**RE20 SW documentation**

# List of documentation updates

Version 1.0 – V.BRAULT/M.JACONELLI/ N. VITIELLO

Creation of SW documentation (SW version 1.0.0)

Table des matières

[1 List of documentation updates 1](#_Toc45816562)

[2 Mission 4](#_Toc45816563)

[2.1 Main requirements 4](#_Toc45816564)

[2.2 System description 4](#_Toc45816565)

[3 Function description 7](#_Toc45816566)

[3.1 Overview 7](#_Toc45816567)

[3.2 Respirator control 8](#_Toc45816568)

[3.3 User commands 9](#_Toc45816569)

[3.3.1 System buttons 9](#_Toc45816570)

[3.3.2 Setpoint buttons 10](#_Toc45816571)

[3.4 Inputs processing 11](#_Toc45816572)

[3.4.1 Analogic value 11](#_Toc45816573)

[3.4.2 Physical value 11](#_Toc45816574)

[3.4.3 Static adjustment 11](#_Toc45816575)

[3.4.4 Dynamic auto adjustment 11](#_Toc45816576)

[3.4.5 Filtering 12](#_Toc45816577)

[3.4.6 Diagnostic 12](#_Toc45816578)

[3.4.7 Volumetric air flow sensors 12](#_Toc45816579)

[3.5 Respirator control 13](#_Toc45816580)

[3.6 Cycle control 16](#_Toc45816581)

[3.6.1 Concept 16](#_Toc45816582)

[3.6.2 Library overview 17](#_Toc45816583)

[3.6.3 State Manager 17](#_Toc45816584)

[3.6.4 Pressure target 18](#_Toc45816585)

[3.6.5 Pressure processing 21](#_Toc45816586)

[3.6.6 Blower speed target 26](#_Toc45816587)

[3.6.7 Pressure valve command 28](#_Toc45816588)

[3.7 Air Volume processing 34](#_Toc45816589)

[3.8 Warning 35](#_Toc45816590)

[3.9 Output processing 39](#_Toc45816591)

[3.9.1 Blower motor speed control 39](#_Toc45816592)

[3.9.2 Blower and patient valve control 39](#_Toc45816593)

[3.9.3 Actuator test mode 40](#_Toc45816594)

[3.10 LCD display 41](#_Toc45816595)

# Mission

## Main requirements

This SW was developed during the COVID period in order to offer a low-cost alternative in case of new epidemic. Based on technical exchanges with doctors and specialists, the initial requirements were defined in order to ensure the control of:

* Peak pressure
* Plateau pressure
* End-expiratory pressure
* Frequency

Finally, we committed to provide a pressure control ventilation in which the user can define the inhalation pressure, the exhalation pressure, the frequency. The ratio I:E (Inspiration / Expiration) is enforced by the system to 1:3 in the SW version 1.0.0. Considering the last technical requirements, 2x buttons are now available and could be used for this purpose.

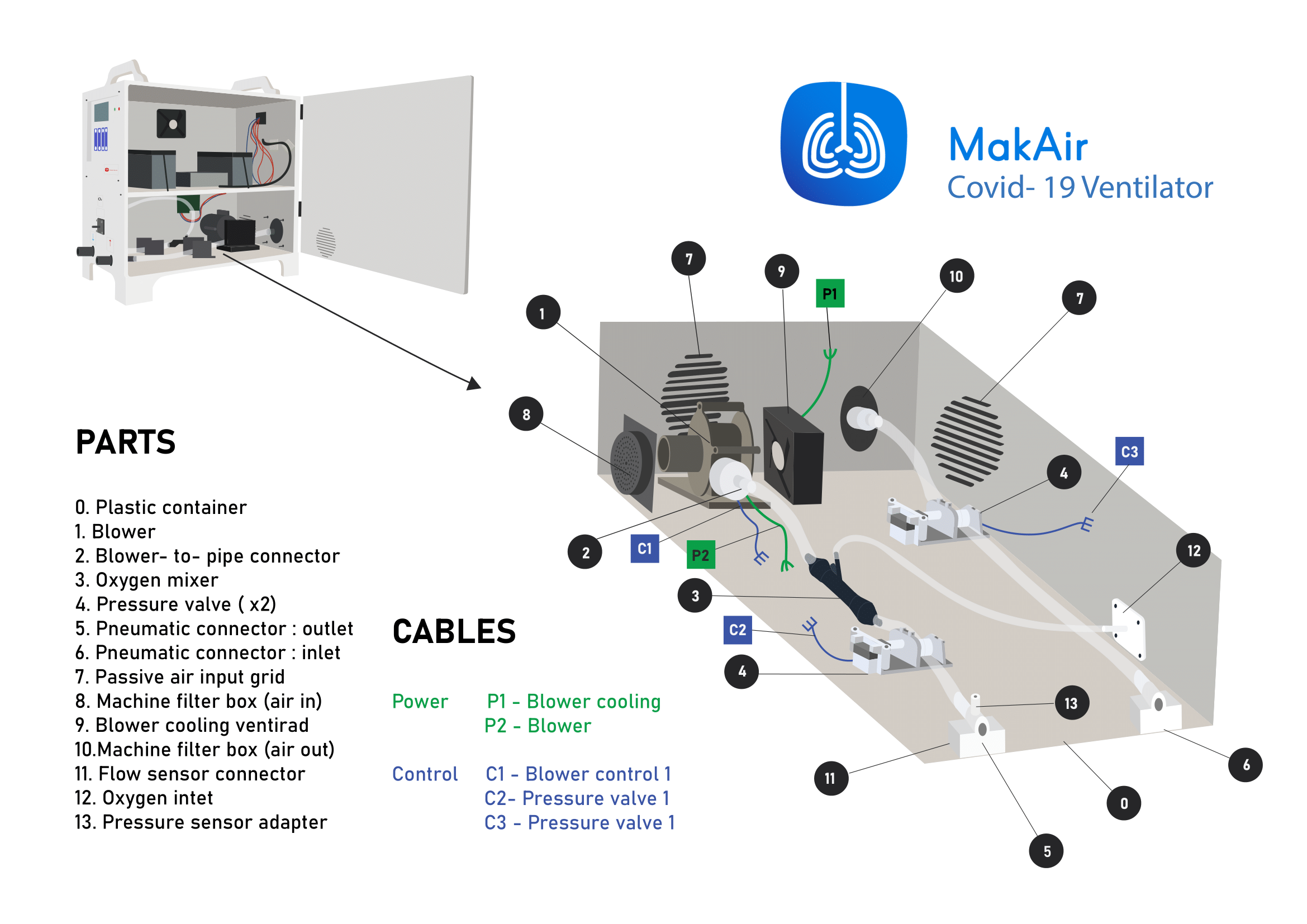


The expected control performance is +/- 15% on the pressure levels & +/-10% on the frequency. In addition, it was requested to give an air-flow information with an accuracy of +/-15%.

## System description

Mechanically-speaking, the minimum air-system is composed of the following sub-components that are plugged together:

* Air pump (called "Blower")
* Air pump casing fit (called "Blower Holder")
* Valve system (called "Pressure Valve")
* Oxygen Mixer valve (called "Oxygen Mixer")
* Air filter casing (patient variant) (called "Patient Filter Box")
* Air filter casing (machine variant) (called "Machine Filter Box"; intake + exhaust)
* Connectors (called "Pneumatic Connectors")
* Fan support (called "Fan Holder")



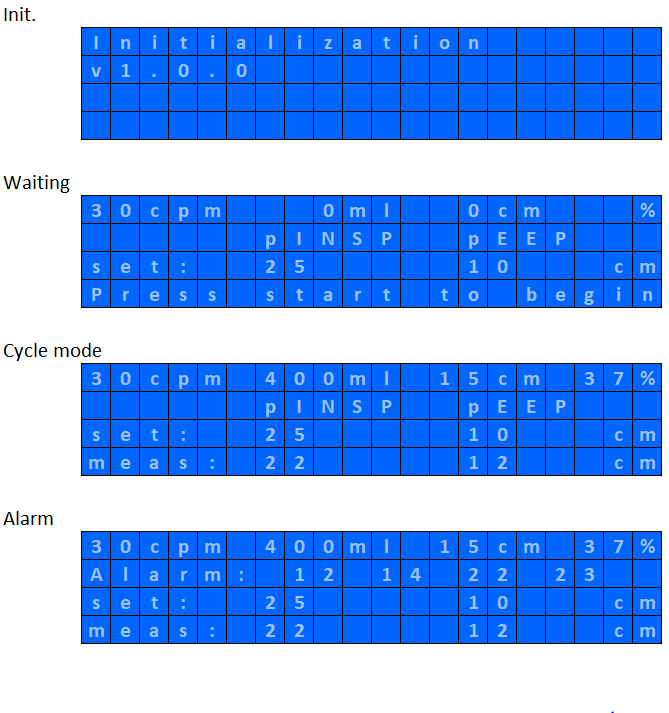
Two valve systems are installed to control respectively the inspiration and the expiration. Each one is based on a servomotor mechanically connected to a cam disc which squeezes a flexible tube. Each servomotor expects a rotary angle setpoint.

The blower can be considered as an electrical compressor and expects a speed setpoint. The driver ensures then the speed closed-loop control. The speed range allowed in the SW version 1.0.0 is [20 000;55 000] RPM.

In our version, two flowmeters mounted respectively on the intake or exhaust circuit are available. They are composed of a mechanical restriction, one differential pressure sensor and one static pressure sensor. The flow is computed from the pressure drop and the pressure upstream the restriction.

Finally, the LCD display gives back the main pieces of information to the user such as:

* the required frequency
* the required inspiration pressure
* the required exhalation pressure
* the end-inspiratory pressure
* the end-expiratory pressure (PEEP)
* the tidal volume (Vt)
* the instantaneous pressure
* the alarms



The user can define pressure (x2) and frequency setpoints. 2x buttons (+) & (-) are available per setpoint. To start the cycles, it is necessary to press the button start. To stop them, you need to press twice the stop button.

Buzzer and lights are available for the alarms but not used in this SW version.

Finally, a service button is available inside the respirator to activate a future “end of line” test mode.

# Function description

In this chapter, we provide a description of the Simulink model “NucleoRespirator.slx” used for coding the SW version 1.0.0. This model is the assembly of protected libraries dedicated to specific sub-functions. Please find the list below:



For more details, please refer to the following chapters.

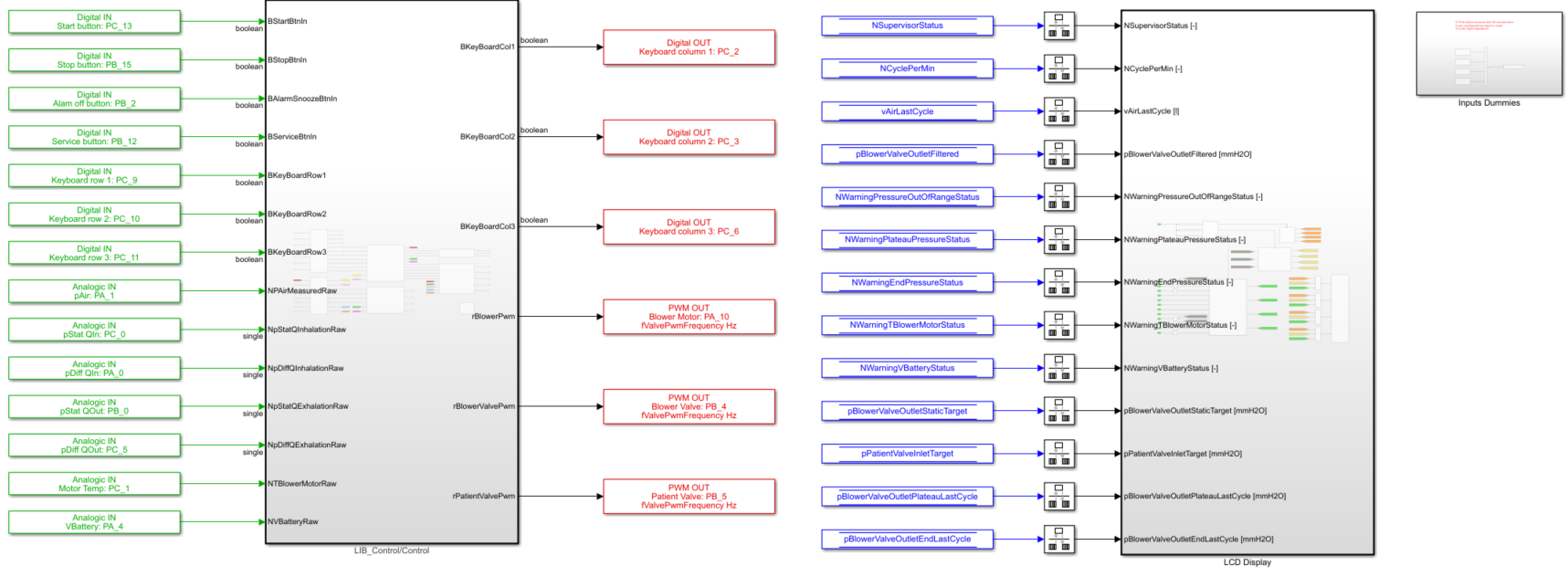
## Overview

The highest level is composed of 3x submodules:

* “Control”: this block ensures the respirator control
* “LCD Display”: this block ensures the writing command to the LCD display
* “Inputs Dummies”: this block ensures that all unused pins are configured as inputs in order to get a high impedance

The main inputs (buttons, pressure signals, …) and outputs (valve commands, blower speed command, LCD, …) of the applicative layer are defined at this level.

*NucleoRespirator*

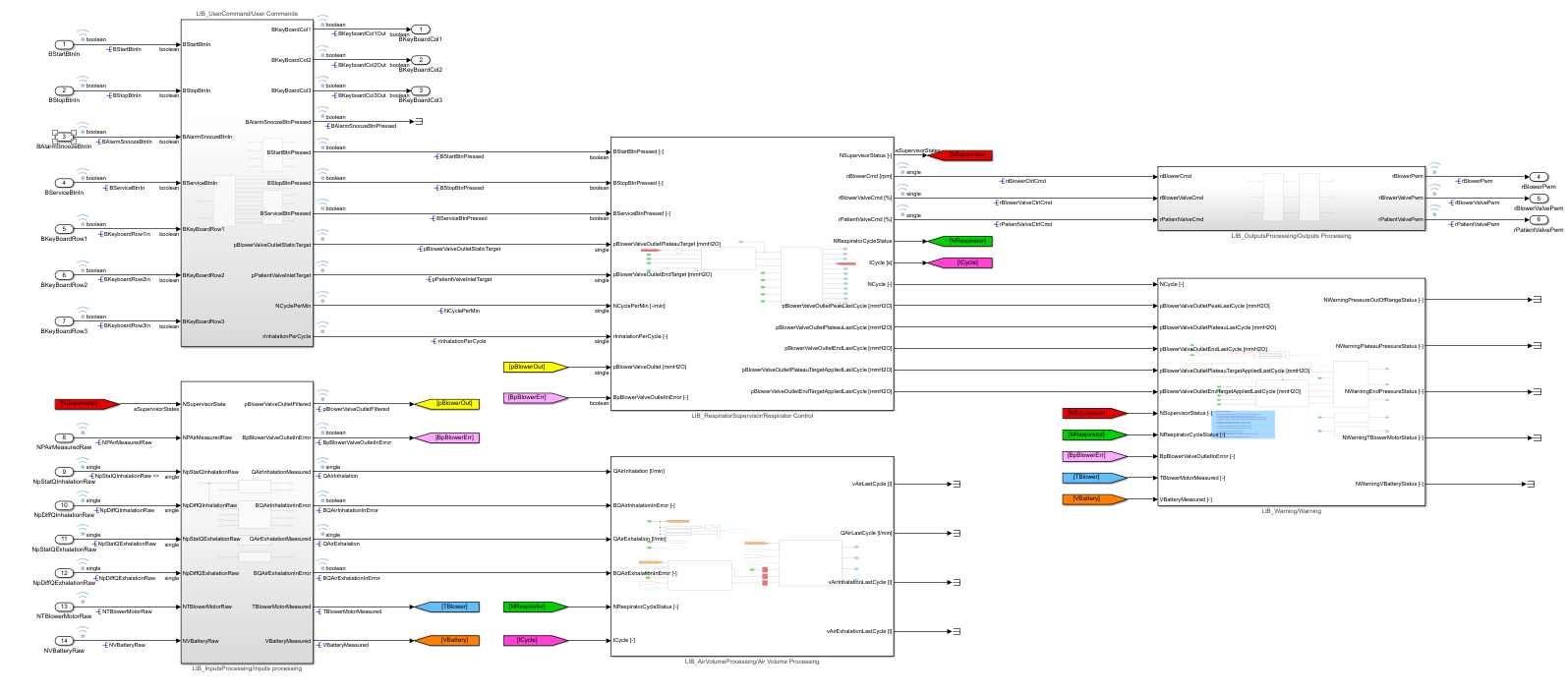


## Respirator control

The library control is composed of 6x sub-modules:

* “User Commands”: this block ensures the interface with the buttons and the computation of the setpoints.
* “Inputs processing”: this block ensures the acquisition and the linearization of the pressure sensors (5x), the blower temperature and the battery voltage. An offset learning is implemented for the pressure sensors.
* “Respirator control”: this block ensures the high-level control of the respirator and computes the command of the valves and of the blower.
* “Outputs Processing”: this block computes the final setpoints. A bench mode is available.
* “Volume Processing”: this block computes the tidal volume.
* “Warning”: this block detects and confirms the failures.

*NucleoRespirator/Control*



## User commands

User commands block makes the acquisition of user buttons and computes respiratory cycle setpoints.

There are two kinds of button:

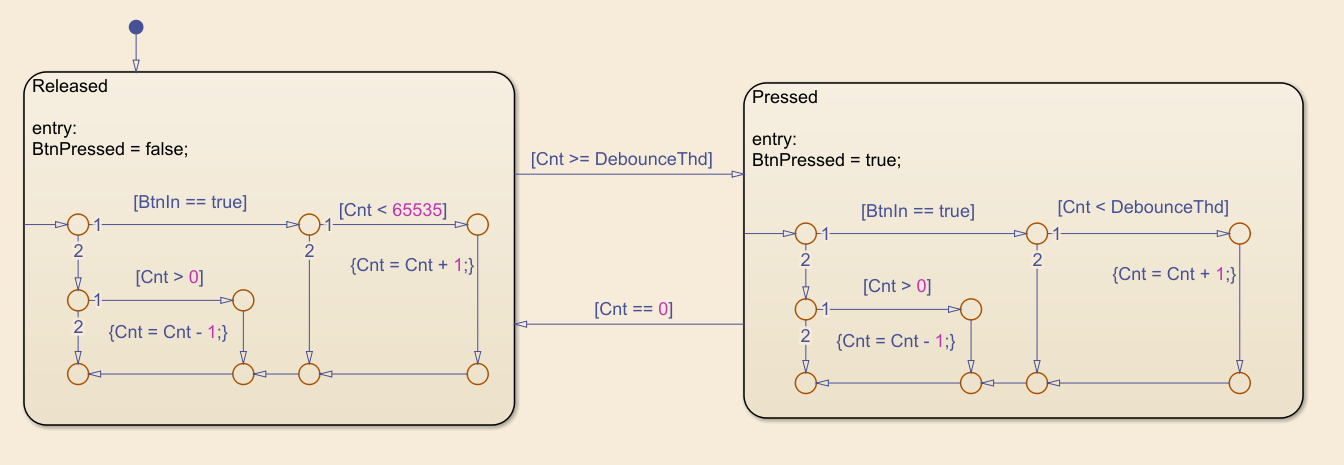
* System buttons which have a dedicated digital input
* Setpoint buttons which are multiplexed through a matrix in order to save I/O of the microcontroller.

### System buttons

Digital IOs “BXXXBtnIn”, where “XXX” is the button’s name, are acquired and filtered by a debounce timer to give “BXXXBtnPressed” states. When a button is pressed its value is true otherwise its state is false.

Debounce timer is managed in finite state machine.

*NucleoRespirator/Control/User Commands/System Commands/Button Processing/Push button debouce FSM*



In the “Released” state, when the digital input is true, a counter is incremented up to the debounce threshold or up to the maximum value. When digital input is false, counter is decremented down to 0. When the counter is greater or equal to the debounce threshold, button goes to “Pressed” state and the output is true. When digital input is false, debounce counter is decremented down to 0, when counter reach 0 button goes to “Released” state and the output is false

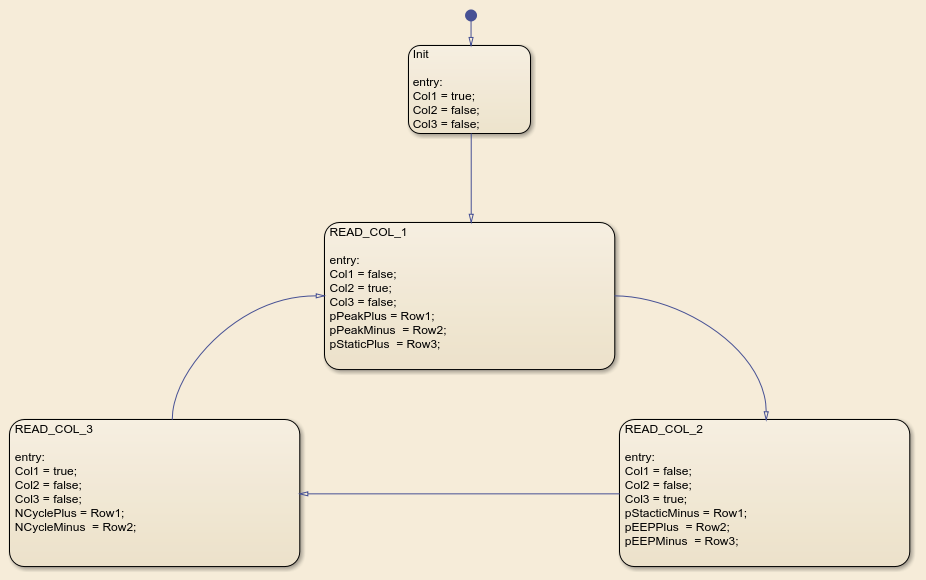
### Setpoint buttons

Setpoints buttons are multiplexed through a 3x3 matrix where columns are microcontroller digital outputs and rows are microcontroller digital inputs. Then for each column selected by the microcontroller output we read 3 rows states. That way we can address up to 9 (3x3) buttons using only 6 (3+3) microcontroller IOs.

|  |  |  |  |
| --- | --- | --- | --- |
|  | Column 1 | Column 2 | Column 3 |
| Row 1 | Pcrete plus | Pplateau minus | nb/min plus |
| Row 2 | Pcrete minus | PEP plus | nb/min minus |
| Row 3 | Pplateau plus | PEP minus | Not used |

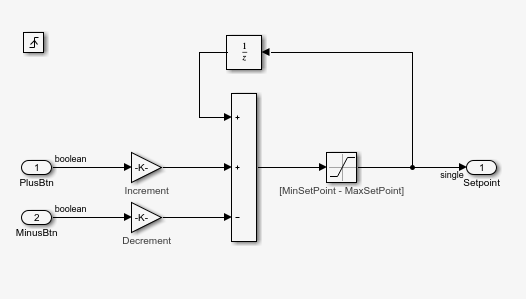
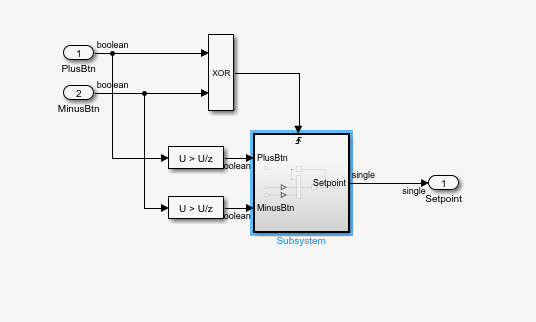
Buttons matrix is managed by a finite state machine, sweeping matrix columns over and over.

