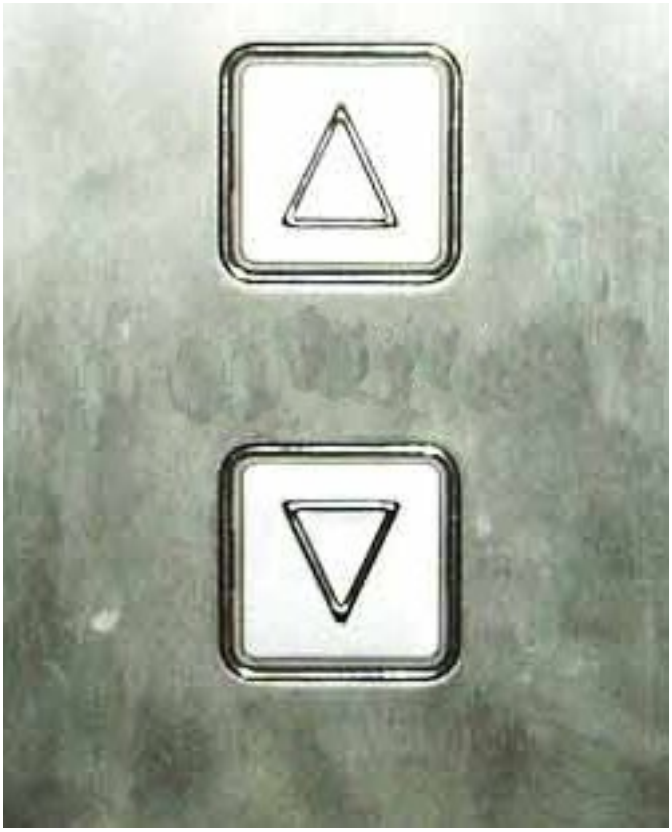
Outputs

```
snsrUpButtons
snsrDownButtons
snsrFloorButtons
snsrCableEngine
snsrElevatorFloor
snsrDoorEngine
snsrDoorPosition
```

Overview


input
output
PhyUpButtonLights
PhyUpButtons
ctrlUpButtonLights
snsrUpButtons
PhyDownButtonLights
PhyDownButtons
ctrlDownButtonLights
snsrDownButtons
PhyFloorButtonLights
PhyFloorButtons
ctrlFloorButtonLights
snsrFloorButtons
PhyCableEngine
ctrlCableCommand
snsrCableEngine
PhyElevatorFloor
PhyElevatorBetweenTwoFloors
snsrElevatorFloor
PhyDoorEngine
ctrlDoorCommand
snsrDoorEngine
PhyDoorPosition
snsrDoorPosition

Button-Array

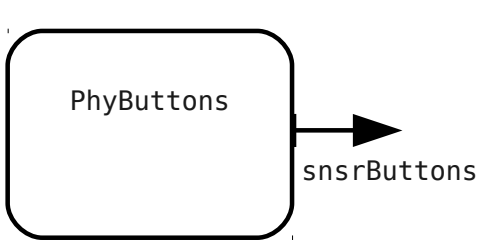


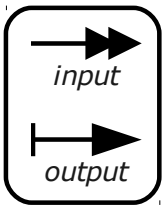
Short Description

The button-array consists of several buttons. The status of these buttons is represented by a function. A button can either be pressed or released. The status of the buttons can be read by an output function.

The array reaches from floor a to floor b, where $a,b \in \mathbb{N}$.
TRUE stands for pressed and FALSE stands for released.

Abstract Representation





Internal Variables	PhyButtons
Interface Variables	snsrButtons

Formal Representation

Variables

PhyButtons
snsrButtons

Invariants

inv1: PhyButtons ∈ a‥b ⟶ BOOL
inv2: snsrButtons ∈ a‥b ⟶ BOOL
inv3: snsrButtons = PhyButtons