*NucleoRespirator/Control/User Commands/Keyboard Matrix/Keyboard Matrix Mangement FSM*



Buttons states are then filtered using the same debouncing strategy as system buttons.

Finally, respiratory cycle setpoints are computed based on buttons states rising edge events. “Unit delay” block is set with an initial output corresponding to the default value of the considered setpoint and then according to the button pressed, setpoint is incremented or decremented. Thanks to “XOR” block, case where both plus and minus buttons are simultaneously pressed is ignored.



## Inputs processing

Inputs processing block makes the acquisition, linearization, adjustment, filtering and diagnostic of all sensors fitted in the respirator.

### Analogic value

The “Analog Input” block provided with the Simulink hardware support package for Nucleo board, output a floating-point value between 0 and 1 where 0 is the ground level (0V) and 1 is the VIN level (3.3V). Value given by the “Analog Input” block is then multiplied by 3300 (VAnalogInputSpan) to give sensor analogic value in millivolt.

### Physical value

Measurement physical value is computed based on the analogic value. The “Input Scaling” block mask has four parameters:

* Electrical range min (): Analogic value giving the lower boundary of measurement range
* Electrical range max (): Analogic value giving the upper boundary of measurement range
* Physical value min (): Lower boundary of measurement range
* Physical value max (): Upper boundary of measurement range

With these four parameters, block initialization function computes linearization gain () and offset () using the relation below:

and

At runtime, physical value is computed using relation below:

*Remark: Blower motor temperature measurement does not use the strategy described above. Analogic value is used as input of a lookup table giving temperature values for a range of analogic value.*

### Static adjustment

A static offset may be applied on physical measurement. That feature is currently not used (in 1.0.0) and all offset are set to zero.

*Remark: Blower motor temperature measurement does not have any static offset.*

### Dynamic auto adjustment

For pressure measurements only, when respirator main state is “Init”, an automatic offset is computed in order to set measurement value to zero.

Dynamic offset is integrated, at each time step, taking the half of the difference between physical measurement and zero. Final offset is saturated between two min and max thresholds. If one of those thresholds is hit or if the difference between current and previous offset values is greater than a third threshold, offset is declared wrong and measurement is then declared wrong (In Error).

### Filtering

Physical value, after static and dynamic adjustments, is filtered by a first order low pass filter in order to remove noise.

### Diagnostic

Measurement is declared wrong (In Error), when its analogic value is outside of a valid range for a certain amount of time. When analogic value is greater than threshold max or smaller than threshold min, a counter is incremented by 25% at each computation step. When analogic value comes back into the valid range, counter is decreased by 10%. When the counter reaches 100%, measurement is declared in error.

*Remark: Blower motor temperature measurement and battery voltage measurement do not have any diagnostic.*

### Volumetric air flow sensors

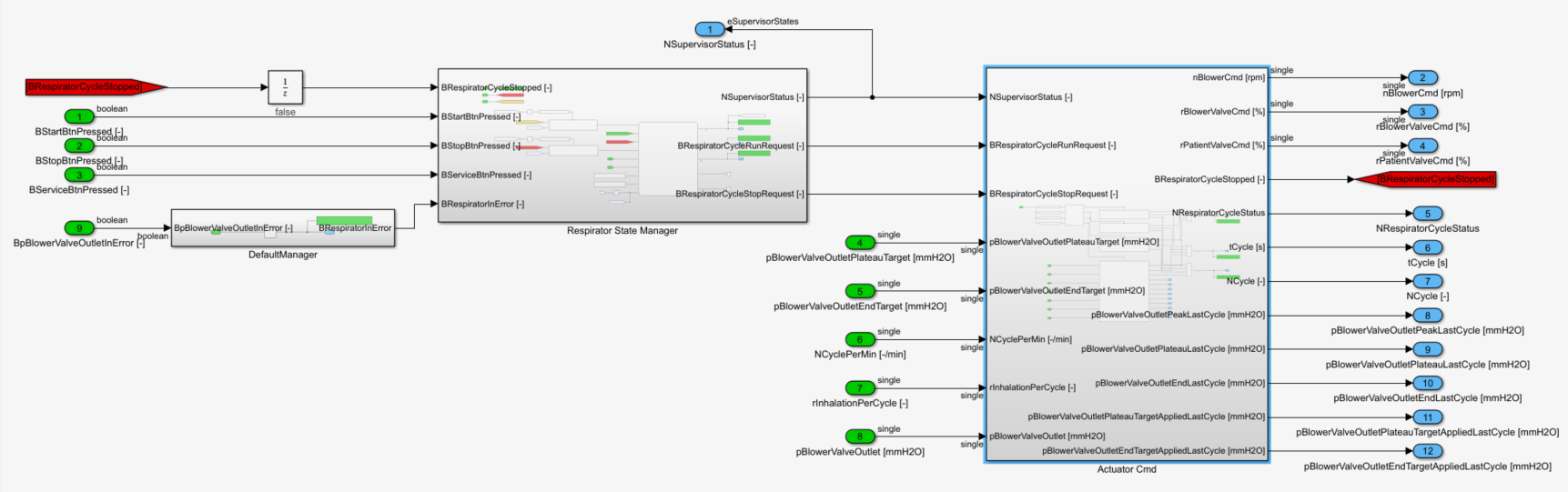
Inhalation and exhalation air flow measurement are computed based on static and differential pressure measurements using the formula below

Where ,  ,  ,  α et β are constant values that may be different between inhalation and exhalation flows.

## Respirator control

Respirator control is composed of 3x modules to manage the defaults, the respirator state and the associated commands.

*NucleoRespirator/Control/Respirator Control*

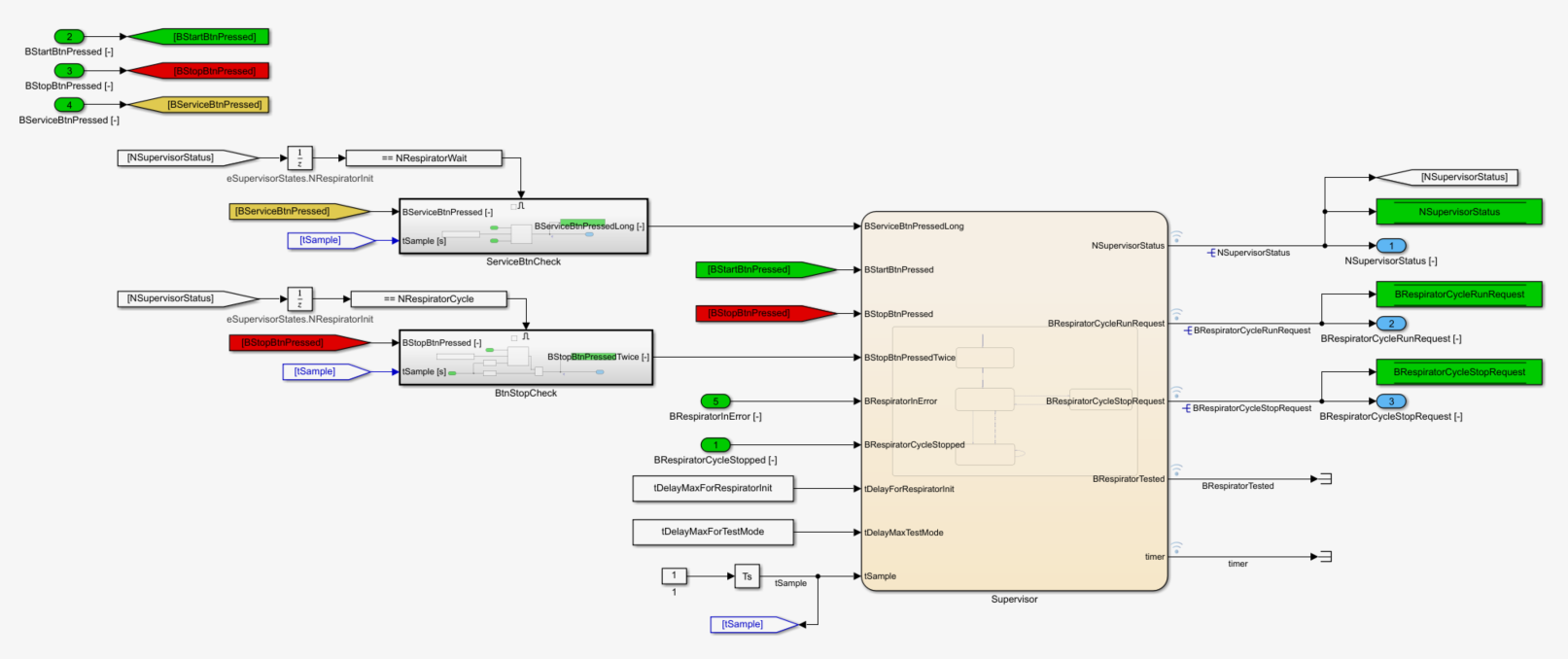


A supervisor is implemented in order to manage 4 modes:

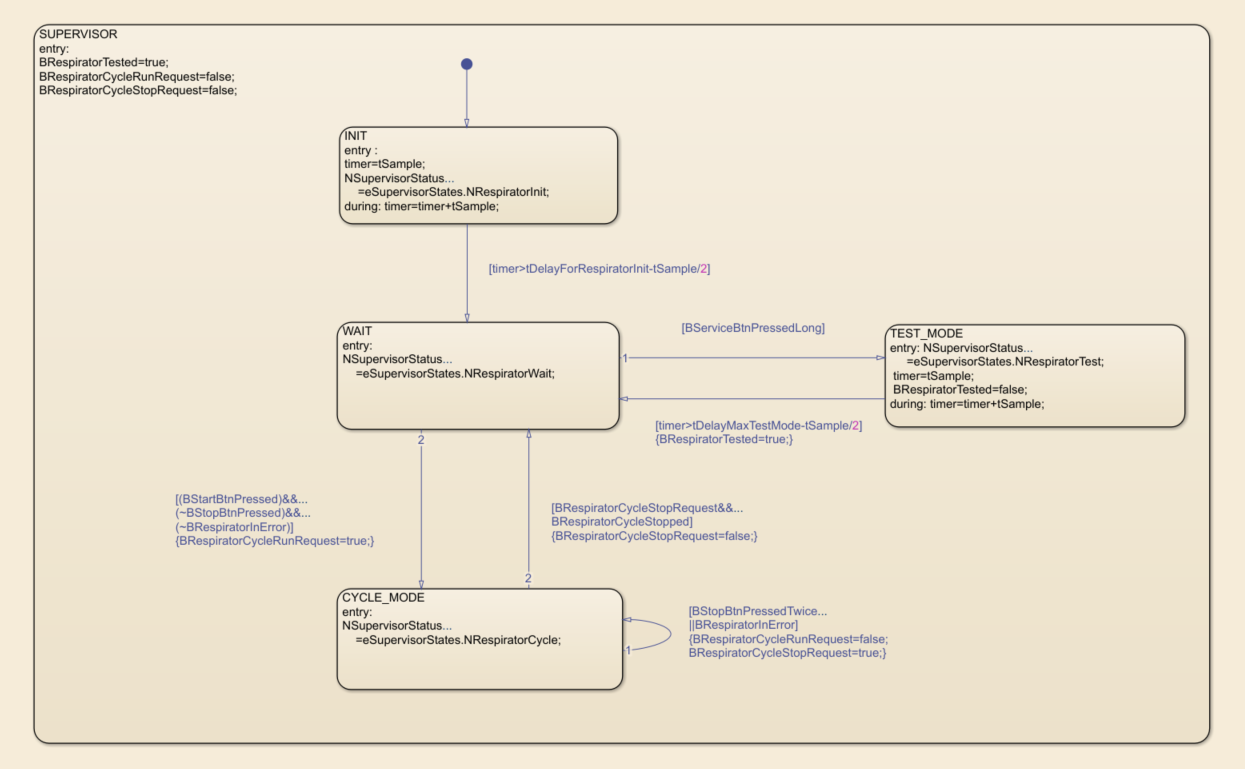
* “Init”: dedicated to sensor init
* “Wait”: right after the “init” mode or triggered by a double press of the button “stop”
* “Test”: triggered by a long press of the service button
* “Cycle”: triggered by a single press of the button “play”

The “init” mode lasts during tDelayMaxForRespiratorInit. The “test” mode was not implemented in SW version 1.0.0. A simple timer is integrated for validation purpose. Once the button play is triggered, the supervisor requests the launch of the ventilation. The ventilator stops the respiratory cycles in case of stop request or failure.

*NucleoRespirator/Control/Respirator Control/Respirator State Manager*

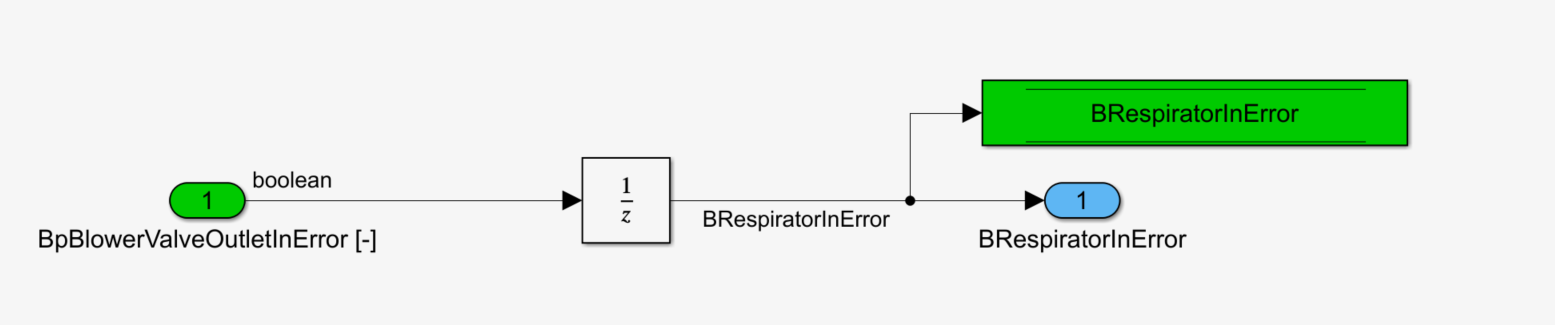


*NucleoRespirator/Control/Respirator Control/Respirator State Manager/Supervisor*



In case of failure on pBlowerValveOutlet, the respirator is considered in error and is locked in waiting mode. This sensor is the only one used for control purpose.

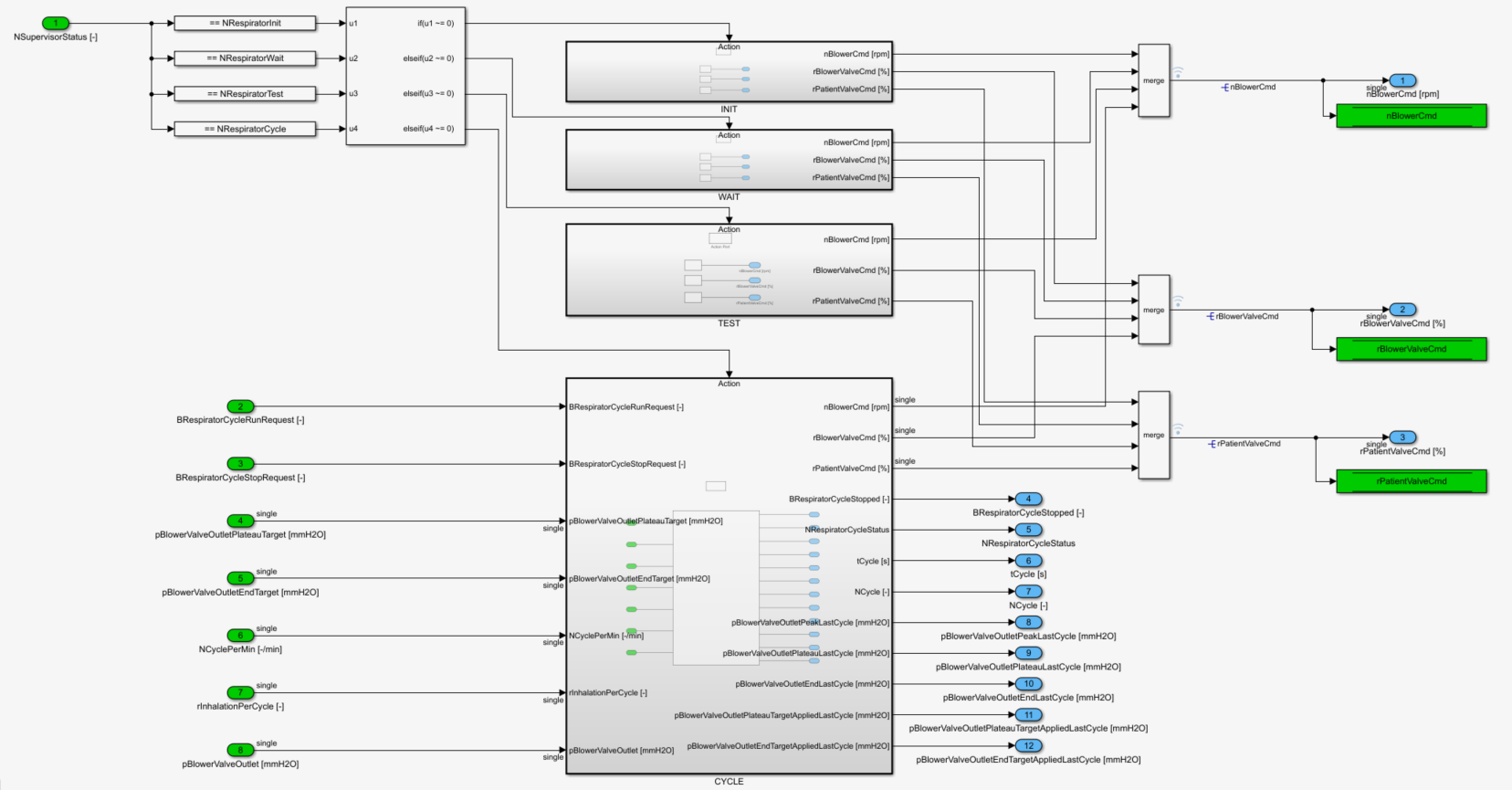
*NucleoRespirator/Control/Respirator Control/DefaultManager*



Regarding the commands, we enforce the following setpoints:

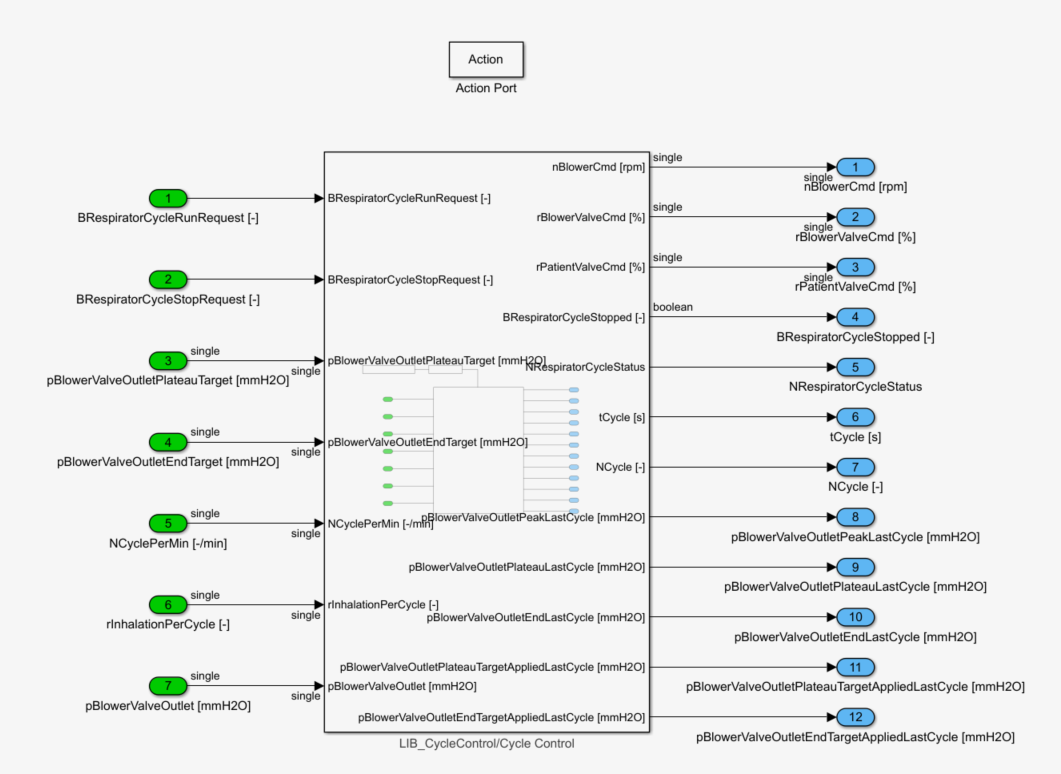
|  |  |  |  |
| --- | --- | --- | --- |
|  | **Blower valve (%)** | **Patient valve (%)** | **Blower speed (RPM)** |
| ***INIT*** | 100% | 100% | 0 RPM |
| ***WAIT*** | 0% | 100% | 0 RPM |
| ***TEST*** | 50% | 50% | 0 RPM |
| ***CYCLE*** | *Refer to the library “Cycle Control”* | | |

*NucleoRespirator/Control/Respirator Control/Actuator Cmd*



In case of cycles, the commands are defined in the library “Cycle Control”.

*NucleoRespirator/Control/Respirator Control/Actuator Cmd/CYCLE*



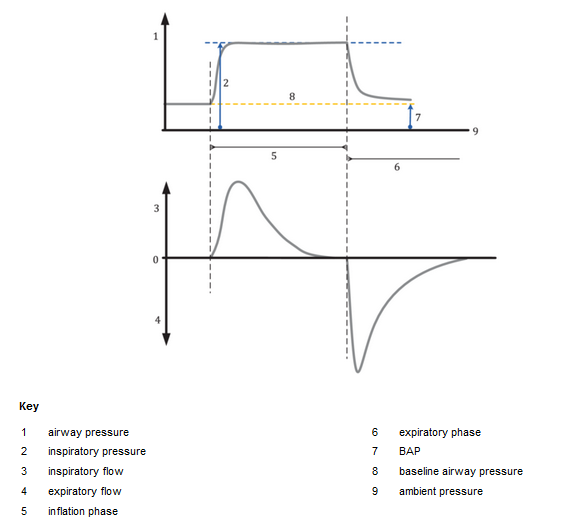
## Cycle control

### Concept

The respirator relies on a pressure-control mode. This means that it must reach and control a pressure level specified by the user during the inhalation (pBlowerValveOutletPlateauTarget) and another one during the exhalation (pBlowerValveOutletEndTarget).

*Remark: The naming is the one used for the initial concept developed by Makair in order to control the peak, plateau and end-expiratory pressure. In the literature, the plateau pressure is the airway pressure during an inspiratory pause when the flow at the patient-connection port is zero. In certain conditions, it gives a good indication of the end-inspiratory average alveolar pressure. Nevertheless, with a pressure-control inflation, it is not guaranteed to achieve a constant pressure level and no airway flow. Therefore, no valid measurement can be made. In a next release, it would be appropriate to update the naming.*

Please find below the behavior expected:



Lungs are characterized by a resistance and a compliance. The dynamic of the system may vary in a large range for an adult depending on the pathology.

|  |  |  |
| --- | --- | --- |
| ***Adult*** | R (cmH2O/L/sec) | C (mL/cmH2O) |
| Normal | 5 | 70 |
| Resistive & obstructive | 15 | 100 |
| Resistive & passive | 10 | 30 |

As an indication, please find a few standard ranges:

|  |  |  |
| --- | --- | --- |
|  | R (cmH2O/L/sec) | C (mL/cmH2O) |
| Adult | 4-6 | 70-100 |
| Child | 20-30 | 10-40 |
| Baby | 30-50 | 3-5 |

Based on this, we implemented a pressure controller based on:

* A learning function of the blower speed in order to achieve a given response time

Here, we target to standardize the dynamic of the system despite the various range of R & C.

* 1x pressure PID controller applied on the blower valve command during inhalation
* 1x pressure PID controller applied on the patient valve command during exhalation

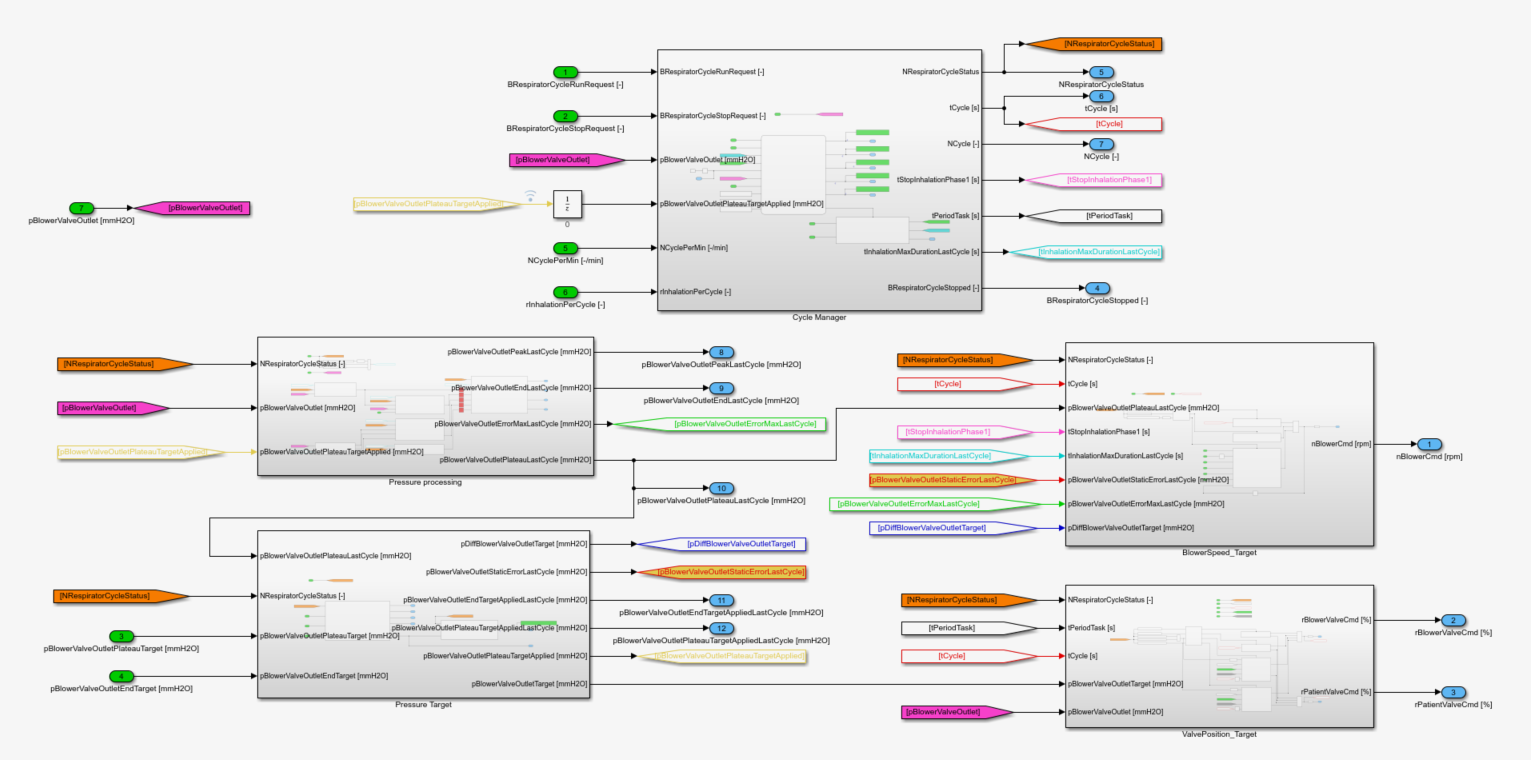
### Library overview

In SW version 1.0.0, the library “Cycle control” is composed of 5 sub-modules:

* “Cycle manager”: state manager monitoring each phase of the respiratory cycle
* “Pressure Target”: computation of the pressure targets
* ‘Pressure processing”: processing of pBlowerValveOutlet
* “BlowerSpeed\_Target”: computation of the blower speed target
* “ValvePosition\_Target”: pressure PID controller

The function is active in case of actuator test mode deactivation.

*NucleoRespirator/Control/Respirator Control/Actuator Cmd/CYCLE/Cycle Control/Cycle Control Wrapper/Cycle pController*

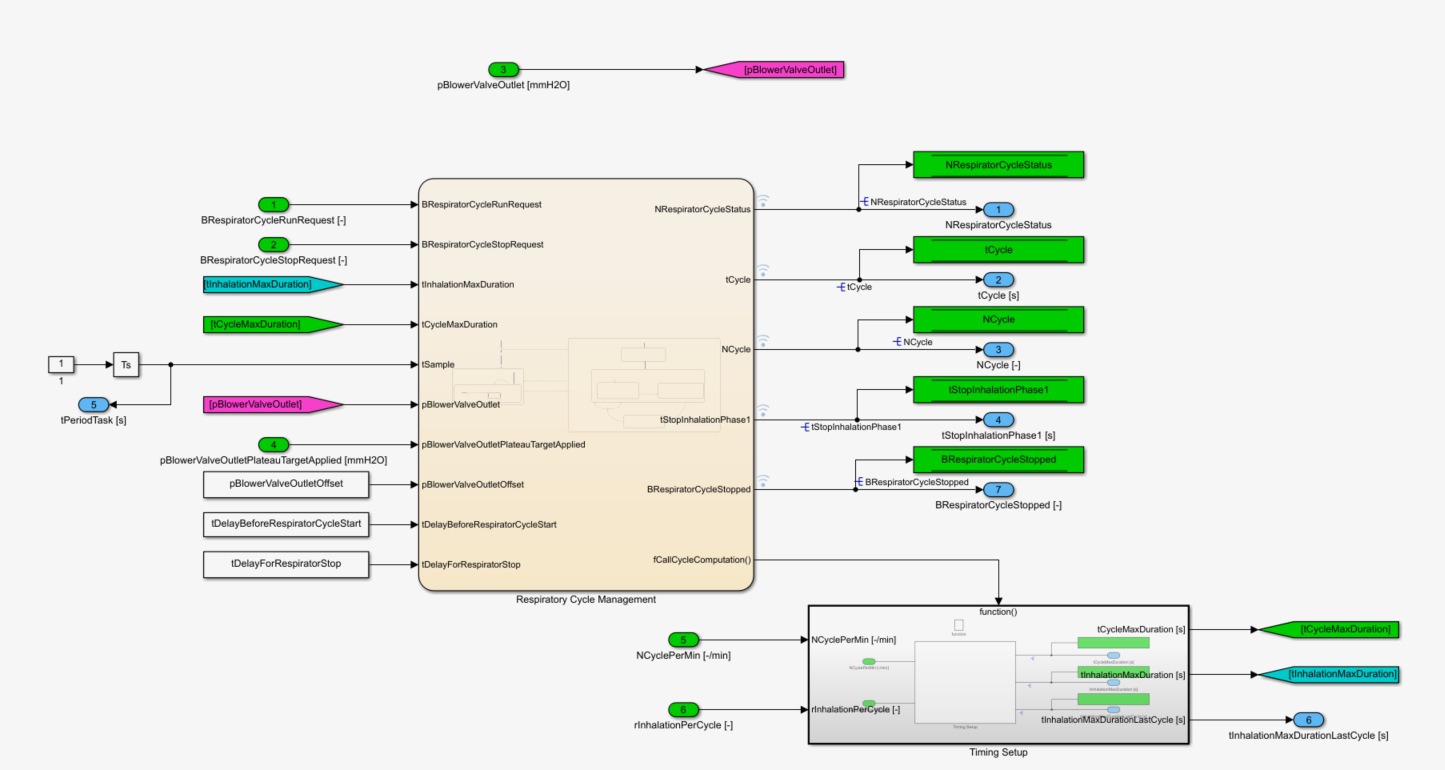


### State Manager

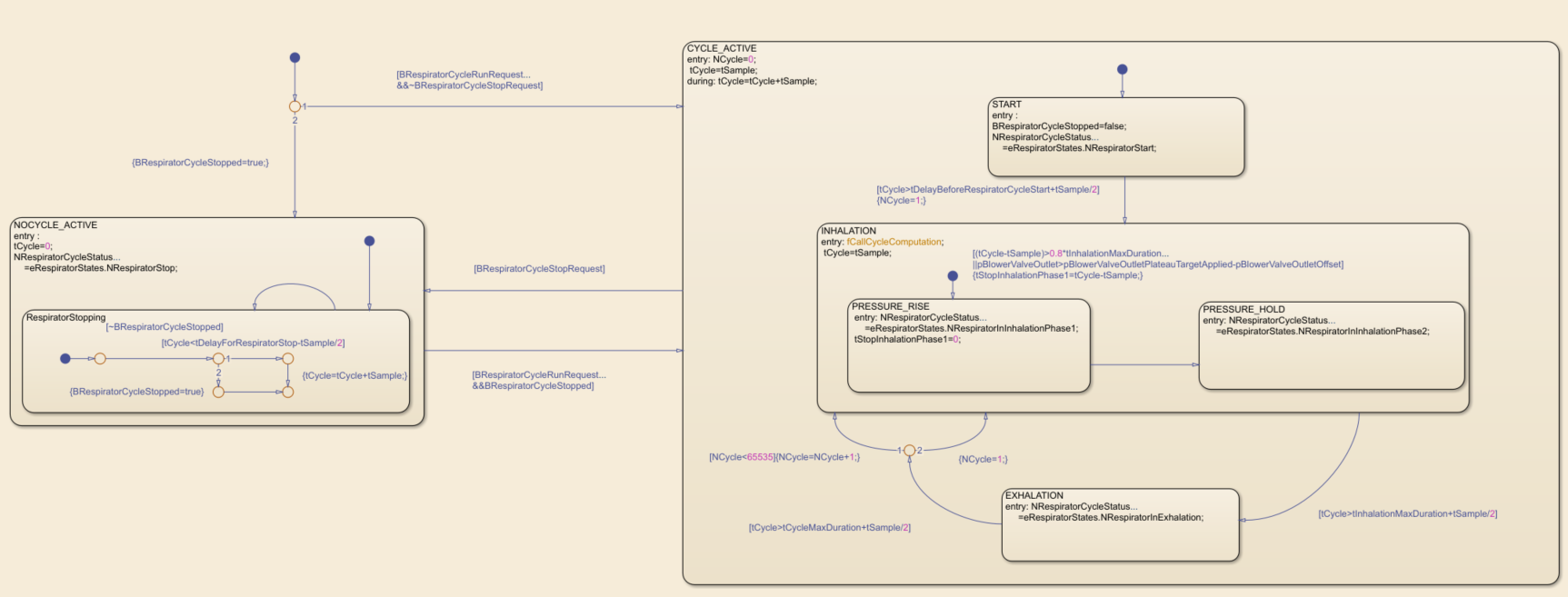
For cycle management, a Stateflow is implemented in order to handle 5 states:

* “Start”: 1st state in case of cycles’ request in order to start the blower and valid until tCycle < tDelayBeforeRespiratorCycleStart
* “Inhalation phase 1”: 1st inhalation phase until the pressure reaches the target
* “Inhalation phase 2”: 2nd inhalation phase valid until tCycle < tInhalationMaxDuration
* “Exhalation”: exhalation phase valid until tCycle<tCycleMaxDuration
* “Stop”: stop of the respiratory cycles after tDelayForRespiratorStop

At each cycle start, the module “Timing Setup” computes the maximum duration of the inhalation phase & of the cycle from the requested frequency & the ratio I:E.

*NucleoRespirator/Control/Respirator Control/Actuator Cmd/CYCLE/Cycle Control/Cycle Control Wrapper/Cycle pController/Cycle Manager* 

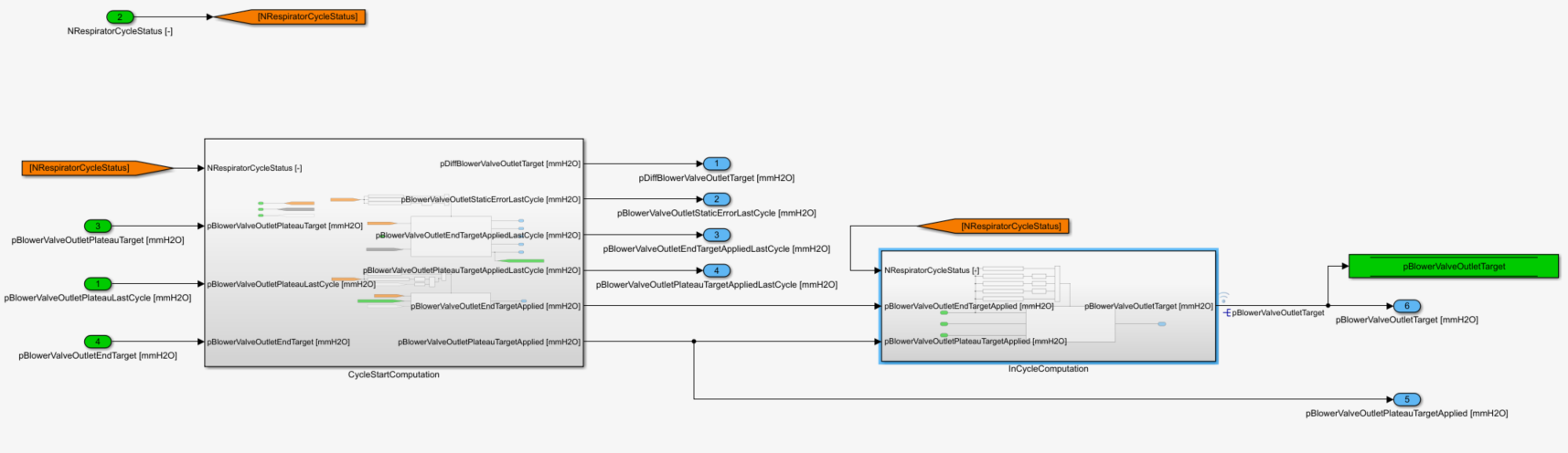
*NucleoRespirator/Control/Respirator Control/Actuator Cmd/CYCLE/Cycle Control/Cycle Control Wrapper/Cycle pController/Cycle Manager/Respiratory Cycle Management*



### Pressure target

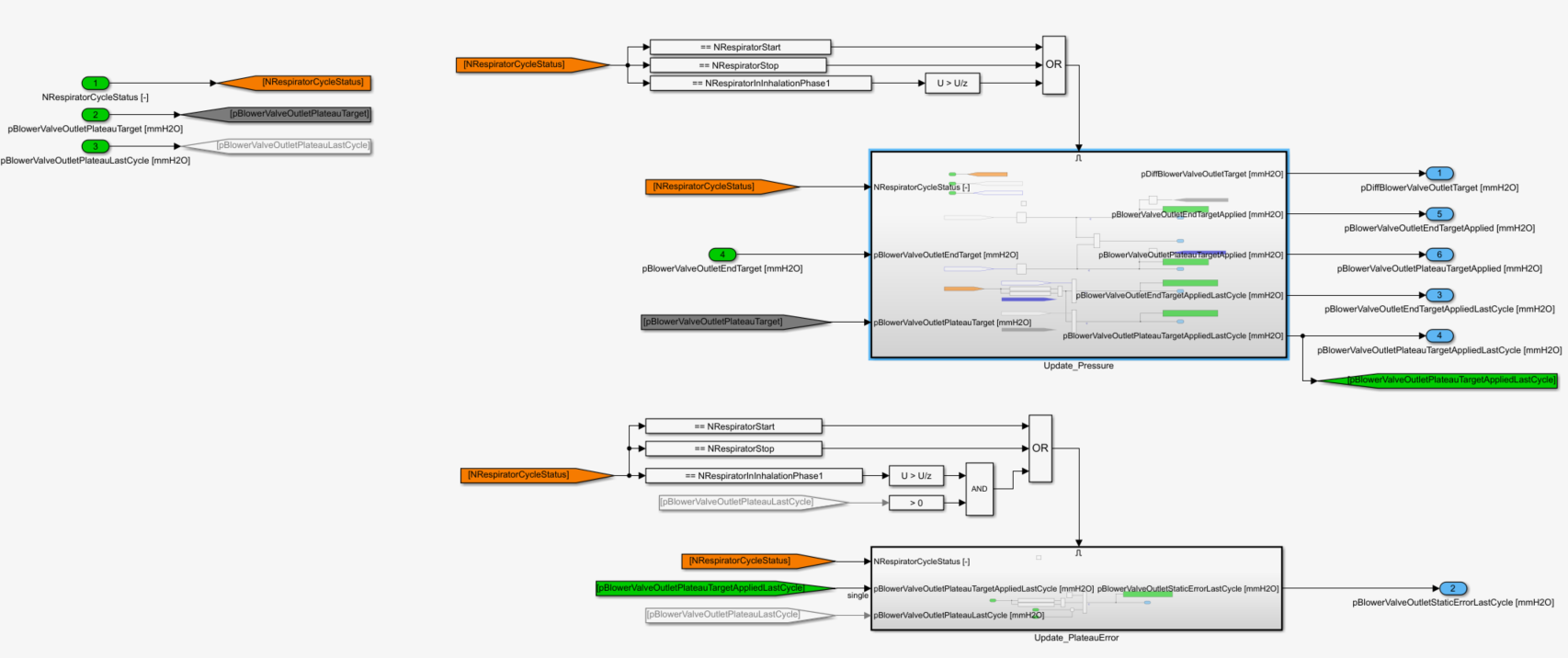
The module “CycleStartComputation” computes the pressure targets and the closed-loop error. The module “InCycleComputation” defines the pressure setpoint to be applied for each phase.

*NucleoRespirator/Control/Respirator Control/Actuator Cmd/CYCLE/Cycle Control/Cycle Control Wrapper/Cycle pController/Pressure Target*



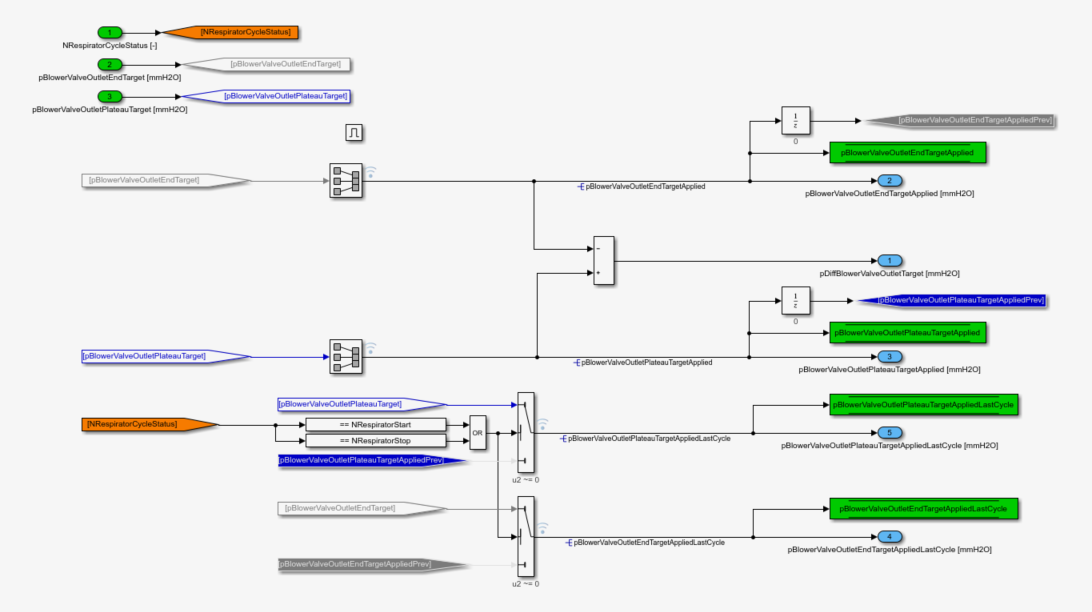
The functions are computed during the phases “start”, “stop” and at each cycle start. The static error computation of the previous cycle requests in addition a valid plateau measurement.