Events

```
INITIALISATION ≙
  BEGIN
    act1: PhyButtons ≔ a‥b × {FALSE}
    act2: snsrButtons ≔ a‥b × {FALSE}
  END

USER_PRESSES_BUTTON ≙
  ANY
    floor
  WHERE
    grd1: floor ∈ a‥b
    grd2: PhyButtons(floor) = FALSE
  THEN
    act1: PhyButtons(floor) ≔ TRUE
    act2: snsrButtons(floor) ≔ TRUE
  END

USER_RELEASES_BUTTON ≙
  ANY
    floor
  WHERE
    grd1: floor ∈ a‥b
    grd2: PhyButtons(floor) = TRUE
  THEN
    act1: PhyButtons(floor) ≔ FALSE
    act2: snsrButtons(floor) ≔ FALSE
  END
```

Button-Light-Array

Short Description

The button-light-array consists of several lights integrated in buttons. The status of these lights is represented by a function. A light can either be on or off. The lights can be turned on and off by a input function.

The array reaches from floor a to floor b, where $a,b \in \mathbb{N}$.
TRUE stands for on and FALSE stands for off.

Abstract Representation

Internal Variables	PhyButtonLights
Interface Variables	ctrlButtonLights

Formal Representation

Variables

PhyButtonLights
ctrlButtonLights

Invariants

inv1: $PhyButtonLights \in a..b \rightarrow BOOL$
inv2: $ctrlButtonLights \in a..b \rightarrow BOOL$

Events

```
INITIALISATION ≙
  BEGIN
    act1: PhyButtonLights ≔ a‥b × {FALSE}
    act2: ctrlButtonLights ≔ a‥b × {FALSE}
  END

TURNS_ON_BUTTON_LIGHT ≙
  ANY
    floor
  WHERE
    grd1: floor ∈ a‥b
    grd2: ctrlButtonLights(floor) = TRUE
    grd3: PhyButtonLights(floor) = FALSE
  THEN
    act1: PhyButtonLights(floor) ≔ TRUE
  END

TURNS_OFF_BUTTON_LIGHT ≙
  ANY
    floor
  WHERE
    grd1: floor ∈ a‥b
    grd2: ctrlButtonLights(floor) = FALSE
    grd3: PhyButtonLights(floor) = TRUE
  THEN
    act1: PhyButtonLights(floor) ≔ FALSE
  END
```

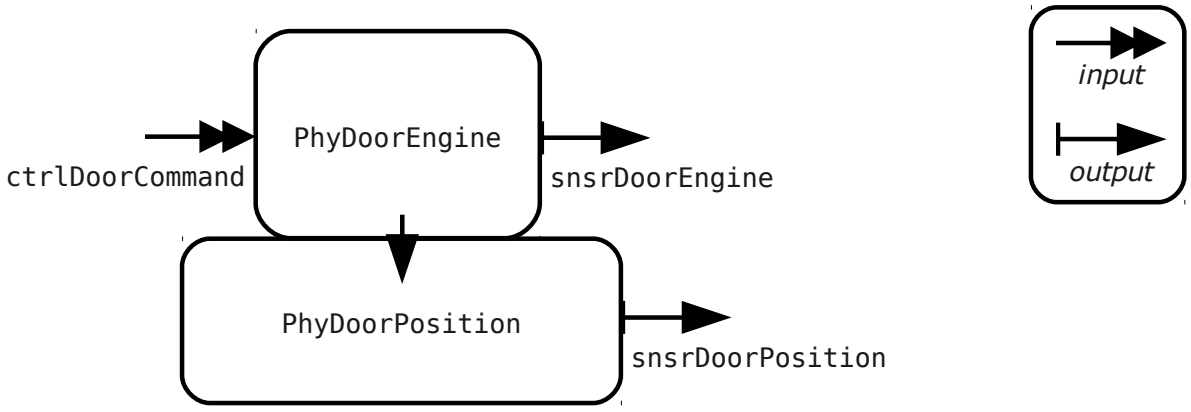
Door

Short Description

The door is connected to a door-engine by a transmission. The door-engine is an electric motor that can be operated in both direction. If the door-engine gets the command `DOOR_STOP` it stops and the door's position does not change. When it gets the command `DOOR_OPEN` it opens the door and when it gets the command `DOOR_CLOSE` it closes the door.