*NucleoRespirator/Control/Respirator Control/Actuator Cmd/CYCLE/Cycle Control/Cycle Control Wrapper/Cycle pController/Pressure Target/CycleStartComputation*



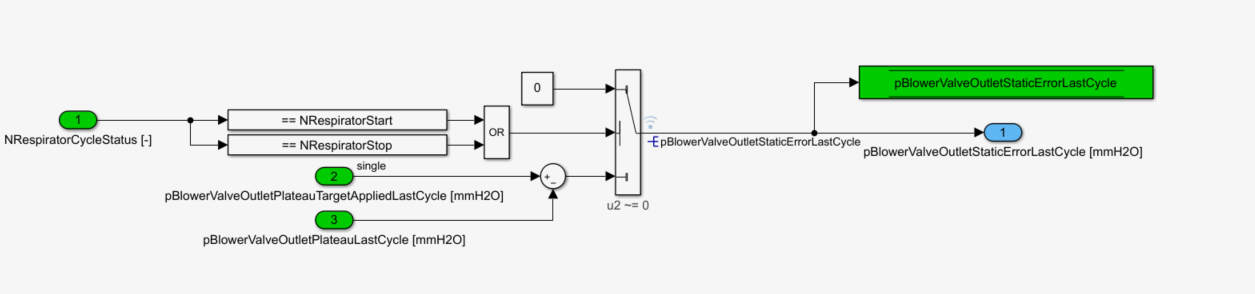
The value “…*LastCycle*” is enforced to the current setpoint during the phases “start” & “stop”. Then, it is connected to the setpoint of the previous cycle.

*NucleoRespirator/Control/Respirator Control/Actuator Cmd/CYCLE/Cycle Control/Cycle Control Wrapper/Cycle pController/Pressure Target/ CycleStartComputation/ Update\_Pressure*



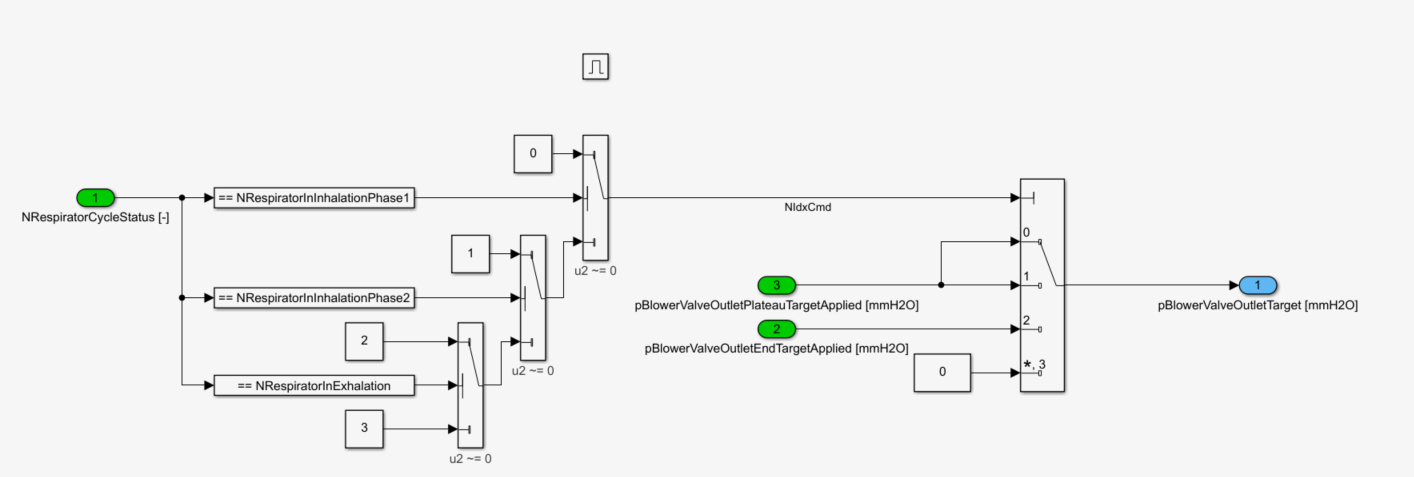
During the phases “start” & “stop”, the static error is set to zero. Otherwise, it is computed from the setpoint & the measurement of the previous cycle.

*NucleoRespirator/Control/Respirator Control/Actuator Cmd/CYCLE/Cycle Control/Cycle Control Wrapper/Cycle pController/Pressure Target/ CycleStartComputation/ Update\_PlateauError*



pBlowerValveOutletPlateauTargetApplied is dedicated to the inhalation phase. pBlowerValveOutletEndTargetApplied is set during the exhalation. For all other cases, the pressure target is null.

*NucleoRespirator/Control/Respirator Control/Actuator Cmd/CYCLE/Cycle Control/Cycle Control Wrapper/Cycle pController/Pressure Target/ InCycleComputation/ UpdatePressureTarget*



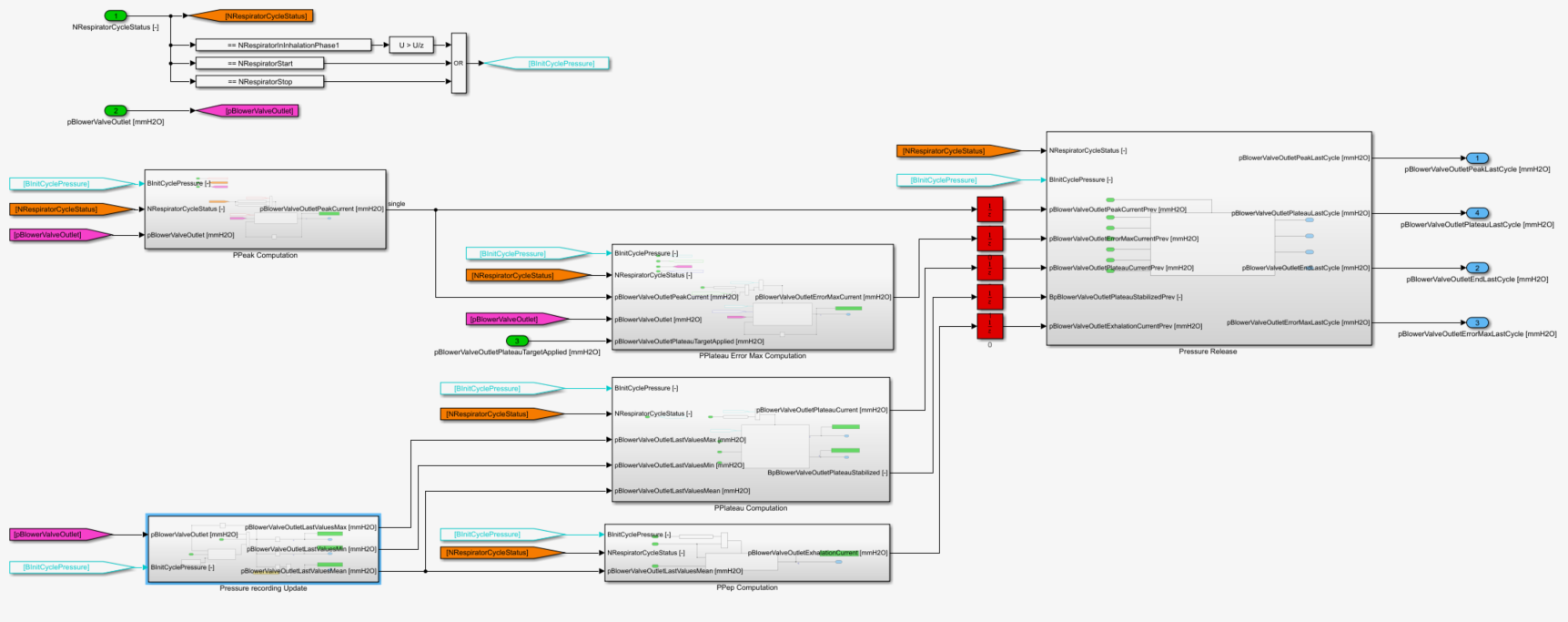
### Pressure processing

The module “Pressure processing” is responsible for the computation of all indicators required for the pressure control or the warning feature. The main outputs are:

* pBlowerValveOutletPeakLastCycle: maximum pressure reached during inhalation of previous cycle
* pBlowerValveOutletPlateauLastCycle: stabilized pressure level reached during inhalation of previous cycle
* pBlowerValveOutletEndLastCycle: end-expiratory pressure of previous cycle
* pBlowerValveOutletErrorMaxLastCycle: closed-loop max error measured during inhalation of previous cycle

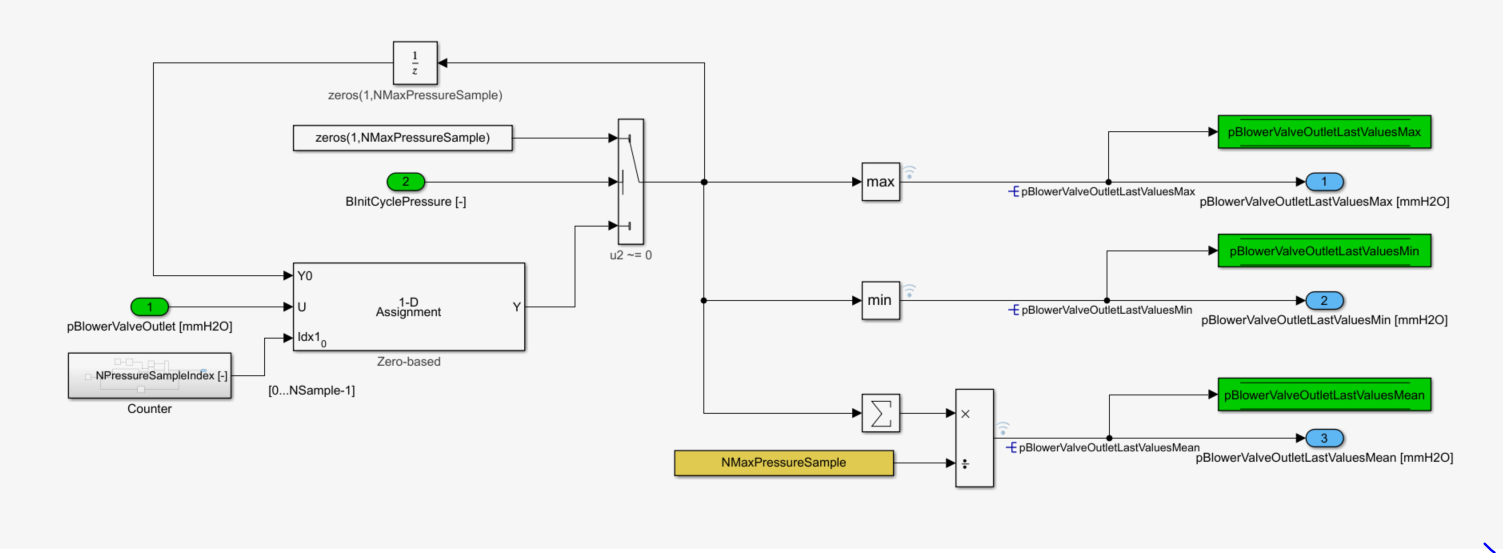
Each indicator is monitored continuously during the running cycle and is released at each cycle end.

*NucleoRespirator/Control/Respirator Control/Actuator Cmd/CYCLE/Cycle Control/Cycle Control Wrapper/Cycle pController/Pressure processing*



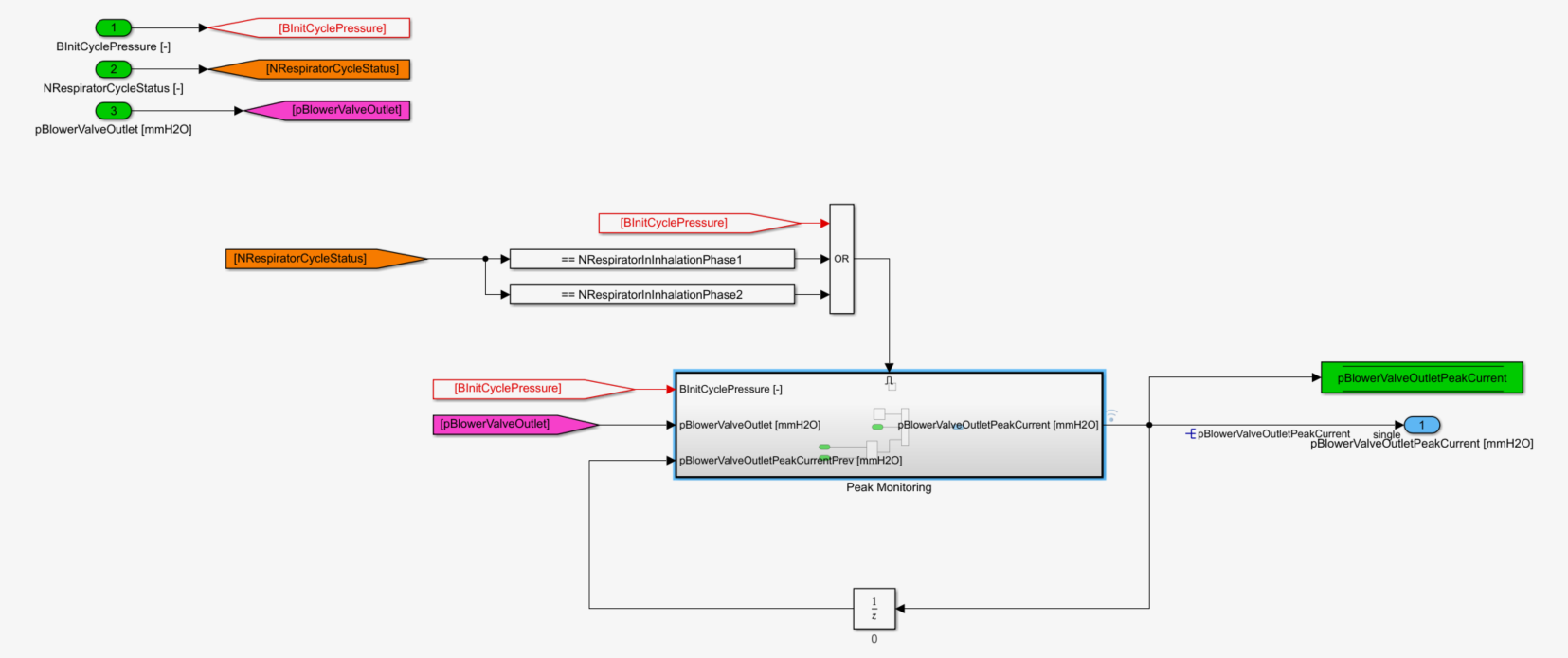
A moving-average of NMaxPressureSample is used for the computation of the plateau pressure and the end-expiratory pressure.

*NucleoRespirator/Control/Respirator Control/Actuator Cmd/CYCLE/Cycle Control/Cycle Control Wrapper/Cycle pController/Pressure processing/Pressure recording Update*

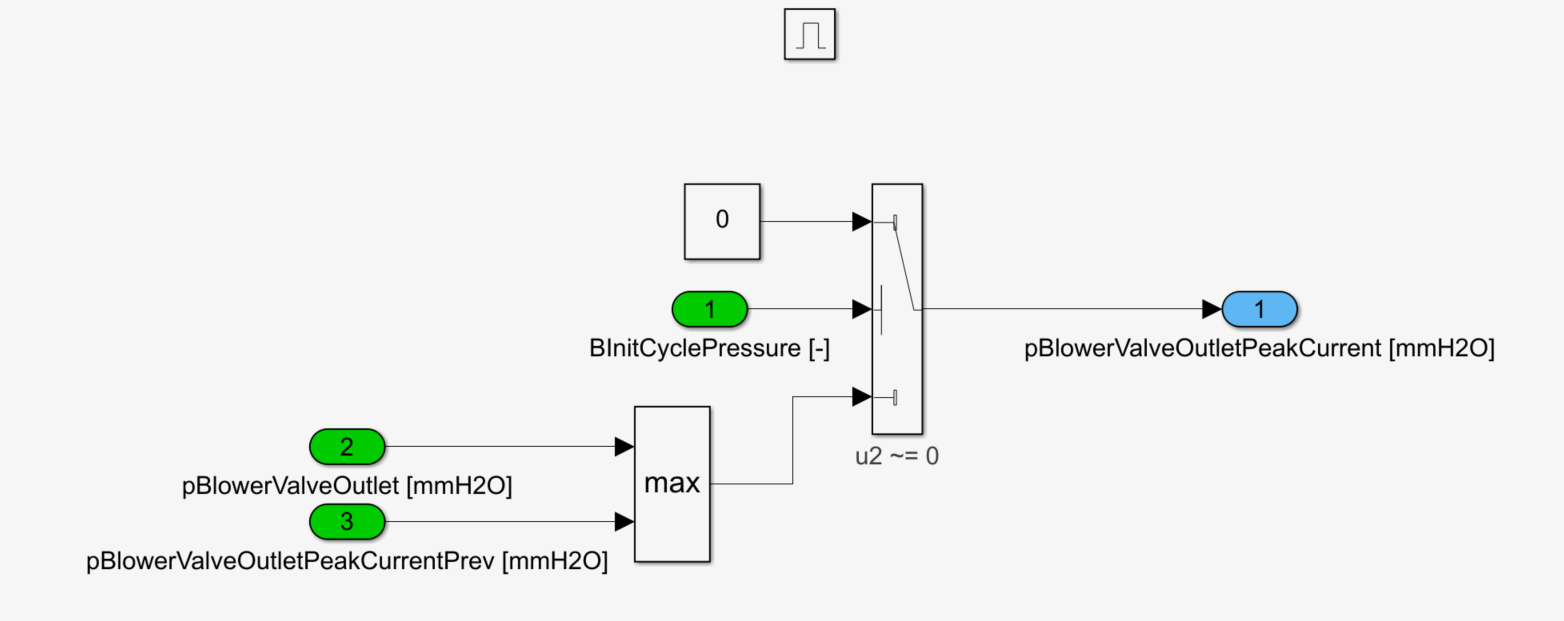


The peak pressure is monitored during the inhalation phase and initialize at each cycle start.

*NucleoRespirator/Control/Respirator Control/Actuator Cmd/CYCLE/Cycle Control/Cycle Control Wrapper/Cycle pController/Pressure processing/PPeak Computation*

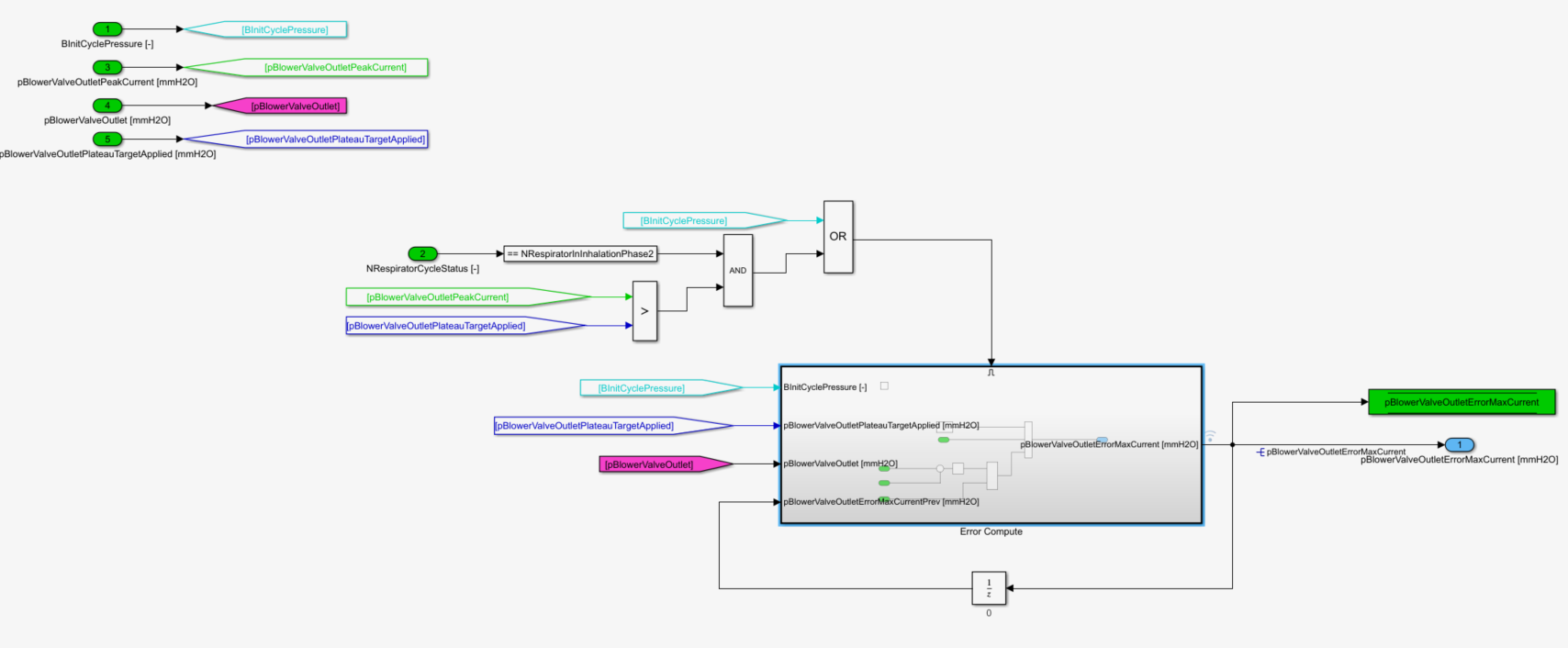


*NucleoRespirator/Control/Respirator Control/Actuator Cmd/CYCLE/Cycle Control/Cycle Control Wrapper/Cycle pController/Pressure processing/PPeak Computation/Peak Monitoring*