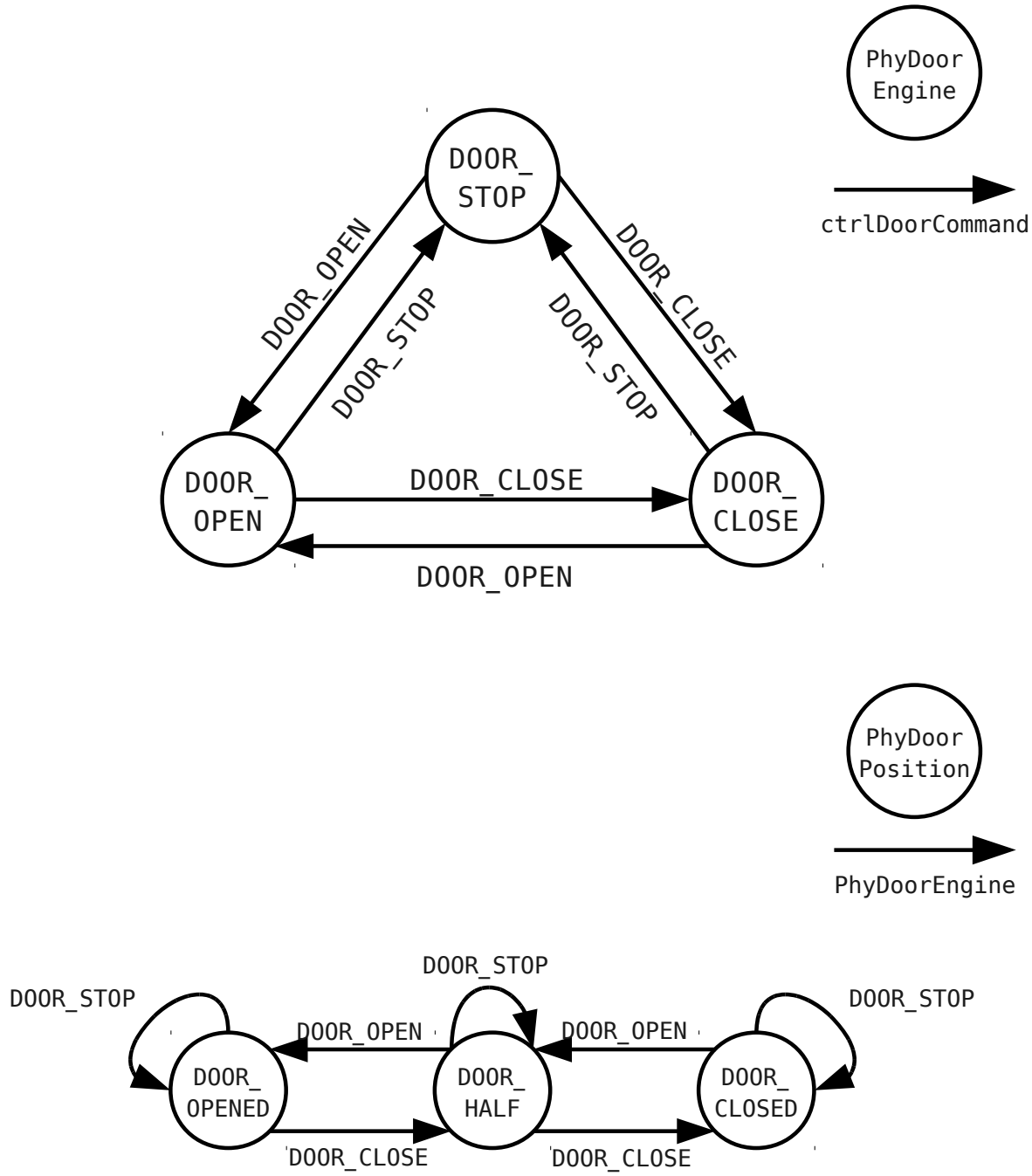
`snsrDoorPosition` reflects the position of the door and `snsrDoorEngine` reflects the engine's state.

Abstract Representation



Internal Variables	`PhyDoorEngine, PhyDoorPosition`
Interface Variables	`CtrlDoorCommand, snsrDoorEngine, snsrDoorPosition`

State-Chart


PhyDoor Engine
ctrlDoorCommand
DOOR_ STOP
DOOR_OPEN
DOOR_STOP
DOOR_STOP
DOOR_CLOSE
DOOR_ OPEN
DOOR_CLOSE
DOOR_ CLOSE
DOOR_OPEN
PhyDoor Position
PhyDoorEngine
DOOR_STOP
DOOR_STOP
DOOR_STOP
DOOR_OPEN
DOOR_OPEN
DOOR_ OPENED
DOOR_ HALF
DOOR_ CLOSED
DOOR_CLOSE
DOOR_CLOSE

Formal Representation

Sets

DOOR_COMMAND
DOOR_POSITION

Constants

DOOR_STOP
DOOR_OPEN
DOOR_CLOSE
DOOR_OPENED
DOOR_HALF
DOOR_CLOSED

Axioms

axm1: partition(DOOR_COMMAND, {DOOR_STOP}, {DOOR_OPEN}, {DOOR_CLOSE})
axm2: partition(DOOR_POSITION, {DOOR_OPENED}, {DOOR_HALF}, {DOOR_CLOSED})

Variables

PhyDoorEngine
PhyDoorPosition
ctrlDoorCommand
snsrDoorEngine
snsrDoorPosition

Invariants

inv1: PhyDoorEngine ∈ DOOR_ENGINE
inv2: PhyDoorPosition ∈ DOOR_POSITION
inv3: ctrlDoorCommand ∈ DOOR_ENGINE
inv4: snsrDoorEngine ∈ DOOR_ENGINE
inv5: snsrDoorEngine = PhyDoorEngine
inv6: snsrDoorPosition ∈ DOOR_POSITION
inv7: snsrDoorPosition = PhyDoorPosition

Events

```
INITIALISATION ≙
  BEGIN
    act1: PhyDoorEngine ≔ DOOR_STOP
    act2: PhyDoorPosition ≔ DOOR_CLOSED
    act3: ctrlDoorCommand ≔ DOOR_STOP
    act4: snsrDoorEngine ≔ DOOR_STOP
    act5: snsrDoorPosition ≔ DOOR_CLOSED
  END

DOOR_OPENS_WHEN_CLOSED ≙
  WHEN
    grd1: PhyDoorPosition = DOOR_CLOSED
    grd2: ctrlDoorCommand = DOOR_OPEN
  THEN
    act1: PhyDoorPosition ≔ DOOR_HALF
    act2: PhyDoorEngine ≔ DOOR_OPEN
    act3: snsrDoorPosition ≔ DOOR_HALF
    act4: snsrDoorEngine ≔ DOOR_OPEN
  END

DOOR_OPENS_WHEN_HALF ≙
  WHEN
    grd1: PhyDoorPosition = DOOR_HALF
    grd2: ctrlDoorCommand = DOOR_OPEN
  THEN
    act1: PhyDoorPosition ≔ DOOR_OPENED
    act2: PhyDoorEngine ≔ DOOR_OPEN
    act3: snsrDoorPosition ≔ DOOR_OPENED
    act4: snsrDoorEngine ≔ DOOR_OPEN
  END

DOOR_CLOSES_WHEN_OPENED ≙
  WHEN
    grd1: PhyDoorPosition = DOOR_OPENED
    grd2: ctrlDoorCommand = DOOR_CLOSE
  THEN
    act1: PhyDoorPosition ≔ DOOR_HALF
    act2: PhyDoorEngine ≔ DOOR_CLOSE
    act3: snsrDoorPosition ≔ DOOR_HALF
    act4: snsrDoorEngine ≔ DOOR_CLOSE
  END
```

```
DOOR_CLOSES_WHEN_HALF ≜
  WHEN
    grd1: PhyDoorPosition = DOOR_HALF
    grd2: ctrlDoorCommand = DOOR_CLOSE
  THEN
    act1: PhyDoorPosition ≔ DOOR_CLOSED
    act2: PhyDoorEngine ≔ DOOR_CLOSE
    act3: snsrDoorPosition ≔ DOOR_CLOSED
    act4: snsrDoorEngine ≔ DOOR_CLOSE
  END

DOOR_CLOSES_WHEN_CLOSED ≜
  WHEN
    grd1: PhyDoorPosition = DOOR_CLOSED
    grd2: ctrlDoorCommand = DOOR_CLOSE
    grd3: PhyDoorEngine ≠ DOOR_CLOSE
  THEN
    act1: PhyDoorEngine ≔ DOOR_CLOSE
    act2: snsrDoorEngine ≔ DOOR_CLOSE
  END

STOP_DOOR_ENGINE ≜
  WHEN
    grd1: PhyDoorEngine ≠ DOOR_STOP
    grd2: ctrlDoorCommand = DOOR_STOP
  THEN
    act1: PhyDoorEngine ≔ DOOR_STOP
    act2: snsrDoorEngine ≔ DOOR_STOP
  END
```