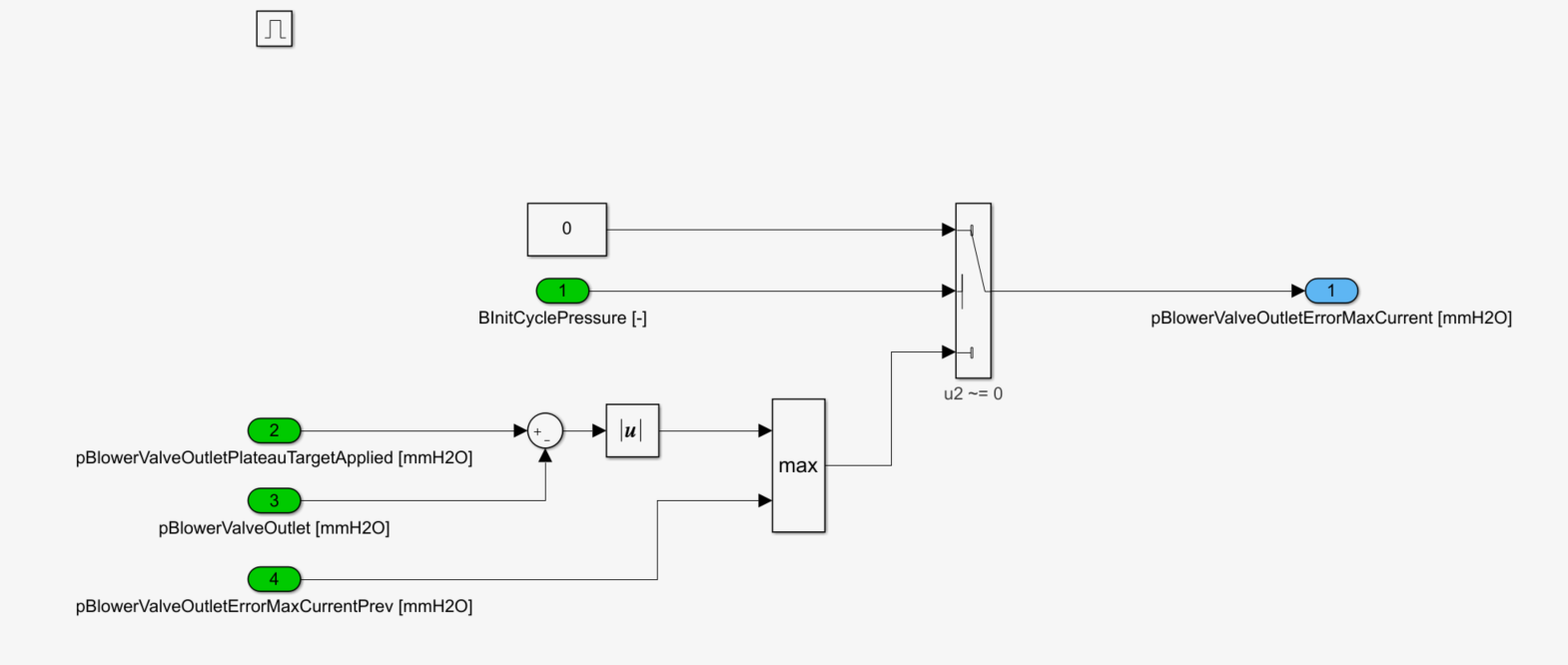


The closed-loop error is an unsigned signal monitored during the 2nd inhalation phase once the peak pressure is higher than the target. In addition, it is initialized at each cycle start.

*NucleoRespirator/Control/Respirator Control/Actuator Cmd/CYCLE/Cycle Control/Cycle Control Wrapper/Cycle pController/Pressure processing/PPlateau Error Max Computation*

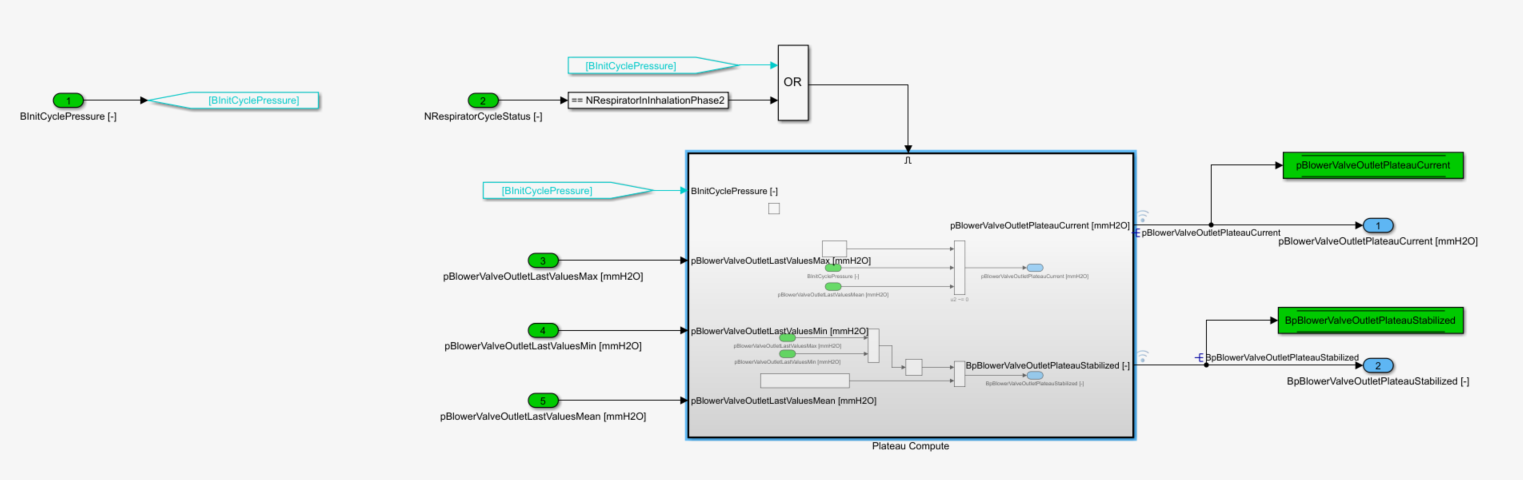


*NucleoRespirator/Control/Respirator Control/Actuator Cmd/CYCLE/Cycle Control/Cycle Control Wrapper/Cycle pController/Pressure processing/PPlateau Error Max Computation/Error Compute*

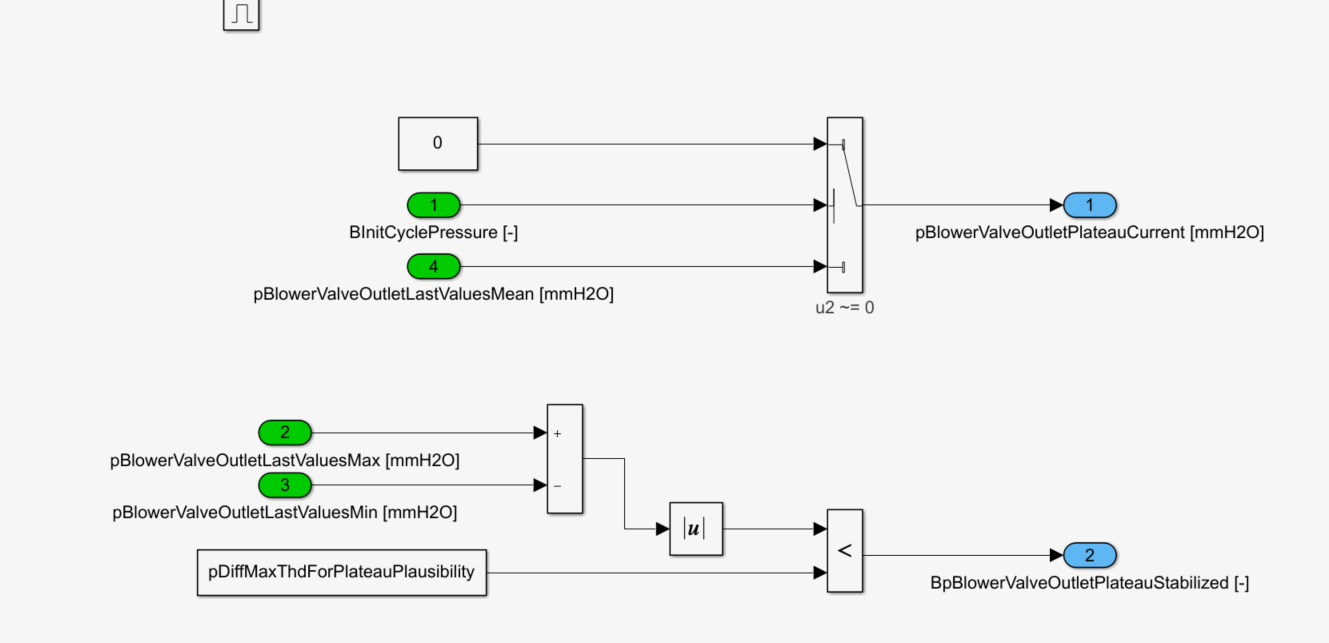


The plateau pressure is the last value of the moving-average computed during the inhalation phase. It is initialized at each cycle start. In addition, the system checks if the control is stabilized or not.

*NucleoRespirator/Control/Respirator Control/Actuator Cmd/CYCLE/Cycle Control/Cycle Control Wrapper/Cycle pController/Pressure processing/PPlateau Computation*

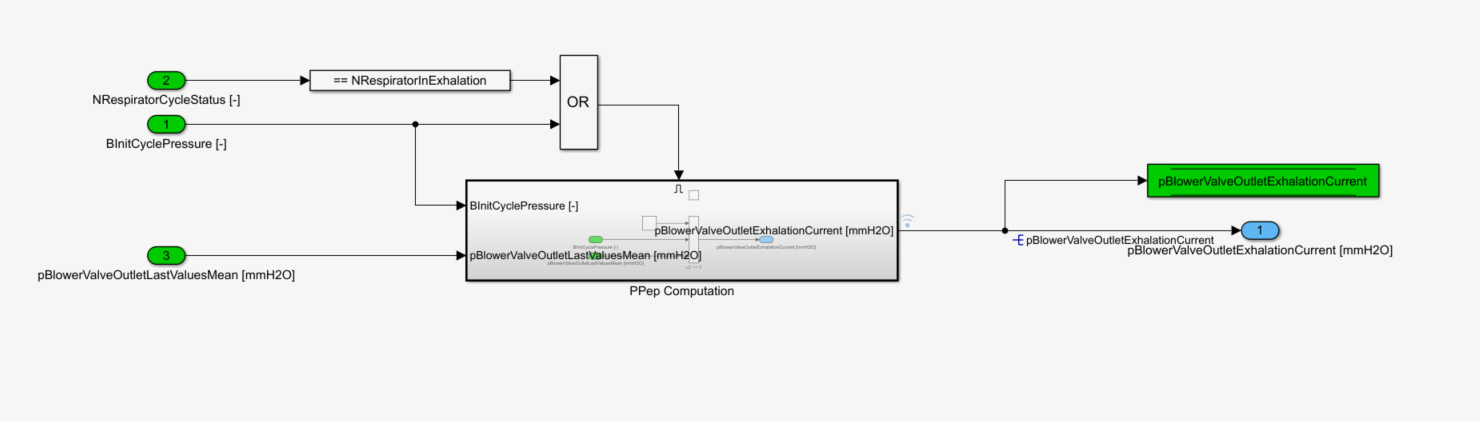


*NucleoRespirator/Control/Respirator Control/Actuator Cmd/CYCLE/Cycle Control/Cycle Control Wrapper/Cycle pController/Pressure processing/PPlateau Computation/Plateau Compute*

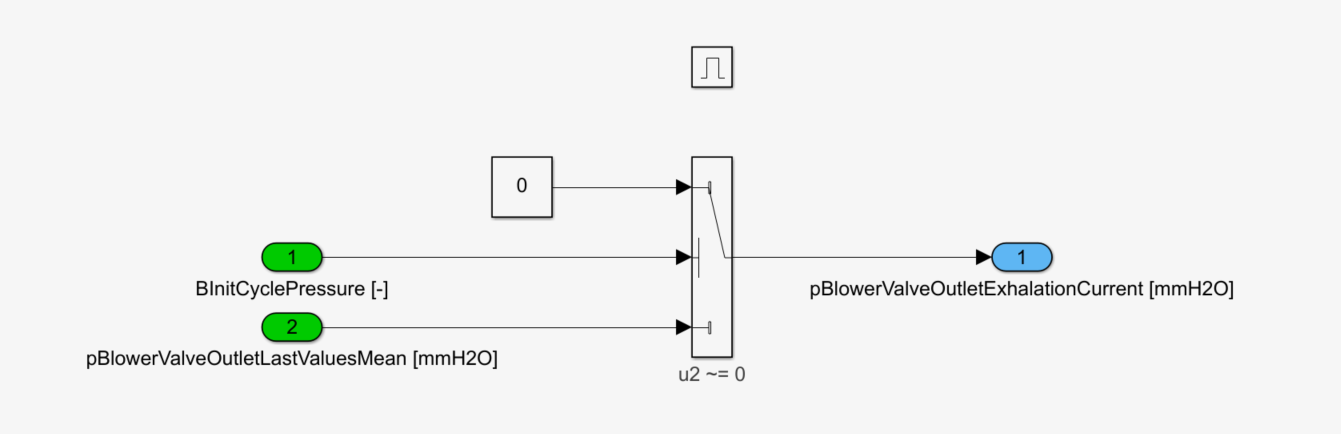


The end-expiratory pressure is the last value of the moving-average computed during the exhalation phase. In addition, it is initialized at each cycle start.

*NucleoRespirator/Control/Respirator Control/Actuator Cmd/CYCLE/Cycle Control/Cycle Control Wrapper/Cycle pController/Pressure processing/PPep Computation*

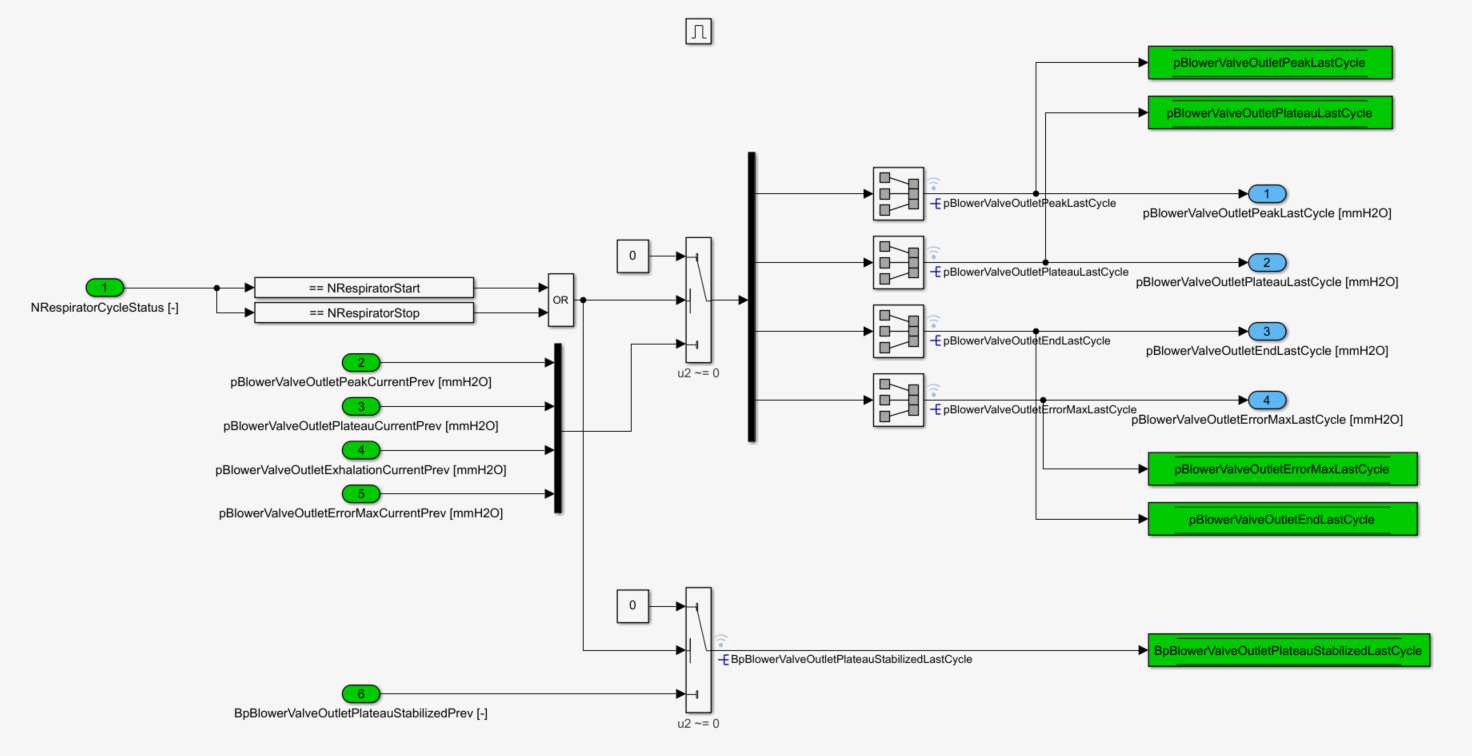


*NucleoRespirator/Control/Respirator Control/Actuator Cmd/CYCLE/Cycle Control/Cycle Control Wrapper/Cycle pController/Pressure processing/PPep Computation/PPep Computation*



Finally, all the indicators are set to zero during the phases “start” and “stop”. Otherwise, the function just provides the last signal of the previous cycle at each cycle start.

*NucleoRespirator/Control/Respirator Control/Actuator Cmd/CYCLE/Cycle Control/Cycle Control Wrapper/Cycle pController/Pressure processing/Pressure Release /Release*

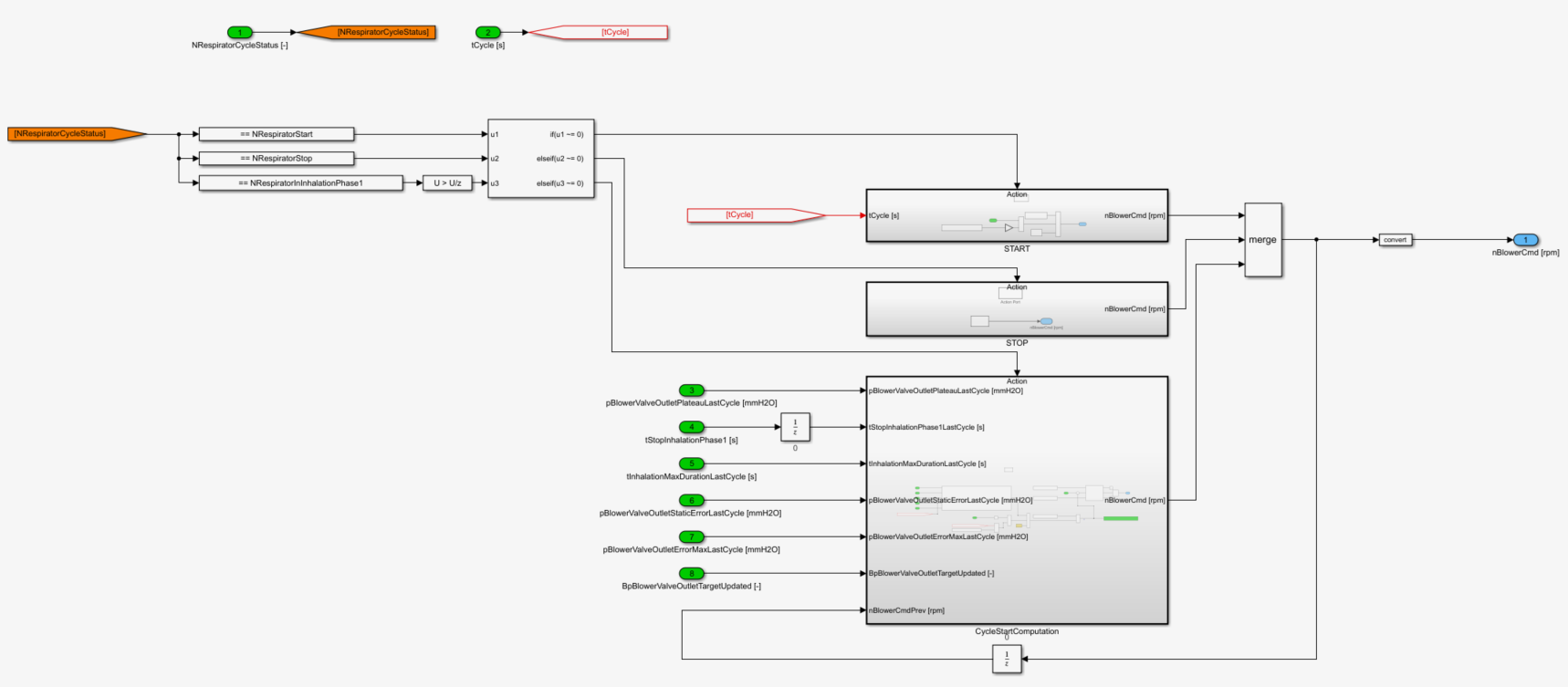


### Blower speed target

The blower speed target is:

* switched to NBlowerCmdInit after tDelayBeforeRespiratorCycleStart/2 in “start” mode
* set to zero in “stop” mode
* computed in the module “CycleStartComputation” at each cycle start

*NucleoRespirator/Control/Respirator Control/Actuator Cmd/CYCLE/Cycle Control/Cycle Control Wrapper/Cycle pController/BlowerSpeed\_Target*

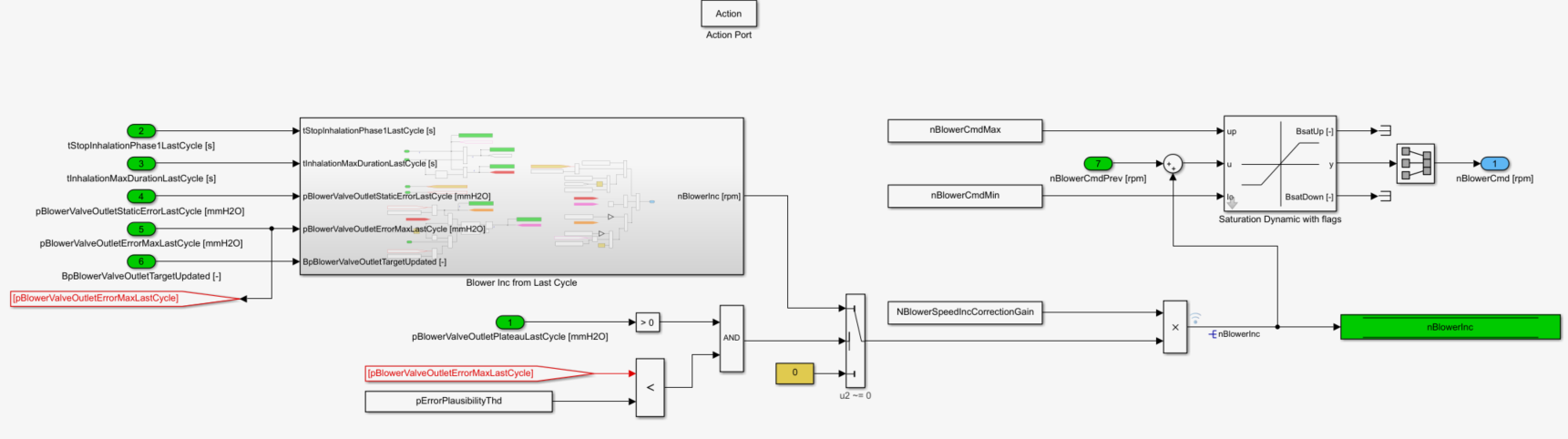


*NucleoRespirator/Control/Respirator Control/Actuator Cmd/CYCLE/Cycle Control/Cycle Control Wrapper/Cycle pController/BlowerSpeed\_Target/START*



The blower speed command is saturated by nBlowerCmdMin & nBlowerCmdMax. In case of plausibility issue on pBlowerValveOutletErrorMaxLastCycle or no plateau detection, the speed increment is set to zero. Otherwise, the increment is computed in the module “Blower Inc. from Last Cycle”.

*NucleoRespirator/Control/Respirator Control/Actuator Cmd/CYCLE/Cycle Control/Cycle Control Wrapper/Cycle pController/BlowerSpeed\_Target/CycleStartComputation*



In principle, the system tries to reach a response time defined in the map tPhase1MaxTarget=f(pInhlation-pExhalation). If the response time is above the target, the function computes a positive increment. Three parameters are available:

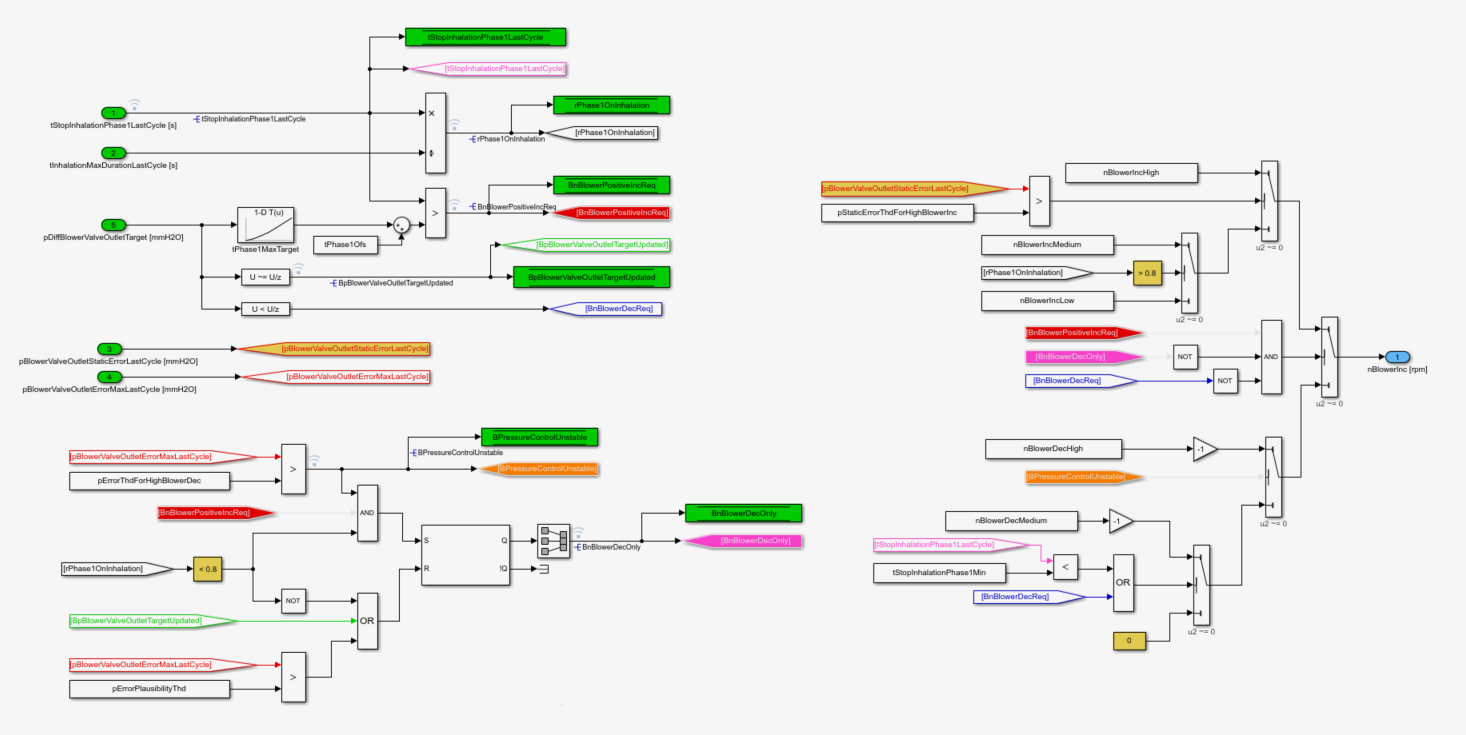
* nBlowerHighInc: if the static error is above pStaticErrorThdForHighBlowerInc
* nBlowerMediumInc: if the response time is longer than 80% of the inhalation duration
* nBlowerlowInc

In case of setpoint reduction or too fast response time (<tStopInhalationPhase1Min), a decrement is applied. Finally, a mode “DecOnly” is implemented in case:

* tPhase1MaxTarget < Response time < 80% of the inhalation phase
* Overshoot > pErrorThdForHighBlowerDec

By this, the system promotes a stable system over a dynamic one. This mode is removed in case of setpoint update or plausibility issue.

*NucleoRespirator/Control/Respirator Control/Actuator Cmd/CYCLE/Cycle Control/Cycle Control Wrapper/Cycle pController/BlowerSpeed\_Target/CycleStartComputation/Blower Inc from Last Cycle*



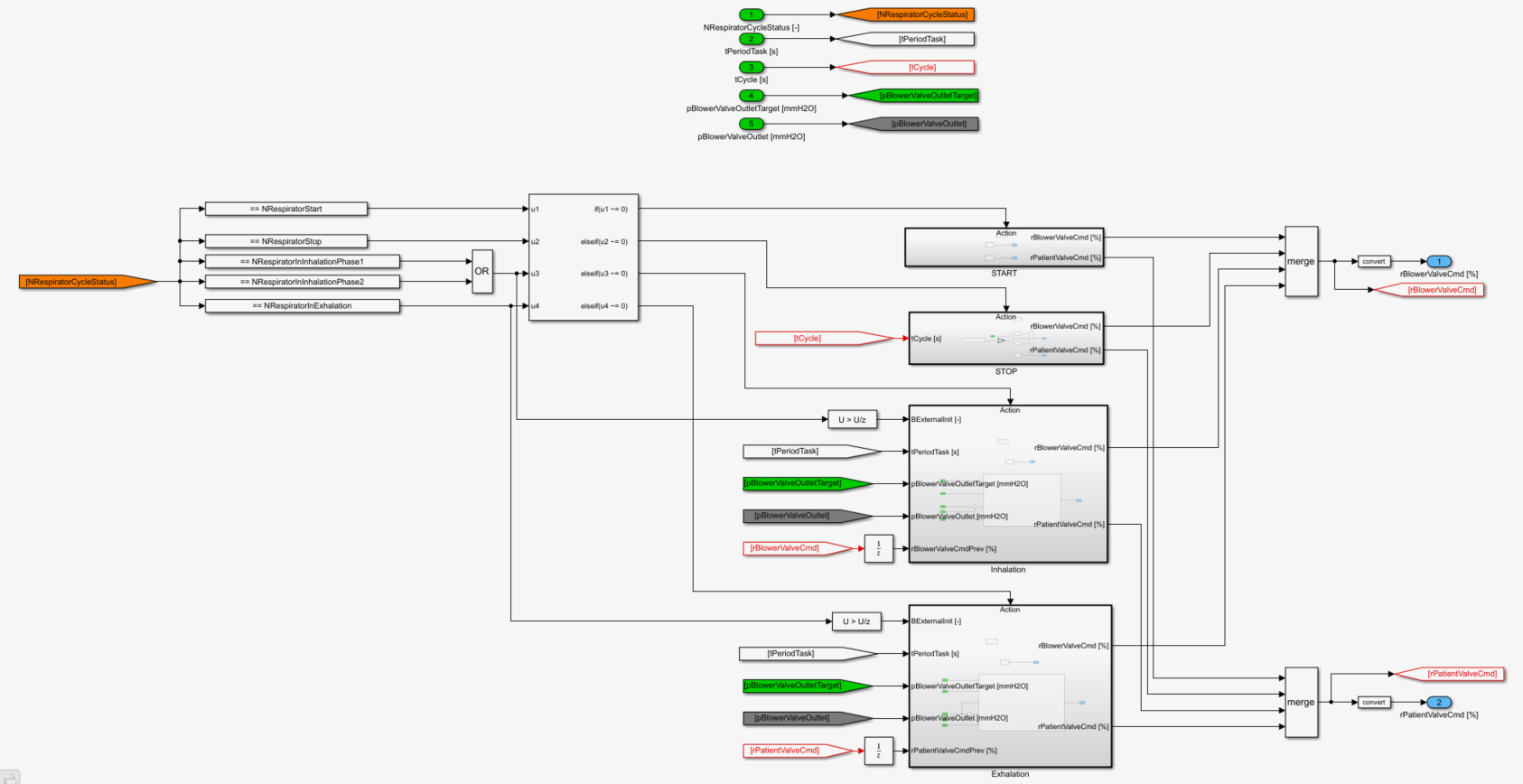
### Pressure valve command

This module defines the pressure valve commands (0-100%) as detailed below:

|  |  |  |
| --- | --- | --- |
|  | **Blower valve** | **Patient valve** |
| **Start** | 0% | 100% |
| **Stop** | 0% 🡪 100% if t>tDelayForRespiratorStop | 100% |
| **Inhalation** | PID | 0% |
| **Exhalation** | 0% | PID |

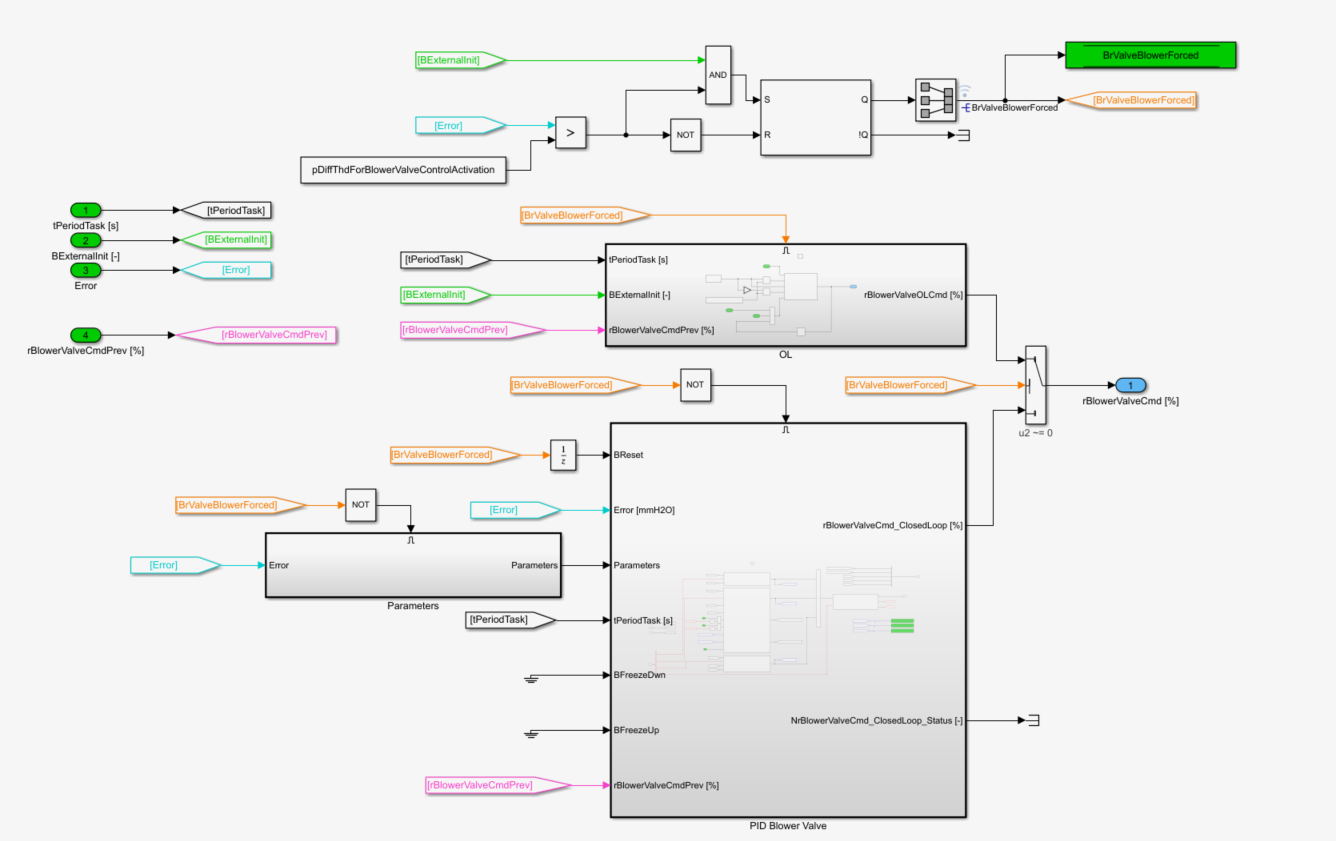
The PID structure is similar for each valve except the PID parameters. Therefore, we will detail only the inhalation sub-module.

*NucleoRespirator/Control/Respirator Control/Actuator Cmd/CYCLE/Cycle Control/Cycle Control Wrapper/Cycle pController/ValvePosition\_Target*



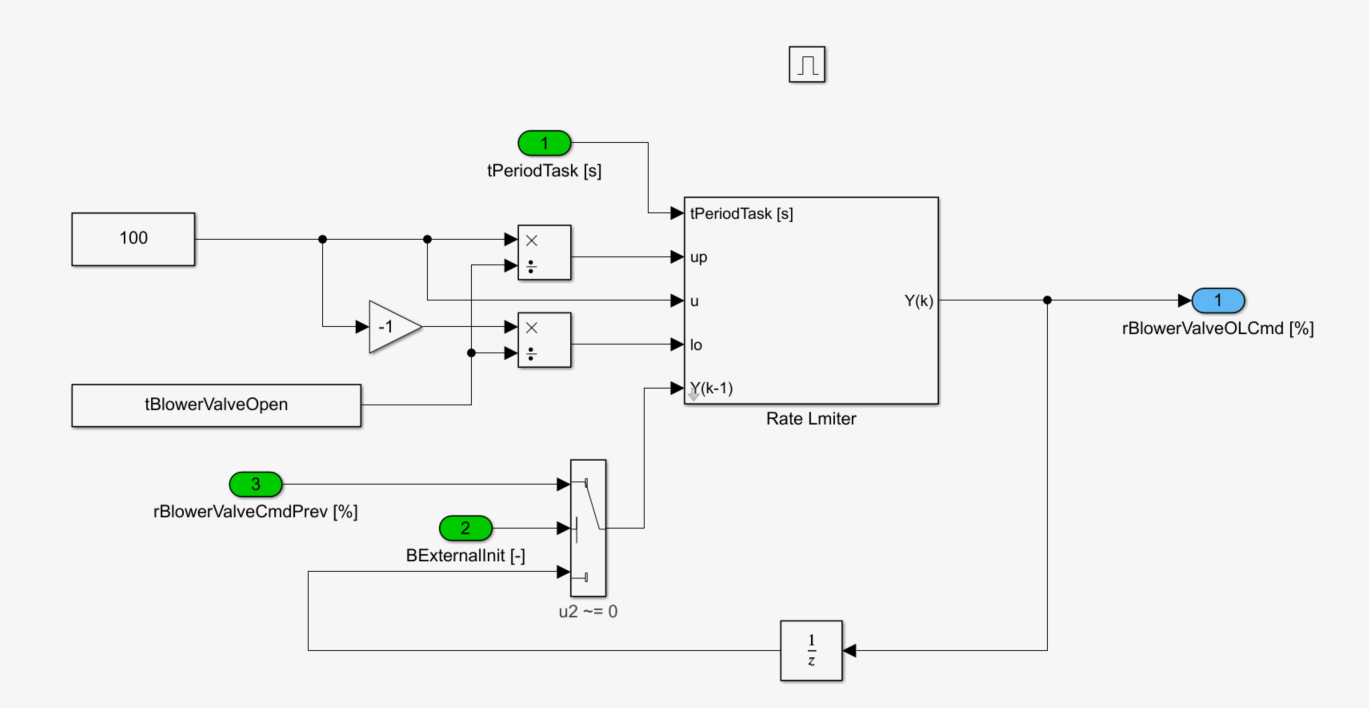
At cycle start, the function enforces first an OL mode which can be considered as a ramp from 0 to 100% during tBlowerValveOpen. Once the pressure error is below pDiffThdForBlowerValveControlActivation, the PID control is activated.

*NucleoRespirator/Control/Respirator Control/Actuator Cmd/CYCLE/Cycle Control/Cycle Control Wrapper/Cycle pController/ValvePosition\_Target/Inhalation/Blower Valve Control*



This module computes the OL command. It is a rate limiter initialized at inhalation start. After tBlowerValveOpen, the command is 100%.

*NucleoRespirator/Control/Respirator Control/Actuator Cmd/CYCLE/Cycle Control/Cycle Control Wrapper/Cycle pController/ValvePosition\_Target/Inhalation/Blower Valve Control/OL*



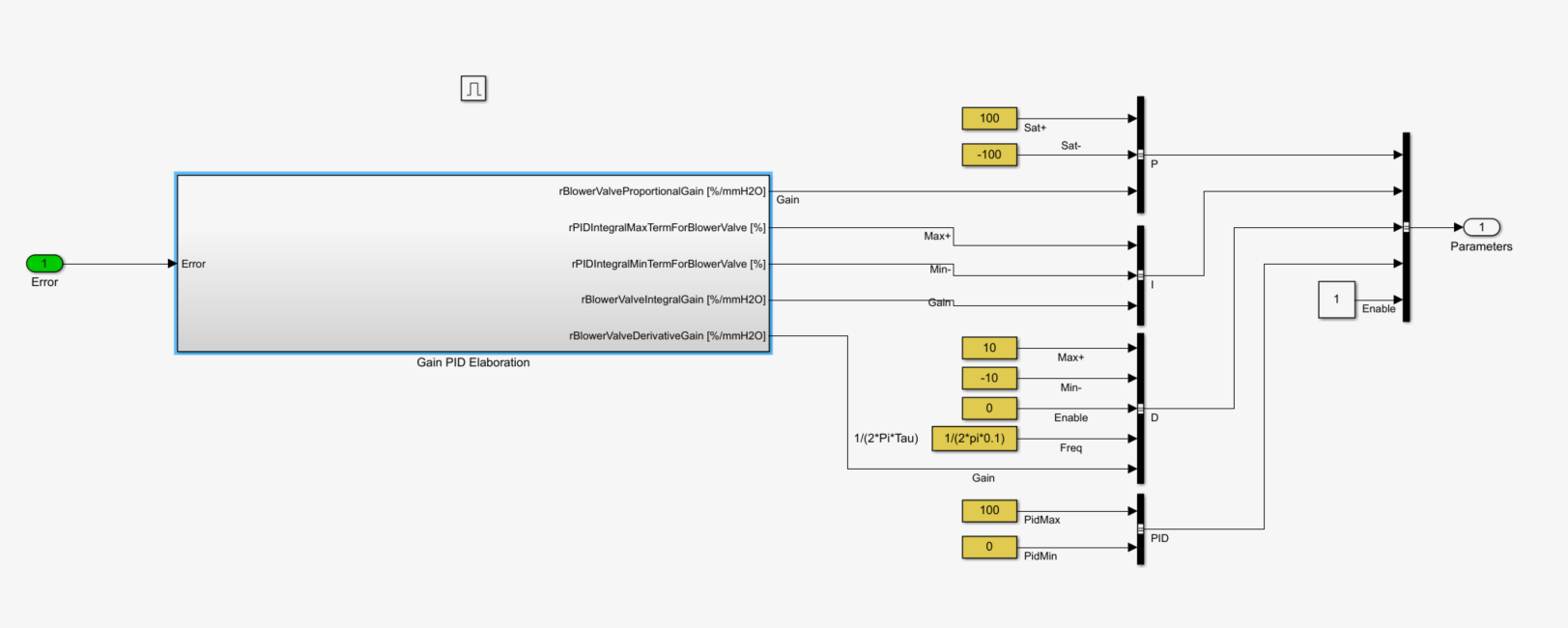
This block details the PID parameters. Please find below an overview of the parameters for the blower valve:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **P** | **I** | **D** | **PID** |
| ***Gain*** | NBlowerValvePropGain | NBlowerValveIntegralGain  or  NBlowerValveIntegralHighGain | NBlowerValveDerivativeGain |  |
| ***Max*** | 100% | rPIDIntegralMaxTermForBlowerValve  =100% | 10% | 100% |
| ***Min*** | -100% | rPIDIntegralMinTermForBlowerValve  =0% | -10% | 0% |
| ***Extra*** |  |  | **Enable = 0**  Freq=1.59Hz | NA |

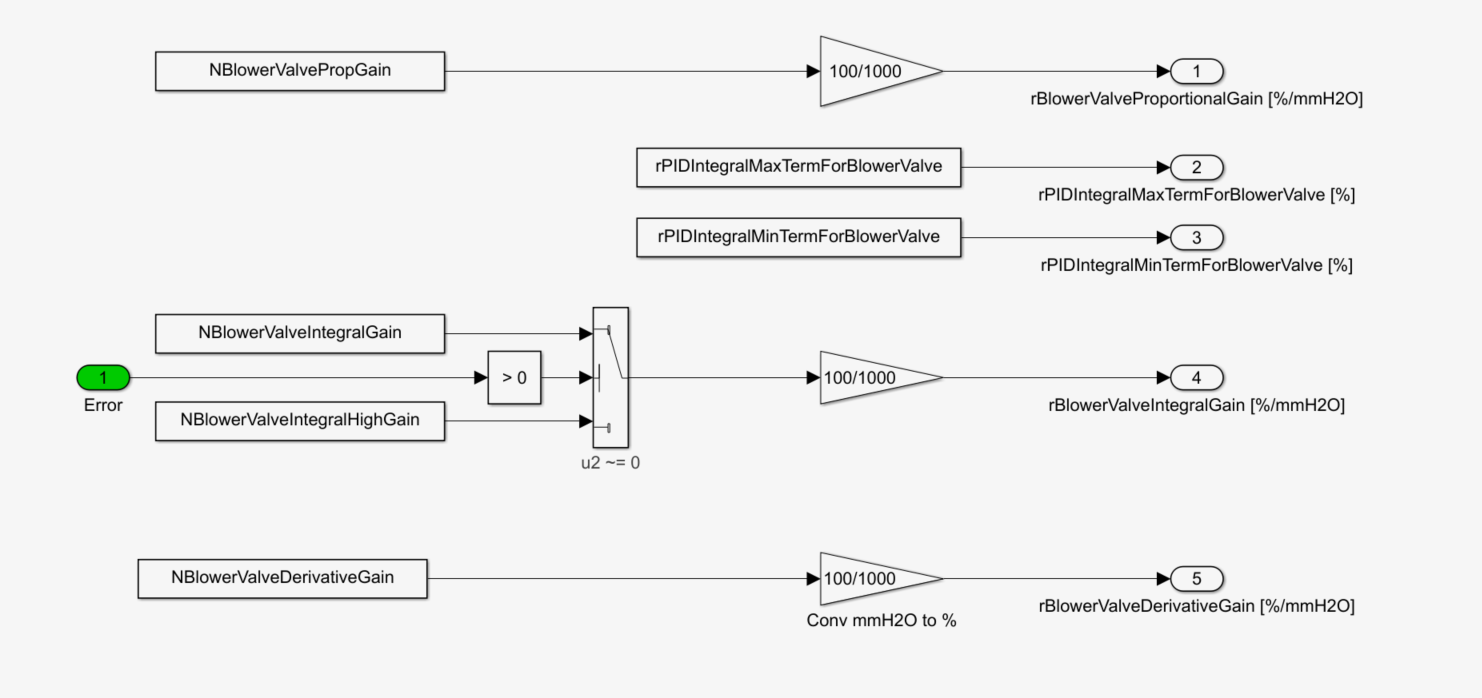
For the patient valve, a correction factor for the proportional & integral gain was added in function of the pressure setpoint.