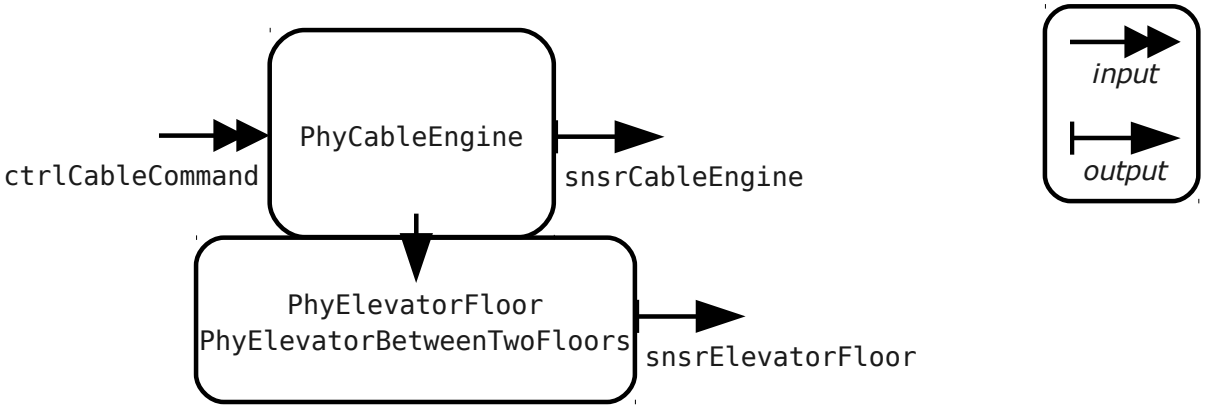
Cabin

Short Description

The cabin is connected to a cable-engine by a cable. The cable-engine is an electric motor that can be operated in both direction. If the cable-engine gets the command `CABLE_STOP` it stops and the cabin's position does not change. When it gets the command `CABLE_WIND` it winds the cable and the cabin moves upwards. When it gets the command `CABLE_UNWIND` it unwinds the cable and the cabin moves downwards.

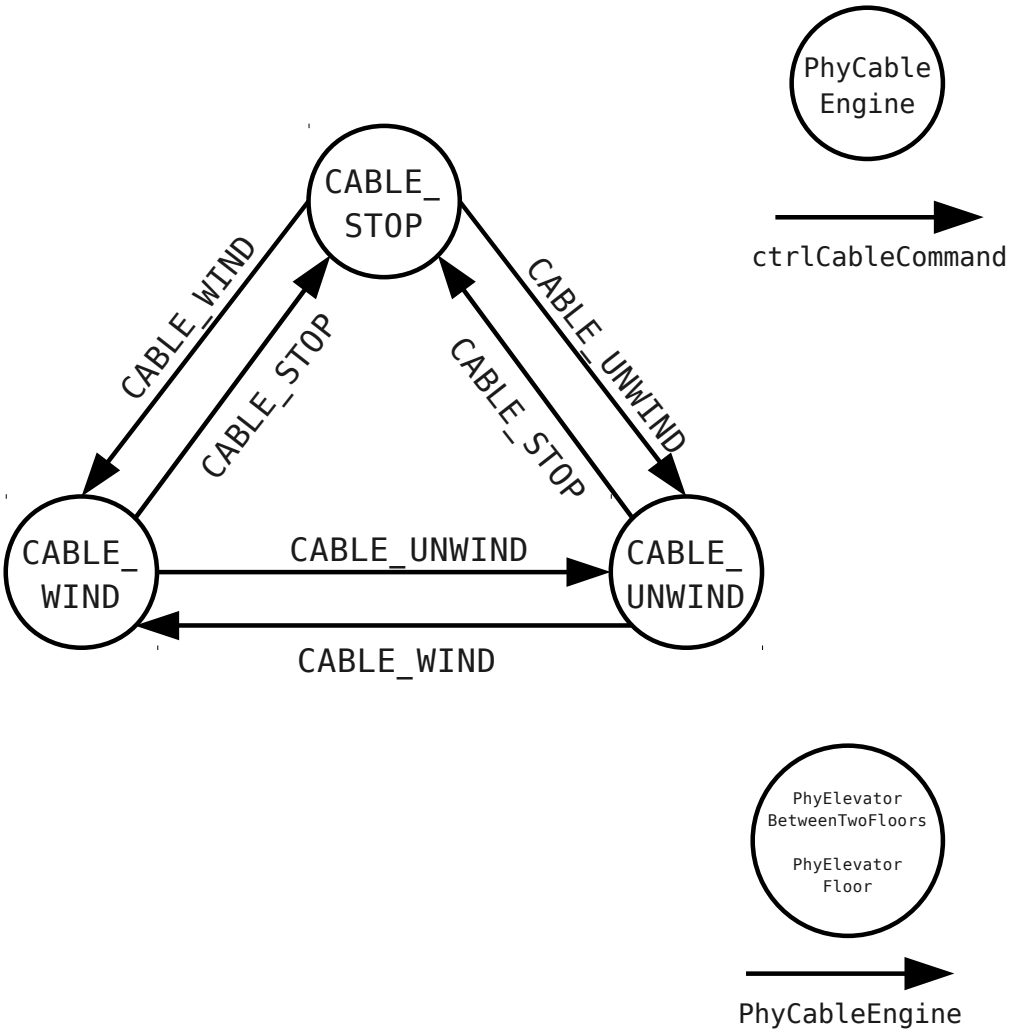
`snsrDoorPosition` reflects the position of the door and `snsrDoorEngine` reflects the engine's state.

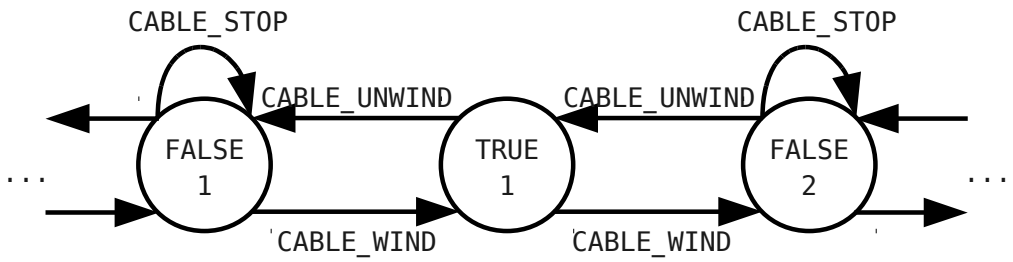
Abstract Representation



Internal Variables	`PhyCableEngine, PhyElevatorFloor, PhyElevatorBetweenTwoFloors`
Interface Variables	`CtrlCableCommand, snsrCableEngine, snsrElevatorFloor`

State-Chart


PhyCable
Engine
ctrlCableCommand
CABLE_
STOP
CABLE_WIND
CABLE_STOP
CABLE_STOP
CABLE_UNWIND
CABLE_
WIND
CABLE_UNWIND
CABLE_
UNWIND
CABLE_WIND
PhyElevator
BetweenTwoFloors
PhyElevator
Floor
PhyCableEngine


CABLE_STOP
CABLE_STOP
CABLE_UNWIND
CABLE_UNWIND
FALSE
1
TRUE
1
FALSE
2
...
...
CABLE_WIND
CABLE_WIND

Formal Representation

Sets

CABLE_COMMAND

Constants

CABLE_STOP
CABLE_WIND
CABLE_UNWIND
LAST_FLOOR

Axioms

axm1: partition(CABLE_COMMAND, {DOOR_STOP}, {DOOR_WIND}, {DOOR_UNWIND})
axm2: LAST_FLOOR ∈ ℕ1