The derivative term is deactivated in version 1.0.0.

*NucleoRespirator/Control/Respirator Control/Actuator Cmd/CYCLE/Cycle Control/Cycle Control Wrapper/Cycle pController/ValvePosition\_Target/Inhalation/Blower Valve Control/Parameters*

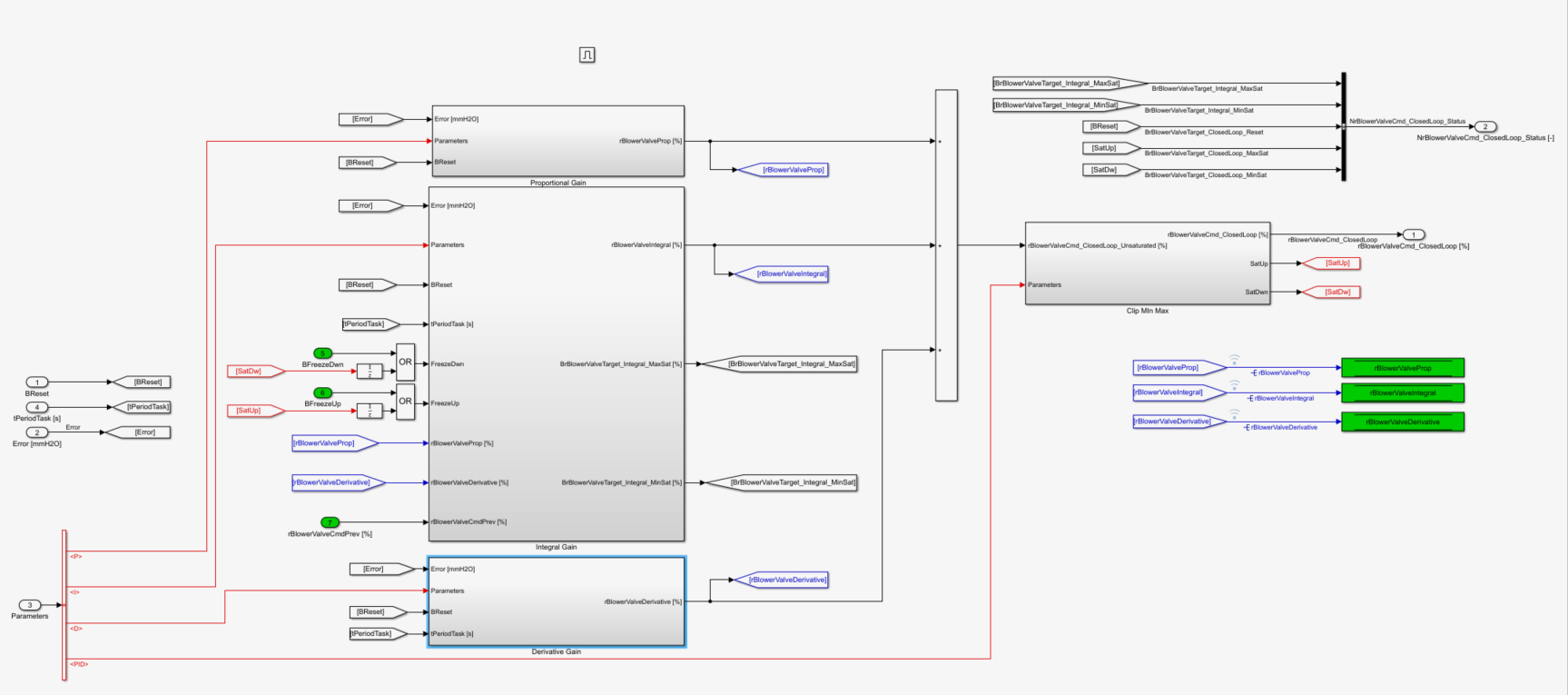


*NucleoRespirator/Control/Respirator Control/Actuator Cmd/CYCLE/Cycle Control/Cycle Control Wrapper/Cycle pController/ValvePosition\_Target/Inhalation/Blower Valve Control/Parameters/Gain PID Elaboration*



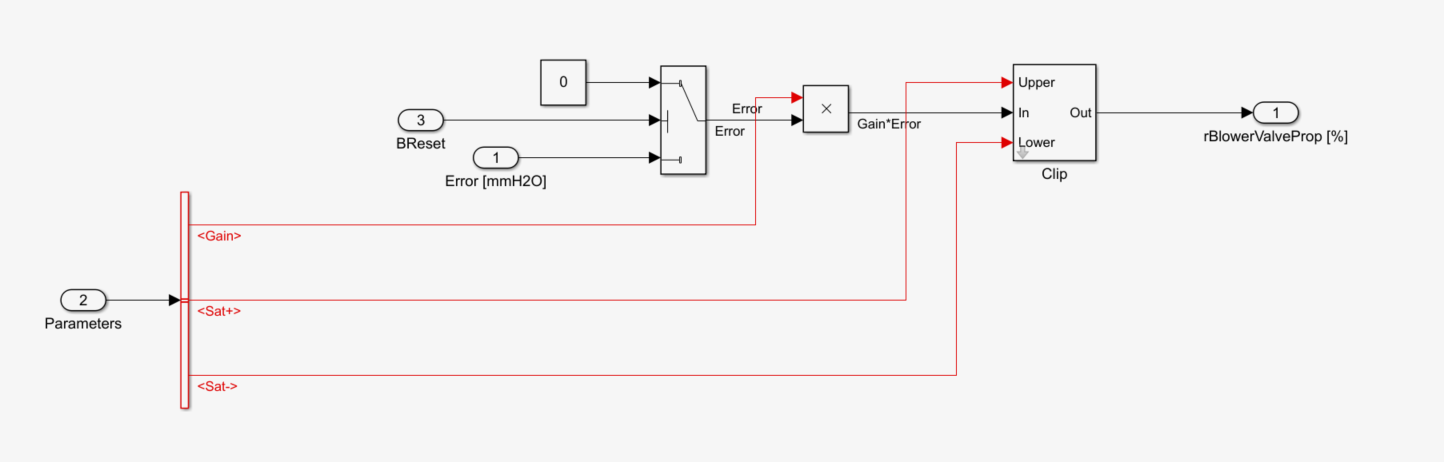
The following blocks compute the PID terms which are added altogether and then saturated.

*NucleoRespirator/Control/Respirator Control/Actuator Cmd/CYCLE/Cycle Control/Cycle Control Wrapper/Cycle pController/ValvePosition\_Target/Inhalation/Blower Valve Control/PID Blower Valve*



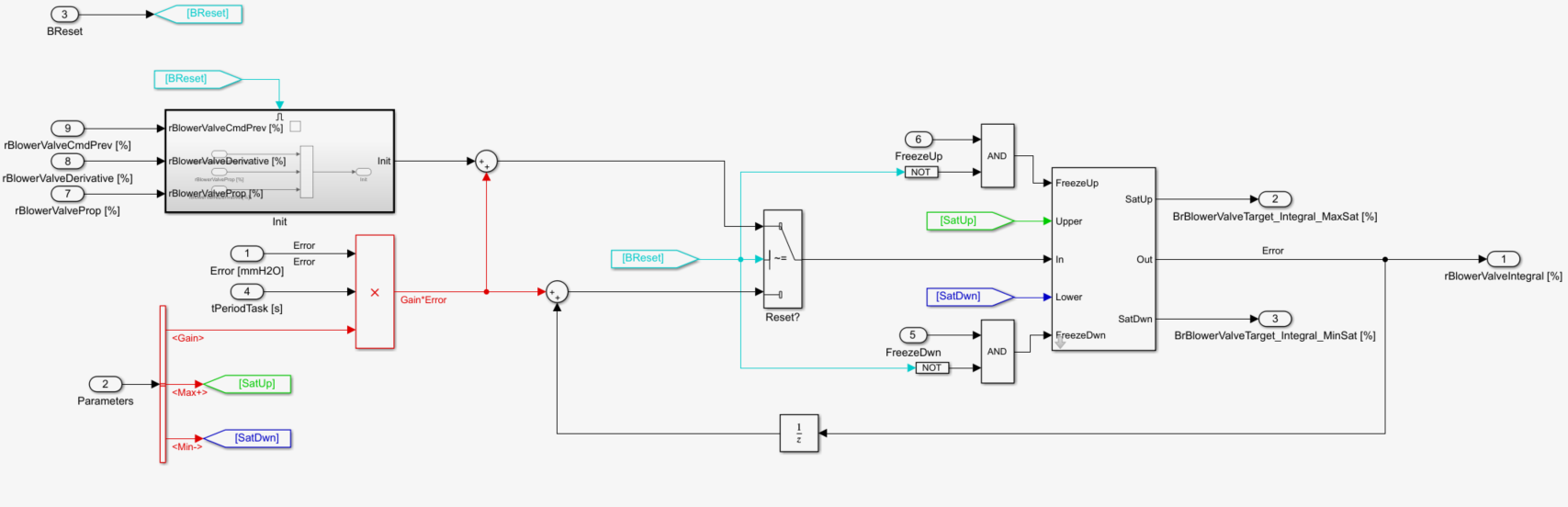
The proportional term is reset at init and saturated.

*NucleoRespirator/Control/Respirator Control/Actuator Cmd/CYCLE/Cycle Control/Cycle Control Wrapper/Cycle pController/ValvePosition\_Target/Inhalation/Blower Valve Control/PID Blower Valve/Proportional Gain*



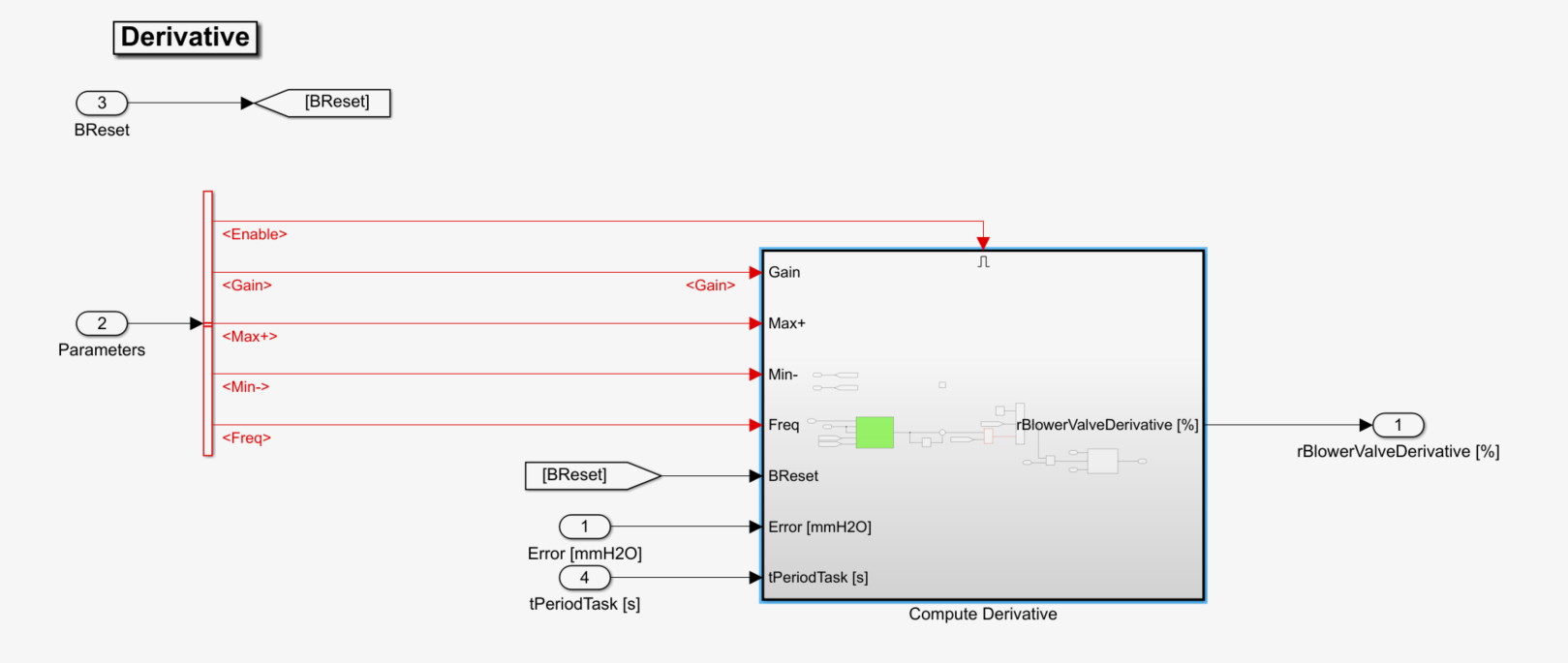
The integral term is initialized based on the last valve command, the proportional term and the derivative term in order to ensure the continuity of the command. It is saturated by its min & max values and frozen if the valve command is saturated.

*NucleoRespirator/Control/Respirator Control/Actuator Cmd/CYCLE/Cycle Control/Cycle Control Wrapper/Cycle pController/ValvePosition\_Target/Inhalation/Blower Valve Control/PID Blower Valve/Integral Gain*



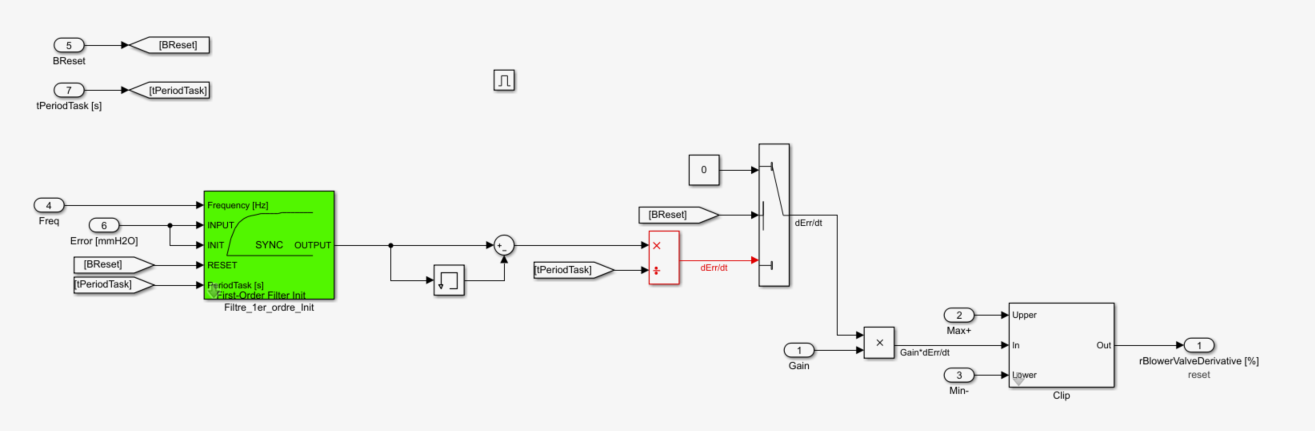
The derivative term is computed if the feature is enabled.

*NucleoRespirator/Control/Respirator Control/Actuator Cmd/CYCLE/Cycle Control/Cycle Control Wrapper/Cycle pController/ValvePosition\_Target/Inhalation/Blower Valve Control/PID Blower Valve/Derivative Gain*



The error is filtered in order to calm down the derivative. At init, the derivative term is set to zero. Finally, the signal is saturated by its minimum and maximum.

*NucleoRespirator/Control/Respirator Control/Actuator Cmd/CYCLE/Cycle Control/Cycle Control Wrapper/Cycle pController/ValvePosition\_Target/Inhalation/Blower Valve Control/PID Blower Valve/Derivative Gain/Compute Derivative*



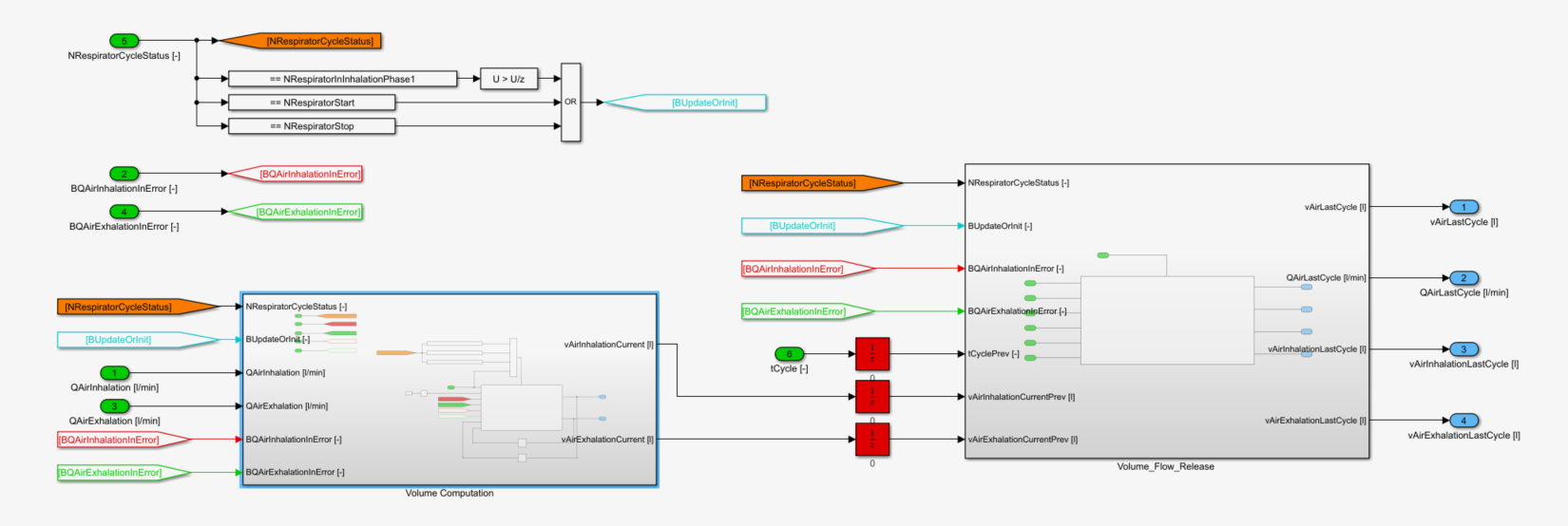
## Air Volume processing

This block is dedicated to the computation of the tidal volume based on the instantaneous flow. it computes continuously a volume from the inhalation flowmeter and another one from the exhalation flowmeter. At the end of each cycle, the system provides the final tidal volume which is equal to:

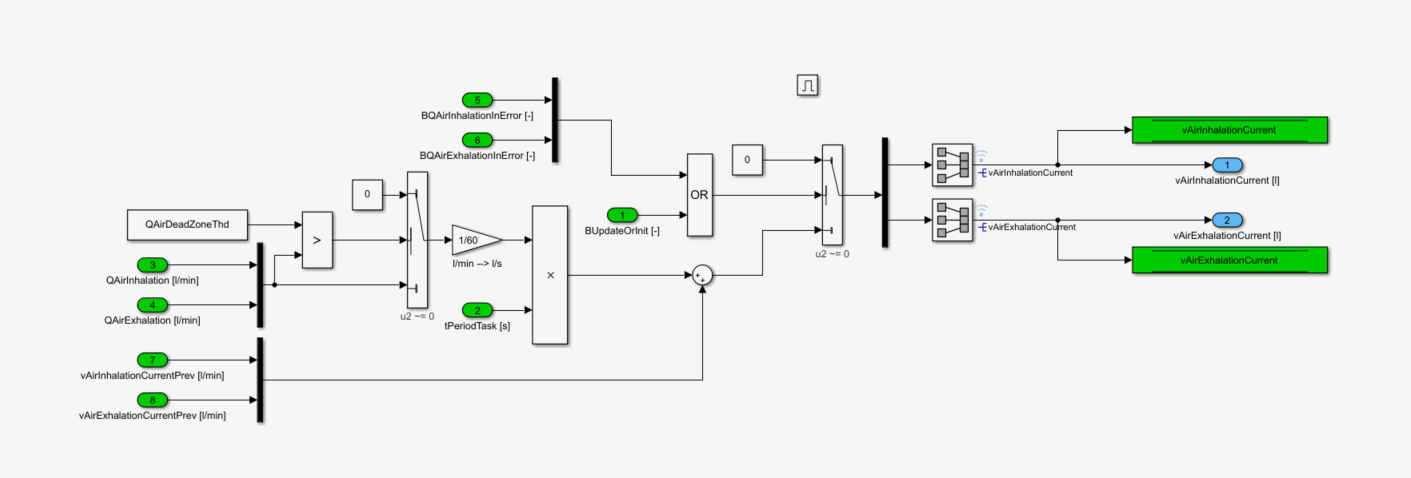
* The average of both volume if no error
* The inhalation volume if an error occurs on the exhalation flowmeter
* The exhalation volume if an error occurs on the inhalation flowmeter

In case of error, at init and at each cycle start, the volume is set to 0. A dead zone is implemented in order to remove the impact of the noise & of a small offset on the differential pressure sensor.