Variables

PhyCableEngine
PhyElevatorFloor
PhyElevatorBetweenTwoFloors
ctrlCableCommand
snsrCableEngine
snsrElevatorFloor

Invariants

inv1: PhyCableEngine ∈ CABLE_ENGINE
inv2: PhyElevatorFloor ∈ 0‥LAST_FLOOR
inv3: PhyElevatorBetweenFloors ∈ BOOL
inv4: PhyElevatorBetweenFloors = TRUE ⇒ PhyElevatorFloor ≠ LAST_FLOOR
inv5: ctrlCableCommand ∈ CABLE_ENGINE
inv6: snsrCableEngine ∈ CABLE_ENGINE
inv7: snsrCableEngine = PhyCableEngine
inv8: snsrElevatorFloor ∈ −1‥LAST_FLOOR
inv9: PhyElevatorBetweenFloors = TRUE ⇒ snsrElevatorFloor = −1
inv10: PhyElevatorBetweenFloors = FALSE ⇒ snsrElevatorFloor = PhyElevatorFloor

Events

```
INITIALISATION ≙
  BEGIN
    act1: PhyCableEngine ≔ CABLE_STOP
    act2: PhyElevatorFloor ≔ 0
    act3: PhyElevatorBetweenTwoFloors ≔ FALSE
    act4: ctrlCableCommand ≔ CABLE_STOP
    act5: snsrCableEngine ≔ CABLE_STOP
    act6: snsrElevatorFloor ≔ 0
  END

ELEVATOR_LEAVES_FLOOR_UP ≙
  WHEN
    grd1: PhyElevatorBetweenTwoFloors = FALSE
    grd2: PhyElevatorFloor ≠ LAST_FLOOR
    grd3: ctrlCableCommand = CABLE_WIND
  THEN
    act1: PhyElevatorBetweenTwoFloors ≔ TRUE
    act2: PhyCableEngine ≔ CABLE_WIND
    act3: snsrElevatorFloor ≔ −1
    act4: snsrCableEngine ≔ CABLE_WIND
  END

ELEVATOR_REACHES_FLOOR_UP ≙
  WHEN
    grd1: PhyElevatorBetweenTwoFloors = TRUE
    grd2: ctrlCableCommand = CABLE_WIND
  THEN
    act1: PhyElevatorBetweenTwoFloors ≔ FALSE
    act2: PhyElevatorFloor ≔  PhyElevatorFloor + 1
    act3: PhyCableEngine ≔ CABLE_WIND
    act4: snsrElevatorFloor ≔ PhyElevatorFloor + 1
    act5: snsrCableEngine ≔ CABLE_WIND
  END
```

```
ELEVATOR_LEAVES_FLOOR_DOWN ≜
  WHEN
    grd1: PhyElevatorBetweenTwoFloors = FALSE
    grd2: PhyElevatorFloor ≠ 0
    grd3: ctrlCableCommand = CABLE_UNWIND
  THEN
    act1: PhyElevatorBetweenTwoFloors ≔ TRUE
    act2: PhyElevatorFloor ≔ PhyElevatorFloor − 1
    act3: PhyCableEngine ≔ CABLE_UNWIND
    act4: snsrElevatorFloor ≔ −1
    act5: snsrCableEngine ≔ CABLE_UNWIND
  END

ELEVATOR_REACHES_FLOOR_DOWN ≜
  WHEN
    grd1: PhyElevatorBetweenTwoFloors = TRUE
    grd2: ctrlCableCommand = CABLE_UNWIND
  THEN
    act1: PhyElevatorBetweenTwoFloors ≔ FALSE
    act2: PhyCableEngine ≔ CABLE_UNWIND
    act3: snsrElevatorFloor ≔ PhyElevatorFloor
    act4: snsrCableEngine ≔ CABLE_UNWIND
  END

STOP_CABLE_ENGINE ≜
  WHEN
    grd1: PhyCableEngine ≠ CABLE_STOP
    grd2: ctrlCableCommand = CABLE_STOP
  THEN
    act1: PhyCableEngine ≔ CABLE_STOP
    act2: snsrCableEngine ≔ CABLE_STOP
  END
```