*NucleoRespirator/Control/Air Volume Processing*



*NucleoRespirator/Control/Air Volume Processing/Volume Computation/Peak Monitoring*



*NucleoRespirator/Control/Air Volume Processing/Volume\_Flow\_Release /Release*

**

## Warning

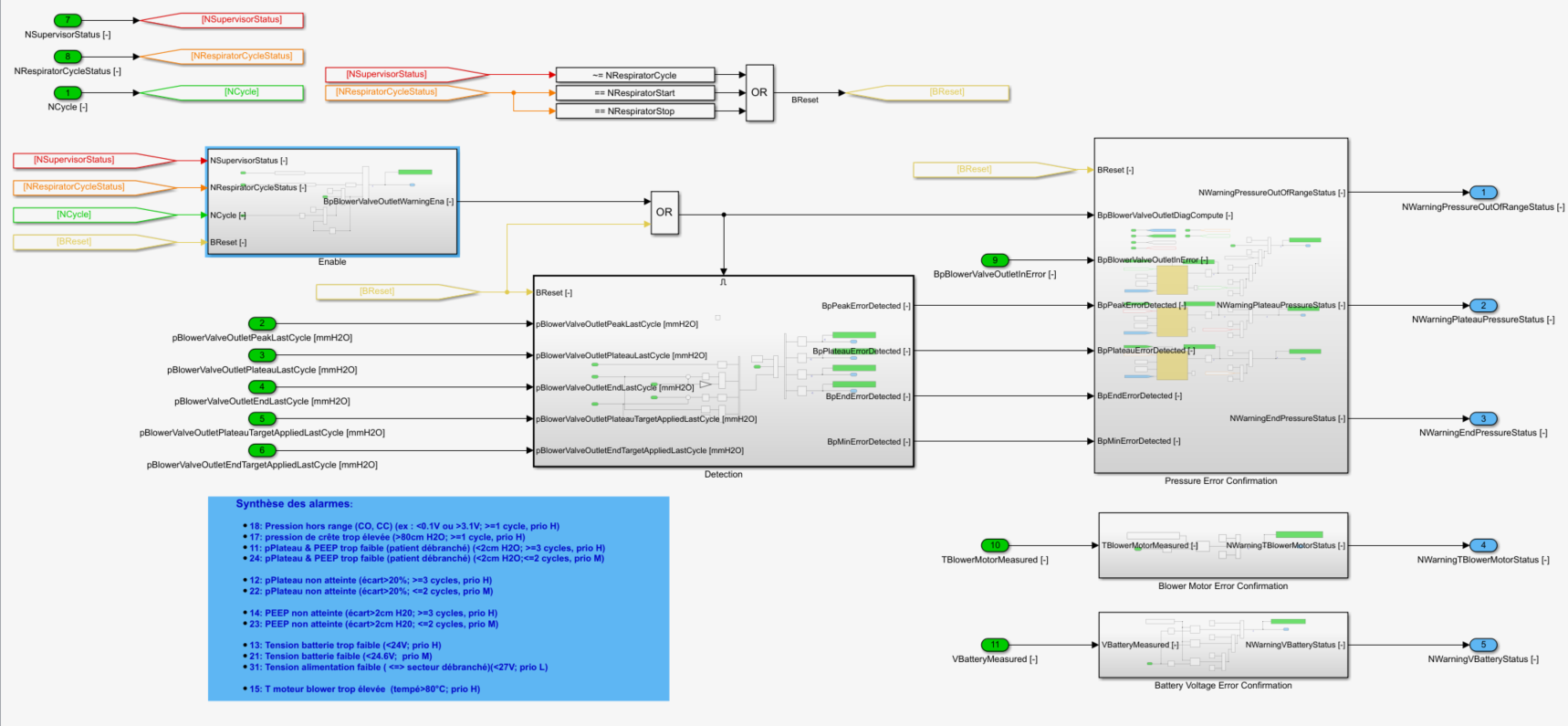
The purpose of this module is to monitor the main indicators of the respirator and inform the user in case of issues. 5 alarms can be displayed on the screen. Thus, the different failures were sorted out in 5 categories. Please find more detail below:

|  |  |  |  |
| --- | --- | --- | --- |
| **Alarm** | **Description** | **Amount of cycles** | **Prio** |
| 18 | Pressure out of range (SCB or SCG) | >=1 | H |
| 17 | Peak pressure too high (>80 cmH2O) | >=1 | H |
| 11 | Plateau & end-expiratory pressure too low (<2cmH2O) | >=3 | H |
| 24 | Plateau & end-expiratory pressure too low (<2cmH2O) | <=2 | M |
| 12 | Plateau pressure not reached (error >20%) | >=3 | H |
| 22 | Plateau pressure not reached (error >20%) | <=2 | M |
| 14 | End-expiratory pressure not reached (error >2cmH20) | >=3 | H |
| 23 | End-expiratory pressure not reached (error >2cmH20) | <=2 | M |
| 13 | Battery voltage too low (<24V) |  | H |
| 21 | Battery voltage too low (<24.6V) |  | M |
| 31 | Battery voltage too low (<27V⬄ respirator unplugged) |  | L |
| 15 | Blower temperature too high (>80°C) |  | H |

This function is composed of 5 sub-modules:

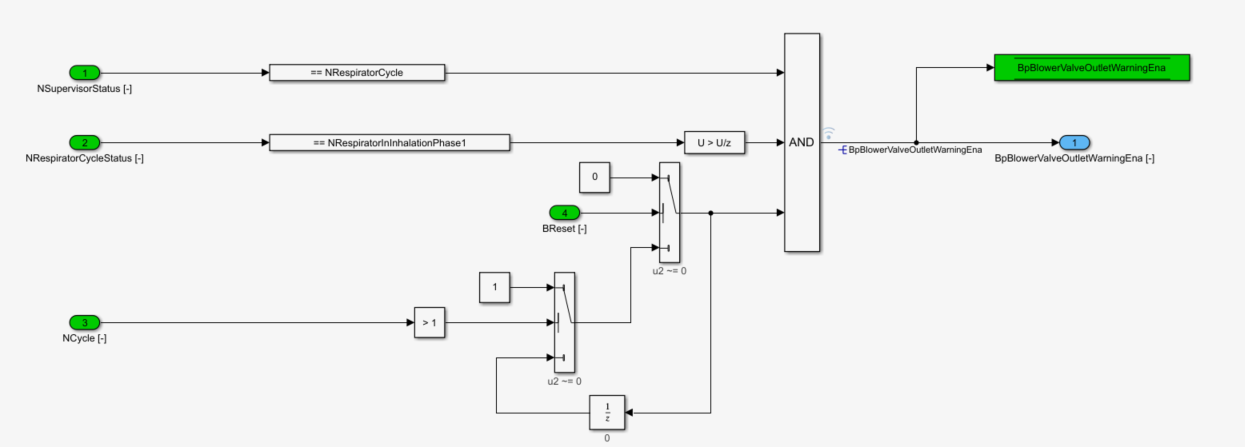
* Enable
* Error Detection
* Pressure error confirmation
* Blower temperature error confirmation
* Battery voltage error confirmation

*NucleoRespirator/Control/Warning*



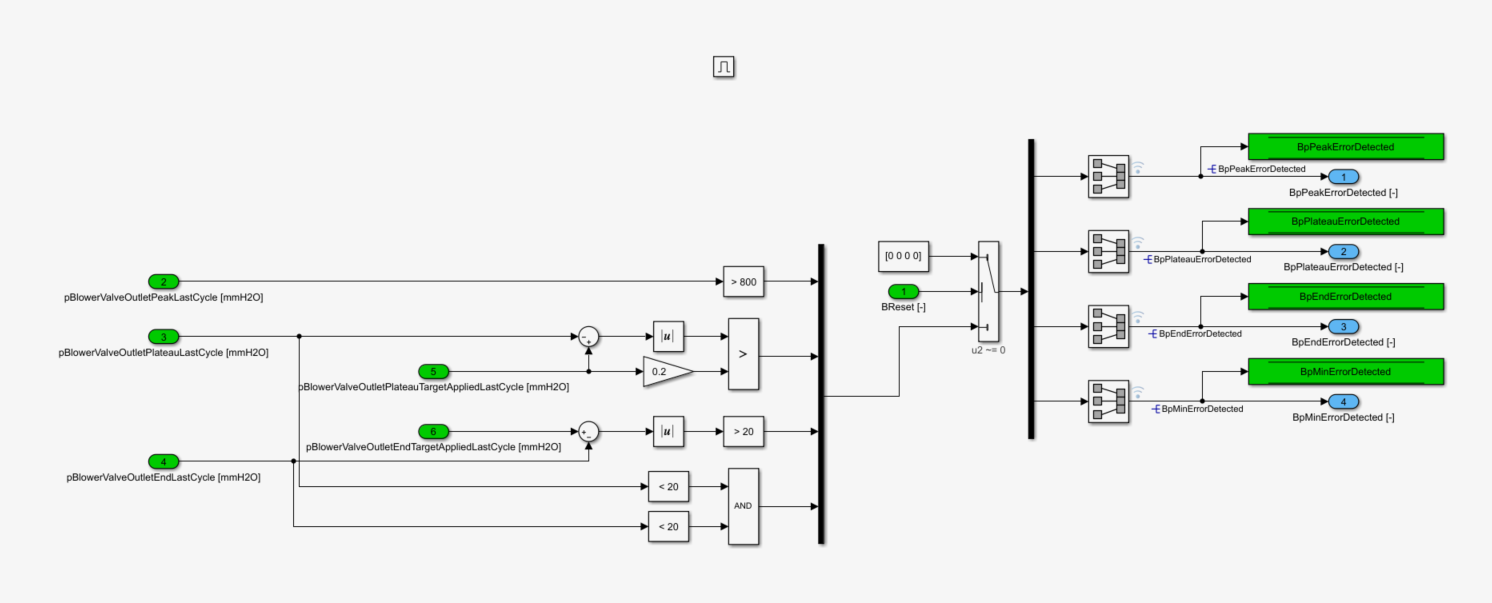
The error detection is enabled at each cycle start as long as the cycle mode is active. At the start of the first cycle, the detection is disabled because there is no valid previous cycle.

*NucleoRespirator/Control/Warning/Enable*



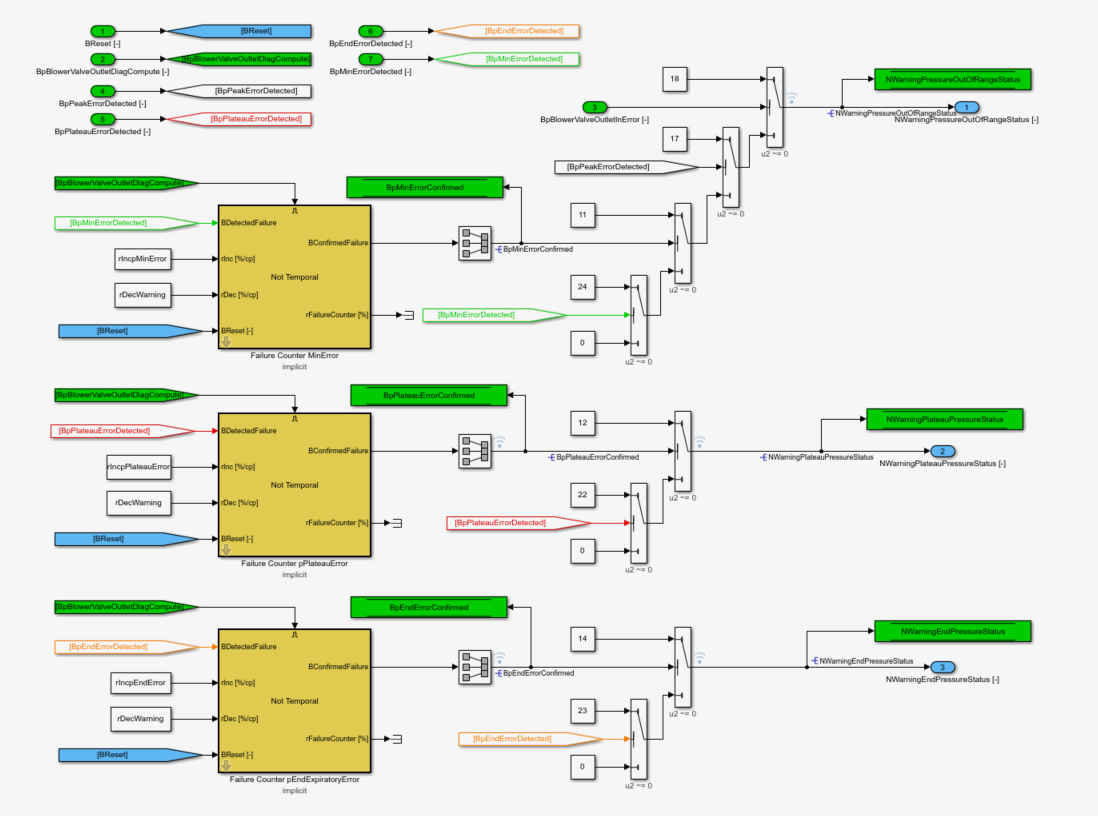
The detection of pressure errors is implemented here after based on the specification.

*NucleoRespirator/Control/Warning/Detection*



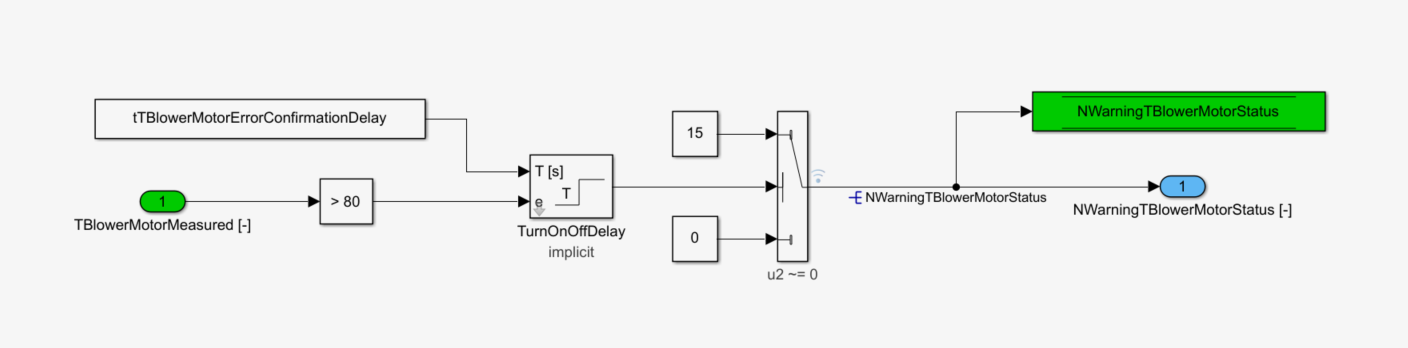
Failure counters are implemented in order to confirm the errors. In case of confirmation, the associated alarm rises. In case of multiple failures, the feature provides the alarm index with the highest priority.

*NucleoRespirator/Control/Warning/Pressure Error Confirmation*



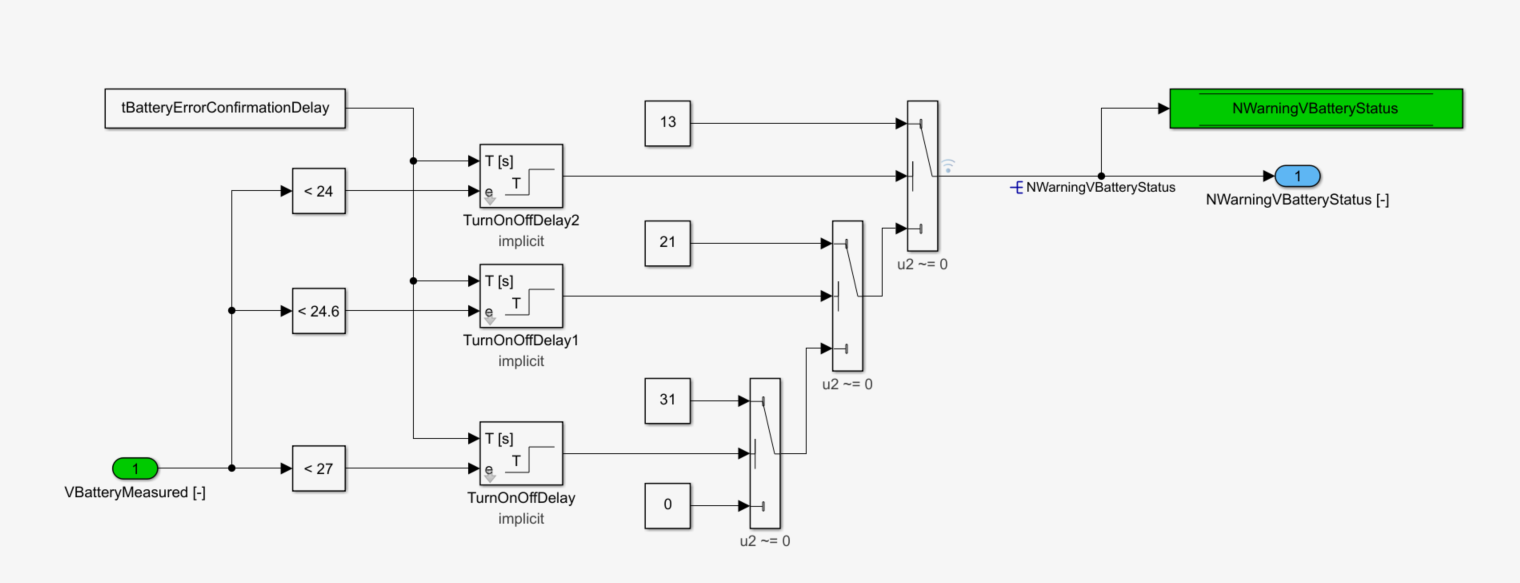
A temperature failure is confirmed after tTBlowerMotorErrorConfirmationDelay.

*NucleoRespirator/Control/Warning/Blower Motor Error Confirmation*



A voltage failure is confirmed after tBatteryErrorConfirmationDelay. In case of multiple failures, the feature provides the alarm index with the highest priority.

*NucleoRespirator/Control/Warning/Battery Voltage Error Confirmation*



## Output processing

Output processing block computes actuators target set by the respirator control and drive actuators through the microcontroller outputs.

Control targets may be overridden using the actuator test mode. In that mode, control targets are replaced by user manual targets.

### Blower motor speed control

Blower PWM output block uses two configuration parameters:

* Physical value min: Motor speed for 0% PWM
* Physical value max: Motor speed for 100% PWM

Based on those two parameters blower motor speed target is linearly converted into a PWM command. If the PWM ratio is smaller than the PWM min value, PWM output is set to 0%.

### Blower and patient valve control

For blower and patient valves, control output is a valve opening percentage. When control outputs 0% it means 0% opened (fully closed), when control outputs 100% it means 100% opened (fully open).

The control opening target is first converted into an angular position, then the angular position is converted into a PWM ratio which is used to drive the valve stepper motor.

* The valve driver block uses 4 parameters:
* Fully closed angle: Valve angular position for which the valve is fully closed
* Fully open angel: Valve angular position for which the valve is fully open
* Zero-degree PWM: PWM ratio giving a stepper motor position of 0°
* 100% PMW position: Stepper motor position for 100% PWM

Based on those for parameters, 2 linear conversions are computed. The first one is used to convert valve opening target into an angular position. The second one to convert angular position into a PWM ratio.

Final PWM command is saturated between 0 and 100%

### Actuator test mode

Actuator test mode is active when calibration parameter “NActuatorActive” is not equal to “None”. This parameter takes four values:

* None: Actuator test mode is disabled, and all applied actuator targets come from the control
* BlowerMotor: Actuator test mode is active, and user can manually drive blower motor speed.
* BlowerValve: Actuator test mode is active, and user can manually drive blower valve position
* PatientValve: Actuator test mode is active, and user can manually drive patient valve position

Target for the actuator tested is set using calibration parameter “rActuatorCmdMaxRatio” which define the ratio of maximum target to be applied.

For example, let’s say that blower motor speed range is 0 to 55 000 rpm.

If you set 0 in “rActuatorCmdMaxRatio”, test target will be 0 rpm (55 000 x 0).

If you set 0.5 in “rActuatorCmdMaxRatio”, test target will be 22 500 rpm (55 000 x 0.5).

If you set 1 in “rActuatorCmdMaxRatio”, test target will be 55 000 rpm (55 000 x 1).

When test mode is active, untested actuators targets are taking default values.

## LCD display

LCD screen updating function is running with at a sample time of 200 milliseconds (while control is running at a sample time of 10 milliseconds).

This function basically outputs four strings of characters which are all 20 characters long.

Those four strings are passed to LCD screen driver (MBED Text LCD [https://os.mbed.com/users/simon/code/TextLCD//file/308d188a2d3a/TextLCD.cpp/](https://os.mbed.com/users/simon/code/TextLCD/file/308d188a2d3a/TextLCD.cpp/)) which handle all the tricky magic of LCD screen hardware interface.

There are three main display modes:

* Initialization at the respirator start up showing the current software version of the respirator.
* Waiting when respirator is powered up but not doing anything. In this mode current respiration setpoints are showed
* Cycle mode when respiration cycles are running. In this mode cycle setpoints are displayed as well as different system measurements and alarms if any.